UNIVERSIDADE FEDERAL DE GOIÁS (UFG)

INSTITUTO DE INFORMÁTICA

PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

CARLOS DANIEL DE SOUSA BEZERRA

Arquitetura Modular para Navegação Autônoma com Aceleração de Aprendizagem, Fusão Sensorial e Comunicação Colaborativa

Goiânia
2025

UNIVERSIDADE FEDERAL DE GOIÁS
INSTITUTO DE INFORMÁTICA

TERMO DE CIÊNCIA E DE AUTORIZAÇÃO (TECA) PARA DISPONIBILIZAR VERSÕES ELETRÔNICAS DE TESES

E DISSERTAÇÕES NA BIBLIOTECA DIGITAL DA UFG

Na qualidade de titular dos direitos de autor, autorizo a Universidade Federal de Goiás (UFG) a disponibilizar, gratuitamente, por meio da Biblioteca Digital de Teses e Dissertações (BDTD/UFG), regulamentada pela Resolução CEPEC nº 832/2007, sem ressarcimento dos direitos autorais, de acordo com a Lei 9.610/98, o documento conforme permissões assinaladas abaixo, para fins de leitura, impressão e/ou download, a título de divulgação da produção científica brasileira, a partir desta data.

O conteúdo das Teses e Dissertações disponibilizado na BDTD/UFG é de responsabilidade exclusiva do autor. Ao encaminhar o produto final, o autor(a) e o(a) orientador(a) firmam o compromisso de que o trabalho não contém nenhuma violação de quaisquer direitos autorais ou outro direito de terceiros.

1. Identificação do material bibliográfico

[ ] Dissertação [X] Tese [ ] Outro*:_____

*No caso de mestrado/doutorado profissional, indique o formato do Trabalho de Conclusão de Curso, permitido no documento de área, correspondente ao programa de pós-graduação, orientado pela legislação vigente da CAPES.

Exemplos: Estudo de caso ou Revisão sistemática ou outros formatos.

2. Nome completo do autor

Carlos Daniel de Sousa Bezerra

3. Título do trabalho

Arquitetura Modular para Navegação Autônoma com Aceleração de Aprendizagem, Fusão Sensorial e Comunicação Colaborativa

4. Informações de acesso ao documento (este campo deve ser preenchido pelo orientador)

Concorda com a liberação total do documento [X] SIM [ ] NÃO[1]

[1] Neste caso o documento será embargado por até um ano a partir da data de defesa. Após esse período, a possível disponibilização ocorrerá apenas mediante:
a) consulta ao(à) autor(a) e ao(à) orientador(a);
b) novo Termo de Ciência e de Autorização (TECA) assinado e inserido no arquivo da tese ou dissertação.
O documento não será disponibilizado durante o período de embargo.
Casos de embargo:
- Solicitação de registro de patente;
- Submissão de artigo em revista científica;
- Publicação como capítulo de livro;
- Publicação da dissertação/tese em livro.

Obs. Este termo deverá ser assinado no SEI pelo orientador e pelo autor.

Documento assinado eletronicamente por **Carlos Daniel De Sousa Bezerra**, **Discente**, em 10/11/2025, às 15:21, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

Documento assinado eletronicamente por **Flavio Henrique Teles Vieira**, **Professor do Magistério Superior**, em 10/11/2025, às 15:30, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020 .

A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **5743470** e o código CRC **FE5D2AF9**.

Referência: Processo nº 23070.047861/2025-90 SEI nº 5743470

CARLOS DANIEL DE SOUSA BEZERRA

Arquitetura Modular para Navegação Autônoma com Aceleração de Aprendizagem, Fusão Sensorial e Comunicação Colaborativa

Tese apresentada ao Programa de Pós–Graduação em Ciência da Computação, do Instituto de Informática da Universidade Federal de Goiás, como requisito parcial para obtenção do título de Doutor em Ciência da Computação.

Área de concentração: Ciência da Computação.

Orientador: Prof. Dr. Flávio Henrique Teles Vieira

Goiânia
2025

Ficha de identificação da obra elaborada pelo autor, através do
Programa de Geração Automática do Sistema de Bibliotecas da UFG.

Bezerra, Carlos Daniel de Sousa
Arquitetura Modular para Navegação Autônoma com Aceleração de Aprendizagem, Fusão Sensorial e Comunicação Colaborativa [manuscrito] / Carlos Daniel de Sousa Bezerra. - 2025.
CLXXV, 175 f.: il.

Orientador: Prof. Dr. Flávio Henrique Teles Vieira.
Tese (Doutorado) - Universidade Federal de Goiás, Instituto de Informática (INF), Programa de Pós-Graduação em Ciência da Computação, Goiânia, 2025.
Bibliografia.
Inclui gráfico, tabelas, algoritmos, lista de figuras, lista de tabelas.

1. Navegação Autônoma. 2. DQN. 3. Filtro de Kalman. 4. LoRa. 5. Fusão de Sensores. I. Vieira, Flávio Henrique Teles , orient. II. Título.

CDU 004

UNIVERSIDADE FEDERAL DE GOIÁS

INSTITUTO DE INFORMÁTICA

ATA DE DEFESA DE TESE

Ata nº **24** da sessão de Defesa de Tese de **Carlos Daniel de Sousa Bezerra**, que confere o título de Doutor em Ciência da Computação, na área de concentração em Ciência da Computação.

Aos oito dias do mês de outubro de dois mil e vinte e cinco, a partir das catorze horas, via sistema de webconferência, realizou-se a sessão pública de Defesa de Tese intitulada **"Arquitetura Modular para Navegação Autônoma com Aceleração de Aprendizagem, Fusão Sensorial e Comunicação Colaborativa"**. Os trabalhos foram instalados pelo Orientador, Professor Doutor Flávio Henrique Teles Vieira (EMC/UFG) com a participação dos demais membros da Banca Examinadora: Professor Doutor Álisson Assis Cardoso (EMC/UFG), membro titular externo; Professor Doutor Carlos Galvão Pinheiro Junior (EMC/UFG), membro titular externo; Professor Doutor João Paulo da Silva Fonseca (EMC/UFG), membro titular externo; e Professor Doutor Leonardo da Cunha Brito (EMC/UFG), membro titular externo. A realização da banca ocorreu por meio de videoconferência. Durante a arguição os membros da banca não fizeram sugestão de alteração do título do trabalho. A Banca Examinadora reuniu-se em sessão secreta a fim de concluir o julgamento da Tese, tendo sido o candidato **aprovado** pelos seus membros. Proclamados os resultados pelo Professor Doutor Flávio Henrique Teles Vieira, Presidente da Banca Examinadora, foram encerrados os trabalhos e, para constar, lavrou-se a presente ata que é assinada pelos Membros da Banca Examinadora, aos oito dias do mês de outubro de dois mil e vinte e cinco.

TÍTULO SUGERIDO PELA BANCA

Documento assinado eletronicamente por **Leonardo Da Cunha Brito**, **Professor do Magistério Superior**, em 08/10/2025, às 17:18, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020 .

Documento assinado eletronicamente por **Flavio Henrique Teles Vieira**, **Professor do Magistério Superior**, em 08/10/2025, às 17:18, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020 .

Documento assinado eletronicamente por **Carlos Galvao Pinheiro Junior**, **Professor do Magistério Superior**, em 08/10/2025, às 17:18, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020 .

Documento assinado eletronicamente por **Alisson Assis Cardoso**, **Professor do Magistério Superior**, em 08/10/2025, às 17:18, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020 .

Documento assinado eletronicamente por **Joao Paulo Da Silva Fonseca**, **Professor do Magistério Superior**, em 08/10/2025, às 17:18, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020 .

Documento assinado eletronicamente por **Carlos Daniel De Sousa Bezerra**, **Discente**, em 08/10/2025, às 17:28, conforme horário oficial de Brasília, com fundamento no § 3º do art. 4º do Decreto nº 10.543, de 13 de novembro de 2020.

A autenticidade deste documento pode ser conferida no site https://sei.ufg.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0, informando o código verificador **5687624** e o código CRC **E16E2B9A**.

Referência: Processo nº 23070.047861/2025-90 SEI nº 5687624

CARLOS DANIEL DE SOUSA BEZERRA

Arquitetura Modular para Navegação Autônoma com Aceleração de Aprendizagem, Fusão Sensorial e Comunicação Colaborativa

Tese defendida no Programa de Pós–Graduação em Ciência da Computação, do Instituto de Informática da Universidade Federal de Goiás como requisito parcial para obtenção do título de Doutor em Ciência da Computação, aprovada em 08 de Outubro de 2025, pela Banca Examinadora constituída pelos professores:

Prof. Dr. Flávio Henrique Teles Vieira
Instituto de Informática – UFG
Escola de Engenharia Elétrica, Mecânica e de Computação - UFG
Presidente da Banca

Prof. Dr. Álisson Assis Cardoso
Escola de Engenharia Elétrica, Mecânica e de Computação – UFG

Prof. Dr. Carlos Galvão Pinheiro Júnior
Escola de Engenharia Elétrica, Mecânica e de Computação – UFG

Prof. Dr. João Paulo da Silva Fonseca
Escola de Engenharia Elétrica, Mecânica e de Computação – UFG

Prof. Dr. Leonardo da Cunha Brito
Escola de Engenharia Elétrica, Mecânica e de Computação – UFG

Carlos Daniel de Sousa Bezerra

Graduado em Engenharia Elétrica pela Pontifícia Universidade Católica de Goiás (2016). Mestre em Engenharia Elétrica e de Computação pela Universidade Federal de Goias (2019). Durante seu mestrado, atuou na área de otimização e controle de potência em termogeradores utilizando inteligência artificial. Atualmente dedica seus estudos nas áreas de Navegação Autônoma, Comunicação Robótica e Inteligência Artificial. É docente efetivo no Instituto Federal de Ciência e Tecnologia de Goiás (IFG) atuando em diversas disciplinas do curso de Engenharia de Elétrica e Telecomunicações.

Primeiramente a Deus toda honra e toda glória. Dedico este trabalho aos meus pais, Carlos Alberto Bezerra e Bernadete Coelho de Sousa.

Agradecimentos

Ao refletir sobre a jornada que foi a elaboração deste trabalho, sinto uma profunda gratidão por todos aqueles que contribuíram de maneira significativa para a sua realização. Primeiramente, agradeço a Deus por me proporcionar saúde e disposição para seguir em frente mesmo diante dos desafios impostos pela pandemia da COVID-19.

Um agradecimento especial aos meus pais, Carlos Alberto Bezerra e Bernadete Coelho, ambos professores, que me inspiraram com seu exemplo e me apoiaram incondicionalmente ao longo de minha vida profissional e acadêmica. Sem o amor e o incentivo deles, este trabalho não teria sido possível.

Expresso minha sincera gratidão ao meu orientador, professor Flávio Henrique Teles Vieira, pelas valiosas orientações e pela oportunidade de envolver-me em projetos de pesquisa que tanto enriqueceram meu aprendizado. Sua paciência e estímulo foram essenciais para o desenvolvimento e conclusão deste trabalho.

Minha gratidão também se estende a toda minha Família. À minha namorada, Karla Alves, agradeço pelo seu suporte contínuo e compreensão sobre a importância deste trabalho para a minha carreira e vida pessoal.

Por fim, agradeço aos meus colegas do grupo de pesquisa InComm da Universidade Federal de Goiás. As discussões em nossas reuniões foram fundamentais para o aprimoramento das técnicas empregadas neste estudo. A colaboração de cada um de vocês foi um pilar para o sucesso deste projeto.

Na guerra, o vitorioso vence primeiro e depois vai à batalha, enquanto o derrotado entra primeiro na batalha e depois procura vencer.

Sun Tzu,
General e estrategista militar chinês, autor de "A Arte da Guerra".

Resumo

Bezerra, C.D.S. **Arquitetura Modular para Navegação Autônoma com Aceleração de Aprendizagem, Fusão Sensorial e Comunicação Colaborativa**. Goiânia, 2025. 173p. Tese de Doutorado Instituto de Informática, Universidade Federal de Goiás.

Este trabalho propõe uma arquitetura modular para a navegação autônoma de robôs móveis em ambientes dinâmicos, aliando os seguintes módulos funcionais: (i) controle inteligente por aprendizado por reforço profundo, (ii) fusão de sensores visuais e não visuais, (iii) módulo de segurança anti-colisão baseado em visão computacional, e (iv) comunicação híbrida colaborativa. O algoritmo EKF-DQN proposto acelera o processo de aprendizagem ao integrar predições de estado fornecidas pelo Filtro de Kalman Estendido. O sistema de fusão sensorial combina dados de sensores visuais e não visuais por meio de estratégias como *Late Fusion*, permitindo ao agente adquirir uma percepção robusta do ambiente. Para segurança operacional, foi implementado um módulo de detecção de pessoas que identifica risco de colisão. Na camada de comunicação, propõe-se protocolo híbrido que combina: (i) a tecnologia LoRa, de longo alcance e baixo consumo, utilizada para comunicação com estações remotas (R2I); e (ii) o middleware DDS, aplicado à comunicação de curto alcance entre robôs (R2R), garantindo sincronização e troca de informações sensoriais em tempo real. Com foco na robustez da conectividade LoRa, desenvolveu-se o algoritmo "Taxa Forçada de Dados", que força adaptativamente o fator de espalhamento espectral com base na qualidade do enlace, superando o desempenho do tradicional algoritmo *Adaptive Data Rate* (ADR) em cenários com mobilidade constante. Para viabilizar a transmissão de dados sensoriais de alta resolução, como os gerados por sensores LiDAR, foi empregada uma técnica de compressão baseada em autoencoders para transmissão LoRa, atingindo uma redução média de 82% no tamanho do payload, sem perda significativa de acurácia. Os resultados, obtidos tanto em simulações quanto em experimentos reais com robôs, demonstram melhorias expressivas em desempenho de navegação. O EKF-DQN proposto atingiu 93.33% de taxa de sucesso e foi superior ao tradicional D3QN. Com a inserção do módulo de segurança, foi alcançado 72% de taxa de sucesso com pessoas em movimento no ambiente.

Palavras–chave

Navegação Autônoma, DQN, Filtro de Kalman, LoRa, Fusão de Sensores, MARL, DDS.

Abstract

Bezerra, C.D.S. **Development of an Autonomous Robotic Navigation System with Accelerated Reinforcement Learning, Sensor Fusion and LoRa Communication**. Goiânia, 2025. 173p. PhD. Thesis Instituto de Informática, Universidade Federal de Goiás.

This work proposes a modular architecture for autonomous navigation of mobile robots in dynamic environments, integrating functional modules: (i) intelligent control through deep reinforcement learning, (ii) fusion of visual and non-visual sensors, (iii) a vision-based anti-collision safety module, and (iv) collaborative hybrid communication. The proposed EKF-DQN algorithm accelerates the learning process by integrating state predictions provided by the Extended Kalman Filter. The sensor fusion system combines data from visual and non-visual sensors through strategies such as Late Fusion, allowing the agent to acquire a robust perception of the environment. For operational safety, a person detection module was implemented to identify collision risks. In the communication layer, a hybrid protocol is proposed that combines: (i) LoRa technology, known for its long range and low power consumption, used for communication with remote stations (R2I); and (ii) the DDS middleware, used for short-range robot-to-robot (R2R) communication, ensuring real-time synchronization and sensory data exchange. To enhance the robustness of LoRa connectivity, a new algorithm called "Forced Data Rate" was developed, which adaptively enforces the spreading factor based on link quality, outperforming the traditional Adaptive Data Rate (ADR) algorithm in scenarios with constant mobility. To enable the transmission of high-resolution sensory data, such as those generated by LiDAR sensors, a compression technique based on autoencoders was used, achieving an average payload size reduction of 82% without significant loss of accuracy. Results obtained from both simulations and real-world robot experiments demonstrated significant improvements in navigation performance. The proposed EKF-DQN achieved a 93.33% success rate and outperformed the traditional D3QN. With the integration of the safety module, a 72% success rate was achieved in environments with moving people.

Keywords

Autonomous Navigation, DDQN, Kalman Filter, LoRa, Sensor Fusion, DDS, MARL

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Algoritmos

CAPÍTULO 1

Introdução

Atualmente o mundo passa por uma transformação digital conhecida como quarta revolução industrial e chamada também de Industria 4.0. O termo "transformação digital" significa mudanças relevantes que estão ocorrendo na sociedade, principalmente no setor de negócios, em função da rápida adoção de tecnologia. Os modelos de negócios existentes serão expandidos ou substituídos por serviços orientados a dados digitais [Schumann et al. 2017].

Neste novo cenário mundial, as grandes empresas travam uma disputa quanto à quantidade de investimentos em ferramentas digitais, tais como inteligência artificial, eletrônica embarcada, conectividade e sensoriamento para alavancar seus lucros e competitividade.

No trabalho de [Andreja 2017], é explicado o conceito de Indústria 4.0, originado na Alemanha, que se baseia em nove pilares fundamentais que revolucionam o setor industrial: i) análise de grandes volumes de dados (*Big Data*), ii) robótica autônoma, iii) simulação, iv) integração de sistemas, v) internet das coisas (*Internet of Things* - IoT), vi) cibersegurança, vii) computação em nuvem, viii) manufatura aditiva e ix) realidade aumentada. Uma característica central deste modelo de produção é o uso integrado desses pilares para monitorar processos em tempo real, reduzindo a necessidade de intervenção humana, otimizando tarefas produtivas e aumentando a segurança nos processos industriais.

Conforme discutido em [Andreja 2017], a implementação dos conceitos da Indústria 4.0 está diretamente associada aos Sistemas Ciber-Físicos (*Cyber-physical systems* - CPS) e à Internet das Coisas (*Internet of things* - IoT). Os CPS representam a integração entre os componentes físicos dos processos produtivos e o controle descentralizado, unindo *hardware* e conectividade avançada, possibilitada pela IoT. Esses sistemas são projetados para coletar, monitorar e trocar dados de processos industriais em tempo real. Um elemento crucial para o funcionamento eficiente desses sistemas é o gerenciamento de *Big Data* – grandes volumes de dados, geralmente armazenados na nuvem, cuja análise exige a atuação de especialistas para extrair informações que sustentem a definição de estratégias para otimizar a tomada de decisões.

A tecnologia da Indústria 4.0 tem habilitado as fábricas a incorporar Robôs Móveis Autônomos (*Autonomous Mobile Robots* - AMR) em suas linhas de montagem, categorizando-os como CPS conforme descrito por Javid [Mohd et al. 2021]. Ao contrário dos veículos operados manualmente, os AMRs operam independentemente do controle humano direto. Eles são equipados com sensores visuais e não visuais, além de possuírem arquiteturas de controle avançadas para navegação em ambientes dinâmicos. Os AMRs têm o potencial de substituir ou auxiliar a força de trabalho humana, especialmente em áreas de risco do processo produtivo, onde podem melhorar significativamente a segurança do trabalho e reduzir a ocorrência de acidentes. Além disso, a mobilidade dos AMRs pode aumentar significativamente a eficiência, a velocidade e o volume de produção em determinados processos.

1.1 Trabalhos Relacionados

1.1.1 Indústria 4.0 e Sistemas Robóticos

Os autores em [Vermesan et al. 2020] explicam que na IoT, os dispositivos robóticos podem se comunicar com outros dispositivos, ou CPS, do ambiente fabril, aprender de forma autônoma e interagir com segurança com o ambiente e humanos. Além disso a tecnologia permite obter métricas para viabilizar a manutenção preditiva e prever comportamento de falha. As aplicações de IoT podem fazer uso da inteligência individual, colaborativa e coletiva de AMRs, bem como informações da infraestrutura no contexto operacional para planejar, implementar e realizar tarefas sob diferentes condições ambientais e incertezas. Os autores denominam a IoT destinada a aplicações de robótica como Internet das Coisas Robóticas (*Internet of Robotic Things* - IoRT).

Sistemas robóticos, especialmente os Sistemas de Múltiplos Robôs (*Multi-Robot Systems* - MRS), vêm ganhando destaque em aplicações práticas e de alto impacto, como na medicina, indústria, agricultura e defesa [Jawhar et al. 2018]. Os MRS ampliam significativamente as capacidades dos robôs modernos, permitindo a execução de tarefas mais complexas por meio de ações distribuídas e coordenadas. Para que essa coordenação ocorra de forma eficaz, a comunicação entre os robôs se torna um componente essencial, viabilizando o compartilhamento de informações em tempo real e a tomada de decisões [Jawhar et al. 2018].

1.1.2 Comunicação Robótica: LoRaWAN

Nesse contexto, a tecnologia LoRa (*Long Range*) e o protocolo LoRaWAN surgem como soluções promissoras para atender às necessidades de comunicação em sistemas distribuídos e aplicações IoT. LoRa é uma tecnologia de comunicação sem fio

que utiliza a modulação por espalhamento espectral (CSS - *Chirp Spread Spectrum*), permitindo comunicações de longo alcance e baixo consumo de energia, características ideais para cenários de baixa largura de banda e alta eficiência energética. O protocolo LoRaWAN, por sua vez, é uma camada de rede que opera sobre a tecnologia LoRa, responsável pelo gerenciamento de comunicação em larga escala, incluindo dispositivos e *gateways* em redes IoT.

Pesquisadores como [Haque et al. 2020] têm explorado o uso do protocolo LoRaWAN em cenários de veículos móveis, com foco na comunicação veicular *Vehicle-to-everything* (V2X). Nesse estudo, os autores propõem uma solução que utiliza comunicação direta *device-to-device* para reduzir a latência, conectando veículos diretamente (*Vehicle-to-Vehicle* - V2V) à infraestrutura (*Vehicle-to-Infrastructure* - V2I), evitando o roteamento de dados através do servidor LoRaWAN. Essa abordagem modular e de design compacto é ideal para sistemas legados, dispensando a necessidade de *hardware* adicional. Os resultados indicam que a arquitetura é adequada para aplicações em rodovias, com desempenho estável em velocidades de 15 a 50 km/h.

Embora existam semelhanças, o foco desta tese difere significativamente de [Haque et al. 2020] em vários aspectos. Neste trabalho, desenvolve-se uma rede de comunicação dedicada a dispositivos robóticos móveis, com uma infraestrutura inspirada em sistemas V2X, mas adaptada ao contexto de ambientes industriais. O protocolo LoRa é utilizado como estratégia R2I (*Robot-to-Infrastructure*), permitindo que os robôs se conectem a uma infraestrutura remota, garantindo comunicação robusta em cenários de longa distância. Apesar de não ser implementada a comunicação *device-to-device* diretamente via LoRa, este tipo de comunicação é explorado por meio de uma abordagem complementar de curto alcance (*short-range*), a qual será detalhada em seções posteriores deste trabalho.

Uma das vantagens significativas do protocolo de comunicação sem fio LoRaWAN é sua capacidade de adaptar parâmetros de comunicação para otimizar conexões de *uplink* e *downlink*, conforme discutido por Raza [Raza, Kulkarni e Sooriyabandara 2017]. Essa flexibilidade é devida à modulação na camada física, que é ajustável através do algoritmo de Taxa de Dados Adaptativa (*Adaptive Data Rate* - ADR) [Semtech 2019]. Esse algoritmo é projetado para otimizar os parâmetros da modulação digital utilizada. No entanto, existem potenciais melhorias e alternativas ao ADR que podem ser exploradas. De acordo com [Park, Lee e I.Joe 2020], técnicas de inteligência artificial aplicadas ao LoRa apresentam melhorias de até 15% em eficiência de transmissão de dados e consumo energético, comparadas ao ADR.

No contexto deste trabalho, avalia-se especificamente a performance do módulo de comunicação LoRa para robôs móveis, com o objetivo de analisar a eficácia do ADR e propor variações a este algoritmo. Diferentemente do estudo de

[Ghazali, Teoh e Rahiman 2021], que oferece poucos detalhes sobre implementações práticas, este trabalho detalha a implementação da rede LoRa, fornecendo detalhes técnicos que contribuem para um entendimento real, estabelecendo uma distinção clara em relação às pesquisas existentes.

O algoritmo ADR é utilizado de forma nativa em diversos módulos de comunicação LoRa, otimizando a comunicação através do ajuste da modulação digital com base nas condições do canal [Raza, Kulkarni e Sooriyabandara 2017]. No entanto, este trabalho propõe o algoritmo "Taxa Forçada de Dados", que supera o ADR ao forçar a utilização de diferentes fatores de espalhamento espectral (*Spreading Factors* - SFs), especialmente em cenários de movimentação contínua de robôs, aumentando a confiabilidade da transmissão de dados.

Apesar das vantagens do LoRa em termos de baixo consumo de energia e longo alcance, a tecnologia apresenta limitações significativas relacionadas ao tamanho do pacote de dados úteis (*payload*). O tamanho máximo do *payload* diminui consideravelmente com o aumento do SF, o que impacta diretamente a capacidade de transmissão de dados de sensores de alta demanda, como o LiDAR. Essas restrições tornam essencial a implementação de técnicas que permitam a transmissão eficiente em redes de banda estreita. Estudos como o de [Väänänen e Hämäläinen 2021] avaliam técnicas de compressão de dados, focando em algoritmos como *Delta Encoding* e *Run-Length Encoding* (RLE), que alcançaram taxas de compressão de até 53,86% para dados de sensores com resolução de 12 bits. Embora eficazes para sensores de baixo *payload*, essas abordagens são insuficientes para lidar com as demandas de sensores volumétricos, como o LiDAR (*Light Detection and Ranging*), sensor indispensável em tarefas de mapeamento e localização.

É evidente o potencial das redes LoRa para aplicações em cenários de mobilidade, sobretudo no contexto da comunicação com Robôs Móveis Autônomos (AMRs). Apesar disso, essa ainda é uma área incipiente, com escassez de documentos técnicos detalhados sobre implementações reais. Este trabalho busca justamente preencher essa lacuna, apresentando uma proposta metodológica sólida e detalhada para o uso da rede LoRa como meio de comunicação para AMRs. O potencial dos AMRs, aliás, vai muito além do ambiente fabril. No setor de *food delivery*, por exemplo, investimentos crescentes visam automatizar as entregas por meio de robôs autônomos conectados, utilizando redes como a LoRa para garantir a eficiência, a rastreabilidade e a segurança operacional [Cardona et al. 2020]. De forma semelhante, no contexto hospitalar durante a pandemia da COVID-19, AMRs foram empregados em ambientes com alto risco de contaminação, realizando tarefas como entrega de medicamentos, aferição de temperatura e descontaminação por radiação ultravioleta. Nessas situações, a confiabilidade da comunicação remota, especialmente em ambientes desafiadores, foi essencial, papel que pode ser assumido por tecnologias de longo alcance e baixo consumo como o LoRa

[Cardona et al. 2020].

1.1.3 Navegação e Controle Inteligente por Aprendizado por Reforço Profundo

Para além da comunicação, a capacidade de navegação autônoma destes AMRs, bem como a tomada de decisão, surgem a partir de três grandes áreas da robótica móvel: i) percepção, ii) planejamento e iii) atuação [Dudek e Jenkin 2010]. Os sensores presentes em um robô móvel autônomo são utilizados para perceber as condições de navegação. Já a área de planejamento se refere a um rol algoritmos embarcados nos processadores destes robôs e são usados para decidir qual caminho deve ser percorrido durante a execução da missão. Os atuadores são os responsáveis por acionar os elementos eletromecânicos que movimentam os AMRs, tais como motores das rodas, articulações entre outros.

De acordo com os autores em [Fayyad et al. 2020], uma consideração importante na área de percepção é a escolha adequada do tipo de sensor a ser empregado, bem como a sua configuração ideal. Esses sensores serão utilizados para imitar a capacidade humana de sentir e tomar decisões. Agrupar, ou fundir, sensores pode ser uma boa estrategia, aumentando assim a habilidade do agente se locomover de forma adequada dentro de um ambiente. As vantagens da área da ciência da computação em constante avanço, a Aprendizagem de Máquina (*Machine Learning*), motivaram suas aplicações em robótica e se tornou importante aliado na tomada de decisão de AMRs.

A Aprendizagem de Máquina, subcampo da Inteligência Artificial (IA), é uma importante área de estudo para aprendizado de sistemas autônomos. Esta ciência pode ser dividida em três categorias de problemas: i) Aprendizado Supervisionado, ii) Não supervisionado e iii) Aprendizado por Reforço (*Reinforcement Learning* - RL).

O RL é uma técnica de aprendizado de máquina especialmente eficaz para problemas de navegação de Robôs Móveis Autônomos (AMRs). Segundo [Krohn, Beyleveld e Bassens 2019], o RL é indicado para sistemas sequenciais de decisão, nos quais um agente, como um robô, interage com um ambiente (por exemplo, um pátio industrial), ocupando um determinado estado (sua posição atual) e executando ações que modificam esse estado. Após cada ação, o agente recebe um sinal de recompensa, que indica a qualidade daquela decisão no contexto do ambiente.

O objetivo do RL é fazer com que o agente aprenda uma política ótima que maximize a recompensa esperada ao longo do tempo, ou seja, para cada par estado-ação, o agente deve aprender qual decisão trará melhores resultados. Em essência, o agente aprende por tentativa e erro, buscando sempre maximizar as boas recompensas e evitar comportamentos que levem a punições [Geron 2019].

Uma das vantagens do RL é a possibilidade de navegar um veículo móvel sem pré planejamento de rota, sem conhecimento prévio de um mapa e sem intervenção manual, pois o agente aprende por iteração, assim como o aprendizado natural, sendo capaz de generalizar novas ações com uma possível mudança do ambiente.

Recentemente, os autores em [Srichandan, Dhingra e Hota 2021] publicaram uma pesquisa envolvendo o uso de Filtro de Kalman (KF) para melhorar o aprendizado por reforço aplicado ao controle de um robô de duas rodas, semelhante ao problema do pêndulo invertido. Os autores aplicam o KF para filtrar informações ruidosas nos dados sensoriais antes de serem usados como estado no algoritmo de aprendizado por reforço *Q-Learning* tradicional. Esta técnica mostrou-se capaz de melhorar a precisão dos estados medidos por um sensor, neste caso a IMU (Unidade de Medição Inercial), aumentando assim a estabilização das recompensas obtidas e a resposta transiente do sistema de controle. Portanto, a partir dos resultados apresentados em [Srichandan, Dhingra e Hota 2021], somos levados a investigar se, além de proporcionar maior estabilização, o KF pode acelerar uma técnica de aprendizado por reforço mais nova ou atual, como o *Deep Reinforcement Learning* (DRL).

Os autores em [Ahumada, Nettle e Solis 2013] apresentam um estudo sobre a aceleração do aprendizado por reforço usando o KF e foram pioneiros neste tipo de aplicação. O objetivo dos autores era treinar um goleiro para o futebol de robôs, no campeonato conhecido na comunidade como RoboCup. O método de aprendizado por reforço utilizado foi o *Q-Learning* tradicional, um método tabular [Sutton e Barto 2018]. A previsão de estados com o KF foi aplicada para estimar a posição da bola que o goleiro deveria interceptar, um passo à frente do estado atual. Os resultados mostram que o método proposto foi eficaz em acelerar o *Q-Learning* tradicional para a aplicação observada. No presente trabalho, a tarefa proposta é diferente, isto é, o objetivo está relacionado à navegação autônoma com um robô móvel diferencial.

1.1.4 Fusão de Sensores Visuais e Não Visuais

Segundo os autores em [Michael, Bodo e Vittorio 2019], técnicas de Aprendizado de Máquina podem ser usadas para fundir informações sensoriais. Vale destacar que a área de fusão sensorial (ou multimodal) não é somente aplicada em AMRs. Tanto a fusão de sensores quanto a fusão de informações podem ser definidas como o processo de gerenciamento e tratamento de dados provenientes de vários tipos de fontes para melhorar alguns critérios específicos para tarefas de decisão [Fayyad et al. 2020].

Atualmente a área médica, aliada com pesquisadores de computação e inteligência artificial, emprega o uso da fusão de dados para aumentar a precisão de diversos diagnósticos médicos que usam imagens (ressonância magnética, raio X entre outros) e

dados tabulares, tais como exames de sangue, gênero e sexo do paciente. Os autores em [Duanmu et al. 2020], pesquisadores da área médica e computação, desenvolveram um método de fusão de dados para classificação de pacientes com o diagnóstico de câncer de próstata a partir da integração de exames de imagem com exames complementares (dados tabulares). O método denominado *Interative-Model* apresentou melhorias relevantes quando comparado ao uso de imagens, 5% em termos de acurácia e 41% na métrica de avaliação F1 Score. Essa métrica expressa o equilíbrio entre dois fatores importantes: precisão (proporção de acertos entre as previsões positivas feitas pelo modelo) e revocação (proporção de acertos entre todos os casos realmente positivos).

Os autores [Fayyad et al. 2020] desenvolvem uma abrangente revisão de literatura sobre métodos de fusão de sensores em veículos autônomos, especificamente nas áreas de percepção, localização e mapeamento utilizando aprendizado de máquina. Os autores destacam aplicações recentes na literatura em tarefas de detecção de pedestres e estradas através da fusão do sensor LIDAR com imagens. Destacam também outras aplicações como posicionamento com sensores proprioceptivos, tais como o GPS, fundidos com imagens. Além disso os pesquisadores também apresentam a categorização dos métodos de fusão existentes na literatura. Um ponto que chama a atenção na revisão de literatura é a ausência de técnicas de aprendizado por reforço alidada a fusão de dados, as técnicas apresentadas são de aprendizado supervisionado. Em [Bednarek, Kicki e Krzysztof 2020], os autores explicam que a área de fusão multimodal aplicada a sistemas robóticos ainda é emergente.

Os autores em [Guo, Liu e Chen 2022] propõem técnica de fusão de dados aplicado ao aprendizado por reforço no reconhecimento de ações, ou gestos humanos. Nesta pesquisa são fundidas três imagens, com diferentes capacidades de representação e domínio, combinadas através do algoritmo de ajuste de pesos proposto através da técnica *Twin Delayed Deep Deterministic* (TD3), um método de RL. Os resultados indicam que o método de fusão aumentou a performance da classificação de gestos humanos em até 14%.

Um aspecto promissor a ser explorado é a aplicação dessa abordagem de fusão de dados diretamente em algoritmos de aprendizado por reforço voltados para controle de navegação. Em outras palavras, investigar como a fusão de múltiplas fontes de dados pode ser integrada de maneira nativa a arquiteturas de controle, a fim de melhorar a eficácia e a robustez do sistema em tarefas de navegação autônoma.

1.1.5 Comunicação Híbrida Colaborativa

No contexto da comunicação colaborativa, destaca-se o campo do *Multi-agent Reinforcement Learning* (MARL), ou Aprendizado por Reforço Multiagente

[Jiang, Su e Lu 2024], que estuda como múltiplos agentes podem aprender simultaneamente em um ambiente compartilhado. Cada agente busca otimizar seu comportamento com base em interações locais, enquanto a coordenação entre eles é essencial para atingir objetivos coletivos.

Grande parte dos estudos em MARL focam na colaboração via recompensas compartilhadas, mas poucos abordam explicitamente como os agentes podem trocar experiências completas (estado, ação, recompensa, próximo estado) em redes descentralizadas. Essa troca pode ser implícita ou explícita, sendo esta última mais promissora para acelerar o aprendizado coletivo.

Em [Yi et al. 2022], os autores propõem um método de comunicação com dois níveis de otimização: um nível superior, com compartilhamento de recompensas entre vizinhos, e um nível inferior, com políticas ajustadas localmente. Embora promova maior coordenação, a comunicação proposta limita-se à troca de recompensas, sem detalhar um protocolo estruturado de comunicação real.

Um levantamento abrangente sobre MARL, como o dos autores [Huh e Mohapatra 2024], detalha que a comunicação é uma capacidade vital para a coordenação entre agentes, especialmente em ambientes com informações imperfeitas e observabilidade parcial. Este estudo classifica a comunicação em: infraestrutura, isso é, onde os agentes se comunicam, como em redes de comunicação direta entre vizinhos, representação (como a informação é processada) e aprendizado de comunicação (o conteúdo das mensagens e a política de comunicação). Mensagens explícitas podem incluir observações, ações, recompensas, crenças, objetivos e intenções. A comunicação explícita e estruturada é comumente utilizada na prática, com mecanismos de comunicação R2R empregados não apenas para a troca de informações, mas também para habilitar e automatizar tarefas cooperativas.

Especificamente para redes veiculares, [Althamary, Huang e Lin 2019] evidenciam o potencial do MARL para alocação de recursos, descongestionamento da rede principal (*data offloading*) e Comunicação Ultra-Confiável e de Baixa Latência (*Ultra-Reliable Low Latency Communication* - URLLC) em cenários V2X. O estudo mostra que a comunicação multiagente descentralizada, adaptada à alta mobilidade, pode melhorar significativamente a eficiência energética, reduzir latência e aumentar a confiabilidade, reforçando a relevância de protocolos colaborativos robustos para aplicações no mundo real.

Nesta tese, propõe-se o compartilhamento de experiências completas entre robôs vizinhos, utilizando o DDS (*Data Distribution Service*) como *middleware* de comunicação. A especificação do algoritmo de comunicação, aliada à implementação prática de rede distribuída, representa uma contribuição em relação às abordagens anteriores.

Para viabilizar essa troca de informações em tempo real e com alta confiabilidade

entre robôs, é fundamental o uso de uma infraestrutura de software robusta e modular. Nesse contexto, o ROS2 (*Robot Operating System 2*) [Macenski et al. 2022] surge como plataforma de referência, sendo o sistema operacional de código aberto mais atual voltado para aplicações robóticas.

Embora o ROS2 também ofereça *frameworks* robustos e amplamente testados para navegação autônoma, com suporte a mapeamento, localização e planejamento de trajetórias como o Nav2 (*Navigation2*) [Macenski et al. 2020], neste trabalho opta-se por desenvolver algoritmos próprios.

A motivação central deste trabalho está em investigar estratégias alternativas de controle e comunicação em ambientes modulares, priorizando flexibilidade, leveza computacional e integração direta com mecanismos de aprendizado por reforço e colaboração entre robôs. Diferentemente de abordagens tradicionais baseadas em stacks consolidadas, como o Nav2, optou-se por desenvolver uma solução original a fim de explorar novas possibilidades em navegação autônoma, especialmente no que diz respeito à comunicação em longas distâncias, uma capacidade ainda não plenamente suportada pelo ROS2. Assim, a experimentação de algoritmos próprios torna-se essencial para fomentar avanços significativos no campo, permitindo avaliar arquiteturas mais adaptáveis a cenários reais e conectados. Além de preencher uma lacuna relevante na literatura, esta tese estabelece fundamentos que podem futuramente ser integrados ao próprio ROS2, ampliando a aplicabilidade e a flexibilidade de ambas as abordagens em diferentes contextos operacionais.

1.2 Objetivos e Hipótese

O objetivo geral deste trabalho é desenvolver uma arquitetura de controle avançada para a navegação autônoma de robôs móveis, estruturada em três pilares principais: (i) robótica cognitiva, (ii) comunicação colaborativa e (iii) fusão sensorial aprimorada. A arquitetura proposta também incorpora um módulo de segurança anti-colisão, responsável por detectar obstáculos e pessoas em tempo real, além de um mecanismo de coordenação multiagente, voltado à cooperação entre robôs em vizinhança. Essa abordagem visa permitir que múltiplos agentes atuem de forma colaborativa e segura em ambientes compartilhados.

Para atingir esses objetivos, desenvolve-se três algoritmos especializados, cada um focado em um componente chave da arquitetura de controle: (i) o **Algoritmo de Fusão**, responsável por integrar dados de diversos sensores (visuais e não visuais), proporcionando uma percepção precisa e unificada do ambiente; (ii) o **Algoritmo de Aceleração**, que utiliza técnicas avançadas para otimizar o processo de aprendizado por reforço, permitindo que o robô adapte rapidamente seu comportamento em resposta a

mudanças no ambiente e novas tarefas; e (iii) o **Algoritmo de Comunicação**, que é dividido em atuação de longa distância e curta distância (robôs em vizinhança).

A principal justificativa para a elaboração deste trabalho é atender às necessidades crescentes de navegação autônoma de AMRs, especialmente diante do aumento exponencial de aplicações na Indústria 4.0. Embora o tema aborde áreas como aprendizado por reforço, fusão sensorial e comunicação LoRa, o trabalho é delimitado pela proposta de uma arquitetura integrada e modular de controle. Esta arquitetura se concentra em otimizar a navegação de robôs móveis em ambientes industriais dinâmicos e remotos, enfrentando de forma coesa os desafios de eficiência no aprendizado, comunicação em redes de baixa largura de banda e coordenação entre múltiplos agentes.

A metodologia adotada neste trabalho combina simulações computacionais com validações em testes reais de campo, permitindo avaliar o desempenho da arquitetura proposta por meio de métricas robustas, como taxa de sucesso e recompensa média, tanto em cenários controlados quanto em condições operacionais adversas. Na avaliação do módulo de comunicação, especificamente o *Long Range*, os testes foram desenvolvidos em parque municipal com presença de obstáculos físicos, árvores e edificações, o que permitiu testar a robustez da comunicação LoRa e a confiabilidade dos algoritmos em situações de mobilidade e interferência. Além disso, os módulos do sistema foram projetados para operar de forma independente, com integrações parciais realizadas via ROS2, permitindo validação conjunta entre componentes como o algoritmo de aceleração (EKF-DQN), fusão de sensores e comunicação via LoRa.

A hipótese central deste trabalho propõe que a arquitetura desenvolvida, que integra um módulo de controle baseado em *Deep Reinforcement Learning* (DRL), técnicas avançadas de fusão sensorial e mecanismos de comunicação inteligentes de curto e longo alcance, é capaz de tornar mais eficiente a navegação autônoma de robôs móveis em ambientes com alta variabilidade e baixa infraestrutura de rede. O uso do filtro de Kalman como ferramenta de previsão de estados melhora a convergência e a precisão do agente, enquanto o uso do DDS como meio de comunicação *short-range* permite o compartilhamento eficiente de dados sensoriais entre múltiplos robôs. Já para comunicação de longo alcance, a arquitetura aplica técnicas de compressão por autoencoder para viabilizar a transmissão de dados de sensores de alto volume, como LiDAR.

O mecanismo de comunicação baseado em LoRa desempenha um papel essencial para garantir uma arquitetura de controle com maior robustez em conectividade, mesmo em áreas remotas, possibilitando o envio de comandos e a recepção de telemetria em situações onde Wi-Fi e redes móveis não estão disponíveis. Embora esse módulo não interfira diretamente no processo de aprendizado, ele garante conectividade robusta, viabilizando aplicações reais em ambientes industriais, agrícolas ou áreas de difícil acesso.

Por fim, a integração desses elementos em uma única arquitetura modular,

validada em simulações e testes controlados em campo, evidencia o potencial da proposta para atender aos desafios da Indústria 4.0. A estrutura modular facilita a substituição ou aprimoramento de componentes, além de permitir adaptações a diferentes tipos de robôs e aplicações. Assim, este trabalho representa um passo importante na direção de sistemas robóticos mais eficientes, autônomos e conectados, mesmo em condições de conectividade restrita e ambientes não estruturados.

Contribuições do Trabalho

As principais contribuições deste trabalho são apresentadas a seguir. Ressalta-se que os itens **(2)**, **(3)** e **(5)** constituem contribuições inovativas, conforme evidenciado por artigos publicados em periódicos internacionais qualificados.

1. **Arquitetura integrada para navegação robótica**:
    - Proposta de um sistema avançado de navegação para robôs móveis, projetado para atender os requisitos da Indústria 4.0, como modularidade, conectividade robusta e descentralização, com foco em aplicações em áreas remotas, onde a conectividade tradicional não está disponível.
    - Embora alinhada ao estado da arte, esta arquitetura serve como base para as contribuições inovadoras subsequentes.
2. **(Contribuição inovativa) Sistema de controle baseado em aprendizado por reforço acelerado**:
    - Proposição de uma arquitetura de controle autônomo combinando *Deep Q-Network* (DQN) com previsões do Filtro de Kalman (EKF), acelerando o processo de aprendizagem.
    - O método proposto reduz etapas de exploração e melhora a eficiência da navegação em ambientes dinâmicos.
    - Resultados e validações foram publicados em periódico internacional [Bezerra, Vieira e Soares 2024].
3. **(Contribuição inovativa) Fusão de sensores para navegação aprimorada**:
    - Desenvolvimento de uma estratégia de fusão entre sensores visuais e não visuais.
    - A integração da percepção com aprendizagem reforçada contribui para decisões mais estáveis durante a navegação.
    - A abordagem foi reportada em [Bezerra, Vieira e Carneiro 2023].
4. **Módulo de controle de colisão baseado em detecção de pessoas**:

- Implementação de um sistema de segurança anti-colisão para garantir operações seguras, mesmo em cenários de alta complexidade ou ambientes com múltiplos obstáculos.

5. **(Contribuição inovativa) Adaptação da tecnologia LoRa para cenários robóticos**:
   - Proposição de uma estratégia original de controle do *Spreading Factor* (SF) para robôs móveis, superando limitações do ADR em cenários dinâmicos.
   - Desenvolvimento de um mecanismo de compressão via autoencoders para sensores de alto payload (ex.: LiDAR), permitindo transmissão eficiente em redes LoRa.
   - Esta contribuição é inédita na literatura e foi publicada em periódico internacional [Bezerra, Cardoso e Vieira 2025].
6. **Comunicação cooperativa multiagente via DDS**:
   - Integração do middleware DDS (*Data Distribution Service*) para comunicação local de baixa latência entre robôs.
   - Estruturação de uma abordagem colaborativa multiagente para compartilhamento de percepções e coordenação de decisões.

Resumo das contribuições inovativas:

- Controle híbrido DQN + EKF para aceleração do aprendizado (Item 2)
- Fusão sensorial para navegação robusta (Item 3)
- Compressão via autoencoders e adaptação do SF para LoRa em robótica móvel (Item 5)

Essas contribuições foram validadas experimentalmente e apresentadas em periódicos internacionais qualificados, incluindo *IEEE Internet of Things Journal* (A1), *Applied Sciences* (A3) e *Emerging Telecommunications Technologies* (A3).

A próxima seção descreve os **artigos, publicações e contribuições** geradas a partir desta tese de doutorado.

1.3 Artigos e Publicações

Este trabalho não apenas avança o campo da navegação autônoma de robôs móveis, mas também já deixou uma marca significativa no cenário acadêmico internacional. As pesquisas realizadas e os resultados obtidos contribuíram para a produção de três artigos que foram publicados em revistas internacionais indexadas, duas classificadas como

Qualis A3 e uma como **Qualis A1**, reconhecendo sua relevância e impacto na comunidade científica. Ademais, os achados deste estudo foram apresentados em prestigiados congressos internacionais, incluindo o **LARS** (*Latin American Robotics Symposium*), o maior evento de robótica da América Latina, indexado pelo **IEEE** (*Institute of Electrical and Electronic Engineers*).

Além das contribuições acadêmicas, o reconhecimento da qualidade e inovação deste trabalho foi evidenciado através de prêmios recebidos em eventos locais, como a Escola Regional de Engenharia. O autor também mantém seu perfil atualizado na plataforma ORCID: https://orcid.org/0000-0002-0628-5276. Ao todo foram sete publicações, destacadas na Tabela 1.1 abaixo:

Tabela 1.1: Produções acadêmicas relacionadas à tese

Nº	**Referência**	**Tipo**	**Qualis**	SCOPUS (CiteScore %)
1	**BEZERRA, C. D. S.**, CARDOSO, A. A., VIEIRA, F. H. T. *Utilizing Autoencoders for Latent Representation and Efficient Transmission of LiDAR Data via LoRa in ROS. IEEE IoT Journal*, 2025. DOI: 10.1109/JIOT.2025.3530241	Periódico	A1	Q1 (75–100%)
2	**BEZERRA, C. D. S.**, CARNEIRO, D. P. Q., VIEIRA, F. H. T. *Autonomous Robotic Navigation Approach Using Deep Q-Network Late Fusion and People Detection-Based Collision Avoidance. Applied Sciences*, 2023. DOI: 10.3390/app132212350	Periódico	A3	Q2 (50–75%)
3	**BEZERRA, C. D.**, SOARES, A., VIEIRA, F. H. T. *Deep-Q-Network hybridization with extended Kalman filter for accelerate learning in autonomous navigation with auxiliary security module. Emerging Telecommunications Technologies (Wiley)*, 2024. DOI: 10.1002/ett.4946	Periódico	A3	Q2 (50–75%)
4	**BEZERRA, C. D.**, OLIVEIRA JR, A., VIEIRA, F. H. T. *Controle de Espalhamento com DRL para LoRa/LoRaWAN. ERIGO*, 2021. DOI: https://doi.org/10.5753/erigo.2021.18433	Congresso	B4	—
5	**BEZERRA, C. D.**, CARDOSO, I. H. L., VIEIRA, F. H. T. *DRL with CNN for Navigation using Virtual Training. LARS/IEEE*, 2023. DOI: https://doi.org/10.1109/LARS/SBR/WRE59448.2023.10332922	Congresso	B2	—
6	**BEZERRA, C. D.**, VIEIRA, F. H. T. *Aprendizado Profundo para Robôs com Treinamento Virtual. CBIC*, 2023. DOI: http://dx.doi.org/10.21528/CBIC2023-112	Congresso	B4	—
7	**BEZERRA, C. D. S.** et al. *Otimização da Área de Cobertura LoRaWAN para Comunicação Robótica R2X. SBPO*, 2024. DOI: 10.59254/sbpo-2024-193404	Congresso	B1	—

**Qualis referente à área de Ciência da Computação.*

Observações:

- Os dados de Qualis seguem a classificação da CAPES para a área de Ciência da Computação (Quadriênio 2017–2020).
- O artigo nº 4 foi premiado como melhor artigo completo na ERI-GO 2021.
- As métricas de citação e visualização foram atualizadas em novembro de 2025 a partir das plataformas IEEE, MDPI, Wiley, CBIC, SBPO e Google Scholar.

O trabalho está organizado da seguinte forma: O Capitulo 2 apresenta as tecnologias que possibilitam a implementação de sistemas de navegação autônoma, como sensores, microprocessadores e modelo de sistemas. O Capítulo 3 apresenta as principais ferramentas de inteligência artificial e visão de máquina utilizadas na proposta. Já o Capítulo 4 fundamenta conceitos sobre comunicação robótica com as coisas (*Robot to Everything* - R2X) e comunicação sem fio. O Capítulo 5 dispõe a metodologia de implementação do sistema de navegação proposto. Os resultados da proposta são apresentados no Capítulo 6. Por fim, o Capítulo 7 apresenta as conclusões, bem como trabalhos futuros.

CAPÍTULO 2

Tecnologias Habilitadoras Para Navegação Autônoma

Este capítulo aborda conceitos técnicos essenciais para a navegação autônoma de robôs, detalhando os tipos de sensores utilizados para percepção, localização e mapeamento. Além disso, discute-se a aplicação de algoritmos fundamentais no processo de navegação, como o Filtro de Kalman e técnicas de Fusão Sensorial. O capítulo também explora diferentes modelos de microprocessadores e sistemas operacionais específicos para robótica. Adicionalmente, apresenta-se o modelo cinemático diferencial do robô utilizado neste estudo, fornecendo uma base para compreender como o robô calcula e ajusta seus movimentos no ambiente.

2.1 Navegação de Sistemas Móveis Autônomos

A área de navegação autônoma é um dos grandes desafios atuais da engenharia e computação. A sua evolução técnica, precisa e segura, poderá levar a diversos impactos em aplicações e serviços em nosso dia a dia, seja com veículos automotores autônomos em ambientes urbanos, robótica autônoma, veículos aéreos não tripulados, entre outras.

Um exemplo prático de navegação autônoma em ambientes industriais é ilustrado na Figura 2.1. O diagrama representa uma planta industrial típica, na qual robôs móveis autônomos (*Autonomous Mobile Robots* – AMRs) circulam entre diferentes estações de trabalho.

No cenário apresentado, os AMRs se deslocam de forma independente entre as estações, executando tarefas como coleta de dados, monitoramento e suporte à supervisão dos processos produtivos. O módulo de Controle de Navegação Autônoma é responsável por definir e ajustar rotas em tempo real, adaptando-se às condições do ambiente com o auxílio de sensores. Já o Controle de Colisão atua para garantir a operação segura dos robôs ao lado de operadores humanos. Por fim, os dados coletados são utilizados para fins de manutenção preditiva, construção de gêmeos digitais e comunicação entre os próprios robôs, permitindo uma supervisão mais eficiente e inteligente da planta industrial.

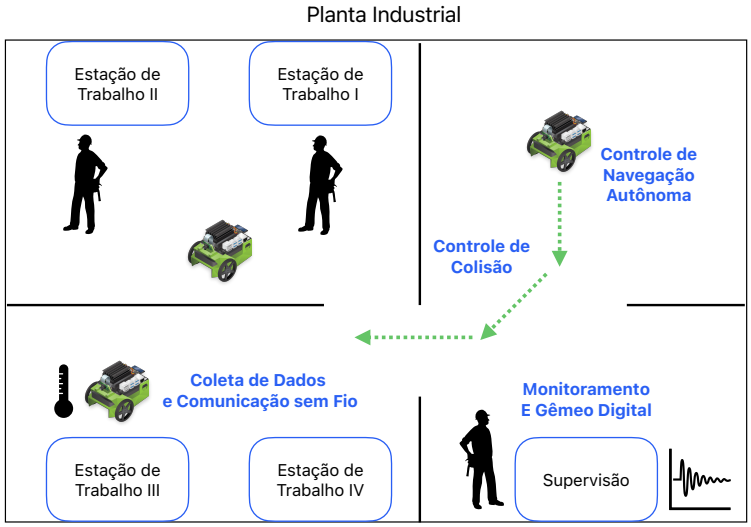


Figura 2.1: Exemplo de navegação autônoma de robôs em uma planta industrial.

A navegação autônoma, como representado na Figura 2.1, está alinhada com os desafios discutidos pelos autores [Dudek e Jenkin 2010], que classificam os principais problemas da navegação em quatro áreas:

- **Planejamento de Rota**: Como planejar um caminho do ponto (a, b) até (c, d), evitando colisões ao longo do trajeto.
- **Localização**: Como o robô determina sua posição no espaço, utilizando medições sensoriais locais, como dados de sensores IMU, encoders ou GPS.
- **Percepção**: Como o robô identifica áreas ocupadas por objetos, obstáculos, ou pessoas, e decide quais caminhos estão livres e seguros para a navegação.
- **Mapeamento**: Como o robô constrói e mantém uma representação do ambiente, permitindo que ele saiba sua localização a todo momento, mesmo em cenários dinâmicos.

Essas tarefas são executadas por meio de uma variedade de algoritmos, sendo as informações necessárias para sua implementação obtidas através de sensores. Esses dispositivos físicos são essenciais para que o robô obtenha dados sobre o ambiente e sua própria posição. Com isso, o robô consegue estimar seu deslocamento e a posição relativa dos objetos, facilitando a navegação e a interação com o ambiente [Osorio et al. 2014].

2.2 Localização e Percepção

A localização de pessoas, objetos e veículos (robô, drone, carro, etc.) em um ambiente representa um desafio em várias aplicações atuais e emergentes que exigem a análise e compreensão da cena circundante. Exemplos dessas aplicações incluem navega-

ção autônoma, realidade aumentada para indústria ou assistência a pessoas, mapeamento, e até mesmo entretenimento [Michael, Bodo e Vittorio 2019].

Os sensores de um sistema móvel autônomo são elementos essenciais que influenciam diretamente a resposta de controle e a tomada de decisão. Isso ocorre porque os sensores são responsáveis por perceber o ambiente ao redor, detectando obstáculos presentes e a dinâmica do meio. A dinâmica pode se referir, por exemplo, à movimentação dos obstáculos ao longo do tempo, o que é fundamental para garantir uma navegação segura e eficiente.

Sensores heterogêneos são dispositivos com diferentes capacidades de medição, faixas operacionais (*ranges*) e tipos de variáveis sensoriais, sendo utilizados conforme a aplicação desejada. Entre os principais sensores empregados em sistemas de navegação, destacam-se o Sistema Global de Navegação por Satélite (*Global Navigation Satellite System* – GNSS), o Sistema de Navegação Inercial (*Inertial Navigation System* – INS), câmeras RGB (*Red, Green, Blue*) e RGB-D (*Red, Green, Blue and Depth*), além de sensores baseados em laser, como o LiDAR (*Laser Imaging Detection and Ranging*). Esses sensores fornecem informações quantitativas essenciais sobre o ambiente físico, permitindo ao sistema interpretar e reagir adequadamente ao seu entorno. A escolha adequada de sensores para uma tarefa de navegação deve considerar diversos fatores, como o tempo de processamento dos dados brutos, a compatibilidade com o microcontrolador ou placa de controle utilizada, e o papel específico que o sensor desempenhará na arquitetura do robô.

2.2.1 Sensores de Posicionamento

Existem basicamente duas famílias de sensores de posicionamento, isto é, elementos capazes de fornecer informações sobre a localização do robô móvel no espaço, sendo eles: i) INS e ii) GNSS.

O Sistema de Navegação Inercial (*Inertial Navigation System* - INS) é amplamente utilizado para guiar mísseis e aeronaves em missões espaciais. A operação do INS baseia-se em medições feitas por um submódulo desse sistema, a Unidade de Medição Inercial (*Inertial Measurement Unit* - IMU), que é composta principalmente por acelerômetros e giroscópios. O IMU determina a aceleração (variação de velocidade ao longo do tempo) nos três eixos de Euler (*pitch, roll, yaw*) medindo a taxa de variação angular e a aceleração em relação a um eixo de referência. A Figura 2.2 ilustra os ângulos de Euler em que o sensor se baseia, assim como um chip real do IMU usado em muitos modelos de robôs, modelo MP6050.

O MPU-6050 é um módulo sensor que combina um giroscópio de 3 eixos e um acelerômetro de 3 eixos em um único chip. Ele utiliza tecnologia MEMS (sistemas

microeletromecânicos) para medir a velocidade angular e a aceleração linear ao longo dos três eixos ortogonais (x, y, z). Existem diferentes tecnologias de acelerômetros no mercado, tais como: piezoelétricos, por indução magnética ou capacitiva, sendo esta última a utilizada no MPU-6050.

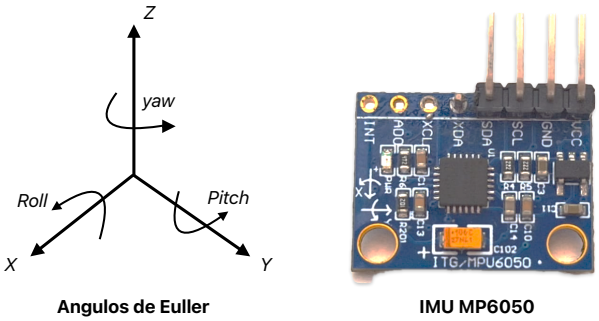


Figura 2.2: Angulos de Euller no sistema de medição inercial e IMU MP6050 - Autoria Própria.

O uso do Sistema Global de Navegação por Satélite (*Global Navigation Satellite System* - GNSS) é uma solução funcional e eficiente há vários anos. Diversos países continuam contribuindo para a construção de sistemas que atendam às necessidades dos usuários globais. Nesse mercado, destacam-se as seguintes tecnologias: o *Global Positioning System* (GPS), uma solução americana; o GLONASS, uma solução russa; e o BeiDou, desenvolvido pela China. Todas essas tecnologias operam em três segmentos de medição, denominados: i) Segmento Espacial, ii) Segmento de Controle e iii) Segmento de Usuário.

O Segmento Espacial consiste em satélites GNSS distribuídos uniformemente ao redor da órbita terrestre. Cada um desses satélites é responsável por transmitir um pseudocódigo aleatório, codificado e criptografado, contendo dados de sua órbita em várias frequências de ondas de rádio, conhecidas como Banda L (1151-1310 MHz). O Segmento de Controle tem a função de maximizar a acurácia e confiabilidade dos serviços GNSS, utilizando antenas localizadas em estações base no solo terrestre para monitorar e corrigir as trajetórias dos satélites.

O Segmento de Usuário consiste nos dispositivos receptores GNSS utilizados pelos usuários finais, como smartphones, veículos, robôs e sistemas de navegação. Esses dispositivos captam os sinais emitidos pelos satélites e utilizam algoritmos para calcular com precisão a posição do usuário em termos de latitude, longitude e altura elipsoidal. O cálculo é feito com base na triangulação de múltiplos sinais de diferentes satélites, em um sistema de referência global, o *World Geodetic System* 84 (WGS84).

A Figura 2.3, adaptada de [Michael, Bodo e Vittorio 2019], apresenta os segmentos utilizados na tecnologia de posicionamento global. Através desses sinais trocados com os satélites em órbita, os dados de localização do usuário são estimados e ajustados para garantir a precisão necessária em diversas aplicações.

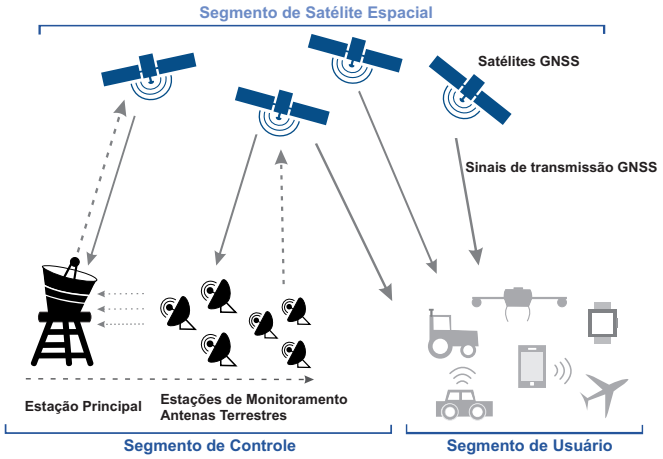


Figura 2.3: Exemplo do sistema de posicionamento global - Adaptado de [Michael, Bodo e Vittorio 2019]

O diagrama de blocos disposto pela Figura 2.4 apresenta um resumo dos sistemas de posicionamento explanados até aqui. Em resumo o GNSS fornece informações sobre as coordenadas georreferenciadas do robô e o INS fornece informações referentes à orientação do robô, também chamada de pose do robô.

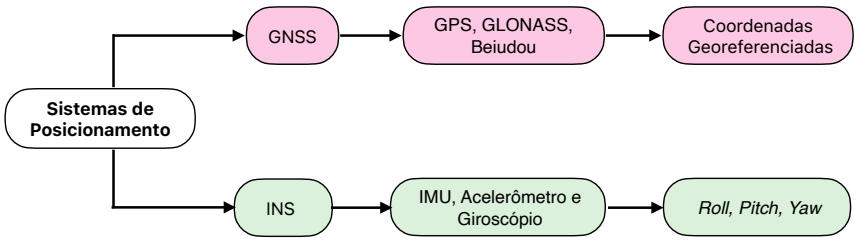


Figura 2.4: Sistemas de localização - Autoria Própria.

Como exemplo, a Figura 2.5 ilustra o módulo NEO 6M. O módulo GPS NEO-6M é uma solução popular e acessível para rastreamento de localização em diversos projetos de eletrônica e robótica. Fabricado pela u-blox, este módulo é conhecido por sua confiabilidade e precisão na obtenção de coordenadas geográficas.

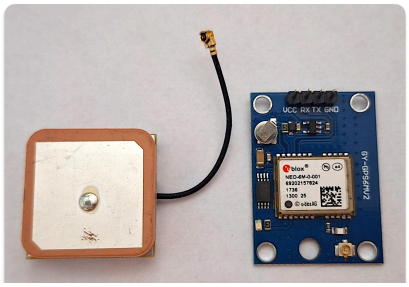


Figura 2.5: Módulo GPS NEO 6M e antena de porcelana - Autoria Própria

2.2.2 Sensores Perceptivos

De acordo com [Jahromi, Theja e Cetin. 2020], os sensores perceptivos são amplamente utilizados para capturar as características do ambiente em uma variedade de veículos móveis autônomos. Analogamente aos órgãos sensoriais humanos, esses sensores proporcionam uma "visão" do ambiente, permitindo a execução de tarefas como detecção de objetos, segmentação de obstáculos e percepção da dinâmica dos objetos para a tomada de decisão. Entre os principais tipos de sensores perceptivos, destacam-se as câmeras RGB, RGB-D e IR (Infravermelho). Neste tópico, é apresentada uma fundamentação teórica sobre os sensores utilizados neste trabalho, com foco específico nas câmeras RGB e RGB-D.

A câmera RGB (*Red, Green, Blue*) capta imagens utilizando o modelo de cores aditivas, onde a combinação das três cores básicas pode criar um espectro variado de cores. Essas câmeras operam com um de dois tipos de tecnologias de captura de imagem: o Dispositivo de Carga Acoplada (*Charged Coupled Device* - CCD) e o Semicondutor Complementar de Óxido Metálico (*Complementary Metal-Oxide Semiconductor* - CMOS). Ambas as tecnologias possuem características específicas que influenciam a qualidade da imagem e a eficiência do sensor. O CCD é conhecido por sua alta qualidade de imagem e sensibilidade à luz, enquanto o CMOS é favorecido por seu baixo custo de produção e menor consumo de energia, tornando-o mais adequado para aplicações onde a eficiência energética é crucial.

Segundo [Osorio et al. 2014], o CCD é um dispositivo semicondutor composto por uma matriz de elementos sensíveis à luz, chamados de *pixels*. Quando os fótons incidem nos *pixels*, eles liberam sinais elétricos. Esses sinais são então transferidos para um registrador do CCD, onde são amplificados e digitalizados. Ainda de acordo com [Osorio et al. 2014], o funcionamento do CMOS é bastante semelhante ao do CCD. No entanto, o CMOS possui um conjunto de transistores dedicados que facilitam a amplificação dos sinais coletados pela matriz de *pixels*.

A câmera RGB-D (D-Depth) é um tipo de câmera RGB que fornece informações

adicionais além da imagem padrão. Esse tipo de câmera captura dados de profundidade da cena, como a distância dos objetos presentes, e os representa por meio de um mapa de cores, onde áreas mais claras indicam objetos mais próximos e áreas mais escuras indicam objetos mais distantes. A câmera RGB-D tem sido amplamente utilizada na navegação autônoma e se popularizou no mercado através do *videogame Xbox*, da empresa *Microsoft*. As câmeras RGB-D possuem um *hardware* sofisticado, composto principalmente por três sensores: i) CMOS, ii) Câmera Infravermelha, e iii) Emissor LASER. O princípio de funcionamento da câmera RGB-D baseia-se na projeção de um padrão de *pixels* e na estimativa da deformação da cena, utilizando o emissor LASER e o receptor infravermelho. O cálculo dessa deformação é feito por meio de um esquema de triangulação, conforme ilustrado na Figura 2.6, que leva em consideração a posição do emissor LASER, da câmera infravermelha e do objeto.

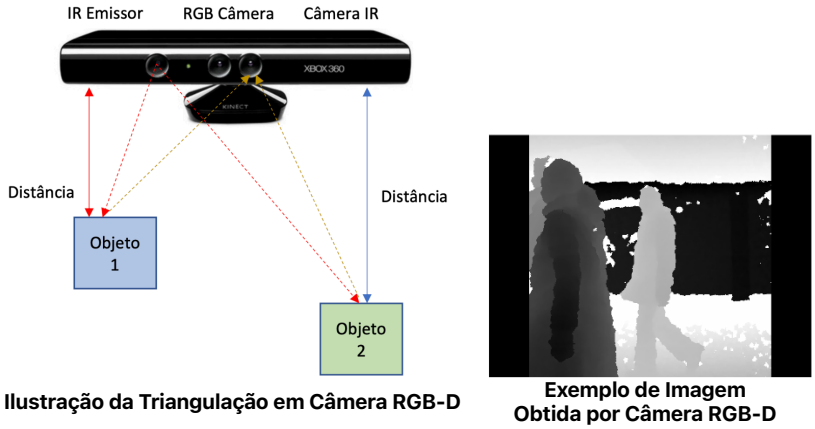


Figura 2.6: Triangulação na Percepção RGB-D e Exemplo de Imagem Capturada - Autoria Própria.

2.2.3 Sensores para Mapeamento

O processo de mapeamento em robótica móvel é essencial para que os robôs possam navegar eficientemente em seus ambientes. O mapeamento permite que o robô construa uma representação do espaço ao seu redor, o que é crucial tanto para a navegação quanto para a realização de tarefas específicas em ambientes dinâmicos ou desconhecidos. Esse processo é geralmente realizado com o uso de sensores que detectam e medem a distância até os objetos ao redor, criando mapas detalhados do ambiente.

Entre os sensores mais utilizados para mapeamento estão os sensores de *Time of Flight* (ToF) e os LIDARs (*Light Detection and Ranging*), ambos utilizando feixes de laser para medir distâncias. Os sensores ToF funcionam emitindo um único feixe de laser e medindo o tempo que esse feixe leva para retornar ao sensor após refletir em um objeto.

Essa medição permite calcular a distância entre o sensor e diversos pontos do ambiente, fornecendo uma imagem de profundidade precisa em tempo real.

Os sensores LIDAR operam de maneira semelhante, mas a grande diferença é que eles emitem múltiplos feixes de laser em diferentes direções, permitindo o mapeamento em 360 graus. Ao captar a luz refletida desses múltiplos feixes, o LIDAR gera uma nuvem de pontos, que é uma representação tridimensional detalhada do ambiente. Essa capacidade de capturar múltiplos pontos simultaneamente torna os LIDARs extremamente eficientes para o mapeamento de áreas internas ou externas com alta precisão e cobertura completa.

Um exemplo acessível de sensor LIDAR é o LD19, ilustrado na Figura 2.7. Esse LIDAR representa uma opção de baixo custo ideal para projetos de robótica móvel, sendo capaz de realizar um escaneamento completo de 360 graus com um alcance de até 12 metros. Alimentado por uma fonte de 12V, ele utiliza o protocolo de comunicação UART, o que facilita significativamente sua integração com uma variedade de microcontroladores e microprocessadores.

Figura 2.7: Sensor LIDAR LD19 e Exemplo de Aplicação do LIDAR com Mapeamento de Ambiente - Autoria Própria

Embora o LD19 não ofereça a mesma resolução ou alcance dos modelos mais avançados, ele ainda fornece dados suficientes para aplicações menos exigentes ou para projetos educacionais. Isso torna a tecnologia LIDAR acessível a entusiastas, estudantes e pesquisadores. Esse sensor é particularmente útil em ambientes internos ou em aplicações onde a precisão de longo alcance não é crítica, mas onde medições de distâncias curtas e precisas são necessárias para o mapeamento e navegação de robôs.

2.3 Arquiteturas de Controle

A arquitetura de controle de um AMR é definida por [Hayes-Roth 1995] como uma coleção de componentes estruturais, denominados de **módulos**, onde percepção, raciocínio e ação ocorrem de forma integrada. De acordo com [Arkin 1998] e

[Osorio et al. 2014], uma arquitetura de controle está relacionada à parte de *software* de um AMR, caracterizando os módulos que devem ser utilizados no sistema para cumprir um determinado objetivo. Além disso, ela é responsável pela interação entre esses módulos durante a execução das ações do robô no ambiente.

Ainda de acordo com [Osorio et al. 2014], os módulos básicos presentes em um sistema de controle de um robô móvel podem ser classificados em três grupos: Módulos de Percepção e Sensoriamento, Planejamento de Tarefas e Ação. O módulo de Percepção refere-se à forma como os sensores do robô detectam o ambiente, ou seja, como são realizadas as medições de grandezas necessárias para localizar, orientar e controlar o robô. Os Módulos de Planejamento utilizam algoritmos computacionais para definir a rota que o robô seguirá durante a missão proposta. Já os Módulos de Ação são responsáveis por acionar, desabilitar ou controlar localmente os atuadores do robô, como motores de rodas ou de juntas.

A classificação da arquitetura de controle está relacionada à característica de reatividade ou deliberação e pode ser dividida em: Puramente Deliberativa, Puramente Reativa ou Híbrida. Na robótica, deliberação significa tomada de decisão, planejamento da ação a ser tomada. Desta forma, as arquiteturas deliberativas são em grande parte compostas por módulos, ou algoritmos, de planejamento. Os autores [Osorio et al. 2014] citam um exemplo de deliberação: um robô que transporta roupas dentro de uma casa. Este robô tem conhecimento do mapa da casa e seu algoritmo de planejamento traça uma rota para que ele possa pegar as roupas de um quarto e levá-las para a lavanderia. Observa-se que neste caso, o robô necessita de um conhecimento mapa do ambiente em que irá realizar suas atividades.

A reatividade, por outro lado, está ligada à resposta imediata a uma leitura sensorial obtida em um determinado momento. Por exemplo o robô não tem conhecimento do mapa do ambiente, mas foi programado para que, ao perceber um obstáculo, realize um movimento de deflexão para evitar colisões. Este robô pode seguir navegando na casa e em algum momento, devido a sua capacidade de desviar, chegar ao destino, mas não é algo garantido. Vale destacar que o controle reativo tem como principal característica resposta a uma mudança abrupta gerada no sistema, como o ato de desviar. Ressalta-se ainda que as arquiteturas de controle podem ser combinadas até certo ponto, formando as arquiteturas híbridas, que podem combinar algoritmos reativos e deliberativos. A Figura 2.8 ilustra a abordagem clássica referentes as arquiteturas de controle em robótica móvel.

Segundo os autores [Carreras 2003] [Carreras et al. 2005], a evolução da Inteligência Artificial levou a novas formas de entender e desenvolver arquiteturas de controle para sistemas robóticos móveis, e um dos principais conceitos explorados em sua tese é o de Sistemas Baseados em Comportamento. Um sistema baseado em comportamento é semelhante a uma arquitetura reativa. Sistemas reativos fornecem respostas rápidas e em

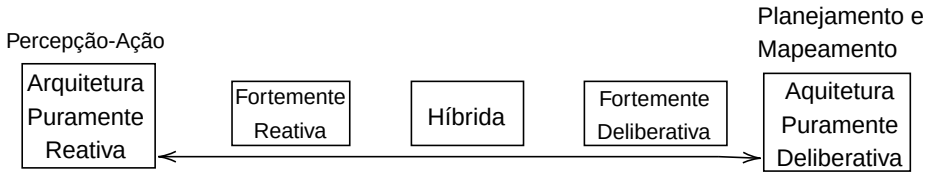


Figura 2.8: Arquiteturas clássicas de controle - Adaptada de [Osorio et al. 2014].

tempo real usando uma coleção de regras pré-programadas e, portanto, esses sistemas são incapazes de deliberar ou aprender novos comportamentos. Por outro lado, o Sistemas Baseados em Comportamento podem armazenar estados em uma representação distribuída, ou seja: estímulo, comportamento e resposta do robô. Isso permite um grau de deliberação e reação ao mesmo tempo. O robô aprende diferentes comportamentos ou diferentes métodos de coordenação que melhoram o desempenho da navegação.

De acordo com [Stefano. 2021], o *Reinforcement Learning* (RL) é uma das várias técnicas que podem ser usadas para implementar arquiteturas baseadas em comportamento. Esta técnica será discutida em mais detalhes no Capítulo 3. O RL tem sido amplamente utilizado para aprender a estrutura interna dos comportamentos, mapeando os estados percebidos para controlar as ações, enquanto maximiza as recompensas futuras acumuladas. A Figura 2.9 ilustra uma arquitetura baseada em comportamento.

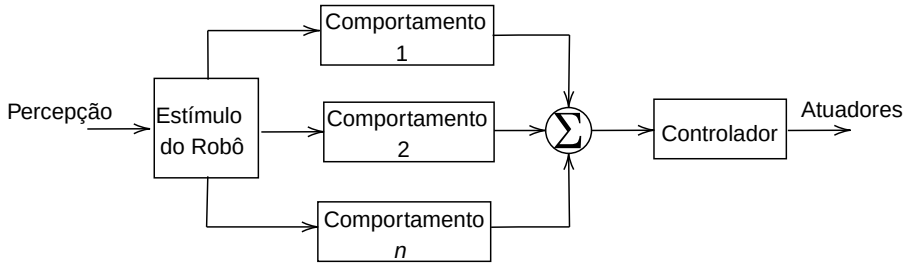


Figura 2.9: Arquitetura híbrida baseada em comportamento - Adaptada de [Stefano. 2021].

Uma parte crucial de qualquer arquitetura de controle em robótica móvel é a capacidade de obter estimativas precisas sobre o estado do robô e seu ambiente. Para que as decisões de controle sejam eficazes, os módulos de percepção devem fornecer dados confiáveis, como posição, orientação e velocidade, que muitas vezes são obtidos por sensores que podem sofrer interferências ou ruídos. Nesse contexto, o Filtro de Kalman desempenha um papel essencial, ao permitir que o robô estime e corrija continuamente sua estimativa de estado, mesmo na presença de ruído sensorial. A seguir, será discutido

o uso do Filtro de Kalman como uma ferramenta fundamental para melhorar a precisão das estimativas de estado em arquiteturas de controle robótico.

2.4 Filtro de Kalman

Outra ferramenta essencial para a navegação autônoma é o Filtro de Kalman (*Kalman Filter* - KF) [Kalman e Others 1960]. Esse algoritmo permite estimar os estados que refletem a dinâmica de um sistema ou processo linear. O KF é especialmente útil para corrigir informações provenientes de medições sensoriais incertas ou ruidosas, ponderando-as com o modelo de transição de estados do sistema. Isso resulta em uma estimativa ótima do estado do sistema.

Para sistemas não lineares, o Filtro de Kalman Estendido (*Extended Kalman Filter* - EKF) oferece uma alternativa mais sofisticada. O EKF adapta-se a esses sistemas através de linearizações realizadas durante o ciclo de predição e atualização do algoritmo, permitindo uma aplicação eficaz em cenários onde o KF convencional não seria adequado.

No processo de correção do KF, as estimativas de estado preditas e as medições são combinadas para gerar uma estimativa ótima. A Figura 2.10 ilustra visualmente esse processo de fusão de informações, onde as distribuições gaussianas representam a incerteza na predição do estado, a medição e a estimativa de estado final.

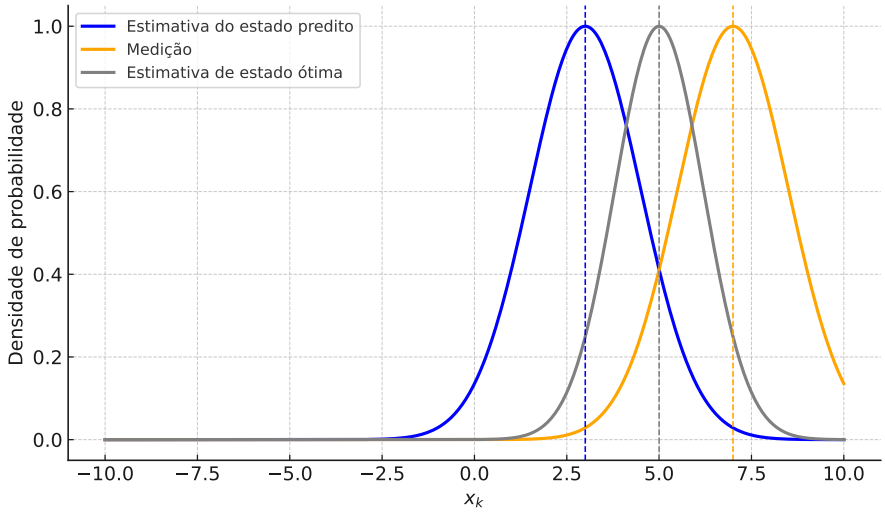


Figura 2.10: Filtro de Kalman: Correção de Estado com Distribuições Gaussianas.

Como ilustrado na Figura 2.10, o KF pondera a predição do estado, representada pela curva gaussiana azul, com a medição ruidosa, mostrada pela curva laranja. A combinação dessas informações resulta na curva cinza, que representa a estimativa de estado ótima. Esse processo permite que o KF corrija as incertezas associadas a medições sensoriais e predições feitas com base no modelo do sistema.

O Filtro de Kalman (KF) teve uma aplicação significativa como estimador de trajetórias para os módulos espaciais durante a missão Apollo, que tinha como objetivo levar o homem à Lua. Além desse marco histórico, o KF também encontra inúmeras aplicações na teoria de sistemas de controle, especialmente no campo do controle moderno [Chrif e Kadda 2014] [Lichota, Dul e Karbowski 2020]. Por ser um algoritmo computacionalmente eficiente, o KF pode ser facilmente integrado a métodos de inteligência artificial para resolver problemas envolvendo incertezas sensoriais e predição de estados, sem aumentar significativamente o custo computacional.

De acordo com a teoria de controle no espaço de estados, um sistema pode ser descrito por um conjunto de parâmetros ou variáveis que caracterizam seus aspectos físicos e comportamentos. Este conjunto de variáveis são chamadas de variáveis de estado e normalmente são descritas matematicamente pelo vetor $\mathbf{x}$. Nem toda variável de interesse pode ser medida diretamente em um processo. Por isso $\mathbf{x}$ pode ser estimado por meio de um vetor de medições $\mathbf{z}$. O KF é capaz de estimar um estado ótimo para o sistema, combinando a matriz de medição $\mathbf{Z}$, obtida com sensores, com o modelo em que o sistema evolui no tempo (instante k), ou seja x_k. Uma predição de Kalman é matematicamente descrita por [Aguirre 2015]:

$$\mathbf{x}_{k+1} = \mathbf{A}\mathbf{x}_k + \mathbf{B}u \tag{2-1}$$

onde $\mathbf{A}$ é a matriz de transição de estados que descreve como o sistema evolui de um estado para o outro. Caso o sistema seja controlado, acrescenta-se ainda o produto da matriz de controle $\mathbf{B}$ com o vetor de controle u, elementos estes que descrevem a ação externa de uma variável de controle.

Todo sensor possui erros associados ao processo de medição. Como o KF é de natureza gaussiana, ou seja toda sua modelagem matemática parte de associações de funções de probabilidade gaussiana, as incertezas também são modeladas a partir desta função de densidade de probabilidade. A matriz de covariança $\mathbf{P}$ representa as incertezas do modelo e sua equação de atualização é descrita por:

$$\mathbf{P}_{k+1} = \mathbf{A}\mathbf{P}_k\mathbf{A}^T + \mathbf{Q} \tag{2-2}$$

onde $\mathbf{Q}$ é uma matriz que representa ruídos externos ao sistema, por exemplo ruídos relacionados a pista onde o veículo autônomo vai navegar. Por questões de simplificação, diversos autores adotam $\mathbf{Q} = 0$. As equações (2-1) e (2-2) são termos da etapa de predição do KF. Elas descrevem como o sistema evolui gradualmente com o passar do tempo. Entretanto observa-se que não existe nenhuma informação sobre a medição sensorial do processo, ou seja, é uma transição de estado exclusivamente baseada em modelo matemático. Desta forma é necessário inserir uma etapa do KF que correlaciona as

medições sensoriais, denominada etapa de correção. A primeira equação da etapa de correção consiste em calcular o ganho Kalman, dado por:

$$\mathbf{K}_k = \mathbf{P}_k\mathbf{H}^T(\mathbf{H}\mathbf{P}_k\mathbf{H}^T + \mathbf{R})^{-1} \tag{2-3}$$

onde **H** é a matriz de transferência sensorial. Esta matriz é responsável por indicar quais sensores serão utilizados no processo de correção do KF. A matriz de covariança sensorial **R** apresenta as incertezas relacionadas aos sensores utilizados.
A atualização do estado com as medições é dada por:

$$\mathbf{x}_{k+1} = \mathbf{x}_{k+1} + \mathbf{K}_k(\mathbf{Z}_k - \mathbf{H}x_{k+1}) \tag{2-4}$$

A atualização da matriz de covariança é dada por:

$$\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H})\mathbf{P}_{k+1} \tag{2-5}$$

No EKF são calculadas as derivadas parciais da equação de transição de estados, chamada matriz jacobiana (jF). A aplicação desta matriz basicamente lineariza um sistema não linear sobre um ponto de operação e é aplicada basicamente sobre a matriz **B**, que contém as não linearidades do sistema. A matriz apresentada abaixo representa a jacobiana (jF) [Dudek e Jenkin 2010].

$$jF = \begin{bmatrix} \frac{\delta\ f1}{\delta\ x} & \frac{\delta\ f1}{\delta\ y} & \frac{\delta\ f1}{\delta\ z} \\ \frac{\delta\ f2}{\delta\ x} & \frac{\delta\ f2}{\delta\ y} & \frac{\delta\ f2}{\delta\ z} \\ \frac{\delta\ f3}{\delta\ x} & \frac{\delta\ f3}{\delta\ y} & \frac{\delta\ f1}{\delta\ z} \end{bmatrix} \tag{2-6}$$

O Algoritmo 1 dispõe todas as etapas explanadas acima em relação a aplicação do filtro de Kalman como um preditor de estados, apresentando a etapa de predição e atualização do filtro.

Embora o Filtro de Kalman seja amplamente utilizado para fusão de sensores tradicionais, como LIDAR e IMU, ele apresenta limitações quando se trata da fusão de dados sensoriais multimodais, como imagens capturadas por câmeras. A fusão de dados provenientes de diferentes modalidades, como visão e sensores de distância, exige abordagens mais sofisticadas para capturar a complexidade desses dados. Na próxima seção, exploraremos as diferentes topologias de fusão de dados, como fusão precoce (early fusion) e fusão tardia (late fusion), que possibilitam uma integração mais robusta de informações provenientes de múltiplos sensores.

Algorithm 1 Filtro de Kalman

Entrada: $\mathbf{x}_{k-1}$ (estado anterior), $\mathbf{P}_{k-1}$ (matriz de covariância anterior), $\mathbf{u}_k$ (controle de entrada), $\mathbf{z}_k$ (medição atual)
Parâmetros: $\mathbf{A}$ (matriz de transição de estado), $\mathbf{B}$ (matriz de controle), $\mathbf{H}$ (matriz de medição), $\mathbf{Q}$ (covariância do processo), $\mathbf{R}$ (covariância da medição)

procedure PREDIÇÃO
    $\mathbf{x}_k^- = \mathbf{A}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_k$
    $\mathbf{P}_k^- = \mathbf{A}\mathbf{P}_{k-1}\mathbf{A}^T + \mathbf{Q}$
end procedure

procedure ATUALIZAÇÃO
    $\mathbf{K}_k = \mathbf{P}_k^-\mathbf{H}^T(\mathbf{H}\mathbf{P}_k^-\mathbf{H}^T + \mathbf{R})^{-1}$
    $\mathbf{x}_k = \mathbf{x}_k^- + \mathbf{K}_k(\mathbf{z}_k - \mathbf{H}\mathbf{x}_k^-)$
    $\mathbf{P}_k = (\mathbf{I} - \mathbf{K}_k\mathbf{H})\mathbf{P}_k^-$
end procedure

Saída: $\mathbf{x}_k$ (estado estimado), $\mathbf{P}_k$ (matriz de covariância atualizada)

2.5 Fusão de Dados Sensoriais

A Fusão de dados, ou informações sensoriais, é uma habilidade constantemente utilizada por seres humanos em suas atividades do dia a dia que exigem, principalmente, a compreensão físicas dos objetos ao seu redor. Um exemplo de fusão de dados é a união dos sentidos visuais com o tato (habilidade de tocar) para destreza e reconhecimento de objetos.

O estudo da interação entre informações de diferentes sentidos foi observada e testada experimentalmente pelos pesquisadores [McGurk e MacDonald 1976]. O Efeito McGurk basicamente confirma que a visão do ser humano pode interferir em sua audição, uma espécie de ilusão sonora, onde o individuo acredita que ouve um determinado fonema, quando na verdade está ouvindo outro.

Segundo [Fayyad et al. 2020], a fusão sensorial é a tarefa de gerenciar e acoplar dados e informações obtidas através de diferentes tipos sensores para melhorar um critério específico ou a tomada de decisão em um determinado processo. A técnica de fusão normalmente ocorre computacionalmente por meio de um algoritmo que executa esta tarefa. O dado sensorial fundido é enriquecido, aprimorado e se torna mais confiável que os gerados por sensores individuais separadamente.

Hoje a fusão sensorial é aplicada em diversas áreas de conhecimento tais como: i) área militar, ii) área médica, iii) sensoreamento remoto, iv) robótica e veículos autônomos. Em AMRs a percepção do ambiente através da combinação de diferentes sensores colaboram para a correta tomada de decisão. [Fayyad et al. 2020] afirma que nas últimas décadas diversas pesquisas surgiram visando o desenvolvimento e melhoria de métodos

de fusão para AMRs. No campo da robótica as abordagens clássicas para fusão de dados oriundos de sensores são através de métodos probabilísticos baseados em inferência Bayseana, entre eles o Filtro de Kalman.

[Bednarek, Kicki e Krzysztof 2020] destacam que com o crescimento da tecnologia de sensores multimodais, isto é, sensores que fornecem dados em diferentes dimensões, tamanhos e particularidades, a aplicação dos métodos Bayseanos começaram a apresentar dificuldades, pois estes métodos trabalham normalmente com pequenas dimensões, ou dados de características homogêneas. Para superar este problema, o campo do Aprendizado de Máquina e Inteligência Artificial têm sido útil e alvo das atuais pesquisas em fusão sensorial.

Atualmente existem diversos esquemas modernos de fusão de sensores listados na literatura [Bednarek, Kicki e Krzysztof 2020] [Fayyad et al. 2020]. Os três principais esquemas de fusão são: i) Fusão Antecipada (*Early Fusion*), ii) Fusão de Características (*Feature Fusion*) e iii) Fusão Tardia (*Late Fusion*). A Figura 2.11 ilustra os três esquemas citados.

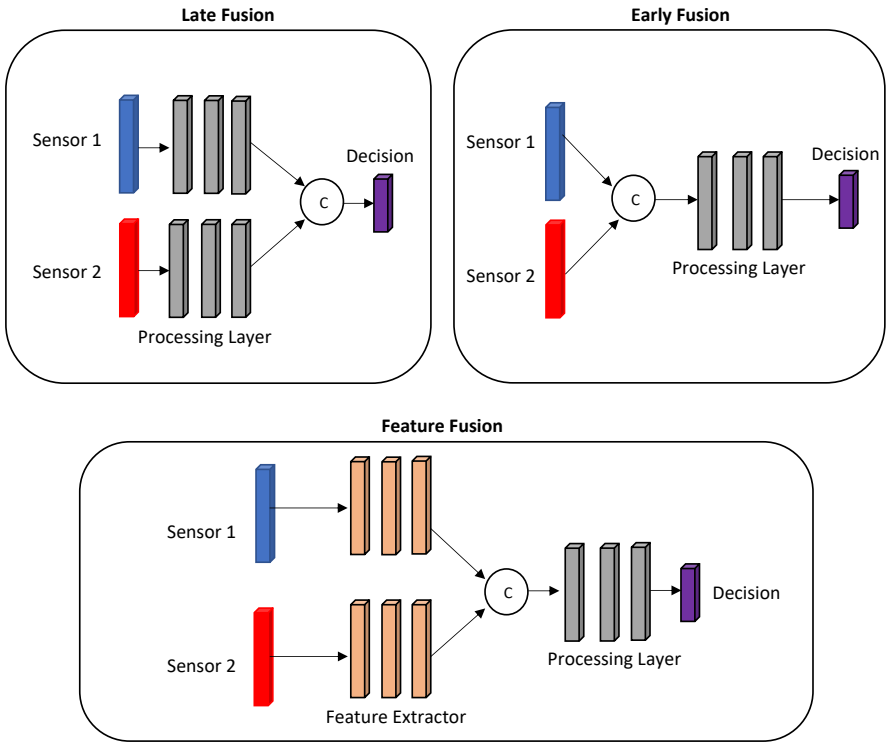


Figura 2.11: Representação dos atuais métodos de fusão sensorial - Autoria Própria

Na Fusão Unificada os dados oriundos de diferentes sensores são unificados em sua forma inicial, em sua forma bruta (*raw level*). Teoricamente esta fusão ocorre após a leitura imediata dos sensores. Já a Fusão de Características, como o próprio nome diz,

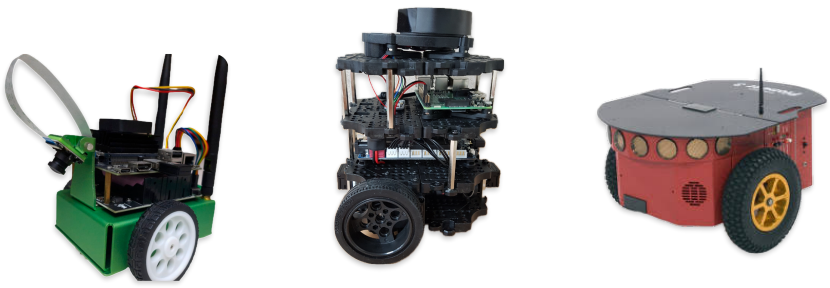
um extrator de características (normalmente uma rede neural) é usado após a leitura dos dados brutos do sensor para fundir as principais propriedades daquele determinado dado. Na Fusão Tardia, múltiplos classificadores (normalmente uma rede neural profunda) são usados antes da união de dados, o que significa que esta técnica está mais próxima possível de um nível de decisão.

Após explorar as técnicas de fusão de dados sensoriais, que permitem uma percepção mais precisa e robusta do ambiente, é fundamental compreender como essas informações se integram ao modelo matemático que governa o movimento do robô. As próximas seções descrevem o modelo matemático do robô diferencial, utilizado nesta tese, bem como o processo de controle de atuadores e odometria.

2.6 Modelo Cinemático Diferencial

Existem diversos modelos mecânicos de AMR no mercado, tais como robôs móveis com rodas, robôs humanoides, robôs aéreos não tripulados entre outros. O modelo cinemático é o modelo matemático que reproduz a movimentação mecânica do robô, seja ela de rotação ou translação, em um determinado plano. O modelo cinemático é essencial para o desenvolvimento do sistema de controle do AMR, bem como a compreensão da sua dinâmica.

Neste trabalho enfatiza-se o modelo de AMR denominado diferencial. O acionamento diferencial é talvez o mecanismo de acionamento mais simples possível [Dudek e Jenkin 2010]. Ele é frequentemente usado em ambientes internos, pequenos e de baixo custo. Alguns robôs como o Jetbot, TurtleBot3 e Pioneer 3-DX também utilizam essa tecnologia. Esses robôs são ilustrados na Figura 2.12. Nesta tese, utiliza-se nos experimentos práticos e teóricos com o robô Jetbot e Pioneer 3-DX. Como retratado na Figura 2.13, um AMR de acionamento diferencial consiste em duas rodas montadas em um eixo comum controlados por motores separados.



Robô Jetbot Robô Turtlebot3 Robô Pionner 3dx

Figura 2.12: Modelos Comerciais de Robôs Diferenciais.

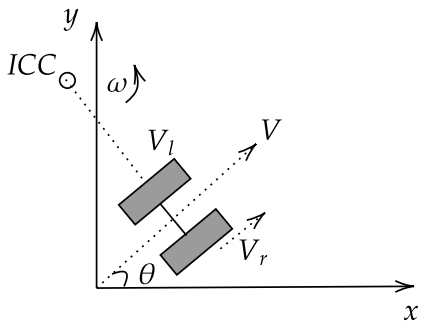


Figura 2.13: Representação em vista superior do robô móvel diferencial no plano cartesiano - Autoria Própria.

Para que o robô diferencial execute um movimento de rolamento (translação) deve existir um ponto de referência entre as rodas. Este ponto de referência é conhecido na literatura como Centro Instantâneo de Curvatura (*Instantaneous Center of Curvature* - ICC). O AMR diferencial possui três graus de liberdade: i) sua posição *(x,y)* no plano e ii) sua orientação θ. Este vetor posição $[x, y, \theta]$ é conhecido na literatura como pose do robô.

A velocidade de controle das rodas da direita e esquerda V_l e V_r respectivamente, determinam o movimento do veículo. Ao variar a velocidade relativa das duas rodas a velocidade angular ω sobre o ICC pode ser variada, possibilitando a escolha de uma trajetória. Se V_l for maior que V_r o robô tende a rotacionar no sentido horário, caso contrário, o robô tende a rotacionar no sentido anti-horário.

$$\omega(R + \frac{l}{2}) = V_r \tag{2-7}$$

$$\omega(R + \frac{l}{2}) = V_l \tag{2-8}$$

onde l é o comprimento do eixo, a distância entre o centro das duas rodas. R é a distância entre o ICC e o ponto médio entre as duas rodas. R pode ser simplificado como:

$$R = \frac{l}{2} \frac{V_l + V_r}{V_l - V_r} \tag{2-9}$$

Se $V_l = V_r$, então o raio $R = \infty$ e o robô se move em uma trajetória retilínea. Se $V_l = -V_r$ então o raio $R = 0$ e o robô rotaciona sobre o próprio eixo, mantendo-se no mesmo lugar.

Conforme explicado em [Dudek e Jenkin 2010], robôs de tração diferencial são muito sensíveis à velocidade relativa entre as duas rodas. Pequenos erros nas velocidades aplicadas resultam em trajetórias diferentes. Por esta razão alguns AMRs de tração diferencial utilizam uma roda a mais, sem tração e localizada em seu centro geométrico,

para aumentar o equilíbrio do robô em determinados terrenos.

A velocidade linear resultante do AMR diferencial é dada por:

$$V = \frac{V_l + V_r}{2} \tag{2-10}$$

Considerando que a entrada de controle $u(t)$ para um AMR com tração diferencial é:

$$u(t) = [V, \omega] \tag{2-11}$$

onde V é a velocidade linear do robô e ω é a velocidade angular em relação ao ICC.
A partir de V e ω é possível controlar os movimentos do AMR em cada eixo do plano xy. Em geral o AMR é capaz de se mover em uma direção particular θ a uma velocidade $V(t)$. A dinâmica da pose (vetor de posicionamento) do robô diferencial em relação ao tempo pode ser calculada de acordo com as integrais abaixo:

$$x(t) = \int_0^t V(t)\ cos(\theta)(t)dt \tag{2-12}$$

$$y(t) = \int_0^t V(t)\ sin(\theta)(t)dt \tag{2-13}$$

$$\theta(t) = \int_0^t \omega(t)dt \tag{2-14}$$

A partir destas integrais, o modelo cinemático, em termos matriciais, pode ser simplificado de acordo com a equação abaixo:

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} r\cos(\theta)(t)/2 & r\cos(\theta)(t)/2 \\ r\sin(\theta)(t)/2 & r\sin(\theta)(t)/2 \\ r/L & -r/L \end{bmatrix} \times \begin{bmatrix} \omega_r(t) \\ \omega_l(t) \end{bmatrix} \tag{2-15}$$

onde r é o raio da roda do robô. Simplificando o modelo para a entrada de controle $u(t)$:

$$\begin{bmatrix} x' \\ y' \\ \theta' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \times \begin{bmatrix} V \\ \omega \end{bmatrix} \tag{2-16}$$

A partir do modelo cinemático do AMR diferencial é possível projetar controladores reativos locais, tais como o Proporcional Integral e Derivativo (PID) e o regulador linear quadrático (LQR). Além disso o modelo cinemático permite a aplicação do Filtro de Kalman, descrito na Seção 2.4, que exige o conhecimento prévio do modelo matemático do sistema. A próxima seção apresenta conceitos de controladores locais.

2.7 Controladores

Controladores locais e globais diferem fundamentalmente em escopo e aplicação dentro da navegação autônoma de robôs.

Os controladores locais são aplicados nos atuadores do AMR para controlar a dinâmica destes veículos dotados de rodas. Geralmente estes controladores são reativos e estão em um nível inferior, isto é, um nível abaixo do planejamento das trajetórias. Neste controle objetiva-se manter as velocidades, ou trajetórias, estabilizadas em torno de um ponto de equilíbrio perante as possíveis forças externas que possam perturbar o sistema [Osorio et al. 2014].

Em contraste, controladores globais operam em um nível mais alto, geralmente envolvendo o planejamento de trajetória e a estratégia de navegação a longo prazo. Eles utilizam uma visão mais abrangente do ambiente, frequentemente integrando dados de múltiplas fontes e sensores para fazer decisões estratégicas sobre o percurso do robô. Esses controladores são cruciais para otimizar a rota em termos de eficiência energética, tempo de viagem ou segurança geral, integrando objetivos globais com as condições locais processadas pelos controladores locais.

De forma básica, existem dois tipos de controle local: i) malha aberta e ii) malha fechada. No controle em malha aberta o valor desejado, ou variável de controle, não é monitorada por intermédio de sensores. Por exemplo, deseja-se que o AMR navegue em uma trajetória a $1m/s$, este valor é conhecido como *setpoint*. Aplica-se uma tensão elétrica sobre o atuador do AMR, no caso os motores elétricos das rodas, esta tensão reflete em uma velocidade linear e angular do robô ($u(t) = [V, \omega]$) este vetor é conhecido como vetor de controle. No controle em malha aberta não é possível que o algoritmo de controle ajuste de forma exata o *setpoint*, e necessita da intervenção humana para um ajuste via tentativa e erro da variável de controle para manter *setpoint*.

A alternativa ao controle de malha aberta é o controle de malha fechada. Sob controle de malha fechada o valor de saída do sistema é monitorado e o erro entre o *setpoint* e o atual valor é usado para acionar um algoritmo de controle que ajusta o dispositivo para corresponder ao valor desejado [Dudek e Jenkin 2010]. A Figura 2.14 ilustra um sistema de malha de controle fechada aplicada em um AMR.

O Controle Proporcional-Integral-Derivativo (PID) representa um dos pilares do controle automático devido à sua simplicidade e eficácia em uma vasta gama de aplicações. No controle PID, o sinal de controle é gerado com base em três termos: o erro proporcional, que responde à magnitude do erro atual; o termo integral, que acumula erros passados, ajudando a eliminar o erro residual de estado estacionário; e o derivativo, que prevê erros futuros, melhorando a resposta transitória e a estabilidade. Aplicado a robôs móveis, o controle PID pode efetivamente gerenciar a dinâmica de movimentação,

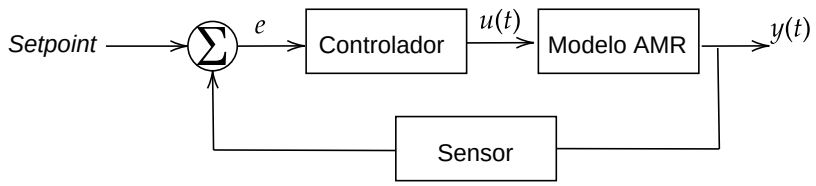


Figura 2.14: Representação de um controle em Malha Fechada - Autoria Própria.

ajustando a velocidade e a direção para alcançar um desempenho de navegação desejado com precisão.

Técnicas de Inteligência Artificial (IA), especialmente o aprendizado de máquina e o aprendizado profundo, revolucionou a abordagem tradicional de controle em sistemas robóticos, principalmente em controladores globais. Algoritmos de IA permitem que o controle do robô seja adaptativo e sensível ao contexto, aprendendo a partir da interação contínua com o ambiente. Por exemplo, redes neurais podem ser treinadas para predizer ações de controle ótimas em situações complexas onde modelos matemáticos tradicionais são ineficazes. Esses controladores baseados em IA não apenas aprimoram a precisão e eficiência, mas também possibilitam que o robô desenvolva comportamentos mais sofisticados, como a adaptação a mudanças inesperadas no ambiente ou a otimização do comportamento com base em objetivos de longo prazo.

2.8 Odometria

A odometria é uma técnica fundamental utilizada em sistemas de navegação autônoma para estimar a posição e a orientação de um robô ao longo do tempo. Baseando-se na integração das velocidades medidas, a odometria acumula informações de movimento para determinar a trajetória percorrida pelo robô. Este método é amplamente utilizado devido à sua simplicidade e baixo custo de implementação, sendo essencial para diversas aplicações em robótica móvel. Em muitos casos, a odometria é complementada por outras fontes de dados através da fusão sensorial, melhorando a precisão e a robustez das estimativas de posição e orientação.

A odometria geralmente se baseia na contagem de pulsos de encoders acoplados às rodas do robô. Os encoders medem a rotação das rodas, permitindo calcular a distância percorrida com base no número de pulsos e no perímetro das rodas. Para robôs com duas rodas motrizes, as equações básicas da odometria são derivadas do modelo de movimento diferencial, conforme descrito a seguir. Adicionalmente, a odometria pode ser obtida através da fusão de dados de sensores IMU (Unidade de Medição Inercial) e encoders,

combinando medições de aceleração e velocidade angular com a rotação das rodas para fornecer estimativas mais precisas de movimento.

Considerando um robô com duas rodas, a posição (x, y) e a orientação θ podem ser atualizadas usando as seguintes equações diferenciais:

$$\Delta s = \frac{r}{2}(\Delta\phi_R + \Delta\phi_L) \tag{2-17}$$

$$\Delta\theta = \frac{r}{d}(\Delta\phi_R - \Delta\phi_L) \tag{2-18}$$

$$\Delta x = \Delta s \cos\left(\theta + \frac{\Delta\theta}{2}\right) \tag{2-19}$$

$$\Delta y = \Delta s \sin\left(\theta + \frac{\Delta\theta}{2}\right) \tag{2-20}$$

Onde:

- Δs é a distância percorrida pelo robô.
- $\Delta\theta$ é a mudança na orientação.
- Δx e Δy são as mudanças nas coordenadas x e y, respectivamente.
- r é o raio das rodas.
- d é a distância entre as rodas.
- $\Delta\phi_R$ e $\Delta\phi_L$ são as variações dos ângulos das rodas direita e esquerda, respectivamente.

Uma das ferramentas mais comuns para o cálculo de odometria em robótica é o *Robot Operating System* (ROS) [Macenski et al. 2022]. O ROS oferece pacotes específicos para odometria, como o `nav_msgs/Odometry`, que facilita a integração de dados de sensores e encoders, permitindo que o robô estime sua posição e orientação em tempo real.

O ROS utiliza tópicos e mensagens para trocar informações entre os diversos componentes do sistema robótico. O tópico `odom` é frequentemente utilizado para publicar dados de odometria. A mensagem `nav_msgs/Odometry` contém informações sobre a pose do robô (posição e orientação) e suas velocidades linear e angular.

Além disso, o simulador CoppeliaSim (anteriormente V-REP) também pode realizar cálculos de odometria em ambientes simulados. Ele oferece uma plataforma versátil para testar a navegação de robôs, permitindo o uso de sensores virtuais, como encoders e IMU, para estimar a posição e orientação do robô. Esses dados podem ser publicados em tópicos ROS, simulando a operação real do robô e permitindo uma validação robusta de algoritmos de odometria antes da implementação física.

2.9 Sistemas Microprocessados

2.9.1 Microprocessadores

Microprocessadores desempenham um papel crucial na robótica moderna, agindo como o cérebro dos robôs ao processar dados de entrada e saída rapidamente. Esses componentes são responsáveis por executar instruções de *software* que controlam e monitoram ações em tempo real, desde a navegação até a manipulação complexa de objetos. A capacidade dos microprocessadores de realizar múltiplas tarefas simultaneamente e com alta precisão é fundamental para o avanço da automação e da inteligência em sistemas robóticos. Os microprocessadores avançados permitem que robôs executem algoritmos sofisticados necessários para operações autônomas complexas [Lynch e Park 2017].

O microprocessador é o elemento que integra controladores, sensores e atuadores, garantindo que todas as partes do sistema robótico comuniquem-se de maneira eficiente e sincronizada. Essa integração é essencial para a coordenação de tarefas e para o funcionamento do robô, pois permite a execução de comandos precisos e a resposta adequada a estímulos ambientais, essenciais para a autonomia e a funcionalidade do robô em ambientes variados. Dentre os microprocessadores modernos, destacam-se a série Raspberry Pi e os computadores da linha *Jetson NVIDIA*.

O *Raspberry Pi* é uma série de computadores de placa única que revolucionou a educação em ciência da computação e se tornou um pilar nos projetos de prototipagem e sistemas embarcados. Lançado inicialmente em 2012, cada versão subsequente do *Raspberry Pi* tem visto melhorias significativas em termos de processamento e capacidades gráficas. Equipado com um processador ARM, o *Raspberry Pi* oferece um desempenho robusto suficiente para executar sistemas operacionais completos como o Linux, sendo ideal para aplicações que requerem conectividade de rede, processamento multimídia e automação residencial. Seu baixo custo e alta flexibilidade o tornam ideal para educação, prototipagem e projetos de robótica onde custo e acessibilidade são críticos.

O *NVIDIA Jetson Nano* é um computador de placa única especialmente projetado para aplicações de aprendizado de máquina e visão computacional. Parte da linha *Jetson*, que é focada em computação de borda e IA, o *Jetson Nano* equipa um processador *quad-core* ARM Cortex-A57 e uma Unidade de Processamento Gráfico (GPU) *NVIDIA Maxwell* com 128 núcleos CUDA. Este *hardware* é especificamente otimizado para tarefas de inferência de IA em tempo real, tornando-o uma escolha excepcional para projetos que envolvem robótica avançada, drones, e dispositivos IoT inteligentes que necessitam de processamento de dados complexos e análise visual no dispositivo. Além disso, o suporte da *NVIDIA* para o ecossistema *Jetson* com ferramentas de desenvolvimento e bibliotecas de aprendizado de máquina como o TensorRT e o cuDNN facilita significativamente o desenvolvimento de aplicações de IA.

Os pinos de propósito geral (*General Purpose Input/Output* - GPIOs) são pinos em microcontroladores e computadores de placa única que podem ser programados para operar como entradas ou saídas digitais. Eles são usados para interação direta com outros componentes eletrônicos e dispositivos externos, permitindo que o microcontrolador ou o computador envie sinais (*output*) ou receba sinais (*input*). As GPIOs são fundamentais em projetos de eletrônica e robótica porque permitem a implementação de funcionalidades como controlar motores, integrar sensores, e muito mais.

As GPIOs desempenham um papel crucial no desenvolvimento e operação de sistemas robóticos autônomos. Esses pinos programáveis em computadores de placa única como a *NVIDIA Jetson Nano* e microcontroladores são essenciais para a interação entre o *software* de controle e o *hardware* físico do robô, facilitando uma ampla gama de funcionalidades necessárias para a automação e a navegação autônoma.

As placas Jetson possuem múltiplos pinos GPIO com funcionalidades distintas. Os GPIOs digitais, configuráveis como entradas ou saídas, são utilizados para acionar dispositivos digitais, ou para ler o estado de interruptores e botões. Para o controle de movimento e ajuste de iluminação, os pinos de Modulação por Largura de Pulso (PWM) são essenciais, permitindo o controle preciso da velocidade dos motores. Além disso, a *Jetson Nano* facilita a comunicação entre dispositivos através de várias interfaces: os pinos I2C permitem a conexão de múltiplos dispositivos com apenas dois fios, ideal para sensores e periféricos. A interface UART é usada para comunicação serial com outros microcontroladores ou computadores. Detalhes técnicos sobre a interface UART é discutida no Capítulo 5 (Sistema de Navegação Autônoma).

2.9.2 SDK e Sistemas Operacionais

Um SDK (*Software Development Kit*) é um conjunto de ferramentas de software que facilita o desenvolvimento de aplicações em diversas plataformas, fornecendo componentes essenciais como bibliotecas, documentação e guias de implementação. Na robótica, o ROS (*Robot Operating System*) [Macenski et al. 2022] é um dos SDKs mais influentes, oferecendo uma coleção de ferramentas e bibliotecas projetadas para ajudar na criação de aplicações robóticas complexas. O ROS fornece funcionalidades que incluem controle de dispositivos de baixo nível, implementação de funcionalidades comumente usadas, troca de mensagens entre processos e gestão de pacotes, o que o torna uma ferramenta útil para muitos desenvolvedores robóticos.

Embora o ROS ofereça muitas vantagens, em nosso trabalho optamos por não utilizá-lo como uma ferramenta de controle de navegação direta. Em vez disso, desenvolveremos nosso próprio algoritmo de controle de navegação, buscando inovar e explorar metodologias que possam oferecer soluções mais adaptadas às necessidades específicas

dos projetos de navegação autônoma. No entanto, o ROS será utilizado de maneira indireta, especificamente para possibilitar o processo de comunicação robótica. Este processo será explicado no Capítulo 4 da tese.

2.10 Conclusões

Este capítulo apresentou diversos conceitos, algoritmos e tecnologias envolvendo robótica móvel, sensoriamento e sistemas de controle que possibilitam a implementação de sistemas navegação autônoma. Todos os sensores descritos neste capítulo são utilizados na metodologia de desenvolvimento desta tese. Além disso a definição de ferramentas clássicas, como o Filtro de Kalman e o levantamento do modelo matemático do robô móvel diferencial são essenciais para o funcionamento de módulos específicos na arquitetura de controle proposta, como por exemplo o método de Aceleração de Aprendizado.

No próximo capítulo serão descritos conceitos sobre Inteligência Artificial e Visão de Máquina. Estes conceitos contribuem para a implementação de sistemas de controle inteligente para o módulo de navegação autônoma. O controle inteligente pode ser aplicado de forma local, controlando alguns atuadores de forma reativa ou pode ser aplicado como um controlador deliberativo, planejando também a trajetória que o robô deve seguir.

CAPÍTULO 3

Técnicas de Inteligência Artificial

O Aprendizado por Reforço é uma das técnicas mais utilizadas para navegação envolvendo robôs móveis autônomos (AMRs), e diversos pesquisadores têm investigado novos algoritmos e melhorias nos métodos existentes [Kober, Bagnell e Peters 2013] [Zhu e Zhang 2021]. Por essa razão, este capítulo apresenta e aprofunda a fundamentação teórica voltada para o Aprendizado por Reforço. Inicialmente, é introduzido o conceito geral de Aprendizado de Máquina, seguido pela definição técnica das Redes Neurais Artificiais.

Além disso, exploramos técnicas de Aprendizado Supervisionado, uma vez que o módulo de segurança do robô utiliza um detector baseado em redes convolucionais treinado por aprendizado supervisionado para identificar obstáculos. Também é abordado o uso de Autoencoders para compressão de dados, fundamentais para viabilizar a transmissão eficiente dos dados de sensores pela rede LoRa. De forma complementar, discute-se também o Aprendizado por Reforço Multiagente (MARL), utilizado para coordenar múltiplos agentes, como robôs móveis em tarefas cooperativas, integrando aspectos de navegação autônoma e comunicação distribuída.

3.1 Aprendizado de Máquina

O Aprendizado de Máquina (*Machine Learning* - ML) é uma área da ciência da computação constituída por um conjunto de algoritmos que auxiliam o computador a aprender, ou automatizar, tarefas a partir de dados. Diferentemente da programação tradicional, onde os computadores são explicitamente programados para a execução de uma tarefa especifica, no ML o computador aprende as regras, ou padrões, envolvidos naquela determinada tarefa [Krohn, Beyleveld e Bassens 2019].

De acordo com [Geron 2019], o ML é aplicável em problemas onde as soluções existentes exigem muita configuração manual ou listas longas de regras e problemas complexos para os quais não existe uma boa solução quando utiliza-se uma abordagem de programação tradicional. O ML pode ser classificado de acordo com a quantidade e o tipo de supervisão empregada no treinamento, onde suas principais categorias são: i)

Aprendizado Supervisionado, ii) Aprendizado Não Supervisionado, iii) Aprendizado por Reforço.

No Aprendizado Supervisionado os dados de treinamento são fornecidos ao algoritmo. Estes dados incluem as soluções desejadas, ou seja, as respostas (*labels*) do problema para uma determinada entrada. A classificação e a regressão são tarefas típicas desta categoria de problema.

Classificação refere-se à atribuição de um rótulo específico para uma determinada entrada. Um exemplo clássico dessa abordagem é o fornecimento de imagens de animais para um algoritmo de ML, onde o algoritmo deve aprender a classificar a imagem com base em seu rótulo, por exemplo, identificar se a imagem representa um gato ou um cachorro, caracterizando um problema de duas classes. Por outro lado, a regressão busca prever um valor numérico para uma entrada específica. Um exemplo de regressão é a previsão do valor de um imóvel com base em atributos de entrada, como bairro, localização e número de quartos.

No Aprendizado Não Supervisionado, os dados não possuem rótulos, ou seja, não existe uma resposta definida para o problema. O sistema tenta, de forma autônoma, aprender os padrões presentes nos dados, sem supervisão. Geralmente, os algoritmos não supervisionados trabalham com agrupamentos (*clusters*), separando os dados em grupos com características semelhantes. Outra tarefa comum em aprendizado não supervisionado é a redução de dimensionalidade, cujo objetivo é simplificar os dados, mantendo as informações mais relevantes.

Um algoritmo de Aprendizado por Reforço visa capacitar um agente (quem está aprendendo) a adquirir conhecimento por meio de interações com o ambiente, sendo uma das técnicas que melhor imita a natureza da aprendizagem humana [Krohn, Beyleveld e Bassens 2019]. A qualidade das ações realizadas por esse agente é medida através de uma função de recompensa ou penalização, que orienta o agente a maximizar seus ganhos ao longo do tempo. No Aprendizado por Reforço, não existem dados rotulados (*labels*) fornecidos inicialmente. Todo o processo de aprendizagem depende da exploração do agente em um ambiente, seguindo uma abordagem de tentativa e erro.

A próxima seção apresenta o algoritmo versátil das Redes Neurais Artificiais, presentes em praticamente todas as técnicas de ML, incluindo o aprendizado por reforço.

3.2 Redes Neurais Artificiais

As Redes Neurais Artificias (RNA) são algoritmos versáteis presentes no Aprendizado Supervisionado, Não Supervisionado e Por Reforço. Elas estão na essência das técnicas modernas de aprendizado profundo (*Deep Learning* - DL). O DL permite o trabalho com dados altamente complexos, permitindo tarefas essenciais da indústria 4.0, tais

como: Reconhecimento de Imagens e Fala, Sistemas de Recomendação, Controle Automático e Navegação Autônoma.

Como o próprio nome diz, as Redes Neurais foram inspiradas no neurônio biológico humano. Inicialmente, o primeiro modelo de rede, denominado Perceptron, foi proposto por [Rosenblatt 1958]. Este modelo tinha sua aplicabilidade limitada a um separador linear que realizava operações binárias típicas de portas lógicas "OU"e "E".

Os autores [Minsky e Papert 1969] demonstraram que a limitação de classificação de uma Rede Neural do tipo Perceptron poderia ser contornada através do acréscimo de mais camadas no modelo, uma rede conhecida como Perceptron de Múltiplas Camadas (*MultiLayer Perceptron* - MLP). A rede MLP foi um marco na história da inteligência artificial, pois, estas redes, bem como seu algoritmo de treinamento deram origem as redes neurais modernas empregadas em DL. A Figura 3.1 apresenta uma rede MLP típica, contendo uma camada de entrada de dados, camadas de processamento (camadas ocultas), unidades denominadas de neurônios e uma camada de saída.

O treinamento de redes neurais ocorre através do algoritmo de retropropagação do erro denominado *backpropagation* [Rumelhart, Hinton e Williams 1986]. Este algoritmo consiste em apresentar um padrão à rede, calcular o erro entre a saída e este padrão, retropropagar o erro calculando de que forma as mudanças nos pesos w afetam o erro.

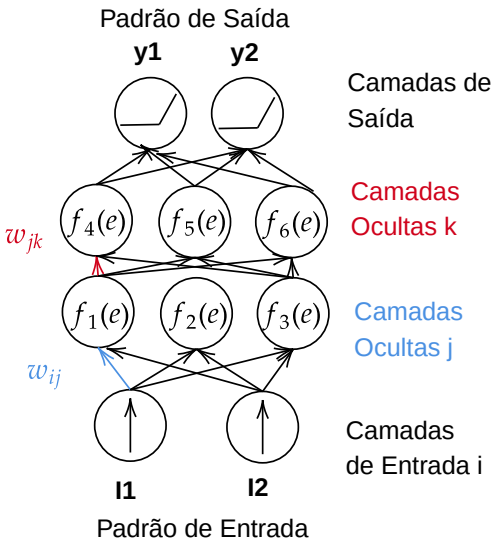


Figura 3.1: Rede Neural Multilayer Perceptron (MLP) - Autoria Própria.

3.2.1 Gradiente Descendente

O Gradiente Descendente é o principal algoritmo de otimização utilizado no treinamento de redes neurais, ajustando iterativamente os pesos para minimizar o erro

de previsão. No contexto de redes neurais, ele opera em cada iteração do treinamento ao recalcular o gradiente, aplicando-o para atualizar os pesos na direção de maior redução de erro. Essa atualização dos pesos pode ser expressa de forma geral como:

$$w_{ij} \leftarrow w_{ij} - \alpha \frac{\partial L}{\partial w_{ij}}, \tag{3-1}$$

onde L representa a função de perda da rede e α define a magnitude da mudança dos pesos em direção ao mínimo do erro. As funções de perda L são discutidas na seção 3.2.4.

Todas as camadas da rede neural apresentada na Figura 3.1 são totalmente conectadas; por essa razão, redes MLP também são conhecidas como Fully Connected Networks. Quando uma rede neural possui duas ou mais camadas ocultas, ela é chamada de rede neural profunda (*Deep Neural Network* - DNN) [Geron 2019]. No entanto, essa definição não é tão simplista. Com o aumento do número de camadas para resolver problemas complexos, como a detecção de objetos em imagens, surgem diversos desafios no treinamento, como o desaparecimento e a explosão do gradiente (*vanishing and exploding gradients*), o crescimento exacerbado de parâmetros treináveis e modelos que se tornam extremamente lentos e custosos em termos computacionais. A área de DNN lida com esses problemas em redes de alta complexidade, utilizando uma série de técnicas matemáticas e computacionais para tornar o treinamento viável.

3.2.2 Funções de Ativação

As funções de ativação desempenham um papel crucial nas Redes Neurais Artificiais, transformando a saída linear dos neurônios em valores que são usados para gerar previsões ou classificações. As funções de ativação aplicam uma transformação não linear à saída de cada neurônio, permitindo que a rede capture relações complexas nos dados.

Essas funções são posicionadas na transição entre as camadas de uma rede neural e aplicadas à saída de cada neurônio, introduzindo uma não-linearidade fundamental para o aprendizado. Elas aparecem obrigatoriamente na saída de cada camada oculta, e em algumas arquiteturas, na camada de saída também.

Neste trabalho, as principais funções de ativação utilizadas foram Sigmoide, Linear e ReLU. A seguir, são detalhadas suas características e propriedades.

A função Sigmoide é uma das funções de ativação clássicas em redes neurais e é dada pela expressão:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{3-2}$$

Essa função mapeia os valores de entrada para uma faixa contínua entre 0 e 1, o que é útil para modelos de classificação binária, onde as saídas podem ser interpretadas como probabilidades.

A função Linear é dada pela expressão:

$$f(x) = x \tag{3-3}$$

Essa função é usada principalmente nas camadas de saída para problemas de regressão, onde a saída pode ser um valor contínuo. A função linear preserva a escala dos dados e não adiciona nenhuma não-linearidade ao sistema.

A função ReLU é uma das mais populares em redes neurais profundas devido à sua simplicidade e eficiência. Ela é definida como:

$$f(x) = \max(0, x) \tag{3-4}$$

A função ReLU mapeia valores negativos para zero e mantém valores positivos inalterados.

A escolha da função de ativação é crucial para o desempenho da rede neural. No contexto deste trabalho, as funções Sigmoide e ReLU foram usadas em camadas ocultas para garantir que a rede possa modelar relações não lineares nos dados, enquanto a função Linear foi aplicada nas camadas de saída para tarefas de regressão. A ReLU é especialmente eficiente para redes profundas, enquanto a Sigmoide continua sendo útil para problemas de classificação binária. A Figura 3.2 dispõe as três funções de ativação tratadas nessa seção.

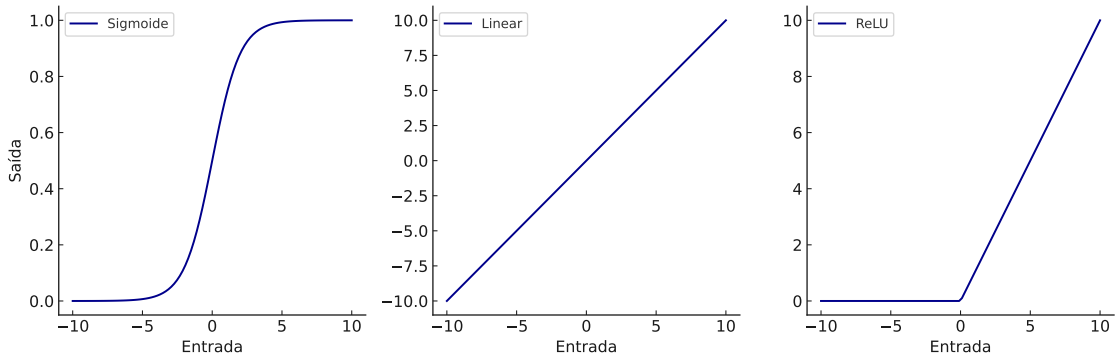


Figura 3.2: Funções de ativação Sigmoide, Linear e ReLU - Autoria Própria.

3.2.3 Treinamento e Implementação de Redes Neurais

A presente seção descreve os principais elementos para a implementação de redes neurais no contexto deste trabalho, abordando as funções de perda fundamentais e o processo de ajuste de pesos por meio do treinamento. Estes conceitos são aplicáveis tanto em aprendizado supervisionado quanto em aprendizado por reforço, e são essenciais para a implementação dos módulos de segurança e compressão de dados.

Funções de Perda

A função de perda é um componente essencial do treinamento de redes neurais, pois quantifica a diferença entre a previsão do modelo e o valor real, orientando o ajuste dos pesos para reduzir esta diferença ao longo do treinamento. As principais funções de perda utilizadas neste trabalho são:

- **Entropia Cruzada Binária** - Para tarefas de classificação binária, como a detecção de proximidade de pessoas, a entropia cruzada binária é usada para medir o quão próximas as previsões estão dos valores reais:

$$L_{CE} = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)], \tag{3-5}$$

  onde:

  - N é o número de amostras no conjunto de treinamento,
  - y_i é o rótulo verdadeiro da amostra i,
  - $\hat{y}_i$ é a previsão do modelo para a amostra i.

- **Erro Quadrático Médio (MSE)** - Para tarefas de compressão de dados, como no caso do autoencoder para compressão de dados sensoriais, o erro quadrático médio é apropriado para medir a diferença entre os dados originais e reconstruídos:

$$L_{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2, \tag{3-6}$$

  onde:

  - N é o número de amostras,
  - y_i saída (resposta original),
  - $\hat{y}_i$ saída (resposta predita) da rede.

Algoritmo de Treinamento de Redes Neurais

O treinamento de redes neurais segue um processo iterativo de ajuste de pesos, com base no cálculo do gradiente da função de perda em relação aos parâmetros da rede. O

Algoritmo 2 de treinamento geral utilizado para redes neurais supervisionadas é disposto abaixo:

Algorithm 2 Algoritmo de Treinamento de Redes Neurais

Entrada: Conjunto de dados de treinamento (X, Y), modelo f com parâmetros θ, função de custo L, taxa de aprendizado α
Saída: Modelo treinado com parâmetros θ^*
for *cada época* **do**
Forward Pass: Calcular as previsões $\hat{Y} = f(X; \theta)$
Cálculo da Perda: Calcular a perda $J = L(\hat{Y}, Y)$
Backward Pass: Calcular os gradientes $\nabla_\theta J$
Atualização dos Pesos: Atualizar os parâmetros $\theta = \theta - \alpha \nabla_\theta J$
Retornar modelo treinado f com parâmetros θ^*

A próxima seção introduz o campo da Visão de Máquina (*Machine Vision*), um subcampo importante da área de aprendizado profundo (DL). A Visão de Máquina é um dos principais recursos empregados no sistema de navegação proposto nesta tese, especialmente pelo uso de redes neurais convolucionais (CNN), que são essenciais para processar e interpretar dados visuais. Essas redes convolucionais permitem ao sistema detectar e analisar objetos no ambiente, proporcionando uma percepção visual robusta e confiável para a navegação autônoma.

3.3 Visão Computacional

A área de Visão Computacional (*Computer Vision*) é um campo de estudo da DL e foi inspirada na visão biológica humana. Esta área de estudo tem como objetivo realizar o reconhecimento de objetos e imagens para tomada de decisão. A Figura 3.3 adaptada de [Krohn, Beyleveld e Bassens 2019] ilustra a linha do tempo que apresenta momentos históricos do DL, abordagens clássicas de aprendizado de máquina bem como estudos sobre a visão humana.

Em 1959 os pesquisadores [Hubel e Wiesel 1962] da Universidade de Hopkins publicaram seus estudos sobre o processamento da informação visual no cérebro humano. Foram apresentadas várias imagens à gatos anestesiados, ao mesmo tempo a atividade do cortex primário visual, primeira parte do cérebro que recebe a entrada dos olhos, era analisada. Quando as imagens do projetor que o animal observada eram parcialmente modificadas, os neurônios do cortex apresentam uma atividade inicialmente anormal, estavam disparando.

Em experiências futuras descobriu-se que os neurônios que recebem estímulos visuais do olho são, em geral, mais responsivos a imagens com formato mais simples.

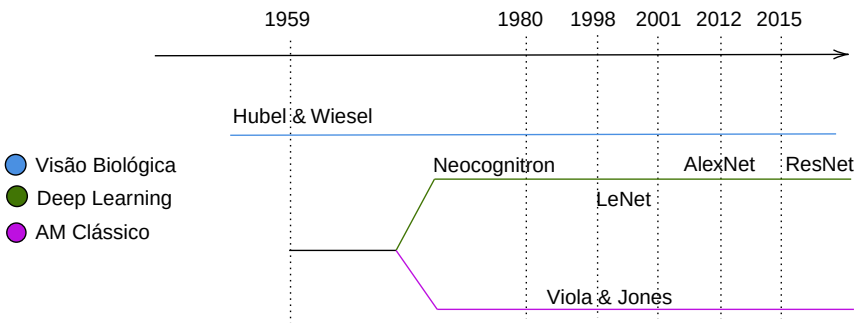


Figura 3.3: Linha do Tempo de Momentos Históricos no Campo da Visão de Máquina - Adaptado de [Krohn, Beyleveld e Bassens 2019].

Eles foram chamados de neurônios simples. Estes neurônios são especialistas em processar informações como orientação e borda de um determinado objeto, após processar estas informações mais simples, o estímulo é repassado a unidades mais complexas do cérebro. Em resumo, observou-se que a imagem no cérebro humano é processada em múltiplas camadas especialistas, iniciando no cortex primário. Os estudos de Hubel e Wisel inspiraram as evoluções técnicas que ocorreram no campo de *machine vision*.

Em 1980 o engenheiro eletricista Kunihiko Fukushima propôs uma arquitetura para *machine vision* baseada nos estudos de Hubel e Wiesel, denominada *neocognitron* [Fukushima e Miyake 1982]. Esta rede neural é capaz de reconhecer padrões em imagens, ela consiste em camadas auto-organizáveis, uma espécie de técnica não supervisionada, compostas por fotorreceptores (neuro receptores simples de Hubel e Wiesel) e camadas mais complexas. Esta arquitetura ficou conhecida no meio acadêmico por ser a primeira técnica que reconheceu dígitos escritos a mão com eficiência significativa, além disso inspirou outras arquiteturas de rede para processamento de imagens.

Foi em 1998 que as redes neurais dedicadas ao *machine vision* atingiram resultados ainda mais surpreendentes. A arquitetura LeNet-5 apresentada em [LeCun et al. 1989], ilustrada na Figura 3.4, foi projetada com unidades denominadas de blocos convolucionais que permitiram o processamento eficaz da imagem. Além disso a rede contou com recursos de DL que até hoje são amplamente utilizados em outras arquiteturas modernas, como por exemplo as funções de ativação sigmoidal. A rede de [LeCun et al. 1989] também ficou conhecida como rede convolucional (*Convolutional Neural Network* - CNN).

O componente mais importante de uma CNN é a camada convolucional, que tem este nome por operar matematicamente por meio do operador convolução sobre a

imagem de referência. Em uma camada convolucional os neurônios não estão totalmente conectados a cada pixel da imagem, eles se conectam apenas aos *pixels* que estão sobre o seu campo receptivo, chamado de *kernel*. A convolução é uma espécie de multiplicação que se move, deslisa, sobre a imagem. Esta particularidade permite que a rede se concentre em aprender características de baixo nível na primeira camada oculta da CNN. Nas camadas subsequentes da CNN outras características são processadas, no final todas estas características são reunidas [Geron 2019].

3.3.1 Fundamentos Matemáticos das Redes Convolucionais

A CNN opera diretamente sobre imagens representadas como matrizes de pixels. Uma imagem digital pode ser representada matematicamente como uma matriz I de dimensões $H \times W$, onde cada elemento $I(i,j)$ representa a intensidade de cor no ponto (i,j), com H sendo a altura e W a largura da imagem. No caso de uma imagem em escala de cinza, temos:

$$I = \begin{bmatrix} I(1,1) & I(1,2) & \cdots & I(1,W) \\ I(2,1) & I(2,2) & \cdots & I(2,W) \\ \vdots & \vdots & \ddots & \vdots \\ I(H,1) & I(H,2) & \cdots & I(H,W) \end{bmatrix}.$$

Quando a imagem é colorida (RGB), podemos representá-la como três matrizes correspondentes aos canais de cor vermelho, verde e azul.

A convolução é realizada ao aplicar um filtro K (ou *kernel*) de tamanho $m \times m$ que se move sobre a matriz de pixels da imagem. Esse filtro extrai características específicas, como bordas e texturas, ao multiplicar elemento a elemento e somar os resultados, conforme definido pela expressão:

$$S(i,j) = \sum_{u=0}^{m-1} \sum_{v=0}^{m-1} K(u,v) \cdot I(i+u, j+v), \tag{3-7}$$

onde: $S(i,j)$ é o valor do pixel na posição (i,j) da matriz resultante da convolução. $K(u,v)$ é o valor do filtro ou *kernel* na posição (u,v). $I(i+u, j+v)$ é o valor do pixel na posição $(i+u, j+v)$ da imagem original.

O filtro se move ao longo da imagem, multiplicando-se com o valor dos pixels em uma janela local e criando uma nova matriz que representa as características detectadas na imagem.

Após a convolução, a operação de pooling é usada para reduzir a dimensionalidade da representação da imagem, facilitando o treinamento e reduzindo a complexidade computacional da rede. O pooling mais comum é o *max pooling*, que seleciona o valor

máximo em uma região $p \times p$ da imagem convolucionada. A operação de pooling é dada por:

$$P(i,j) = \max_{0 \leq u < p, 0 \leq v < p} S(i \cdot p + u, j \cdot p + v), \tag{3-8}$$

onde $P(i,j)$ é o valor resultante do pooling na posição (i,j). $S(i \cdot p + u, j \cdot p + v)$ é o valor do pixel da imagem convolucionada na posição $(i \cdot p + u, j \cdot p + v)$.

Após a convolução e o pooling, é comum aplicar uma função de ativação para introduzir não linearidade ao modelo. A função de ativação mais utilizada em CNNs é a ReLU. A função ReLU ajuda a rede a aprender representações complexas dos dados, mantendo valores positivos e zerando valores negativos, o que acelera o treinamento e melhora a convergência.

3.3.2 Arquiteturas Típicas

Na rede LeNet-5 [LeCun et al. 1989], ilustrada na Figura 3.4, é possível verificar a entrada de uma imagem monocromática, apenas 1 canal, de comprimento 32×32 pixels.

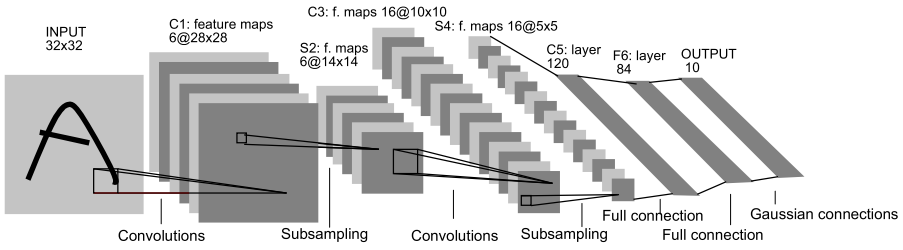


Figura 3.4: Rede Convolucional LeNet-5 - Retirado de [LeCun et al. 1989]

Inicialmente a rede conta com uma camada convolucional composta por 6 mapas de características com *kernels* 5×5 o que resulta em uma redução da imagem original para 28×28 pixels. A camada S2 possui 6 mapas com *kernels* 2×2 que resulta em uma redução da imagem para 14×14 pixels. Estas reduções (*subsampling*) ocorrem até a camada S4, que possui 16 mapas com *kernels* 2×2. Ao final todas estas características são conetadas em três redes *Fully Connected* com respectivamente 120, 84 e 10 neurônios.

A camada de saída da LeNet-5, bem como de diversas arquiteturas CNN, é especial. Como o problema inicialmente tratado por [LeCun et al. 1989] era de classificação de dígitos escritos a mão, esta camada de saída mede a probabilidade daquela saída pertencer a uma determinada classe. Vale destacar que foram consideradas 10 classes (10 dígitos por exemplo) e por isso a saída contém 10 neurônios. A função custo, denominada

entropia cruzada, é utilizada para esta finalidade, ela penaliza as previsões ruins, gerando gradientes maiores e acelerando a convergência da rede neural.

Outras arquiteturas dedicadas ao *machine vision* surgiram na literatura, muitas delas em função do concurso *Imagenet Large Scale Visual Recognition Challenge* (ILSVRC). O ILSVRC conta um banco de dados representado por centenas e milhares de imagens. O projeto tem sido fundamental para o avanço da visão computacional e da pesquisa em aprendizado profundo. Uma famosa arquitetura vencedora do concurso foi a proposta por [He et al. 2015], denominada ResNet50.

3.3.3 Arquitetura de Redes Residuais (ResNet)

A arquitetura ResNet, ilustrada na Figura 3.5, é uma rede neural convolucional (CNN) profunda composta, originalmente, por 50 camadas, organizadas em blocos residuais. Esta arquitetura foi projetada para resolver problemas relacionados ao treinamento de redes muito profundas, como o desaparecimento ou explosão de gradientes, por meio da introdução de conexões residuais que permitem uma aprendizagem mais eficiente e estável.

A rede inicia com uma camada convolucional de tamanho 7×7, seguida por uma normalização em lote (*Batch Normalization*) e uma operação de *Max Pooling* 3×3, que reduz a dimensionalidade da imagem de entrada. Após esse estágio de pré-processamento, têm início os blocos residuais.

Os blocos residuais são os componentes centrais da ResNet. Cada bloco recebe uma entrada $\mathbf{x}$ e é responsável por aprender um mapeamento residual $\mathcal{F}(\mathbf{x})$, geralmente composto por operações convolucionais e funções de ativação não lineares. A grande inovação introduzida é a *conexão de atalho* (*shortcut connection*), que realiza uma soma direta entre a entrada original $\mathbf{x}$ e a saída do mapeamento residual $\mathcal{F}(\mathbf{x})$, resultando na saída $\mathbf{y} = \mathcal{F}(\mathbf{x}) + \mathbf{x}$. Esta estrutura facilita o fluxo de gradientes durante o treinamento, permitindo que redes muito profundas sejam otimizadas de forma eficaz.

Isso permite que o modelo aprenda não apenas o mapeamento original, mas também um mapeamento residual, o que facilita a propagação do gradiente durante o treinamento Se o mapeamento ideal for a identidade, o bloco residual apenas aprende a zerar os pesos, preservando a informação original da entrada. Com isso, os blocos residuais permitem a construção de redes muito mais profundas e eficazes para tarefas complexas como reconhecimento de imagens e navegação autônoma [Zhang et al. 2021].

Ao final da rede, uma camada de *Global Average Pooling* é utilizada para reduzir ainda mais a dimensionalidade, transformando as características aprendidas em vetores prontos para a classificação. A saída final da rede é uma camada totalmente

conectada (*Fully Connected* - FC), responsável por gerar a predição final, que pode ser uma classificação ou uma detecção de objetos, dependendo da tarefa.

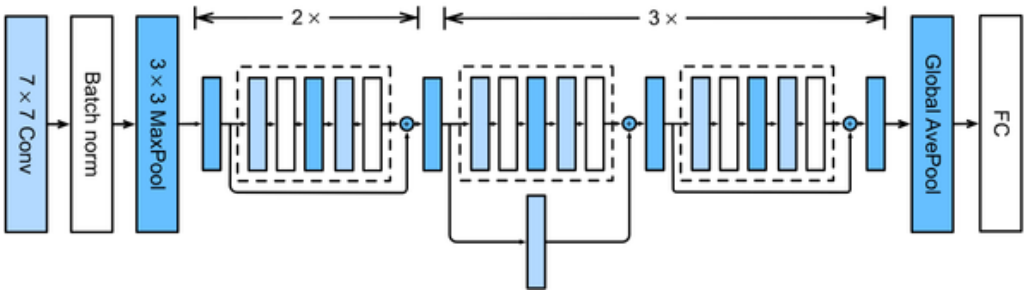


Figura 3.5: Arquitetura da ResNet, retirada de [Zhang et al. 2021]

As redes convolucionais são essenciais para a detecção de padrões visuais e objetos em uma cena, sendo amplamente utilizadas para a percepção em sistemas de robótica. No entanto, além da percepção visual, robôs móveis autônomos também precisam aprender a tomar decisões baseadas nas interações com o ambiente, o que exige algoritmos que vão além da simples classificação Nesse contexto, entra o Aprendizado por Reforço. A próxima seção introduz os conceitos básicos sobre técnicas de aprendizado por reforço.

3.4 Aprendizado Por Reforço

Para que um AMR execute sua tarefa proposta, como navegar em uma pista, sobrevoar um determinado ambiente, resgatar um objeto, entre outras, é necessário um algoritmo de controle, onde o aprendizado por reforço possa ser uma solução.

Basicamente, o aprendizado por reforço é composto pelos seguintes elementos: i) agente, ii) estado atual, iii) ambiente, iv) sinal de reforço e v) estado futuro. O agente atua no ambiente em um estado inicial, sua ação no ambiente gera o sinal de recompensa e ocorre uma transição de estado. Esses estados e suas transições, de acordo com [Krohn, Beyleveld e Bassens 2019], podem ser modelados matematicamente usando cadeias de Markov. O objetivo é maximizar a função de recompensa usando os sinais de *feedback* fornecidos pelo ambiente. Este problema de otimização pode ser decomposto e resolvido usando a equação de Bellman, conhecida como método das Diferenças Temporais (TD) descrito na equação abaixo [Krohn, Beyleveld e Bassens 2019].

$$Q^{\pi}(s,a) = r + \gamma \max Q^{\pi}(s',a') \tag{3-9}$$

onde r é a recompensa imediata obtida no ambiente, γ o fator de desconto, s e a um par de estado de ação atual, respectivamente. s' e a' são um par de estado futuro e ação futura.

A função $Q(s,a)$ é uma função de valor, que avalia a qualidade de um par de estados e ações realizados pelo agente no ambiente. De acordo com a Equação 3-8, é possível observar que o valor de um par de estados e ações (s,a) é medido pela recompensa imediata obtida com aquela ação no ambiente, bem como pela recompensa futura que um determinado estado futuro (s') fornecerá.

3.4.1 Deep Q-Networks

Uma vez que $Q(s,a)$ é uma função e as redes neurais são excelentes aproximantes universais, esse recurso é usado para aproximar essa função ótima Q por meio do algoritmo de diferenças temporais. Assim, no algoritmo DQN (*Deep Q-Networks*), a rede neural profunda é utilizada em uma estratégia de regressão. No algoritmo *Q-learning*, um método de RL tradicional e clássico, a função Q é mapeada através de uma tabela de estados visitados (método tabular). Se a representação do ambiente permitir vários estados, a representação desses estados através de uma matriz, ou tabela, torna-se inviável. Uma vez que a rede neural aprende diretamente a função Q, o DQN se torna mais atraente em ambientes com um grande número de estados possíveis. A aproximação de uma rede neural $\hat{Q}$ é dada por:

$$Q(s,a) \approx \hat{Q}(s,a,\theta) \tag{3-10}$$

onde $Q(s,a)$ é a função de valor estimada por meio da Equação de Bellman e $\hat{Q}(s,a,\theta)$ é a função aproximada pelos pesos sinápticos θ, ou seja, a saída da rede neural.

A rede neural do DQN normalmente utiliza em sua saída uma função de ativação linear, pois o problema é basicamente uma regressão. A função de ativação das camadas intermediárias da rede normalmente é a ReLu [Krohn, Beyleveld e Bassens 2019], uma função frequentemente utilizada em camadas intermediárias de DL, pois evita problemas como explosão ou desaparecimento do gradiente. O diagrama de blocos ilustrado na Figura 3.6 apresenta uma rede neural DQN, onde suas entradas referem-se à medição de estados no ambiente e sua saída refere-se ao número de ações possíveis. Já os passos do DQN são detalhados no Algoritmo 3.

Os estados da rede DQN são medidos através dos sensores do AMR, que por sua vez é denominado "agente". Estes estados são processados por uma rede neural. Caso estes estados sejam imagens, por exemplo obtidas por uma câmera RGB, este processamento pode ocorrer por meio de uma Rede Neural Convolucional (CNN).

A saída da rede $(Q_1, Q_2, \ldots, Q_n)$ possui número de neurônios igual ao número de ações possíveis, sendo que cada ação é avaliada por uma função Q. O agente tende a selecionar a ação que maximiza o valor de Q, segundo uma política denominada *gulosa* (*greedy*). O AMR executa no ambiente, por meio de seus atuadores, a ação definida pela

rede neural. A função custo (*loss function*) utilizada é o Erro Quadrático Médio (*Mean Squared Error* – MSE) entre o valor-alvo y e o valor estimado pela rede:

$$L(\theta) = \mathbb{E}\left[(y - \hat{Q}(s,a,\theta))^2\right], \tag{3-11}$$

sendo o valor-alvo definido por:

$$y = r + \gamma \max_{a'} \hat{Q}(s',a',\theta^-), \tag{3-12}$$

onde θ^- representa os pesos da *rede-alvo* (*target network*), mantida fixa por um intervalo de tempo para garantir estabilidade no treinamento.

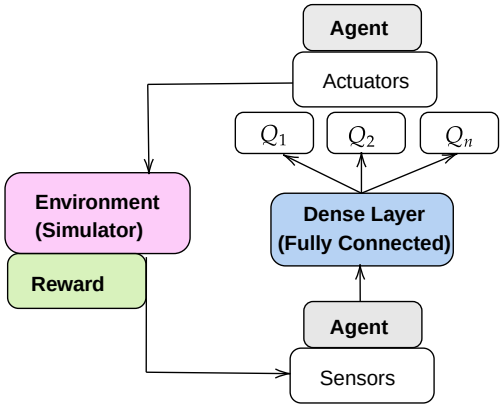


Figura 3.6: Diagrama de Blocos de uma Rede DQN - Autoria Própria

Algorithm 3 Algoritmo DQN

Inicializar o **agente** DQN; Definir número de *EPISODIOS*;
Leitura dos Sensores (inicializa s)
while *done* = *False* **do**
  `/* Estado Terminal = False */`
  Gera uma Probabilidade P
  if $\epsilon > P$ **then**
    Seleciona uma ação a aleatória;
  if $\epsilon < P$ **then**
    $a = \text{argmax}(\hat{Q}(s_k, a, \theta)$
  Executa a Ação a no Ambiente e Observa o novo estado s_{k+1} e a Recompensa R
  Memorização (s_k, a, R, s_{k+1}, *done*) `/* experience replay */`
  $Q(s,a) = R + \gamma \max Q(s_{k+1}, a)$ `/* Estima o Alvo */`
  do $s = s_{k+1}$
performa o MSE e o gradiente descendente $(Q - \hat{Q}(s_{k+1}, a, \theta)^2)$
Decai a probabilidade de exploração ϵ

O procedimento de aprendizagem das redes DQN ocorre com o uso da memória de experiência (*Memory Replay* ou *Experience Replay*). Este procedimento consiste em adquirir amostras aleatórias do tamanho do *mini-batch*, ou seja, tamanho do lote de dados utilizados para treinar cada época da rede neural. Estas amostras correspondem as experiências obtidas pelo agente durante o processo exploratório. O processo exploratório nada mais é que a probabilidade em aplicar uma ação aleatória determinar uma ação, também aleatória no ambiente. Esta probabilidade vai decaindo ao longo das épocas de treinamento (probabilidade de exploração ϵ). A memória de aprendizagem é computacionalmente representada por um *deque* (ou fila) contendo os estados, ações e recompensas que são atualizados a cada nova experiência. Amostrar um pequeno subconjunto de memórias no largo conjunto de experiências torna o processo de aprendizagem mais eficiente. Por esta razão o *deque* utilizado é normalmente grande.

3.4.2 Double Deep Q-Networks

O método Double DQN (DDQN), proposto por [Hasselt, Guez e Silver 2016], é uma variação do algoritmo DQN tradicional, projetado para reduzir a superestimação dos valores de ação $Q(s,a)$ durante o treinamento. A superestimação ocorre no DQN clássico devido à aplicação do operador max tanto para seleção quanto para avaliação da ação na política de aprendizado, o que pode induzir o agente a estimar erroneamente ações como mais vantajosas do que realmente são, especialmente nas fases iniciais de exploração.

Para mitigar esse problema, o DDQN utiliza duas redes neurais distintas: a rede principal $Q^{\pi}(s,a;\theta)$, responsável por selecionar a ação que maximiza o valor de Q, e a rede-alvo $Q_{\text{target}}(s,a;\theta^{-})$, usada exclusivamente para avaliar essa ação. O valor-alvo y é, portanto, calculado por:

$$y = r + \gamma\, Q_{\text{target}}\left(s', \arg\max_{a'} Q^{\pi}(s',a';\theta);\theta^{-}\right), \tag{3-13}$$

onde θ representa os pesos da rede principal, e θ^{-} os pesos da rede-alvo, que são atualizados periodicamente para garantir estabilidade no treinamento.

A saída da rede $Q^{\pi}(s,a;\theta)$ é composta por um vetor de valores $Q_1, Q_2, \ldots, Q_n$, correspondentes às n ações possíveis. A ação selecionada é aquela que maximiza o valor de Q, segundo uma política *greedy*. A função de custo utilizada para treinar a rede principal é o Erro Quadrático Médio (MSE) entre o valor-alvo y e a estimativa da rede:

$$L(\theta) = \mathbb{E}\left[(y - Q^{\pi}(s,a;\theta))^2\right], \tag{3-14}$$

proporcionando uma atualização mais confiável e menos sujeita à superestimação, o que resulta em políticas mais estáveis e eficazes. A Figura 3.7 ilustra estas redes neurais

através de um diagrama de blocos, apresentando também os passos de treinamento da rede DDQN. É possível verificar neste diagrama as duas redes neurais independentes, a rede $Q^{\pi}(s', a')$ (azul) e a rede Q_{target} (roxo).

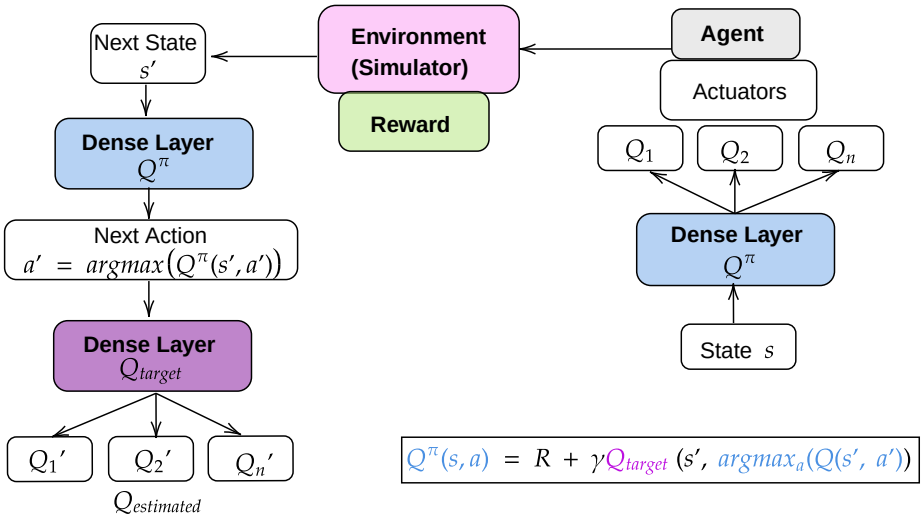


Figura 3.7: Diagrama de Blocos de uma Rede Double DQN - Autoria Própria

Onde as aplicações de redes DQN apresentam alta variabilidade na qualidade da política estimada pela rede, isto é, em certos episódios dos treinamentos o agente apresenta política com recompensas altas e em outros episódios políticas com recompensas baixas, é indicado o uso da rede DDQN. Normalmente este algoritmo apresenta uma maior estabilidade da curva de aprendizagem, o processo de evolução das recompensas, quando comparado ao DQN.

A próxima seção descreve técnicas e métodos de como transferir o aprendizado obtido pelos algoritmos citados para robôs e sistemas reais.

3.4.3 Transferência de Aprendizado (*Sim to Real*)

O conceito de transferência *sim-to-real* ganhou destaque devido aos desafios de implantar modelos de aprendizado por reforço profundo (DRL) no mundo real da robótica [Zhu et al. 2023] [Zhao, Queralta e Westerlund 2020]. Essa transferência é crucial porque, embora os ambientes de simulação ofereçam um meio controlado, seguro e econômico para treinar sistemas robóticos, muitas vezes eles não representam com precisão as dinâmicas do mundo real.

Os autores [Zhao, Queralta e Westerlund 2020] discutem várias metodologias destinadas a diminuir essa lacuna, garantindo que o conhecimento adquirido em ambientes virtuais se traduza efetivamente o sucesso operacional em configurações físicas.

Algorithm 4 Algoritmo DDQN

Inicializar a **rede de avaliação** Q e a **rede alvo** $\hat{Q}$ com pesos iguais; Definir número de $EPISODIOS$;
Leitura dos Sensores (inicializa s)
while *done* = *False* **do**
  /* Estado Terminal = False */
  Gera uma Probabilidade P
  if $\epsilon > P$ **then**
    Seleciona uma ação a aleatória;
  else
    $a = \arg\max_a Q(s,a,\theta)$ /* Seleção de ação usando a rede de avaliação */
  Executa a Ação a no Ambiente e Observa o novo estado s' e a Recompensa R
  Memorização (s, a, R, s', $done$) /* experience replay */
  /* Atualização dos pesos da rede de avaliação */
  $Q(s,a) \leftarrow Q(s,a) + \alpha\left(R + \gamma\hat{Q}(s', \arg\max_a Q(s',a,\theta)) - Q(s,a)\right)$ /* DDQN Update */
  A cada C passos, atualize $\hat{Q}$ para os pesos de Q: $\hat{Q} \leftarrow Q$
  do $s = s'$

Vários métodos foram desenvolvidos para aprimorar a fidelidade da transferência *sim-to-real*. A randomização de domínio descrita por [Zhu et al. 2023], por exemplo, introduz variabilidade nos parâmetros de simulação (como iluminação e texturas), o que ajuda os modelos a generalizar melhor para condições do mundo real [Tobin et al. 2017]. A Figura 3.8 ilustra uma randomização de domínio por meio uma pista de treinamento quadrada, com delimitação da pista um robô móvel. O cenário pode apresentar diferentes cores e texturas. Com a randomização de domínio, cria-se uma variedade de cenários de simulação, com propriedades aleatórias. É provável que este modelo possa se adaptar ao ambiente do mundo real, já que se espera que o sistema real seja uma amostra nessa rica distribuição de variações de treinamento.

Outra abordagem, a identificação de sistemas, envolve o desenvolvimento de modelos matemáticos precisos que imitam mais de perto as dinâmicas do mundo real, melhorando assim o realismo do simulador. Técnicas de adaptação de domínio também são empregadas para alinhar os recursos aprendidos nas simulações com aqueles encontrados em ambientes reais.

As aplicações práticas dessas metodologias abrangem vários aspectos da robótica, desde tarefas de navegação até manipulações complexas envolvendo múltiplos agentes ou braços robóticos. Cada aplicação apresenta desafios únicos e muitas vezes requer uma abordagem sob medida para a transferência *sim-to-real*, dependendo dos requisitos específicos da tarefa e do ambiente em que o robô opera.

Os ambientes reais são significativamente mais complexos e imprevisíveis do que as simulações. Na robótica móvel, isso inclui variáveis como terreno irregular,

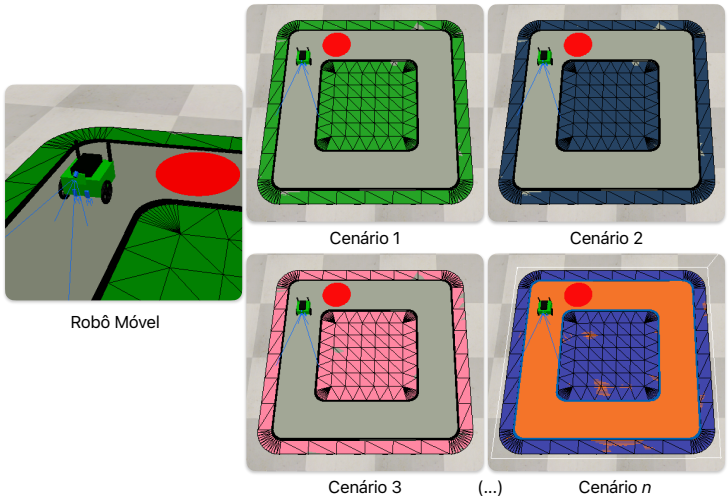


Figura 3.8: Exemplo de randomização de domínio: diferentes cenários - Autoria Própria.

obstáculos dinâmicos, mudanças de iluminação, e interações humanas, que podem não ser completamente modeladas em um ambiente simulado. Assim, estratégias de sim-to-real devem focar na generalização desses elementos para garantir que os robôs possam operar de forma segura e eficaz fora do simulador.

Os sensores usados em robôs móveis, como LIDAR, câmeras e sensores ultrassônicos, podem ter desempenhos distintos em ambientes simulados em comparação com o mundo real devido a diferenças na precisão do sensor, ruído, e respostas a condições ambientais. As técnicas descritas por [Zhu et al. 2023], como a randomização de domínio e adaptação de domínio, podem ser usadas para treinar os sistemas de percepção dos robôs para se adaptarem e generalizarem a partir de dados simulados para dados reais.

3.5 Rede Neural Autoencoder

Os autoencoders são uma classe específica de redes neurais projetada para aprender uma representação comprimida e significativa dos dados de entrada e, em seguida, reconstruí-los da forma mais precisa possível [Bank, Koenigstein e Giryes 2021]. Essa capacidade os torna úteis para tarefas como compressão de dados, remoção de ruído, redução de dimensionalidade e detecção de anomalias. A principal ideia por trás dos autoencoders é que eles codificam os dados de entrada em um espaço de dimensionalidade reduzida (espaço latente) e depois decodificam essa representação para recuperá-los.

3.5.1 Fundamentos dos Autoencoders

Os autoencoders consistem em duas partes principais: o *encoder* e o *decoder*. O *encoder*, denotado como $A : \mathbb{R}^n \to \mathbb{R}^p$, comprime os dados de entrada em uma representação de dimensionalidade reduzida. O *decoder*, $B : \mathbb{R}^p \to \mathbb{R}^n$, reconstrói os dados originais a partir dessa representação comprimida. A Figura 3.9, adaptada de [Bank, Koenigstein e Giryes 2021], apresenta um modelo de rede Autoencoder composta pelo Encoder e Decoder.

Figura 3.9: Arquitetura da Rede Autoencoder

O objetivo do treinamento de um autoencoder é minimizar o erro de reconstrução, que mede o quão próximo os dados reconstruídos estão dos dados de entrada originais. Formalmente, a função objetivo é:

$$\arg\min_{A,B} \mathbb{E}\left[\Delta(x, B \circ A(x))\right],$$

onde Δ é uma função de perda de reconstrução, tipicamente a norma ℓ_2, que representa a distância entre os dados de entrada x e sua reconstrução pelo *decoder*, $B(A(x))$.

No caso especial em que tanto A quanto B são operações lineares, o autoencoder funciona de maneira semelhante à Análise de Componentes Principais (PCA). No entanto, autoencoders podem ir além de transformações lineares, permitindo capturar estruturas não lineares nos dados.

3.5.2 Regularização em Autoencoders

Muitas vezes, autoencoders necessitam de técnicas de regularização para evitar que aprendam simplesmente a função identidade. A regularização garante que a representação comprimida seja significativa e ajuda a evitar o *overfitting*. Algumas técnicas comuns de regularização incluem:

- **Autoencoders Esparsos**: Impõem restrições de esparsidade nas ativações nas camadas ocultas, forçando a rede a usar menos neurônios para representar os dados de entrada.
- **Autoencoders Denoising**: Adicionam ruído aos dados de entrada e treinam a rede para reconstruir os dados originais, sem ruído, incentivando o aprendizado de características robustas.
- **Autoencoders Contrativos**: Regularizam o modelo penalizando a sensibilidade da representação latente a pequenas perturbações na entrada. Isso é feito minimizando o Jacobiano do *encoder*, tornando a representação latente menos sensível a pequenas variações na entrada.

3.5.3 Aplicações dos Autoencoders

Os autoencoders têm uma ampla gama de aplicações, incluindo:

- **Compressão de Dados**: Ao aprender uma representação comprimida dos dados de entrada, os autoencoders podem reduzir a dimensionalidade dos dados preservando suas características mais importantes.
- **Remoção de Ruído**: Os autoencoders *denoising* são usados para limpar dados de entrada ruidosos, como imagens ou sinais, reconstruindo os dados originais sem ruído.
- **Detecção de Anomalias**: Ao aprender os padrões normais dos dados, os autoencoders podem detectar anomalias, já que estas tendem a apresentar maiores erros de reconstrução.

3.6 Aprendizado por Reforço Multiagente

O *Multi-Agent Reinforcement Learning* (MARL) é a extensão natural do Aprendizado por Reforço (*Reinforcement Learning* - RL) para cenários em que múltiplos agentes interagem em um mesmo ambiente compartilhado, buscando atingir objetivos individuais ou coletivos. Ao contrário do RL tradicional, no qual o ambiente é considerado estacionário do ponto de vista do agente único, no MARL a evolução simultânea das políticas dos demais agentes torna o ambiente não-estacionário, aumentando a complexidade do aprendizado [Huh e Mohapatra 2024]. Essa característica exige mecanismos adicionais de coordenação e comunicação multiagente para que as políticas aprendidas sejam estáveis e eficientes.

Formalmente, o MARL pode ser modelado como um jogo estocástico ou um processo de decisão de Markov descentralizado (Dec-MDP), definido por um conjunto de agentes N, um espaço de estados globais S, um espaço de ações conjuntas $A =$

$A_1 \times \cdots \times A_N$, uma função de recompensa r (individual ou compartilhada) e uma função de transição T. Em ambientes descentralizados, cada agente observa apenas uma parte do estado global, podendo comunicar-se com outros agentes de forma limitada. Essa descentralização pode ser natural, por restrições físicas como alcance de comunicação, ou artificial, imposta para reduzir o custo computacional ou de rede [Huh e Mohapatra 2024].

A natureza das interações entre agentes em MARL pode ser classificada em três categorias principais: *cooperativa*, quando todos compartilham o mesmo objetivo e otimizam uma recompensa global; *competitiva* (ou adversarial), quando os objetivos são conflitantes; e *mista*, combinando cooperação e competição em diferentes momentos ou aspectos da tarefa. Essa classificação influencia diretamente o desenho das funções de recompensa, dos mecanismos de coordenação e das estratégias de comunicação entre os agentes [Yi et al. 2022].

Do ponto de vista das abordagens de treinamento, destacam-se: i) *aprendizado independente*, no qual cada agente trata os demais como parte do ambiente; ii) treinamento centralizado com execução descentralizada (*Centralized Training with Decentralized Execution* — CTDE), em que informações globais são usadas no treinamento, mas cada agente atua com base apenas nas suas observações locais; e iii) *treinamento totalmente descentralizado*, que depende de comunicações limitadas e é mais escalável, porém mais desafiador [Huh e Mohapatra 2024]. Em muitos cenários, especialmente na robótica e em redes de comunicação, o sucesso depende do projeto de protocolos de comunicação que sejam adaptativos e eficientes.

Em aplicações robóticas, o MARL permite a coordenação de múltiplos robôs móveis para execução de tarefas cooperativas, como exploração de ambientes, transporte de cargas e navegação em formações. Já em redes de comunicação, notadamente no contexto *Vehicle-to-Everything* (V2X), o MARL é aplicado para otimizar a alocação de recursos, *caching* de conteúdo, *data offloading* e controle de potência, considerando restrições de latência e confiabilidade, como nas comunicações URLLC [Althamary, Huang e Lin 2019]. Nessas aplicações, os agentes aprendem a tomar decisões locais que contribuem para o desempenho global, mesmo em ambientes altamente dinâmicos e com conectividade intermitente.

3.7 Conclusões

Este capítulo apresentou os fundamentos teóricos que sustentam o desenvolvimento dos algoritmos de navegação propostos nesta tese. Foram abordados os principais conceitos relacionados às redes neurais profundas, com ênfase nas redes convolucionais (CNNs), amplamente utilizadas para extração de características visuais a partir de imagens RGB-D. Além disso, discutiram-se em detalhes os métodos de aprendizado por re-

força profundo, como o DQN e o DDQN, essenciais para a tomada de decisão autônoma no sistema proposto. Tais abordagens constituem o núcleo da arquitetura de navegação, sendo aplicadas tanto na escolha das ações do robô quanto na fusão sensorial. Por fim, também foram introduzidos os conceitos de aprendizado supervisionado, empregados no desenvolvimento do módulo de segurança e algoritmo de compressão por autoencoder. Todos esses elementos, em conjunto, compõem a base computacional de inteligência artificial utilizada ao longo deste trabalho. O próximo capítulo dispõe os conceitos sobre redes de comunicação sem fio em sistemas móveis autônomos.

CAPÍTULO 4

Internet das Coisas Robóticas e Conectividade

Este capítulo apresenta a fundamentação teórica das redes de comunicação sem fio aplicadas à robôs móveis autônomos. Inicialmente, são introduzidos os conceitos de Internet das Coisas Robóticas (*Internet of Robotic Things* - IoRT), destacando sua importância e aplicações, especialmente em navegação autônoma. Em seguida, discute-se a comunicação baseada no *middleware Data Distribution Service* (DDS), amplamente utilizado em sistemas robóticos que operam no *Robot Operating System* (ROS). Posteriormente, é apresentada a tecnologia de comunicação selecionada para as tarefas de navegação robótica em longa distância: o protocolo LoRaWAN. Por fim, são explorados os conceitos relacionados ao controle de parâmetros de modulação no protocolo LoRa, com foco nos fatores de espalhamento espectral, além de técnicas de compressão de dados que viabilizam a comunicação eficiente em banda estreita.

4.1 Comunicação Robótica com as Coisas

Os sistemas Ciberfísicos (*Cyber-Physical Systems* - CPS) são *hardwares* genéricos compostos por sensores, microcontroladores entre outros dispositivos eletrônicos e computacionais que auxiliam a integração de processos físicos com o mundo digital [Andreja 2017]. Existem CPS presentes em diversos setores da economia: como saúde, energia, agricultura, logística e manufatura industrial. A principal tecnologia que conecta os CPS no contexto da Indústria 4.0 é a Internet das Coisas (*Internet of Things* - IoT) [Pflaum e Gölzer 2018]. A IoT é a infraestrutura de rede baseada em protocolos de comunicação, que permite e possibilita a comunicação entre os CPS.

A integração entre IoT e CPS tem desempenhado um papel crucial na viabilização de avanços significativos na Indústria 4.0, particularmente em cenários que demandam automação e conectividade em tempo real. A IoT fornece a base tecnológica necessária para que dispositivos e sistemas CPS se comuniquem e compartilhem dados instantaneamente, possibilitando o monitoramento, o controle e a análise de processos físicos de maneira mais segura e eficaz [Pflaum e Gölzer 2018].

Um dos desdobramentos dessa integração pode ser observado em aplicações de robótica autônoma, especialmente em ambientes industriais inteligentes, onde a comunicação eficiente entre robôs móveis é um fator crítico para a navegação em cenários dinâmicos e complexos. Em configurações com múltiplos robôs, a troca de informações entre agentes permite melhorar a eficiência da exploração, a tomada de decisões e a execução de tarefas colaborativas [Jawhar et al. 2018] [Rusu et al. 2016].

Dentro desse contexto, destaca-se a abordagem de controle descentralizado, na qual não há um agente central que coordena o comportamento dos demais. Cada robô atua de forma autônoma no ambiente, tomando decisões com base em suas próprias observações e experiências locais [Jiang, Su e Lu 2024]. Para viabilizar a cooperação mesmo sem um centro de controle, é comum empregar técnicas de comunicação distribuída, que permitem o compartilhamento de informações entre os agentes. Essa comunicação, contudo, impõe desafios relevantes relacionados à Qualidade de Serviço (QoS), incluindo latência, confiabilidade da transmissão de dados e eficiência na coordenação das ações. Abordagens recentes têm explorado o uso de aprendizado por reforço multiagente, permitindo que os robôs aprendam a partir de suas próprias experiências e das interações com os demais, otimizando a navegação e a cooperação em sistemas descentralizados [Yi et al. 2022].

Essa necessidade de comunicação distribuída e tomada de decisão autônoma entre robôs leva ao surgimento de arquiteturas mais avançadas, nas quais a conectividade não se limita à troca de mensagens, mas envolve também o uso de inteligência embarcada e distribuída. Dentro desse panorama, destaca-se a Internet das Coisas Robóticas (*Internet of Robotic Things* – IoRT), uma subárea da IoT voltada especificamente para a conexão e integração de dispositivos robóticos a sistemas CPS.

Esse conceito, relativamente recente na literatura técnica, está alinhado aos princípios de digitalização, interoperabilidade e orientação por dados característicos da quarta revolução industrial. De acordo com [Batth, Nayyar e Nagpal 2018], a IoRT vai além da mera conectividade: ela permite que robôs monitorem eventos, combinem dados de múltiplos sensores e utilizem inteligência distribuída para tomar decisões autônomas e otimizar suas ações em tempo real.

De acordo com [Ovidiu et al. 2020], em IoRT os Veículos Móveis Autônomos (*Autonomous Mobile Robots* - AMR) podem se comunicar com outros CPS, outros robôs móveis, veículos e humanos. Isso permite aprender e aperfeiçoar as habilidades destes AMR a partir da comunicação de dados, surgindo assim novas capacidades técnicas como a automanutenção, autoconsciência entre outras. Estas habilidades facilitam a realização de tarefas sob diferentes condições ambientais e incertezas. Podem contribuir, ainda, com outros algoritmos de controle. Desta forma, IoRT se tornou uma ferramenta essencial dentro de arquiteturas de navegação autônoma.

De acordo com [Romeo et al. 2020], o resultado da integração da IoRT em sistemas robóticos é a capacidade de monitorar eventos no ambiente industrial, mesclando dados de sensores de uma variedade de fontes. Técnicas de Inteligência Artificial podem ser usadas para, a partir da disposição dos dados coletados, determinar as melhores ações ligadas as tarefas atribuídas aos robôs.

Outro termo recentemente encontrado na literatura, também derivado da área de IoT, é a Comunicação Veicular das Coisas (*Vehicle-to-Everything* - V2X) [Chen et al. 2020]. Essa área de pesquisa é semelhante ao IoRT, compartilhando diversas técnicas de desenvolvimento e protocolos de redes de comunicação sem fio. A principal diferença está no porte dos veículos móveis: enquanto o IoRT se concentra em robôs móveis, o V2X é aplicado principalmente a veículos autônomos que operam em vias públicas, geralmente de grande porte. O foco do V2X é atender a objetivos específicos relacionados à navegação, segurança e conectividade em sistemas de transporte inteligentes.

Dentro do escopo das tecnologias de comunicação V2X, destacam-se os conceitos de Unidade a Bordo (*On-Board Unit* - OBU) e Unidade de Estrada (*Roadside Unit* - RSU), que são elementos fundamentais para viabilizar a troca de informações essenciais à segurança e eficiência do trânsito. As OBUs, instaladas nos veículos, permitem a comunicação entre os próprios veículos e com as RSUs. As RSUs, por sua vez, são dispositivos fixos posicionados ao longo das vias, que facilitam a interação entre as OBUs e a infraestrutura de transporte, promovendo maior integração e coordenação no sistema viário.

A OBU é um dispositivo embarcado em veículos que possui capacidades de comunicação sem fio, permitindo a troca de dados com RSUs e outros veículos no ambiente V2X. Elas desempenham um papel crucial em várias aplicações, incluindo sistemas de alerta de colisão, assistência de manutenção de faixa, controle adaptativo de velocidade e sistemas de alerta de ponto cego. As OBUs equipam os veículos com a habilidade de "perceber"o ambiente ao redor por meio da troca de informações de segurança, como posição, velocidade e direção, aumentando assim a consciência situacional do motorista e dos sistemas autônomos de condução. Essas unidades são pilares que permitem a implementação efetiva de sistemas inteligentes de transporte, contribuindo significativamente para a visão de mobilidade autônoma e conectada.

As RSUs são componentes fixos localizados ao longo das vias, projetadas para comunicar-se com as OBUs dos veículos que passam. Estas unidades funcionam como pontos de acesso à infraestrutura de rede maior, possibilitando não apenas a comunicação veículo-infraestrutura (V2I) mas também servindo como intermediários na disseminação de informações entre veículos (V2V). As RSUs são essenciais para a implantação de Sistemas de Transporte Inteligente (*Intelligent Transportation Systems* - ITS), fornecendo

serviços como sinalização de tráfego adaptativa, informações sobre condições de tráfego e alertas de emergência.

Dentro da perspectiva da IoRT, as RSUs podem ser compreendidas como parte da infraestrutura física de suporte à conectividade robótica móvel, representando um elo fundamental entre os agentes ciberfísicos e a rede digital. Segundo [Villa et al. 2021], a arquitetura da IoRT é estruturada em cinco camadas funcionais: i) Camada Física, ii) Camada de Enlace, iii) Camada de Rede, iv) Camada de Transporte e v) Camada de Aplicação.

A camada física representa o mais baixo nível de um arquitetura IoRT. Esta camada é composta por diversos sensores e atuadores de um sistema, isto é, pode ser composta por elementos de percepção e atuação de diversos robôs. Essa tecnologia pode desenvolver e atualizar aplicativos inteligentes em sistemas robóticos, gerenciar remotamente atividades distribuídas e aumentar a tolerância a falhas, para obter uma melhoria no desempenho geral do sistema [Romeo et al. 2020].

A conexão desta rede de sensores e atuadores ocorre na camada de Enlace através dos protocolos de comunicação. Portanto nessa camada é possível encontrar *gateways* e transmissores em geral. As técnicas de comunicação como WiFi, *Bluetooth Low Energy* (BLE), *Radio Frequency Identification* (RFID), *Near Field Communication* (NFC) e comunicação baseada em Redes de Sensores sem Fio (Wireless Sensores Network - WSN) são usadas para obter a transmissão de informações entre sistemas robóticos de curta e longa distância [Belbachir e Benabid 2018].

A Camada de Rede e Controle desempenha um papel fundamental na integração eficiente dos componentes de *hardware* da camada física [Romeo et al. 2020]. Ela gerencia a interconexão entre dispositivos IoRT e a internet, utilizando protocolos como o IPv6. Essa camada é essencial para habilitar a comunicação de longo alcance, permitindo que os dados coletados pelos dispositivos sejam transmitidos de forma eficiente para servidores ou outros dispositivos remotos. O uso do protocolo IPv6 é especialmente vantajoso, pois ele suporta uma grande quantidade de dispositivos conectados simultaneamente, além de oferecer melhor desempenho no roteamento de pacotes de dados, contribuindo para a escalabilidade e a robustez da infraestrutura IoRT [Villa et al. 2021].

Complementando a Camada de Rede, a Camada de Transporte assegura a comunicação de ponta a ponta entre dispositivos e sistemas na rede IoRT. Nesta camada, os protocolos mais amplamente utilizados são o UDP (*User Datagram Protocol*) e o TCP (*Transmission Control Protocol*). O UDP se destaca em aplicações que exigem baixa latência e podem tolerar perdas eventuais de pacotes, sendo ideal para transmissão de dados sensoriais em tempo real ou para controle de robôs móveis. Por outro lado, o TCP é preferido em aplicações que demandam alta confiabilidade, garantindo a entrega completa e ordenada dos pacotes, o que é crucial para tarefas como transmissão de atualizações

críticas de software ou comandos de controle em sistemas de missão crítica.

A Camada de Aplicação representa o mais alto nível da arquitetura IoRT. Ela é composta por sistemas supervisórios, softwares de controle, interfaces homem-máquina (HMI) e serviços que integram funcionalidades de monitoramento, análise e tomada de decisão. Para viabilizar essa integração, utilizam-se protocolos como CoAP (Constrained Application Protocol), MQTT (Message Queue Telemetry Transport), REST API (Representational State Transfer), XMPP (Extensible Messaging and Presence Protocol) e TLS (Transport Layer Security).

Middlewares de comunicação desempenham um papel fundamental nesta camada, atuando como uma ponte entre os dispositivos robóticos (ou sistemas CPS) e os serviços da aplicação. Tecnologias como DDS, Zenoh, ZeroMQ e MQTT exemplificam diferentes abordagens de middleware que permitem troca de mensagens em tempo real, com diferentes garantias de qualidade de serviço (QoS) [Zhang et al. 2024]. Além disso, essa camada pode incorporar algoritmos de Inteligência Artificial e Machine Learning, que permitem o tratamento inteligente dos dados e a tomada de decisões autônomas, promovendo uma integração otimizada entre sistemas robóticos e aplicações da IoT.

Além dos avanços e conceituações em torno dos CPS e da IoRT, um conceito emergente que tem ganhado espaço na literatura técnica é o de *Robot-to-Everything* (R2X). Inspirado pela estrutura e aplicabilidade do V2X na comunicação veicular, o R2X expande esse paradigma para o domínio robótico, conectando robôs não apenas entre si, mas também a uma ampla gama de entidades, como sensores, infraestruturas, sistemas computacionais e até seres humanos. Enquanto a IoRT concentra-se na integração de dispositivos robóticos com sistemas ciberfísicos e redes IoT para otimizar percepção, monitoramento e controle, o R2X propõe uma abordagem mais abrangente, promovendo interações diretas e eficientes entre robôs e seu ecossistema.

Essa proposta está alinhada com a visão da IoRT, mas a expande ao incorporar maior interatividade e cooperação. Segundo [Jawhar et al. 2018], o R2X não apenas aprimora a comunicação e a colaboração entre robôs móveis autônomos (AMRs) e outros componentes dos CPS, mas também integra capacidades avançadas, como autoaprendizado e automanutenção. Essas características são essenciais para que os robôs desempenhem tarefas de forma eficiente em ambientes incertos e em constante mudança, reforçando o papel do R2X como uma evolução natural do IoRT em sistemas robóticos modernos.

A próxima seção aborda especificamente o *Data Distribution Service* (DDS), um protocolo de *middleware* amplamente utilizado para comunicação em sistemas distribuídos [Liu et al. 2018]. No contexto da IoRT, o DDS pode ser classificado na camada de transporte e, em parte, na camada de rede, pois ele gerencia a troca de dados em tempo real entre dispositivos e sistemas distribuídos, incluindo robôs móveis. O DDS opera sobre

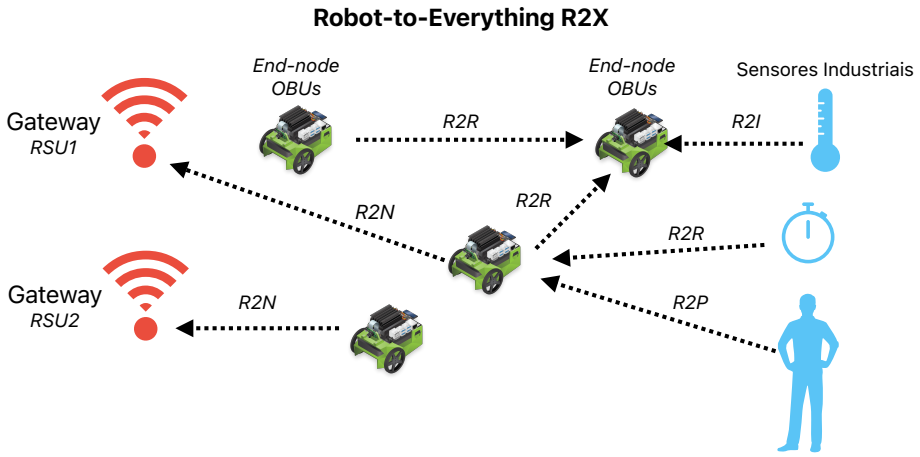


Figura 4.1: Proposta de sistema de comunicação robótica R2X - Autoria Própria.

protocolos de transporte como TCP e UDP, fornecendo uma infraestrutura robusta para comunicação *Robot-to-Robot* (R2R) e *Robot-to-Everything* (R2X), elementos essenciais para sistemas multi-robôs e CPS.

4.2 Middlewares de Comunicação

Um *middleware* de comunicação é uma camada de *software* que atua como intermediária entre aplicações distribuídas e os recursos de rede, permitindo que diferentes módulos ou dispositivos compartilhem dados de forma eficiente e padronizada. Em sistemas distribuídos, o *middleware* abstrai as complexidades da comunicação, gerenciando conexões, formatos de mensagem, sincronização e qualidade de serviço. Embora não seja formalmente parte do modelo OSI, sua funcionalidade abrange principalmente as camadas de sessão (5) e aplicação (7), oferecendo suporte à gestão de conexões, controle de estados e interoperabilidade entre diferentes dispositivos e protocolos [Liu et al. 2018] [Zhang et al. 2024].

O ROS [Macenski et al. 2022], discutido no Capítulo 2, surgiu como um dos *middlewares*, de natureza ampla, mais populares para sistemas robóticos modulares. Trata-se de um ambiente *open-source* que provê suporte à comunicação entre diferentes componentes de um sistema robótico, estruturado por meio de tópicos (*publish/subscribe*) e serviços (*request/response*). O ROS 2, lançado em 2015, representa a evolução do ROS clássico com foco em aplicações de tempo real, segurança e escalabilidade. Diferentemente da versão anterior, o ROS2 foi projetado para operar sobre diversos *middlewares* de comunicação, conhecidos como ROS Middleware Interfaces (RMWs).

O RMW (*ROS Middleware Interface*) é uma camada de abstração que conecta o ROS 2 a implementações específicas de *middlewares* de comunicação. Ele permite que o ROS2 seja compatível com diferentes padrões, sendo o DDS o principal deles. Essa arquitetura flexível garante que o desenvolvedor possa trocar o *backend* de comunicação conforme os requisitos da aplicação, priorizando por exemplo, confiabilidade ou baixa latência. Os RMWs implementam os mecanismos de publicação/assinatura, gerenciamento de conexões e políticas de QoS entre os nós do sistema.

O DDS é um padrão de comunicação baseado no modelo de publicação/assinatura (*publish/subscribe*) projetado para sistemas distribuídos em tempo real. DDS facilita a comunicação eficiente e confiável entre os componentes de um sistema robótico, sendo amplamente utilizado em aplicações que exigem alta performance e baixa latência [Liu et al. 2018].

No contexto da IoRT, o DDS serve como a espinha dorsal para comunicação em tempo real, enquanto o ROS fornece a interface de alto nível para o desenvolvimento e execução de aplicações robóticas. A combinação dessas tecnologias possibilita a criação de sistemas robóticos autônomos robustos, capazes de operar em ambientes industriais e remotos, onde a comunicação precisa ser eficiente e segura. A integração entre DDS e ROS2 é especialmente relevante para aplicações que exigem respostas rápidas, como navegação autônoma, monitoramento e controle em ambientes dinâmicos.

4.2.1 Processo de Comunicação DDS

O DDS é utilizado tanto em redes LAN quanto WAN, permitindo a troca de dados entre múltiplos nós de maneira rápida e previsível. No contexto de robótica, o DDS coleta e distribui mensagens geradas pelos nós, utilizando tabelas de publicação e assinatura e mecanismos de descoberta para gerenciar a transmissão das mensagens [Liu et al. 2018].

A Figura 4.2 ilustra a arquitetura lógica do DDS como *middleware* de comunicação em um ambiente com múltiplos nós ROS 2.

Cada nó ROS (Node) atua como publicador (Publisher) ou assinante (Subscriber) e se comunica com o espaço global de dados do DDS (Global Data Space) através de entidades chamadas Data Writers e Data Readers. As mensagens são organizadas em tópicos (Topics), que funcionam como canais de comunicação desacoplados — isto é, o publisher e o subscriber não precisam conhecer diretamente um ao outro. Os tópicos são agrupados em domínios ROS (ROS Domains), permitindo segmentar a comunicação em contextos distintos (por exemplo, diferentes robôs ou subsistemas).

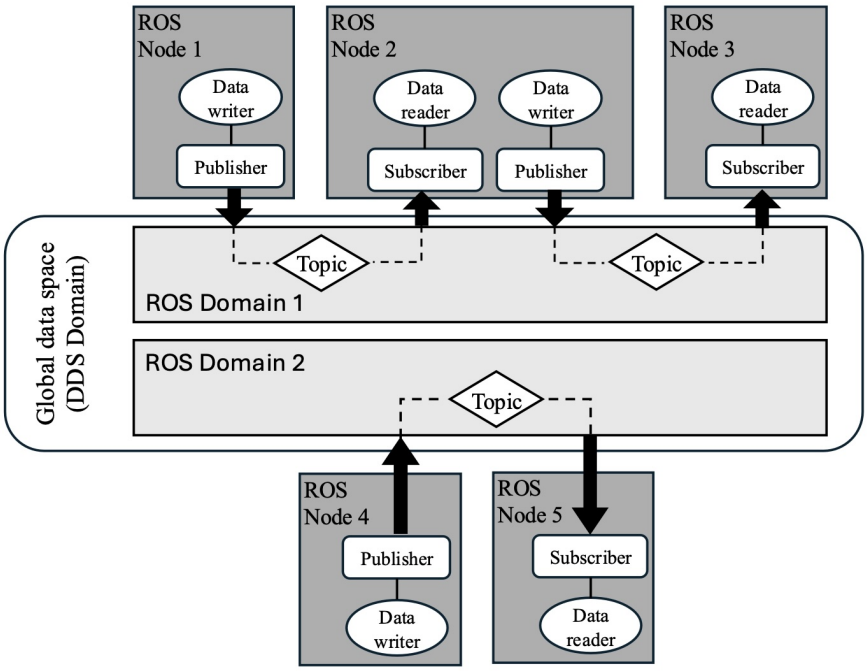


Figura 4.2: Arquitetura do DDS no Sistema ROS2 - Retirado de [Zhang et al. 2024]

4.2.2 Utilização do DDS no ROS

O DDS foi projetado para operar sobre redes baseadas nos protocolos TCP/IP ou UDP/IP, utilizando-os como camada de transporte. Isso ocorre porque o DDS não implementa diretamente os mecanismos físicos e de enlace de dados; ele depende de uma infraestrutura de rede subjacente para realizar o transporte efetivo dos pacotes entre os nós. Na maioria das implementações, o DDS utiliza o protocolo RTPS (Real-Time Publish-Subscribe), que é encapsulado em UDP/IP, por oferecer menor latência, suporte a *multicast* e melhor desempenho em sistemas distribuídos com múltiplos assinantes. No entanto, em cenários onde a confiabilidade do transporte é crítica (por exemplo, em redes não determinísticas ou em comunicação pela internet), é possível utilizar o DDS sobre TCP/IP, que oferece controle de entrega e reordenação dos pacotes.

O processo de comunicação no DDS é gerenciado por mecanismos de descoberta automática, que identificam e conectam dinamicamente nós publicadores e assinantes, eliminando a necessidade de configuração manual. Além disso, o DDS suporta diferentes tipos de mensagens, incluindo:

- **Mensagens de Publicação (PUB)**: Dados enviados pelos nós publicadores.
- **Mensagens Não Publicadas (UNP)**: Mensagens ainda não processadas pelos publicadores.
- **Mensagens de Assinatura (SUB)**: Dados que os nós assinantes recebem dos tópicos subscritos.

- **Mensagens Não Assinadas (UNS)**: Mensagens aguardando consumo pelos assinantes.
- **Mensagens de Dados (DAT)**: Dados efetivamente transmitidos entre os nós.

O fluxo de comunicação inicia-se com a coleta das mensagens pelos *buffers* de publicação. Em seguida, essas mensagens são transmitidas pelo canal de comunicação e recebidas pelos *buffers* de assinatura dos nós inscritos. Essa estrutura garante um alto grau de previsibilidade, baixa latência e entrega eficiente, mesmo em sistemas robóticos distribuídos e dinâmicos.

Por exemplo, em um robô equipado com um sensor LiDAR, o nó responsável pela aquisição de dados sensoriais publica as informações do ambiente em tempo real em um tópico específico, como `/scan`. Outros nós, como os responsáveis por mapeamento, planejamento de rotas ou detecção de obstáculos, podem se inscrever nesse tópico para receber os dados e processá-los conforme necessário. Esse modelo de comunicação permite que o robô integre os dados sensoriais ao seu sistema de navegação de maneira eficiente e com baixa latência.

O DDS assegura a *Qualidade de Serviço* (QoS) por meio de políticas configuráveis, como garantias de entrega de mensagens, priorização e controle de largura de banda. Essas políticas são particularmente importantes em cenários robóticos dinâmicos, onde é necessário garantir a entrega oportuna de informações críticas, como detecção de obstáculos ou comandos de emergência.

Além das vantagens citadas, as ferramentas de visualização, como o *RVIZ*, e os pacotes de cálculo de odometria, como o *nav_msgs/Odometry*, permitem desenvolver, monitorar e testar algoritmos complexos de maneira eficiente. Por exemplo, a partir do RVIZ é possível acompanhar em tempo real as movimentações do robô, criando uma espécie de gêmeo digital do robô físico em ambiente virtual [Elbakry et al. 2023].

A arquitetura de comunicação baseada em DDS no ROS2 garante que as mensagens de alta prioridade sejam transmitidas e processadas preferencialmente, melhorando a confiabilidade e a performance do sistema [Liu et al. 2018]. Isso é crucial em aplicações de robótica onde a segurança e a resposta em tempo real são essenciais.

4.2.3 Limitações do DDS e Conectividade em Ambientes Remotos

Como já explanado no tópico anterior, a comunicação tradicional baseada em DDS utiliza Wi-Fi (ou qualquer outra rede IP) como meio de transmissão. A troca de informações entre robôs ou entre robôs e servidores ocorre por meio de protocolos que operam sobre a pilha TCP/IP. No entanto, em situações onde o Wi-Fi ou outra rede IP não esteja disponível, a comunicação nativa entre tópicos do ROS2 é interrompida, uma vez que os nós dependem da conectividade IP para se comunicar.

Apesar das inúmeras vantagens, o DDS apresenta algumas limitações. Uma das principais restrições é que o DDS se baseia em redes com camada de transporte TCP, o que pode não ser ideal para redes de larga escala (WAN) devido a possíveis latências e perdas de pacotes. Tecnologias atuais como o Wi-Fi têm limitações em alcance e infraestrutura, especialmente em cenários remotos e de grande área, como fazendas, agricultura, grandes áreas industriais abertas, operações de mineração, plantas de tratamento de água e missões de resgate em ambientes devastados. Além disso, a complexidade da implementação do DDS pode exigir configurações e ajustes detalhados para garantir a performance desejada [Liu et al. 2018].

A Figura 4.1 ilustra o cenário de operação de robôs em uma área agrícola, destacando a transição da comunicação tradicional ROS2-DDS via Wi-Fi para o uso de tecnologias LPWAN (*Low Power Wide Area Networks*), como por exemplo a tecnologia LoRa em áreas fora de cobertura de redes tradicionais.

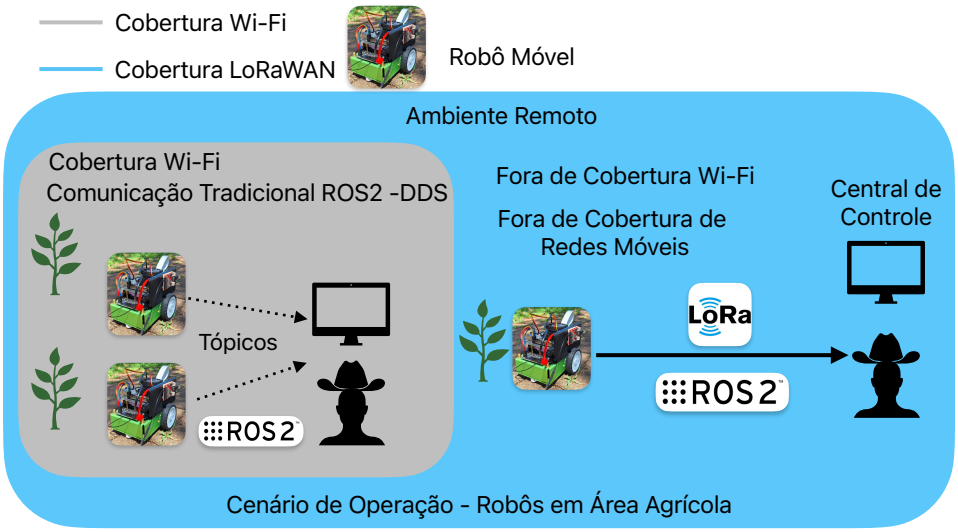


Figura 4.3: Aplicação de Comunicação Baseada em ROS2 em Ambiente Remoto - Autoria Própria.

Essa abordagem de expansão de conectividade permite monitorar e controlar os robôs em tempo real, mesmo em ambientes desafiadores e distantes de um centro de comando. Nesse exemplo verifica-se a ilustração de robôs atuando no monitoramento de uma hipotética produção agrícola. O centro de controle está distante da produção. Não existe cobertura de redes tradicionais IP ou redes celulares. Nesse caso uma integração com redes LPWAN passa a ser uma solução para estender a aplicação o uso do ROS2.

Com sua capacidade de longo alcance e baixo consumo de energia, o LoRaWAN [Semtech 2019] emerge como uma solução promissora para comunicação em áreas remotas e de difícil acesso. A integração do ROS2 com LoRaWAN é um dos pontos-chave desta

pesquisa, pois permite que robôs enviem dados sensoriais de forma confiável, mesmo em ambientes onde redes Wi-Fi ou móveis não estão disponíveis.

Visando o processo de comunicação na arquitetura de navegação autônoma discutida neste trabalho, propõe-se uma metodologia inovadora para expandir a conectividade baseada em ROS2 em redes LoRaWAN, utilizando uma combinação de técnicas de compressão de dados e recursos baseados em inteligência artificial. Essa abordagem não só amplia as capacidades de comunicação dos robôs móveis, mas também viabiliza a aplicação de sistemas de navegação robótica em cenários industriais e agrícolas desafiadores. A próxima seção detalhará os conceitos e fundamentos do protocolo LoRaWAN, que sustentam essa integração.

4.3 Protocolo de Comunicação LoRa

O protocolo LoRa (*Long Range*) é uma das diversas possibilidades de comunicação de rede sem fio em ambiente IoRT [Villa et al. 2021]. O protocolo LoRa é um dos mais utilizados em redes IoT tradicionais, principalmente em redes de área ampla (*Long Power Wide Area Networks - LPWAN*) [Park, Lee e I.Joe 2020]. Este protocolo inclui duas camadas do modelo OSI/ISO: física (LoRa RF) e camada de enlace (LoRaWAN).

Sua principal característica é a capacidade de realizar comunicações de longa distância e baixa potência, atendendo requisitos para IoT que redes sem fio convencionais não costumam atender. Uma rede LoRa possui parâmetros de comunicação adaptáveis para manter boas conexões de *uplink* e *downlink* [Raza, Kulkarni e Sooriyabandara 2017].

O LoRa possui vantagens em relação a outros protocolos de rede sem fio, como por exemplo o seu alcance, com aplicações em uma faixa de até 15km. Em outros protocolos, como por exemplo o Zigbee, muito utilizado em comunicações sem fio industriais, possui a faixa de alcance de aproximadamente 100m. Para percorrer as mesmas distâncias que outras tecnologias sem fio como LoRa ou WiMAX podem oferecer, o Zigbee deve escolher comunicações *multi-hop* em vez de comunicações *device-device* (DD) [Haque et al. 2020].

A modulação utilizada no LoRa, ou seja, a forma como o sinal (informação) é transmitido pela rede e a forma que é manipulada no transceptor é digital e chamada de *Chirp Spread Spectrum* (CSS), uma tecnologia proprietária desenvolvida pela empresa [Semtech 2019]. Esta modulação é semelhante à tradicional *Frequency Shift Keying* (FSK), com diferenças em sua portadora e largura de banda. O CSS utiliza uma série de pulsos para modulação semelhantes a rampas, denominadas de *Compressed High Intensity Radar Pulse* (chirp).

A eficiência em termos de taxa de dados e potência desta modulação depende principalmente de três parâmetros: i) largura de banda *(Bandwidth - BW)*, fator de espalhamento espectral (*Spreading Factor - SF)* e taxa de codificação *(Code Rate - CR*). Um sinal digital modulado pela tecnologia CSS é composto pelo preâmbulo e carga útil de informação (*payload*). Os pulsos *chirp* (uma espécie de rampa) são portadores de dados espaçados no domínio da frequência. O SF representa a forma em uma quantidade de *bits* (ou símbolo) é espaçado ou modulado, podendo assumir seis valores distintos: do SF 7 ao SF 12. A quantidade de *chips* necessários para codificar um sinal digital é dado por:

$$N_{chips} = 2^{SF} \tag{4-1}$$

Desta forma, na modulação CSS, o SF define dois parâmetros na modulação: i) o número de *chips* necessários para codificar um sinal digital e ii) a quantidade de *chips* que cada símbolo vai carregar.

A taxa de dados (*bit rate*) R_b *bps* do LoRa é dada por:

$$R_b = SF \times \frac{\frac{4}{4+CR}}{\frac{2^{SF}}{BW}} \tag{4-2}$$

A informação a ser transmitida é resultado de um código gerado a partir dos pulsos *chips*. A Figura 4.4 abaixo ilustra uma série de *chips* lineares usados para modular um sinal digital em uma determinada largura de banda (B), em função do tempo.

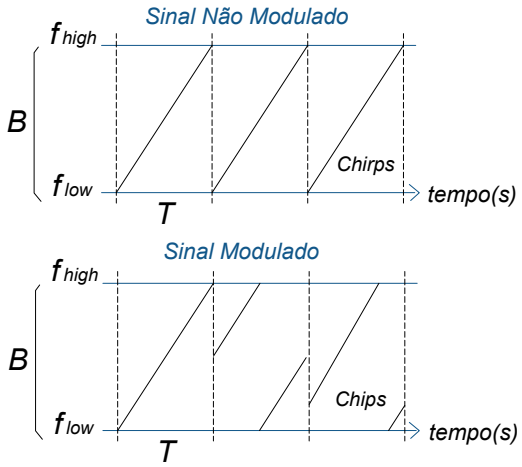


Figura 4.4: Frequencia × Tempo: Sinal CSS - Autoria Própria.

Dois sinais são ilustrados na Figura 4.4. No primeiro, nenhum sinal digital está sendo modulado, onde é possível notar um salto linear, interrupto, entre a frequência f_{low} até f_{high}. No segundo é possível verificar pequenos pulsos *chips* espaçados resultantes da codificação de um determinado símbolo (informação digital). Cada símbolo é codificado

por uma quantidade de *chips*, onde esta quantidade e seu espalhamento no domínio da frequência está relacionada ao fator SF escolhido.

Por exemplo, suponha-se a transmissão do símbolo digital [01011111] que representa o valor decimal 95, ilustrado pela Figura 4.5. Em uma modulação CSS, caso seja selecionado o fator SF 7, o sinal seria codificado em uma série de pulsos de 2^{SF}, resultando em 128 *chips*. Caso seja escolhido o fator SF 12, o sinal seria codificado em série de pulsos de 2^{SF}, resultando em 4096 *chips*. Fica evidente que quanto maior o SF, maior o número de *chips* que um símbolo carregará. A escolha da melhor SF em uma modulação CSS, seus impactos, vantagens e desvantagens são descritos na próxima seção.

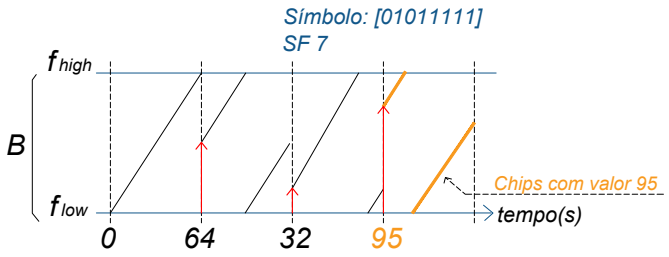


Figura 4.5: Exemplo de Modulação do Símbolo Digital - Autoria Própria.

4.4 Controle de Espalhamento Espectral - SF

Uma rede IoRT, baseada em veículos móveis, exige mobilidade, isto é, os transmissores e receptores da rede estão em constante movimentação. O robô móvel autônomo (AMR) pode ser visto como um *gateway* da rede LoRa, um dispositivo que recebe uma quantidade massiva de dados relevantes para a otimização de suas habilidades ou dados necessários para monitoramento de um processo industrial.

Os CPS presentes no ambiente industrial ou em um determinado processo são transmissores de dados e podem ser vistos como nós finais da rede LoRa (*end-nodes*). Ou seja, os *end-nodes*, que podem ser móveis ou fixos transmitem uma série de informações ao *gateway* da rede LoRa incorporado ao robô móvel.

Se um dispositivo transmissor de dados, por exemplo um sensor, está muito distante do *gateway* LoRa, torna-se necessário aumentar o fator SF na modulação CSS, mantendo a informação transmitida por mais tempo no ar (*ToA*). Entretanto este aumento faz com que a taxa de transferência de dados decaia e a potência de transmissão aumente, comprometendo a eficiência energética dos módulos.

Como exemplo, a Tabela 4.1 abaixo apresenta alguns valores de *ToA* estimados com o auxílio da calculadora LoraTools (https://www.loratools.nl/#/airtime)

para transmissão de um pacote de 25 bytes. É possível confirmar que o aumento da SF exige um maior tempo para manutenção do pacote a ser transmitido no meio.

Tabela 4.1: *ToA* estimado para cada SF.

ToA(ms)	1646	921	460	230	127	70
SF	12	11	10	9	8	7

Quando a transmissão de um pacote de dados ocorre de maneira simultânea, no mesmo canal, existe a probabilidade destes pacotes se colidirem, corrompendo a informação. Após uma colisão, normalmente é necessário retransmitir um pacote, diminuindo assim a capacidade do canal de comunicação e a eficiência da rede [Xu et al. 2020]. Para evitar as colisões, a rede LoRa adota o protocolo de acesso randômico ALOHA. A escolha do SF altera o tempo médio para transmissão do frame (T_{rf}), que é diretamente proporcional ao *ToA* daquela determinada SF. Ou seja, a escolha do SF está relacionado ao desempenho do protocolo ALOHA [Okinaka et al. 1975] na contenção das colisões entre pacotes.

Diante destas possibilidades, observa-se a expectativa dos parâmetros SF serem otimizados durante um processo de comunicação entre dispositivos LoRa. A Figura 4.6 ilustra a relação entre o *ToA* e a taxa de dados para diferentes fatores de espalhamento espectral. Quanto menor o *ToA* na transmissão de dados entre um dispositivo final e o *gateway*, menor é a chance da ocorrência de colisões de pacotes. Todavia, quanto maior a distância entre o *gateway* e o dispositivo final, necessita-se disponibilizar a informação por mais tempo no espaço, aumentando assim o *ToA*, a potência e intensidade do sinal (RSSI) no *gateway* e consequentemente o SF.

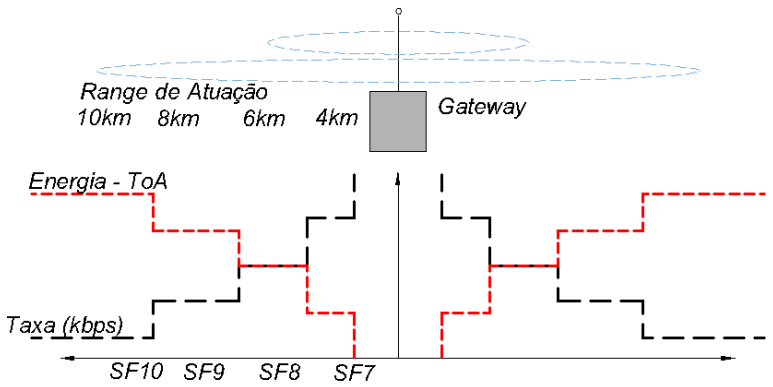


Figura 4.6: Impactos funcionais nas alterações do SF - Adaptado de [Semtech 2019].

4.4.1 Taxa de Dados Adaptativa

Atualmente o protocolo LoRa especifica o algoritmo Taxa de Dados Adaptativa (*Adaptative Data Rate - ADR*) [Lehong et al. 2020]. Este algoritmo é capaz de otimizar parâmetros da camada física, como a seleção do SF, a taxa de dados e o ”Tempo no Ar” (*Time on Air - ToA)* e tem por objetivo minimizar os efeitos causados pelo aumento de usuários na rede. Basicamente o ADR usa a relação sinal ruído *(Signal to Noise Ratio - SNR)* para determinar bons parâmetros LoRa RF. Portanto é uma técnica iterativa, baseada em leituras de estados coletados e controle em tempo real.

O ADR é controlado pelo *network server* e envolve o seguinte processo de monitoramento: O *network server* monitora a taxa de sucesso das mensagens (ACKs), SNR (Signal-to-Noise Ratio), e a margem de link de cada dispositivo final.

Com base nos dados e métricas de desempenho coletadas, o *network server* decide se o fator de espalhamento espectral dados SF e a potência de transmissão devem ser ajustadas.

Se o dispositivo final recebe ACKs para suas mensagens, e a margem de link está acima do limite necessário, a rede pode decidir aumentar a taxa de dados (diminuir o SF) ou diminuir a potência de transmissão. Se o dispositivo não recebe ACKs ou a margem de link está abaixo do limite, a rede pode diminuir a taxa de dados (aumentar o SF) ou aumentar a potência de transmissão. A Figura 4.7 ilustra o algoritmo do ADR.

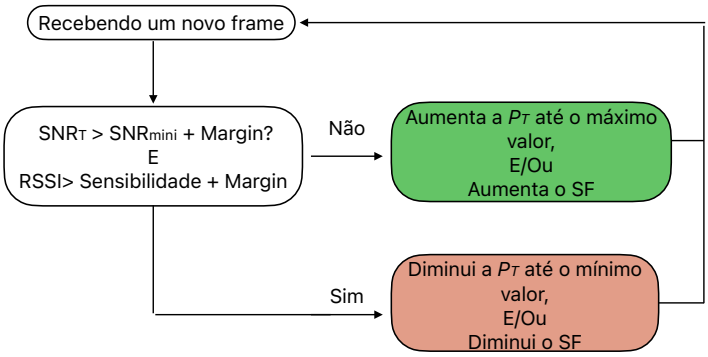


Figura 4.7: Fluxograma do algoritmo ADR.

No fluxograma o *gateway* concentrador do LoRaWAN recebe um novo *frame*, um pacote de dados enviado por um *end-node*. A partir deste *frame* estima-se o SNR_T desta transmissão. O protocolo LoRa trabalha com um SNR mínimo para demodular sinais (SNR_{min}), que se encontra na faixa de -20dB a -7.5dB. Se o SNR_T for maior que o SNR_{min} então o ADR reduz o SF e a potência de transmissão, pois aumenta a eficiência energética dos módulos, uma vez que menores SF garantem uma potência de

transmissão (P_T) menor. Se o SNR_T for menor que o SNR_{min} então o ADR aumenta o SF, aumentando assim a P_T.

Segundo [Park, Lee e I.Joe 2020] as pesquisas atuais visam modificar o ADR existente, para melhorar o seu rendimento ou adicionar tecnologias, como aprendizado de máquina para uso confiável dos recursos de rede. Para o desenvolvimento de um *gateway* inteligente capaz de realizar de forma otimizada a seleção dos parâmetros de modulação LoRa, é necessário considerar três pontos essenciais: **i) colisões de pacotes, ii) atenuação do sinal e iii) taxa de transferência.**

Propõe-se neste artigo uma técnica de alteração do SF de modo interativo, computacionalmente simples e que não envolva o feedback do SNR, fazendo uma varredura de três SF durante o loop do microcontrolador. Ou seja, realiza-se a tentativa de envio do sinal com SFs diferentes, forcando a mudança de espalhamento espectral. Denominamos esta variação do algoritmo de ADR de Taxa Forçada de Dados (Forced Data Rate). Este algoritmo faz parte do sistema de navegação autônoma, melhor descrito no Capítulo 5 (Sistema de Navegação Autônoma).

Levantamentos bibliográficos realizados indicam que a literatura voltada para este assunto ainda é incipiente em métodos baseados em aprendizagem por reforço. Assim, a proposta de controle de espalhamento espectral contribui no atual estado da arte nos seguintes pontos:

- novos algoritmos visando a melhoria de sistemas multiusuário;
- técnicas alternativas ao ADR existente;
- sistema de controle robusto/inteligente de parâmetros da camada física LoRa RF.

4.4.2 Infraestrutura LoRa

Para implementar uma rede LoRa - LoRaWAN, é necessário compreender a infraestrutura básica que envolve gateways, end-points e a configuração das antenas. Esta seção descreve esses componentes e suas características específicas, especialmente no contexto de uma rede robótica.

O gateway é um componente central em uma rede LoRaWAN, atuando como um ponto de acesso que conecta os dispositivos finais (end-points) à rede central. Ele recebe dados dos end-points via rádio e os encaminha para um servidor de rede através de uma conexão de backhaul, que pode ser Ethernet, Wi-Fi, ou celular.

Os gateways LoRaWAN possuem uma alta sensibilidade para captar sinais de longa distância, sendo capazes de gerenciar a comunicação de milhares de dispositivos simultaneamente. Em uma rede robótica, o gateway geralmente é fixo, proporcionando uma base estável para a coleta e retransmissão de dados sensoriais e de controle.

Os end-points são dispositivos de comunicação final na rede LoRaWAN, responsáveis por enviar e receber dados. Em uma rede robótica, os end-points são normalmente transceptores de rádio alocados nos robôs. Esses dispositivos são móveis e se comunicam com o gateway para transmitir dados sensoriais, comandos de controle e outras informações relevantes. Os end-points utilizam a modulação do protocolo LoRa (CSS) para comunicação de longa distância com baixo consumo de energia, sendo ideais para aplicações em robótica onde a eficiência energética é crucial.

O posicionamento adequado das antenas é fundamental para garantir uma cobertura eficiente e uma comunicação robusta na rede LoRa. As antenas devem ser posicionadas de forma a minimizar obstáculos físicos que possam causar interferências no sinal. Em ambientes abertos, como áreas industriais, agrícolas ou de mineração, as antenas dos gateways devem ser instaladas em locais elevados para maximizar a linha de visada (LOS) com os end-points móveis.

4.4.3 Aplicação da rede LoRa em Dispositivos em Mobilidade

A rede LoRa tem se destacado como uma tecnologia promissora para a comunicação em dispositivos de Internet das Coisas (IoT) devido à sua capacidade de comunicação de longo alcance e baixo consumo de energia. No entanto, a aplicação da rede LoRa em dispositivos móveis, como comunicação robótica, apresenta desafios específicos, especialmente em relação ao algoritmo de controle de espalhamento espectral (ADR - Adaptive Data Rate).

O algoritmo ADR foi originalmente concebido para ajustar a taxa de dados com base nos valores de SNR (Signal-to-Noise Ratio) e RSSI (Received Signal Strength Indicator), que são coletados de dispositivos em posições estáticas. Essa abordagem se mostrou eficiente para dispositivos estacionários, onde as condições do canal de comunicação permanecem relativamente estáveis ao longo do tempo.

Entretanto, quando aplicados a dispositivos móveis, como os robôs em movimento, esses parâmetros de *feedback* podem ser distorcidos devido às constantes mudanças de posição e à variação das condições ambientais. Essa distorção impacta negativamente a eficácia do ADR, resultando em ajustes inadequados do espalhamento espectral. Estudos, como o de [Benkahla et al. 2019], destacam que o esquema básico do ADR pode não ser eficiente no caso de nós móveis. O ADR tradicional realiza a adaptação da taxa de dados somente após a recepção de um conjunto de frames. Essa abordagem não considera adequadamente a degradação do sinal causada pela mobilidade do nó ou pela presença de obstáculos móveis. Esperar pela coleta deste conjunto de frames pode ser muito demorado para se adaptar rapidamente a novas situações.

Por essa razão o módulo de comunicação LoRa apontado neste trabalho apre-

senta uma proposta de modificação para suportar a comunicação entre os dispositivos em movimento.

Algumas pesquisas e trabalhos também apresentam integrações de redes LoRa com sistemas operacionais ROS. O trabalho descrito em [Karatzia, Kolios e Ellinas 2023] foca na coordenação de enxames de drones (Veículos Aéreos Não Tripulados - UAVs) em missões críticas. A integração dos frameworks LoRa e ROS possibilita uma comunicação robusta e eficiente entre os drones, essencial para operações de resgate e vigilância. A principal contribuição é a demonstração de um sistema que melhora a coordenação e a execução de missões em ambientes adversos.

Em [Manuel, Faied e Krishnan 2022], é proposta uma nova arquitetura de comunicação baseada em LoRa para missões de busca e resgate. A arquitetura apresentada melhora significativamente a comunicação em áreas remotas e de difícil acesso, onde redes tradicionais não são viáveis. Este trabalho é relevante por demonstrar a viabilidade do uso do LoRa para apoiar operações de emergência, garantindo a transmissão de dados vitais em situações de resgate.

Os autores em [Santos, Silvestre e Cunha 2023] desenvolveram e validaram experimentalmente uma rede de sensores sem fio baseada em LoRa para vigilância de incêndios florestais. A pesquisa enfatiza a importância de monitorar vastas áreas de floresta onde a cobertura de redes convencionais é limitada. A principal contribuição é a criação de uma rede robusta que permite a detecção precoce e a resposta rápida a incêndios, utilizando a combinação de tecnologias LoRa e sensores.

No entanto, embora o protocolo LoRa ofereça vantagens significativas, como baixo consumo de energia e longo alcance, ele apresenta limitações importantes relacionadas ao tamanho do payload, especialmente em cenários que envolvem a transmissão de grandes volumes de dados sensoriais, como os provenientes de sensores de alta carga, por exemplo, LiDAR. A Tabela 4.2 ilustra as restrições do protocolo em diferentes *Spreading Factors* (SF), demonstrando como o aumento do SF, necessário para melhorar a sensibilidade do receptor e alcançar distâncias maiores, reduz drasticamente a taxa de transmissão e o tamanho máximo do payload.

Tabela 4.2: Limitações de Payload no Protocolo LoRa

SF	**Taxa de Transmissão (kbps)**	**Tempo de Transmissão (ms)**	**Payload Máximo (bytes)**
SF7	5.47	56	222
SF8	3.13	103	222
SF9	1.76	185	115
SF10	0.98	371	51
SF11	0.54	741	51
SF12	0.29	1483	51

Os valores apresentados na tabela podem ser calculados a partir de equações que

relacionam o tempo de transmissão, a taxa de dados e o tamanho máximo do *payload* ao fator de espalhamento espectral SF. Para o caso de $SF7$, o tempo de transmissão de um símbolo (T_s) pode ser estimado pela relação:

$$T_s = \frac{2^{SF}}{BW} \tag{4-3}$$

Substituindo $SF = 7$ e $BW = 125\,\text{kHz}$ (valores típicos do protocolo LoRa), temos:

$$T_s = \frac{2^7}{125 \times 10^3} = \frac{128}{125 \times 10^3} = 1.024\,\text{ms}.$$

O tempo total no ar (T_{Air}) para a transmissão de um pacote depende do número de símbolos transmitidos ($n_{symbols}$), que pode ser calculado considerando o tamanho do $Payload$ e outros parâmetros do protocolo. O número total de símbolos é dado por:

$$n_{symbols} = n_{preamble} + 4.25 + \max\left(\left\lceil \frac{8 \cdot Payload - 4 \cdot SF + 28 + 16 - 20 \cdot H}{4 \cdot (SF - 2 \cdot DE)} \cdot (CR + 4) \right\rceil, 0\right) \tag{4-4}$$

Para o caso de $Payload = 222\,\text{bytes}$, $SF = 7$, $CR = 1$ (*Code Rate* de 4/5), $H = 1$ (cabeçalho explícito) e $DE = 0$ (otimização desativada), e considerando $n_{preamble} = 8$ $n_{symbols} = 329.25$.

O tempo total de transmissão (T_{Air}) é, então:

$$T_{Air} = n_{symbols} \cdot T_s = 329.25 \cdot 1.024\,\text{ms} \approx 56\,\text{ms}. \tag{4-5}$$

Por fim, o tamanho máximo do payload ($Payload_{max}$) é determinado pelo tempo máximo permitido para a transmissão e pela quantidade de símbolos que podem ser transmitidos dentro desse limite. No caso de $SF7$, $Payload_{max} = 222\,\text{bytes}$ já é determinado pelas restrições regionais e pela estrutura do protocolo.

Essa limitação reforça que o LoRa é um protocolo de banda estreita (*narrow-band*), otimizado para aplicações de baixa taxa de transmissão e consumo energético reduzido, o que o torna ideal para cenários onde o tráfego de dados é esporádico e os volumes de informação são pequenos, como em sensores ambientais e dispositivos IoT tradicionais. No entanto, essas características também impõem desafios significativos em aplicações que envolvem sensores de alta carga de trabalho, como LiDARs, que requerem taxas de transmissão e tamanhos de payload muito superiores às capacidades oferecidas pelo LoRa.

A compressão de dados é um processo destinado a reduzir o tamanho de um arquivo ao codificar suas informações de forma mais eficiente. O objetivo principal da compressão é diminuir a quantidade de dados necessária para armazenar ou transmitir um arquivo, sem perda significativa de informação. Diversas técnicas podem ser empregadas

para a compressão de dados, cada uma com características e vantagens específicas conforme o tipo de aplicação.

Uma técnica clássica de compressão é o ZIP, um método de compressão sem perdas amplamente utilizado devido à sua simplicidade e aplicabilidade geral em vários tipos de arquivos [Jayasankar, Thirumal e Ponnurangam 2021]. O ZIP é particularmente eficaz em dados com elevado nível de redundância ou padrões repetitivos, como documentos de texto, planilhas ou arquivos contendo grande quantidade de metadados. Para isso, emprega algoritmos como o DEFLATE, que reduzem o tamanho dos arquivos ao identificar e codificar sequências repetidas de dados, operando de forma eficiente em formatos altamente estruturados.

No entanto, quando se tratam de dados especializados, como aqueles gerados por sensores LiDAR, métodos tradicionais como o ZIP podem não ser tão eficientes. Dados LiDAR geralmente consistem em fluxos contínuos de medições numéricas, os quais apresentam baixa redundância e poucos padrões repetitivos, dificultando o aproveitamento das estratégias usadas pelo ZIP. Como consequência, a taxa de compressão obtida costuma ser limitada, produzindo arquivos comprimidos ainda relativamente grandes.

Nesse contexto, ao revisar a literatura, é evidente que as pesquisas sobre redes de longa distância integradas ao *Robot Operating System* (ROS) ainda estão em desenvolvimento e enfrentam barreiras importantes. As limitações do protocolo LoRa representam um desafio significativo, especialmente quando se busca utilizá-lo para a transmissão de dados sensoriais em áreas remotas. Esta tese, no âmbito do **Algoritmo de Comunicação**, propõe uma solução inovadora para esse problema ao investigar e implementar métodos de compressão de dados baseados em técnicas de aprendizado profundo, como autoencoders. A integração de LoRaWAN com o ROS, associada à compressão de dados, possibilita a transmissão eficiente de informações sensoriais de alto *payload* em redes de banda estreita. Essa abordagem não apenas supera as limitações impostas pelo tamanho do *payload*, mas também assegura a confiabilidade da transmissão, ampliando as possibilidades de aplicação do sistema em ambientes dinâmicos e sem infraestrutura de redes tradicionais, como Wi-Fi ou redes móveis.

4.5 Requisitos de QoS para Navegação Autônoma

A navegação autônoma de robôs móveis impõe requisitos rigorosos de Qualidade de Serviço (QoS) na comunicação entre sensores, atuadores e unidades de controle. Em ambientes dinâmicos, onde decisões devem ser tomadas em tempo real, a latência de comunicação torna-se um fator decisivo para a confiabilidade e a segurança da operação. Aplicações de navegação em tempo real, como evitação de obstáculos e controle de trajetória, exigem latências inferiores a 10 ms, enquanto aplicações como localização e

mapeamento simultâneo (SLAM) toleram latências na faixa de 10 a 100 ms. Em tarefas de monitoramento remoto ou compartilhamento de mapas em sistemas multi-robô, tolera-se latência maior, geralmente entre 100 ms e 500 ms, desde que acompanhada de garantias de confiabilidade e ordenação.

Para atender a esses requisitos, *middlewares* com suporte explícito a QoS são fundamentais. O DDS destaca-se nesse cenário por oferecer um modelo de comunicação orientado a dados com configurações refinadas de QoS, como confiabilidade (*reliable/best effort*), tempo de vida das mensagens (*lifespan*), deadline, latência mínima esperada (*latency-budget*) e histórico de entrega (*history*). Essas funcionalidades permitem adaptar a comunicação às necessidades de cada subsistema do robô, possibilitando, por exemplo, comunicação entre sensores LiDAR e módulos de decisão, ao mesmo tempo em que dados de menor prioridade, como telemetria ou *logs*, são transmitidos com configurações mais tolerantes. O uso de DDS é, por isso, amplamente adotado na robótica baseada em ROS2, viabilizando sistemas distribuídos e cooperativos em tempo real.

Por outro lado, o protocolo LoRa apresenta características que limitam seu uso em tarefas de controle robótico direto. Com latência média superior a 500 ms e largura de banda entre 0.3 e 50 kbps, o LoRa não atende aos requisitos típicos de tempo real exigidos por sistemas de navegação. Contudo, seu alcance de comunicação superior a 10 km e sua eficiência energética tornam o protocolo atrativo para aplicações de sinalização de eventos, telemetria de longo alcance ou comunicação auxiliar em ambientes remotos, nos quais tecnologias como Wi-Fi ou 4G não estão disponíveis. Isso tem motivado a comunidade científica a investigar modelos híbridos em sistemas IoRT, nos quais o LoRa é utilizado de forma complementar em arquiteturas de rede adaptativa, possibilitando que robôs compartilhem informações de estado ou alertas mesmo em regiões de baixa cobertura.

Diante das limitações dos diferentes protocolos de comunicação, surge a proposta de uma arquitetura híbrida que combina tecnologias de curto alcance, como DDS, com protocolos de longo alcance, como o LoRa. Nessa abordagem, o DDS é utilizado para comunicação em curto alcance (*short range*), como entre robôs próximos, sensores embarcados e unidades de controle, aproveitando sua capacidade de oferecer baixa latência, alta taxa de atualização e suporte avançado a QoS. Por outro lado, o LoRa é integrado como um canal auxiliar para comunicação em cenários onde a infraestrutura de rede é limitada ou inexistente, permitindo a transmissão de mensagens críticas, como status de missão, alertas ou coordenação remota, em distâncias que ultrapassam o alcance das redes *Wi-Fi* ou *Bluetooth*. Essa combinação entre *Short Range* e *Long Range* permite ampliar a autonomia, a resiliência e a escalabilidade de sistemas robóticos distribuídos, especialmente em ambientes adversos ou com infraestrutura limitada, como áreas rurais, industriais ou de resgate.

O próximo Capítulo, dispõe a metodologia de implementação do Sistema de Navegação Autônoma, detalha a proposta do **Algoritmo de Comunicação** para Long Range-LoRa, e Short Range (DDS) dentre outros algoritmos/módulos que compõem o sistema como um todo.

4.6 Conclusões

Este capítulo apresentou os conceitos fundamentais da comunicação robótica moderna, abordando as tecnologias IoRT (*Internet of Robotic Things*) e R2X (*Robots to Everything*). Foram explorados os aspectos teóricos e infraestruturais do protocolo LoRa, destacando sua aplicabilidade em cenários industriais e remotos. Além disso, detalhou-se a comunicação baseada no Data Distribution Service (DDS) em redes LAN, demonstrando a viabilidade de sua expansão utilizando a tecnologia LoRa para atender a sistemas robóticos distribuídos em áreas de difícil acesso.

Adicionalmente, discutiram-se os desafios relacionados ao controle do fator de espalhamento espectral (SF) em dispositivos móveis, evidenciando as limitações do LoRa em ambientes dinâmicos e as soluções potenciais para otimizar a comunicação em cenários de mobilidade.

O próximo capítulo detalha a metodologia de implementação do sistema de navegação autônoma proposto. Será apresentada a integração do controle de espalhamento espectral no módulo de comunicação LoRa, bem como o desenvolvimento de um algoritmo para transmissão de dados sensoriais baseados em tópicos padrão DDS. Essa abordagem visa expandir a área de cobertura do sistema, garantindo uma comunicação eficiente e confiável em ambientes desafiadores e de baixa infraestrutura de conectividade.

CAPÍTULO 5

Proposta do Sistema Modular de Navegação Autônoma

Este capítulo descreve a metodologia de implementação da proposta desta tese: um sistema de navegação autônoma inteligente composto por quatro principais módulos/algoritmos: i) **Módulo de Controle de Navegação e Fusão de Sensores**, ii) **Módulo de Aceleração de Aprendizado**, iii) **Módulo de Comunicação Colaborativa** e iv) **Módulo de Segurança**. Inicialmente é apresentada uma visão geral da proposta de navegação, onde se discute a metodologia de testes, se simulação ou testes reais práticos. Em seguida, cada um dos cinco módulos que compõem o sistema de navegação proposto é detalhado em suas respectivas seções.

5.1 Proposta de Sistema de Navegação Autônoma

A Figura 5.1 ilustra o diagrama de blocos com cada elemento do sistema de navegação proposto. Observa-se a presença de quatro níveis de cores no diagrama do sistema de navegação autônoma, cada um representando diferentes módulos e funcionalidades:

- **Blocos azuis**: Representam os elementos do Módulo de Controle de Navegação e Fusão de Sensores, responsável por controlar os atuadores durante a navegação autônoma. Este é um modelo de aprendizado por reforço, onde o método de Fusão de Sensores está integrado ao algoritmo, permitindo a tomada de decisões baseada em múltiplas fontes de dados sensoriais.
- **Blocos magenta (rosa)**: Indica o Módulo de Segurança, cuja função é evitar colisões diretas com pessoas no ambiente em que o robô navega. Este módulo é crucial para garantir a segurança tanto do robô quanto dos seres humanos ao seu redor.
- **Blocos amarelos**: Representa o Módulo de Aceleração de Aprendizado, que tem a função de acelerar a curva de aprendizagem do algoritmo de controle e fusão de sensores. Este módulo utiliza uma técnica baseada em predições por Filtro de

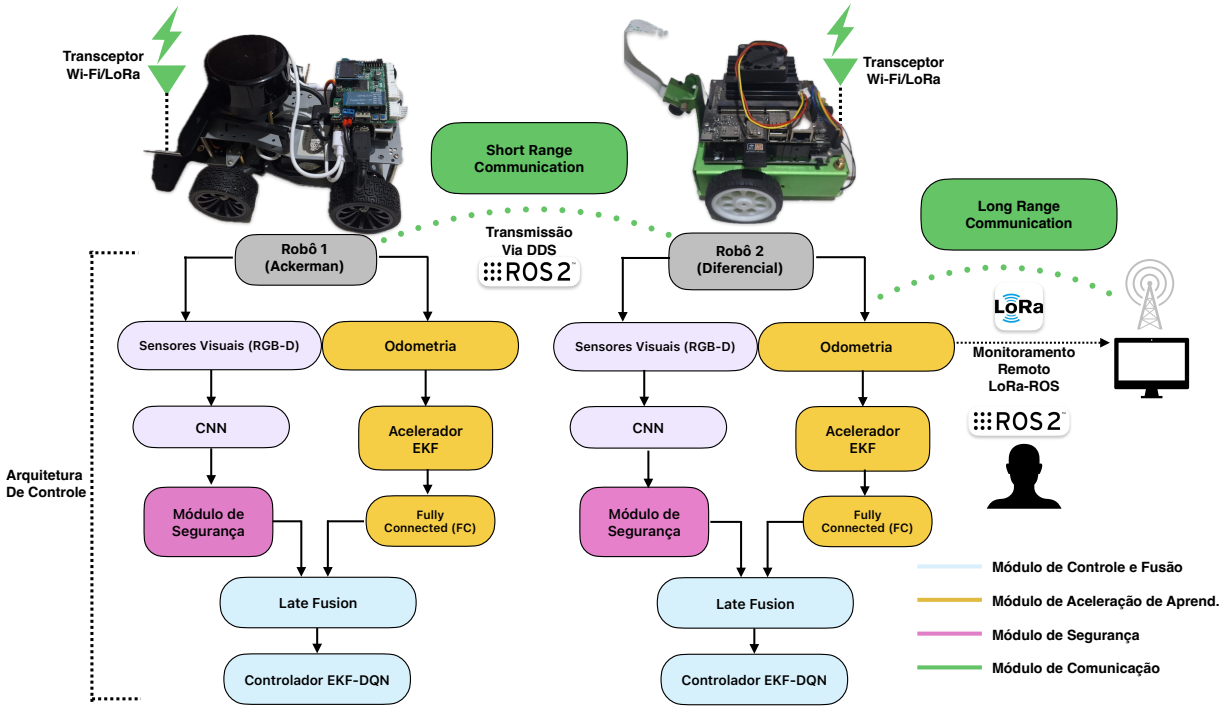


Figura 5.1: Sistema de Navegação Autônoma Proposto - Autoria Própria.

Kalman para otimizar o desempenho do sistema de navegação e aumentar a sua performance em um tempo reduzido.

- **Blocos verdes**: Ilustram o Módulo de Comunicação Sem Fio que possibilita a comunicação IoRT com sistemas ciberfísicos (*CPS*). Este módulo é essencial para permitir telemetria em ambientes industriais e também para viabilizar a atuação multiagente, permitindo que robôs cooperem tanto em cenários de longa distância (*Long Range*) quanto em curtas distâncias (*short range*). No modo *Long Range*, a comunicação entre robôs distribuídos é realizada por meio do protocolo LoRaWAN, que permite a transmissão de dados sensoriais mesmo em ambientes com infraestrutura de rede limitada. Já no modo *Short Range*, é utilizado o middleware DDS (*Data Distribution Service*) sobre rede Wi-Fi, que promove a troca eficiente de informações entre robôs vizinhos, otimizando o desempenho da navegação autônoma e a coordenação local entre agentes.

Ainda na Figura 5.1, observa-se que os dados provenientes da odometria dos robôs móveis autônomos (AMRs) através dos blocos amarelos, alimentam dois módulos fundamentais: o Módulo de Controle e Fusão de Sensores e o Módulo de Aceleração de Aprendizado. Cada um destes módulos são compostos por **Algoritmos Computacionais**, respectivamente o Algoritmo de Fusão de Sensores e o Algoritmo de Aceleração de Aprendizado. O Algoritmo de Aceleração de Aprendizado, descrito em

[Bezerra, Vieira e Soares 2024], tem como objetivo principal otimizar (ou acelerar) a eficiência do processo de aprendizado por reforço do controlador, antecipando estados futuros e prováveis por meio do Filtro de Kalman Estendido (EKF), inserindo estes estados na memoria de repetição. Já o Algoritmo de Fusão de Sensores, apresentado em [Bezerra, Vieira e Carneiro 2023], representa o núcleo decisório do sistema, sendo responsável por combinar informações de múltiplas fontes sensoriais, tanto visuais quanto não visuais, para gerar o sinal de controle que aciona os atuadores (motores) dos robôs. Algoritmo de Fusão unifica dados heterogêneos de sensores, como câmeras RGB-D (visuais) e sensores inerciais, encoders e GPS (não visuais). Os algoritmos EKF-DQN (aceleração) e Fusão de Sensores foram desenvolvidas especificamente para este sistema de navegação autônoma, sendo ambas contribuições originais desta pesquisa.

O Módulo de Comunicação é um componente essencial da arquitetura proposta, permitindo que os Robôs Móveis Autônomos (AMRs) operem de forma colaborativa em uma rede robótica inteligente, dentro do contexto da Internet das Coisas Robóticas (IoRT). Este módulo se desdobra em duas abordagens complementares de comunicação: ***Short Range*** (curta distância) e ***Long Range*** (longa distância), cada uma atendendo a diferentes demandas operacionais do sistema.

A comunicação de *Long Range* é implementada por meio do protocolo LoRaWAN, possibilitando o envio de dados sensoriais provenientes dos robôs para estações remotas de monitoramento. Essa abordagem, contribuição desta tese descrita em [Bezerra, Cardoso e Vieira 2025], é particularmente desafiadora, uma vez que o LoRa é um protocolo de banda estreita, com limitações na transmissão de cargas úteis de alto volume (*high payload*), como é o caso de sensores LiDAR e câmeras RGB-D.

Para contornar essa limitação, desenvolve-se um algoritmo de compressão baseado em *autoencoders*, capaz de reduzir a dimensionalidade dos dados sensoriais. Com isso, torna-se viável a transmissão eficiente de dados em redes LPWAN (*Low Power Wide Area Network*) através da transmissão do conceito de "espaço latente sensorial", mantendo a integridade da informação crítica para navegação e monitoramento remoto.

Além disso, propõe-se também no Módulo de Comunicação o algoritmo de Taxa Forçada de Dados (TFD) [Bezerra et al. 2024], uma variação do mecanismo ADR (*Adaptive Data Rate*) nativo do LoRaWAN. Esse algoritmo permite controlar o espalhamento espectral (*Spreading Factor* - SF) de forma mais eficaz para dispositivos em movimento, o que representa uma inovação, considerando que o LoRa foi originalmente concebido para dispositivos estáticos. A rede foi testada em ambiente real, com rádios transceptores RA-08H e *gateways* SX1302, utilizando microcontroladores RP2040, possibilitando a construção de um cenário dinâmico de comunicação do tipo *Robot-to-Everything* (R2X). Métricas como RSSI e SNR foram analisadas para validar o desempenho da comunicação com robôs em movimento real.

A comunicação de curta distância (*Short Range*) visa possibilitar cooperação multi-robô por meio de redes locais baseadas no *middleware* DDS (*Data Distribution Service*), sistema de comunicação nativo do ROS2. Para isso desenvolve-se o algoritmo $MARL_{Q_{DDS}}$ (*Multi-Agent Reinforcement Learning* - MARL), que permite o compartilhamento de experiências entre agentes vizinhos em um ambiente de aprendizado por reforço distribuído. Esta contribuição, também original desta tese, foi submetida e aceita para publicação no ***Congresso Brasileiro de Telecomunicações e Processamento de Sinais*** (SbrT - 2025). O artigo encontra-se em fase de publicação nos anais.

Nesta abordagem, cada robô é responsável por compartilhar (publicar) informações e receber (assinar) dados relevantes através de tópicos de comunicação definidos na rede ROS2. Esses tópicos funcionam como canais nomeados, nos quais os robôs podem enviar ou escutar mensagens específicas, como por exemplo, dados de sensores ou estados, recompensas ou até mesmo ações sugeridas por suas redes neurais. Ao publicar nesses canais, um robô disponibiliza seus dados para os demais. Ao assinar, ele passa a receber automaticamente todas as mensagens transmitidas nesse tópico. Essa distribuição de informações entre os robôs permite a disseminação de conhecimento aprendido individualmente, funcionando como uma espécie de rede neural coletiva e crítica. Como resultado, o sistema se torna mais robusto, com uma convergência mais rápida do aprendizado e uma capacidade aprimorada de adaptação a mudanças no ambiente.

Cada subseção deste capítulo é dedicada à descrição e validação dos módulos do sistema de navegação autônoma proposto. Cada módulo, controle, fusão sensorial, aprendizado, comunicação de curto alcance e comunicação de longo alcance, emprega uma metodologia de testes específica, adequada às suas características funcionais e à disponibilidade de recursos.

Nem todos os módulos puderam ser validados em cenários práticos/reais devido a limitações de infraestrutura e equipamentos no momento da implementação. Dessa forma, alguns testes foram conduzidos em ambientes simulados com o auxílio de ferramentas como o CoppeliaSim e o ROS2, ferramentas estas que são amplamente reconhecidas na comunidade científica por sua elevada fidelidade em relação a ambientes reais. Outros testes foram implementados e avaliados em experimentações práticas, com robôs físicos e sistemas embarcados e rádios de comunicação. Essa divisão permitiu um equilíbrio entre rigor técnico e viabilidade prática, mantendo o foco na comprovação funcional das abordagens desenvolvidas.

5.2 Ambientes de Experimentação - Simulação e Testes Práticos

O simulador escolhido para o desenvolvimento dos testes envolvendo navegação autônoma foi o *CoppeliaSim V-REP* [Rohmer, Singh e Freese 2013]. Trata-se de uma plataforma amplamente consolidada no meio acadêmico, capaz de reproduzir com alto grau de fidelidade comportamentos físicos, modelos robóticos e interações com o ambiente.

Vale destacar que, embora o *CoppeliaSim V-REP* seja extremamente eficiente na simulação de sensores, atuadores e ambientes dinâmicos, ele não possui suporte nativo à simulação de redes sem fio LPWAN, como o LoRaWAN. Isso se deve às particularidades do protocolo, principalmente relacionadas à sua camada física, como controle de espalhamento espectral, atenuação por distância, perda de pacotes e comportamento sob mobilidade aspectos que são difíceis de serem simulados com fidelidade sem dispositivos físicos. Por esse motivo, todas as validações referentes à comunicação *Long Range*, baseadas no protocolo LoRaWAN, foram realizadas por meio de testes práticos em ambiente real, com a implementação física de rádios e microcontroladores, conforme detalhado na Seção 5.5.

Por outro lado, a comunicação *Short Range*, baseada no uso do middleware DDS integrado ao ROS2, foi parcialmente testada com o suporte do simulador. Embora o *CoppeliaSim* não possua integração nativa com o DDS, ele foi utilizado para executar os testes de navegação autônoma, fornecendo os dados sensoriais (ex: imagens, odometria) necessários para que os algoritmos de comunicação distribuída, como o algoritmo de compartilhamento de experiências multi-robô $MARL_{Q_{DDS}}$, fossem validados sobre a rede de comunicação local do ROS2. Neste caso, a simulação atuou como geradora dos dados, enquanto o ROS2 (executado externamente ao simulador) foi responsável pela troca real de mensagens entre nós e robôs simulados, reproduzindo cenários semelhantes aos utilizados em ambientes reais.

Portanto, o *CoppeliaSim* é utilizado como núcleo de simulação dos principais módulos do sistema de navegação autônoma, incluindo o **Módulo de Navegação e Fusão, o Módulo de Segurança e o Módulo de Aceleração de Aprendizado**. A avaliação da comunicação é conduzida em duas frentes distintas: no contexto de redes *Long Range*, por meio de testes práticos com a tecnologia LoRaWAN, e no contexto de redes *Short Range*, explorando a troca de informações entre robôs em uma rede local (WiFi-LAN) utilizando o middleware DDS com auxílio do ROS2.

O *CoppeliaSim* permite a modelagem e controle de robôs móveis com diversos graus de liberdade, sensores visuais e inerciais, e a simulação de ambientes industriais ou urbanos. No contexto desta tese, desenvolve-se um cenário que representa o protótipo de ambiente fabril, contendo quatro estações de trabalho (*Workstations*), obstáculos fixos e

pessoas móveis.

Uma das principais vantagens da plataforma está em sua capacidade de simular ambientes dinâmicos. Foram incorporados pessoas móveis, que possibilitam testar o sistema de navegação autônoma em situações realistas de tráfego compartilhado. A simulação permite que o robô realize tarefas como navegação entre estações de trabalho, reconhecimento de objetos, tomada de decisões autônomas baseadas em sensores visuais (RGB-D), e principalmente, evitar colisões com obstáculos estáticos e dinâmicos.

O simulador oferece uma Interface de Programação de Aplicações (API) remota que permite o controle completo do ambiente a partir de linguagens externas. Nesta tese, utiliza-se exclusivamente a linguagem *Python* para desenvolver todos os algoritmos de controle, fusão sensorial e aprendizado por reforço. Essa escolha se justifica pela ampla disponibilidade de bibliotecas de inteligência artificial, como *Keras* [Chollet et al. 2015], utilizado para o treinamento de redes neurais profundas e compressão de dados sensoriais.

A Figura 5.2 ilustra o ambiente de testes desenvolvido, com robô autônomo, sensores simulados, obstáculos e as estações de trabalho (*Workstation*) modeladas no *CoppeliaSim*.

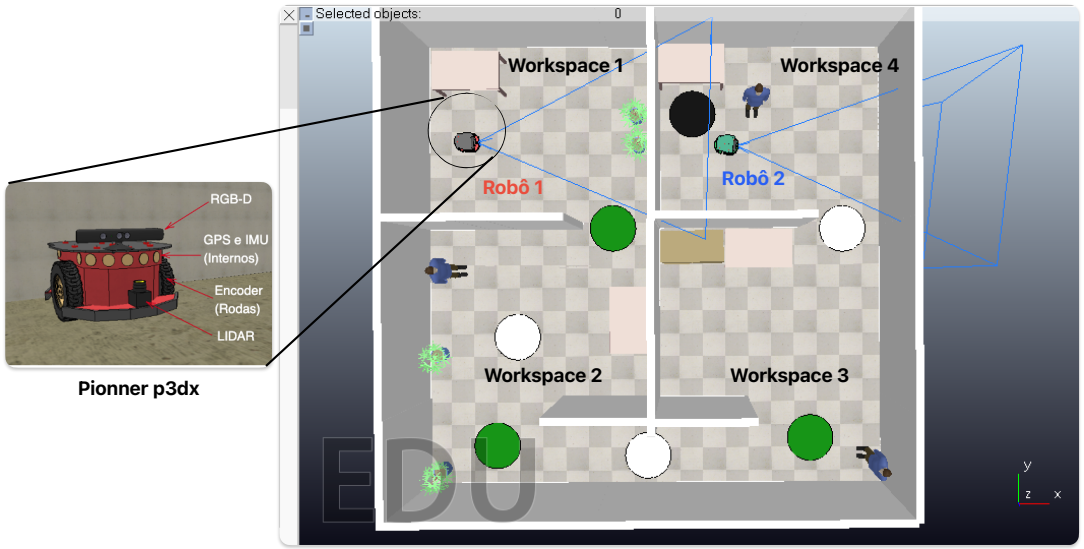


Figura 5.2: Ambiente de Testes - Autoria Própria.

No ambiente representado é possível verificar a presença circunferências indicadas pelas cores: i) verde, ii) branco e iii) preto. Essas circunferências são apenas pontos de controle para detectar a passagem do robô por aquele determinado ambiente, auxiliando no cálculo da função de recompensa. Durante os testes, o AMR deve navegar entre os diferentes workspaces representados no ambiente simulado, evitando colisões com pessoas e obstáculos distribuídos ao longo do percurso. O objetivo da missão é que o

robô consiga sair, por exemplo, do Workspace 1 e alcançar o Workspace 4, ou vice-versa, utilizando estratégias de navegação inteligente, sem infringir regras de segurança ou sair das rotas delimitadas. Quando o AMR realiza esse trajeto com sucesso, considera-se que a missão foi completada. O ambiente foi projetado para representar um cenário dinâmico, contendo pessoas em movimento e barreiras físicas, exigindo do sistema de navegação um alto grau de percepção e tomada de decisão. O AMR não possui conhecimento do mapa do ambiente e nem planejamento de trajetória pré definida. Ele deve aprender a navegar a partir de uma série de experimentações, o que é típico do aprendizado por reforço.

Vale destacar algumas observações importantes em relação à proposta desta tese. Embora seja possível utilizar o consagrado algoritmo de navegação do ROS2, o **Navigation 2 (Nav2)** [Macenski et al. 2020], trata-se de uma abordagem conceitualmente distinta. O Nav2 depende de um mapa prévio do ambiente, o qual é utilizado para gerar trajetórias de controle e navegação do AMR. Por outro lado, a proposta aqui apresentada baseia-se em um **sistema de aprendizado por reforço** com navegação *mapless*, ou seja, que dispensa o conhecimento prévio do ambiente. Nesse modelo, o AMR aprende a navegar de forma autônoma a partir de suas próprias interações e experiências no ambiente.

No simulador, o modelo de AMR selecionado para a execução dos testes é o diferencial, especificamente o modelo **Pioneer 3DX**, conforme apresentado na Figura 5.2. Os sensores implementados incluem: câmera RGB-D, GPS e IMU integrados ao *hardware* do robô, além do encoder acoplado às rodas.

Adicionalmente, utiliza-se o LIDAR, um sensor amplamente empregado para localização e mapeamento simultâneo (SLAM). No contexto deste trabalho, o LIDAR é utilizado para mapear o ambiente e auxiliar no monitoramento do posicionamento do robô. Embora não seja um sensor obrigatório para o algoritmo de controle proposto, sua inclusão enriquece a visualização e proporciona uma referência adicional para a navegação e a localização precisa do AMR.

5.3 Módulo de Aceleração de Aprendizado

De acordo com [Ahumada, Nettle e Solis 2013], um dos problemas do aprendizado por reforço é a velocidade de convergência. Como o agente aprende por iterações, se o espaço de exploração, ou seja, o espaço em que o agente interage com o ambiente for altamente dimensional (muitos estados possíveis) é necessário ampliar o número de episódios de treinamento e consequentemente torná-lo mais dispendioso. Esta situação pode ser crítica para o agente que necessita de um aprendizado rápido, ou ainda, necessita se adaptar ligeiramente a novas condições de navegação, como por exemplo a mudança de um percurso, novos obstáculos entre outros.

Desta forma, o Algoritmo de Aceleração de Aprendizado tem como objetivo propor um método de aceleração de aprendizagem aplicado à redes DQN em navegação autônoma [Bezerra, Vieira e Soares 2024]. A hipótese principal é que algoritmo de aceleração aumentaria a convergência da curva de aprendizado do robô garantindo que o mesmo aprenda as ações adequadas para o controle de navegação de forma mais rápida.

Primeiramente, apresenta-se a proposta de adaptação do Filtro de Kalman Estendido (EKF) ao problema de navegação. Uma das principais premissas para utilização do KF ou EKF é conhecer a dinâmica do sistema, ou seja modelar o sistema no espaço de estados. A dinâmica do AMR, inicialmente apresentada na Equação (2-16) da Seção 2, é adaptada para aplicação do KF por meio das equações matriciais abaixo:

$$\begin{bmatrix} x_k \\ y_k \\ z_k \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x_{k-1} \\ y_{k-1} \\ z_{k-1} \end{bmatrix} + \begin{bmatrix} \cos\theta\ t_k & 0 \\ \sin\theta\ t_k & 0 \\ 0 & t_k \end{bmatrix} \times \begin{bmatrix} v_{k-1} \\ \omega_{k-1} \end{bmatrix} \tag{5-1}$$

Na equação acima, cada variável representa uma componente do modelo de movimento do robô móvel autônomo (AMR), adaptado ao espaço de estados. A seguir, descrevem-se os significados de cada uma:

- x_k, y_k, z_k: Posição estimada do robô no instante de tempo k, sendo x e y as coordenadas no plano e z a orientação angular (geralmente associada ao ângulo θ).
- x_{k-1}, y_{k-1}, z_{k-1}: Posição e orientação do robô no instante de tempo anterior $(k-1)$.
- v_{k-1}: Velocidade linear estimada do robô no instante anterior $(k-1)$, geralmente obtida por odometria ou sensores inerciais.
- ω_{k-1}: Velocidade angular (taxa de variação do ângulo) no instante anterior $(k-1)$.
- θ: Ângulo de orientação do robô no plano, utilizado para calcular a direção do movimento.
- t_k: Intervalo de tempo entre a leitura $k-1$ e k (tempo de amostragem do sistema).

Ao comparar a Equação (5-1) com a formulação clássica de um sistema dinâmico no espaço de estados:

$$\mathbf{x}_k = \mathbf{A}\mathbf{x}_{k-1} + \mathbf{B}\mathbf{u}_{k-1} \tag{5-2}$$

é possível identificar, de forma implícita, a presença das matrizes **A** e **B**, bem como dos vetores de estado **x** e de controle **u**.

No caso do modelo de navegação proposto, o vetor de estado $\mathbf{x}_k$ é composto pelas variáveis x_k, y_k e z_k, representando respectivamente as coordenadas cartesianas e a orientação angular do robô no instante k. A matriz **A** assume a forma de uma matriz identidade, refletindo a preservação dos estados na ausência de entrada de controle. Já a

matriz de controle **B** expressa a dinâmica do sistema, incorporando a dependência não linear da orientação angular θ para o deslocamento nas direções x e y. Por fim, o vetor de controle $\mathbf{u}_{k-1}$ é formado pela velocidade linear v_{k-1} e a velocidade angular ω_{k-1} do robô, que são as entradas diretamente controláveis no sistema.

Imaginemos agora um veículo chegando próximo a uma curva. Espera-se que o DQN note a presença desta curva quando seu estado for percebido por um sensor e assim a melhor ação será tomada para aquele determinado estado. Para tal, propomos a seguinte função de recompensa para o algoritmo de aprendizagem da DQN:

$$R = \begin{cases} dist_{k-1} - dist_k & \text{se está vivo} \\ -1, & \text{se colidiu} \\ +1, & \text{se atingiu o alvo} \end{cases} \tag{5-3}$$

$dist_k$ e $dist_{k-1}$ são respectivamente as distâncias do robô ao alvo no instante k. Esse cálculo permite recompensar ações que reduzem cada vez mais a distância até o alvo. Se o robô colidir com uma parede do ambiente, ele sofre uma penalidade de -1 ponto. Se o alvo local for atingido, o agente recebe uma recompensa de 1 ponto.

Como mencionado anteriormente, o método DQN requer uma representação adequada do estado do robô para tomar decisões de forma eficiente. Caso fossem utilizadas apenas as coordenadas cartesianas obtidas por odometria ou GPS, mesmo com a filtragem pelo Filtro de Kalman (EKF), o agente não teria acesso à orientação do robô, uma informação essencial para distinguir, por exemplo, se o robô está executando uma curva ou se movimentando em linha reta. Para contornar essa limitação, é fundamental incluir também as velocidades linear (v_L) e angular (ω) do robô, que podem ser obtidas a partir dos sensores de odometria (encoders) e da IMU, respectivamente. Essas variáveis permitem capturar a dinâmica do movimento de forma mais completa. Assim, nos testes realizados com o Algoritmo de Aceleração de Aprendizado, propõe-se que o vetor de estado s do DQN seja composto por:

$$s = [x, y, v_L, \omega] \tag{5-4}$$

Neste contexto, x e y representam as coordenadas cartesianas da posição do robô, enquanto v_L e ω correspondem às suas velocidades linear e angular, respectivamente. Esses dados podem ser obtidos por meio de diferentes sensores de posicionamento: o GPS fornece coordenadas absolutas, enquanto a odometria, composta por sensores como encoders e IMU, fornece informações sobre velocidade e orientação do robô. A combinação dessas variáveis resulta em uma representação mais completa e informativa do estado do agente.

Cabe destacar que a coordenada z não é incluída no vetor de estado s, pois a

navegação do robô ocorre em um plano bidimensional. No entanto, a variação angular em torno do eixo z (*yaw*) e representada por ω é de grande relevância, pois determina a direção do movimento do robô ao longo do plano.

O Algoritmo 5 apresenta os passos do EKF-DQN proposto. O EKF-DQN permite que o estado do robô (informações de posição e velocidades) possa ser estimado um ou mais passos antes das medições sensoriais reais, graças à predição do movimento do veículo realizada pelo EKF. A Figura 5.3 ilustra o princípio do EKF para predição de estados futuros.

Algorithm 5 Algoritmo de Aceleração de Aprendizado EKF-DQN

Inicializar o **agente** DQN; Iniciar a Matriz de Covariança (P) Definir número de $EPISODIOS$;
Leitura dos Sensores (inicializa s)
while *done* = *False* **do**
  `/* Estado Terminal = False */`
  Dado s, **EKF** estima o estado s_{k+1}
  Gera uma Probabilidade p
  if $\epsilon > p$ **then**
    Seleciona uma ação a aleatória;
  if $\epsilon < p$ **then**
    $a = \text{argmax}(\hat{Q}(s_{k+1}, a, \theta)$
  Executa uma Ação a no Ambiente e Observa o novo estado s_{k+1} a Recompensa R
  Dado s_{k+1} medido, **EKF** estima o estado s_{k+2}
  Memorização das Experiências (s_{k+1}, a, s_{k+2}, R, $done$) `/* experience replay acelerado */`
  $Q(s,a) = R + \gamma \max Q(s_{k+2}, a)$ `/* estima o alvo da rede neural */`
  do $s = s_{k+1}$
  Atualiza a Matriz de Covariança (P)
performa o MSE e o gradiente descendente ($Q - \hat{Q}(s_{k+1}, a, \theta)^2$)
Decai a probabilidade de exploração ϵ

O **Algoritmo EKF-DQN** propõe uma estratégia de navegação autônoma que integra aprendizado por reforço profundo DQN com filtragem Bayesiana por meio do Filtro de Kalman Estendido (EKF). Inicialmente, o agente DQN é instanciado, assim como a matriz de covariância do EKF e o número de episódios de treinamento. A cada iteração, os sensores do robô fornecem o vetor de estado inicial s, o qual é estimado pelo EKF para gerar uma predição refinada do estado s_{k+1}. Uma probabilidade p é então gerada para decidir entre exploração (ação aleatória) e exploração guiada pela política (seleção da ação a que maximiza $Q(s_{k+1}, a, \theta)$). A ação escolhida é executada no ambiente, e o robô observa o novo estado e a recompensa R recebida. O EKF é novamente utilizado para estimar o próximo estado s_{k+2}, que será armazenado juntamente com a experiência $(s_k, a, s_{k+2}, R, done)$ no buffer de *experience replay*. A função Q é atualizada com base no valor futuro estimado, e os parâmetros da rede neural são ajustados por gradiente

descendente, minimizando o erro quadrático médio entre a previsão $\hat{Q}$ e o valor-alvo da função Q. Por fim, o valor de ϵ, responsável por regular a taxa de exploração, é atualizado gradualmente a cada episódio, enquanto a matriz de covariância do EKF também é ajustada com base nas novas observações. Essa abordagem busca fornecer maior estabilidade na estimativa dos estados e acelerar o processo de aprendizado do agente.

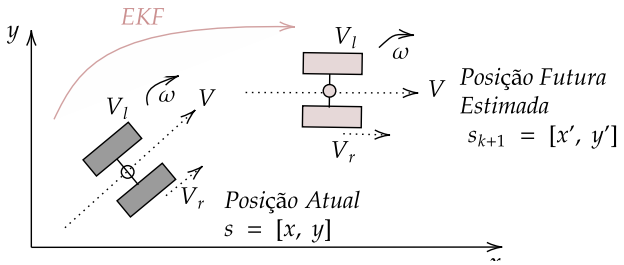


Figura 5.3: Princípio do EKF para predição de estados futuros - Autoria Própria.

A Figura 5.4 ilustra o diagrama de blocos que compara o método proposto (EKF-DQN) com o DQN tradicional. O diagrama ilustra como a predição do estado realizada pelo EKF antecipa e refina a percepção do agente antes da tomada de decisão, permitindo ao DQN atuar com maior acurácia. No DQN tradicional, o estado do robô é estimado diretamente por sensores (em laranja). No EKF-DQN (em azul), os estados são processados pelo Filtro de Kalman Estendido, que gera uma previsão do estado futuro um passo a frente s_{k+1}. Esta estimativa é fornecida ao agente DQN, que então decide a ação a ser tomada. Após a ação ser executada, o novo estado s_{k+2} também é estimado via EKF e armazenado em um buffer de memória.

Em síntese, a integração do Filtro de Kalman Estendido (EKF) com o agente DQN representa uma estratégia promissora para acelerar o processo de aprendizado em tarefas de navegação autônoma. Ao fornecer estimativas mais precisas dos estados futuros, o EKF contribui para a estabilidade e eficiência do treinamento da rede, reduzindo o impacto de leituras ruidosas além de repassar ao agente uma informação a frente (um passo a frente). Desta forma é esperado que o algoritmo EKF-DQN proporcione uma convergência mais rápida e uma política de navegação mais robusta.

Com o objetivo de complementar o desempenho do sistema de navegação autônoma, a próxima seção aborda os métodos de fusão de sensores propostos nesta tese.

5.4 Módulo de Controle e de Fusão de Sensores

O módulo de Controle e Fusão de Sensores é composto essencialmente pelo Algoritmo de Fusão de Sensores, cuja á função é unificar as percepções dos sensores

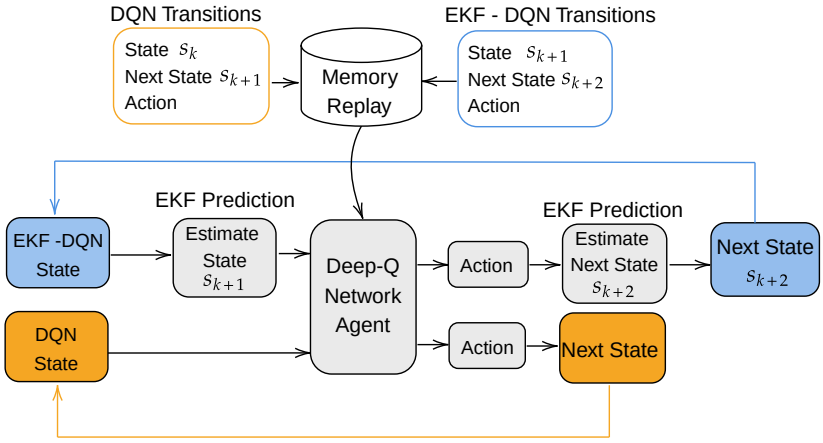


Figura 5.4: Método de Aceleração de Aprendizado Proposto [Bezerra, Vieira e Soares 2024] - Autoria Própria.

não visuais e visuais para uma tomada de decisão única (ação de controle). Conforme discutido na seção anterior, o sistema é acelerado com base nos dados fornecidos pelos sensores não visuais (EKF-DQN), pois essa representação de estado permite a estimativa de um estado futuro, modelando a dinâmica do robô de maneira eficaz. A inclusão de imagens capturadas pela câmera RGB-D aprimora significativamente a percepção necessária para uma ação de controle precisa, fornecendo ao robô uma visão detalhada do ambiente.

Para o desenvolvimento do algoritmo de Fusão de Sensores proposto nesta tese, foram considerados dois métodos amplamente discutidos na literatura: i) *Late Fusion* [Ryu e Kim 2021, Feng et al. 2021] e ii) *Iterative Fusion* [Duanmu et al. 2020]. A proposta desta pesquisa consiste em integrar essas abordagens com redes DDQN, uma variante aprimorada da DQN, visando melhorar a capacidade de decisão em ambientes com múltiplas fontes sensoriais. Além disso, são introduzidas variações do método *Late Fusion*, adaptadas ao contexto de navegação robótica. Dessa forma, duas novas estratégias de fusão sensorial são apresentadas: i) *DDQN-Late Fusion* e ii) *DDQN-Iterative Fusion* [Bezerra, Vieira e Carneiro 2023], ambas desenvolvidas como contribuições originais desta tese.

O Double DQN (DDQN) é uma variante da rede DQN que também permite o uso do EKF-DQN, conforme discutido na seção anterior. Verifica-se que o uso do DDQN proporciona maior estabilidade no treinamento, o que não impede a aplicação do EKF-DQN. A combinação do DDQN com o EKF-DQN resulta em um treinamento mais estável, melhorando a eficiência e a precisão do sistema de navegação.

Para o desenvolvimento do algoritmo e execução dos testes propostos, foram utilizados quatro sensores embarcados no AMR dentro do ambiente simulado CoppeliaSim: i) uma câmera RGB-D, responsável por fornecer informações visuais e de profundidade;

ii) um módulo de posicionamento global (GPS simulado), que oferece coordenadas absolutas; iii) uma unidade de medição inercial (IMU), que provê dados de aceleração e orientação angular; e iv) encoders simulados, que monitoram a rotação das rodas. Embora o CoppeliaSim forneça diretamente a posição absoluta do robô no plano de forma precisa, em ambientes reais essa informação normalmente é estimada por meio da odometria, construída a partir da dos dados dos encoders com a IMU. Dessa forma, o simulador permite reproduzir, de maneira controlada, o comportamento esperado em aplicações físicas.

A Figura 5.5 demonstra graficamente o processo de fusão sensorial denominado *Late Fusion* [Feng et al. 2021]. O robô AMR coleta dados de sensores visuais (RGB-D) e sensores não-visuais (Encoder, IMU e odometria). Os dados visuais são processados por uma rede convolucional (CNN), enquanto os dados não-visuais passam por uma camada totalmente conectada (FC1). As saídas dessas duas redes são então combinadas por meio de uma operação de fusão, indicada no diagrama pelo símbolo de soma (+). Essa operação representa a concatenação ou combinação linear dos vetores de características extraídos de ambas as fontes sensoriais. O vetor resultante alimenta uma rede DDQN com uma camada final totalmente conectada (FC), responsável por estimar os valores de ação Q' correspondentes a cada possível ação do robô. Dessa forma, o sistema integra eficientemente informações heterogêneas, permitindo ao agente aprender e agir com base em uma percepção mais completa do ambiente.

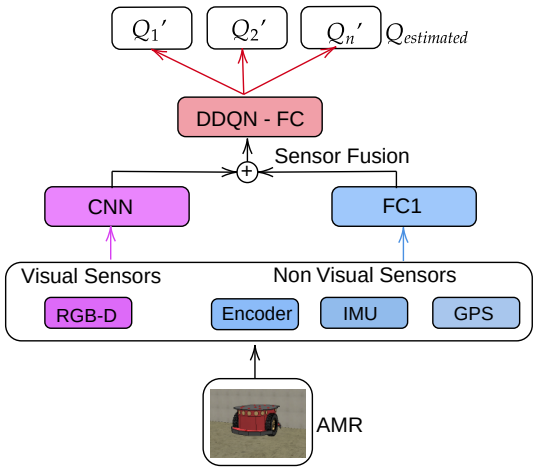


Figura 5.5: Fluxograma do Algoritmo de Controle e Fusão: Aprendizado por Reforço Baseado em DDQN Late Fusion - Autoria Própria.

O *Iterative Fusion*, conforme explicado no Capítulo 2, é um método proposto por [Duanmu et al. 2020] e originalmente aplicado na área médica (exames clínicos e de imagem). No *Iterative Fusion* original, exames clínicos relacionados a imagens são processados por uma rede CNN, onde os autores selecionam a VGG-13 para este fim,

uma arquitetura já consagrada na literatura. Os exames relacionados a dados tabulares, isto é, dados de exames clínicos como os de sangue, dados categóricos como sexo entre outros são processados por uma rede *Fully Connected*. A fusão, ou combinação dos dados, ocorre especificamente nas camadas convolucionais intermediárias da VGG-13, de forma que a rede tabular (*Fully Connected*) multiplica seus valores de saída pelos mapas de características obtidos pela CNN. O autor chama esta técnica como *Channel Wise Multiplication* e afirma que são criados mapas convolucionais interativos, semelhante a mecanismos de atenção. Nos testes desenvolvidos na área média, a técnica superou em termos de performance uma *Late Fusion* Tradicional.

Nesta tese, a ideia de fusão iterativa proposta por [Duanmu et al. 2020] é adaptada para o contexto de navegação autônoma em ambientes reais, que demandam um processamento mais leve e eficiente. A principal modificação está na arquitetura da rede convolucional (CNN) utilizada para processar imagens RGB-D. Diferente do cenário médico apresentado por [Duanmu et al. 2020], que utilizava imagens de alta resolução em exames de ressonância magnética e uma arquitetura profunda como a VGG-13, optou-se aqui por uma CNN mais rasa (*shallow*). Essa decisão baseia-se na menor complexidade visual das cenas observadas pelo robô móvel autônomo (AMR), o que permite uma extração de características suficiente com uma rede mais leve e adequada ao hardware embarcado. A mesma CNN é empregada nas abordagens de *Late Fusion* e *Iterative Fusion*, garantindo consistência nos testes comparativos. A Tabela 5.1 apresenta a configuração adotada para a rede convolucional utilizada.

Tabela 5.1: Estrutura da Rede CNN utilizada para o processamento da imagem RGB-D.

Camada CNN	**Número de Filtros**	**Kernel**	**Stride**
1ª Camada	16	(5, 5)	5
2ª Camada	32	(3, 3)	2
3ª Camada	32	(2, 2)	2

A rede CNN é responsável por produzir um vetor de características com 512 elementos, oriundo da última camada convolucional *flattenizada* (comprimida). Em paralelo, os dados não-visuais são processados por uma rede densa composta por duas camadas totalmente conectadas (FC1, FC2), dispondo a dimensionalidade dos dados em um vetor de 32 elementos. A fusão entre os dados visuais e não-visuais ocorre por meio da concatenação direta desses vetores (512 + 32), gerando uma única entrada com 544 características que alimenta a rede DDQN-FC, responsável por estimar os valores $\hat{Q}(s, a, \theta)$. A Tabela 5.2 resume a estrutura da rede de fusão sensorial, desde os blocos CNN e FC até a combinação final.

Na estratégia de fusão proposta po [Duanmu et al. 2020] ocorre a multiplicação elemento a elemento, conhecida como *Channel-Wise Multiplication* entre as camadas *fully connected* com as camadas convolucionais. Nesta técnica, para cada camada con-

Tabela 5.2: Arquitetura da Rede de Fusão Sensorial com Concatenação Direta (*Late Fusion*).

Bloco	Camada	Neurônios	Descrição
CNN (RGB-D)	Flatten	512	Saída da CNN aplicada à imagem RGB-D
FC (Sensores Não-Visuais)	FC1 FC2	32 32	Processamento inicial dos dados Encoder, IMU e GPS Saída final da rede não-visual
Fusão		**512 + 32 = 544**	Concatenação (entrada para DDQN-FC)
FC Final (DDQN)	FC-Ação	3	Camada de saída com 3 neurônios representando as ações possíveis

volucional da CNN (*Layer 1, Layer 2 e Layer 3*), é realizada uma multiplicação ponto a ponto com a saída correspondente da rede FC (FC1, FC2 e FC3). A Figura 5.5 ilustra essa abordagem, destacando a correspondência direta entre os canais visuais e os canais derivados dos sensores não-visuais. Essa multiplicação atua como um mecanismo de atenção, modulando a resposta da CNN com base na percepção não-visual, o que pode melhorar a robustez da rede em cenários com ruído visual ou ambiguidade perceptual. A Figura 5.6 apresenta o esquema do método de fusão *Iterative Fusion* adaptado para o problema de navegação.

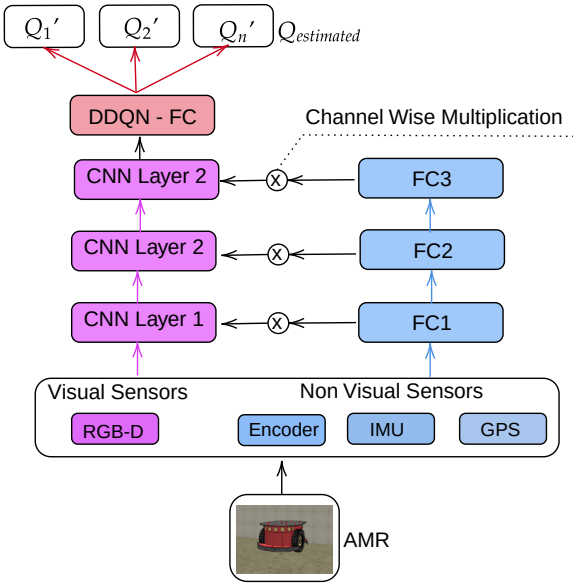


Figura 5.6: Aprendizado por Reforço Baseado em DDQN Iterative Fusion - Autoria Própria.

Em ambas as abordagens, a política de decisão do agente é aprendida por meio do algoritmo *Double Deep Q-Network* (DDQN). O objetivo é estimar a função de valor de ação $Q(s, a)$, que representa o retorno esperado ao se tomar a ação a no estado s, seguindo uma determinada política. A rede neural parametrizada por θ estima esse valor como $\hat{Q}(s, a, \theta)$.

Durante o treinamento, a rede é otimizada minimizando a diferença entre o valor estimado $\hat{Q}(s, a, \theta)$ e um *valor-alvo* calculado por:

$$Q_{\text{target}} = R + \gamma \cdot Q(s', \arg\max_{a'} \hat{Q}(s', a', \theta), \theta^{-}), \tag{5-5}$$

onde:

- R representa a recompensa recebida ao executar a ação a no estado s;
- s' é o novo estado observado após a ação;
- γ é o fator de desconto, responsável por ponderar as recompensas futuras;
- $\arg\max_{a'} \hat{Q}(s', a', \theta)$ é a ação que maximiza o valor da rede principal;
- $Q(s', \cdot, \theta^{-})$ utiliza os pesos da *rede-alvo* congelada (θ^{-}) para avaliar a ação escolhida.

A utilização de duas redes distintas, uma para a escolha da melhor ação e outra para a avaliação do valor dessa ação, reduz o viés de superestimação, comum no DQN original.
A função de perda utilizada para o treinamento é definida como:

$$\mathcal{L}(\theta) = \left(Q_{\text{target}} - \hat{Q}(s, a, \theta)\right)^2, \tag{5-6}$$

a qual representa o erro quadrático médio (*Mean Squared Error, MSE*) entre a estimativa atual da rede principal e o valor-alvo calculado. Essa função é minimizada iterativamente usando técnicas de descida do gradiente, como o otimizador *Adam*.

As duas variantes propostas, *DDQN-Late Fusion* e *DDQN-Iterative Fusion*, diferem quanto à estratégia de fusão sensorial (visão e odometria), mas compartilham a mesma base de treinamento DDQN. Ambas foram testadas no ambiente simulado representado na Figura 5.2, e sua performance foi avaliada com base em:

- Convergência do aprendizado (curvas de recompensa média por episódio);
- Robustez da política de navegação frente a obstáculos e ruído;
- Eficácia da fusão sensorial em comparação com versões unimodais (somente visão).

Após a apresentação das estratégias de fusão sensorial para aprimorar a percepção do robô móvel autônomo (AMR), esta tese avança agora para a discussão do **Módulo de Segurança**, responsável por garantir a integridade do sistema durante a navegação.

5.5 Módulo de Segurança Anti-Colisão

O principal objetivo de utilizar o módulo de segurança é evitar colisões do robô móvel com pessoas em movimento durante o processo de navegação [Bezerra, Vieira e Soares 2024]. Dessa forma, o algoritmo do módulo de segurança é inserido na arquitetura de navegação autônoma para apoiar o controlador principal

que utiliza o algoritmo de fusão de sensores e EKF-DQN. Este módulo de segurança é baseado em um processo de aprendizado supervisionado, mais especificamente, um classificador binário que utiliza visão computacional.

Neste trabalho, a ideia é que toda vez que uma pessoa se aproximar do robô, um processo de frenagem ocorra para evitar uma colisão. Se a pessoa se afastar ou sair do campo de visão da câmera, o controlador principal EKF-DQN começa a funcionar novamente. O módulo de segurança tem prioridade para realizar frenagens sobre comandos do controlador principal.

O módulo de segurança utiliza a rede neural convolucional (CNN) ResNet-50 [He et al. 2015] para detectar a forma de humanos na cena. Basicamente, a ResNet-50 é usada como um classificador binário, que verifica a existência de pessoas próximas ou distantes (ou nenhuma pessoa) na cena. Como é necessário ter uma noção de profundidade na cena, a câmera RGB-D é usada como sensor visual.

A ResNet-50 foi originalmente projetada para processar imagens RGB de três canais. Neste trabalho, enfrentamos o desafio de adaptar imagens RGB-D para processamento pela arquitetura ResNet-50, que é inerentemente projetada para lidar com imagens RGB de três canais. Para superar essa limitação, empregamos uma técnica específica de transformação de dados dentro do simulador Coppelia VREP. Inicialmente, o simulador gera imagens RGB-D usando uma câmera RGB-D virtual, resultando em dados que contêm tanto informações de cor RGB convencionais quanto um canal de profundidade adicional. O passo crítico é converter essa informação de profundidade em intensidades de cor. O simulador realiza isso mapeando valores de profundidade para um espectro de cor correspondente. Por exemplo, diferentes distâncias são representadas por diferentes tons de cinza ou cores, onde a variação na cor reflete mudanças na profundidade. Esse mapeamento cria uma representação visual da profundidade que pode ser integrada aos canais de cor RGB.

Após a transformação do canal de profundidade em intensidades de cor, procedemos à fusão com os canais RGB existentes, produzindo uma nova imagem RGB. Esta nova imagem não apenas retém a informação de cor original, mas também encapsula a informação de profundidade modificada, agora representada como variações na intensidade da cor. Consequentemente, a imagem resultante, que é uma composição de três canais RGB, pode ser processada pela rede ResNet-50 sem necessidade de ajustes na arquitetura da rede. Esta técnica nos permite utilizar a ResNet-50, uma rede neural convolucional projetada para imagens RGB padrão, em aplicações que requerem análise de informações RGB-D, aproveitando as capacidades avançadas de processamento e reconhecimento de padrões da rede.

A função de saída da rede ResNet50 é uma sigmóide. Se o valor de saída da ResNet50 for maior que 0,9, assume-se que há uma alta chance de haver uma pessoa

próxima ao robô. Se for menor que esse valor, o caminho é considerado livre e o algoritmo EKF-DQN opera o controle de navegação. A ideia é treinar a ResNet-50 com um conjunto de dados de imagens RGB-D. Propomos realizar esse treinamento combinando imagens de pessoas reais neste conjunto de dados com imagens sintéticas de pessoas coletadas no simulador Coppelia.

O conjunto de dados utilizado para o treinamento foi composto por uma combinação de imagens sintéticas, geradas no simulador Coppelia, e imagens reais provenientes do conjunto de dados apresentado em [Luber, Spinello e Arras 2011], capturadas no hall da Universität Freiburg por meio de três sensores Kinect montados verticalmente. Essas imagens incluem pessoas caminhando ou paradas, observadas sob diferentes orientações e níveis de oclusão. Para compor o banco de dados secundário utilizado no treinamento, foram selecionadas 341 imagens, considerando qualitativamente a presença de pessoas em posições próximas e distantes dos sensores, uma vez que essa informação não está explicitamente rotulada. A Figura 5.7 mostra um exemplo de imagem sintética obtida no Coppelia, enquanto a Figura 5.8 apresenta uma imagem real do banco de dados original. Além disso, foram aplicadas técnicas de data augmentation, como RandomAffine, RandomHorizontalFlip e RandomPerspective, visando aumentar a variabilidade e robustez do conjunto de treinamento.

Figura 5.7: Imagens de pessoas sintéticas em cena.

Figura 5.8: Imagens de pessoas reais em cena [Luber, Spinello e Arras 2011].

5.5.1 Fluxo de Operação do Módulo de Segurança

A Figura 5.9 dispõe uma representação do fluxo de operação do módulo de segurança, desde o processo de criação do banco de dados até a implementação (*deploy*) no robô.

A função de custo utilizada é a `nn.BCEWithLogitsLoss`. Esta função de perda combina a aplicação da função sigmoide com a entropia cruzada binária (BCE). A BCE mede a diferença entre as previsões do modelo (após a aplicação da sigmoide) e os rótulos verdadeiros. Matematicamente, a entropia cruzada binária é definida como:

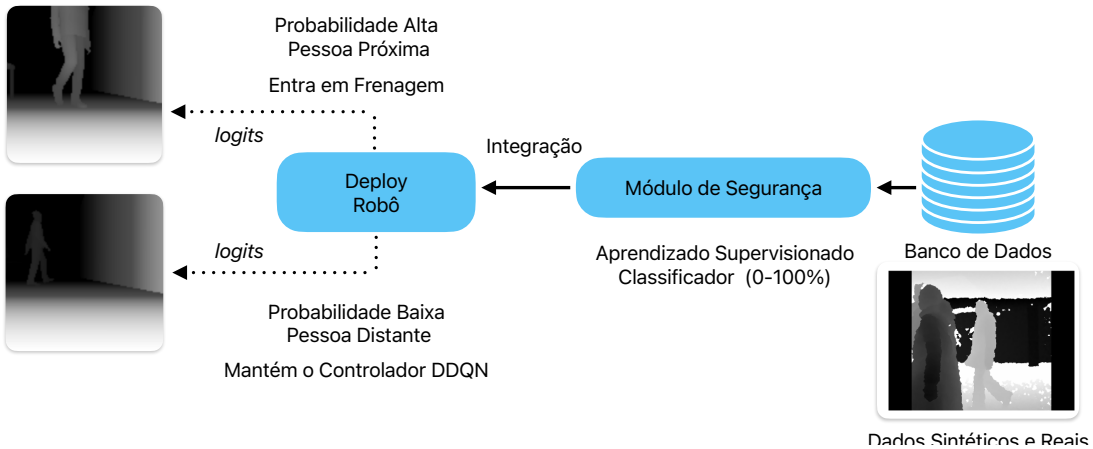


Figura 5.9: Fluxograma de Representação do Módulo de Segurança.

$$L = -\frac{1}{N}\sum_{i=1}^{N}[y_i \log(\sigma(z_i)) + (1 - y_i)\log(1 - \sigma(z_i))] \tag{5-7}$$

Onde:

- N é o número de amostras.
- y_i é o rótulo verdadeiro da amostra i.
- z_i é o logit (saída do modelo antes da sigmoide) para a amostra i.
- $\sigma(z_i)$ é a saída da função sigmoide aplicada ao logit z_i.

A acurácia é uma métrica de desempenho que mede a proporção de previsões corretas feitas pelo modelo em relação ao total de previsões. Para calcular a acurácia, as saídas do modelo são primeiro transformadas em probabilidades utilizando a função sigmoide. Em seguida, um limiar (geralmente 0.5) é aplicado para converter essas probabilidades em previsões binárias (0 ou 1).

A acurácia é então calculada como:

$$\text{Acurácia} = \frac{\text{Número de Previsões Corretas}}{\text{Número Total de Amostras}} \tag{5-8}$$

A função sigmoide é uma função de ativação que mapeia qualquer valor real para o intervalo (0, 1). No contexto de classificação binária, a sigmoide é usada para converter os *logits* (saídas lineares do modelo) em probabilidades. A função sigmoide é definida como:

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{5-9}$$

Onde z é o logit. A saída da sigmoide pode ser interpretada como a probabilidade de uma amostra pertencer à classe positiva (1). Em resumo, o fluxo de operação do módulo de segurança é:

1. **Entrada**: Imagens em escala de cinza capturadas pelo sensor.
2. **Modelo**: As imagens são passadas pelo modelo ResNet-18 modificado.
3. **Logits**: A saída do modelo antes da sigmoide é um valor contínuo (logit).
4. **Sigmoide**: Os logits são transformados em probabilidades pela função sigmoide.
5. **Perda**: A função de custo `nn.BCEWithLogitsLoss` calcula a diferença entre as previsões e os rótulos verdadeiros.
6. **Treinamento**: Os pesos do modelo são ajustados para minimizar a função de custo.
7. **Predições**: Durante a inferência, as probabilidades são transformadas em previsões binárias usando um limiar.
8. **Acurácia**: A acurácia é calculada comparando as previsões binárias com os rótulos verdadeiros.

Com a percepção, tomada de decisão definidas pelos módulos de fusão sensorial, Aceleração de Aprendizado e Anti-Colisão, é necessário garantir que os dados essenciais à navegação e coordenação do robô sejam compartilhados eficientemente com outros agentes ou sistemas externos. Para isso, a próxima seção apresenta o **Módulo de Comunicaçã**o, responsável pela transmissão e recepção de informações entre o robô e elementos da infraestrutura inteligente e ou outros robôs autônomos.

5.6 Módulo de Comunicação: Long Range e Short Range

A comunicação entre robôs e sistemas externos é um elemento fundamental para garantir a conectividade, o monitoramento e a coordenação eficiente de tarefas em aplicações de navegação autônoma. Embora robôs móveis autônomos (AMRs) possam operar de forma descentralizada, ou seja, executando tarefas localmente sem depender de outros agentes, esta tese parte da hipótese de que a comunicação coordenada, quando bem estruturada, é capaz de acelerar a realização de tarefas e aumentar a robustez do sistema. Esta premissa está diretamente alinhada ao conceito de *Internet of Robotic Things* (IoRT), no qual dispositivos robóticos estão conectados a redes que possibilitam troca de informações sensoriais e de controle, seja com outros robôs (R2R), com usuários humanos (R2H), ou com infraestrutura remota (R2I).

Dessa forma, a arquitetura proposta nesta tese incorpora um **módulo de comunicação** projetado especificamente para viabilizar o conceito de *Robot-to-Everything* (R2X), permitindo que o robô compartilhe seu estado, receba comandos remotos e in-

teraja com outros dispositivos ciberfísicos em tempo real, mesmo quando fora de redes locais convencionais.
O módulo de comunicação foi dividido em duas abordagens complementares:

- **Long Range:** voltada para cenários de monitoramento remoto e comunicação com infraestrutura, utilizando o protocolo LoRaWAN, de baixa taxa de transmissão e longo alcance;
- **Short Range:** focada na troca de mensagens entre robôs próximos, viabilizando aplicações de aprendizado multiagente e cooperação descentralizada, utilizando o middleware DDS sobre a infraestrutura do ROS2.

A seguir, descrevem-se em detalhes ambas as abordagens, começando pela estratégia de longo alcance.

5.6.1 Comunicação Long Range com LoRaWAN

O módulo de comunicação *Long Range* foi desenvolvido com o objetivo de permitir a comunicação de dados sensoriais entre robôs móveis autônomos (AMRs) e sistemas de infraestrutura remota, especialmente em cenários onde não há disponibilidade de redes tradicionais como Wi-Fi ou Ethernet. Este módulo materializa a abordagem *Robot-to-Infrastructure* (R2I), sendo essencial para aplicações de monitoramento remoto, como em áreas industriais externas ou regiões de preservação ambiental.

Para viabilizar essa comunicação em ambientes amplos e sem cobertura convencional, foi implementada uma integração entre o sistema operacional robótico ROS2 e a rede LoRaWAN. O protocolo LoRaWAN, amplamente utilizado em redes LPWAN (*Low Power Wide Area Network*), opera com baixa taxa de transmissão e longo alcance, podendo alcançar até 15 km em áreas rurais. Entretanto, possui limitações severas quanto à largura de banda e ao tamanho do *payload*, que pode variar entre 51 e 222 bytes, a depender do *spreading factor* (SF) utilizado.

Dado que sensores como o LiDAR geram grandes volumes de dados — um único varrimento pode gerar aproximadamente 1300 bytes — foi necessário desenvolver um mecanismo de compressão que tornasse viável sua transmissão pela rede LoRaWAN. Para isso, o módulo emprega um algoritmo de codificação baseado em *autoencoders*, que transforma os dados do sensor em uma representação latente compactada, suficientemente pequena para ser transmitida com eficiência, preservando ao máximo a fidelidade do sinal original após a reconstrução no destino.

A arquitetura física do módulo foi implementada com microcontroladores RP2040 e rádios RA-08H [Shenzhen Ai-Thinker Technology Co., Ltd 2022], integrados a um *gateway* baseado no transceptor SX1302 [Semtech Corporation 2020], operando na

infraestrutura da plataforma TTN (The Things Network). A partir disso, o robô é capaz de publicar dados sensoriais relevantes, como a saída do LiDAR ou odometria, para servidores remotos, mesmo em ambientes isolados, onde a comunicação R2I é crítica para o sucesso da missão. A Figura 5.10 apresenta os rádios e gateway utilizados para testes com a rede LoRa (Módulo de Comunicação Long Range).

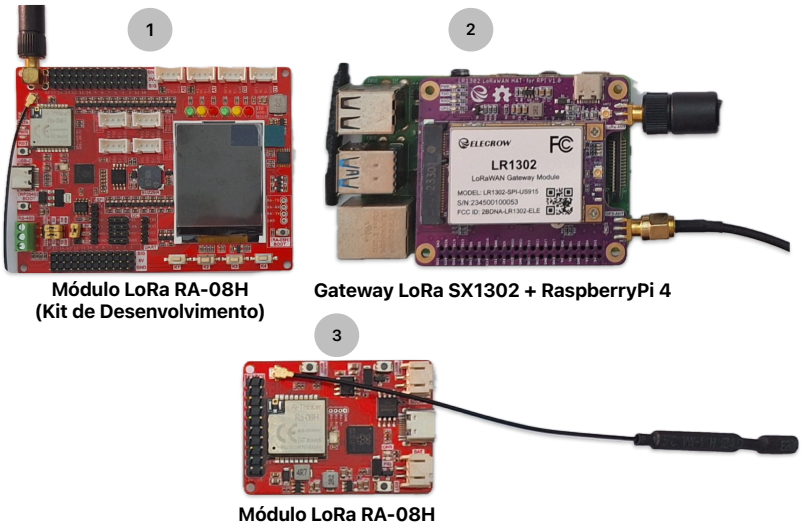


Figura 5.10: Módulos LoRa utilizados para testes em Long Range.

Figura 5.10: Arquitetura de comunicação LoRa adotada. (1) e (3) representam rádios LoRa RA-08H, integrados ao microcontrolador RP2040, responsáveis pela transmissão dos dados adquiridos pelos sensores embarcados. (2) refere-se ao gateway LoRa baseado no módulo SX1302, conectado a uma Raspberry Pi 4 rodando sistema operacional Linux. Esse conjunto é responsável pela recepção dos pacotes LoRa e pela implementação da camada de aplicação LoRaWAN, possibilitando o envio dos dados para servidores remotos por meio da Internet.

Além disso, foi desenvolvido um algoritmo para empacotamento e desempacotamento dos dados estruturados conforme o modelo de tópicos do ROS2, permitindo que, mesmo operando em uma rede LPWAN, o robô mantenha compatibilidade com ferramentas ROS, como visualização em tempo real e rastreamento de trajetória.

A metodologia proposta nesta tese envolve a compressão dos dados brutos do LiDAR por meio de um *autoencoder* treinado para gerar uma representação latente compacta e compatível com o limite de *payload* do protocolo LoRaWAN. Uma vez codificados, os dados são formatados em mensagens específicas que são enviadas, via comunicação serial, para um módulo RA-08H conectado ao transmissor. Este módulo, por sua vez, realiza a modulação LoRa e transmite a mensagem ao gateway SX1302.

Após a recepção dos dados pelo gateway, as mensagens são encaminhadas por meio da plataforma TTN para um servidor remoto, onde ocorre a descompactação dos

dados e a reconstituição da varredura original do LiDAR. Esse processo permite que os dados captados por robôs em campo possam ser analisados em tempo real por centros de controle ou aplicações de monitoramento remoto, possibilitando a implementação de estratégias como vigilância, mapeamento e apoio à navegação.

A Tabela 5.3 resume os principais parâmetros utilizados durante os experimentos, incluindo potência de transmissão, antenas, frequência e distância entre os dispositivos.

Tabela 5.3: Configurações Experimentais do Sistema LoRa

Parâmetro	**Valor**
Frequência de Transmissão	915 MHz sobre canal de 125 kHz
Spreading Factor (SF)	SF7 (maior capacidade de payload entre os SF disponíveis)
Potência de Transmissão	17 dBm (ajustada via comando AT, limite máximo do módulo: +22 dBm)
Distância Transmissor–Gateway	4 Testes: **2m (T0)**, **20m (T1)**, **50m (T2)**, **120m (T3)**
Antena do Transmissor	Antena omnidirecional articulada, 5 dBi, 915 MHz
Antena do Gateway	Antena omnidirecional, 12 dBi, 915 MHz, instalada a 6m do solo

Os testes práticos foram organizados em quatro fases (T0, T1, T2 e T3), com foco em analisar a estabilidade da transmissão, a perda de pacotes e a viabilidade da comunicação em diferentes cenários. A Figura 5.11 ilustra o arranjo experimental utilizado para os testes de campo.

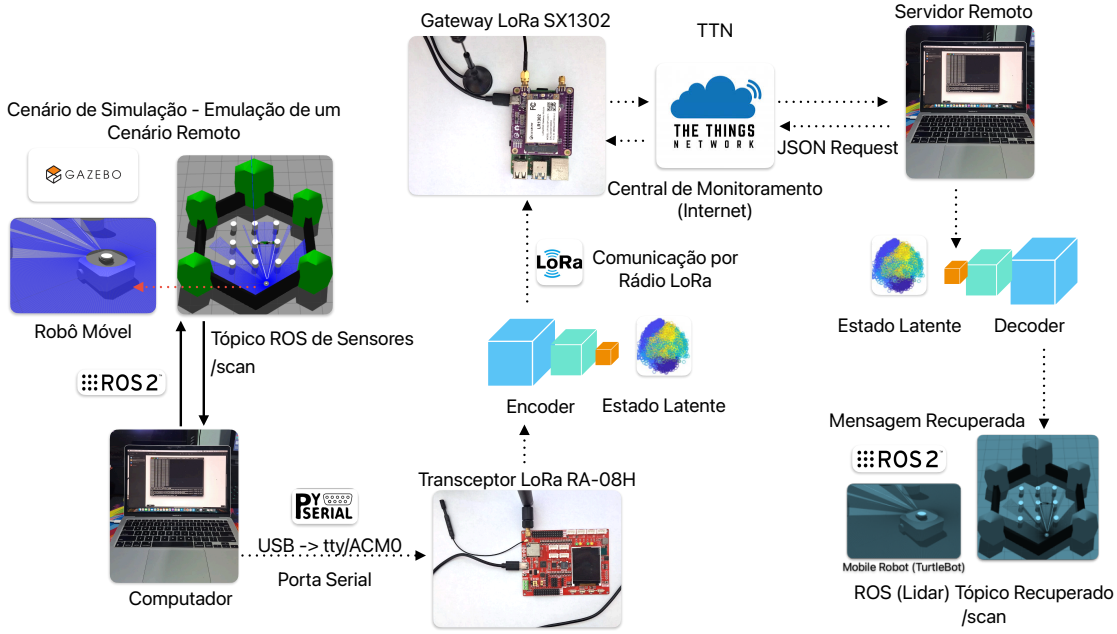


Figura 5.11: Proposta da Integração Long Range.

A seguir, detalha-se o Algoritmo *Long Range* com LoRaWAN, que estrutura o processo de transmissão e recepção de dados LiDAR via LoRaWAN, dividindo-o em três

etapas principais:

1. **Compressão dos Dados:** os dados do tópico `/scan` são extraídos e processados em tempo real por um *autoencoder*, resultando em uma representação latente de menor dimensão, adequada ao tamanho do *payload* da rede LoRaWAN.
2. **Formatação e Transmissão da Mensagem:** a saída latente do autoencoder é codificada em pacotes *float32*. Esses pacotes são enviados via UART para o módulo LoRa, que os transmite para o gateway SX1302.
3. **Recepção e Decodificação:** os dados recebidos pelo gateway são encaminhados a um servidor por meio da rede TTN. No destino, os pacotes são decodificados, descompactados pelo *decoder* do autoencoder, e reinseridos no ambiente ROS2 para posterior análise ou uso por outros nós da aplicação.

Esse processo permite validar a proposta de comunicação remota de dados sensoriais em sistemas embarcados e robóticos, mesmo sob restrições severas de largura de banda. A arquitetura proposta demonstra ser escalável, podendo ser adaptada para outros sensores ou tipos de dados, desde que respeitados os limites impostos pela rede LPWAN. A Figura 5.12 ilustra o cenário dos testes realizados com o Algoritmo *Long Range* que compõe o Módulo de Comunicação. Trata-se de um cenário em um parque municipal, composto por diversos obstáculos físicos, tais como edificações, árvores entre outros.

a) Testes de Long Range
Ambiente: Parque Municipal

b) Aparato de Testes: Computador para Emular o LIDAR e Rádio RA-08H

c) Posicionamento da Antena do Gateway: 6m de Altura

Figura 5.12: Proposta da Integração Long Range: Ambiente de Teste.

5.6.2 Compressão dos Dados

A primeira etapa do algoritmo proposto consiste na compressão dos dados brutos do sensor LiDAR utilizando uma rede neural autoencoder profunda. Considerando que um escaneamento típico do sensor LiDAR possui alta dimensionalidade, no caso deste trabalho, 360 valores de distância por varredura, não é viável transmitir essas informações diretamente pela rede LoRaWAN, devido às restrições rigorosas de carga útil (payload), que para SF7 e largura de banda de 125kHz, geralmente não ultrapassa 222 bytes.

Suponha que os dados de entrada do LiDAR contenham os 360 pontos por leitura, com cada ponto representado por um valor em precisão `float32`. Isso resulta em uma carga útil de dados inicial de:

$$360 \text{ pontos} \times 4 \text{ bytes por ponto} = 1440 \text{ bytes}$$

Ao utilizar o autoencoder, os dados são transformados em um espaço latente de menor dimensionalidade, por exemplo, reduzindo-os para 64 neurônios na camada latente. Mesmo utilizando `float32` na representação latente, a carga útil de dados é significativamente reduzida:

$$64 \text{ pontos} \times 4 \text{ bytes por ponto} = 256 \text{ bytes}$$

Essa redução de aproximadamente 1440 bytes para 256 bytes representa uma compressão substancial (82,22%), diminuindo o tamanho dos dados em quase seis vezes. No entanto, a redução excessiva do número de neurônios na camada latente pode comprometer a qualidade dos dados reconstruídos. No presente trabalho, utilizam-se 64 neurônios como um bom compromisso entre compressão e fidelidade dos dados recuperados.

Após a compressão, os 256 bytes de dados comprimidos são segmentados em pacotes menores, com aproximadamente 40 bytes cada, para transmissão via rede LoRaWAN. Essa segmentação é essencial para respeitar os limites de carga útil do protocolo, garantindo uma transmissão eficiente e livre de erros. Neste exemplo, os 256 bytes são divididos em 7 pacotes de aproximadamente 40 bytes cada. A compressão é realizada em tempo real, logo após a publicação do escaneamento LiDAR no tópico `/scan` dentro do ambiente ROS2. Uma vez comprimido, o vetor latente é serializado para a unidade de rádio via UART-módulo LoRa RA-08H. A interface com o módulo é realizada por meio da biblioteca `pyserial`, também em Python.

Cada mensagem codificada é estruturada com um cabeçalho contendo marcador de início e índice de sequência, permitindo a identificação e reconstrução dos dados no destino. Também são utilizados mecanismos de verificação, como CRC e temporizadores, para aumentar a robustez do sistema frente às interferências do meio físico.

O módulo LoRa, configurado com fator de espalhamento SF7 e potência de transmissão de 17 dBm (conforme apresentado na Tabela 5.3), realiza a modulação e transmissão dos dados através do canal de 915 MHz até o gateway. Este gateway (SX1302), conectado a um Raspberry Pi, recebe as transmissões e as encaminha para a rede LoRaWAN (The Things Network) via conexão IP segura.

5.6.3 Recebimento e Decodificação da Mensagem

Na extremidade receptora, os dados são recuperados da rede TTN utilizando integração via requisições JSON. Em seguida, esse vetor é inserido no decodificador (decoder) do modelo autoencoder previamente treinado, que reconstrói uma aproximação do escaneamento LiDAR original. Esse dado reconstruído pode ser visualizado, analisado ou reintegrado ao ambiente ROS2, possibilitando o monitoramento remoto do ambiente sensoriado mesmo em condições de conectividade limitada.

A acurácia do dado reconstruído é avaliada utilizando métricas como o Erro Quadrático Médio (MSE), assegurando que, mesmo após uma compressão agressiva, as informações recuperadas sejam suficientemente precisas para tarefas de navegação e mapeamento. Esse arcabouço completo de comunicação, da aquisição no ROS2 à transmissão LoRaWAN e à recuperação posterior, comprova a viabilidade de estender a percepção robótica a regiões remotas.

O treinamento de um autoencoder envolve o ajuste dos pesos da rede neural de forma a minimizar a diferença entre os dados de entrada e a saída reconstruída. Esse processo requer um conjunto de dados que represente adequadamente o tipo de informação que se deseja comprimir. Durante o treinamento, o codificador (encoder) aprende a transformar os dados de entrada em um espaço latente comprimido, enquanto o decodificador (decoder) aprende a reconstruir com precisão os dados originais a partir dessa representação compacta.

O Algoritmo 6 ilustra detalhadamente todas as etapas envolvidas nesse fluxo, desde a aquisição dos dados até sua reinterpretação no lado do receptor, demonstrando como a integração entre aprendizado de máquina e protocolos de comunicação de longo alcance pode superar limitações de banda e viabilizar aplicações robóticas em áreas remotas.

Algorithm 6 Algoritmo de Comunicação Long Range com Transmissão via LoRa e Compressão por Autoencoder - [Bezerra, Cardoso e Vieira 2025]

Entrada: Dados do LiDAR provenientes do tópico ROS `/scan` — 360 leituras `float32` (aproximadamente 1440 bytes)
Saída: Dados do LiDAR reconstruídos publicados no tópico ROS `/scan`
Inicialização:
1. Iniciar o nó ROS para leitura dos dados do LiDAR a partir do tópico `/scan`.
2. Estabelecer comunicação serial com o transceptor LoRa via USB.
3. Configurar o módulo LoRa com comandos AT (frequência: 915 MHz, SF: 7, potência: 17 dBm).

Etapa 1: Leitura e Pré-processamento dos Dados
4. Ler os dados do LiDAR a partir do tópico ROS (360 leituras, tipo `float32`).
5. Normalizar os dados para prepará-los para a compressão.

Etapa 2: Compressão com Autoencoder (MLP)
6. Inserir os dados de 360 dimensões no codificador (primeira parte do MLP).
7. Comprimir os dados para uma representação latente de 64 dimensões:
- Camada densa totalmente conectada com 360 entradas e 64 saídas.
- Função de ativação: ReLU.

Etapa 3: Preparação para Transmissão
8. Dividir a representação latente de 64 dimensões (256 bytes) em 7 quadros (frames) de até 40 bytes cada (o último pode ser menor).
9. Transmitir cada quadro via LoRa utilizando comandos AT.

Etapa 4: Recepção e Reconstrução no Gateway
10. Coletar todos os quadros no gateway.
11. Estruturar os quadros recebidos em um objeto JSON.
12. Publicar o objeto JSON na camada de aplicação do TTN (The Things Network).

Etapa 5: Recuperação dos Dados no Servidor Remoto
13. Realizar uma requisição HTTP para obter o objeto JSON do TTN.
14. Extrair a representação latente (64 dimensões) do JSON.

Etapa 6: Reconstrução com o Decodificador (MLP)
15. Inserir a representação latente no decodificador (segunda parte do MLP).
16. Reconstruir os dados originais do LiDAR (360 dimensões):
- Camada densa totalmente conectada com 64 entradas e 360 saídas.
- Função de ativação: Linear.

Etapa 7: Publicação dos Dados no ROS
17. Estruturar os dados reconstruídos em uma mensagem compatível com ROS.
18. Publicar os dados reconstruídos no tópico `/scan`.

Repetição:
19. Repetir o processo continuamente para novas leituras do LiDAR.

A Tabela 5.4 resume os hiperparâmetros utilizados durante o treinamento do autoencoder.

Tabela 5.4: Hiperparâmetros Utilizados no Treinamento do Autoencoder

Hiperparâmetro	**Valor**
Tamanho da Entrada	360
Dimensão do Espaço Latente	64
Número de Camadas (Codificador)	1 Camada Densa
Número de Camadas (Decodificador)	1 Camada Densa
Função de Ativação (Codificador)	ReLU
Função de Ativação (Decodificador)	Linear
Função de Perda	Erro Quadrático Médio (MSE)
Otimizador	Adam
Tamanho do Lote (Batch Size)	256
Número de Épocas	250
Taxa de Aprendizado (Otim. Adam)	Padrão (0,001)
Normalização dos Dados	Não Aplicada
Inicialização dos Pesos	Inicialização de Xavier
Embaralhamento Durante o Treinamento	Sim
Divisão para Validação e Teste	10% e 15%

5.6.4 Fator de Espalhamento Espectral em Cenário de Mobilidade

Além das abordagens de compressão e roteamento apresentadas no módulo *Long Range*, foi realizado um estudo prático sobre o comportamento do mecanismo nativo do LoRaWAN denominado *Adaptive Data Rate* (ADR) em cenários de mobilidade. Com o intuito de testar sua capacidade de manter o enlace de comunicação sob movimentação constante, foi proposto um experimento com sensores de baixa taxa de transmissão para validação da robustez do protocolo. É importante destacar que poucos trabalhos na literatura abordaram os limites e testes da rede LoRaWAN em cenários de mobilidade, e menos ainda exploraram implementações práticas reais. Trabalhos como o de [Benkahla et al. 2019] utilizaram apenas simulações computacionais com uma modelagem do meio físico simplificada.

O robô utilizado para testes, Robô JetBot, é guiado no mesmo cenário da seção de *Long Range* descrita anteriormente, um parque municipal com muitas árvores e obstáculos. Este robô movimenta com o auxílio de um controle remoto a uma velocidade constante de aproximadamente 2 km/h. Entre o robô e o *gateway*, são transmitidas informações de temperatura ambiente (°C) e umidade do ar (%). Apesar de serem informações sensoriais simples, a ideia principal é avaliar o algoritmo ADR em condições de movimentação e propor adequações que poderiam melhorar o processo de comunicação. O objetivo não é a complexidade dos dados, mas sim avaliar o desempenho do ADR em situações reais de variação de canal, obstrução e mudança de topologia.

A Figura 5.13 ilustra o cenário real proposto para o desenvolvimento de testes de comunicação robótica em movimentação para controle do espalhamento espectral.

Observa-se a presença de marcos que determinam o trajeto de movimentação do robô (pontos de marcação). Os pontos/marcos são numerados: P1, P2, P3 e P4. A Tabela

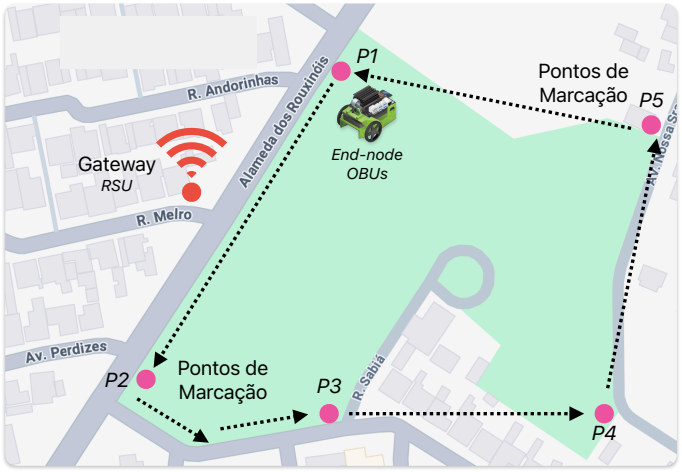


Figura 5.13: Cenário real para comunicação robótica em Long Range para avaliação do controle de espalhamento espectral - Autoria Própria.

5.5 abaixo dispõe a distância do referido ponto em relação ao *gateway* concentrador. A Figura 5.14 ilustra o robô Jetbot adaptado com o transceptor LoRa embarcado.

Tabela 5.5: Distâncias dos Pontos em Relação ao Gateway

Ponto	**Distância até o Gateway (m)**
P1	135
P2	130
P3	170
P4	300
P5	270

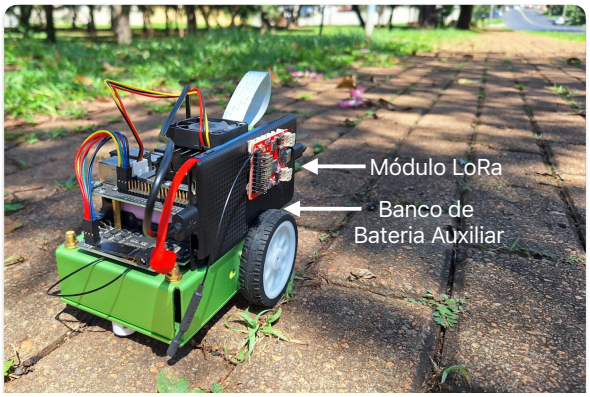


Figura 5.14: Adaptação do robô Jetbot com transceptor LoRa - Autoria Própria.

5.6.5 Controle do Espalhamento Espectral

Como alternativa ao ADR, foi proposto o algoritmo **Taxa Forçada de Dados**, que atua de forma mais agressiva na tentativa de manter a conectividade. Em vez de

aguardar a análise do SNR e ajustar a *Spreading Factor* (SF) com base em médias amostrais, o algoritmo forçado tenta a comunicação sequencialmente por diferentes SFs (SF7, SF8, SF10, SF12), mudando imediatamente para a próxima opção caso não receba o ACK do *gateway*. Essa abordagem visa mitigar a latência de resposta do ADR em ambientes dinâmicos, garantindo a entrega da mensagem mesmo com consumo energético potencialmente maior. O Taxa de Dados Forçada é disposto através do Algoritmo 7, conforme segue:

Algorithm 7 Algoritmo de Taxa Forçada de Dados

```
Entrada: Lista de SFs [SF7, SF8, SF10, SF12], número máximo de tentativas N
Saída: SF que garante a comunicação
Inicializar SF atual para SF7
Inicializar contador de tentativas i ← 0
while i < N do
    Enviar mensagem com SF atual
    if Receber ACK then
        Retornar SF atual
    else
        Incrementar contador de tentativas i ← i + 1
        Atualizar SF atual para próxima SF na lista
Retornar falha de comunicação
```

O algoritmo ADR, por outro lado, ajusta a SF com base na análise contínua do SNR (*Signal-to-Noise Ratio*) e da potência do sinal, tentando otimizar a taxa de dados e a eficiência energética. No entanto, isso pode não ser eficiente em cenários de mobilidade, onde o SNR pode variar rapidamente devido ao movimento do robô e a presença de obstáculos.
O algoritmo nativo **ADR** possui as seguintes características:

- Análise do SNR: Ajusta a SF com base no SNR e potência do sinal.
- Otimização da Eficiência : Tenta otimizar a taxa de dados e a eficiência energética.
- Latência na Resposta: Pode não responder rapidamente a mudanças rápidas no ambiente devido ao tempo necessário para analisar os dados de SNR.

A proposta de **Taxa Forçada de Dados** apresenta as seguintes características:

- Ajuste Baseado em ACK: Ajusta a SF apenas com base na recepção ou não do ACK.
- Resposta Rápida: Troca rapidamente a SF para garantir a comunicação, sem esperar por uma análise detalhada do SNR.
- Simplicidade: Mais simples de implementar, pois não requer análise contínua do SNR.

A execução destes testes ampliou a compreensão da aplicabilidade do LoRa em cenários de mobilidade, fornecendo insumos experimentais para ajustes nos parâmetros de controle de espalhamento espectral e estabelecendo fundamentos para arquiteturas de comunicação robustas em ambientes industriais dinâmicos.

Com isso, encerra-se a descrição dos módulos de comunicação de longo alcance (*Long Range*). A seguir, será apresentado o módulo de comunicação de curto alcance (*Short Range*), que utiliza o protocolo DDS para compartilhamento de ações e estados entre múltiplos robôs em uma arquitetura colaborativa de aprendizado por reforço.

5.7 Comunicação em Curto Alcance (Short Range)

Para viabilizar a cooperação entre múltiplos robôs em ambientes com conectividade local, foi desenvolvido um algoritmo de comunicação de curto alcance baseado em DDS (*Data Distribution Service*), denominado $MARL_{Q_{DDS}}$ — *Multiagent Reinforcement Learning with Double Deep Q-Network and DDS*. Esse algoritmo é responsável por orquestrar o compartilhamento eficiente de informações entre agentes, garantindo que apenas as experiências mais relevantes sejam transmitidas, a fim de otimizar o aprendizado multi-robô.

A comunicação entre os robôs é estruturada em mensagens padronizadas, categorizadas em três tipos principais: imagens RGB-D, odometria e experiências (transições de estados). A Tabela 5.6 resume os campos principais dessas mensagens e os respectivos tópicos DDS utilizados para sua publicação. Todas as mensagens incluem um cabeçalho com identificador e *timestamp*, um *payload* com os dados sensoriais ou experiências, e um código de verificação de erros (CRC), como ilustrado na Figura 5.15.

O processo de transmissão de estados ocorre em ciclos de aprendizado. A cada cinco episódios, a partir do episódio 30, cada robô avalia localmente suas experiências armazenadas, por meio da rede neural $Q_i(s, a)$, responsável por estimar a utilidade de cada transição (s, a, r, s'). As k transições com maior valor Q são selecionadas e encapsuladas em mensagens do tipo "Experiência" e publicadas no tópico ROS2 `/robot/experience`.

Tabela 5.6: Formato das Mensagens DDS

Tipo	Campos Principais	Tópico DDS
Imagem RGB-D	ID do robô, timestamp, imagem RGB e imagem de profundidade (Base64)	/robot/sensor/rgbd
Odometria	ID do robô, timestamp, posição (x, y, θ), velocidade (linear e angular)	/robot/sensor/odometry
Experiências	$(s, a, r, s', done)$	/robot/experience

Para gerenciar o intercâmbio de experiências entre agentes, é utilizado o **Serviço de Reprodução de Experiências** (*Memory Replay Service*), implementado como um nó

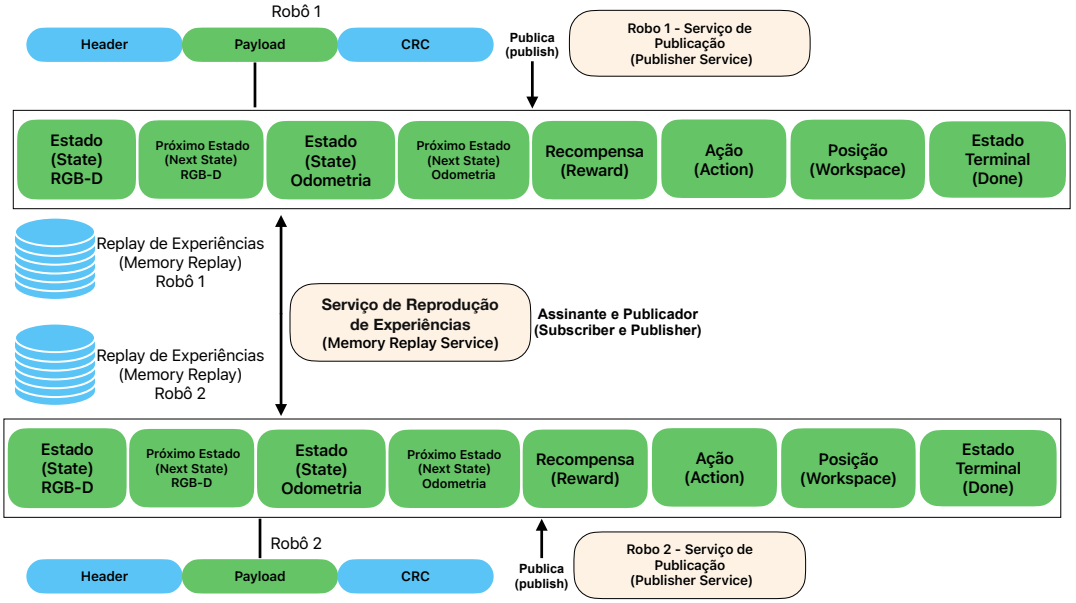


Figura 5.15: Estrutura de Distribuição de Estados Proposto pelo Algoritmo Short Range $MARL_{Q_{DDS}}$.

DDS que atua tanto como assinante quanto como publicador. Esse serviço ouve o tópico `/robot/experience`, coleta as transições publicadas por todos os robôs e redistribui essas experiências para as memórias locais dos demais agentes. Esse fluxo garante que apenas transições de alta relevância de acordo com a política de aprendizado *Q-Learning* sejam compartilhadas na rede, promovendo uma troca seletiva e eficiente.

O funcionamento completo do sistema de comunicação *short range* é descrito no Algoritmo 8, que mostra o ciclo de execução dos agentes, a publicação e recepção de experiências via DDS, e a atualização da rede neural Q por meio do método *Double Deep Q-Network* (DDQN). Esse protocolo permite não apenas uma comunicação robusta, mas também uma estratégia de aprendizado distribuído.

5.8 Critérios de Avaliação dos Módulos e Apresentação dos Resultados

Para uma análise objetiva do sistema de navegação proposto, cada módulo foi avaliado com base em métricas específicas, de acordo com seus objetivos e funcionalidades. A seguir, detalham-se os critérios adotados para análise dos resultados de cada componente.

- **Módulo de Controle e Fusão Sensorial**: este módulo é avaliado inicialmente, visto que as simulações do Módulo de Aceleração de Aprendizagem integram um

Algorithm 8 Algoritmo de Comunicação Short Range: Algoritmo $MARL_{Q_{DDS}}$ Multi-Robô com Compartilhamento da Rede Q via DDS

Inicializar cada robô i com rede Q_i e buffer de experiências B_i Definir a política π_i baseada em Q_i e os parâmetros de exploração Definir os tópicos DDS para troca de estados e melhores ações
for $e = 0$ *até* E **do**
  for *cada robô* i **do**
    Observar estado atual s_i e selecionar ação $a_i \sim \pi_i(s_i)$
    Executar a_i, observar novo estado s_i' e recompensa r_i
    Armazenar transição (s_i, a_i, r_i, s_i') em B_i
    if $e > 30$ *e* $e \,\%\, 5 == 0$ **then**
      Selecionar as k melhores ações locais com maior valor $Q_i(s, a)$
      Publicar essas ações e seus valores estimados no tópico `/robot/experience`
      Receber melhores ações estimadas de outros robôs via DDS
      Atualizar o buffer B_i com as ações recebidas do *memory replay service*
  for *cada robô* i **do**
    Amostrar um minibatch (s, a, r, s') de B_i
    Calcular $y = r + \gamma \cdot Q_{\text{target}}(s', \arg\max_{a'} Q(s', a'))$
    Minimizar o erro: $\mathcal{L} = (y - Q(s, a))^2$
    Atualizar os parâmetros da rede Q_i

mecanismo de fusão sensorial para processar conjuntamente dados visuais e não visuais. Dessa forma, é necessário, primeiramente, identificar qual técnica de fusão oferece os melhores resultados isoladamente. Para isso, diferentes estratégias de fusão de sensores são analisadas quanto ao impacto no desempenho do robô durante a navegação. A principal métrica utilizada é a Recompensa Média (RA) ao longo do processo de treinamento.

- **Módulo de Aceleração de Aprendizagem**: após a definição da estratégia de fusão, avalia-se a capacidade do algoritmo proposto (EKF-DQN) em acelerar o processo de aprendizado em comparação com abordagens consagradas da literatura, como o DQN e o D3QN. As principais métricas utilizadas são a Recompensa Média (RA) e a Taxa de Sucesso (*Success Rate* - SR), obtidas durante o treinamento dos agentes.
- **Módulo de Segurança**: avalia-se a capacidade do sistema em realizar frenagens seguras e efetivas em ambientes dinâmicos. O foco está no comportamento do robô diante de situações de risco, analisando sua resposta a obstáculos móveis em tempo real.
- **Módulo de Comunicação**: este módulo é subdividido em duas frentes:
  - *Comunicação de Longo Alcance (LoRaWAN)*: é avaliada a viabilidade da transmissão de sensores com grande volume de dados, como o LiDAR, por meio da compressão via autoencoder e transmissão do espaço latente. As

métricas utilizadas são: tempo de envio, taxa de transmissão (bits/s), taxa de entrega de pacotes (PDR), latência e vazão total da rede.

– *Comunicação de Curto Alcance*: avalia-se o desempenho do sistema multiagente em funcionamento completo, com a integração dos módulos de controle, fusão e aceleração em um ambiente de agentes. O algoritmo $MARL_{Q_{DDS}}$ é comparado com abordagens isoladas de aprendizado por reforço, destacando-se sua capacidade de melhorar o desempenho por meio da comunicação entre agentes.

Essa organização permite que cada módulo seja analisado individualmente quanto ao seu desempenho e à sua contribuição para a navegação autônoma robusta e eficiente. Por fim, são apresentados os resultados obtidos com o sistema completo, no qual os módulos de aceleração, fusão sensorial, controle e comunicação operam de maneira integrada. Essa etapa final visa demonstrar a viabilidade e a eficiência da arquitetura proposta em um cenário mais próximo do uso real.

5.9 Resumo das Arquiteturas de Redes Neurais

Esta seção apresenta um resumo das redes neurais utilizadas nos módulos desenvolvidos no sistema proposto. São descritas suas camadas principais, funções de ativação, função de perda e o objetivo de cada arquitetura.

5.9.1 Fusão Sensorial - DDQN com CNN e Sensores Não Visuais

Tabela 5.7: Resumo da arquitetura DDQN com fusão sensorial (RGB-D + sensores não-visuais + EKF).

Item	**Descrição**
Tipo de Rede	Deep Q-Network (DDQN) com fusão sensorial (*Late Fusion*) e aceleração com EKF
Propósito	Navegação autônoma com tomada de decisão baseada em múltiplas entradas sensoriais
Fluxo Arquitetural	**RGB-D**: CNN(Conv2D $\rightarrow$ ReLU $\rightarrow$ MaxPool) $\times$ 2 $\rightarrow$ Flatten(512) **Sensores**: FC1(32) $\rightarrow$ ReLU $\rightarrow$ FC2(32) $\rightarrow$ ReLU **Fusão**: Concatenação(512 + 32 = 544) **Decisão**: FC(256) $\rightarrow$ ReLU $\rightarrow$ FC(3) saída
Estimativa de Estado com EKF	Aplicação do EKF aos dados de odometria antes da entrada no buffer de memória *Replay* Estimativa de s_{k+1}, s_{k+2} com base nos sensores (IMU + Encoders)
Função de Ativação	ReLU nas camadas ocultas
Função de Custo	Erro Quadrático Médio (MSE)
Função de Valor-Alvo	$Q_{\text{target}} = R + \gamma \cdot Q(s', \arg\max_{a'} \hat{Q}(s', a', \theta), \theta^{-})$
Tipo de Aprendizado	Aprendizado por reforço (off-policy) com armazenamento de transições enriquecidas no Deque
Aceleração de Aprendizado	EKF antecipa e suaviza estados futuros antes do armazenamento, reduzindo ruído e inconsistências

5.9.2 Autoencoder - Compressão de Dados LiDAR (Módulo Long Range)

Tabela 5.8: Resumo da arquitetura do autoencoder para compressão de dados LiDAR.

Item	**Descrição**
Tipo de Rede	Autoencoder totalmente denso (MLP)
Propósito	Redução de dimensionalidade para transmissão via LoRa (compressão)
Fluxo Arquitetural	**Codificador**: Entrada(360) → Dense(128) → ReLU → Dense(64) → Latente(64) **Decodificador**: Latente(64) → Dense(128) → ReLU → Dense(360) → Saída
Função de Ativação	ReLU nas camadas ocultas, Linear na camada de saída
Função de Custo	Erro Quadrático Médio (MSE)
Tipo de Aprendizado	Não supervisionado (auto-supervisado com entrada = saída)
Dados	515 Leituras de LIDAR. 75% Treino, 10% Validação, 15% teste

5.9.3 Módulo de Segurança - ResNet50 Binária

Tabela 5.9: Resumo da arquitetura do Módulo de Segurança com ResNet50.

Item	**Descrição**
Tipo de Rede	CNN (ResNet50) com fine-tuning para classificação binária
Propósito	Identificação de risco iminente de colisão com parada preventiva do robô
Fluxo Arquitetural	ResNet50 pré-treinada (camadas convolucionais congeladas) + GlobalAveragePooling → FC(1) → Sigmoid
Função de Ativação	ReLU nas camadas internas, Sigmoid na saída
Função de Custo	Binary Cross-Entropy (BCE)
Tipo de Aprendizado	Supervisionado (classificação binária)
Dados	341 Imagens. 75% Treino, 10% Validação, 15% teste

As tabelas acima sintetizam as principais arquiteturas de redes neurais aplicadas no sistema proposto, evidenciando suas diferenças estruturais e objetivos específicos em cada módulo.

5.10 Conclusões

Este capítulo apresentou, em detalhes, a metodologia utilizada para o desenvolvimento dos algoritmos e módulos que compõem a arquitetura proposta de navegação robótica autônoma. A estrutura foi dividida em módulos funcionais que atuam de forma integrada para garantir a eficiência, segurança e conectividade do sistema em ambientes desafiadores.

Inicialmente, foi descrito o **Módulo de Controle de Navegação e Fusão**, responsável pela estimativa do estado do robô a partir de múltiplas fontes de dados. Foram propostos e analisados dois métodos distintos de fusão, permitindo avaliar qual abordagem apresenta melhor desempenho em termos de robustez e precisão de localização.

Na sequência, apresentou-se o **Módulo de Aceleração** de aprendizado por reforço, onde foi integrado o Filtro de Kalman Estendido (EKF) com redes DQN, formando o algoritmo EKF-DQN. Essa integração visa otimizar a aprendizagem do agente por meio da previsão de estados, reduzindo a quantidade de interações necessárias com o ambiente e, consequentemente, acelerando a convergência do treinamento.

Foi descrito também o **Módulo de Segurança**, concebido como uma camada de detecção e prevenção de colisões, fundamental para a operação segura do robô em ambientes com obstáculos e incertezas. A atuação desse módulo ocorre paralelamente à política aprendida, podendo sobrescrever ações potencialmente perigosas.

Em seguida, abordou-se o **Módulo de Comunicação**, dividido em duas categorias: *Long Range* e *Short Range*. No módulo de longo alcance (*Long Range*), utilizando o protocolo LoRa, foram exploradas duas abordagens complementares: uma voltada à camada física, com a análise do desempenho do mecanismo ADR (*Adaptive Data Rate*) em ambientes com movimentação; e outra voltada à camada de aplicação, onde foi desenvolvido um protocolo de comunicação para transmissão de dados do ROS via LoRa, possibilitando a interconexão de nós ROS em cenários de conectividade expandida (WAN). Além disso, foi proposto o algoritmo de comunicação *Long Range* com compressão via autoencoder, demonstrando viabilidade para a transmissão de dados sensoriais, como LiDAR, em redes de baixa largura de banda.

Por fim, no contexto de comunicação de curto alcance (Short Range), foi apresentado o protocolo $\text{MARL}_{\text{Q}_{\text{DDS}}}$, que permite o compartilhamento distribuído de experiências entre robôs utilizando DDS (Data Distribution Service). Esse mecanismo possibilita a criação de um sistema multiagente mais eficiente, capaz de aprender de forma colaborativa por meio da troca de estados e melhores ações estimadas entre os agentes.

Cada seção metodológica descreveu de forma clara os procedimentos, parâmetros e decisões adotadas durante a implementação dos módulos, permitindo reprodutibilidade e compreensão aprofundada das soluções propostas.

No próximo capítulo, serão apresentados os resultados obtidos a partir da execução e avaliação prática de cada módulo e algoritmo discutido nesta seção. Esses resultados têm como objetivo validar a efetividade das abordagens desenvolvidas, demonstrando suas contribuições para o avanço da navegação autônoma inteligente, segura e conectada.

CAPÍTULO 6

Resultados Experimentais

6.1 Resultados

Neste capítulo, são apresentados os resultados experimentais obtidos a partir da avaliação dos módulos que compõem o sistema de navegação proposto. Os testes foram realizados individualmente para cada componente, a saber: Módulo de Aceleração de Aprendizagem, Módulo de Controle e Fusão Sensorial, Módulo de Segurança e Módulo de Comunicação. Conforme discutido no Capítulo 5, cada módulo foi desenvolvido com o objetivo de otimizar aspectos específicos da navegação autônoma.

6.2 Módulo de Controle e Fusão de Sensores

Nesta seção, são apresentados os resultados referentes à implementação das estratégias de fusão sensorial em conjunto com algoritmos de aprendizado por reforço. O objetivo é analisar o impacto da combinação de múltiplos sensores na qualidade da navegação autônoma.

Os resultados foram obtidos por meio de simulações realizadas no ambiente descrito na Figura 5.2, apresentada no Capítulo 5. Inicialmente, são mostrados os resultados da execução de missões robóticas utilizando apenas um sensor (sem fusão). Em seguida, são apresentados os resultados obtidos com a aplicação das técnicas de fusão sensorial descritas na metodologia, permitindo a comparação entre os diferentes cenários de percepção e tomada de decisão.

6.2.1 Avaliação Sem Fusão de Sensores

Como etapa preliminar, foi realizada uma avaliação do sistema utilizando apenas um sensor perceptivo, especificamente, a câmera RGB-D, sem aplicação de técnicas de fusão sensorial. Essa análise teve como objetivo servir de linha de base para comparações futuras.

Além disso, buscou-se justificar o uso do algoritmo DDQN no módulo de fusão. Para isso, comparou-se inicialmente o desempenho do algoritmo DQN, uma abordagem mais elementar de aprendizado por reforço. Ambos os algoritmos foram avaliados sob as mesmas condições experimentais, considerando um único sensor de entrada.

A Figura 6.1 apresenta o resultado da execução do DQN, representado por um gráfico de Recompensa Média obtida em função dos Episódios de Treinamento. Este gráfico permite observar a estabilidade do aprendizado e o desempenho do agente ao longo do tempo.

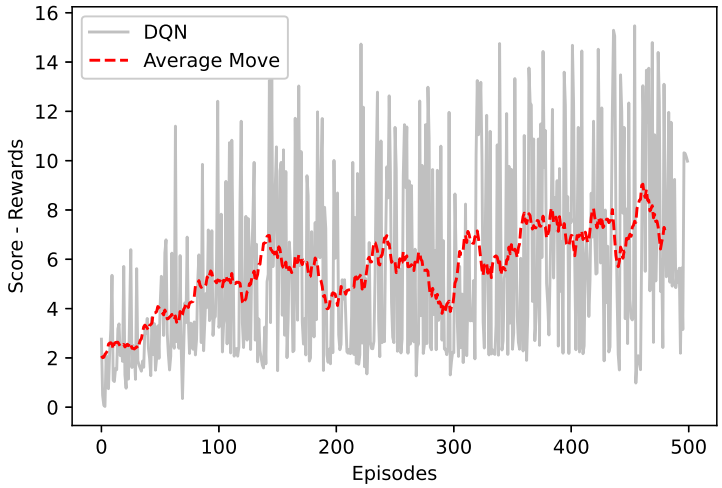


Figura 6.1: Curva de Aprendizado com Recompensas Obtidas (Rewards) da Rede DQN.

É possível observar que a taxa de crescimento da média móvel é relativamente baixa, alcançando uma média máxima de ≈ 8. Isso indica que com o método DQN e uso exclusivo do sensor visual RGB-D o AMR aprende poucas ações eficientes referentes a missão proposta. Além disso é possível observar alta volitividade nas recompensas brutas obtidas por episódio (linha cinza do gráfico), um indicativo que a estabilidade do método é fraca.

A Figura 6.2 apresenta o resultado de implementação do DDQN por meio do gráfico de Episódios de Treinamento $\times$ Recompensas (*Episodes* $\times$ *Rewards*).

É possível observar uma maior tendência de maximização das recompensas obtidas. O método também apresentou maior estabilidade e alcançou a média máxima de ≈ 10. A diferença entre o custo computacional do DQN e DDQN é praticamente imperceptível. Estes foram indicativos para seleção do DDQN no trabalho em questão.

Vale ressaltar que apesar da constatação de melhorias referentes ao método DDQN, o uso exclusivo do sensor RGB-D não foi suficiente para a conclusão da missão proposta. A percepção pela câmera RGB-D resulta em estados suficientes para navegar sem colisão, isto devido ao nível de profundidade da cena fornecida pelo sensor, entretanto

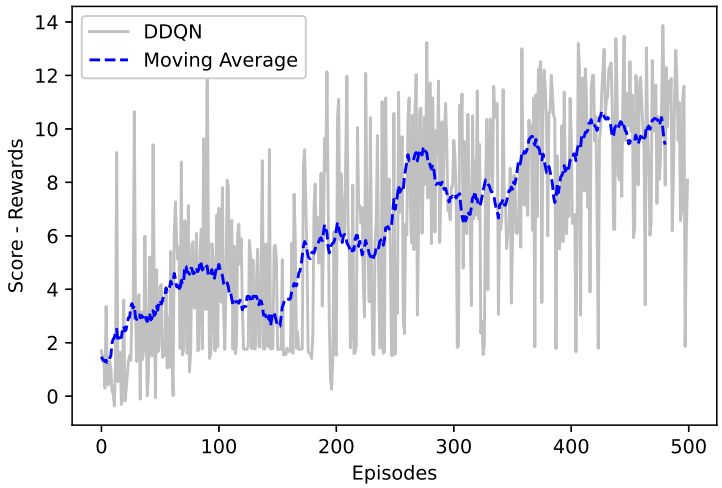


Figura 6.2: Curva de Aprendizado com Recompensas Obtidas (Rewards) pela Rede DDQN.

não é suficiente para representar precisamente os pontos necessários para passar por cada estação de trabalho com exatidão.

6.2.2 Aplicação dos Métodos de Fusão de Sensores

Após verificar que o método DDQN utilizando apenas a câmera RGB-D apresenta limitações, foram aplicadas as propostas de fusão de sensores visuais e não visuais. Os métodos implementados foram o DDQN com *Late Fusion* e o DDQN com *Iterative Fusion*. A Figura 6.3 Episódios $\times$ Recompensas (*Episodes* $\times$ *Rewards*) apresenta a média móvel das recompensas obtidas ao longo dos episódios para cada abordagem, incluindo o DDQN simples (sem fusão). Observa-se, visualmente, uma superioridade do método DDQN com *Late Fusion* em relação às demais abordagens.

Entretanto, além da análise gráfica, é essencial avaliar métricas quantitativas que indicam a qualidade da navegação. A Taxa de Sucesso (*Success Rate* - SR) representa a proporção de episódios em que o robô móvel autônomo (AMR) completou a missão com sucesso, isto é, visitou corretamente as quatro estações de trabalho. Essa métrica é calculada pela razão entre o número de episódios bem-sucedidos e o total de episódios. Durante o processo de aprendizagem por reforço, especialmente nas fases iniciais de exploração (*exploration*), é comum observar taxas de sucesso reduzidas. A melhoria dessa taxa ocorre progressivamente na fase de explotação (*exploitation*), à medida que o agente aprende políticas mais eficazes.

A Tabela 6.1 apresenta um resumo das métricas de desempenho para os algoritmos com e sem fusão de sensores.

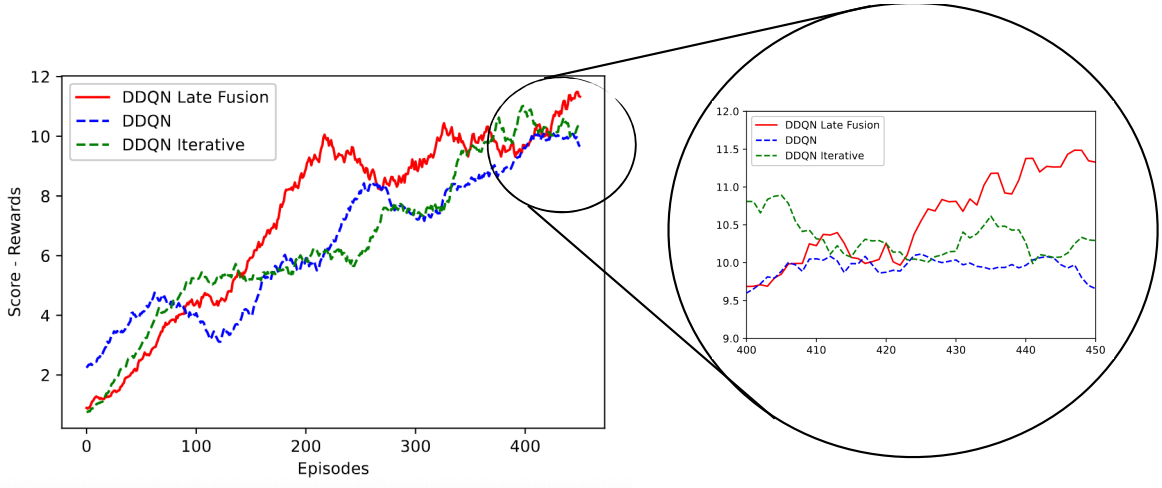


Figura 6.3: Curva de Aprendizado com Recompensas Obtidas (Rewards): Comparação entre os métodos de fusão e o DDQN sem fusão.

Tabela 6.1: Métricas de desempenho entre métodos de fusão durante o treinamento.

Proposta	**Taxa de Sucesso (SR %)**	**Recompensa Média (RA)**
DDQN *Late Fusion*	28,2	7,38
DDQN *Iterative Fusion*	11,6	6,43
DDQN (sem fusão)	1,4	5,55

Observa-se, na Tabela 6.1, que o algoritmo DDQN sem fusão apresenta uma Taxa de Sucesso significativamente inferior quando comparado aos métodos com fusão de sensores. No entanto, ao analisar a Recompensa Média (RA), embora também haja uma diferença entre os métodos, ela não é tão expressiva quanto na SR, indicando que a fusão sensorial contribui especialmente para a execução correta das missões completas, mais do que para pequenas melhorias graduais nas recompensas obtidas.

É importante destacar que os resultados apresentados na Tabela 6.1 foram obtidos **durante a fase de treinamento dos agentes**. Nessa etapa, o comportamento do robô é fortemente influenciado pelo processo de exploração (*exploration*), no qual as ações são escolhidas de forma parcialmente aleatória para promover o aprendizado do ambiente. Como consequência, a Taxa de Sucesso tende a ser relativamente baixa, mesmo quando o algoritmo apresenta tendência de melhoria. À medida que o treinamento avança e a política do agente converge (fase de *exploitation*), é esperado um aumento gradual dessa métrica.

Além disso, ao comparar os métodos avaliados, observa-se a superioridade do algoritmo DDQN-*Late Fusion*, que apresentou os melhores resultados tanto em termos de recompensa média quanto em taxa de sucesso. Essa evidência reforça a eficácia da fusão tardia de dados visuais e não visuais na melhoria da tomada de decisão do agente. Dessa forma, optou-se pela adoção do DDQN-*Late Fusion* como técnica de fusão sensorial

padrão a ser integrada ao sistema completo de navegação autônoma apresentado neste trabalho.

6.3 Módulo de Aceleração de Aprendizagem

Para avaliar o desempenho dos algoritmos DQN e EKF-DQN no simulador, as seguintes métricas são consideradas: i) Média de recompensas obtidas e ii) Taxa de sucesso, que representa o número de conclusões da missão proposta em um conjunto de episódios. Através dessas métricas, é possível avaliar se o EKF-DQN acelerou ou não o método DQN, e concluir a hipótese levantada na introdução deste artigo sobre a melhoria de desempenho. Na seção 6.1, essas métricas são apresentadas detalhadamente por meio de tabelas. Há duas maneiras de apresentar essas métricas: i) Agente na fase de treinamento e ii) Implantação do Agente Treinado. Vale destacar que, no treinamento de RL, a Taxa de sucesso não chega a 100% e é relativamente baixa, pois o início do aprendizado é composto por fases de exploração (*exploration*), onde as ações são aleatórias. É comum observar a convergência do algoritmo durante o treinamento com uma diminuição na taxa de exploração (ϵ), fase chamada de exploração. Uma taxa de sucesso mais alta pode ser observada na Implantação do Agente, onde o robô já possui seu sistema de controle treinado.

As configurações de hiperparâmetros utilizadas nas redes neurais dos algoritmos DQN e EKF-DQN foram mantidas idênticas, de forma a garantir uma comparação justa entre os métodos. Tanto a rede convolucional (CNN) responsável pela extração de características visuais, quanto as camadas totalmente conectadas (FC1 e FC2) que processam os dados não-visuais e realizam a etapa de decisão, foram parametrizadas conforme apresentado nas Tabelas 5.1 e 5.2. Essas arquiteturas compõem o módulo de controle e fusão sensorial dos respectivos algoritmos e foram adotadas com os mesmos valores de tamanho de camada, função de ativação e configuração da camada de saída.

A Tabela 6.2 abaixo mostra os principais parâmetros mencionados da rede neural totalmente conectada, tais como: i) taxa de aprendizado (α), ii) taxa de exploração (ϵ) e iii) tamanho do lote.

Tabela 6.2: Hiperparâmetros da Rede Neural Artificial.

Taxa de Aprendizado (α)	**Tamanho do Lote (*batch size*)**	**Memória de Replay**	**Taxa de Exploração (ϵ)**
1×10^{-3}	32	6×10^{3}	0,995

Na Figura 6.6, os gráficos das médias móveis do treinamento de aprendizado por reforço são destacados, tanto para os algoritmos DQN quanto para o EKF-DQN (*Score* × Episódios). Ambos os algoritmos convergem para a mesma recompensa média (aproximadamente 25), porém observa-se que na região próxima ao episódio 100 é

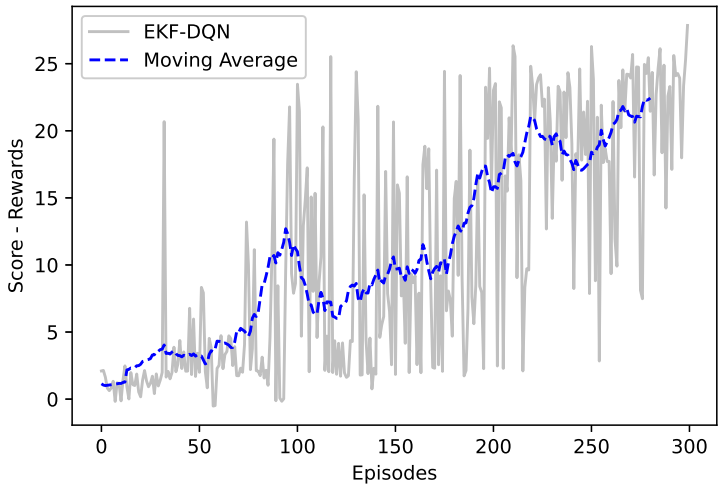


Figura 6.4: Curva de Aprendizado com Recompensas Obtidas (Rewards): Algoritmo EKF-DQN.

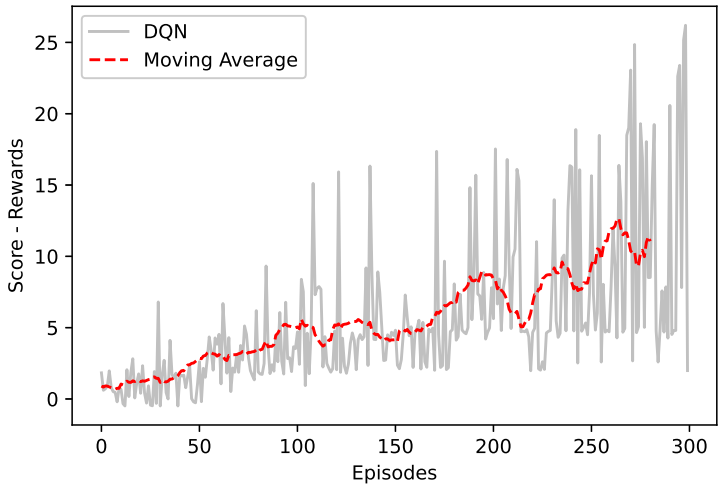


Figura 6.5: Curva de Aprendizado com Recompensas Obtidas (Rewards): Algoritmo DQN.

possível verificar o efeito da aceleração, onde a curva de aprendizado azul (EKF-DQN) apresentou a maior recompensa média nesta fase de aprendizado. Nessa região, a média móvel de recompensas obtidas é de aproximadamente 5 para o DQN (curva laranja), enquanto para o EKF-DQN a média é de aproximadamente 8. Observa-se que o DQN puro precisou de mais 130 episódios para atingir o nível de recompensa já obtido pelo EKF, pois apenas no episódio 200 o DQN puro pode alcançar uma recompensa média de aproximadamente 8. Isso ressalta o processo de aceleração de aprendizado do EKF.

Na mesma Figura 6.6, comparamos a eficácia da nossa proposta com outros algoritmos na literatura, como o Dueling Double DQN (D3QN). O método Advantage

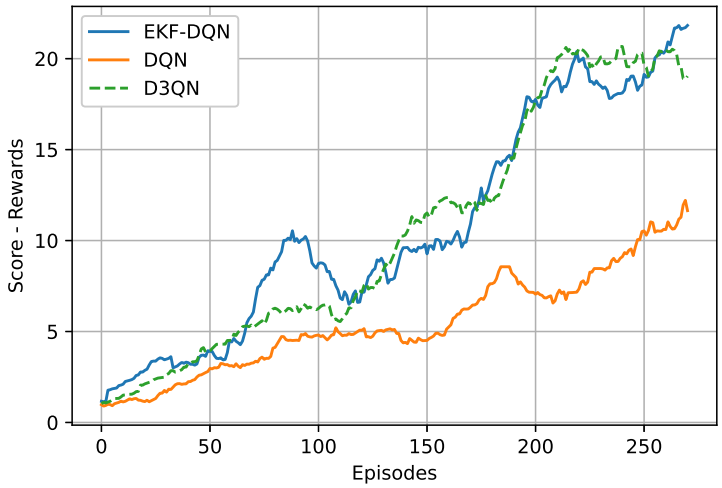


Figura 6.6: Curva de Aprendizado com Recompensas Obtidas (Rewards): Comparação do EKF-DQN com outros algoritmos de Aprendizado por Reforço.

Actor-Critic (A2C) também foi testado, sendo um método baseado em política (*on policy*), e, portanto, apresenta outra metodologia de aprendizado diferente da família de algoritmos *off policy*. Em nossa pesquisa sobre questões de navegação, especificamente nosso problema e modelagem, observamos que métodos baseados em política não se mostraram tão eficazes quanto o esperado, razão pela qual nem mesmo foram exibidos no gráfico. Uma razão plausível para essa ineficácia pode ser a própria modelagem intrínseca do problema, que talvez tenha características ou nuances que não se alinhem diretamente com abordagens baseadas em política. Além disso, problemas complexos de navegação podem exigir soluções personalizadas ou otimizadas especificamente para o cenário em questão. Isso sugere que mais investigações e possíveis adaptações dos métodos baseados em política podem ser necessárias para alcançar melhorias significativas no desempenho.

O D3QN, representado pela curva verde na Figura 6.6, apresentou resultados mais competitivos em comparação ao EKF-DQN. O D3QN possui uma arquitetura de rede ligeiramente mais complexa devido à camada de separação entre valores e benefícios. Isso pode aumentar o custo computacional durante o treinamento e a inferência em comparação com um DQN padrão. Integrar o Filtro de Kalman ao DQN oferece uma alternativa promissora ao D3QN em termos de desempenho, conforme observado em nossos experimentos. No entanto, um dos maiores benefícios de nossa proposta reside em sua eficiência computacional. Enquanto o D3QN requer uma arquitetura de rede neural modificada para diferenciar entre valor e vantagem, nossa proposta mantém a arquitetura DQN intacta, complementando-a apenas com o Filtro de Kalman.

6.3.1 Métricas de Desempenho

A Tabela 6.3 apresenta um resumo do desempenho entre os algoritmos durante a etapa de treinamento (DQN e o proposto EKF-DQN) com base nas seguintes métricas: taxa de sucesso e recompensa média.

Tabela 6.3: Resultados dos Testes de Simulação - Estado de Treinamento. Abreviações: RA = Recompensa Média; SR = Taxa de Sucesso (Success Rate).

Teste	**DQN**		**EKF-DQN**		**D3QN**	
	RA	SR	RA	SR	RA	SR
1º	10,94	15,00%	8,74	17,00%	10,34	15,33%
2º	6,04	5,33%	8,75	8,66%	8,83	17,00%
3º	5,53	2,00%	10,81	24,00%	8,74	15,33%
Média RA	**7,50**	–	**9,43**	–	**9,30**	–
Média SR	–	**7,44%**	–	**16,55%**	–	**15,88%**

A Tabela 6.4 mostra os resultados da simulação em um ambiente estático (sem movimentação de pessoas) após o treinamento do agente. Esses testes refletem um tipo de implantação do agente, onde é possível medir seu desempenho após o treinamento. Na fase de teste, os valores de peso do EKF-DQN são fixos e usados para controlar o robô móvel. Nota-se um aumento na taxa de sucesso (aproximando-se de 93,33%), assim como na recompensa média (23,56) em comparação com a fase de treinamento.

Tabela 6.4: Resultados da Implantação do Agente Treinado – Ambiente Estático. Abreviações: RM = Recompensa Média, SR = Taxa de Sucesso (Success Rate).

Número do Teste	**RM – EKF-DQN s_{k+1}**	**SR – EKF-DQN s_{k+1}**
1º	22,36	67%
2º	23,25	100%
3º	23,63	100%
4º	24,13	100%
5º	24,42	100%
Média Final	**23,56**	**93,33%**

Nós avaliamos o desempenho do algoritmo para diferentes valores de passo p. No entanto, notamos que pior desempenho foi obtido para valores de p iguais ou superiores a 1. Portanto, para sermos concisos, usamos o passo s_{k+1}. Quando se usa outro passo, como s_{k+2}, ou seja, passando a previsão dois passos à frente do estado dado pelo EKF para repetição de memória, isso representa uma tentativa de acelerar ainda mais o processo. Esta tentativa propagou alguma instabilidade no treinamento, e por isso adotamos o valor de s_{k+1}. Vale mencionar que a taxa de sucesso apresentada na Tabela 6.3 foi calculada considerando todos os episódios de treinamento, portanto, seu valor é relativamente baixo devido à convergência do algoritmo na fase de exploração.

Os algoritmos DQN e EKF-DQN apresentam variáveis aleatórias, ou seja, a própria rede neural apresenta inicializações, principalmente de pesos sinápticos, que

dependem de funções probabilísticas e podem assumir valores diferentes para testes diferentes. Isso significa que, em um determinado teste, uma rede pode ser inicializada em uma região relativamente "boa", facilitando a aprendizagem do agente. Portanto, foi decidido realizar três testes diferentes para cada algoritmo, gerando assim maior confiabilidade nas comparações de desempenho. Outro método utilizado para analisar os resultados foi a estratégia de gerar uma única semente para o framework de números aleatórios no Python.

6.3.2 Avaliação via Teste Estatístico

O teste *t-Student* é um método estatístico utilizado para comparar valores médios entre dois experimentos diferentes, a fim de analisar a diferença entre os experimentos [Haslwanter 2016]. Nestes experimentos esta técnica é utilizada para confirmar se o desempenho do algoritmo EKF-DQN é superior ao do DQN. Para tanto, foram levantadas duas hipóteses e calculados os valores **t** e **p** (parâmetros) do teste *t-Student* para avaliar a média das taxas de sucesso dos experimentos, conforme mostrado na Tabela 6.3:

- Hipótese 0 (H_0): O desempenho do EKF-DQN não é superior ao do algoritmo DQN. Nesta hipótese, suas taxas de sucesso são iguais.
- Hipótese 1 (H_1): O desempenho do EKF-DQN é superior ao do algoritmo DQN. Nesta hipótese, suas taxas de sucesso são diferentes.

O valor-**p** calculado foi $0,0102$ e o valor-**t** foi -3.345 para os cinco testes independentes realizados. Este resultado indica que com um nível de confiança de 95% rejeitamos a hipótese nula, ou seja, conclui-se que a rede DQN realmente se beneficia da hibridização com a EKF com nível de significância de 5%.

6.4 Módulo de Segurança

Para avaliar o desempenho do módulo de segurança, cinco pessoas sintéticas foram adicionadas ao ambiente de simulação e circularam aleatoriamente pela pista. A função do módulo auxiliar é frear o veículo autônomo a fim de evitar colisões com essas pessoas. Para isso, a rede ResNet50 ajustada para essa finalidade atua como um detector de pessoas ao receber imagens da câmera RGB-D. Caso uma pessoa seja identificada próxima ao robô, a alimentação dos motores é interrompida.

A Figura 6.7 apresenta o cenário operacional do módulo de segurança. Basicamente, quando a probabilidade de presença de uma pessoa nas proximidades do robô atinge cerca de 90%, o módulo de segurança é acionado. Esse valor de limiar foi utilizado para evitar frenagens quando as pessoas estão muito distantes do robô. No entanto,

vale ressaltar que esse limite pode ser ajustado conforme o desempenho observado nas simulações.

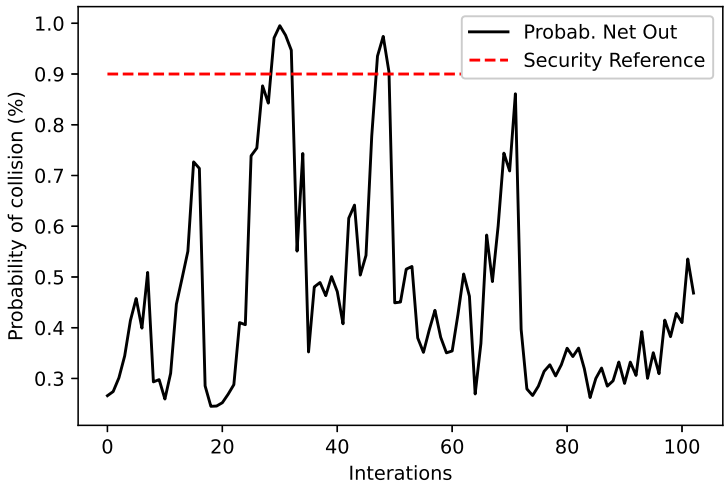


Figura 6.7: Probabilidade de Colisão e Atuação do Módulo de Segurança com pessoas em movimento – vídeo disponível em: https://bit.ly/3reEZrU.

Observa-se que, na simulação que originou o gráfico em questão, o módulo atuou duas vezes, ou seja, em ambas as ocasiões a probabilidade de ação ultrapassou o limite definido.

A Tabela 6.5 apresenta os resultados do agente atuando em ambiente dinâmico (ambiente com movimentação de pessoas). A fase de teste representa uma etapa posterior ao treinamento, em que os valores dos pesos do EKF-DQN estão fixos.

Tabela 6.5: Resultados do Agente Treinado no Ambiente Dinâmico.

Nº do Teste	**EKF-DQN s_{k+1}**		**EKF-DQN s_{k+1} + Módulo de Segurança**	
	RM	**SR**	**RM**	**SR**
1º	12,06	20%	18,63	60%
2º	20,84	40%	21,41	80%
3º	11,80	20%	21,56	80%
4º	8,30	20%	18,91	80%
5º	5,23	20%	15,42	60%
Média Final	11,84		19,18	
SR Total	24%		72%	

Considerando cinco episódios de teste, foram avaliadas as performances do controlador EKF-DQN sem o módulo de segurança e com o módulo de segurança. Observa-se que a recompensa média total sem o módulo de segurança foi de 11,84, enquanto com o módulo de segurança atingiu 19,18.

Nos cinco episódios de teste, o agente com o módulo de segurança conseguiu executar a missão proposta, apresentando uma taxa média de sucesso de 72%. Dessa

forma, pode-se afirmar que o módulo de segurança aumentou a taxa de sucesso em aproximadamente 200% quando comparado ao sistema de controle sem o módulo de segurança.

Um vídeo foi gravado para demonstrar o funcionamento do controlador EKF-DQN com o módulo de segurança em um ambiente com presença de pessoas em movimento. O vídeo pode ser acessado no link: `https://bit.ly/3reEZrU`.

As Figuras 6.8 e 6.9 ilustram o desempenho do robô durante a navegação autônoma utilizando o algoritmo EKF-DQN, em conjunto com o Módulo de Segurança e o Módulo de Fusão Sensorial. A Figura 6.8 apresenta uma sequência de quadros (frames) com a movimentação do robô ao longo do ambiente, indicada pelas setas azuis. Observa-se que o robô percorre os quatro espaços de trabalho (workspaces) definidos no cenário, enquanto pessoas também se movimentam, simulando um ambiente dinâmico. A ausência de colisões durante toda a trajetória evidencia a eficácia do Módulo de Segurança, que atua preventivamente ao detectar a presença de obstáculos móveis. A Figura 6.9, por sua vez, complementa essa análise ao exibir o traçado completo da trajetória seguida pelo robô, permitindo visualizar com maior clareza as regiões exploradas e os desvios realizados durante a missão.

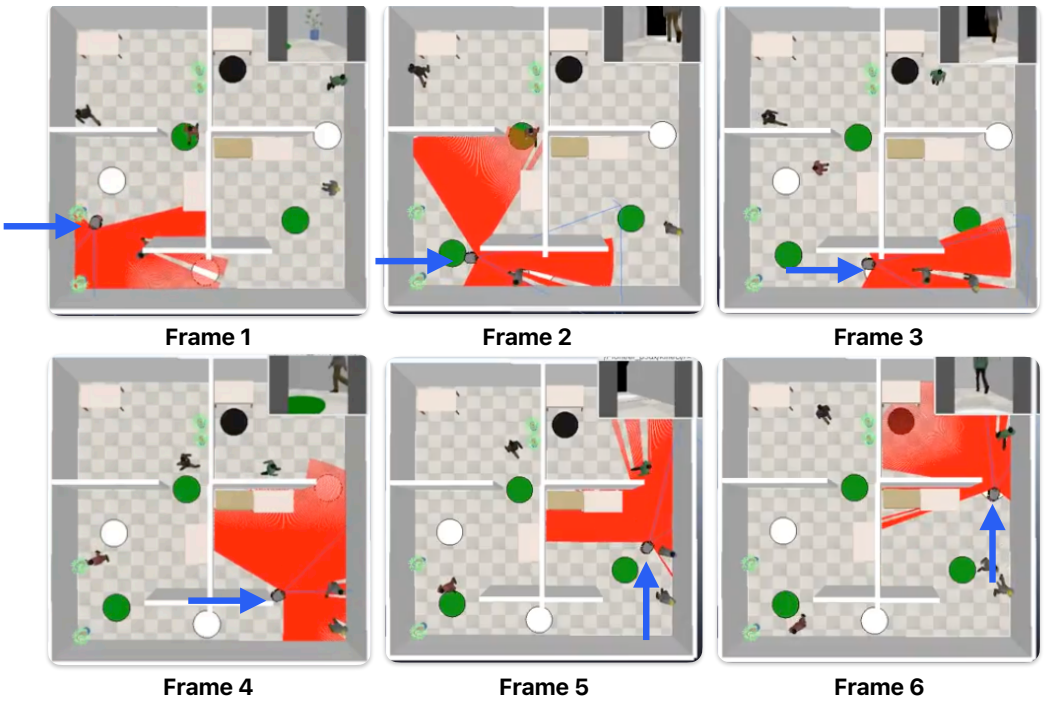


Figura 6.8: Trajetória Frame-Frame do Robô Móvel com Algoritmo EKF-DQN + Módulo de Segurança + Módulo de Fusão.

Figura 6.9: Trajetória do Robô Móvel com Algoritmo EKF-DQN + Módulo de Segurança + Módulo de Fusão.

6.5 Módulo de Comunicação: Long Range

Nesta seção, são apresentados e discutidos os resultados obtidos a partir do Módulo de Comunicação, com foco especial no algoritmo *Long Range*. São analisadas as etapas de formatação, envio e recuperação dos pacotes transmitidos via LoRaWAN. Além disso, são incluídas tabelas com métricas comparativas que permitem uma avaliação detalhada do desempenho da transmissão e da eficácia do sistema proposto.

Os resultados estão organizados da seguinte forma:

- **Resultados Iniciais: Teste de Bancada (T0)**
  Inicialmente, são apresentados os resultados obtidos no teste de bancada (T0), realizado em ambiente controlado, onde o gateway e o nó final foram posicionados sobre a mesma mesa, conectados a um computador desktop. Esses resultados fornecem uma linha de base para avaliar o desempenho do sistema em condições ideais, destacando a eficácia na transmissão e recuperação dos dados comprimidos no espaço latente.
- **Testes de Campo (T1, T2, T3)**
  Em seguida, são apresentados os resultados dos testes de campo (T1, T2, T3), realizados em um parque municipal, sob condições reais. Esses testes avaliam o impacto da variação de distância entre o nó final e o gateway, bem como as limitações associadas, incluindo falhas na entrega de pacotes e seus efeitos na recuperação dos dados.

Essa estrutura visa apresentar uma progressão clara desde condições ideais até cenários mais desafiadores, permitindo uma avaliação abrangente do desempenho e das limitações do sistema.

Também foi realizada uma análise da capacidade de reconstrução do autoencoder sobre os pacotes transmitidos e o processo de transmissão dos dados comprimidos pelo espaço latente na rede LoRaWAN. Essa análise inclui a avaliação da eficácia do autoencoder na compressão e reconstrução dos dados LiDAR, bem como a capacidade da rede em suportar esse tipo de transmissão com eficiência.

6.5.1 Avaliação do Desempenho do Autoencoder – Teste de Bancada (T0)

Após o processo de *JOIN*, o módulo RF LoRa inicia o envio das mensagens codificadas utilizando o algoritmo LoRa-ROS para o *gateway*. Como descrito anteriormente, esta subseção apresenta os resultados relacionados ao treinamento e à avaliação do desempenho do *autoencoder* no teste T0.

O processo de *JOIN* é uma etapa essencial no protocolo LoRaWAN, responsável por autenticar e registrar o dispositivo (nó final) na rede. Durante essa etapa, o nó envia uma requisição de associação (*Join Request*) contendo seu identificador e chave de autenticação. O *network server*, ao aceitar a requisição, responde com uma mensagem de aceitação (*Join Accept*), estabelecendo parâmetros como endereço do dispositivo, chaves de sessão e configurações de comunicação. Somente após essa autenticação bem-sucedida o nó passa a ter permissão para transmitir dados pela rede LoRaWAN.

A Figura 6.10 mostra a curva de treinamento do modelo autoencoder ao longo de 250 épocas. As curvas representam as perdas de treinamento e validação, que diminuem rapidamente no início, indicando aprendizado efetivo nas etapas iniciais. Com o aumento das épocas, os valores das perdas se estabilizam, sugerindo que o modelo está convergindo e não apresenta mais melhorias significativas. A proximidade entre as curvas de perda de treinamento e validação sugere que o modelo não sofre de sobreajuste e generaliza bem para dados não vistos.

O treinamento foi realizado utilizando um conjunto de dados com 515 leituras LiDAR obtidas no ambiente Gazebo. O conjunto foi dividido em 75% para treinamento, 10% para validação e 15% para teste, assegurando uma avaliação robusta do desempenho do modelo. O tamanho da entrada do modelo consiste em 360 neurônios ou leituras (360×4 bytes por ponto = 1440 bytes). A saída do codificador contém 64 neurônios (pontos no espaço latente), resultando em uma redução de carga útil para 64×4 bytes = 256 bytes.

A Figura 6.11 apresenta uma análise comparativa entre os dados LiDAR originais (lado esquerdo) e os dados reconstruídos pelo autoencoder (lado direito). Cada exemplo ilustra a intensidade em função do número de amostras, demonstrando o quão bem o autoencoder consegue replicar o sinal original após compressão e descompressão.

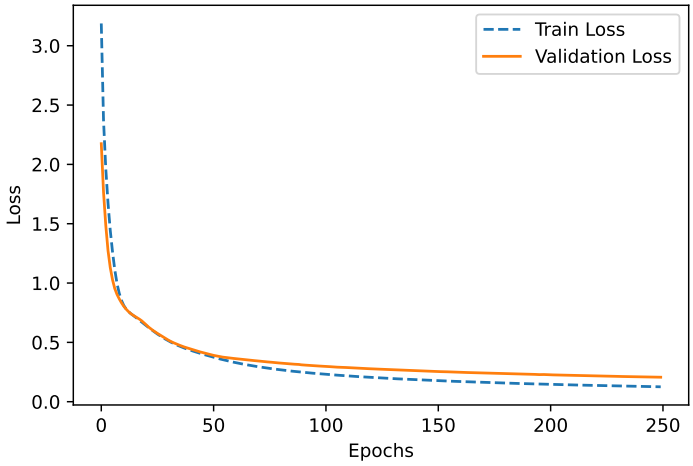


Figura 6.10: Evolução da Função Custo de Treinamento e Validação (*Train Loss and Validation Loss*) do Autoencoder em 250 Épocas.

Nos dois primeiros exemplos, os dados reconstruídos se aproximam bastante dos originais, evidenciando a eficácia do modelo na preservação das principais características. No entanto, nos exemplos de três a cinco, observa-se uma redução na qualidade da reconstrução, indicando limitações do modelo na codificação de padrões mais complexos. De modo geral, o autoencoder apresentou boa capacidade de compressão e reconstrução, com restrições em padrões menos frequentes.

A Figura 6.12 ilustra a matriz de correlação entre as variáveis de entrada e saída do conjunto de testes para a avaliação do autoencoder treinado. A matriz foi construída por meio do cálculo do coeficiente de correlação de Pearson (ρ) entre cada par de variáveis originais (x_i) e reconstruídas (y_j), conforme a equação:

$$\rho(x_i, y_j) = \frac{\text{Cov}(x_i, y_j)}{\sigma_{x_i} \cdot \sigma_{y_j}},$$

onde $\text{Cov}(x_i, y_j)$ é a covariância entre x_i e y_j, e σ_{x_i} e σ_{y_j} são os desvios padrão de x_i e y_j, respectivamente.

A matriz gerada demonstra a relação linear entre cada variável de entrada (leituras LiDAR originais) e de saída (reconstruídas). Valores de ρ próximos de 1 indicam forte correlação positiva, -1 indicam correlação negativa, e 0 indicam ausência de correlação linear. O conjunto de testes é composto por 78 medições, cada uma com 360 leituras.

Na imagem, observa-se uma diagonal central forte, que revela alta correlação entre as variáveis originais e suas reconstruções, indicando que o autoencoder foi capaz de preservar as principais características dos dados. A correlação média na diagonal principal foi de 83,74%.

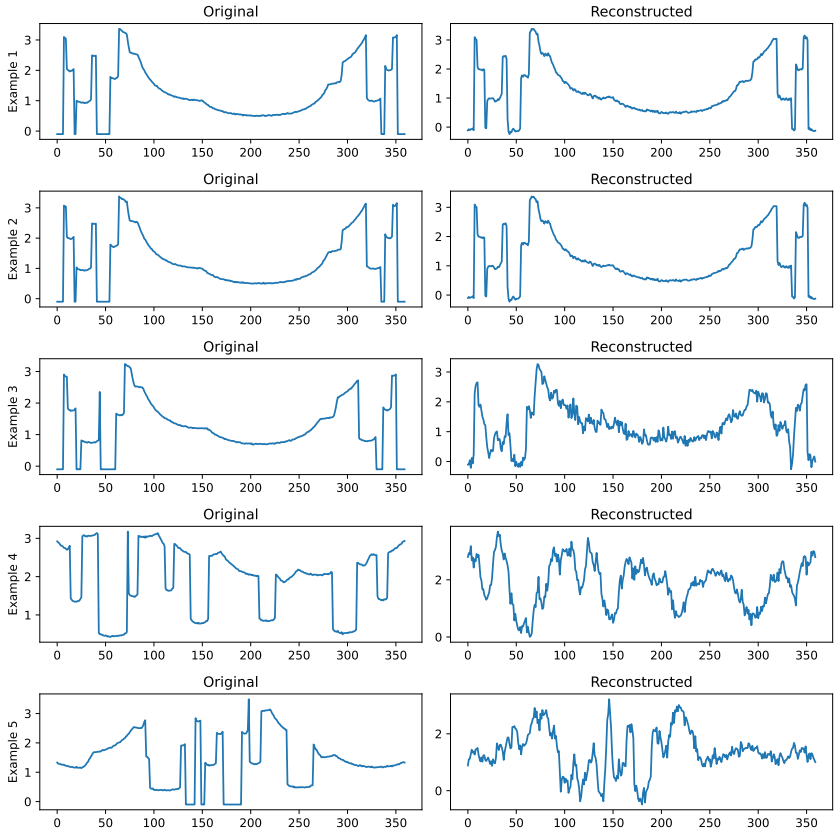


Figura 6.11: Comparação: Dados LiDAR Originais (Esquerda) e Reconstruídos pelo Autoencoder (Direita).

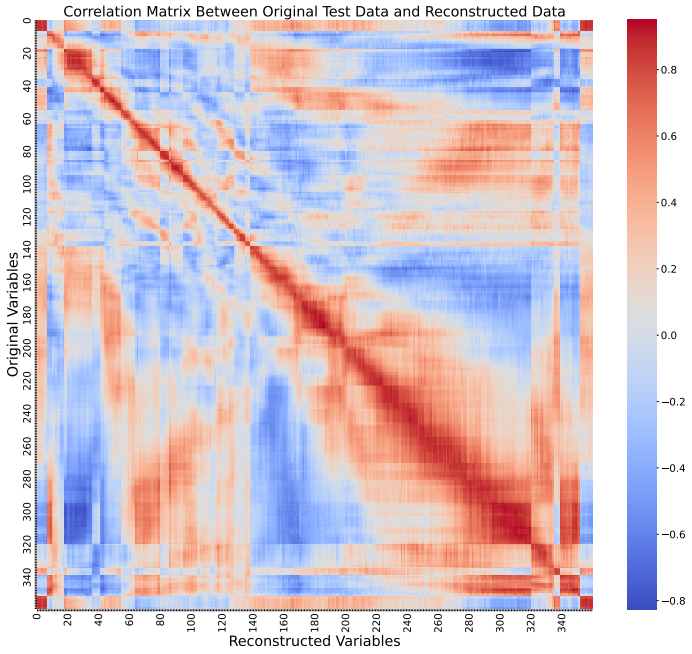


Figura 6.12: Matriz de Correlação entre os dados de teste originais e os dados reconstruídos.

6.5.2 Avaliação do Desempenho do LoRaWAN com Diferentes Distâncias (Testes T1, T2 e T3)

Após a avaliação do desempenho do autoencoder nos testes de treinamento e validação, o sistema foi testado em condições reais, com distâncias significativas entre o gateway e o módulo transmissor, para investigar o impacto na recuperação dos pacotes transmitidos.

A Tabela 6.6 apresenta as métricas relacionadas a cada quadro transmitido via LoRaWAN em diferentes distâncias (T0, T1, T2 e T3).

Tabela 6.6: Métricas por Quadro Enviado via LoRaWAN em Diferentes Distâncias.

Distância (m)	Métrica	Quadro 1	Quadro 2	Quadro 3	Quadro 4	Quadro 5	Quadro 6	Quadro 7
2 m (T0)	Tempo de Envio (ms)	4000,00	6000,00	6000,00	6000,00	8000,00	4000,00	6000,00
	Taxa de Transmissão (bits/s)	80,00	53,33	53,33	53,33	40,00	80,00	53,33
30 m (T1)	Tempo de Envio (ms)	9000,00	9000,00	9000,00	9000,00	9000,00	14000,00	-
	Taxa de Transmissão (bits/s)	35,56	35,56	35,56	35,56	35,56	22,86	-
50 m (T2)	Tempo de Envio (ms)	9000,00	9000,00	9000,00	9000,00	9000,00	9000,00	-
	Taxa de Transmissão (bits/s)	35,56	35,56	35,56	35,56	35,56	35,56	-
120 m (T3)	Tempo de Envio (ms)	130000,00	36000,00	9000,00	9000,00	-	-	-
	Taxa de Transmissão (bits/s)	2,46	8,89	35,56	35,56	-	-	-

Tabela 6.7: Métricas de Comunicação para o JSON Completo em Diferentes Distâncias.

Métrica	2 m	30 m (T1)	50 m (T2)	120 m (T3)
Total de Quadros Enviados	7	6	6	4
Total de Quadros Recebidos	7	6	6	4
PDR (Taxa de Entrega de Pacotes) (%)	100,00	85,71	85,71	57,14
Latência Total (ms)	40000,00	54000,00	54000,00	184000,00
Vazão (bits/s)	62,22	35,56	35,56	19,13

A escolha de um tamanho de pacote fixo de 40 bytes foi essencial para proporcionar maior flexibilidade no controle do Spreading Factor (SF), mesmo que esse fator não tenha sido alterado durante os testes, permanecendo fixado em SF7. Em redes LoRaWAN, o tamanho do payload pode limitar o uso de certos fatores de espalhamento, como SF7, SF8 e SF9. Pacotes maiores tendem a restringir a aplicação de fatores de espalhamento mais baixos, impactando negativamente a capacidade de transmissão e a eficiência energética.

Conforme observado no Teste T0 (teste de bancada), o tempo de transmissão dos quadros variou entre 4000ms e 8000ms, resultando em uma taxa de transmissão entre 40,00 bits/s e 80,00 bits/s. O tempo médio de transmissão dos quadros foi de 5714,28ms, enquanto a taxa média de transmissão foi de 58,85 bits/s, demonstrando um comportamento consistente em condições ideais de bancada. Nos testes de campo (T1, T2 e T3), observou-se uma diminuição progressiva na taxa de transmissão à medida que a distância aumentava. No pior caso, registrado no Teste T3, a taxa de transmissão caiu para 2,46 bits/s para o primeiro quadro, evidenciando o impacto das condições adversas.

Esse comportamento era esperado, uma vez que ambientes com obstáculos e maiores distâncias afetam diretamente as taxas de dados e a eficiência da comunicação. Além disso, foi observada uma perda significativa de pacotes (PDR abaixo de 100%) durante esses testes.

Outro ponto crítico é o aumento da latência. No Teste T3, o tempo total para transmissão do conjunto de quadros foi de 184000ms, o que pode ser crítico em aplicações que requerem baixa latência. Esses resultados evidenciam as limitações práticas da tecnologia LoRaWAN em ambientes urbanos com obstáculos.

Por fim, vale destacar os desafios impostos pelo ambiente de testes. O parque municipal, escolhido para os testes de campo, apresenta árvores como obstáculos naturais, que interferem na propagação do sinal. O gateway foi posicionado a uma altura de apenas 6 metros, limitando sua capacidade de manter algumas zonas de Fresnel desobstruídas, uma limitação que poderia ser mitigada com a instalação do gateway em uma altura maior. Além disso, o gateway foi instalado próximo a uma linha de distribuição de energia de baixa tensão, o que provavelmente causou atenuação do sinal, dificultando ainda mais a comunicação. Outro fator crítico é o uso de antenas omnidirecionais no módulo LoRa e no gateway. A adoção de uma antena direcional no gateway poderia potencialmente melhorar a qualidade do sinal ao direcionar a energia para a área de interesse e reduzir interferências externas.

6.5.3 Limitações do Algoritmo LoRa-ROS em Condições Reais

A Figura 6.13 mostra o gráfico dos dados em coordenadas polares, representando tanto as leituras originais do LiDAR no lado do robô ou veículo (em azul) quanto os dados transmitidos pela rede LoRaWAN e recuperados pelo decodificador (em vermelho). Na primeira figura, correspondente ao **teste T0** (teste de bancada), observa-se que as leituras recuperadas se aproximam das originais. Isso indica que o processo de codificação e decodificação preserva a integridade dos dados sob condições ideais, apesar das limitações de largura de banda da rede LoRaWAN.

Em contraste, a Figura 6.14 ilustra os resultados do **teste T2**, onde ocorreu perda de pacotes, resultando em uma Taxa de Entrega de Pacotes (PDR) de 86%. Essa perda resultou em dados de entrada incompletos para o autoencoder, impactando diretamente a completude do espaço latente. Especificamente, a perda de um quadro corresponde a uma redução de 16% na entrada da rede neural. Nesses casos, os dados ausentes precisam ser preenchidos (padding) para o funcionamento adequado da rede neural. O preenchimento foi feito copiando o quadro anterior ao ponto faltante.

Na Figura 6.14, observa-se que as intensidades do LiDAR reconstruídas pelo método proposto (vermelho) tentam seguir as leituras reais. No entanto, devido à ausência

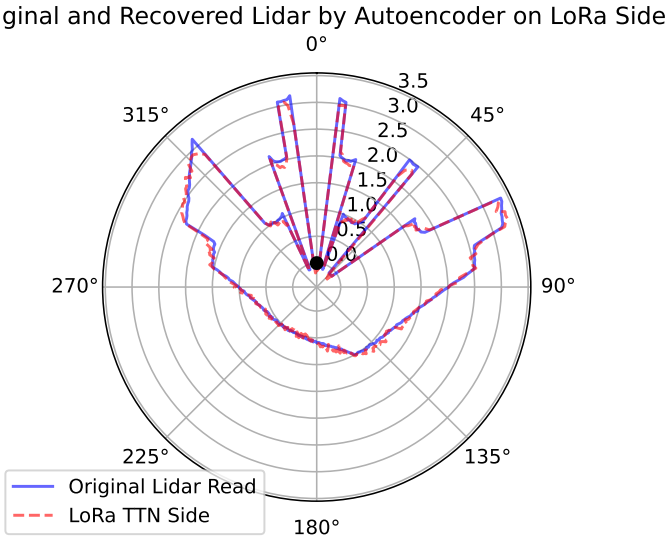


Figura 6.13: Dados LiDAR: Original (Azul) e Recuperado no receptor LoRa (Vermelho) - Coordenadas polares.

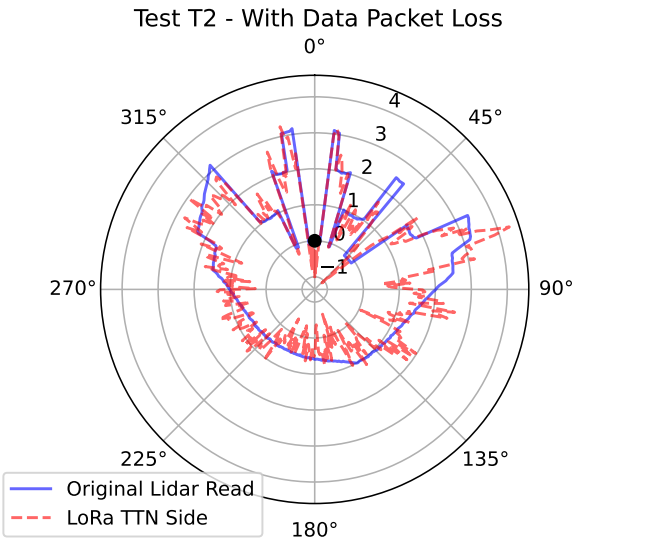


Figura 6.14: Dados LiDAR: Teste T2 com Perda de Pacotes - Coordenadas polares.

de um quadro na recepção (Teste T2), a saída apresenta ruído, mas ainda permitiu a reconstrução razoavelmente precisa do mapa. Para PDR inferiores a 86%, recomenda-se o processo de retransmissão de pacotes, visando corrigir os quadros ausentes.

Esse resultado destaca o potencial do método para tolerar certo nível de perda de pacotes mantendo a funcionalidade. Contudo, atingir uma taxa de entrega de pacotes (PDR) de 100% continua sendo o ideal para garantir a máxima precisão. Como melhoria futura, a incorporação de ruído no processo de treinamento do autoencoder poderia aprimorar sua capacidade de lidar com quadros perdidos durante a transmissão de dados, aumentando a robustez sob condições reais de rede.

6.5.4 Recuperação de Tópico ROS no Sistema de Visualização RViZ

A Figura 6.15 ilustra a recuperação dos dados do LiDAR recebidos via LoRaWAN, que foram publicados em um tópico ROS.

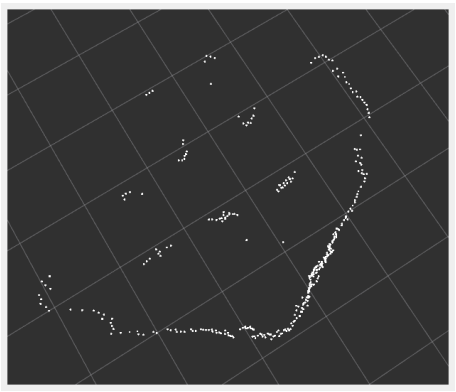


Figura 6.15: Dados LiDAR recuperados no receptor LoRa e publicados no RViz (ROS).

Esse tópico foi então visualizado utilizando o RViZ, uma ferramenta do ecossistema ROS2. A figura representa fielmente a área de navegação do robô, evidenciando com precisão suas posições iniciais. Esse resultado demonstra o potencial de integração entre o ROS2 e redes LoRaWAN, viabilizando o uso de ferramentas avançadas como o SLAM (Localização e Mapeamento Simultâneos). Como o tópico do escaneamento LiDAR está sendo corretamente publicado, torna-se possível explorar todo o conjunto de funcionalidades do ROS2 para aplicações avançadas de navegação e mapeamento em ambientes com conectividade limitada.

6.6 Controle do Fator de Espalhamento Espectral em Long Range

Inicialmente, foi realizada a conexão do *gateway* SX1302 com o servidor *The Things Network* (TTN). A Figura 6.16 dispõe a imagem do servidor apresentando conexão adequada do *gateway* (destaque vermelho).

As parametrizações para estabelecer a conexão são baseadas em identificadores e parâmetros essenciais do protocolo LoRaWAN, como o *gateway* EUI (*Extended Unique Identifier*), que é um identificador único global do *gateway*; o *Device EUI* (DEV EUI), um identificador único para cada dispositivo LoRa e o *Application EUI* (APP EUI), que associa dispositivos a uma aplicação específica no servidor. Adicionalmente, foi necessário configurar a chave de aplicação (*App Key*), que é usada para garantir a segurança na troca de mensagens entre o dispositivo e o servidor. Após a configuração destes parâmetros, o *gateway* foi registrado no servidor TTN e iniciou a comunicação.

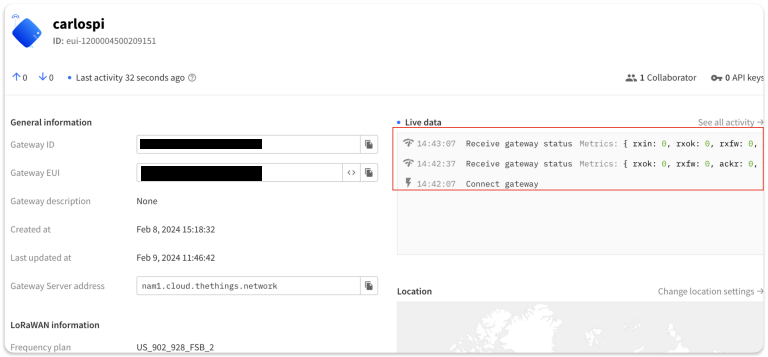


Figura 6.16: Conexão do gateway ao servidor LoRa-TTN.

No segundo passo, foi verificado a conexão dos *end-node* ao *gateway*. Para isso, foi necessário realizar o procedimento de *JOIN*, que é a fase de autenticação e associação dos *end-nodes* com o *gateway* LoRa. Durante esse processo, o *gateway* e o *end-node*, instalado em um robô móvel, foram posicionados em uma distância próxima (aproximadamente 10m). Esse arranjo serviu como teste preliminar que assegurou a comunicação entre os dispositivos e o servidor TTN. A Figura 6.17 apresenta a conexão adequada dos sensores de temperatura e umidade ao *gateway* LoRa. Neste instante, foi possível verificar três medições umidade/temperatura dispostas no servidor *cloud server*: 45% 30, 49% 29, 51%, 29.

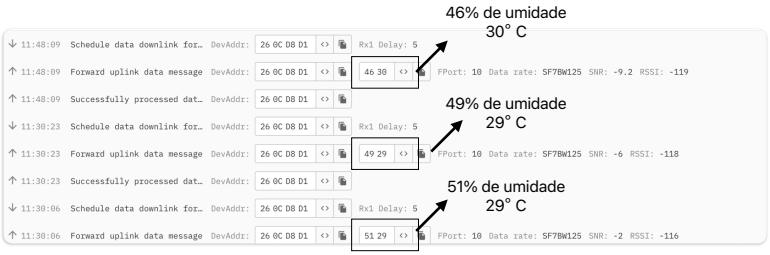


Figura 6.17: Conexão dos sensores de temperatura e umidade ao servidor LoRa-TTN.

O robô móvel equipado com transceptor LoRa e sensor DHT 11 para medição de temperatura e umidade se movimentou pelo cenário de avaliação da Figura **??**, iniciando pelo ponto P1 e seguindo até o ponto P5. Ao completar o percurso, avaliou-se as métricas de comunicação através do servidor TTN. Em cada enlace de comunicação LoRa bem sucedido foram geradas métricas de SNR, RSSI, bem como a SF utilizada na comunicação. Inicialmente, realizou-se o percurso com a parametrização do algoritmo ADR habilitado. A Figura 6.18 apresenta os resultados obtidos a partir do uso do algoritmo ADR.

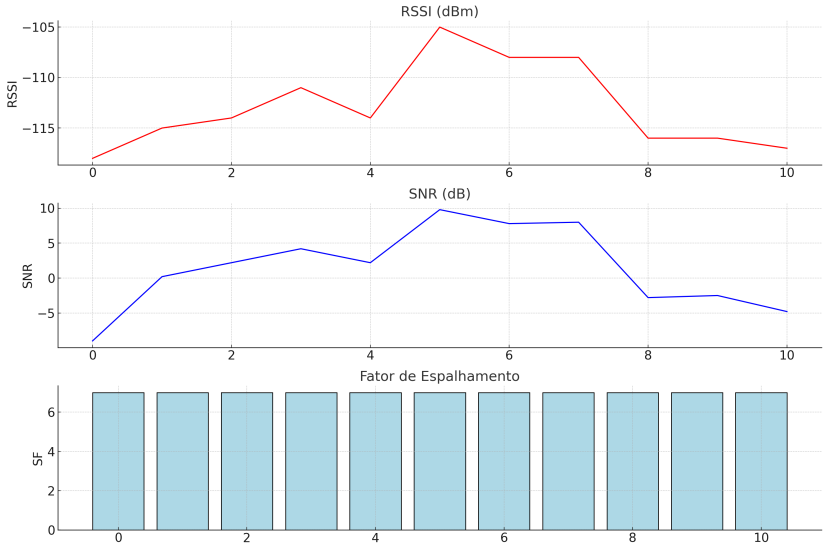


Figura 6.18: Métricas de desempenho: Movimentação do robô P1 - P5 (ADR)

Foi observado que, durante o percurso, apenas 11 medições sensoriais foram recebidas pelo *gateway* e apresentadas ao servidor TTN. Ademais, foi notado que houve perda de sinal do ponto P2 até o ponto P5 e, além disto, o ADR não fez troca de SF, insistindo com o fator de espalhamento espectral SF7. Na décima primeira medição notou-se uma queda acentuada de SNR, representando, precocemente, a perda de sinal.

Percebe-se na Figura 6.18 que todos os valores de medição foram com SF7. O uso do ADR não permitiu que o fator de espalhamento espectral fosse gerenciado adequadamente com o robô móvel em movimento, alcançando assim um *range* menor de cobertura. Uma possível explicação é que o algoritmo ADR necessita de um certo tempo e múltiplas leituras sensoriais para fazer o gerenciamento de SF, uma vez que o mesmo se baseia em leituras de SNR interativas. Com o robô em movimento esse método acaba sendo comprometido.

Outro teste, com o mesmo percurso e cenário de avaliação foi realizado. Desta vez fixou-se o SF8 para avaliar a performance deste fator de espalhamento espectral. Desta vez foram realizadas 16 leituras do trajeto de P1 até P5, destacando a mesma dificuldade de enlace de rádio entre os pontos P3 até P5. A Figura 6.19 apresenta os resultados do teste supracitado.

Ao fixar o SF observa-se certa melhora em relação ao ADR. O número de medições que o servidor recebeu foi relativamente maior, passando de 11 leituras (ADR) para 16 leituras (SF8). Isso quer dizer que o rádio LoRa atingiu *range* de cobertura maior. Apesar dos valores de SNR coletados serem mais baixos do que os valores registrados no primeiro teste, a mudança de SF (SF7 para SF8) garante demodulações com SNR mais baixos, não impactando em perda de enalce.

Pensando nisso, propõe-se uma técnica de alteração do SF interativa, computa-

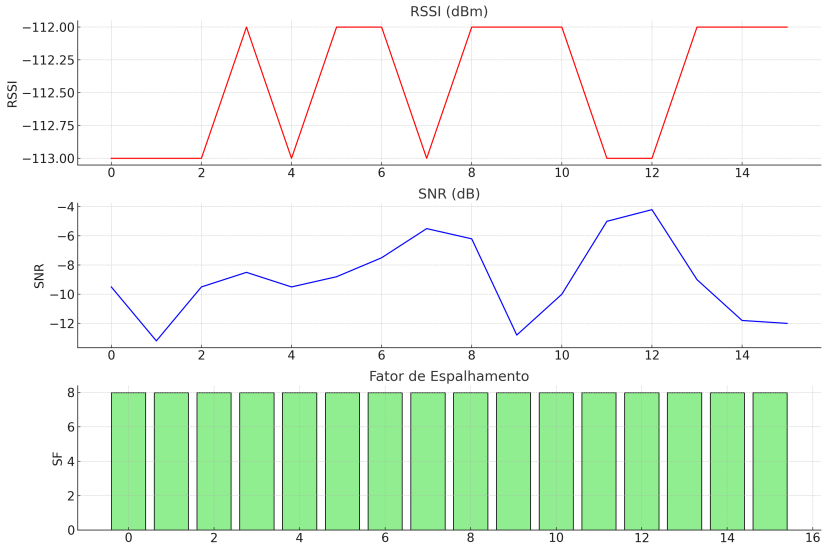


Figura 6.19: Métricas de desempenho: Movimentação do robô P1 - P5 (SF 8).

cionalmente simples e que não envolva o *feedback* do SNR. De acordo com o resultado apresentado na Figura 6.18, aguardar o *feedback* do SNR para decidir sobre a mudança de uma SF (o que o ADR faz) não aparenta ser uma boa estratégia. Sabe-se que o microcontrolador atua dentro de um *loop*, mandando sinais AT para o transceptor LoRa, que por sua vez envia o sinal ao *gateway*. Experimenta-se, então, fazer uma varredura de três SF durante o *loop* do microcontrolador. Ou seja, realiza-se a tentativa de envio do sinal com SFs diferentes, forçando a mudança de espalhamento espectral. Denomina-se esta técnica de Taxa Forçada de Dados (*Forced Data Rate*).

Foi selecionado, então, três SF para a tentativa do envio do sinal: SF7, SF8 e SF10. A SF7 apresenta a maior taxa de dados porém um range menor de atuação. Caso a mensagem não seja entregue com esta SF o microcontrolador solicita ao transceptor LoRa que o mesmo envie o mesmo sinal entretanto com a SF8. Caso a mensagem não seja entregue com esta SF o microcontrolador solicita ao transceptor LoRa que o mesmo envie o mesmo sinal, entretanto, com a SF10. Assim, forçamos o transceptor LoRa a passar por três fatores de espalhamento espectral diferentes sem necessariamente receber o *feedback* do SNR. A Figura 6.20 apresenta o resultado do *Forced Data Rate* proposto, no mesmo trajeto de movimentação do robô (P1 até P5).

Durante a movimentação do robô, observou-se uma maior faixa de cobertura, o que pode ser verificado pelo maior número de medições sensoriais que o servidor TTN recebeu (23 medições). Na Figura 6.20 observa-se ainda o maior aproveitamento dos fatores de espalhamento espectral durante a transmissão. Em determinadas regiões de navegação do robô, a mensagem foi enviada com SF8, SF7 ou SF10. Forçar o uso de outras SF, mesmo de forma não adaptativa, melhorou a faixa de cobertura e a entrega das mensagens do *end-node* até o *gateway*. Vale destacar que na região com baixa cobertura

Figura 6.20: Métricas de desempenho: Movimentação do robô P1 - P5 (Taxa Forçada de Dados).

(P3 até P5) o enlace LoRa permaneceu contínuo, mesmo com latência alta (cerca de 4 minutos de latência) e não houve perda de sinal.

Outros fatores devem ser destacados, como o cenário proposto para a movimentação do robô. Não temos um cenário plano, nivelado, o parque apresenta muitas subidas e descidas. Este desnivelamento contribui para um SNR relativamente baixo. Outro fator é a presença de muitas árvores e obstáculos, o que claramente aumenta o ruído de fundo e colisão de pacotes durante a transmissão. A dificuldade de posicionamento da antena do *gateway* também é um fator complicante.

6.7 Módulo de Comunicação: Short Range

A Figura 6.22 compara o desempenho entre o $MARL_{Q_{DDS}}$ (linha vermelha) e o Pure DDQN (linha azul), onde não há troca de experiências entre os robôs. Além disso, o método proposto também é comparado com uma estratégia de distribuição de experiências aleatórias, sem passar pela rede Q local ($MARL - Rand$).

Os resultados mostram a evolução da recompensa média ao longo das execuções bem-sucedidas para os dois métodos avaliados. Observa-se que o sistema $MARL_{Q_{DDS}}$, ao empregar uma estratégia de compartilhamento da rede Q através do *memory replay service*, atinge recompensas significativamente mais elevadas e estáveis ao longo do tempo. O vídeo disposto em `bit.ly/41QKjAj` apresenta a performance do modelo $MARL_{Q_{DDS}}$ no ambiente de simulação. Em termos de tempo de execução, na Figura 6.22, verifica-se que o $MARL_{Q_{DDS}}$ permanece ativo por períodos mais longos que os demais.

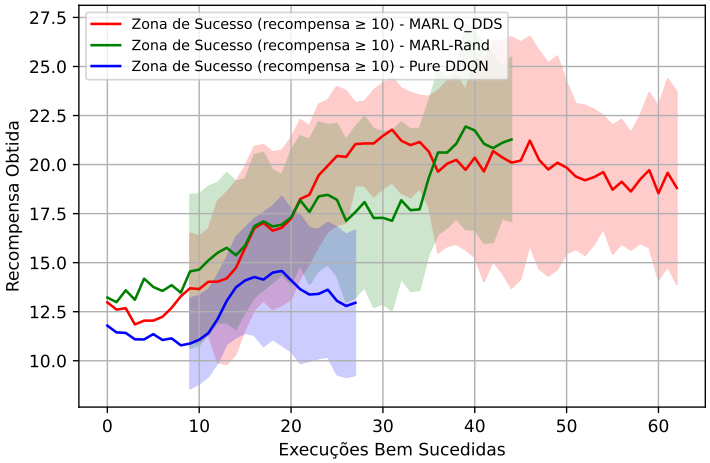


Figura 6.21: Comparativo entre curvas de aprendizado: $MARL_{Q_{DDS}}$, $MARL - Rand$ e Pure DDQN

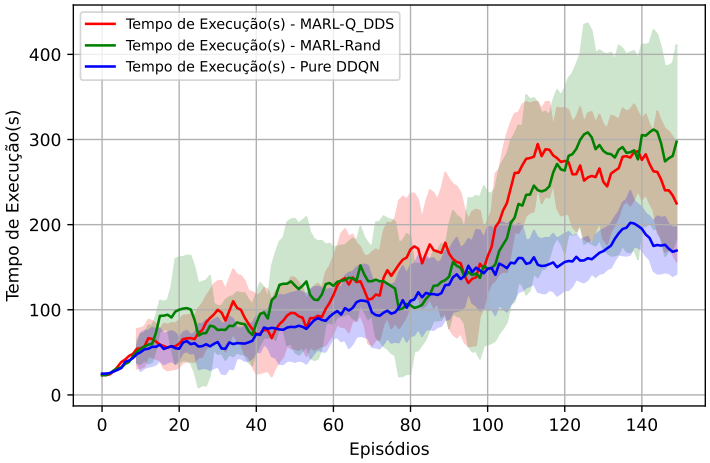


Figura 6.22: Comparativo entre curvas de tempo de execução: $MARL_{Q_{DDS}}$, $MARL - Rand$ e Pure DDQN

Embora isso possa sugerir maior custo computacional, tal aumento está diretamente associado ao **prolongamento da atuação no ambiente**, permitindo que os agentes explorem e concluam missões de maior complexidade. Essa permanência é consequência direta da efetividade da comunicação entre os agentes, diferentemente do Pure DDQN, que tende a encerrar sua atuação de forma prematura. A Tabela 6.8 apresenta o comparativo das técnicas $MARL_{Q_{DDS}}$, ($MARL_{rand}$) e o Pure DDQN. Verifica-se que o algoritmo proposto $MARL_{Q_{DDS}}$ provê uma taxa de sucesso consideravelmente maior do que as outras abordagens consideradas.

Tabela 6.8: Resumo comparativo entre MARL e DQN Puro na fase de treinamento.

Parâmetro	$MARL_{Q_{DDS}}$	**MARL-Rand**	**DDQN**
Tempo médio (s)	154.95	119.27	111.38
Desvio padrão (s)	97.06	78.12	57.06
Taxa de sucesso	**42.00**%	24.00%	18.67%
Execuções bem-sucedidas	**63**	36	28

6.7.1 Validação da Comunicação DDS

Como o CoppeliaSim não suporta DDS, validou-se a comunicação distribuída no ROS2, que adota DDS como middleware nativo. Para isso, os robôs compartilham transições completas de aprendizado (s, s', a, r, $done$) com um nó servidor de memória, conforme o esquema da Figura 5.15. A Tabela 6.9 resume as métricas da comunicação: a vazão média foi de $173.058 bps$, com 208 pacotes trocados e latência média de $0,5 s$. Essa avaliação foi realizada sobre uma rede Wi-Fi de $2.4 GHz$.

Tabela 6.9: Métricas da Comunicação DDS

Métrica	**Valor**
Vazão média (bps)	173058.62
Pacotes recebidos	208
Tamanho médio do pacote (bytes)	86529.00
Latência média (ms)	500

Dessa forma, os resultados apresentados neste capítulo demonstram, de forma quantitativa e qualitativa, os ganhos obtidos com a aplicação dos algoritmos desenvolvidos em cada módulo funcional da arquitetura proposta. As análises contemplaram desde a melhoria da navegação autônoma com técnicas de fusão sensorial e aceleração por predição de estado, até a viabilidade da comunicação colaborativa via *Long Range* e *Short Range*. As integrações realizadas, tanto em simulação quanto em ambiente real, indicam o potencial prático da proposta. No próximo capítulo, são discutidas as principais conclusões obtidas a partir dos experimentos, bem como as perspectivas para trabalhos futuros.

6.8 Demonstrações em Vídeo

Para complementar os resultados apresentados, esta seção destaca demonstrações visuais que evidenciam o funcionamento prático e a integração entre os módulos desenvolvidos no sistema de navegação autônoma proposto. Os vídeos a seguir apresentam o comportamento do agente em cenários simulados com múltiplos elementos do sistema operando em conjunto.

Os vídeos e arquivos de apoio relacionados à implementação prática da arquitetura proposta estão disponíveis no seguinte repositório público:

https://github.com/carlosdbez/tese-navegacao-autonoma

- **Vídeo 1 – Integração dos Módulos de Segurança, Aceleração e Fusão Sensorial:** Demonstra a atuação coordenada dos três principais blocos da arquitetura. O agente é capaz de identificar obstáculos, acelerar o processo de tomada de decisão via DRL otimizado (EKF-DQN), e reagir ao ambiente com base na fusão de sensores visuais e não visuais.
  Link: https://bit.ly/3reEZrU
- **Vídeo 2 – Integração dos Módulos com Comunicação Short Range (MARL-DDS):** Apresenta um cenário multiagente no qual robôs em vizinhança trocam informações sensoriais e estratégias por meio de uma rede distribuída DDS. A comunicação entre agentes possibilita decisões colaborativas e reforça a robustez da arquitetura proposta.
  Link: https://bit.ly/41QKjAj

CAPÍTULO 7

Conclusões e Trabalhos Futuros

7.1 Conclusões

Conforme apresentado no **Capítulo 5** (Sistema de Navegação Autônoma), os algoritmos que compõem a arquitetura proposta foram desenvolvidos de forma modular e submetidos a testes rigorosos, tanto de forma individual quanto por meio de integrações funcionais realizadas em ambiente simulado. Essa abordagem permitiu avaliar a interoperabilidade entre os módulos de navegação inteligente, fusão sensorial e segurança computacional em condições controladas. Adicionalmente, a camada de comunicação de longo alcance (LoRa) foi validada em ambiente real, respeitando as limitações impostas por interferências, obstáculos físicos e restrições de infraestrutura de campo. Dessa forma, as conclusões a seguir refletem o desempenho realista de cada componente e as interações viáveis entre eles, demonstrando a efetividade da proposta e sua capacidade de aplicação prática.

Em relação ao **Módulo de Fusão**, os testes mostraram que o método *Late Fusion* integrado ao DDQN obteve desempenho superior aos demais métodos testados. Enquanto o DDQN puro apresentou uma **Taxa de Sucesso (SR)** de aproximadamente 1,4%, e o DDQN com *Iterative Fusion* atingiu cerca de 11%, o DDQN-*Late Fusion* obteve uma **SR** de aproximadamente 28%. Esse resultado mostra que, embora [Duanmu et al. 2020] defenda a superioridade do método iterativo para tarefas de classificação, na aplicação em navegação autônoma com aprendizado por reforço, a abordagem de *Late Fusion* mostrou-se mais adequada. Conclui-se, portanto, que a escolha da técnica de fusão deve considerar o domínio específico da tarefa.

No **Módulo de Aceleração** de aprendizado, o algoritmo EKF-DQN demonstrou desempenho consistentemente superior ao DQN tradicional. Em cinco rodadas de teste com pesos fixos, o EKF-DQN apresentou maior **Recompensa Média (RM)** e maior **Taxa de Sucesso (SR)** em nove das dez métricas avaliadas. Além disso, o EKF forneceu predições de estado próximas às medições reais, mesmo em trajetórias com curvas, validando sua eficácia como estimador de estados em tempo real. Outra vantagem signifi-

cativa foi seu baixo custo computacional, o que o torna viável para aplicações embarcadas com recursos limitados.

No que se refere ao **Módulo de Segurança**, os resultados mostraram que a integração de uma rede ResNet50 com imagens RGB-D permitiu detectar pessoas próximas ao robô com alta confiabilidade. Quando o sistema identificava uma probabilidade de presença humana acima de 90%, o módulo acionava o freio de emergência, interrompendo a navegação para evitar colisões. Esse comportamento foi validado com sucesso em testes simulados com múltiplos agentes móveis, apresentando uma **SR** de 72%, contra 24% do mesmo sistema sem o módulo de segurança. Assim, o sistema provou ser uma alternativa eficaz para navegação segura em ambientes com presença humana.

Na camada de comunicação, os testes com o protocolo híbrido mostraram que a proposta é aplicável em cenários práticos. A comunicação *Short Range* entre robôs, implementada via *middleware* DDS, permitiu a troca de mensagens de estado em tempo real, com baixa latência e boa confiabilidade. Os resultados indicam, também, que o compartilhamento de experiências entre os robôs pode melhorar significativamente a convergência do aprendizado, permitindo que os agentes aprendam políticas mais rapidamente do que abordagens isoladas, como o *Pure DDQN*. No entanto, algumas limitações foram identificadas, incluindo o uso excessivo de memória durante as simulações, que impediu a continuidade dos testes por períodos mais longos. O $MARL_{Q_{DDS}}$ proposto superou em termos de performance o MARL-Rand e o DDQN (*Pure DDQN*) apresentando uma taxa de sucesso superior: 18% em relação ao MARL-Rand e 23.33% em relação ao *Pure DDQN*.

Na comunicação *Long Range*, o uso de módulos LoRa RA-08H em conjunto com o gateway SX1302 e Raspberry Pi 4 permitiu realizar transmissões em campo com diferentes configurações de Spreading Factor (SF). O ambiente de testes incluiu vegetação densa, moradias e outros obstáculos, o que resultou em perdas por *fading* e limitações no alcance. Ainda assim, a proposta se mostrou funcional, sendo a seleção do SF um ponto crítico.

Visando mitigar limitações da LoRa em aplicações com sensores de alta resolução, como LiDAR, foi proposta uma técnica de compressão baseada em autoencoders. Essa técnica atingiu uma média de 82% de redução no *payload*, viabilizando a transmissão dos dados comprimidos via LoRa com mínima perda de informação. Isso amplia o leque de sensores utilizáveis em arquiteturas robóticas conectadas a redes LPWAN.

A adoção de uma estrutura modular para o desenvolvimento da arquitetura de navegação autônoma mostrou-se uma escolha estratégica e vantajosa. Essa abordagem permitiu o desenvolvimento, teste e validação individualizada de cada componente funcional, como os algoritmos de controle por reforço, os métodos de fusão sensorial, o módulo de segurança baseado em visão e os sistemas de comunicação, facilitando a identificação de limitações e o aprimoramento de cada módulo de forma independente. Além disso, a

modularidade viabilizou integrações parciais em ambientes simulados e reais, demonstrando a flexibilidade da proposta e sua capacidade de adaptação a diferentes cenários. Essa organização modular também representa um passo fundamental para a futura integração plena do sistema, especialmente em plataformas robóticas reais com diferentes configurações, reforçando o potencial de escalabilidade e reutilização dos componentes desenvolvidos.

7.2 Trabalhos Futuros

Como trabalhos futuros, destaca-se a necessidade de realizar a integração completa de todos os módulos do sistema em um pacote unificado baseado em ROS 2, o que permitirá sua implementação em robôs reais com maior flexibilidade e portabilidade. Além disso, propõe-se a investigação de algoritmos de aprendizado por reforço multiagente (MARL) para permitir a coordenação entre múltiplos robôs em ambientes colaborativos. No contexto da comunicação, serão exploradas novas estratégias para o controle adaptativo do espalhamento espectral (Spreading Factor), a fim de aumentar a robustez da comunicação LoRa em cenários com alta mobilidade.

Também será avaliado o uso de tecnologias emergentes de comunicação de longo alcance, como o Narrow Band IoT (NB-IoT), para ampliar a cobertura e a confiabilidade das transmissões em aplicações remotas. Pretende-se ainda aprofundar a pesquisa na transmissão de imagens comprimidas por meio de autoencoders, avaliando algoritmos de reconstrução mais eficientes e escaláveis.

Por fim, propõe-se a aplicação de algoritmos avançados de controle de navegação, como o Proximal Policy Optimization (PPO), visando melhorar a estabilidade e eficiência do aprendizado em ambientes complexos e de alto grau de variabilidade.

Referências Bibliográficas

[Aguirre 2015]AGUIRRE, L. *Introdução à Identificação de Sistemas – Técnicas Lineares e Não-Lineares Aplicadas a Sistemas Reais*. [S.l.]: Editora UFMG, 2015. ISBN 9788570415844.

[Ahumada, Nettle e Solis 2013]AHUMADA, G. A.; NETTLE, C. J.; SOLIS, M. A. Accelerating q-learning through kalman filter estimations applied in a robocup ssl simulation. In: *2013 Latin American Robotics Symposium and Competition*. [S.l.: s.n.], 2013. p. 112–117.

[Althamary, Huang e Lin 2019]ALTHAMARY, I.; HUANG, C.-W.; LIN, P. A survey on multi-agent reinforcement learning methods for vehicular networks. In: *2019 15th International Wireless Communications Mobile Computing Conference (IWCMC)*. [S.l.: s.n.], 2019. p. 1154–1159.

[Andreja 2017]ANDREJA, R. Industry 4.0 concept: Background and overview. *International Journal of Interactive Mobile Technologies (iJIM)*, v. 11, 2017. ISSN 1865-7923.

[Arkin 1998]ARKIN, R. C. *Behavior-Based Robotics*. [S.l.]: MIT Press, 1998.

[Bank, Koenigstein e Giryes 2021]BANK, D.; KOENIGSTEIN, N.; GIRYES, R. *Autoencoders*. 2021. Disponível em: <https://arxiv.org/abs/2003.05991>.

[Batth, Nayyar e Nagpal 2018]BATTH, R. S.; NAYYAR, A.; NAGPAL, A. Internet of robotic things: Driving intelligent robotics of future - concept, architecture, applications and technologies. In: *2018 4th International Conference on Computing Sciences (ICCS)*. [S.l.: s.n.], 2018. p. 151–160.

[Bednarek, Kicki e Krzysztof 2020]BEDNAREK, M.; KICKI, P.; KRZYSZTOF, W. Robustness of multi-modal fusion—robotics perspective. *Electronics*, v. 7, p. 120–124, 2020.

[Belbachir e Benabid 2018]BELBACHIR, A.; BENABID, S. An efficient cooperative exploration strategy for wireless sensor network. *Intelligent Service Robotics*, v. 11, 07 2018.

[Benkahla et al. 2019]BENKAHLA, N. et al. Enhanced adr for lorawan networks with mobility. In: *2019 15th International Wireless Communications Mobile Computing Conference (IWCMC)*. [S.l.: s.n.], 2019. p. 1–6.

[Bezerra, Cardoso e Vieira 2025]BEZERRA, C. D. d. S.; CARDOSO, A. A.; VIEIRA, F. H. T. Utilizing autoencoders for latent representation and efficient transmission of lidar data via lora in ros. *IEEE Internet of Things Journal*, v. 12, n. 5, p. 4579–4590, 2025.

[Bezerra, Vieira e Soares 2024]BEZERRA, C. D. d. S.; VIEIRA, F. H. T.; SOARES, A. d. S. Deep-q-network hybridization with extended kalman filter for accelerate learning in autonomous navigation with auxiliary security module. *Transactions on Emerging Telecommunications Technologies*, v. 35, n. 2, p. e4946, 2024. Disponível em: <https://onlinelibrary.wiley.com/doi/abs/10.1002/ett.4946>.

[Bezerra et al. 2024]BEZERRA, C. D. de S. et al. Otimização da Área de cobertura lorawan para comunicação robótica r2x utilizando controle de espalhamento espectral. In: *Anais do LVI Simpósio Brasileiro de Pesquisa Operacional (SBPO 2024)*. Fortaleza, Brasil: Galoá, 2024. Acesso em: 14 Jul. 2025. Disponível em: <https://proceedings.science/sbpo/sbpo-2024/trabalhos/otimizacao-da-area-de-cobertura-lorawan-para-comunicacao-robotica-r2x-utilizando?lang=en>.

[Bezerra, Vieira e Carneiro 2023]BEZERRA, C. D. de S.; VIEIRA, F. H. T.; CARNEIRO, D. P. Q. Autonomous robotic navigation approach using deep q-network late fusion and people detection-based collision avoidance. *Applied Sciences*, v. 13, n. 22, 2023. ISSN 2076-3417. Disponível em: <https://www.mdpi.com/2076-3417/13/22/12350>.

[Cardona et al. 2020]CARDONA, M. et al. Mobile robots application against covid-19 pandemic. In: *2020 IEEE ANDESCON*. [S.l.: s.n.], 2020. p. 1–5.

[Carreras 2003]CARRERAS, M. *A Proposal of Behavior-Based Control Architeture With Reinforcement Learning for an Autonomous Underwater Robot*. Tese (Doutorado), Spain, 2003.

[Carreras et al. 2005]CARRERAS, M. et al. A behavior-based scheme using reinforcement learning for autonomous underwater vehicles. *IEEE Journal of Oceanic Engineering*, v. 30, n. 2, p. 416–427, 2005.

[Chen et al. 2020]CHEN, S. et al. A vision of c-v2x: Technologies, field testing, and challenges with chinese development. *IEEE Internet of Things Journal*, v. 7, n. 5, p. 3872–3881, 2020.

[Chollet et al. 2015]CHOLLET, F. et al. *Keras*. 2015. https://keras.io.

[Chrif e Kadda 2014]CHRIF, L.; KADDA, Z. M. Aircraft control system using lqg and lqr controller with optimal estimation-kalman filter design. *Procedia Engineering*, v. 80, p. 245–257, 2014. ISSN 1877-7058. 3rd International Symposium on Aircraft Airworthiness (ISAA 2013). Disponível em: <https://www.sciencedirect.com/science/article/pii/S1877705814011771>.

[Duanmu et al. 2020]DUANMU, H. et al. Prediction of pathological complete response to neoadjuvant chemotherapy in breast cancer using deep learning with integrative imaging, molecular and demographic data. In: *Medical Image Computing and Computer Assisted Intervention – MICCAI 2020: 23rd International Conference, Lima, Peru, October 4–8, 2020, Proceedings, Part II*. [S.l.]: Springer-Verlag, 2020. p. 242–252. ISBN 978-3-030-59712-2.

[Dudek e Jenkin 2010]DUDEK, G.; JENKIN, M. *Computational Principles of Mobile Robotics*. 2nd. ed. USA: Cambridge University Press, 2010. ISBN 0521692121.

[Elbakry et al. 2023]ELBAKRY, M. S. et al. Digital twin simulations for connected and automated vehicles: A comprehensive study. In: *2023 3rd International Conference on Electronic Engineering (ICEEM)*. [S.l.: s.n.], 2023. p. 1–6.

[Fayyad et al. 2020]FAYYAD, J. et al. Deep learning sensor fusion for autonomous vehicle perception and localization: A review. *Sensors*, v. 20, 2020.

[Feng et al. 2021]FENG, D. et al. Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges. *IEEE Transactions on Intelligent Transportation Systems*, v. 22, n. 3, p. 1341–1360, 2021.

[Fukushima e Miyake 1982]FUKUSHIMA, K.; MIYAKE, S. Neocognitron: A self-organizing neural network model for a mechanism of visual pattern recognition. *Competition and Cooperation in Neural Nets. Lecture Notes in Biomathematics*, p. 193–202, 1982.

[Geron 2019]GERON, A. (Ed.). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. USA: O'Reilly Media, 2019.

[Ghazali, Teoh e Rahiman 2021]GHAZALI, M. H. M.; TEOH, K.; RAHIMAN, W. A systematic review of real-time deployments of uav-based lora communication network. *IEEE Access*, v. 9, p. 124817–124830, 2021.

[Guo, Liu e Chen 2022]GUO, J.; LIU, Q.; CHEN, E. A deep reinforcement learning method for multimodal data fusion in action recognition. *IEEE Signal Processing Letters*, v. 29, p. 120–124, 2022.

[Haque et al. 2020]HAQUE, K. F. et al. Lora architecture for v2x communication: An experimental evaluation with vehicles on the move. *Sensors*, 2020.

[Haslwanter 2016]HASLWANTER, T. *An Introduction to Statistics with Python: With Applications in the Life Sciences*. Springer International Publishing, 2016. (Statistics and Computing). ISBN 9783319283166. Disponível em: <https://books.google.com.br/books?id=ZBi1DAAAQBAJ>.

[Hasselt, Guez e Silver 2016]HASSELT, H. van; GUEZ, A.; SILVER, D. Deep reinforcement learning with double q-learning. *Proceedings of the AAAI Conference on Artificial Intelligence*, v. 30, n. 1, Mar 2016.

[Hayes-Roth 1995]HAYES-ROTH, B. An architecture for adaptive intelligent systems. *Artif. Intell.*, v. 72, n. 1-2, p. 329–365, 1995.

[He et al. 2015]HE, K. et al. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[Hubel e Wiesel 1962]HUBEL, D.; WIESEL, T. Receptive fields, binocular interaction and functional architecture in cat's visual cortex. *J. Physiol.*, p. 106–154, 1962.

[Huh e Mohapatra 2024]HUH, D.; MOHAPATRA, P. *Multi-agent Reinforcement Learning: A Comprehensive Survey*. 2024. Disponível em: <https://arxiv.org/abs/2312.10256>.

[Jahromi, Theja e Cetin. 2020]JAHROMI, B. S.; THEJA, T.; CETIN., S. Real-time hybrid multi-sensor fusion framework for perception in autonomous vehicles. *Sensors*, v. 20, 2020.

[Jawhar et al. 2018]JAWHAR, I. et al. Networking of multi-robot systems: Architectures and requirements. *Journal of Sensor and Actuator Networks*, v. 7, n. 4, 2018. ISSN 2224-2708. Disponível em: <https://www.mdpi.com/2224-2708/7/4/52>.

[Jayasankar, Thirumal e Ponnurangam 2021]JAYASANKAR, U.; THIRUMAL, V.; PONNURANGAM, D. A survey on data compression techniques: From the perspective of data quality, coding schemes, data type and applications. *Journal of King Saud University - Computer and Information Sciences*, v. 33, n. 2, p. 119–140, 2021. ISSN 1319-1578. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1319157818301101>.

[Jiang, Su e Lu 2024]JIANG, J.; SU, K.; LU, Z. *Fully Decentralized Cooperative Multi-Agent Reinforcement Learning: A Survey*. 2024. Disponível em: <https://arxiv.org/abs/2401.04934>.

[Kalman e Others 1960]KALMAN, R. E.; OTHERS. A new approach to linear filtering and prediction problems. *Journal of basic Engineering*, v. 82, n. 1, p. 35–45, 1960.

[Karatzia, Kolios e Ellinas 2023]KARATZIA, M.; KOLIOS, P.; ELLINAS, G. Implementing mission-critical uav swarm coordination through the integration of lora and ros frameworks. In: *2023 International Conference on Information and Communication Technologies for Disaster Management (ICT-DM)*. [S.l.: s.n.], 2023. p. 1–7.

[Kober, Bagnell e Peters 2013]KOBER, J.; BAGNELL, J. A.; PETERS, J. Reinforcement learning in robotics: A survey. *Int. J. Rob. Res.*, Sage Publications, Inc., USA, v. 32, n. 11, p. 1238–1274, sep 2013. ISSN 0278-3649. Disponível em: <https://doi.org/10.1177/0278364913495721>.

[Krohn, Beyleveld e Bassens 2019]KROHN, J.; BEYLEVELD, G.; BASSENS, A. *Deep Learning Illustrated: A Visual, Interactive Guide to Artificial Intelligence*. USA: Addison Wesley, 2019. (The Addison-Wesley data & analytics series). ISBN 9780135116692.

[LeCun et al. 1989]LECUN, Y. et al. Backpropagation applied to handwritten zip code recognition. *Neural Comput.*, MIT Press, Cambridge, MA, USA, v. 1, n. 4, p. 541–551, dec 1989. ISSN 0899-7667. Disponível em: <https://doi.org/10.1162/neco.1989.1.4.541>.

[Lehong et al. 2020]LEHONG, C. et al. A survey of lorawan adaptive data rate algorithms for possible optimization. In: *2020 2nd International Multidisciplinary Information Technology and Engineering Conference (IMITEC)*. [S.l.: s.n.], 2020. p. 1–9.

[Lichota, Dul e Karbowski 2020]LICHOTA, P.; DUL, F.; KARBOWSKI, A. System identification and lqr controller design with incomplete state observation for aircraft trajectory tracking. *Energies*, v. 13, n. 20, 2020. ISSN 1996-1073. Disponível em: <https://www.mdpi.com/1996-1073/13/20/5354>.

[Liu et al. 2018]LIU, Y. et al. Formal analysis and verification of DDS in ROS2. In: *2018 IEEE*. [S.l.]: IEEE, 2018.

[Luber, Spinello e Arras 2011]LUBER, M.; SPINELLO, L.; ARRAS, K. People tracking in rgb-d data with on-line boosted target models. In: . [S.l.: s.n.], 2011. p. 3844–3849.

[Lynch e Park 2017]LYNCH, K. M.; PARK, F. C. *Modern Robotics: Mechanics, Planning, and Control*. 1st. ed. Cambridge, United Kingdom: Cambridge University Press, 2017.

[Macenski et al. 2022]MACENSKI, S. et al. Robot operating system 2: Design, architecture, and uses in the wild. *Science Robotics*, v. 7, n. 66, p. eabm6074, 2022. Disponível em: <https://www.science.org/doi/abs/10.1126/scirobotics.abm6074>.

[Macenski et al. 2020]MACENSKI, S. et al. The marathon 2: A navigation system. In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. [s.n.], 2020. Disponível em: <https://github.com/ros-planning/navigation2>.

[Manuel, Faied e Krishnan 2022]MANUEL, M. P.; FAIED, M.; KRISHNAN, M. A novel lora lpwan-based communication architecture for search rescue missions. *IEEE Access*, v. 10, p. 57596–57607, 2022.

[McGurk e MacDonald 1976]MCGURK, H.; MACDONALD, J. Hearing lips and seeing voices. *Nature*, v. 264, p. 746–748, 12 1976.

[Michael, Bodo e Vittorio 2019]MICHAEL, Y.; BODO, R.; VITTORIO, M. (Ed.). *Multimodal Scene Understanding*. USA: O'Reilly Media, 2019.

[Minsky e Papert 1969]MINSKY, M.; PAPERT, S. A. *Perceptrons: An Introduction to Computational Geometry*. [S.l.]: The MIT Press, 1969. ISBN 0262534770.

[Mohd et al. 2021]MOHD, J. et al. Substantial capabilities of robotics in enchancing industry 4.0 implementation. *Conitive Robotics*, v. 1, 2021. ISSN 2667-2413.

[Okinaka et al. 1975]OKINAKA, A. et al. Aloha packet broadcasting - a retrospect. In: *Managing Requirements Knowledge, International Workshop on*. Los Alamitos, CA, USA: IEEE Computer Society, 1975. p. 203. Disponível em: <https://doi.ieeecomputersociety.org/10.1109/AFIPS.1975.17>.

[Osorio et al. 2014]OSORIO, F. et al. *Robótica Móvel*. [S.l.]: Editora LTC, 2014. ISBN 9788521623038.

[Ovidiu et al. 2020]OVIDIU, V. et al. Internet of robotic things intelligent connectivity and platforms. *Frontiers in Robotics and AI*, 2020.

[Park, Lee e I.Joe 2020]PARK, G.; LEE, W.; I.JOE. Network resource optimization with reinforcement learning for low power wide area networks. *EURASIP Journal on Wireless Communications and Networking*, n. 176, 2020.

[Pflaum e Gölzer 2018]PFLAUM, A. A.; GöLZER, P. The iot and digital transformation: Toward the data-driven enterprise. *IEEE Pervasive Computing*, v. 17, n. 1, p. 87–91, 2018.

[Raza, Kulkarni e Sooriyabandara 2017]Raza, U.; Kulkarni, P.; Sooriyabandara, M. Low power wide area networks: An overview. *IEEE Communications Surveys Tutorials*, v. 19, n. 2, p. 855–873, 2017.

[Rohmer, Singh e Freese 2013]Rohmer, E.; Singh, S. P. N.; Freese, M. V-rep: A versatile and scalable robot simulation framework. In: *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. [S.l.: s.n.], 2013. p. 1321–1326.

[Romeo et al. 2020]ROMEO, L. et al. Internet of robotic things in industry 4.0: Applications, issues and challenges. In: *2020 7th International Conference on Control, Decision and Information Technologies (CoDIT)*. [S.l.: s.n.], 2020. v. 1, p. 177–182.

[Rosenblatt 1958]ROSENBLATT, F. The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, p. 65–386, 1958.

[Rumelhart, Hinton e Williams 1986]RUMELHART, D. E.; HINTON, G. E.; WILLIAMS, R. J. Learning Representations by Back-propagating Errors. *Nature*, v. 323, n. 6088, p. 533–536, 1986. Disponível em: <http://www.nature.com/articles/323533a0>.

[Rusu et al. 2016]RUSU, A. A. et al. *Policy Distillation*. 2016. Disponível em: <https://arxiv.org/abs/1511.06295>.

[Ryu e Kim 2021]RYU, S.; KIM, S.-C. Embedded identification of surface based on multirate sensor fusion with deep neural network. *IEEE Embedded Systems Letters*, v. 13, n. 2, p. 49–52, 2021.

[Santos, Silvestre e Cunha 2023]SANTOS, F.; SILVESTRE, D.; CUNHA, R. Development and experimental validation of a lora wireless sensor network for wildfire surveillance. In: *2023 IEEE Conference on Control Technology and Applications (CCTA)*. [S.l.: s.n.], 2023. p. 911–917.

[Schumann et al. 2017]SCHUMANN, C.-A. et al. Digital transformation and industry 4.0 as a complex and eclectic change. In: . [S.l.: s.n.], 2017.

[Semtech 2019]SEMTECH. *AN1200.22 LoRa Modulation Basics*. Semtech Corporation, 7 2019.

[Semtech Corporation 2020]Semtech Corporation. *SX1302 LoRa Gateway Baseband Processor Datasheet Rev 1.2*. Camarillo, CA, October 2020. DS.SX1302.W.APP. Disponível em: <https://www.semtech.com/>.

[Shenzhen Ai-Thinker Technology Co., Ltd 2022]Shenzhen Ai-Thinker Technology Co., Ltd. *Ra-08H Specification V1.1.0*. Shenzhen, China, January 2022. Version V1.1.0.

[Srichandan, Dhingra e Hota 2021]SRICHANDAN, A.; DHINGRA, J.; HOTA, K. An improved q-learning approach with kalman filter for self-balancing robot using openai. *J Control Autom Electr Syst 32, 1521–1530*, v. 32, n. 1, Aug 2021.

[Stefano. 2021]STEFANO., N. *Behavioral and Cognitive Robotics: An Adaptive Perspective*. [S.l.]: Institute of Cognitive Sciences and Technologies, National Research Council (CNR-ISTC), 2021. ISBN 9791220082372.

[Sutton e Barto 2018]SUTTON, R. S.; BARTO, A. G. *Reinforcement Learning: An Introduction*. Second. [S.l.]: The MIT Press, 2018.

[Tobin et al. 2017]TOBIN, J. et al. *Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World*. 2017.

[Vermesan et al. 2020]VERMESAN, O. et al. Internet of robotic things intelligent connectivity and platforms. *Frontiers in Robotics and AI*, v. 7, p. 120–124, 2020.

[Villa et al. 2021]VILLA, D. et al. Internet of robotic things: Current technologies, applications, challenges and future directions. *arXiv preprint arXiv:2101.06256*, 2021. Disponível em: <https://arxiv.org/abs/2101.06256>.

[Villa et al. 2021]VILLA, D. et al. *Internet of Robotic Things: Current Technologies, Applications, Challenges and Future Directions*. arXiv, 2021. Disponível em: <https://arxiv.org/abs/2101.06256>.

[Väänänen e Hämäläinen 2021]VääNäNEN, O.; HäMäLäINEN, T. Lora-based sensor node energy consumption with data compression. In: *2021 IEEE International Workshop on Metrology for Industry 4.0 IoT (MetroInd4.0IoT)*. [S.l.: s.n.], 2021. p. 6–11.

[Xu et al. 2020]Xu, Z. et al. S-mac: Achieving high scalability via adaptive scheduling in lpwan. In: *IEEE INFOCOM 2020 - IEEE Conference on Computer Communications*. [S.l.: s.n.], 2020. p. 506–515.

[Yi et al. 2022]YI, Y. et al. *Learning to Share in Multi-Agent Reinforcement Learning*. 2022. Disponível em: <https://arxiv.org/abs/2112.08702>.

[Zhang et al. 2021]ZHANG, A. et al. Dive into deep learning. *arXiv preprint arXiv:2106.11342*, 2021.

[Zhang et al. 2024]ZHANG, J. et al. Comparison of middlewares in edge-to-edge and edge-to-cloud communication for distributed ros 2 systems. *Journal of Intelligent amp; Robotic Systems*, Springer Science and Business Media LLC, v. 110, n. 4, 2024. ISSN 1573-0409. Disponível em: <http://dx.doi.org/10.1007/s10846-024-02187-z>.

[Zhao, Queralta e Westerlund 2020]ZHAO, W.; QUERALTA, J. P.; WESTERLUND, T. Sim-to-real transfer in deep reinforcement learning for robotics: a survey. *CoRR*, abs/2009.13303, 2020. Disponível em: <https://arxiv.org/abs/2009.13303>.

[Zhu e Zhang 2021]ZHU, K.; ZHANG, T. Deep reinforcement learning based mobile robot navigation: A review. *Tsinghua Science and Technology*, v. 26, n. 5, p. 674–691, 2021.

[Zhu et al. 2023]ZHU, W. et al. A survey of sim-to-real transfer techniques applied to reinforcement learning for bioinspired robots. *IEEE Transactions on Neural Networks and Learning Systems*, v. 34, n. 7, p. 3444–3459, 2023.