UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO DE JOINVILLE
CURSO DE ENGENHARIA MECATRÔNICA

VALDOMIRO BOTELHO JUNIOR

CONSTRUÇÃO DE UM ROBÔ AUTÔNOMO DE APLICAÇÃO DE REJUNTE

Joinville
2024

VALDOMIRO BOTELHO JUNIOR

CONSTRUÇÃO DE UM ROBÔ AUTÔNOMO DE APLICAÇÃO DE REJUNTE

Trabalho apresentado como requisito parcial para obtenção do título de bacharel em Engenharia Mecatrônica, no curso de Engenharia Mecatrônica, do Centro Tecnológico de Joinville, da Universidade Federal de Santa Catarina.

Orientador: Dr. Dalton Luiz Rech Vidor

Joinville

2024

RESUMO

Na construção civil, a aplicação de rejunte é um procedimento exaustivo e que requer precisão para garantir o acabamento. Atualmente, essa tarefa é efetuada manualmente por profissionais, o que demanda tempo e esforço físico, além de não ser uma posição adequadamente ergonômica para o trabalhador. A forma atual de aplicação de rejunte resulta em imperfeições, imprecisões, além de desperdício de material. Considerando esse contexto, surge a necessidade de buscar técnicas que automatizem esse processo. Diante desse desafio, apresenta-se a construção de um robô capaz de realizar a tarefa de aplicar rejunte em superfícies de forma autônoma, injetando a massa preparada de maneira precisa e uniforme no vão existente entre cerâmicas aplicadas em pisos, durante a construção de um edifício ou casa. O robô é capaz de identificar a lacuna entre as cerâmicas e a velocidade dos motores de tração. O processo de construção do robô contemplou a escolha de componentes como sensores, motores, atuadores, além de mecanismos necessários para garantir que o robô consiga desempenhar sua função e também um software embarcado em microcontrolador, para completo funcionamento autônomo do robô.

Palavras-chave: veículo autônomo; rejunte; construção civil.

ABSTRACT

In civil construction, grout application is an exhausting procedure that requires precision to ensure a proper finish. Currently, this task is performed manually by professionals, which demands time and physical effort, in addition to not being an adequately ergonomic position for the worker. The current form of grout application results in imperfections, inaccuracies, and waste of material. Considering this context, the need arises to seek techniques that automate this process. Given this challenge, the construction of a robot capable of performing the task of applying grout to surfaces autonomously is presented, injecting the prepared mass precisely and uniformly into the gap between ceramic tiles applied to floors, during the construction of a building or house. The robot is capable of identifying the gap between the tiles and the speed of the traction motors. The robot construction process included the choice of components such as sensors, motors, actuators, in addition to the mechanisms necessary to ensure that the robot can perform its function, as well as embedded software in a microcontroller for the robot's complete autonomous operation.

Keywords: autonomous vehicle; grout; civil construction.

LISTA DE FIGURAS

LISTA DE EQUAÇÕES

LISTA DE ABREVIATURAS E SIGLAS

ABNT – Associação Brasileira de Normas Técnicas

3D – Três dimensões

PI – Proporcional Integrativo

PID – Proporcional Integrativo Derivativo

PMW - Modulação por Largura de Pulso

GPIO – Global Purpose Input and Output

VAs – Veículos Autônomos

DSR - Design Science Research

PLA – Poliácido Láctico

CAD - Computer Aided Design

CC - Corrente contínua

CA – Corrente alternada

LiDAR - Light Detection and Ranging

SUMÁRIO

1. INTRODUÇÃO

A evolução tecnológica tem o potencial devidamente reconhecido para facilitar processos na construção civil (Ahn *et al.*, 2023). Conforme Rocha (2020), a inovação está presente de forma recorrente na indústria da construção, sendo necessária para garantir conforto e segurança, além de controle de custos. Com base no crescente desenvolvimento cientifico para esse setor, as pesquisas de inovação para a construção civil correspondem a uma área que ainda tem muito a ser explorada (Cutieru, 2021). Em vista disso, a análise no processo de aplicação de rejunte pode apontar uma proposta original.

Por definição, rejunte é nome dado à argamassa utilizada para tratamento em peças de porcelanatos, azulejos e outros materiais cerâmicos, com o propósito de compensar as diferenças entre os revestimentos, vedar a massa que fixa o azulejo e assegurar o acabamento final (Monteiro, 2006). Existem relatos de sua utilização desde a pré-história, pois foi descoberta no sul da Galileia um piso de pedra e argamassa que foi construído por volta de 11 mil anos atrás (Carasek, 2010).

A partir desse ponto, novos tipos de peças de cerâmica foram produzidos, há registros do emprego de argamassa em povos gregos, etruscos, egípcios e romanos. Com o tempo, outros métodos de produção de rejunte foram criados e novas tecnologias, com a utilização de cal e cimento foram incrementados ao rejunte, resultando em diferentes tipos de rejunte, com características distintas, que garantem o aproveitamento em cada tipo de aplicação (Carasek, 2010).

De acordo com Monteiro (2006), o rejunte pode ser categorizado em rejunte cimentício, rejunte acrílico e rejunte epóxi. Os preços dos rejuntes variam de acordo com sua composição, sendo o mais barato o cimentício e o mais caro o de epóxi. Logo, em certas ocasiões é viável a utilização do rejunte acrílico, uma vez que o resultado do acabamento é satisfatório, com valor intermediário entre os tipos existentes (Monteiro, 2006).

O rejunte cimentício é composto por cimento, polímero, minerais e pigmentos que tem a função de dar cores ao rejunte. O rejunte acrílico, além dos mesmos matérias do rejunte cimentício, conta com a adição de resina acrílica, o que eleva sua qualidade em relação ao rejunte cimentício. O rejunte epóxi, entretanto, é um composto do tipo bicomponente, cuja mistura de duas massas o forma, sendo que

sua aplicação é efetuada principalmente no interior de piscinas e locais muito úmidos, devido seu alto grau de impermeabilidade e resistência a manchas (Monteiro, 2006).

Conforme a NBR 13.753 da Associação Brasileira de Normas Técnicas (ABNT, 1996), revestimentos cerâmicos devem ser assentados com a utilização de rejunte, de forma que juntas entre cerâmicas sejam preenchidas e, assim, possam vedar o acesso entre a superfície e a base da cerâmica. Além da característica de vedação, a aplicação tem como finalidade dar acabamento e evita o acumulo de tensões devido à movimentação da estrutura (Bonafé, 2016).

Atualmente, a aplicação do rejunte consiste em um método manual, no qual se faz necessário o uso de um recipiente para preparar a massa, uma espátula e luvas de proteção para as mãos. O aplicador mistura a massa até atingir uma consistência maleável, se posiciona por cima da cerâmica já aplicada e com o auxílio de uma espátula, insere um pouco da massa entre os vãos existentes e de forma linear, cobrindo toda a área necessária (Tudo Construção, 2019).

Considerando isso, apresenta-se neste trabalho um veículo autônomo para a tarefa de oferecer a aplicação sem intervenção humana direta, bem como um sistema que objetiva a repetibilidade. A construção de um meio autônomo para aplicação de rejunte envolve os conhecimentos das áreas da robótica, física, microcontroladores e sistemas de controle, todas imprescindíveis para o funcionamento do robô.

Para dotar o robô da capacidade de responder a estímulos externos ao desempenhar à atividade, foram utilizados sensores para leitura de posição do aplicador de rejunte em relação à lacuna, bem como sensores de velocidade e distância percorrida. Conforme Wendling (2010), sensores tem como finalidade informar a um circuito elétrico dados relativos a eventos que ocorrem no mundo físico, para assim ser efetuado o monitoramento do sistema. Dessa forma, a utilização das leituras efetuadas pelos sensores possibilita a programação de tarefas que dependem de informações coletadas no ambiente.

Além dos sensores, o uso de atuadores do tipo motor elétrico e um microcontrolador são fundamentais no projeto de um robô. Atuadores são componentes que recebem um sinal procedente do controlador e atuam para alterar uma variável de estado do sistema a ser controlado (Roggia; Fuentes, 2016). No caso do robô aplicador de rejunte, os atuadores tem a finalidade de imprimir a movimentação do robô e também de realizar a injeção de rejunte nas lacunas entre as cerâmicas.

Para a leitura de dados coletados a partir dos sensores, como também para definir a tarefa de cada atuador, foi utilizado um microcontrolador. Como afirmado por Cardoso (2020), microcontroladores são equipamentos com memórias voláteis e não voláteis, processador e outros periféricos embarcados em uma única placa, tendo à disposição contadores e conversores que funcionam a partir de lógicas aritméticas definidas por meio de um programa. Com a integração de valores obtidos pelos sensores e também de uma lógica inserida no controlador, é possível gerar um programa que possibilite a movimentação do veículo, bem como a execução das tarefas de forma precisa e eficiente.

1.1. OBJETIVOS

Para resolver a problemática de aplicação de rejunte a partir de um robô autônomo, propõe-se os seguintes objetivos.

1.1.1. Objetivo Geral

Criar um veículo autônomo que fará a inserção do rejunte nos espaços existentes entre cerâmicas, preenchendo as lacunas de maneira uniforme, precisa e que não requeira intervenção humana direta.

1.1.2. Objetivos Específicos

- Desenvolver o projeto mecânico e eletroeletrônico do robô;
- Implementar o projeto estrutural utilizando ferramentas CAD;
- Projetar uma placa PCB para alocar os módulos eletrônicos necessários para componentes eletrônicos necessários para a finalidade do robô;
- Desenvolver um software capaz de controlar o robô aplicador de rejunte, acionando os motores, efetuando leituras dos sensores e gerenciando múltiplas tarefas concorrentes;
- Desenvolver e implementar um algoritmo de controle de velocidade dos motores de tração;

- Desenvolver um algoritmo de controle de posição que utilize da leitura de sensores para garantir a posição do bico de injeção em cima da lacuna a ser rejuntada;
- Integrar os sistemas do robô para que trabalhem com a finalidade de rejuntar o piso, controlando a velocidade e posição do veículo;

Com base nos objetivos propostos, foi elaborado o projeto do robô aplicador de rejunte dividindo o desenvolvimento em três principais abordagens: Projeto mecânico, projeto eletroeletrônico e projeto de software embarcado.

2. REVISÃO BIBLIOGRÁFICA

Esse capitulo abordará a automação na construção civil, com ênfase em veículos autônomos como alternativa para otimizar tarefas que tradicionalmente exigem mão de obra humana. Além disso, serão introduzidos os componentes eletrônicos que trabalham em conjunto na finalidade de prover a mobilidade e funcionalidade de veículos autônomos, detalhando os componentes cruciais na conversão de energia elétrica em movimento mecânico controlado.

2.1 AUTOMAÇÃO NA CONSTRUÇÃO CIVIL

Os padrões de vida vêm sofrendo mudanças desde a revolução industrial, atualmente, sistemas autônomos conseguem substituir tarefas rotineiras, perigosas e tediosas de produção (Hwang; Khoshnevis, 2005). A automação visa a utilização de tecnologias provenientes da informática, elétrica e sistemas de controle, juntamente com a integração entre sensores e atuadores para produzir características de repetição e precisão para tarefas que eram atribuídas a humanos. Essa perspectiva tem em vista eliminar os riscos resultantes da exposição humana em tarefas perigosas, como também remover o fator de erro humano, proporcionando assim a produtividade e eficiência (Santos, 2002).

> Uma boa definição para automação é um conjunto de técnicas destinadas a tornar automáticas a realização de tarefas, substituindo o gasto de bio-energia humana, com esforço muscular e mental, por elementos eletromecânicos computáveis [...]. Os benefícios para qualquer processo automação são nítidos: eficiência, segurança, menor custo, maior produção, etc. (SILEVIRA; LIMA, 2003, p. 1).

Na construção civil, existem estudos que abordam o uso de automação. Conforme hwang *et al.* (2005), é possível utilizar a tecnologia de impressão 3D para construção de estruturas com camadas de massa, no qual um sistema aplica diversas camadas de concreto, descartando a utilização de tijolos para criação de paredes, como também o uso de vigas para a sustentação do edifício. Dessa forma, fica evidente a necessidade de automação de processos, visto que trabalhos repetitivos são custosos em tempo, precisão e energia humana.

2.2 VEÍCULOS AUTÔNOMOS

Veículos autônomos (VAs) são sistemas robóticos que se locomovem e realizam tarefas sem intervenção humana direta. De acordo com Jafary *et al.* (2018), eles podem impactar a sociedade humana tanto de forma positiva quanto negativa. Os VAs utilizam de sensores para perceber o ambiente, possuem o potencial de se tornarem cruciais para os seres humanos. Além disso, eles utilizam de algoritmos para interpretar os dados dos sensores e se locomovem através de atuadores. A navegação autônoma é um dos principais desafios desses veículos, visto que envolve a capacidade de se localizar, planejar rotas e evitar obstáculos.

Segundo Liu *et al.* (2019), a segurança é o requisito principal quando se trata da implementação de veículos autônomos na sociedade. Nesse contexto, a utilização de hardware de ponta e a implementação de redundância nesse tipo de sistema é fundamental. Além disso, os sistemas dos VAs requerem processamento de dados em tempo de execução, a fim de garantir a tomada de decisão de maneira imediata e segura durante a sua operação.

Diante disso, é crucial investigar como utilizar de maneira segura e proveitosa a interação entre humanos. Nessa finalidade, veículos autônomos já estão sendo empregados tarefas que antes dependiam exclusivamente da mão de obra humana, como a título de exemplo, a agricultura. O avanço da tecnologia e a globalização tem requisitado cada vez mais desse tipo de equipamento, exigindo avanço de soluções para que viabilizem sua aplicação (Mcglynn; Walters, 2019).

2.3 MAQUINAS ELÉTRICAS: PRINCIPIOS E TECNOLOGIAS

Em maquinas elétricas, a transformação de energia elétrica em energia mecânica é fundamentada na interação entre campos magnéticos e correntes elétricas. Essa conversão ocorre quando uma corrente elétrica flui através de um condutor imerso em um campo magnético, resultando na geração de força motriz. Inversamente, a energia mecânica pode ser convertida em energia elétrica através da indução de corrente elétrica em um condutor, quando esse está em movimento dentro de um campo magnético (Fitzgerald *et al.*, 2014).

Essa interação entre campos magnéticos e correntes é a base para o funcionamento de máquinas elétricas, como motores e geradores. Nos motores, um

campo magnético é gerado, seja por imãs permanentes ou pela circulação de corrente nos enrolamentos presentes no rotor ou estator. Esse campo interage com o campo gerado pelo outro componente, estator ou rotor, respectivamente, resultando em um torque que impulsiona o movimento rotativo (Fitzgerald *et al.*, 2014).

2.3.1 Motor CC

De acordo com Fitzgerald *et al.* (2014), as máquinas elétricas de corrente continua (CC) são caracterizadas por sua versatilidade. Podem ser formadas por estator de imãs permanentes ou a partir de eletroímãs, essa última formação ainda garante dois tipos conexões, em derivação ou em série, cada uma com aplicações especificas. Seu funcionamento é embasado em princípios fundamentais da física, como a Lei de Faraday e a Lei de Lorentz.

Motores CC possuem acionamento através de corrente contínua e apresentam como vantagens o baixo custo para aquisição e a simplicidade do acionamento, visto que o torque depende apenas das características construtivas e da corrente de alimentação. No entanto, sua aplicação não é indicada em ambientes inflamáveis devido ao risco de faíscas serem geradas durante a comutação de escovas. Outra questão pertinente ao uso de motores CC é a durabilidade, visto que o uso de escovas gera desgaste, necessitando de manutenções. (CAMARGO, 2007)

2.3.2 Motor de Passo

Os motores de passo são dispositivos eletromecânicos amplamente utilizados em tarefas que demandem controle de posição. Seu princípio de funcionamento situa-se na conversão de pulsos elétricos em movimentos angulares precisos. O movimento desse tipo de motor é alcançado quando se aplica uma sequência de pulsos elétricos nas bobinas do estator, que por sua vez cria campos magnéticos que direcionam o rotor para a posição de menor relutância magnética, alcançando assim o movimento preciso e síncrono do motor em relação com a sequência de pulsos aplicados (Souza, 2021).

3.3.3 Servo Motor

A implementação de um servomotor consiste na integração de um motor elétrico, seja de corrente contínua (CC) ou corrente alternada (CA), com um transdutor de posição angular (encoder) e um sistema de controle em malha fechada. O encoder, geralmente um dispositivo óptico ou resistivo para essa aplicação, fornece feedback preciso da posição angular do eixo do motor ao sistema de controle. Este, por sua vez, efetua a comparação entre a posição real com a posição desejada, reduzindo o erro obtido através da comparação e ajustando a corrente fornecida ao motor.

3. PROJETO E DESENVOLVIMENTO DO VEÍCULO AUTÔNOMO

Este capítulo detalha o desenvolvimento do robô autônomo para aplicação de rejunte, abrangendo as etapas de projeto, desde a concepção da ideia inicial até a implementação final. São apresentadas a seleção de componentes, restrições necessárias para a implementação prática, os desafios enfrentados e as soluções adotadas para cada subsistema do robô. A metodologia a ser utilizada é a Design Science Research (DSR), devido seu foco em criações inovadoras que propendem a solucionar problemas do mundo real (Angeluci *et al*., 2020).

A metodologia DRS consiste em identificar um problema, propor uma solução inovadora, desenvolver, demonstrar e avaliar o artefato resultante com relação aos objetivos previamente definidos (Hevner *et al*., 2004). Com a identificação do problema, proposta de solução e definição dos requisitos já concluídos, esse capitulo terá o enfoque no desenvolvimento do artefato. Para detalhar cada etapa, esse capitulo será dividido em três abordagens principais: projeto mecânico, projeto eletroeletrônico e projeto de software embarcado. A integração desses subsistemas possui o potencial para resultar em um robô funcional e autônomo, capaz de realizar a aplicação de rejunte de forma eficiente e precisa. Atendendo aos requisitos definidos e validando a solução proposta.

3.1 RESTRIÇÕES E REQUISITOS DE PROJETO

Para a implementação prática do robô, algumas restrições foram impostas.

R001 - O robô deve se movimentar exclusivamente em linha reta, sem a necessidade de efetuar curvas e manobrar em diferentes sentidos;

R002 - O robô deve dispor de duas rodas de tração e um sistema de apoio baseado em roda livre;

R003 - A distância a ser percorrida durante a aplicação da mistura de rejunte deve ser definida pelo usuário antes do início da operação;

R004 - O robô deve ser capaz de injetar a massa através de um bico projetado em Computer Aided Design (CAD) e impresso em material PLA;

R005 - A massa deve ser constituída de rejunte acrílico ou mistura que simule a viscosidade do rejunte;
R006 - A massa deve ser inserida em uma bisnaga ou reservatório similar, estando previamente preparada para a aplicação;
R007 - Um bico injetor deve ser utilizado para transportar a massa entre o reservatório e o local de aplicação;
R008 - O sistema de extração da massa contida dentro do reservatório deve ser projetado e alocado no chassi do robô;
R009 - O robô deverá operar em pisos revestidos por cerâmica, devidamente assentados e nivelados;
R010 - O robô deverá ter uma interface de controle web embarcada;

3.2 PROJETO MECÂNICO

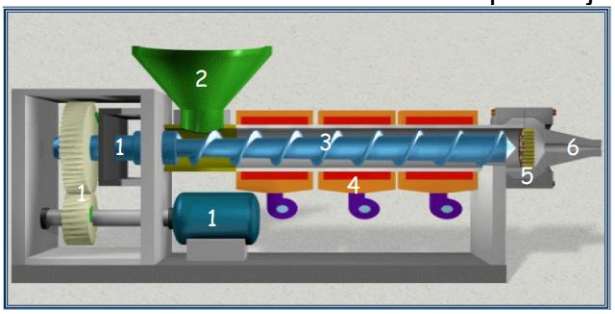
O projeto mecânico compreende o processo de desenvolvimento da parte estrutural do robô aplicador de rejunte. Inicialmente, serão discutidas as propostas de sistema de extrusão, com foco nas vantagens e desvantagens de cada mecanismo. Posteriormente, serão detalhados o projeto e a implementação do sistema de extrusão escolhido, o chassi do robô, a escolha das rodas de apoio e de tração, assim como o desenvolvimento do bico injetor e do rodo assentador.

3.2.1 Comparativo entre sistemas de extrusão: Fuso x rolos compressores

Um dos principais desafios no projeto mecânico do robô, situa-se na escolha do sistema de extração de rejunte. Foram avaliadas duas propostas, uma baseada no sistema de dois rolos compressores e outra baseada em extrusão por fuso. Cada versão possui vantagens objetivas e o estudo aprofundado em cada uma das vertentes propende a gerar resultados que devem impactar de maneira direta no desempenho e qualidade do acabamento.

Na Figura 1, é demonstrado o funcionamento de uma extrusora monorosca que utiliza o sistema fuso com finalidade de misturar e injetar materiais poliméricos através de uma matriz.

Figura 1 - Esquema de uma extrusora monorosca para injeção de plásticos



Fonte: Ferreira (2019).

De acordo com Ferreira (2019), uma extrusora é um equipamento que mistura e transporta materiais através de uma matriz para criar produtos, geralmente polímeros. O motor (1) fornece a força motriz necessária para girar a rosca (3) e assim impulsionar a massa, essa inserida através do funil de alimentação da extrusora (2). A extrusora ilustrada na figura possui a finalidade de injetar polímeros, logo em (4) é aplicada uma resistência para aquecimento, não necessária para a aplicação de rejunte. Por fim, em (5) e (6) estão localizados a matriz e o cabeçote, por onde o material é forçado a sair.

Em conformidade com Quelho (2018), o uso de extrusoras é um dos processos mais utilizados para fabricação de materiais termoplásticos. Acerca da utilização de um sistema de extrusão por rosca sem fim, pode-se citar dentre as vantagens a homogeneidade da massa durante a movimento da rosca, a quantidade de volume de massa que pode ser transportado de forma controlada, além da versatilidade desse tipo de sistema. No contexto específico do robô aplicador de rejunte, o sistema de extrusão por rosca sem fim, apesar os benefícios em termos de homogeneização e controle do fluxo, apresenta algumas desvantagens que precisam ser consideradas.

Em comparação com o sistema de rolo compressor, o mecanismo de extrusão se mostra mais complexo em termos de manutenção e limpeza. Além disso, a necessidade em projetar o sistema de extrusão devidamente dimensionado para suportar a torção do fuso e as forças de compressão, resulta em um custo mais elevado, o que pode ser um fator limitante para projetos com orçamento restrito. Por outro lado, o sistema de extrusão baseado em rolos compressores, demonstrado na Figura 2, se destaca devido a simplicidade em termos de construção.

Figura 2 - Sistema de rolo compressor

Fonte: ANODILAR (2022).

A manivela (1) é utilizada para girar os rolos e acionar o mecanismo de laminação (2), que consiste em dois cilindros paralelos que giram em sentidos opostos. Os ajustes de espessura (3) controlam o quão fina será fina será a espessura da massa. Por fim, a base (4) disponibiliza suporte para a estrutura dos rolos e do mecanismo de ajuste de espessura. Para a aplicação de rejunte, a manivela deve ser substituída por um motor que possua torque e potência suficientes para movimentar o êmbolo e extrair a massa contida em uma bisnaga.

Em virtude da pouca quantidade de peças móveis, o sistema pode ser construído de maneira ágil, com materiais acessíveis ou até mesmo impresso em 3D. Sua manutenção é simples e barata se comparado ao modelo de extrusão por fuso. Sua principal desvantagem está na necessidade de garantia de homogeneidade da massa antes da aplicação. Essa característica pode levar a variações na consistência do rejunte aplicado, comprometendo a qualidade do acabamento, visto que não ocorre uma mistura significativa durante o processo de compressão.

Devido o orçamento restrito do projeto, questões relacionadas a necessidade de limpeza e a viabilidade de impressão 3D, foi escolhida para a sequência desse projeto o sistema de dois rolos compressores para a aplicação de rejunte. A simplicidade construtiva do mecanismo envolvendo esse sistema mostrou-se factível de ser implementado, em comparação a complexidade de um sistema baseado em fuso, que demandaria recursos adicionais para sua produção.

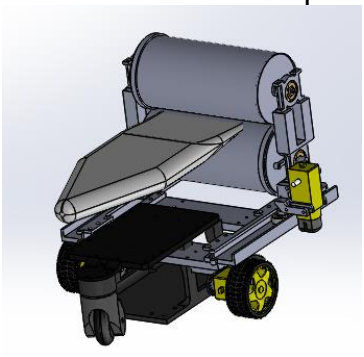
3.2.2 Desenvolvimento do sistema de rolo compressor

A primeira proposta baseia-se em um mecanismo composto de dois cilindros de 75mm de diâmetro e 50mm de comprimento cada, acoplados em um suporte

móvel. O mecanismo se desloca sobre um trilho, pressionando uma bisnaga fixada na estrutura do robô, e que armazena a massa pronta para a aplicação. A compressão da bisnaga gera pressão interna, impulsionando a massa para a cavidade de saída, em direção ao bico aplicador. De maneira simultânea, a força exercida durante a compressão da bisnaga gera uma força de tração no êmbolo, movendo-o em direção a parte fixada da bisnaga.

Rolamentos foram utilizados na finalidade de movimentar o êmbolo dentro do trilho e também para reduzir o atrito entre a estrutura e os rolos. Para essa proposta, foi projetada em CAD uma estrutura para o apoio dos rolos compressores, disponível na Figura 3. Em seguida, como apresentado na Figura 4, foi impresso em 3D um uma base para suportar duas latas que atuam na função de rolo compressor.

Figura 3 - Conceito inicial de robô aplicador de rejunte



Fonte: autoria própria.

Figura 4 - modelo de testes do conceito inicial

Fonte: autoria própria.

Com o protótipo em mãos, foram identificados desafios significativos relacionados ao conceito de torque. Durante os testes, notou-se que o torque gerado pelo motor não era suficiente para movimentar os rolos. De acordo com Halliday (2012), o torque τ é definido como a tendência de rotação ou torção que um corpo

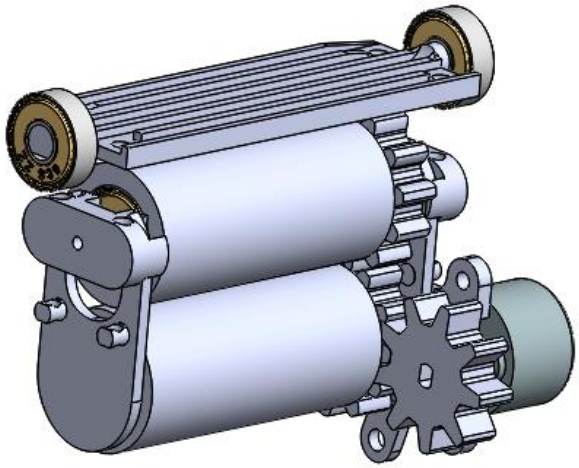
sofre quando submetido a uma força F, ortogonal a um raio de rotação r. O módulo do torque é:

$$\tau = r \times F = rF \sin\theta \quad (1)$$

Sabendo-se que o torque tem relação direta com o raio do rolo compressor, um cilindro de maior diâmetro exige um torque proporcionalmente maior para superar a resistência da massa a ser compactada. Isso resulta na necessidade de uma máquina elétrica com características construtivas adequadas de torque e potência, além de dimensionamento adequado das peças fabricadas em PLA para os esforços mecânicos não impactarem na ruptura da estrutura.

Com a identificação dos problemas apresentados na primeira proposta de design, os rolos foram redimensionados de forma que um tamanho menor se apresentou factível para o desenvolvimento da máquina. Dessa forma, como representada na Figura 5, foi projetada em CAD uma segunda proposta, com a utilização de um conjunto de dois rolos em escala menor, com 32.5mm de diâmetro e 80mm de comprimento cada, e um mecanismo de deslizamento por rolamentos localizado na parte superior da estrutura.

Figura 5 - Sistema de rolo compressor reduzido



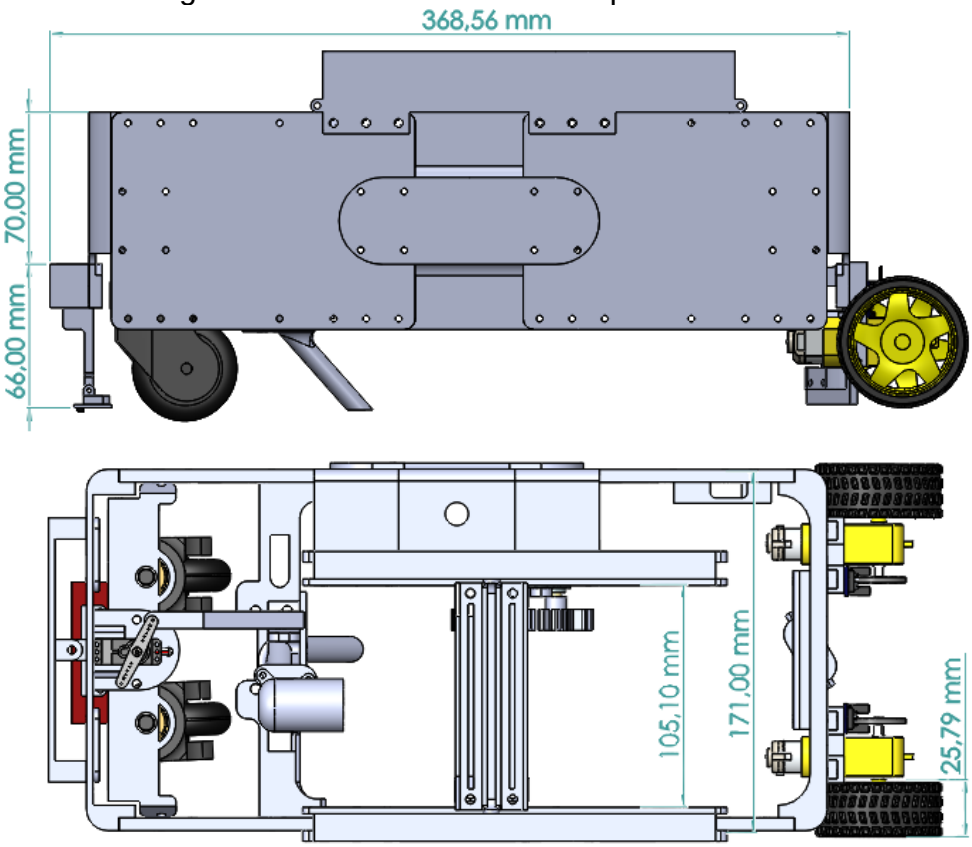
Fonte: autoria própria.

Essa proposta foi impressa em material PLA e com os devidos testes efetuados, percebeu-se que o embolo consegue tracionar de maneira eficiente uma bisnaga. Dessa forma, prosseguiu-se com a criação do chassi.

3.2.3 Chassi

Para o chassi do equipamento conforme a Figura 6, foi optado a utilização de 4 chapas retangulares de PLA, duas para cada lado do robô e projetadas para suportar futuras atualizações e alterações no design. Essas chapas foram interligadas a partir de uma peça auxiliar central, que atua como um elo estrutural entre a parte traseira do robô (rodas, motores de tração, sensores de posição da roda e parte traseira do trilho do embolo rolante) com a parte frontal do robô (bico injetor, sensores de refletância, mecanismo de subida do bico injetor, circuito elétrico, baterias e parte frontal do trilho).

Figura 6 - Vista lateral e vista superior do chassi



Fonte: autoria própria.

Dois trilhos foram dispostos na parte superior do robô, na finalidade de permitir o deslocamento linear do rolo compressor em direção à parte fixa da bisnaga. Na vista superior, nota-se que sua posição foi intencionalmente deslocada do centro

geométrico do modelo, essa necessidade surgiu devido o espaço necessário para acomodar o motor que traciona os rolos compressores.

Para o acoplamento das rodas frontais, foram produzidos no chassi dois engates, cada um com suporte para um rolamento e uma roda de giro livre, permitindo que o robô se movimente apenas com o controle das rodas traseiras, ajustando sua trajetória exclusivamente através da velocidade individual de cada roda. Foi necessária a utilização de duas rodas frontais devido o centro de massa do robô ser variável durante a aplicação do rejunte, visto que o sistema de extrusão se movimenta na direção ao bico, espremendo a bisnaga.

Visando a locomoção do veículo, optou-se por um sistema de tração composto por duas rodas motrizes, cada uma conectada a um motor com caixa de redução integrada. A escolha por rodas como o mecanismo de tração foi fundamentada na simplicidade do mecanismo, além da facilidade de integração com os motores. Alternativas como esteiras possuem maior complexidade de projeto e fabricação, em virtude da quantidade de peças necessárias para o desenvolvimento desse mecanismo, demandando maior investimento de tempo e recursos para o desenvolvimento.

Para alocar o modulo de baterias e a placa de circuito impresso (Figura 7), foi projetada em CAD uma tampa para ser acoplada na seção frontal do robô. A tampa, impressa em 3D, além da finalidade de abrigar a placa PCB e as baterias, também possui furações que permitem atualizações estruturais ou alocação de novos sensores.

Figura 7 - Tampa superior

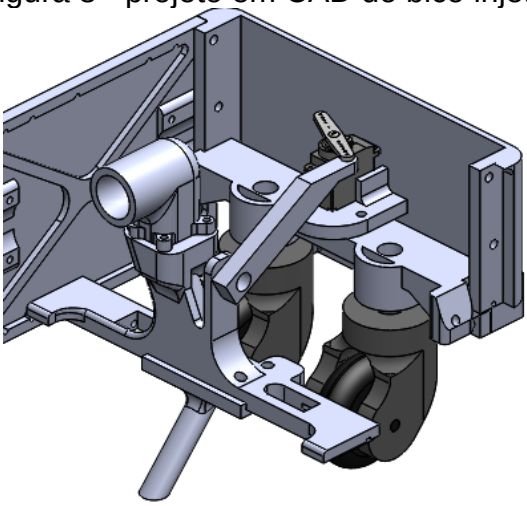
Fonte: autoria própria.

Logo abaixo da tampa foi projetado um suporte para abrigar um servo motor. Sua função é retrair a estrutura do bico enquanto este não estiver em uso e abaixa-lo no momento em que o robô estiver pronto para rejuntar. Mais à frente das rodas frontais é disposto um sensor que identifica a lacuna entre cerâmicas.

3.2.4 Bico de injeção

Para transportar a massa preparada da bisnaga em direção a área de rejuntamento, conforme o requisito R004, foi projetado em CAD um bico injetor modular, composto por três partes interligadas por parafusos de 4mm. Essa configuração auxilia na desmontagem e limpeza após o uso. A entrada do bico foi projetada utilizando-se loft de corte, no intuito de permitir o encaixe de tampas rosqueáveis, as mesmas utilizadas em bisnagas de creme dental. A Figura 8 mostra o projeto em CAD do bico injetor.

Figura 8 - projeto em CAD do bico injetor



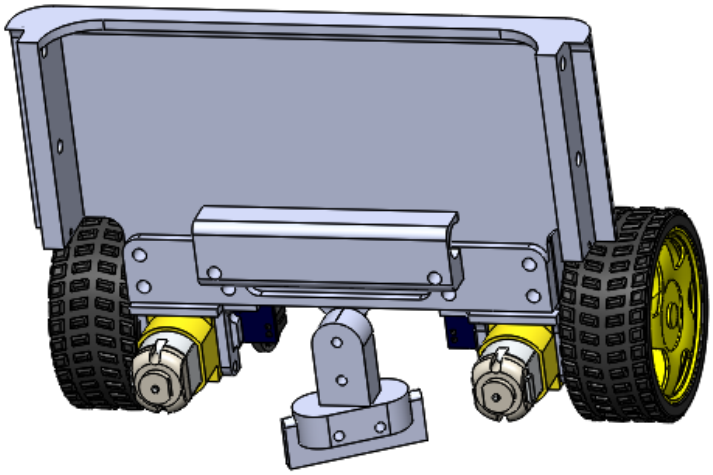
Fonte: autoria própria.

A tubulação possui furo interno de 10mm para a passagem do material. A seção de saída da tubulação apresenta uma curvatura com ângulo de 40º em relação ao eixo vertical, projetada para minimizar os esforços na junta do bico na direção de aplicação do rejunte. Uma haste conectada a um servo motor controla o movimento vertical do bico, permitindo sua retração no momento em que não está em uso e o abaixando no instante adequado para a aplicação da massa.

3.2.5 Rodo assentador

Tendo em vista otimizar a adesão e o preenchimento das lacunas a partir da massa despejada pelo bico de injeção, foi projetado em software CAD um rodo com curvatura de 45º (Figura 9), posicionado na parte inferior do robô, entre as rodas de tração. Sua geometria permite redirecionar os resíduos de massa que não foram assentados para apenas uma direção, na intenção de auxiliar a limpeza subsequente à aplicação de rejunte.

Figura 9 - projeto em CAD do rodo assentador



Fonte: autoria própria.

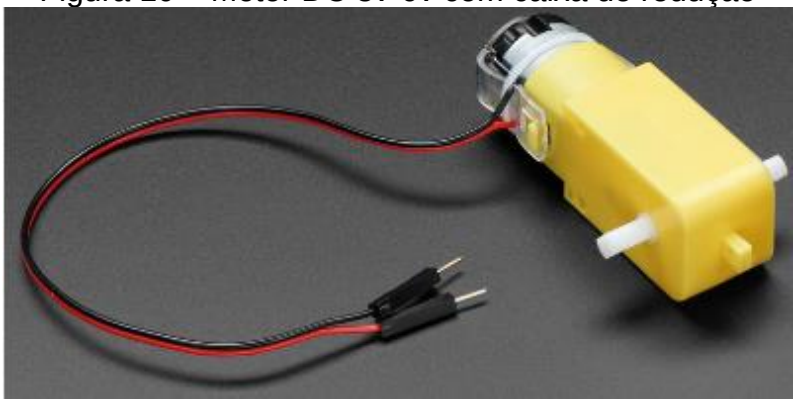
3.3 PROJETO ELETRÔNICO

O projeto eletrônico do robô aplicador de rejunte abrange a integração dos componentes eletrônicos responsáveis pelo controle, acionamento e sensoriamento do robô. Tendo em vista alcançar a performance adequada do robô em linha reta e a aplicação adequada da massa de rejunte, serão utilizados um microcontrolador ESP32, sensores para identificação do vão entre as cerâmicas além de sensores de velocidade, módulos para acionamento de motores e módulos necessários para conversão de corrente e tensão. Nesta seção, serão detalhados os componentes eletrônicos selecionados, suas funções, especificações técnicas e a forma como se integram para imprimir o movimento controlado e autônomo do robô. Também será abordada nessa seção o projeto da placa impressa em PCB.

3.3.1 Motores

Para o propósito do no robô aplicador de rejunte, motores CC de imãs permanentes com caixa de redução integrada, como ilustrado na Figura 10, foram escolhidos para a tração das rodas traseiras. A escolha por esse tipo de motor fundamenta-se em sua simplicidade de acionamento, reduzindo a quantidade de portas necessárias do microcontrolador se comparado com um motor de passo, além da capacidade de controle em malha fechada de velocidade, quando utilizado em conjunto com sensor encoder. Tendo em vista que o robô será utilizado de forma pontual, problemas com o desgaste relacionado a comutação das escovas serão mitigados.

Figura 10 – Motor DC 3v-6v com caixa de redução



Fonte: Adafruit (2018).

Especificações Técnicas do Motor DC 3v-6v com caixa de redução conforme site do fornecedor:

- Tensão Nominal: 3V ~ 6V
- Corrente Contínua sem Carga: 150mA +/- 10%
- Velocidade Mínima de Operação (3V): 90 +/- 10% RPM
- Velocidade Mínima de Operação (6V): 200 +/- 10% RPM
- Faixa de Torque: 0.15Nm ~ 0.60Nm
- Torque de Parada (6V): 0.8kg.cm
- Relação de Engrenagens: 1:48
- Dimensões do Corpo: 70 x 22 x 18mm
- Comprimento dos Fios: 200mm (bitola 28 AWG)
- Peso: 30.6g

Para o acionamento do rolo compressor, optou-se pela utilização de um motor de passo, visto que em condições normais de operação, esse motor permite o controle preciso da velocidade de extrusão. Na Figura 11 é exibido o modelo de motor utilizado:

Figura 11 - Motor de passo 28BYJ-48

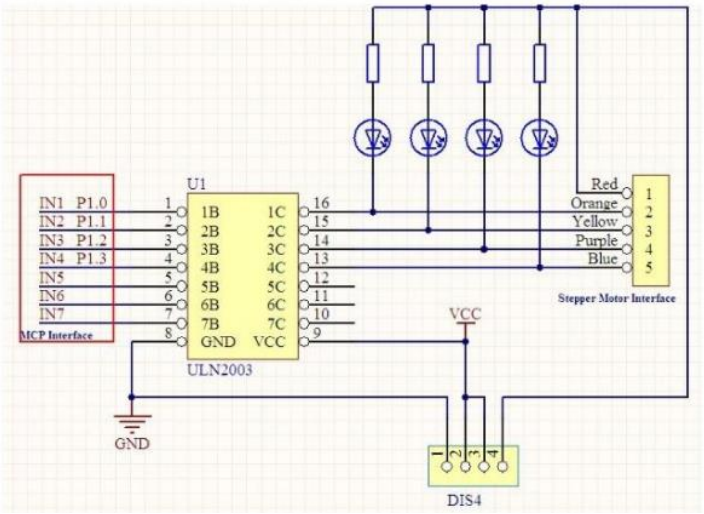
Fonte: autoria própria.

Especificações Técnicas do Motor de passo 28BYJ-48 conforme site do fornecedor:

- Tensão nominal: 5V DC.
- Número de fases: 4.
- Relação de redução: 1/64.
- Ângulo de passo: 5.625°/64. Considerando a relação de redução, o ângulo de passo real é de 0.087890625°.
- Frequência: 100Hz.
- Resistência CC: 50Ω ±7% (25°C).
- Torque: 34.3 mN.m

Motores de passo dependem de drivers de acionamento para interface com microcontroladores. No contexto do robô aplicador de rejunte, o driver ULN2003 (Figura 12) foi selecionado para acionar o motor de passo. Esse driver possui 4 canais de entrada e opera com tensões entre 5v a 12v.

Figura 12 - Driver ULN2003



Fonte: Vellerman (2018).

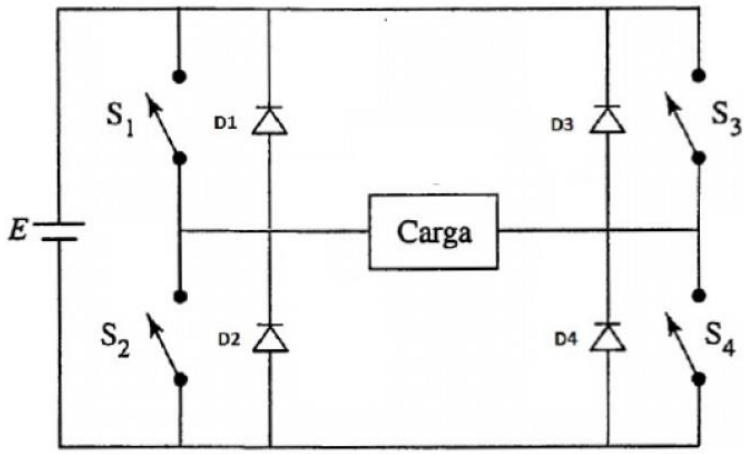
No projeto atual, o servo motor MG90s é utilizado na finalidade de controlar a posição do bico aplicador de massa (Figura 8), levantado a estrutura quando o robô não está aplicando o rejunte e abaixado quando o robô está em operação. Especificações técnicas do servo motor utilizado, conforme o datasheet:

- Peso: 13.4g
- Resistência CC: 50Ω ±7% (25°C).
- Tensão de alimentação: 4.8V até 6V
- Torque de parada: 0.1765 Nm quando alimentado com 4.8V ou 0.2157 quando alimentado com 6V.
- Velocidade de operação: 0.1s / 60º em 4.8V ou 0.08s / 60º quando alimentado com 6V

3.3.2 Módulo ponte H

Para o acionamento controlado dos motores de corrente continua utilizados na tração do veículo, optou-se por um modulo ponte H. Esse modulo é um circuito que, através de chaves, (transistores ou mosfets) aciona um motor de corrente continua através do recebimento de um sinal de modulação por largura de pulso (PWM) de entrada com baixa corrente. Na Figura 13 é demonstrado o circuito elétrico de uma ponte H genérica.

Figura 13 - Inversão de fonte de tensão em ponte completa

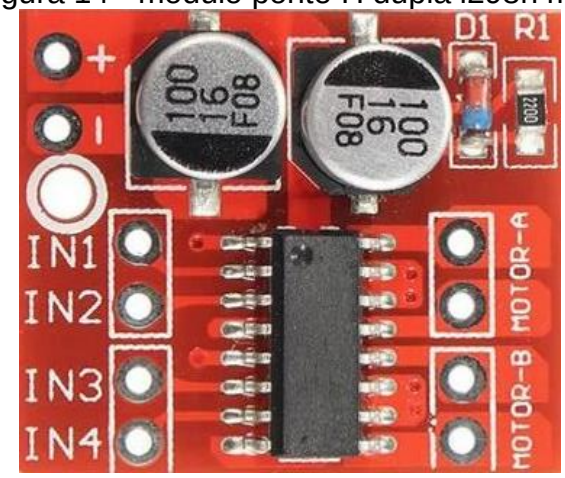


Fonte: Silva et al. (2019).

O acionamento do motor ocorre pela ativação simultânea das chaves S1 e S4 para um sentido, ou S2 e S3 para o outro sentido. A frequência do acionamento dessas chaves define a taxa de comutação entre os estados de condução e bloqueio das chaves. O ciclo de trabalho (duty cycle) do sinal PWM, que representa a razão entre o tempo de condução e o período total do sinal, determina a tensão média aplicada ao motor. Considerando a relação linear entre a tensão aplicada e a velocidade do motor, a modulação do ciclo de trabalho permite o controle preciso da velocidade de rotação (Gonzaga, 2018).

Dessa forma, optou-se pelo modulo ponte H l298n mini Figura 14, devido sua compatibilidade com os motores CC em uso no projeto e seu tamanho reduzido.

Figura 14 - modulo ponte H dupla l298n mini



Fonte: autoria própria.

3.3.3 Regulador de tensão

O regulador de tensão LM2596 é um circuito integrado monolítico que utiliza a topologia step-down (Buck) para conversão CC-CC, que é capaz de alimentar cargas com até 3A de corrente. Disponíveis em versões de saída fixas e ajustáveis, esses dispositivos requerem um número mínimo de componentes externos. Internamente, o regulador possui um oscilador de frequência fixa (150 kHz) que gera um sinal PWM para controlar o chaveamento de potência. O ciclo de trabalho é dinamicamente ajustado por um controlador, que visa manter a tensão de média no valor desejado. (Texas Instrument, 2023).

Para garantir a estabilidade da tensão de alimentação dos componentes eletrônicos utilizados no robô aplicador de rejunte, um regulador de tensão LM2596 é empregado como conversor abaixador tipo Buck. Esse dispositivo reduz a tensão do modulo de baterias, composto por 4 células de íon de lítio modelo 18650 (4.2V cada) para 5V, tensão estável e compatível com os componentes eletrônicos do robô. Através do controle de tensão, o regulador garante que os componentes eletrônicos recebam a tensão correta, mesmo com as variações de tensão inerentes à descarga da bateria.

3.3.4 Encoder incremental

Encoders incrementais são transdutores eletromecânicos que convertem movimento rotativo em sinais digitais pulsados (Carmo, 2015). Neste projeto, foram empregados para a aquisição de dados de deslocamento e velocidade linear do robô aplicador de rejunte. Discos circulares com 20 furos igualmente espaçados foram acoplados aos eixos de cada roda de tração. Utilizando-se módulos de encoder óptico, compostos por emissores e receptores de luz infravermelha, é possível detectar a movimentação do disco devido a passagem da luz infravermelha através dos furos nos discos. Na Figura 15 está disponível a montagem do conjunto modulo óptico e disco incremental.

Figura 15 - Encoder incremental

Fonte: autoria própria.

A cada interrupção do feixe de luz infravermelha, um pulso elétrico é gerado, sendo contabilizado por uma variável de controle. A frequência dos pulsos é diretamente proporcional à velocidade angular da roda, e sabendo o raio da roda, a determinação da velocidade linear e da distância percorrida pode ser efetuada através de cálculos cinemáticos. Conforme Halliday (2012), a velocidade linear (v) de um ponto presente num corpo em rotação é diretamente proporcional à sua velocidade angular (ω) e ao raio (r) da trajetória circular descrita por esse ponto. Essa relação é expressa pela equação 2:

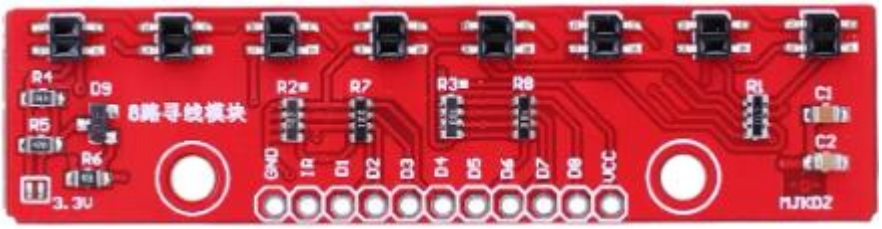
$$v = \frac{d\theta}{dt} \cdot r = \omega \cdot r \tag{2}$$

Sendo v a velocidade linear em metro por segundo, r o raio entre o ponto e o eixo de rotação, em metros e ω a velocidade angular em radiano por segundo, obtida pela leitura do encoder.

3.3.5 Sensores de refletância

Para detectar a região de rejuntamento foi utilizado um módulo composto de oito sensores ópticos de refletância QRE1113. O módulo Figura 16, consiste em um conjunto de sensores com emissores infravermelhos e receptores fototransistores que medem a intensidade de luz refletida pela superfície (FAIRCHILD, 2009). Com base na intensidade de luz infravermelha recebida por cada sensor, é possível identificar a posição do robô relativa à lacuna entre as cerâmicas, e assim controlar os motores de tração a fim de corrigir a trajetória do robô.

Figura 16 - Sensor QRE1333



Fonte: autoria própria.

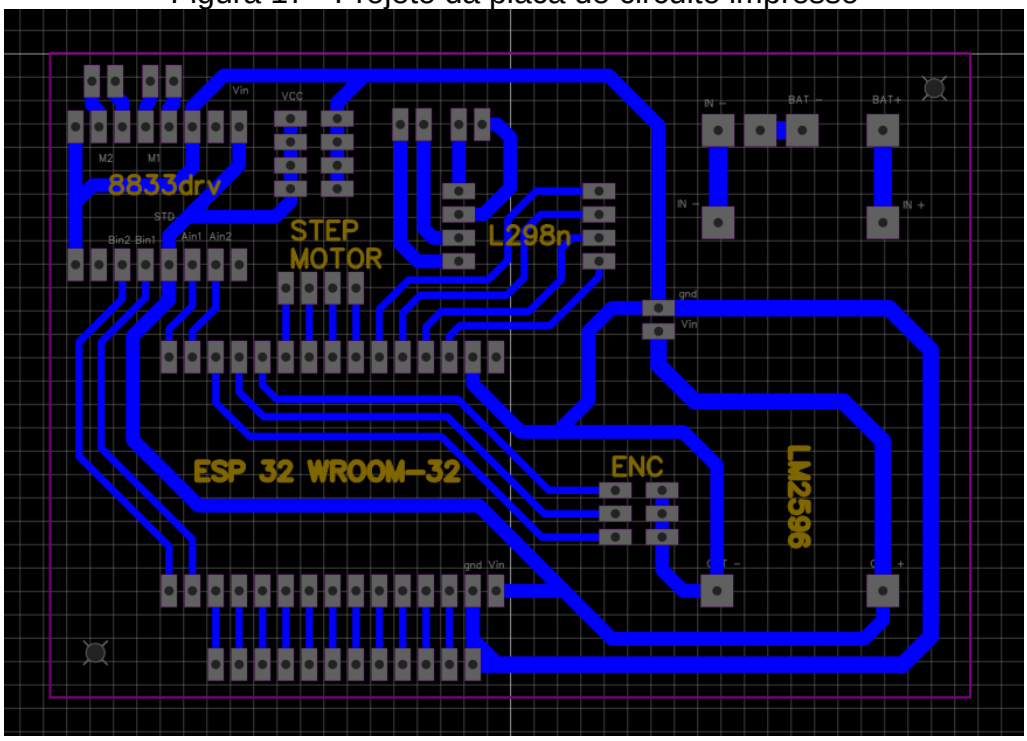
3.3.6 Microcontrolador ESP32-WROOM-32

O ESP32-WROOM-32 é um módulo microcontrolador de alto desempenho de arquitetura Xtensa dual-core de 32 bits, com frequência de clock ajustável entre 80MHz e 240 MHz. Integra conectividade Wi-Fi e bluetooth, além de uma ampla gama de periféricos, como GPIOs, conversores analógico-digital (ADC), temporizadores e interfaces seriais. (Expressif Systems, 2023). Sua função no robô aplicador de rejunte é a coordenação de todos os subsistemas do robô, incluindo a aquisição de dados dos sensores, filtragem dos sinais, processamento e controle dos módulos e drivers responsáveis por imprimir a movimentação autônoma do veículo.

3.3.7 Placa de circuito impresso

A placa de circuito impresso (PCB) do robô aplicador de rejunte (Figura 17), fabricada em fenolite, foi projetada na plataforma EasyEDA para acomodar e interconectar todos os componentes eletrônicos do sistema.

Figura 17 - Projeto da placa de circuito impresso



Fonte: autoria própria.

O layout da PCB inclui regiões dedicadas para a soldagem do módulo regulador de tensão, dos terminais para encaixe do microcontrolador, módulos de ponte H para acionamento dos motores de tração e conectores para os sensores de refletância, encoder e saída para o servo motor. Além disso, foram atribuídas quatro saídas do microcontrolador para habilitar o driver do motor de passo.

3.3.8 Sensor LiDAR

No objetivo de se obter as distâncias relacionadas ao ambiente em que o robô opera, foi efetuada a aquisição de um sensor Light Detection and Ranging (LiDAR) para medição de superfícies. Esse tipo de sensor tem por objetivo efetuar medições de distância pelo uso de um feixe a laser, disparado através de um espelho rotativo. Pela sincronia dos disparos a laser e a velocidade de rotação do motor uma varredura do ambiente. (Weber, 2018). O sensor obtido (Figura 18) infelizmente não possui sua documentação disponível, dificultando sua usabilidade, e diante das dificuldades em se implementar métodos para leituras confiáveis e até mesmo a conexão elétrica do dispositivo, decidiu-se por não o aplicar no robô.

Figura 18 - Sensor LiDAR

Fonte: autoria própria.

3.4 PROJETO DE SOFTWARE EMBARCADO

O projeto de software embarcado (Apêndice A) abrange a coordenação das principais funcionalidades do robô. O software foi desenvolvido em linguagem C++ utilizando-se a plataforma Platform.io, juntamente com diversas bibliotecas que facilitam o gerenciamento de cada subsistemas do robô. Nessa seção, serão abordadas as classes desenvolvidas para o tratamento dos sensores de refletância, encoder, motor de passo e motor CC.

Será explicado também a lógica por trás da comunicação Wi-Fi desenvolvida, o envio de código através de OTA para upload de código sem a utilização de cabo de dados e a página web desenvolvida em HTML, com funcionalidades de iniciar e parar o robô, além de efetuar medições de variáveis. No apêndice A está disponível o diagrama de classes, contendo todas as classes e suas associações.

3.4.1 Classe ReflectanceSensor

Essa classe foi construída no intuito de efetuar o processamento de dados lidos pelos sensores de refletância. Ela utiliza filtros de Kalman para reduzir o ruído da leitura efetuada por cada sensor, com o objetivo de se obter leituras mais precisas. Em seu construtor são atribuídos filtros de Kalman para um vetor de quatro posições, os quais serão utilizados por cada sensor.

No método init (Apêndice E), quatro portas com propósito geral de entrada e saída (GPIOs) são atribuídas para a leitura dos sensores. O comportamento principal da classe ocorre no método Readline, aonde ocorre cada leitura dos sensores passa por seu respectivo filtro Kalman. A função Readline é chamada no programa principal (Apêndice B) a cada período de amostragem e a função init, apenas no início do programa.

3.4.2 Classe Encoder

A classe encoder (Apêndice F), foi construída na finalidade de se obter a distância percorrida por cada roda de tração, assim como suas velocidades individuais. Para se obter a leitura de pulsos lidos pelo encoder de maneira precisa, técnicas de debouncing e histerese foram implementadas nessa classe, minimizando a quantidade de leituras errôneas. O debouncing introduz um atraso na leitura dos sinais, descartando transições rápidas e permitindo apenas mudanças estáveis entre os estados. A histerese define uma faixa de valores no limiar de transição entre os estados, evitando que pequenas flutuações na leitura sejam consideradas transições válidas.

No construtor da classe são inicializados dois filtros Kalman, um para cada sensor encoder, com o objetivo de filtrar ruídos para as medições de velocidade. O método init() efetua a configuração inicial da classe, atribuindo as portas GPIO que serão utilizadas para as leituras do encoder, define o intervalo de histerese e o tempo necessário para debouncing, além de variáveis necessárias para obtenção das velocidades. O método update concentra a funcionalidade central da classe, aonde são efetuadas as contagens de pulsos obtidos pelos sensores, assim como os cálculos de velocidade a cada ciclo de amostragem efetuado no programa principal. As velocidades obtidas são convertidas para cm/s a fim de se adequar a escala compatível com o projeto.

3.4.3 Classe Motor

A classe Motor é responsável pelo acionamento dos motores CC do robô, além de alocar a implementação de um controlador PID individual para o controle de velocidade de cada roda. Em seu construtor, quatro portas GPIO do microcontrolador

são definidas como saída para o controle da ponte H através de PWM. O método init efetua a inicialização das variáveis do controlador PID, incluindo os ganhos proporcional, integral e derivativo. Para o acionamento do motor, dois métodos foram construídos.

O método updatePWM comanda cada motor conectado a ponte H de maneira independente e através de valores de PWM (Apêndice C). Recebendo como entrada duas variáveis que armazenam valores inteiros entre 0 a 255, o método atribui diretamente na saída digital da ponte H os valores recebidos, e assim acionando os motores com tensão entre 0V a 5V de acordo com o valor de PWM recebido. Contudo, devido a não idealidade dos componentes, esse método apresenta limitações no controle preciso da velocidade. Por questões de construção do modulo ponte H e do motor, esses componentes podem apresentar um comportamento distinto para a mesma entrada de PWM, dificultando a obtenção de velocidades idênticas nos dois motores.

Para solucionar isso, foi gerado o método updatePID, que embora possua maior complexidade de implementação, oferece a vantagem de controle por retroalimentação. Esse método recebe como entrada os valores de velocidade de cada encoder juntamente com os valores pretendidos de velocidade. A partir desses valores de entrada, são gerados dois erros, um para cada controlador, representando a diferença entre a velocidade desejada e a velocidade real, lida através dos sensores encoder. As constantes de controlador proporcional, derivativo e integrativo, inicializadas no método init, são aplicadas nos controladores de cada roda a fim de se reduzir o erro obtido dos valores de velocidade.

Para efetuar o controle, um temporizador não bloqueante utilizando a função millis(), pertencente à biblioteca arduino, foi implementado para funcionar em um período fixo de 1ms. A ação proporcional é calculada levando em consideração o erro obtido entre as velocidades real e desejada, multiplicado pela constante proporcional kp. Essa ação corresponde diretamente a magnitude do erro, aumentando a ação do controlador conforme o erro aumente. A ação integral é obtida através da aproximação trapezoidal da soma dos erros passados. Conforme Lamas *et al* (2010), pode-se obter a área de um trapézio através da soma das bases do trapézio multiplicada pela altura do trapézio e dividindo por 2, conforme demonstrado na Equação 3.

$$A = \frac{(B + b) \times h}{2} \quad (3)$$

O ganho derivativo é obtido através da aproximação de primeira ordem para a derivada utilizando o método das diferenças finitas para trás. De acordo com Lage (1997) e Fontana (2019), o método das diferenças finitas transforma uma função continua no tempo em uma representação discreta, e após esse procedimento permite gerar aproximações para as derivadas nos pontos discretos através de expansão de uma função em série de Taylor. A derivada para trás, Equação 4, estima o valor da derivada avaliando o termo atual juntamente com o anterior. Considerando que o espaçamento entre os pontos é constante, e desprezando na série de Taylor os termos de ordem maior ou igual a 2, se obtém a expressão:

$$\left.\frac{dy}{dx}\right|_{x=x_i} \approx \frac{y_i - y_{i-1}}{\Delta x} \quad (4)$$

No sistema aplicado ao robô do rejunte, não é necessário se obter a função continua no tempo do erro, visto que se tem acesso a saída do erro em formato discretizado. Sabendo que o intervalo do temporizador é constante e que cada amostra é espaçada nesse mesmo intervalo T_S, a aproximação da derivada substituindo intervalo de tempo na Equação 5:

$$\left.\frac{dy}{dx}\right|_{x=x_i} \approx \frac{y_i - y_{i-1}}{T_s} \quad (5)$$

Com as formulas definidas, os valores das constantes proporcional, integrativo e derivativo foram multiplicados por cada ação estudada. De acordo com Tannuri (2010), um método frequentemente utilizado para a sintonização do controlador PID é através da tentativa e erro, obtendo em sequência os valores de ganho proporcional, integrativo e derivativo de acordo com a resposta do sistema. Dessa forma, os valores das constantes foram sintonizados manualmente, analisando o comportamento do robô de acordo com as constantes escolhidas.

3.4.4 Classe Stepper

A classe stepper (Apêndice D) foi desenvolvida nesse projeto com objetivo de acionar o motor que movimenta o rolo compressor, encapsulando as funcionalidades da biblioteca AcelStepper para motores de passo. quatro pinos GPIO do microcontrolador são definidos como saída e controlam o drive conectado ao motor de passo, sendo os pinos atribuídos no método init da classe. Também no método init são definidas a velocidade de 400 passos por segundo e aceleração de 300 passos por segundo ao quadrado. Sabendo que o motor efetua 2048 passos por volta, é possível se obter a relação em rad/s da velocidade conforme a equação 6:

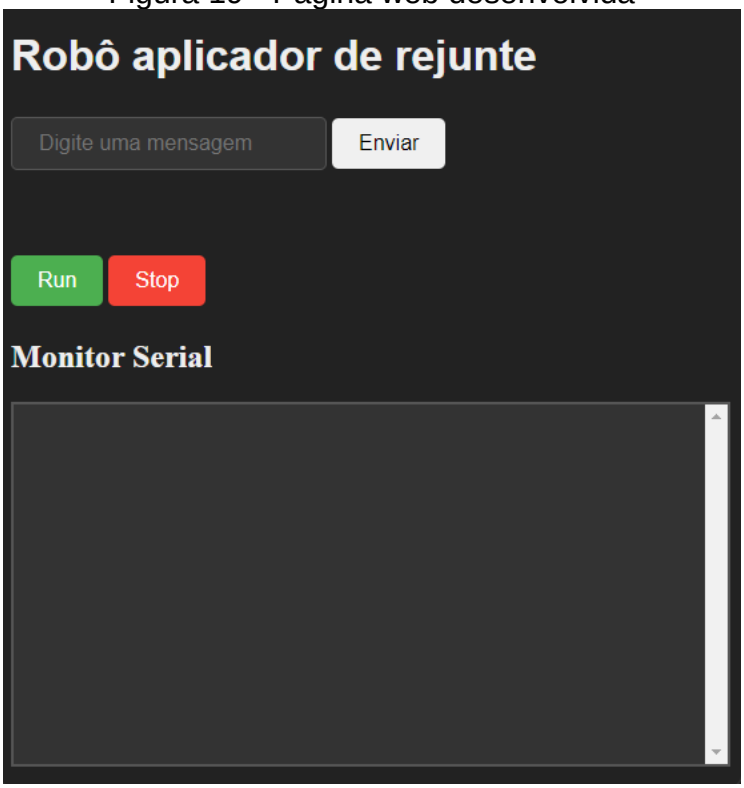
$$\omega = \frac{N\ passos/segundo}{2048\ passos/volta} \cdot 2\pi \quad (6)$$

3.4.5 Programa principal

O programa principal (Apêndice B) coordena a interação entre as classes desenvolvidas para o controle do robô durante a tarefa de rejuntamento. Nele, está encapsulada a lógica central do projeto de software, que consiste na definição dos pinos GPIO que serão gerenciados por cada classe, a configuração e interface web, o recebimento de atualizações do código via OTA, além da implementação do controlador para navegação autônoma em linha reta utilizando os dados dos sensores de refletância.

A interface web é gerenciada por meio da biblioteca AsyncWebServer, que permite a criação de um servidor web assíncrono no microcontrolador, estabelecendo assim a comunicação entre o robô e dispositivos que possuam interface web. Nesse sentido, uma página web foi elaborada em HTML, CSS e JavaScript, contando com entradas de mensagem de texto do usuário, botões para iniciar o robô e pará-lo em situações de emergência, além de uma região para depuração do código, Figura 19.

Figura 19 - Página web desenvolvida



Fonte: autoria própria.

A interação com a página ocorre através de requisições HTTP enviadas do navegador para o servidor embarcado no microcontrolador. O servidor processa as requisições POST direcionadas aos pontos “/Sendmessage” e /Sendcommand” para interpretar comandos start e stop do robô, que controlam a ativação e desativação do robô, respectivamente. Além disso, o servidor também disponibiliza o ponto “/serial”, que envia uma cadeia de caracteres, utilizada para a depuração de dados recebidos dos sensores.

Embora o servidor possua um dicionário de comando permitidos para mitigar a injeção de comandos arbitrários, a implementação da página web apresenta vulnerabilidades a ataques de injeção de código malicioso. Dessa forma, a implementação de soluções contra esse tipo de ataque deve ser considerada tendo em vista garantir a robustez da comunicação.

A tarefa de atualização OTA utiliza a biblioteca ArduinoOTA para permitir a atualização remota do software embarcado no microcontrolador. Inicialmente, o usuário conectado na mesma rede que o microcontrolador deve utilizar uma ferramenta compatível com OTA. Após identificar o IP do microcontrolador na rede, o usuário insere as credenciais de acesso, e então a atualização é transmitida para o

ESP32. Para o recebimento de atualizações ser possível a qualquer instante, a função de atualização OTA é chamada repetidamente no loop principal.

Assim como a atualização OTA, a tarefa de leitura do encoder é realizada no loop principal, com sua frequência de leitura gerenciada por um temporizador não bloqueante implementado na classe encoder. Por fim, a rotina de rejuntamento é inicializada por uma requisição POST contendo o comando start, que altera uma variável de controle da rotina e uma função denomina rejuntar que encapsula todas as seguintes tarefas. Dessa forma, a classe do sensor de refletância identifica a posição da lacuna, e quando essa se encontra centralizada o bico aplicador de rejunte é abaixado, o motor que aciona os rolos compressores é iniciado e o controlador recebe as velocidades de referência para acionar os motores.

A estratégia de controle para a navegação autônoma em linha reta consiste em se definir um controlador PID em cascata, sendo o controlador PID de posição da lacuna conectado em série com os controladores individuas de velocidade das rodas. Uma velocidade de referência é aplicada para ambos os controladores PID de velocidade, juntamente com um termo de erro somado para uma roda e subtraído para a outra roda. O termo de erro é composto pela soma de ganhos proporcionais, derivativos e integrativos implementados de forma análoga ao controlador de velocidades da classe motor, sendo aplicada a sintonização manual.

3.5. DEMONSTRAÇÃO DO ARTEFATO OBTIDO

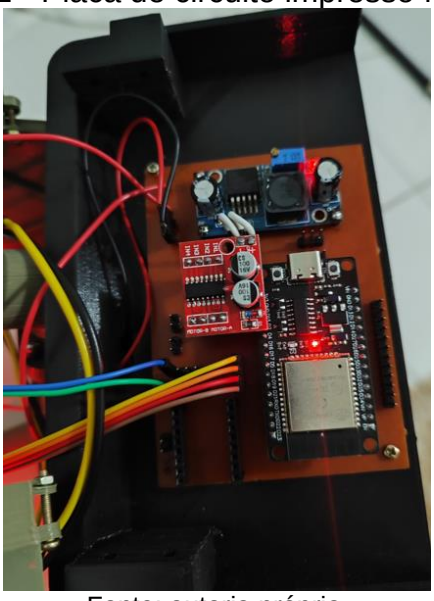
Com a finalização dos sistemas, foram utilizados parafusos de aço com diâmetro de 4mm para fixação dos principais componentes do robô, com exceção ao componente que fixa o sensor de refletância, o qual necessitou de parafusos com 2mm de diâmetro. As peças foram impressas em impressora 3D com configuração de preenchimento em 40% das partes solidas, utilizando-se de dois rolos de filamentos PLA e resultando em 1,9kg a massa total do robô, medida essa efetuada através de uma balança. Após impressas, as peças foram alocadas em suas respectivas posições funcionais conforme mostra a Figura 20.

Figura 20 - Projeto estrutural completo

Fonte: autoria própria.

A placa de circuito impresso (Figura 21) foi equipada com um regulador de tensão LM2596 e diversos conectores femêa para acomodar módulos, como a ponte H e o microcontrolador. Conectores macho foram adicionados para a recepção de sinais provenientes dos sensores de refletância, encoders e para o acionamento do motor de passo. Três pinos de tensão e seis pinos terra foram disponibilizados de forma redundante para permitir novas propostas e alterações pontuais.

Figura 21 - Placa de circuito impresso finalizada



Fonte: autoria própria.

4. RESULTADOS E DISCUSSÕES

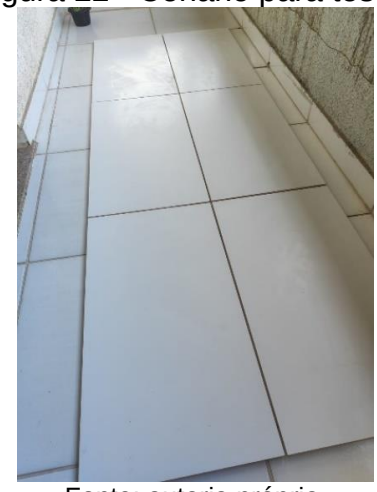
Esse capitulo tem por finalidade demonstrar a avaliação do artefato obtido, conforme padronizado na metodologia DRS. Serão detalhados os procedimentos de sintonização dos controladores PID, além da montagem do ambiente de testes. Serão discutidas também, as limitações observadas durante os testes e a análise das oportunidades de aprimoramento tendo em vista futuras iterações do projeto.

4.1. AVALIAÇÃO DO ARTEFATO OBTIDO

Após a integração das partes mecânicas, elétrica e de software, as leituras dos sensores foram validadas e os controladores foram sintonizados em testes práticos com ajuste manual. Inicialmente, os ganhos integrativos e derivativos foram zerados na intenção de se obter parâmetros satisfatórios de ganhos proporcionais. Em seguida, pequenos valores de ganho integrativo foram testados e para se reduzir o erro em regime permanente. Por fim, o ganho derivativo foi aplicado para aplicar maior velocidade de resposta do sistema.

Com os controladores de velocidade sintonizados, foi montado um cenário prático de aplicação de rejunte, no qual foram assentadas 6 cerâmicas de cor branca, com dimensões de 32cm x 59cm (Figura 22). Nesse local, os sensores de refletância foram calibrados, e o controlador de posição do robô foi sintonizado, utilizando a mesma abordagem de sintonia dos controladores de velocidade.

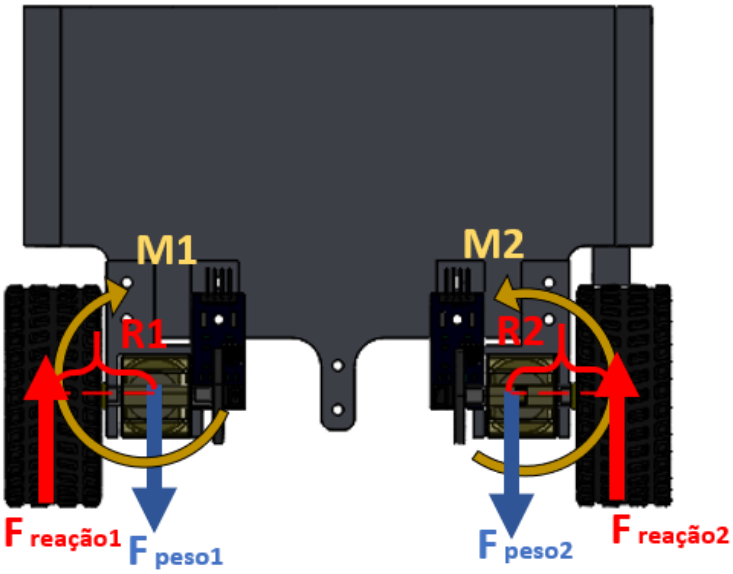
Figura 22 - Cenário para testes



Fonte: autoria própria.

No início dos testes, o veículo obteve bom desempenho em linha reta com ganhos predominantemente proporcionais e derivativos. No decorrer da utilização do robô, o veículo demonstrou comportamento inconsistente em termos de permanecer centralizado na lacuna de rejuntamento, alternando entre resultados satisfatórios e insatisfatórios. Após nova avaliação de todos os sistemas que poderiam apresentar esse tipo de falha, identificou-se um problema de projeto estrutural. Foi efetuada a análise cinemática (Figura 23) da seção traseira da estrutura, e nela foi revelado que as rodas de tração tendem a entrar em contato com a lateral do chassi.

Figura 23 - Análise das forças atuantes na seção traseira do veiculo



Fonte: autoria própria.

Em condições estáticas, a magnitude de cada força de reação é igual a magnitude de sua força peso correspondente. Devido as forças de reação e forças peso estarem deslocadas em um raio R, um momento angular é gerado e as rodas sofrem um movimento de alavanca. Isso resulta em situações de travamento que restringem o movimento das rodas, introduzindo perturbações no sistema de controle do robô.

Também durante os testes práticos, notou-se que o sistema projetado para a expulsão de massa não atingiu a eficiência desejada, produzindo torque insuficiente para espremer a massa a bisnaga por dentro do bico injetor. Dessa forma, novos testes foram efetuados com relação ao conjunto de rolos, e a partir desses testes, constatou-se que o sistema desenvolvido possui a capacidade de espremer a bisnaga apenas quando a mesma não está conectada no bico injetor. Para investigar essa

questão, percebeu-se que será necessária uma análise da perda de energia mecânica no processo de extrusão, levando em consideração a geometria de cada parte do bico injetor, assim como a forma que se é feita a conexão entre as partes.

De acordo com Bulu (2001), a perca de energia de um fluido viscoso pode ser efetuada atráves da equação de Hagen-Poiseuille, considerando a diferença de pressão entre as extremidades do tubo como a relação ao comprimento da tubulação, o diâmetro e a densidade do fluido. No entanto, a massa de rejunte é um fluido não newtoniano, ou seja, sua viscosidade não é constante. Dessa forma, deve ser considerada uma análise por meio de experimentação ou modelagem numérica para entender o comportamento do rejunte dentro da tubulação, e assim, projetar um novo bico injetor.

4. CONCLUSÃO

Em síntese, esse projeto abordou o desenvolvimento de um protótipo de robô autônomo aplicador de rejunte, com a finalidade de automatizar uma tarefa manual, repetitiva e ergonomicamente inadequada. A partir dos objetivos estabelecidos, foram implementados o projeto mecânico, eletroeletrônico e de software embarcado, utilizando como base a metodologia Design Science Research.

Os projetos eletroeletrônico e de software se mostraram promissores, apresentando desafios inerentes ao processo, contudo sem comprometer a funcionalidade do robô. A placa PCB acomodou os módulos e estabeleceu a interface entre os sensores e o microcontrolador de maneira eficaz, garantindo a aquisição e processamento de dados. O projeto de software embarcado, por sua vez, permitiu a interação do usuário com sistema do robô através de uma interface web, além de acomodar toda a lógica responsável por efetuar o rejuntamento de maneira autônoma, utilizando uma arquitetura de controle em cascata para regular a posição do robô em cima da lacuna e as velocidades de cada roda.

No entanto, o projeto mecânico revelou-se um ponto crítico que necessita de aprimoramentos. Problemas estruturais na posição das rodas de tração e no projeto do bico aplicador de rejunte comprometem a capacidade de locomoção controlada e a funcionalidade do sistema. A análise de perda de energia mecânica no processo de extrusão, considerando o comportamento não newtoniano da massa de rejunte é recomendado, tendo em vista a eficiência e funcionalidade adequada desse sistema. Após a integração dos sistemas, os resultados obtidos nos testes revelaram a necessidade de aprimoramento da parte estrutural do robô para o desempenho adequado na tarefa de aplicar rejunte.

REFERÊNCIAS

AHN, H et al. Applicability of smart construction technology: Prioritization and future research directions. **Automation in construction.** v. 153, artigo n. 104953, 2023.

ASSOCIAÇÃO BRASILEIRA DE NORMAS TÉCNICAS. NBR 13.753. **Revestimento de piso interno ou externo com placas cerâmicas e com utilização de argamassa colante** - Procedimento. Rio de Janeiro: ABNT, 1996.

BONAFÉ, G. **Saiba como aplicar rejunte em pisos e azulejos**. AECweb, 24 out. 2016. Disponível em: https://www.aecweb.com.br/revista/materias/saiba-como-aplicar-rejunte-em-pisos-e-azulejos/14571. Acesso em: 09 jun. 2023.

CARASEK, H. **Materiais de construção civil e princípios de ciência e engenharia de materiais.** 2. ed. São Paulo: IBRACON, 2010.

CARDOSO, M. **O que é um microcontrolador**? Robotics and Automation Society - RAS, 23 set. 2020. Disponível em: https://edu.ieee.org/br-ufcgras/o-que-e-um-microcontrolador/. Acesso em: 11 jun. 2023.

CUTIERU, A. **Automação no canteiro de obras**. A. ArchDaily. 03 jul. 2021. Disponível em: https://www.archdaily.com.br/br/963393/automacao-no-canteiro-de-obras. Acesso em: 04 jun. 2023.

DORF, R. C.; BISHOP, R. H. **Sistemas de controle modernos.** 8. ed. Rio de Janeiro: LTC, 1998.

HWANG, D.; KHOSHNEVIS, B. An innovative construction process-contour crafting. *In:* **Proceedings** of the 22nd International Symposium on Automation and Robotics in Construction - ISARC, 11-14 sep. 2005, Ferrara, IT. 2005. p. 01–02. Disponivel em: https://www.iaarc.org/publications/fulltext/isarc2005-03hwang.pdf. Acesso em: 16 jun. 2023

JONES, J. L.; FLYNN, A. M.; SEIGER, B. A. **Mobile robots inspiration to implementation**. 2. ed. Natic, MA: CRC Press, 1998.

MONTEIRO, C**.** Industria 4.0: A tecnologia e as revoluções nas nossas vidas e negócio. **Associação dos Arquitetos, Engenheiros e Eletrotécnicos de Cotia - AETEC**, ed. 33, ano 6, 2006.

NISE, N. S. **Engenharia de sistemas de controle.** 6. ed. Rio de Janeiro: LTC, 2013.

OGATA, K. **Engenharia de controle moderno.** 5. ed. São Paulo: Pearson Prentice Hall, 2010.

CHOQUEHUANCA, C. R. M. **Projeto e controle robusto de um transportador pessoal robótico autoequilibrante.** 2010. Dissertação (Mestrado em Engenharia Mecânica) - Pontifícia Universidade Católica do Rio De Janeiro, Rio de Janeiro, 2010.

ROCHA, B. M. **Modelos e caracteristicas de casas pré-fabricadas no Brasil.** 2020. Trabalho de Conclusão de Curso (Graduação em Engenharia Civil) - Faculdade Presidente Antônio Carlos, Teófilo Otoni, 2020.

ROGGIA, L.; FUENTES, R, C. **Automação industrial**. Colégio Técnico Industrial da Universidade Federal de Santa Maria, Santa Maria, 2016. p. 20-21.

SANTOS, B. S. **Concepção, projeto e construção de um robô autônomo.** 2002. Dissertação (Mestrado em Engenharia Elétrica) - Escola Federal de Engenharia de Itajubá, Itajubá, 2002.

TUDO CONSTRUÇÃO. Como aplicar rejunte? passo a passo. **Tudo Construção**, 03 jan. 2019. Disponivel em: https://www.tudoconstrucao.com/como-aplicar-rejunte-passo-a-passo. Acesso em: 14 jun. 2023.

CARDOSO, M. **O que é um microcontrolador**? Robotics and Automation Society - RAS, 23 set. 2020. Disponível em: https://edu.ieee.org/br-ufcgras/o-que-e-um-microcontrolador/. Acesso em: 11 jun. 2023.

WENDLING, M. **Sensores V2.0.** Universidade Estadual Paulista - UNESP. 15 maio 2010. Disponível em: https://www.feg.unesp.br/Home/PaginasPessoais/ProfMarceloWendling/4---sensores-v2.0.pdf. Acesso em: 11 jun. 2023.

JAFARY et al. **A survey on autonomous vehicles interactions with human and other vehicles**. Conference Paper, 14 set. 2018. Disponível em: https://www.researchgate.net/publication/329233150_A_Survey_on_Autonomous_Vehicles_Interactions_with_Human_and_other_Vehicles. Acesso em 27 jun. 2024.

LIU, S. et al. Edge computing for autonomous driving: opportunities and challenges. em: **PROCEEDINGS OF THE IEEE**, v. 107, n. 8, p. 1697-1716, ago. 2019. Disponível em: https://ieeexplore.ieee.org/abstract/document/8744265/authors#authors. Acesso em: 26 jun.2024.

MCGLYNN, W. J.; WALTERS, M. L. **agricultural robots: future trends for autonomous farming**. Rochester Institute of Technology, New York. abr. 2019.

ANGELUCI, A. C. B. et al. Design science research como método para pesquisas em TIC na educação. **Anais do CIET:EnPED:2020 - (Congresso Internacional de Educação e Tecnologias | Encontro de Pesquisadores em Educação a Distância)**, São Carlos, ISSN 2316-8722, ago. 2020. Disponível em: https://cietenped.ufscar.br/submissao/index.php/2020/article/view/1023. Acesso em 26 jun. 2024.

HEVNER, A. R. et al. **Design science in information systems research.** MIS Quarterly, v. 28, n. 1, p. 75-105, mar. 2004. Disponível em: https://doi.org/10.2307/25148625. Acesso em 25 jun. 2024.

FERREIRA, R. **Entendendo a extrusão de polímeros**. Módulo III. Instituto Federal de Educação, Ciência e Tecnologia Sul Rio-Grandense. 14 jan. 2019.

QUELHO, P. E. **Desenvolvimento de extrusora experimental e software para controle e supervisão das variáveis de extrusão do ABS**. 2018. Dissertação (Mestrado em Engenharia Mecânica) - Universidade Tecnológica Federal do Paraná, Ponta Grossa, 2018.

HALLIDAY, D.; WALKER, J.: RESNICK, R. **Fundamentos da física**: mecânica. v. 1, p. 286-287. ed. Rio de Janeiro: LTC, 2012.

FITZGERALD, A. E.; JR C. K.;UMANS S. D. **Eletric Machinery.** 7. ed. The McGraw-Hill Global Education Holdings, LLC, New York, 2014.

CAMARGO, I. M. **Máquinas de corrente contínua**. Revisão 1. 2007.

SOUZA, D. H. C. **Robotic arm using step motors**: Studies for Optimal Operation and Load Mass Estimation. 2021. Dissertação (Mestrado em Engenharia Mecânica) - Universidade Tecnológica Federal do Paraná, Curitiba, 2021.

GONZAGA, J. A. **Estudo de técnica de modulação por largura de pulso (PWM) aplicado a inversores trifásicos**. 2018. Trabalho de Conclusão de Curso (Graduação em Engenharia Elétrica) - Universidade Tecnológica Federal do Paraná, Curitiba, 2018.

TEXAS INSTRUMENT. **LM2596 simple switcher**, power converter 150-kHz 3-A Step-Down Voltage Regulator. 2023.

CARMO, R. A**. Circuito de monitoramento da velocidade de um motor usando encoder incremental**. 2015. Trabalho de Conclusão de Curso (Graduação em Engenharia de Controle e Automação) - Universidade Tecnológica Federal do Paraná, Curitiba, 2015.

FAIRCHILD. **QRE1113, QRE1113GR**, Miniature reflective object sensor. Datasheet Rev. 1.6.0, set. 2009. Disponível em: https://www.sparkfun.com/datasheets/Robotics/QR_QRE1113.GR.pdf. Acesso em: 12 abr. 2024.

ESPRESSIF SYSTEMS. ESP32-WROOM-32 Datasheet. Version 3.4, 2023.

WEBER, T. P. **Modo de funcionamento e variantes dos sensores lidar**. 2018. Trabalho de Conclusão de Curso (Graduação em Engenharia Cartográfica e de Agrimensura) - Universidade Federal do Paraná, Curitiba, 2018.

LAMAS, W. T. et al. **Ensinando área no ensino fundamental**. 2010. Disponível em: https://www.ibilce.unesp.br/Home/Departamentos/Matematica/ensinandoarea.pdf. Acesso em: 25 jun. 2024.

LAGE, P. L. C. **Modelagem matemática de sistemas de engenharia química**. 1997. Escola piloto de engenharia química. COPPE Universidade federal do rio de janeiro – UFRJ. p.63-68. mar, 1997.

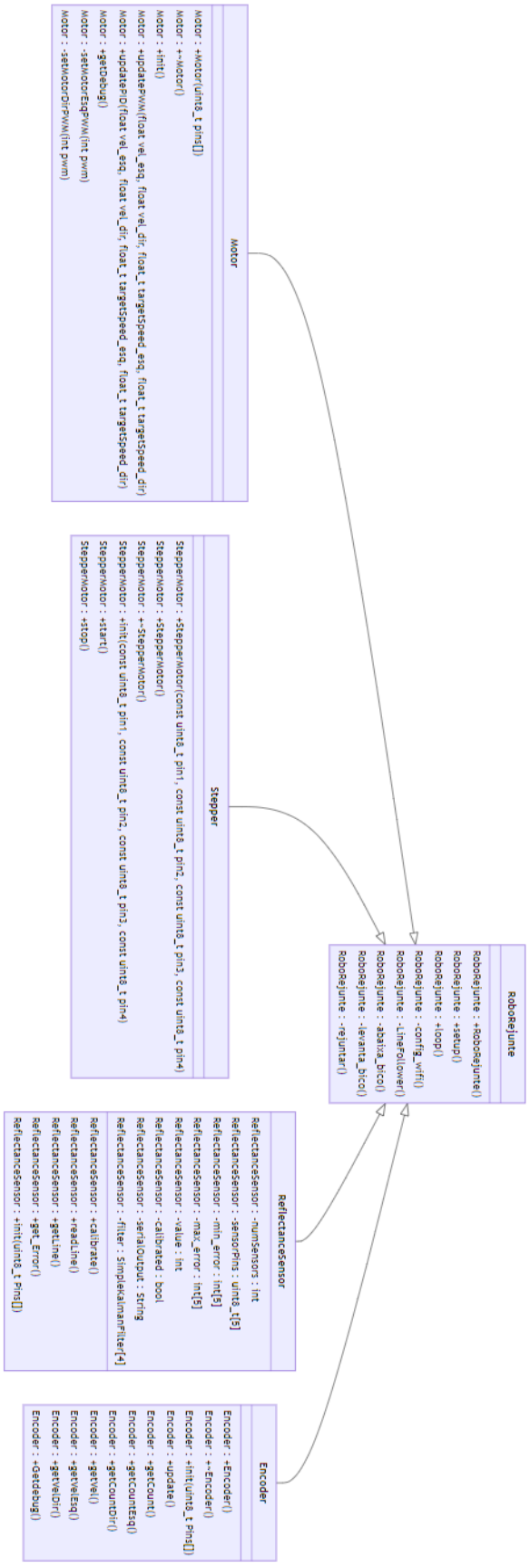
FONTANA, M. E. **Introdução ao método de diferenças finitas com aplicações em engenharia química**. 2019. Dissertação (Mestrado em Engenharia Química) - Universidade Federal do Paraná, Curitiba, 2019.

TANNURI, E. A. **Apostila sobre PID e métodos de sintonia**. Departamento de engenharia mecatrônica e sistemas mecânicos, escola politécnica da USP. nov. 2010.

BULU, A., **Lecture notes – VII: fluid mechanics**. ITU, Istambul, Turkey, 2001. Disponivel em: https://web.itu.edu.tr/~bulu/fluid_mechanics_files/lecture_notes_07.pdf . Acesso em: 26 jun. 2024.

ANODILAR, **Cilindro de massa sova master,** disponível em: https://www.andoficial.com.br/cilindro-de-massas-sova-master-cromado-28cm-sem-definicao. Acesso em: 25 jun. 2024.

APÊNDICE A: Diagrama de classes



Fonte: Autoria própria.

APÊNDICE B - Main.cpp

```
#include <Arduino.h>
#include <WiFi.h>
#include <ESPmDNS.h>
#include <WiFiUdp.h>
#include <ArduinoOTA.h>
#include <ESPAsyncWebServer.h>
#include <SPIFFS.h>
#include <AccelStepper.h>
#include <vector>
#include "pinos.hpp"
#include "../lib/stepper_motor.h"
#include "../lib/reflectance_sensor.h"
#include "../lib/encoder.h"
#include "../lib/motor.h"
#include <string.h>

StepperMotor *stepper_motor = new StepperMotor();
ReflectanceSensor *reflectance_sensor = new ReflectanceSensor();
Encoder *encoder = new Encoder();
uint8_t MOTOR_PINS[] = {MOTOR_PIN_4, MOTOR_PIN_3, MOTOR_PIN_2, MOTOR_PIN_1};
Motor *motor = new Motor(MOTOR_PINS);

void config_wifi();
void LineFollower();
void abaixa_bico();
void levanta_bico();
void rejuntar();


        // 1 revolution = 20 steps = 2*pi*69/2 = 216.769 mm
        // 20 steps = 216.769 mm
        // x steps = y mm
        // portanto, x = y*20/216.769
int mm_to_steps(int mm)
{
    return int(mm*20/216.769);
}
float steps_to_mm(float steps)
{
    return steps*216.769/20;
}
int targetPosition = 40000;


bool motorState = LOW;
bool encrusilhada = false;
const char* ssid = "Valdomiro-not";
```

```
const char* password = "quesenha";
unsigned long currentMillis2 = 0;
unsigned long previousMillis = 0;
unsigned long previousMillis2 = 0;


AsyncWebServer server(80);
bool start_robot = false;
String serialOutput = "";
int servo_ = 0;
bool bico_abaixado = true; // retorna ON quando abaixado e OFF quando
levantado

unsigned long lastcurrentMillis = 0;
bool inverte = false;
//line follower vafiables
uint8_t Ts = 10;
float setpoint = 0.0;

// PID constants
float Kp = 0.0;
float Ki = 0.0;
float Kd = 0.0;
float error = 0;
float baseSpeed = 25.0; //cm/s

float max_limit = 4050;
float min_limit = 3870;
float lastError = 0.0;
float integral = 0.0;
bool follow_line = false;
int outline = 0;
int in_line = 0;


void setup()
{
    Serial.begin(115200);
    uint8_t Pins[] = {SENSOR_PIN_1, SENSOR_PIN_2, SENSOR_PIN_3, SENSOR_PIN_4,
SENSOR_PIN_5};
    uint8_t ENCODER_PINS[] = {ENCODER_PIN_1, ENCODER_PIN_2};
    pinMode(SERVO_MOTOR_1, OUTPUT);
    serialOutput += reflectance_sensor->init(Pins);
    encoder->init(ENCODER_PINS);
    motor->init();
    stepper_motor->init(STEP_MOTOR_PIN_1, STEP_MOTOR_PIN_3, STEP_MOTOR_PIN_2,
STEP_MOTOR_PIN_4);
    delay(1000);
    levanta_bico();
```

```
    // abaixa_bico();
    delay(1000);
    config_wifi();
}

void loop()
{
    ArduinoOTA.handle();
    if (start_robot)
    {

        if (motorState == LOW)
        {
            motorState = HIGH;
        }

        unsigned long currentMillis = millis();
        currentMillis2 = millis();
        encoder->update();
        if (currentMillis - previousMillis >= 10)
        {
            previousMillis = currentMillis;
            rejuntar();
        }
    }
    else
    {
        if (motorState == HIGH)
        {
            motorState = LOW;
            serialOutput += "\n\nRobô parado\nDistancia: " +
String(steps_to_mm(int(encoder->getCountEsq()))) + " mm\n\n" ;
            analogWrite(MOTOR_PIN_1, 0);
            analogWrite(MOTOR_PIN_4, 0);
            stepper_motor->stop();
            motor->init();
            // targetPosition = steps_to_mm(int(encoder->getCountEsq())) +
targetPosition;
        }
    }
}

void LineFollower()
{
    Kp = 0.35;
    Ki = 0.0;
    Kd = 0.0;
    Ts = 10;
    error = 0;
```

```
    in_line = 0;
    float sensorValues[4] = { (4095 - reflectance_sensor->readLine()[0])*-2,
                              (4095 - reflectance_sensor->readLine()[1])*-1,
                              (4095 - reflectance_sensor->readLine()[2])*1,
                              (4095 - reflectance_sensor->readLine()[3])*2};
    for (int i = 0; i < 4; i++)
    {
        error += sensorValues[i];
    }
    integral += ((error+lastError) * Ts)/2;
    float derivative = (error - lastError)/Ts;
    float control = Kp * error +  Ki * integral + Kd * derivative;
    lastError = error;

    // serialOutput += "Control: " + String(control) + " B+ : " +
String(control)+ "\n";
    // control = constrain(control, -55,55);
    control = 0;
    motor->updatePID(encoder->getVelEsq(),
                     encoder->getVelDir(),
                     baseSpeed - control,
                     baseSpeed + control);
    serialOutput += "Erro_debug: " + motor->getDebug() + "\n";
    // serialOutput += "Erro: " + String(baseSpeed- encoder->getVelDir());
    // serialOutput += (" velocidade esq: " + String(encoder->getVelEsq())) +
" |";
    // serialOutput += (" velocidade dir: " + String(encoder->getVelDir())) +
"\n";
}



void abaixa_bico()
{
    if( !bico_abaixado)
    {
        analogWrite(SERVO_MOTOR_1, 120);
        bico_abaixado = true;
    }
}

void levanta_bico()
{
    if(bico_abaixado)
    {
        analogWrite(SERVO_MOTOR_1, 254);
        bico_abaixado = false;
    }
```

```
}

void rejuntar()
{

    // serialOutput += reflectance_sensor->getLine();
    if (!bico_abaixado)
    {
        abaixa_bico();
    }
    // stepper_motor->start();
    LineFollower();
    if (int(encoder->getCountEsq()) > mm_to_steps(targetPosition))
    {
        if (bico_abaixado)
        {
            levanta_bico();
        }
        motor->updatePWM(0,0,0,0);
        start_robot = false;
    }
}

void config_wifi() {
    WiFi.begin(ssid, password);
    while (WiFi.status() != WL_CONNECTED) {
        delay(500);
        Serial.println("Conectando ao WiFi...");
    }
    Serial.println("Conectado ao WiFi!");

    if (!SPIFFS.begin(true)) {
        Serial.println("Erro ao iniciar o SPIFFS!");
        return;
    }

    server.on("/", HTTP_GET, [](AsyncWebServerRequest *request) {
        request->send(SPIFFS, "/index.html", "text/html");
    });

    server.on("/sendMessage", HTTP_POST, [](AsyncWebServerRequest *request)
    {
        String co =  request->argName(0);
        Serial.println("Recebendo comando...");
        Serial.println("Comando recebido: " + request->arg(co));
        Serial.println("Comando recebido: " + request->argName(0));
        serialOutput += "Recebendo comando...\n";
        serialOutput += "Comando recebido: " + request->arg(co) + "\n";
```

```
        if (request->argName(0)) {
            String co =  request->argName(0);
            String command = request->arg(co);
            if ((command == "start") || (command == "run")) {
                start_robot = true;
            }
            else if (command.substring(0, 4) == "vel=")
            {
                serialOutput += "Velocidade: " + command.substring(4) + "\n";
                baseSpeed = command.substring(4).toFloat();
            }
            else if (command.length() > 7 && command.substring(0, 7) ==
"target=")
            {
                targetPosition = command.substring(7).toInt();
            }
            else if (command == "stop")
            {
                analogWrite(MOTOR_PIN_1, 0);
                analogWrite(MOTOR_PIN_2, 0);
                analogWrite(MOTOR_PIN_3, 0);
                analogWrite(MOTOR_PIN_4, 0);
                start_robot = false;
            }
            else if (command == "stop") {
                analogWrite(MOTOR_PIN_1, 0);
                analogWrite(MOTOR_PIN_2, 0);
                analogWrite(MOTOR_PIN_3, 0);
                analogWrite(MOTOR_PIN_4, 0);
                start_robot = false;
            }
            request->send(200, "text/plain", "Comando '" + command + "'
recebido e processado.");
        } else {
            request->send(400, "text/plain", "Parâmetro não encontrado na
requisição.");
        }
        Serial.println("Comando processado.");
    });

    server.on("/sendCommand", HTTP_POST, [](AsyncWebServerRequest *request) {
        String co =  request->argName(0);
        Serial.println("Recebendo comando...");
        Serial.println("Comando recebido: " + request->arg(co));
        Serial.println("Comando recebido: " + request->argName(0));
        if (request->argName(0)) {
            String co =  request->argName(0);
            String command = request->arg(co);
            if ((command == "run") || (command == "start")) {
```

```
                start_robot = true;
            } else if (command == "stop") {
                analogWrite(MOTOR_PIN_1, 0);
                analogWrite(MOTOR_PIN_2, 0);
                analogWrite(MOTOR_PIN_3, 0);
                analogWrite(MOTOR_PIN_4, 0);
                start_robot = false;
            }
            request->send(200, "text/plain", "Comando '" + command + "'
recebido e processado.");
        } else {
            request->send(400, "text/plain", "Parâmetro 'command' não
encontrado na requisição.");
        }
        Serial.println("Comando processado.");
    });

    server.on("/serial", HTTP_GET, [](AsyncWebServerRequest *request) {
        request->send(200, "text/plain", serialOutput);
        if (serialOutput.length() > 300) {
            serialOutput = "";
        }
    });

    server.begin();
    ArduinoOTA.setHostname("robot");
    ArduinoOTA.setPassword("quesenh@");
    ArduinoOTA.begin();
}
```

APÊNDICE C - Motor.h

```
#ifndef MOTOR_H
#define MOTOR_H

#include <Arduino.h>

class Motor
{
public:
    Motor(uint8_t pins[]);
    ~Motor();
    void init();
    void updatePWM(float vel_esq, float vel_dir, float_t targetSpeed_esq,
float_t targetSpeed_dir);
```

```
    void updatePID(float vel_esq, float vel_dir, float_t targetSpeed_esq,
float_t targetSpeed_dir);
    String getDebug();

private:
    String debug;
    uint32_t debugdelay;
    void setMotorEsqPWM(int pwm);
    void setMotorDirPWM(int pwm);
    uint8_t pins[4];
    unsigned long currentMillis;
    unsigned long currentMillisIntegrator;
    float Ts;
    float vel_corr;

    // float targetSpeed_esq;
    float currentSpeed_esq;
    float lastError_esq;
    float integral_esq;
    float Kp_esq, Ki_esq, Kd_esq;

    // float targetSpeed_dir;
    float currentSpeed_dir;
    float lastError_dir;
    float integral_dir;
    float Kp_dir, Ki_dir, Kd_dir;
    unsigned long lastUpdateTime;

    float PWM_esq;
    float PWM_dir;
};

#endif
```

Motor.cpp

```
#include "../lib/motor.h"

Motor::Motor(uint8_t pins[])
{
    for (int i = 0; i < 4; i++)
    {
        this->pins[i] = pins[i];
        pinMode(this->pins[i], OUTPUT);
        analogWrite(this->pins[i], 0);
    }
```

```
}

Motor::~Motor() {}

void Motor::init()
{
    debug = "";
    currentSpeed_esq = 0;
    lastError_esq = 0;
    integral_esq = 0;

    Kp_esq = 0.055;
    Ki_esq = 0.001;
    Kd_esq = 0.010;

    currentSpeed_dir = 0;
    lastError_dir = 0;
    integral_dir = 0;

    Kp_dir = 0.055;
    Ki_dir = 0.001;
    Kd_dir = 0.010;

    lastUpdateTime = 0;
    debugdelay = 0;
    currentMillisIntegrator = 0;
    currentMillis = 0;
    PWM_dir = 0;
    PWM_esq = 0;

    Ts = 0.01;
}

void Motor::updatePWM(float vel_esq, float vel_dir, float_t targetSpeed_esq,
float_t targetSpeed_dir)
{
    if (millis() - currentMillis >= Ts*1000) //10 ms
    {
        currentMillis = millis();
        // float ajuste = constrain((vel_esq + vel_dir)/2, 0.5, 1);
        //          100 - 53,5 cm/s
        //         1.8691589 cm/s

        analogWrite(pins[0], targetSpeed_esq); // LEFT FORWARD
        analogWrite(pins[1], 0);   // LEFT BACKWARD
        analogWrite(pins[2], 0);   // RIGHT BACKWARD
        analogWrite(pins[3], targetSpeed_dir); // RIGHT FORWARD
    }
}
```

```
void Motor::updatePID(float vel_esq, float vel_dir, float_t targetSpeed_esq,
float_t targetSpeed_dir)
{
    if (millis() - currentMillis >= Ts*1000) //10 ms
    {
        currentMillis = millis();
        // // Controlador PID para a roda esquerda
        float error = (targetSpeed_dir) - vel_dir;
        float proportional = Kp_esq * error;
        integral_esq += ((error+lastError_esq) * Ts)/2;
        float derivative = (error - lastError_esq) / Ts;
        float output = proportional + (Ki_esq * integral_esq) + Kd_esq *
derivative;
        lastError_esq = error;
        PWM_esq += output;
        setMotorEsqPWM(PWM_esq);
        if ( debug.length() < 500)
            debug += "Saida PWM: " + String(PWM_esq) + ", vel_dir: " +
String(vel_dir) +  " || ";
        else
            debug = "";

        // // // Controlador PID para a roda direita
        error = (targetSpeed_esq) - vel_esq;
        proportional = Kp_dir * error;
        integral_dir += ((error+lastError_dir) * Ts)/2;
        derivative = (error - lastError_dir) / Ts;
        output = proportional + (Ki_dir * integral_dir) + Kd_dir * derivative;
        lastError_dir = error;
        PWM_dir += output;
        setMotorDirPWM(PWM_dir);
        if ( debug.length() < 500)
            debug += "Saida PWM: " + String(PWM_dir) + ", vel_esq: " +
String(vel_esq) +  "\n";
        else
            debug = "";
    }
}

void Motor::setMotorEsqPWM(int pwm)
{
    pwm = constrain(pwm, -255, 255); // limita o valor de pwm entre -255 e 255

    if (pwm > 0)
    {

        analogWrite(pins[0], pwm);
        analogWrite(pins[1], 0);
```

```
    }
    // else
    // {
    //     pwm = -pwm;
    //     analogWrite(pins[0], 0);
    //     analogWrite(pins[1], pwm);
    // }

}

void Motor::setMotorDirPWM(int pwm)
{
    pwm = constrain(pwm, -255, 255); // limita o valor de pwm entre -255 e 255

    if (pwm > 0)
    {
        analogWrite(pins[2], 0);
        analogWrite(pins[3], pwm);
    }
    // else
    // {
    //     pwm = -pwm;
    //     analogWrite(pins[2], pwm);
    //     analogWrite(pins[3], 0);
    // }
}

String Motor::getDebug()
{
    return debug;
}
```

APÊNDICE D - StepperMotor.h

```
#include <AccelStepper.h>

class StepperMotor {
public:
    StepperMotor(const uint8_t pin1, const uint8_t pin2, const uint8_t pin3,
const uint8_t pin4);
    StepperMotor(){}
    ~StepperMotor();
    void init(const uint8_t pin1, const uint8_t pin2, const uint8_t pin3,
const uint8_t pin4);
```

```
    void start();
    void stop();
private:
    int pin1_;
    int pin2_;
    int pin3_;
    int pin4_;
    AccelStepper *stepper;
    bool steppermoving = false;
    long targetPosition = 0;

};
```

StepperMotor.cpp

```
#include "../lib/stepper_motor.h"


StepperMotor::StepperMotor(const uint8_t pin1, const uint8_t pin2, const
uint8_t pin3, const uint8_t pin4) {}
// StepperMotor::StepperMotor() {}

StepperMotor::~StepperMotor() {}

void StepperMotor::init(const uint8_t pin1, const uint8_t pin2, const uint8_t
pin3, const uint8_t pin4)
{
    stepper = new AccelStepper(AccelStepper::FULL4WIRE, pin1, pin2, pin3,
pin4);
    stepper->setMaxSpeed(400.0);
    stepper->setAcceleration(300.0);
}

void StepperMotor::start()
{
    unsigned long currentMillis = millis();
    if (!steppermoving) {
            targetPosition = 50*2048; // a cada 50 voltas sentido horario da
face superior do motor
            stepper->moveTo(targetPosition);
            steppermoving = true;
        }
        stepper->run();
        if (stepper->distanceToGo() == 0) {
            steppermoving = false;
        }
}
void StepperMotor::stop()
{
```

```
    stepper->stop();
     if (steppermoving)
    {
        steppermoving = false;
    }
}
```

APÊNDICE E - ReflectanceSensor.h

```
#ifndef REFLECTANCE_SENSOR_H
#define REFLECTANCE_SENSOR_H

#include <Arduino.h>
#include <SimpleKalmanFilter.h>
#include <vector>
class ReflectanceSensor {
public:
    ReflectanceSensor();
    ~ReflectanceSensor();
    void calibrate();
    std::vector<float> readLine();
    String getLine();
    int get_Error();
    String init(uint8_t Pins[]);

private:
    const int numSensors = 5;
    uint8_t sensorPins[5];
    int min_error[5];
    int max_error[5];
    int value;
    bool calibrated = false;

    String serialOutput;
    SimpleKalmanFilter filter[4];
};

#endif // REFLECTANCE_SENSOR_H
```

ReflectanceSensor.cpp

```
#include "../lib/reflectance_sensor.h"

ReflectanceSensor::ReflectanceSensor() : filter{SimpleKalmanFilter(2.0, 2.0,
0.01),
```

```
                                                SimpleKalmanFilter(2.0, 2.0,
0.01),
                                                SimpleKalmanFilter(2.0, 2.0,
0.01),
                                                SimpleKalmanFilter(2.0, 2.0,
0.01)}
{}

ReflectanceSensor::~ReflectanceSensor()
{
}
void ReflectanceSensor::calibrate()
{
    unsigned long previousMillis = 0;
    for (uint8_t i = 0; i < numSensors; i++)
    {
        min_error[i] = 4095;
        max_error[i] = 0;
    }
    while(calibrated == false)
    {
        if (millis() - previousMillis >= 10)
        {
            for (uint8_t i = 0; i < numSensors; i++)
            {
                if(min_error[i] > analogRead(sensorPins[i]))
                {
                    min_error[i] = analogRead(sensorPins[i]);
                }
                if(max_error[i] < analogRead(sensorPins[i]))
                {
                    max_error[i] = analogRead(sensorPins[i]);
                }
            }
        }
        if (millis() - previousMillis >= 3000)
        {
            calibrated = true;
        }
        previousMillis = millis();
    }
}

String ReflectanceSensor::init(uint8_t Pins[]) {
    pinMode(Pins[0], OUTPUT);
    digitalWrite(Pins[0], LOW);
    for (int i = 1; i < numSensors; i++) {
        sensorPins[i] = Pins[i];
        pinMode(sensorPins[i], INPUT);
```

```
        serialOutput += "Sensor " + String(i) + " pin: " +
String(sensorPins[i]) + "\n";
    }
    return serialOutput;
}
String ReflectanceSensor::getLine() {
    serialOutput = "";
    for (uint8_t i = 1; i < numSensors; i++)
    {
        // serialOutput += "R" + String(sensorPins[i]) +": "+
String(filter[i].updateEstimate(analogRead(sensorPins[i]))) + " ";
        serialOutput += "R" + String(sensorPins[i]) +": "+ String(filter[i-
1].updateEstimate(analogRead(sensorPins[i]))) + " ";
    }
    return serialOutput;
}

std::vector<float> ReflectanceSensor::readLine() {
    std::vector<float> line;
    for (uint8_t i = 1; i < numSensors; i++)
    {
        line.push_back(filter[i-1].updateEstimate(analogRead(sensorPins[i])));
    }
    return line;
}

int ReflectanceSensor::get_Error() {
    value = 0;

    value += analogRead(sensorPins[1]) * -2;
    value += analogRead(sensorPins[2]) * -1;
    value += analogRead(sensorPins[3]) * 1;
    value += analogRead(sensorPins[4]) * 2;
    return value;
}
```

APÊNDICE F - Encoder.h

```
#ifndef ENCODER_HPP
#define ENCODER_HPP

#include <Arduino.h>
#include <SimpleKalmanFilter.h>

class Encoder {
public:
```

```
    Encoder();
    ~Encoder();
    void init(uint8_t Pins[]);
    void update();
    String getCount();
    float getCountEsq();
    float getCountDir();
    String getVel();
    float getVelEsq();
    float getVelDir();
    String Getdebug();
private:

    SimpleKalmanFilter filterEsq;
    SimpleKalmanFilter filterDir;

    bool calc_hysteresis(uint32_t value, bool last);
    uint8_t pins[2];
    uint32_t Count[2];
    uint32_t Count_vel[2];
    uint32_t interval;
    uint32_t hysteresis;

    float velEsq;
    float velDir;
    bool lastSample[2];
    bool currentSample[2];

    unsigned long last_tEsq;
    unsigned long last_tDir;
    unsigned long tEsq;
    unsigned long tDir;

    unsigned long previousMillis;
    unsigned long debounce;
    unsigned long lastDebounce[2];
    unsigned long Ts;
    String debug;


};

#endif // ENCODER_HPP
```

Encoder.cpp

```
#include "../lib/encoder.h"
```

```
Encoder::Encoder()
    : filterEsq(1.0, 1.0, 0.01), filterDir(1.0, 1.0, 0.01) {}
Encoder::~Encoder() {}

void Encoder::init(uint8_t Pins[])
{
    interval = 3900;
    hysteresis = 100;
    Count[0] = 0;
    Count[1] = 0;
    Count_vel[0] = Count[0];
    Count_vel[1] = Count[1];
    Ts = 1;
    debounce = 2;
    lastDebounce[0] = 0;
    lastDebounce[1] = 0;

    pins[0] = Pins[0];
    pins[1] = Pins[1];
    pinMode(pins[0], INPUT);
    pinMode(pins[1], INPUT);
    lastSample[0] = false;
    lastSample[1] = false;
    currentSample[0] = calc_hysteresis(analogRead(pins[0]),lastSample[0]);
    currentSample[1] = calc_hysteresis(analogRead(pins[1]),lastSample[1]);
    debug = "";
    last_tDir = 0;
    last_tEsq = 0;
    tDir = 0;
    tEsq = 0;
}

void Encoder::update()
{
    // calculo de pulsos
    if (millis() - previousMillis >= Ts)
    {
        if(((currentSample[0] != lastSample[0]) && lastSample[0] == 0)  &&
(millis() - lastDebounce[0] > debounce))
        {
            Count[0]++;
            last_tDir = tDir;
            tDir = millis();
            lastDebounce[0] = tDir;
        }
        if (((currentSample[1] != lastSample[1]) && lastSample[1] == 0 ) &&
(millis() - lastDebounce[1] > debounce))
        {
```

```
            Count[1]++;
            last_tEsq = tEsq;
            tEsq = millis();
            lastDebounce[1] = tEsq;
        }
        previousMillis = millis();
        lastSample[0] = currentSample[0];
        lastSample[1] = currentSample[1];

        currentSample[0] =
calc_hysteresis(analogRead(pins[0]),currentSample[0]);
        currentSample[1] =
calc_hysteresis(analogRead(pins[1]),currentSample[1]);
    }

    // calculo de velocidade
    if ((millis() - last_tEsq) > 150)
    {
        velEsq = 0; // cm/s
    }
    else if ((tEsq - last_tEsq) > 1)
    {
        velEsq = float(PI/10)/(float(tEsq - last_tEsq)*0.001); //Rad/s
        velEsq = filterEsq.updateEstimate(velEsq); // Rad/s
        velEsq = (velEsq*0.0345)*100; // cm/s
    }
    if ((millis() - last_tDir) > 150)
    {
        velDir = 0; // cm/s
    }
    else if ((tDir - last_tDir) > 1)
    {
        velDir = float(PI/10)/(float(tDir - last_tDir)*0.001); //Rad/s
        velDir = filterDir.updateEstimate(velDir); // Rad/s
        velDir = (velDir*0.0345)*100; // cm/s
    }


}

bool Encoder::calc_hysteresis(uint32_t value, bool last)
{
    if (value < (interval - hysteresis))
    {
        return false;
    }
    else if (value > (interval + hysteresis))
    {
        return true;
```

```
    }
    else
    {
        return last;
    }
}

String Encoder::getCount()
{
    return String(Count[0]) + " " + String(Count[1]);
}
String Encoder::getVel()
{
    return "Velocidade esquerda: " + String(velEsq) + " cm/s |" + "Velocidade
direita: " + String(velDir) + " cm/s";
}
float Encoder::getVelEsq()
{
    return velEsq;
}
float Encoder::getVelDir()
{
    return velDir;
}

float Encoder::getCountEsq()
{
    return Count[0];
}
float Encoder::getCountDir()
{
    return Count[1];
}

String Encoder::Getdebug()
{
    return debug;
}
```

APÊNDICE G - Index.html

```
<!DOCTYPE html>
<html>
<head>
    <title>HOME - Aplicador de rejunte</title>
    <meta charset="UTF-8">
    <style>
        body {
            background-color: #222;
            color: #eee;
        }

        #serialOutputContainer {
            border: 2px solid #555;
            padding: 10px;
            overflow-y: scroll;
            height: 250px;
            font-family: monospace;
            white-space: pre-wrap;
            background-color: #333;
            color: #f0f0f0;
        }

        input {
            background-color: #333;
            color: #eee;
            border: 1px solid #555;
            padding: 10px 20px;
            border-radius: 5px;
            font-size: 16px;
            margin: 5px 0;
        }

        button {
            padding: 10px 20px;
            border-radius: 5px;
            font-size: 16px;
            margin: 5px 0;
            border: none;
        }

        h1 {
            font-family: sans-serif;
        }

        #startButton {
            background-color: #4CAF50; /* Cor verde */
```

```
            color: white;
        }

        #stopButton {
            background-color: #f44336; /* Cor vermelha */
            color: white;
        }

        button:hover {
            opacity: 0.8; /* Efeito de hover mais sutil */
        }
    </style>
    <script>
        function sendMessage() {
            var message = document.getElementById("messageInput").value;
            var xhttp = new XMLHttpRequest();
            xhttp.open("POST", "/sendMessage", true);
            xhttp.setRequestHeader("Content-type", "application/x-www-form-
urlencoded");
            xhttp.send("message=" + message);
        }

        function sendCommand(command) {
            var xhttp = new XMLHttpRequest();
            xhttp.open("POST", "/sendCommand", true);
            xhttp.setRequestHeader("Content-type", "application/x-www-form-
urlencoded");
            xhttp.onreadystatechange = function() {
                if (this.readyState == 4 && this.status == 200) {
                    console.log(this.responseText);

                    // Atualizar a interface aqui (por exemplo, mudar a cor
dos botões)
                    if (command == "start") {
                        document.getElementById("startButton").style.backgroun
dColor = "#4CAF50";
                    } else if (command == "stop") {
                        document.getElementById("stopButton").style.background
Color = "#f44336";
                    }
                }
            };
            xhttp.send("command=" + command);
        }

        function updateSerialOutput() {
            var xhttp = new XMLHttpRequest();
            xhttp.onreadystatechange = function() {
                if (this.readyState == 4 && this.status == 200) {
```

```
                    var serialOutput =
document.getElementById("serialOutput");
                    serialOutput.textContent = this.responseText;
                    var container =
document.getElementById("serialOutputContainer");
                    container.scrollTop = container.scrollHeight;
                }
            };
            xhttp.open("GET", "/serial", true);
            xhttp.send();
        }

        setInterval(updateSerialOutput, 1000); // Atualizar a cada 1 segundo
    </script>
</head>
<body>
    <h1>Robô aplicador de rejunte</h1>

    <input type="text" id="messageInput" placeholder="Digite uma mensagem">
    <button onclick="sendMessage()">Enviar</button>

    <br><br><br><br>

    <button id="startButton" onclick="sendCommand('start')">Run</button>
    <button id="stopButton" onclick="sendCommand('stop')">Stop</button>

    <h2>Monitor Serial</h2>
    <div id="serialOutputContainer">
        <pre id="serialOutput"></pre>
    </div>
</body>
</html>
```