UNIVERSIDADE FEDERAL DE SANTA CATARINA
CENTRO TECNOLÓGICO, DE CIÊNCIAS EXATAS E EDUCAÇÃO
DEPARTAMENTO DE ENG. DE CONTROLE, AUTOMAÇÃO E COMPUTAÇÃO
CURSO DE GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Bruno Luís Trevisani

Construção de um sistema LIDAR para escaneamento de ambientes em 3D

Blumenau
2024

Bruno Luís Trevisani

Construção de um sistema LIDAR para escaneamento de ambientes em 3D

Trabalho de Conclusão de Curso de Graduação em Engenharia de Controle e Automação do Centro Tecnológico, de Ciências Exatas e Educação da Universidade Federal de Santa Catarina como requisito para a obtenção do título de Engenheiro de Controle e Automação.
Orientador: Prof. Carlos Moratelli, Dr.

Blumenau
2024

Ficha catalográfica gerada por meio de sistema automatizado gerenciado pela BU/UFSC.
Dados inseridos pelo próprio autor.

Trevisani, Bruno Luís
Construção de um sistema LIDAR para escaneamento de ambientes em 3D / Bruno Luís Trevisani ; orientador, Carlos Roberto Moratelli, 2024.
53 p.

Trabalho de Conclusão de Curso (graduação) - Universidade Federal de Santa Catarina, Campus Blumenau, Graduação em Engenharia de Controle e Automação, Blumenau, 2024.

Inclui referências.

1. Engenharia de Controle e Automação. 2. LIDAR. 3. Reconstrução de Superfície. 4. Mapeamento interno. I. Moratelli, Carlos Roberto. II. Universidade Federal de Santa Catarina. Graduação em Engenharia de Controle e Automação. III. Título.

Bruno Luís Trevisani

Construção de um sistema LIDAR para escaneamento de ambientes em 3D

Este Trabalho de Conclusão de Curso foi julgado adequado para obtenção do Título de "Engenheiro de Controle e Automação" e aprovado em sua forma final pelo Curso de Graduação em Engenharia de Controle e Automação.

Blumenau, 03 de Julho de 2024.

Banca Examinadora:

Documento assinado digitalmente
Carlos Roberto Moratelli
Data: 11/07/2024 13:25:49-0300
CPF: ***.999.369-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Dr. Carlos Roberto Moratelli
Universidade Federal de Santa Catarina

Documento assinado digitalmente
Ciro Andre Pitz
Data: 11/07/2024 13:33:46-0300
CPF: ***.728.089-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Dr. Ciro André Pitz
Universidade Federal de Santa Catarina

Documento assinado digitalmente
MARCOS VINICIUS MATSUO
Data: 11/07/2024 13:55:11-0300
CPF: ***.580.029-**
Verifique as assinaturas em https://v.ufsc.br

Prof. Dr. Marcos Vinicius Matsuo
Universidade Federal de Santa Catarina

Este trabalho dedico á minha família e amigos que sempre
me auxiliaram e ajudaram em minha trajetória.

RESUMO

Este trabalho aborda a aplicação da tecnologia LIDAR para escaneamento de ambientes internos em 3D, apresentando um protótipo integrado com componentes eletrônicos controlados por algoritmos. O estudo envolve a construção do sistema, incluindo estruturas impressas em 3D, motores, sensor LIDAR e o desenvolvimento de *softwares* para processar os dados em uma imagem 3D, comparando diferentes métodos de reconstrução de superficie. A partir dos dados coletados pelo mapeamento de um ambiente, eles foram processados em uma nuvem de pontos e convertidos para uma imagem 3D. Para essa converção foi aplicado três métodos de reconstrução de superficie, que foram: *Ball-Pivoting*, Poisson e *A-Shapes*. As imagens geradas pelos métodos *Ball-Pivoting* e *A-Shapes*, demostraram lacunas e superficies em lugares indesejados, embora fosse possível traçar um paralelo entre a imagem gerada e o ambiente real. A imagem por Poisson se demostrou mais eficaz, com uma superficie continua e suavizada, embora apresentasse algumas irregularidades. Alguns pontos foram levantados para as causas dessas irregularidades, porém os resultados de um protótipo de baixo custo foram satifatórios e demonstram a importância da integração de *hardware* e *software* para soluções de mapeamento interno acessíveis e eficazes.

Palavras-chave: LIDAR; reconstrução de superfície; *Ball-Pivoting*; Poisson; *A-Shapes*; mapeamento interno.

ABSTRACT

This work addresses the application of LIDAR technology for 3D scanning of indoor environments, presenting a prototype integrated with electronic components controlled by algorithms. The study involves the construction of the system, including 3D printed structures, motors, a LIDAR sensor, and the development of software to process the data into a 3D image, comparing different surface reconstruction methods. The data collected by mapping an environment were processed into a point cloud and converted into a 3D image. For this conversion, three surface reconstruction methods were applied: Ball-Pivoting, Poisson, and A-Shapes. The images generated by the Ball-Pivoting and A-Shapes methods showed gaps and surfaces in undesirable places, although it was possible to draw a parallel between the generated image and the real environment. The image produced by the Poisson method proved to be more effective, with a continuous and smooth surface, although some irregularities were present. Some causes for these irregularities were identified, but the results of a low-cost prototype were satisfactory and demonstrate the importance of integrating hardware and software for accessible and effective indoor mapping solutions.

Keywords: LIDAR; surface reconstruction; Ball-Pivoting; Poisson; A-Shapes; indoor mapping.

LISTA DE FIGURAS

Figura 1 – Princípio de medição do *Time-of-Flight* com laser pulsado. 14
Figura 2 – *Phase-Based* princípio de medição. 14
Figura 3 – Reflexão Regular. 15
Figura 4 – Reflexão Difusa. 15
Figura 5 – Comportamento do Raio na Reflexão Regular. 16
Figura 6 – Coordenada Polar. 17
Figura 7 – Conversão de um Ponto para Coordenadas Polares. 18
Figura 8 – Conversão de um Ponto para Coordenadas Polares Tridimensionais. . . 19
Figura 9 – Pinagem do ESP32. 19
Figura 10 – Componentes do servo motor. 21
Figura 11 – Movimentação do servo motor. 21
Figura 12 – Motor de Relutância Variável. 22
Figura 13 – Motor de Imã Permanente. 23
Figura 14 – Motor Híbrido. 23
Figura 15 – Técnica *Ball-Pivonting*. 25
Figura 16 – Ilustração intuitiva da reconstrução por Poisson em 2D. 26
Figura 17 – Processo de Triangulação. 27
Figura 18 – Diagrama do Protótipo. 31
Figura 19 – Diagrama do Processamento da Imagem. 31
Figura 20 – Estrutura Principal 3D. 32
Figura 21 – Estrutura Móvel 3D. 32
Figura 22 – Eixo Inferior da Estrutura Móvel 3D. 33
Figura 23 – Protótipo Montado. 33
Figura 24 – Motor de passo Nema 17. 34
Figura 25 – Micro servo motor SG90. 35
Figura 26 – Sensor LIDAR TF Mini. 35
Figura 27 – Diagrama do Algoritmo de Reconstrução de Imagem. 36
Figura 28 – Quarto Mapeado. 41
Figura 29 – Perspectiva 1 da Nuvem de Pontos do Ambiente. 42
Figura 30 – Perspectiva 2 da Nuvem de Pontos do Ambiente. 42
Figura 31 – Ambiente reconstruído por Poisson. 43
Figura 32 – Comparação do ambiente reconstruído por Poisson e real. 44
Figura 33 – Ambiente reconstruído por *Ball-Pivoting*. 44
Figura 34 – Comparação do ambiente reconstruído por *Ball-Pivoting* e real. 45
Figura 35 – Ambiente por *Ball-Pivoting* com raio de vizinhança aumentado. 45
Figura 36 – Ambiente reconstruído por *A-Shapes*. 46
Figura 37 – Comparação do ambiente reconstruído por *A-Shapes* e real. 46

Figura 38 – Ambiente reconstruído por *A-Shapes* por outra perspectiva. 47

SUMÁRIO

1 INTRODUÇÃO 11
1.1 OBJETIVO GERAL 12
1.2 OBJETIVOS ESPECÍFICOS 12
1.3 ORGANIZAÇÃO DO TRABALHO 12
2 FUNDAMENTAÇÃO TEÓRICA 13
2.1 TECNOLOGIAS UTILIZADAS 13
2.1.1 LIDAR 13
2.1.2 Reflexão da Luz 14
2.1.3 Coordenadas Polares 16
2.1.3.1 Conversão de Coordenadas Cartesianas para Coordenas Polares Bidimensionais 17
2.1.3.2 Conversão de Coordenadas Cartesianas para Coordenas Polares Tridimensionais 17
2.1.4 ESP32 18
2.1.5 Servo Motor 20
2.1.6 Motor de Passo 21
2.1.6.1 Relutância Variável 22
2.1.6.2 Imã Permanente 22
2.1.6.3 Híbrido 22
2.1.7 *Real-Time Operationg System* 23
2.1.7.1 FreeRTOS 24
2.1.8 Reconstrução de Superfície 24
2.1.8.1 Ball-Pivonting Algorithm 25
2.1.8.2 Reconstrução de Superfície por Poisson 26
2.1.8.3 Triangulação de Delaunay 26
2.1.8.4 Reconstrução de Superfície por A-Shapes 27
2.2 TRABALHOS RELACIONADOS 28
3 PROPOSTA E IMPLEMENTAÇÃO DE UM DISPOSITIVO PARA MAPEAMENTO 3D 30
3.1 DESCRIÇÃO DO PROBLEMA 30
3.2 ARQUITETURA DA SOLUÇÃO 30
3.3 O PROTÓTIPO 30
3.3.1 Reconstrução de Superfície em uma Imagem 3D 36
4 RESULTADOS 40
4.1 AMBIENTE DE TESTES 40
4.2 NUVEM DE PONTOS 40
4.3 RECONSTRUÇÃO DE IMAGEM POR POISSON 41

4.4 RECONSTRUÇÃO DE IMAGEM POR *BALL-PIVOTING* 42
4.5 RECONSTRUÇÃO DE IMAGEM POR *A-SHAPES* 43
5 CONCLUSÃO 48
REFERÊNCIAS 49
ANEXO A – Sistema de Mapeamento de Ambiente 3D . . . 53

1 INTRODUÇÃO

O avanço da tecnologia de sensoriamento tem aberto inúmeras possibilidades para aplicações e estudos que antes não eram viáveis devido à falta de ferramentas adequadas. Um dos campos que se beneficiam significativamente desse progresso é o mapeamento tridimensional de ambientes por sensores de distância. Esse tipo de mapeamento é amplamente utilizado em diversas áreas, como robótica, mineração, arquitetura, engenharia florestal e engenharia civil, entre outras. Com o desenvolvimento contínuo dessas tecnologias, a demanda por mapeamento de ambientes tem crescido, acompanhando o ritmo acelerado das inovações no setor.

O crescimento acelerado nesse setor se deve principalmente aos avanços tecnológicos dos sistemas de escaneamento a LASER (*Light Amplification by Stimulated Emission of Radiation*), que são amplamente utilizados na reconstrução 3D de ambientes internos e externos. A tecnologia LIDAR, pertencente à área de sensoriamento, tem como finalidade a medição da distância entre o sensor e uma superfície por meio da emissão de pulsos LASER. Os sensores LIDAR são muito utilizados para mapeamento de ambiente por gerarem uma resposta rápida, precisa e segura em relação a outras tecnologias (PAVAN; SANTOS, 2015).

A reconstrução de superfície a partir de uma nuvem de pontos é um processo crucial em várias áreas, como modelagem 3D, visão computacional e reconstrução de ambientes virtuais. Esse processo envolve transformar uma coleção de pontos dispersos, frequentemente obtidos de scanners 3D, sensores LIDAR ou técnicas de fotogrametria, em uma superfície contínua que representa a forma de um objeto ou cenário. Existem diversas técnicas para essa transformação, cada uma com suas particularidades. A Triangulação de Delaunay é utilizada para criar redes de triângulos. A-Shapes são uma generalização da triangulação de Delaunay que considera um parâmetro alfa para controlar a precisão e a forma da superfície reconstruída. O *Ball Pivoting Algorithm* (BPA), ou Algoritmo de Pivotamento de Esferas, gera superfícies conectando pontos vizinhos por meio de uma esfera de raio fixo que pivota sobre os pontos. A Reconstrução de Superfície Poisson gera superfícies suaves resolvendo a equação de Poisson.

Sendo assim, este trabalho visa desenvolver um sistema de mapeamento interno de baixo custo, utilizando LIDAR para auxiliar na geração da nuvem de pontos e a reconstrução da superfície a partir dessa nuvem. Com isso, será possível analisar a eficiência do uso do LIDAR em um protótipo econômico, comparando as imagens geradas por diferentes técnicas de reconstrução de superfície.

Para o mapeamento do ambiente, foram projetadas e impressas peças em 3D que servem como suporte para os componentes elétricos do protótipo. O circuito de componentes inclui motor de passo, servo motor, sensor LIDAR, driver para motor, microcontrolador e fonte de alimentação. Para gerar as imagens 3D, foram programados algoritmos em *python*

que capturam os dados gerados pelo protótipo, criando uma nuvem de pontos e gerando uma imagem 3D a partir de uma das técnicas de reconstrução de imagem.

1.1 OBJETIVO GERAL

A proposta deste documento consiste na montagem de um protótipo de baixo custo para mapeamento de um espaço e do processamento dos pontos para reconstrução de uma imagem tri-dimensional.

1.2 OBJETIVOS ESPECÍFICOS

1. Construir um protótipo de baixo custo;
2. Desenvolver um software para os ESP32, responsável pelo controle do protótipo;
3. Implementar os algoritmos de reconstrução de superfície;
4. Comparar a eficiência das técnicas de reconstrução de superfície utilizadas.

1.3 ORGANIZAÇÃO DO TRABALHO

O Capítulo 2 aborda a fundamentação teórica, discutindo as tecnologias e teorias utilizadas, como o LIDAR, motores e métodos de reconstrução de superfície. O Capítulo 3 detalha o desenvolvimento do protótipo, incluindo a escolha dos componentes eletrônicos, a montagem das estruturas e a programação dos algoritmos. O Capítulo 4 descreve os resultados obtidos com o protótipo, incluindo uma discussão sobre a eficiência das técnicas de reconstrução de superfície. O Capítulo 5, a conclusão, destaca o impacto da aplicação do LIDAR no mapeamento 3D e aponta melhorias possíveis no sistema.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo é apresentada a Fundamentação Teórica, que serve como alicerce conceitual para a pesquisa realizada. São explorados os principais conceitos, teorias e estudos relacionados ao tema abordado neste trabalho.

2.1 TECNOLOGIAS UTILIZADAS

2.1.1 LIDAR

LIDAR (*Light Detection and Ranging*), é uma tecnologia, baseada em luz, utilizada para medição de distâncias. O LIDAR utiliza um laser para medir a distância até um objeto, e uma montagem óptica-mecânica para apontar o laser com precisão (HARRAP; LATO, 2010). A principal vantagem do LIDAR, dentre outros sensores que também medem a distância, é a sua rapidez e precisão, onde um modelo simples pode coletar até 1000 posições por segundo. Além disto, o sensor tem dois tipos diferentes de medição, *Time-of-Flight* e *Phase-Based*.

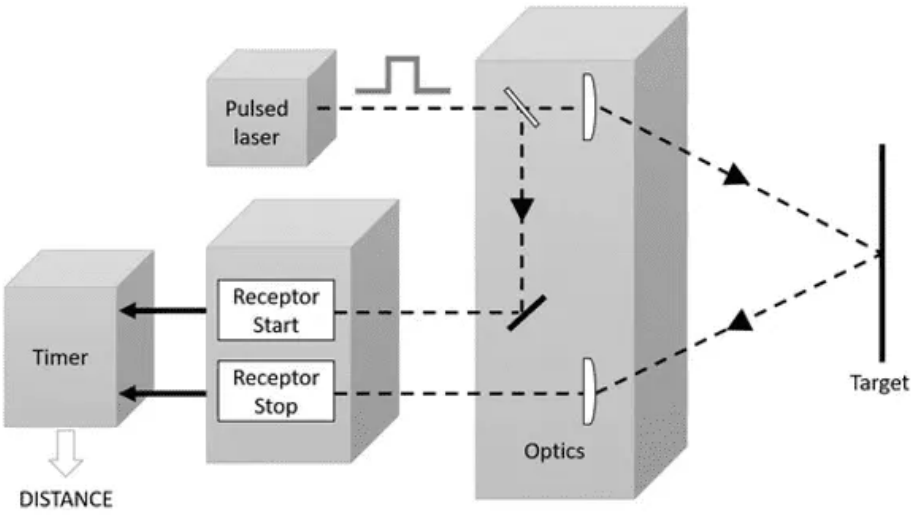
Time-of-Flight consiste em enviar um pulso de laser e medir o tempo de voo que o laser demora para atingir uma superfície e voltar para o sensor óptico. A partir dessa medição é possível calcular a distância que o laser viajou, bem como a distância do LIDAR até a superfície do objeto, através da Equação (1), onde D é a distância entre o LIDAR e uma superfície, c é velocidade da luz, que está sendo dividida por 2 para a distância não ficar duplicada, e ΔT o tempo de voo. Portanto, a distância é proporcional ao tempo que o feixe de laser demorou para percorrer todo o trajeto de viagem. Na Figura 1, está ilustrado um diagrama simplificado do funcionamento do método *Time-of-Flight*.

$$D = \frac{c}{2}\Delta T \tag{1}$$

O LIDAR baseado em fase (*Phase-Based*), o sistema da Figura 2, emprega um laser de forma de onda contínua modulada em amplitude, conhecida como *Amplitude-Modulated Continuous-Wave* (AMCW) (ROYO; BALLESTA-GARCIA, 2019). Ele usa o deslocamento de fase induzido em um sinal periódico de intensidade modulada em sua viagem de ida e volta ao alvo para obter o valor do intervalo. A potência óptica é modulada com uma frequência constante F_M, o valor desta frequência depende do dispositivo, além disso, o formato de onda do laser emitido pode ser senoidal ou quadrado em função da frequência (ROYO; BALLESTA-GARCIA, 2019). Conforme a Equação (2), a distância do LIDAR para uma superfície é calculada a partir da diferença de fase, $\Delta\sigma$, do laser emitido para o recebido.

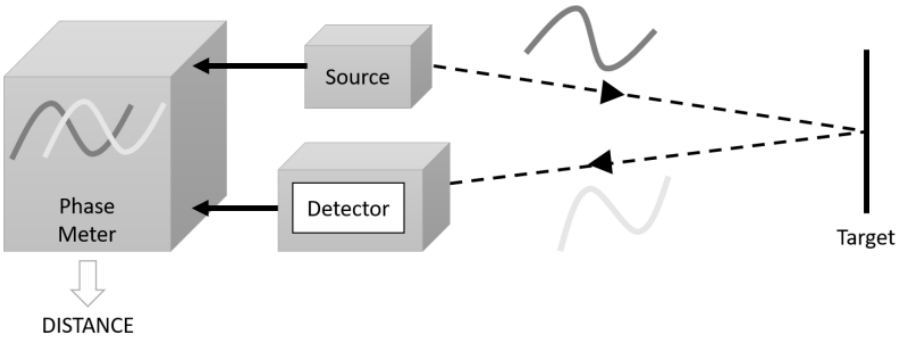
$$D = \frac{c}{2}\frac{\Delta\sigma}{2\pi F_M} \tag{2}$$

Figura 1 – Princípio de medição do *Time-of-Flight* com laser pulsado.



Fonte: (ROYO; BALLESTA-GARCIA, 2019)

Figura 2 – *Phase-Based* princípio de medição.



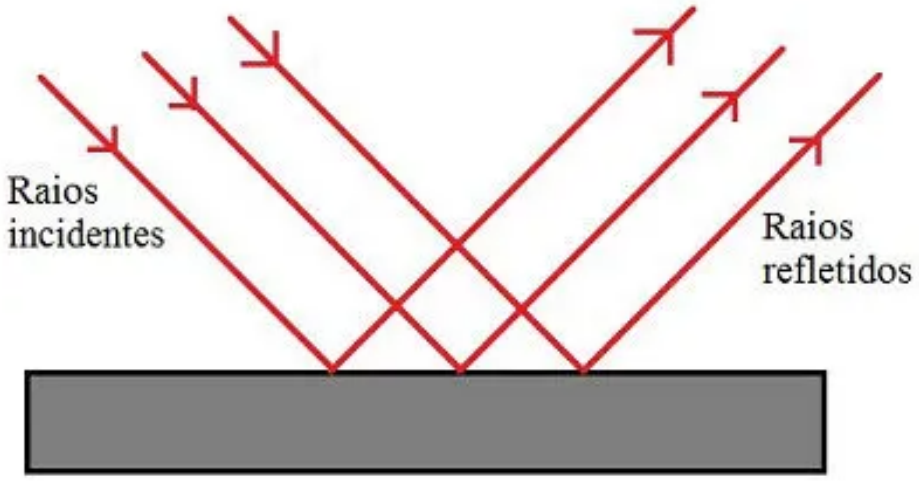
Fonte: (ROYO; BALLESTA-GARCIA, 2019)

2.1.2 Reflexão da Luz

Conforme (HECHT, 2017), quando a luz incide, por um meio de propagação, em um material parte dela é refletida de volta devido à presença de átomos desemparelhados próximos à superfície. A reflexão pode ser especular ou regular, onde a luz é refletida de forma organizada em um ângulo igual ao de incidência conforme Figura 3, ou difusa, onde a luz é dispersa em várias direções conforme Figura 4. A Lei da Reflexão estabelece que o raio incidente, a normal à superfície e o raio refletido estão todos contidos em um plano chamado plano de incidência.

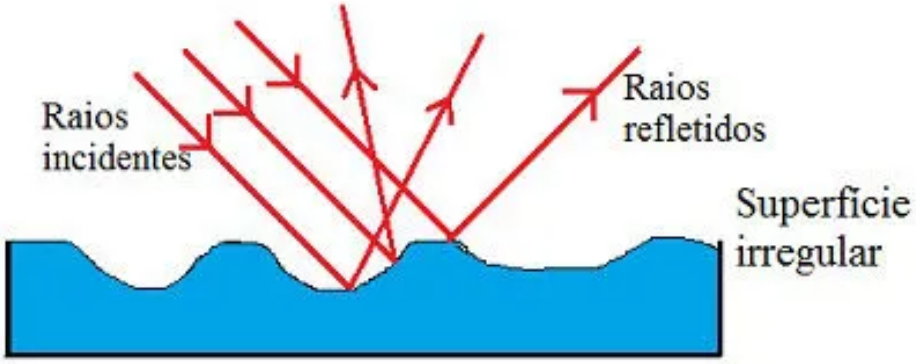
A forma como a luz é refletida depende da uniformidade do meio e do tamanho das irregularidades em relação ao comprimento de onda da luz. Em superfícies lisas, a reflexão é especular e os feixes de luz se combinam de forma coesa, resultando em um único feixe refletido. Por outro lado, em superfícies rugosas, as ondas refletidas podem chegar fora de fase, levando a uma reflexão difusa. A reflexão pode ser observada em

Figura 3 – Reflexão Regular.



Fonte: (MENDES, s.d.)

Figura 4 – Reflexão Difusa.



Fonte: (MENDES, s.d.)

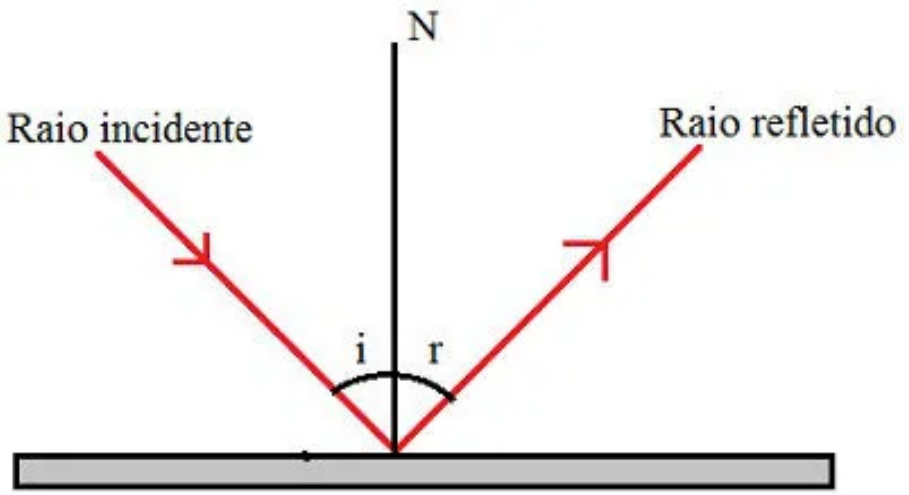
situações do cotidiano, como ao mirar um alvo com um feixe de lanterna refletido em um espelho estacionário.

Como acontece com todos os fenômenos físicos, a reflexão da luz segue certas leis. Para entendê-las melhor, observe a Figura 5. Considere o raio de luz representado na Figura 5, em que:

- i – é o ângulo de incidência.
- r – é o ângulo de reflexão.
- N - é a reta Normal perpendicular à superfície refletora.

Segundo a dedução de (HECHT, 2017), o ângulo de incidência é igual ao ângulo de reflexão,$i = r$, devido à Lei da Reflexão. Essa lei estabelece que, quando um raio de luz incide em uma superfície refletora, o ângulo formado entre o raio incidente e a normal à superfície é igual ao ângulo formado entre o raio refletido e a normal, desde que ambos

Figura 5 – Comportamento do Raio na Reflexão Regular.



Fonte: (MENDES, s.d.)

os raios estejam contidos no mesmo plano. Esse fenômeno ocorre devido à conservação da energia e do momento linear da luz durante o processo de reflexão, resultando na igualdade dos ângulos de incidência e reflexão

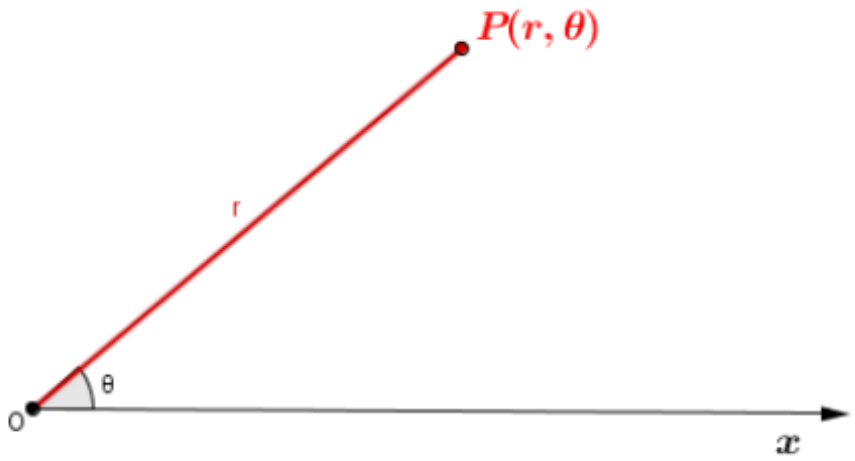
A compreensão da reflexão da luz é essencial em diversas aplicações, desde a fabricação de materiais refletores até o *design* de tecnologias furtivas. A Lei da Reflexão e os princípios de reflexão especular e difusa são fundamentais para o desenvolvimento de dispositivos ópticos e estruturas que manipulam a propagação da luz de maneira controlada. Além disso, a reflexão da luz em interfaces entre diferentes meios desempenha um papel crucial na formação de imagens em espelhos e na transmissão de sinais em fibras ópticas, destacando a importância desse fenômeno na ciência e na tecnologia modernas.

2.1.3 Coordenadas Polares

A história das coordenadas polares remonta aos tempos antigos, com o uso de conceitos de ângulo e raio por povos do primeiro milênio a.C. Hiparco, astrônomo grego, desenvolveu uma tabela de funções de cordas e usou coordenadas polares para determinar posições estelares. A formalização do sistema de coordenadas polares é atribuída a Grégoire de Saint-Vincent e Bonaventura Cavalieri no século XVII, sendo que Cavalieri aplicou-as para resolver problemas relacionados à espiral arquimediana e Blaise Pascal para calcular o comprimento de arcos parabólicos. Isaac Newton explorou as transformações entre coordenadas polares em seu trabalho "*Method of Fluxions*", enquanto Jacob Bernoulli utilizou um sistema com um ponto em uma linha. O termo "coordenadas polares" foi atribuído a Gregorio Fontana e Alexis Clairaut foi o primeiro a pensar em coordenadas polares em três dimensões, com Leonhard Euler sendo o primeiro a desenvolvê-las (COOLIDGE, 1952).

As coordenadas polares servem para descrever um ponto no plano por meio de um angulo, uma direção e amplitude. Para descrever um sistema de coordenadas polares num plano, escolhemos um ponto no plano conhecido como pólo ou origem (0,0) e o denominamos O. Então, desenhamos um raio (semirreta) de O até outro ponto qualquer no plano, denominado P, conforme Figura 6. A distância r formada entre os pontos O e P é denominada raio polar ou raio vetor, e o ângulo θ obtido da rotação do segmento de reta OP até o eixo polar é chamado de ângulo polar ou ângulo vetorial (BAPTISTA, 2017).

Figura 6 – Coordenada Polar.



Fonte: (BAPTISTA, 2017)

2.1.3.1 Conversão de Coordenadas Cartesianas para Coordenas Polares Bidimensionais

Para converter um ponto P no plano cartesiano para coordenadas polares, deve-se sobrepor ambos sistemas de coordenadas de tal forma que o eixo x coincida com o eixo polar e ambas origens coincidam no mesmo ponto. Formando a semirreta OP e projetando P no eixo polar, é possível identificar um triângulo-retângulo, conforme Figura 7. Com isso, basta aplicarmos a já conhecida Lei dos Senos, que obtemos as equações (3) e (4) que descrevem as coordenadas x e y.

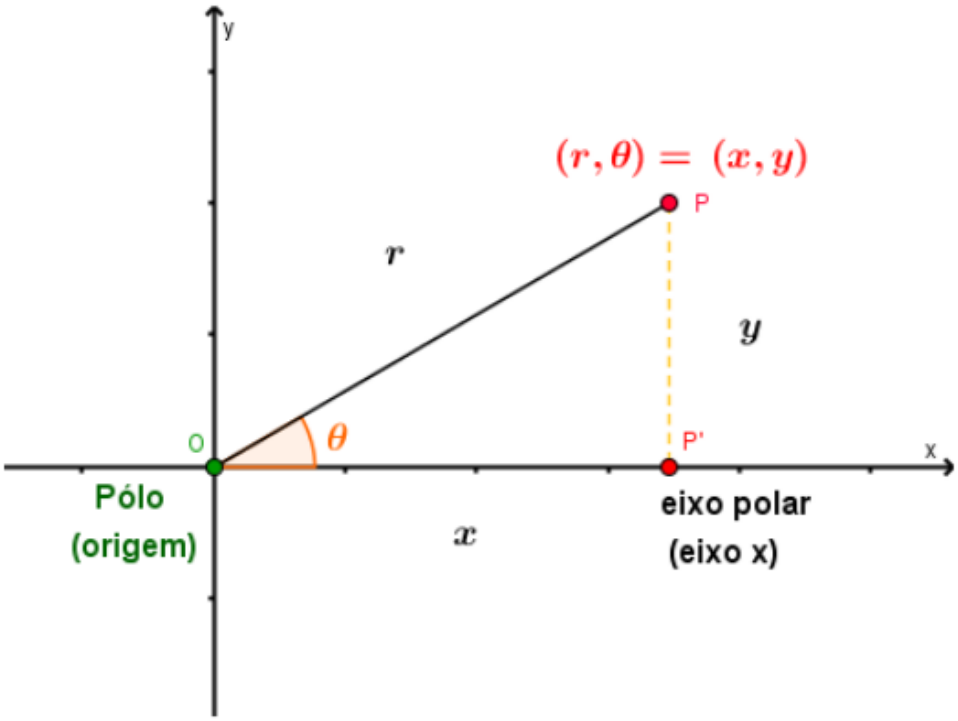
$$x = r.\cos\theta \tag{3}$$

$$y = r.\sin\theta \tag{4}$$

2.1.3.2 Conversão de Coordenadas Cartesianas para Coordenas Polares Tridimensionais

Dado um ponto P qualquer em um espaço tridimensional, conforme Figura 8, é possível descrever sua localização no espaço a partir de $P = (P_x, P_y, P_z)$, onde P_x é a

Figura 7 – Conversão de um Ponto para Coordenadas Polares.



Fonte: (BAPTISTA, 2017)

projeção do ponto P em X, P_y a projeção em Y e P_z a projeção em Z. Também, pode-se descrever sua localização a partir da distância do ponto até a origem, formando uma reta, do ângulo formado entre a reta e o eixo Z e do ângulo formado entre a projeção no plano XY da reta e o eixo X em um espaço tridimensional. Portanto, $P = (\rho, \theta, \alpha)$ onde ρ é a distância da reta OP, θ o angulo formado a partir do eixo X e α o angulo formado a partir do eixo Z. Com isso, as projeções desse ponto podem ser descritas como:

$$P_x = \rho . \sin \alpha . \cos \theta \tag{5}$$

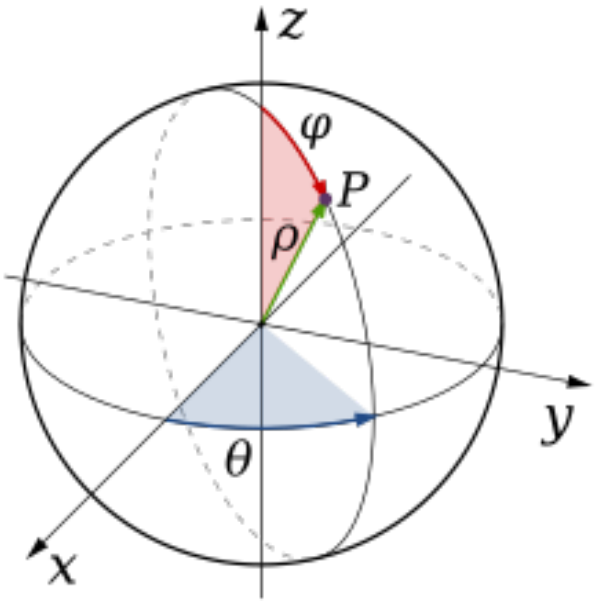
$$P_y = \rho . \sin \alpha . \sin \theta \tag{6}$$

$$P_z = \rho . \cos \alpha \tag{7}$$

2.1.4 ESP32

O ESP32 é uma plataforma poderosa e acessível para o desenvolvimento de aplicações IoT e outros sistemas. Foi concebido pela empresa chinesa *Espressif Systems Company*, compreendendo uma placa microcontroladora e um ambiente de desenvolvimento de software (IDE) usado para escrever e carregar o algoritmo na placa microcontroladora. Esta plataforma oferece uma variedade de ferramentas para comunicação sem fio, incluindo interfaces integradas na placa e diversas bibliotecas. Esta placa possui diversos pinos GPIO

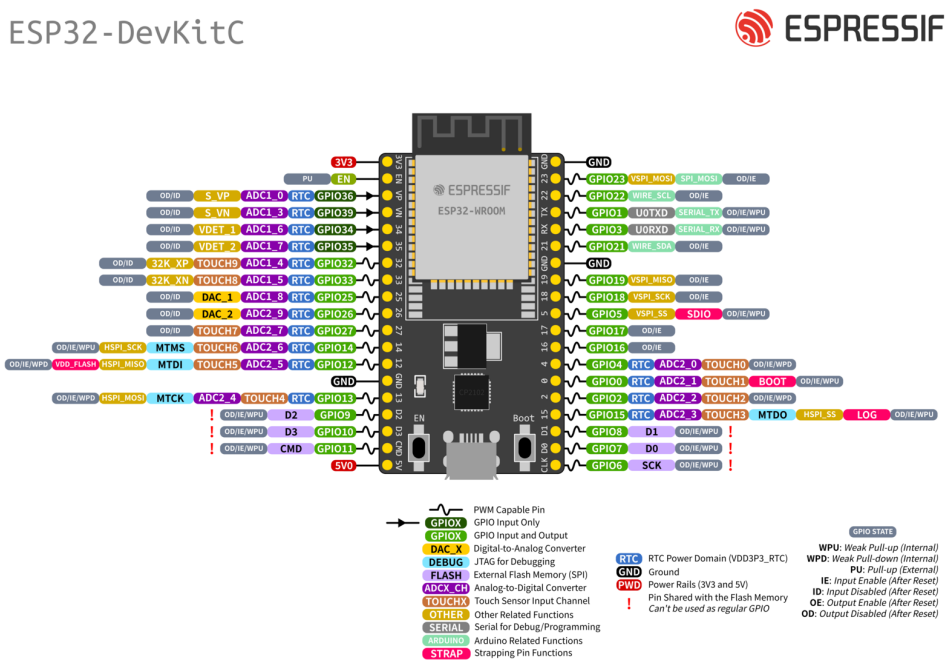
Figura 8 – Conversão de um Ponto para Coordenadas Polares Tridimensionais.



Fonte: (WIKIPEDIA, s.d.)

que permitem a comunicação e o controle de vários dispositivos e sensores, suportando uma ampla gama de interfaces, conforme pode ser observado na Figura 9 que representa sua pinagem. Com todas essas funcionalidades somado ao seu baixo custo, esta plataforma é uma boa opção para estudos e desenvolvimento de protótipos.

Figura 9 – Pinagem do ESP32.



Fonte: (SYSTEMS, s.d.[b])

O ESP32 é dotado de um processador *dual-core*, proporcionando uma capacidade

de processamento elevada que facilita a execução de múltiplas tarefas e o torna eficiente para atividades complexas. Ele possui tecnologia de gerenciamento de energia, que faz com que ele opere com baixo consumo de energia, aumentando sua eficiência quanto isso, tornando-o adequado para projetos alimentados por bateria ou com restrição de energia. Segue abaixo as principais características da ESP32 (SYSTEMS, s.d.[a]):

- Xtensa® *single-/dual-core 32-bit* LX6 microprocessador(s), com capacidade de processamento de até 240 MHz.
- 520 KB de SRAM.
- 16 MB de memória *flash*.
- *Wi-Fi* 802.11 b/g/n.
- *Bluetooth* v4.2 BR/EDR e *Bluetooth* LE.
- 34 pinos GPIO programáveis.
- 17 canais conversor de Analógico-para-Digital.
- 2 canais conversor de Digital-para-Analógico.
- 4 SPI, 3 UART, 2 I2C.
- Tecnologia de baixo consumo de energia TSMC 40 nm.

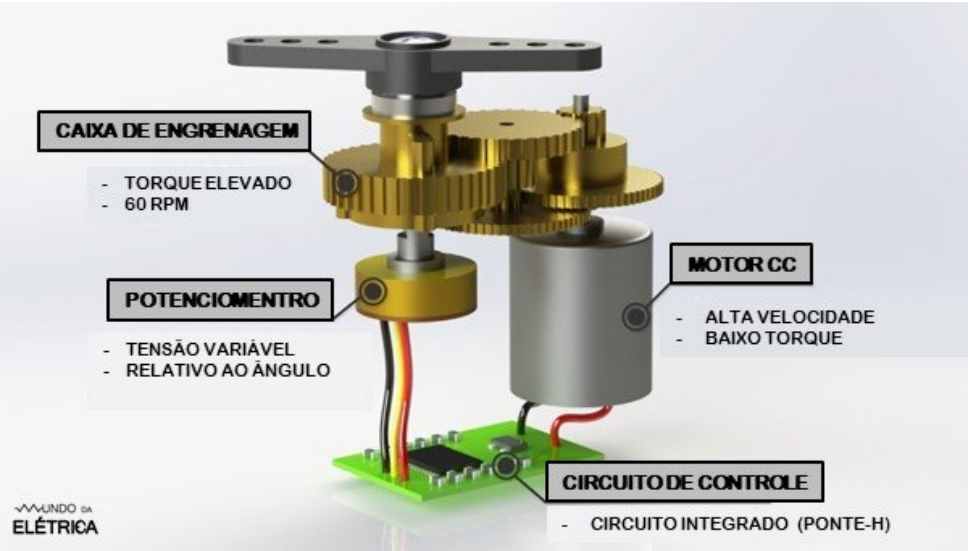
A plataforma ESP fornece flexibilidade para sua programação, podendo ser programado em diversas linguagens e usando diversos *frameworks*. A linguagem mais comumente usada para sua programação é o C++, no qual pode ser desenvolvida usando a IDE Arduino ou PlatformIO. Uma característica bastante positiva dessa plataforma é sua ativa comunidade de usuários. Existem vários recursos online disponíveis que fornecem tutoriais, exemplos de código e ideias de projetos. Isso torna mais fácil para iniciantes começarem a usar o ESP e para usuários mais avançados colaborarem e compartilharem seus trabalhos.

2.1.5 Servo Motor

O servo motor é utilizado em diversas aplicações em que se deseja ter uma precisão e um movimento controlado. Uma característica muito útil desse motor é se movimentar a uma determinada posição e se manter nela mesmo quando exercida uma força na direção oposta. Os servos motores são dispositivos eletro-mecânicos compostos por 4 componentes: um motor DC, engrenagens de redução, circuito de controle e um potenciômetro. Sua estrutura pode ser visualizada na Figura 10. O circuito de controle trabalha juntamento com o potenciômetro para formar um sistema de *feedback*. Com isso é possível controlar de forma precisa a rotação do eixo do motor. Os servos motores mais populares tem seu ângulo limitado entre 0^{o} e 180, onde a variação angular do eixo nessa faixa é definido pela largura de pulso de um sinal, conforme será explicado a seguir.

O controle das posições do eixo do servo motor é feito por meio de um sinal modulado por largura de pulso, o PWM (*Pulse Width Modulation*). Tipicamente, o circuito

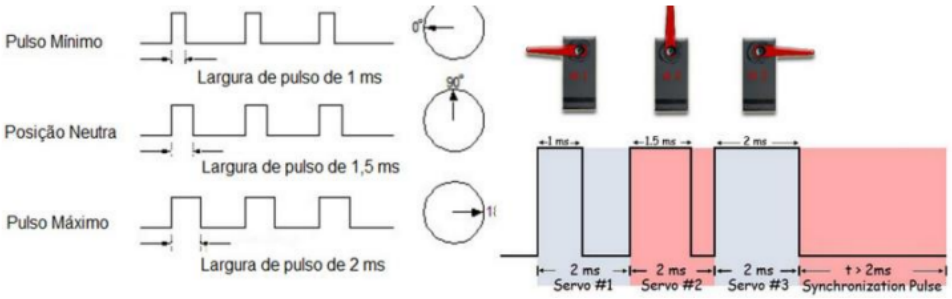
Figura 10 – Componentes do servo motor.



Fonte: (MATTEDE, s.d.)

de controle monitora o sinal de entrada a cada período de 20 ms, onde é detectada a alteração da largura do sinal. Essas alterações representam a rotação do eixo para uma determinada posição. A largura do pulso é relacionada diretamente com o ângulo em que o eixo do servo motor se posicionará, conforme Figura 11.

Figura 11 – Movimentação do servo motor.



Fonte: (EVANS; NOBLE; HOCHENBAUM, 2013)

2.1.6 Motor de Passo

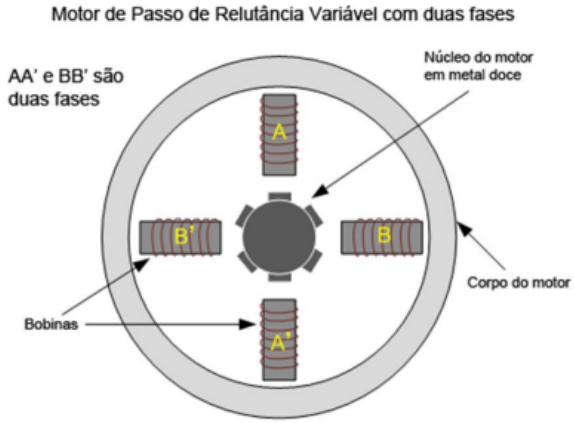
Os Motores de Passo são dispositivos eletro-mecânicos síncronos que convertem pulsos elétricos em movimentos que causam variações angulares discretas. A operação desse motor é baseado na interação entre um campo magnético e corrente elétrica. O rotor desses motores pode rotacionar em pequenos ângulos, denominado passo, quando pulsos elétricos são aplicados a uma determinada frequência nos terminais do motor (BRITES FELIPE GONÇALVES E SANTOS, 2008). O motor de passo é utilizado em diversas aplicações em que se deseja ter precisão e movimento controlados, sendo essas características sua

maior vantagem. Por conta disto, este motor é muito utilizado em impressoras, robôs, automações industriais, scanners, entre outros. Há três tipos de estruturas para esse motor, Relutância Variável, Ímã Permanente e Híbrido, conforme será visto nas próximas seções.

2.1.6.1 Relutância Variável

Este tipo de motor consiste em um eixo feito de ferro, com múltiplos dentes e de 3 a 5 bobinas conectados a um terminal comum. Quando as bobinas são energizados com corrente DC os mesmos formam um campo magnético. A rotação ocorre quando os dentes do rotor são atraídos para os polos do estator energizado, devido à força que aparece, para que o sistema tenha o circuito com menor relutância, tendendo a alinhar o eixo (BRITES FELIPE GONÇALVES E SANTOS, 2008). A estrutura deste tipo de motor pode ser observada na Figura 12.

Figura 12 – Motor de Relutância Variável.



Fonte: (AGNIHOTRI, s.d.)

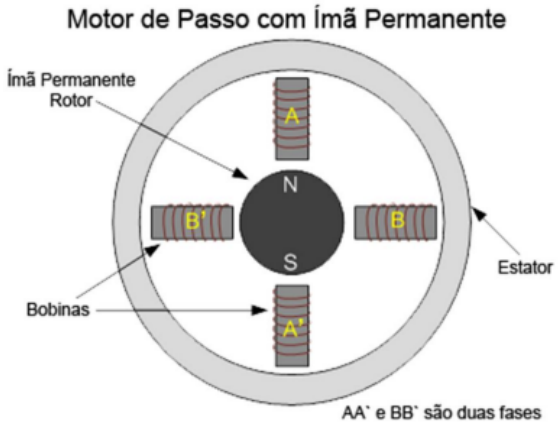
2.1.6.2 Imã Permanente

Este tipo, ilustrado pela Figura 13, possui um ímã permanente em um eixo liso, gerando uma mecânica simples e barata com baixa resolução no angulo mínimo de rotação. A vantagem deste tipo, é que o fato dele possuir um campo magnético permanente que se soma ao campo magnético das bobinas, resultando em um torque maior na partida. Quando uma bobina do estator é ativada, o eixo se alinha com o campo magnético até o estator ser desligado e o estator seguinte ligado.

2.1.6.3 Híbrido

O motor de passo híbrido é mais sofisticado do que o de ímã permanente, mas provém melhor desempenho em comparação com a resolução de passo, torque e velocidade. Ângulos de passo típico de motores híbridos estão entre 3,6º a 0,9º (100-400 passos

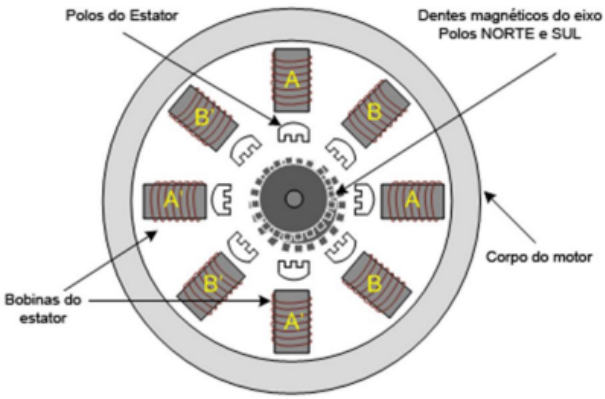
Figura 13 – Motor de Imã Permanente.



Fonte: (AGNIHOTRI, s.d.)

por volta). O motor híbrido combina as melhores características dos motores de ímã permanente e motor de relutância variável. O rotor é multi-dentado como no motor de relutância variável e contem um ímã permanente ao redor do seu eixo, como pode ser visto na Figura 14.

Figura 14 – Motor Híbrido.



Fonte: (AGNIHOTRI, s.d.)

2.1.7 *Real-Time Operationg System*

Real-Time Operationg System (RTOS) são projetados para lidar com tarefas que têm requisitos temporais estritos, garantindo que as operações sejam concluídas dentro de prazos específicos. Eles são essenciais em aplicações onde a resposta rápida e previsível é crucial, como em sistemas embarcados, controle de processos industriais e sistemas críticos de segurança. Os RTOS são caracterizados por sua capacidade de responder a eventos em tempo real, escalonar tarefas com base em prioridades e garantir a execução de tarefas críticas dentro de limites de tempo determinados.

Um dos principais conceitos dos sistemas operacionais em tempo real é a divisão de tarefas em estados como executando, pronto, suspenso e dormente, permitindo um gerenciamento eficiente do tempo de CPU e recursos do sistema. Além disso, os RTOS geralmente empregam algoritmos de escalonamento de tarefas, como *round-robin* ou prioridade preemptiva, para garantir que as tarefas mais importantes sejam executadas no momento certo. A capacidade de lidar com interrupções de *hardware* de forma eficiente e previsível também é uma característica fundamental dos RTOS, garantindo que eventos críticos sejam tratados sem atrasos indesejados (LAPLANTE; OVASKA, 2012).

Outra característica importante dos sistemas operacionais em tempo real é a capacidade de garantir a previsibilidade e determinismo das operações. Isso significa que os RTOS são projetados para fornecer tempos de resposta consistentes e confiáveis, permitindo que os desenvolvedores prevejam o comportamento do sistema em diferentes cenários. Além disso, os RTOS geralmente oferecem mecanismos de sincronização e comunicação entre tarefas, garantindo a integridade dos dados e a coordenação eficiente das operações. Em resumo, os sistemas operacionais em tempo real desempenham um papel crucial em ambientes onde a precisão temporal e a confiabilidade são essenciais, fornecendo uma base sólida para o desenvolvimento de sistemas críticos e responsivos (LAPLANTE; OVASKA, 2012).

2.1.7.1 FreeRTOS

Desenvolvido em parceria com as principais empresas de chips do mundo ao longo de um período de 18 anos, o *FreeRTOS* é um sistema operacional de tempo real líder de mercado para microcontroladores e pequenos microprocessadores. Distribuído gratuitamente sob a licença de código aberto MIT, o *FreeRTOS* inclui um *kernel* e um conjunto crescente de bibliotecas adequadas para uso em todos os setores da indústria (FREERTOS, s.d.). Esse sistema, devido sua ampla comunidade engajada, possui diversos exemplos de aplicações para diferentes cenários. Possui uma baixa sobrecarga de ROM, RAM e processamento, uma imagem do *Kernel FreeRTOS* possui cerca de 6kb a 12kb, e seu núcleo do kernel é contido em apenas 3 arquivos C. Tudo isso o torna simples e de fácil uso, sendo uma ferramento bastante usada para desenvolvedores da área e também por iniciantes.

2.1.8 Reconstrução de Superfície

Reconstrução de superfície é o processo de criar um modelo tridimensional contínuo a partir de um conjunto de pontos discretos coletados no espaço, geralmente por meio de técnicas de escaneamento 3D, como varredura a laser ou fotogrametria. Esses pontos representam as coordenadas na superfície do objeto e ao serem processados por algoritmos de reconstrução de superfície, são conectados para formar uma malha ou uma superfície suave que aproxima a geometria original. Essa técnica é amplamente utilizada

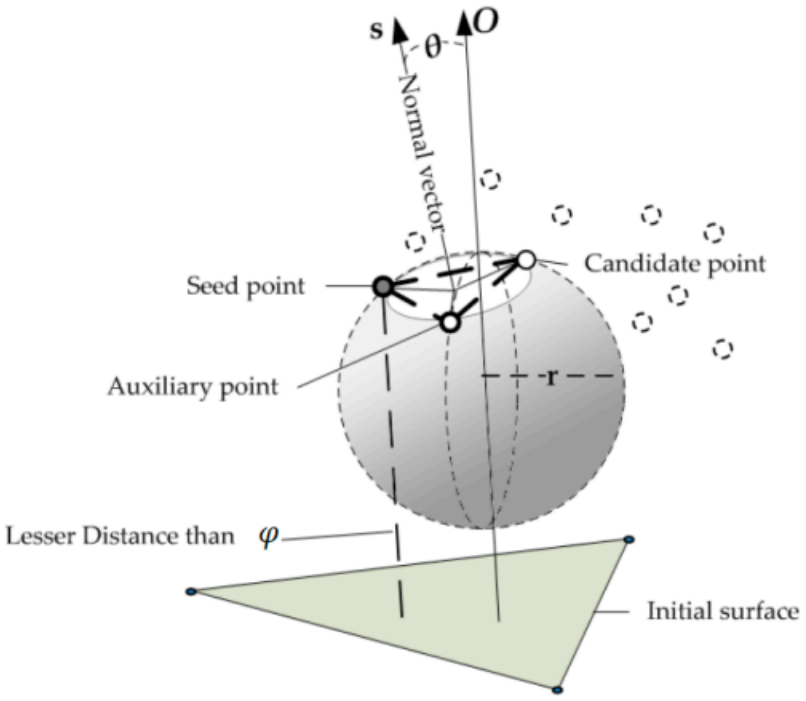
em áreas como engenharia reversa, computação gráfica, arqueologia e medicina para criar representações precisas e detalhadas de objetos ou ambientes reais.

2.1.8.1 Ball-Pivonting Algorithm

O objetivo do algoritmo *Ball-Pivonting Algorithm* (BPA), criado por (BERNARDINI *et al.*, 1999), é reconstruir uma representação de superfície digital a partir de um conjunto desorganizado de pontos em um plano tridimensional, onde os pontos não possuem nenhuma informação de conectividade explícita, geralmente obtidos a partir de um escâner 3D ou outras técnicas de detecção. O algoritmo assume que os pontos representam a superfície de um objeto e visa conectá-los em uma malha, formando triângulos que se aproximam da superfície do objeto.

A partir de uma nuvem de pontos tridimensional, um ponto de início é selecionado aleatóriamente. A partir deste ponto, o algoritmo traça uma esfera de raio r entorno dele, em busca de pontos vizinhos. Encontrados dois pontos a partir do ponto inicial, é verificado condições geométricas, como restrições de ângulos ou limites de curvatura, para garantir que o triângulo formado seja liso e aderente à superfície, conforme Figura 15. A partir desse triângulo formado, a esfera se movimenta pelos vértices em busca novos pontos para continuar formando triângulos e adicionando-os à rede de pontos.

Figura 15 – Técnica *Ball-Pivonting*.



Fonte: (MA; LI, 2019)

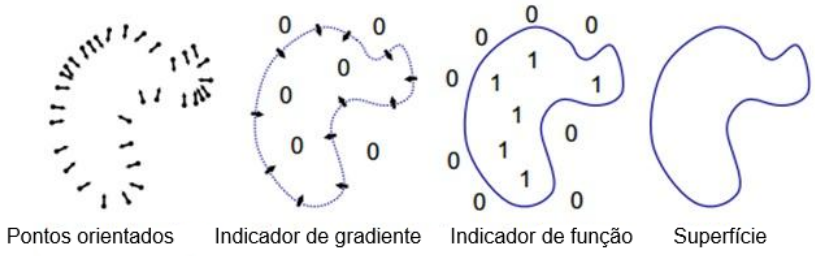
É importante observar que a eficiência e a precisão do BPA podem ser influenciadas por vários fatores, como a escolha da estrutura de dados espaciais, o raio da vizinhança

e as restrições geométricas. O ajuste fino desses parâmetros geralmente é necessário para obter resultados ideais para um conjunto de pontos específicos.

2.1.8.2 Reconstrução de Superfície por Poisson

O algoritmo de reconstrução de superfície pelo método Poisson (KAZHDAN; HOPPE, 2013) é uma técnica bem conhecida para criar superfícies impermeáveis a partir de amostras de pontos orientados adquiridos com *scanners* 3D. A técnica funciona da seguinte maneira: primeiro, é criada uma grade de pontos a partir dos dados de entrada. Em seguida, é gerada uma função de campo escalar que representa a distância dos pontos da grade à superfície implícita. Essa função é obtida resolvendo a equação de Poisson com condições de contorno apropriadas. Por fim, a superfície implícita é extraída a partir da função de campo escalar usando um algoritmo de extração de contorno (KAZHDAN; HOPPE, 2013). Todo esse processo é resumido na Figura 16.

Figura 16 – Ilustração intuitiva da reconstrução por Poisson em 2D.



Fonte: (KAZHDAN; HOPPE, 2013)

Uma das principais vantagens do método Poisson é sua capacidade de lidar com dados ruidosos e incompletos. Além disso, o método é capaz de criar superfícies suaves e contínuas, mesmo em regiões com dados faltantes. No entanto, o método também tem algumas limitações, como a tendência de suavizar demais os dados e a dificuldade de lidar com amostras de pontos com densidades variáveis. Para superar essas limitações, pesquisadores têm explorado modificações no algoritmo original, como a incorporação de restrições de posição ou a utilização de técnicas de filtragem adaptativa.

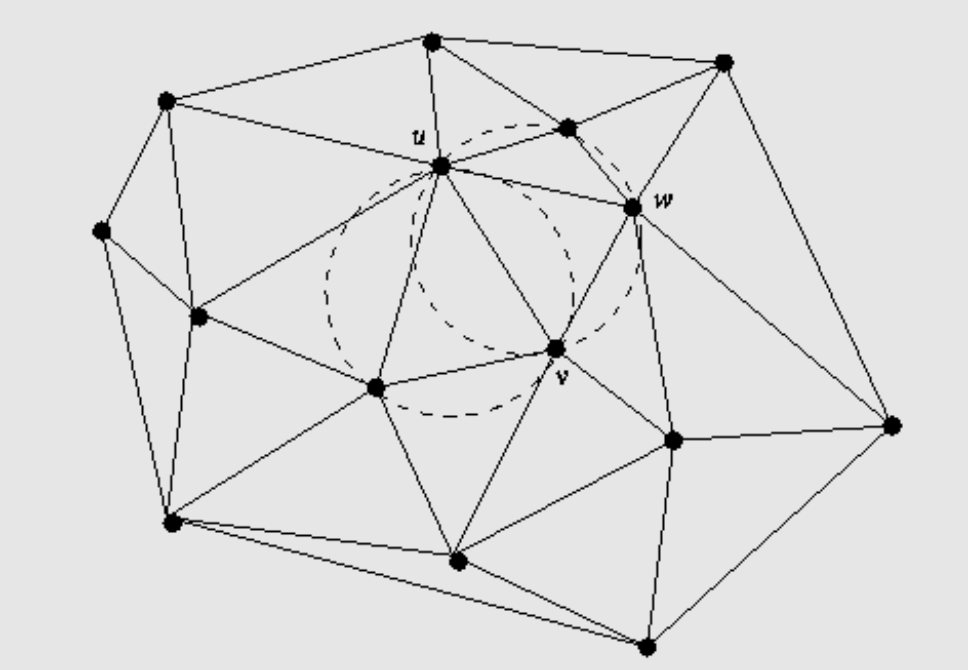
2.1.8.3 Triangulação de Delaunay

A teoria da Triangulação de Delaunay é um conceito fundamental na geometria computacional e na análise de estruturas de dados espaciais. Essa teoria descreve um método para dividir um conjunto de pontos no espaço de um conjunto de triângulos não sobrepostos, conhecidos como a triangulação de Delaunay. A principal característica desta triangulação é que ela maximiza os ângulos mínimos dos triângulos, o que a torna ideal

para aplicações como interpolação, geração de malhas em computação gráfica e análise espacial.

O funcionamento da Triangulação de Delaunay se dá a partir de uma nuvem de pontos no espaço, se inicia incluindo um grande triangulo externo englobando todos os pontos. A partir disto, são formados triângulos a partir dos pontos analisados pelo método de triangulação. Na triangulação, os triângulos formados precisam satisfazer a propriedade da circunferência vazia, onde o triangulo circunscrito não pode ter pontos soltos dentro da circunferência. Conforme exemplo na Figura 17, o círculo que contem os pontos u, v e w não satisfazem, pois possui 4 pontos dentro do círculo.

Figura 17 – Processo de Triangulação.



Fonte: (GUEDES, s.d.)

2.1.8.4 Reconstrução de Superfície por A-Shapes

O "*a-shape*" é definido como um grafo de linha reta que captura a forma do conjunto de pontos, considerando diferente valores de "a". Ele fornece uma representação da forma do conjunto de pontos que pode variar de formas mais finas a formas mais grossas, dependendo do valor de "a". Essa noção é derivada de uma generalização direta de uma definição comum da envoltória convexa (EDELSBRUNNER; KIRKPATRICK; SEIDEL, 1983). A envoltória convexa de um conjunto de pontos em um espaço euclidiano, como o plano, é o menor conjunto convexo que contém todos os pontos desse conjunto. Em outras palavras, a envoltória convexa é a interseção de todos os conjuntos convexos que contêm os pontos originais.

O algoritmo para a construção dos "*A-Shapes*" descrito em (EDELSBRUNNER; KIRKPATRICK; SEIDEL, 1983) funciona da seguinte maneira:

1. Construção da Triangulação de Delaunay:
    - Dependendo do valor de "a" (parâmetro real), é construída a Triangulação de Delaunay dos pontos do conjunto.
2. Determinação dos Pontos Extremos "a":
    - Utilizando as informações da Triangulação de Delaunay, são determinados os pontos extremos "a" do conjunto de pontos.
3. Determinação dos Vizinhos "a":
    - Com base na Triangulação de Delaunay, são identificados os vizinhos "a" de cada ponto do conjunto.
4. Saída do "*a-shape*":
    - O algoritmo gera o grafo dos pontos extremos "a" com todas as conexões de vizinhança "a".

O algoritmo de (EDELSBRUNNER; KIRKPATRICK; SEIDEL, 1983) se baseia nas propriedades da Triangulação de Delaunay para identificar os pontos extremos e seus vizinhos conforme o parâmetro "a". A eficiência do algoritmo é garantida pela estrutura da Triangulação de Delaunay e pela caracterização dos vizinhos "a", permitindo a construção rápida e precisa dos "*A-Shapes*" para diferentes valores de "a".

Essa abordagem algorítmica eficiente e teoricamente fundamentada torna os "*A-Shapes*" uma ferramenta poderosa para a análise da forma de conjuntos de pontos no plano, com aplicações em diversas áreas, como reconhecimento de padrões, visualização de dados e processamento de imagens.

2.2 TRABALHOS RELACIONADOS

Rahim (N.A. RAHIM, 2021) propõe utilizar um LIDAR 2D capaz de rotacionar 360º em seu eixo, coletando todos os pontos ao seu arredor, com um robô móvel. O robô móvel é controlado por um aplicativo via *Bluetooth*, para isso foi usado um Arduino em conjunto com um módulo que suporte essa comunicação. O LIDAR fica em cima do robô coletando pontos do ambiente, que são enviados diretamente para um computador. Utilizou-se o *software* MatLAB para renderização de uma imagem 3D. O autor testou o projeto em um quarto de 4 m x 4,5 m e comparou as distâncias dos pontos a partir do robô, coletados pelo LIDAR, com a distância real, obtendo uma acurácia de 97,8%.

Cheng (CHENG; WANG, 2018) realizou um projeto com um robô com rodas que escaneia o ambiente e gera um mapa de pontos em tempo real, visando identificar obstáculos e determinar uma trajetória alternativa com base nos pontos escaneados. Para isto, o autor utilizou um LIDAR que rotaciona em 360º, um Arduino conectado a uma *shied*

driver para controlar o motor do robô e um computador implementando um algoritmo de geração do mapa de pontos coletados pelo LIDAR. Os algoritmos para geração de pontos são o *Gmapping Slam* (GRISETTI; STACHNISS; BURGARD, 2007) e o *Hector Slam* (KOHLBRECHER *et al.*, 2011). O *Gmapping* é um algoritmo para criar um mapa 2D do ambiente e descrever a posição do robô através do LIDAR e um hodômetro. O *Hector Slam* desempenha uma função similar, mas não precisa do hodômetro para executá-la. Além disso, o autor utilizou o algoritmo *A-star* para definir a trajetória mais curta para o robô seguir, considerando os obstáculos escaneados. Com isso, atingiu-se resultados satisfatórios para ambos algoritmos de geração de mapa de pontos. O Hector tem a vantagem de não precisar de hodômetro, mas precisa de uma maior amostragem de pontos do LIDAR em comparação com o *Gmapping*.

Denysyuk (DENYSYUK; TESLYUK; CHORNA, 2018) projetou um robô integrado ao LIDAR e uma interface interativa em Java para realizar comandos com base no mapa de pontos 2D. O robô é controlado por um Arduino que também fica responsável por transferir os dados do LIDAR e de um giroscópio, utilizado para medir à angulação da direção do LIDAR. No computador, uma aplicação em Java recebe os dados e os processam para um mapa de pontos, além de retornar comandos para o Arduino. A aplicação em Java foi implementada com a arquitetura *Model-View-Control* (MVC), onde a codificação da inicialização do programa com a comunicação e processamento de dados foram separados da codificação da construção da interface e do controle do sistema. Para encontrar o caminho, considerando os obstáculos encontrados, os autores utilizaram teoria de grafos para calculá-lo. Utilizaram coordenadas polares, ou seja, a distância lida pelo LIDAR e o ângulo lido pelo giroscópio, para montar o grafo.

Grewal (GREWAL *et al.*, 2017) propôs criar uma cadeira de rodas autônoma utilizando um LIDAR de rotação em 360º e um sensor de proximidade para mapear o ambiente e detectar obstáculos em tempo real. O LIDAR fica no topo da cadeira, numa haste que fica acima da cabeça do paciente, e o sensor de proximidade fica mais abaixo da cadeira para identificar possíveis obstáculos no ponto cego do LIDAR. O autor integrou esses sensores e mais um odômetro com um Arduino, também usado para controlar o motor da cadeira. Ainda, o Arduino se comunica com um computador que executa uma aplicação com o *framework Robot Operating System* (ROS). Esse framework disponibiliza bibliotecas auxiliares como a *slam gmapping* responsável pela criação do mapa de pontos e a *move base* que realiza cálculo de rotas. Com isso, o autor fez os testes em dois ambientes diferentes, onde não houve colisão em nenhum e apenas um erro de navegação em um deles.

3 PROPOSTA E IMPLEMENTAÇÃO DE UM DISPOSITIVO PARA MAPEAMENTO 3D

Nesse capítulo é apresentada uma proposta de um dispositivo para mapeamento 3D, que visa utilizar um protótipo equipado com um sensor LIDAR para coletar pontos na superfície de um ambiente e processar esses dados a fim de gerar uma imagem 3D aproximada das dimensões reais do local. A arquitetura da solução envolve a integração do protótipo com algoritmos em um computador, destacando a importância do LIDAR e dos sensores de escaneamento a LASER. Além disso, são discutidos métodos de reconstrução de imagem, como *Ball Pivoting*, *A-Shapes* e Reconstrução de Superfície por Poisson, utilizando a biblioteca *python Open 3D* para suporte a esses métodos.

3.1 DESCRIÇÃO DO PROBLEMA

Deseja-se realizar o mapeamento de um ambiente por meio de pontos coletados em sua superfície por um protótipo que utiliza um sensor LIDAR e processar esses dados para criar uma imagem 3D. Para tal, foi projetado um protótipo que, com o LIDAR, fornece os dados necessários para serem convertidos em coordenadas cartesianos tridimensionais (x, y, z). Com as coordenadas coletadas, e adotado um método de reconstrução de superfície partir dessa nuvem de pontos que represente de forma aproximada as dimensões reais do ambiente mapeado.

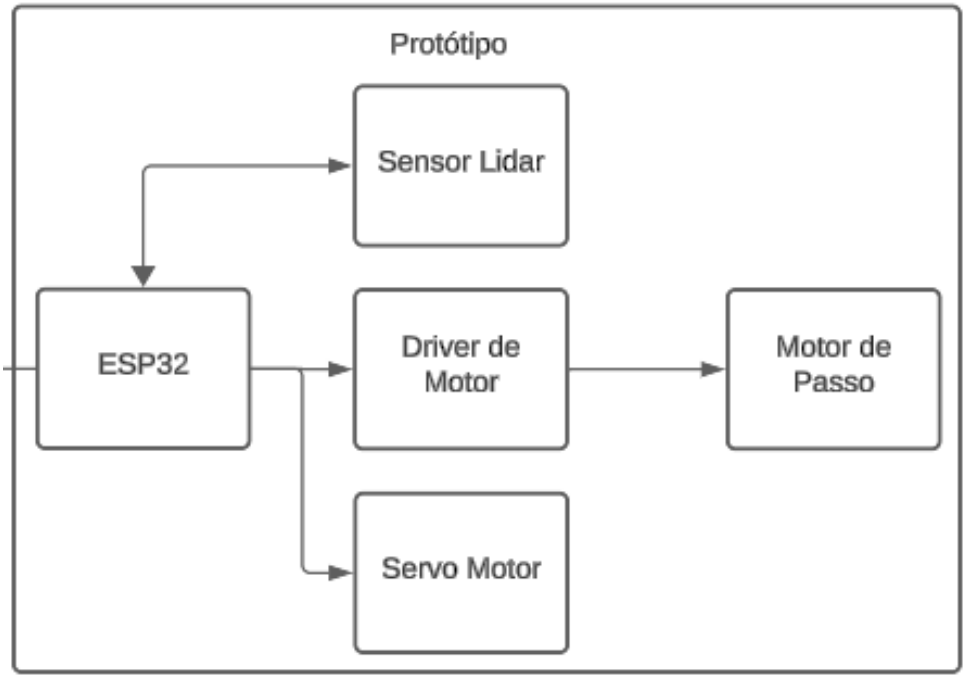
3.2 ARQUITETURA DA SOLUÇÃO

Foi planejado um sistema que consiste na integração de um protótipo com algoritmos rodando em um computador. O protótipo constitui de uma placa microcontroladora que se comunica com os outros dispositivos de apoio conforme à direita da Figura 18. Sobre os algoritmos rodando em um computador, um recebe e processa os dados do protótipo, nomeado de Leitura Serial, e o outro converte os dados em imagem 3D, nomeado de Algoritmo de Renderização de Imagem, como pode ser visto na Figura 19. A figura do diagrama completo pode ser visto no Anexo A, onde pode se observar que o ESP32 também se comunica com um computador por meio de um cabo USB. O detalhamento das duas partes da arquitetura do sistema são discorridas nas subseções seguintes.

3.3 O PROTÓTIPO

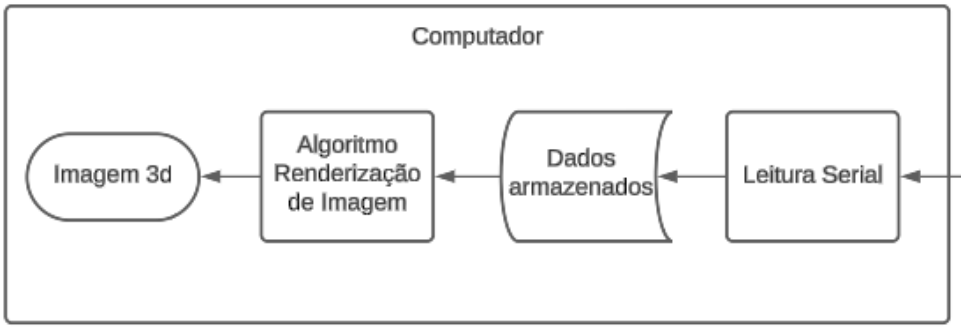
O protótipo é composto pela integração de diversas estruturas mecânicas produzidas por meio de uma impressora 3D, as quais desempenham o papel de suporte para os componentes eletrônicos. A estrutura principal, Figura 20, abriga um motor de passo, enquanto outra estrutura móvel, Figura 21, sustenta um sensor LIDAR e um micro servo motor. Uma polia conecta o eixo do motor de passo ao eixo central e inferior, Figura

Figura 18 – Diagrama do Protótipo.



Fonte: do Autor.

Figura 19 – Diagrama do Processamento da Imagem.

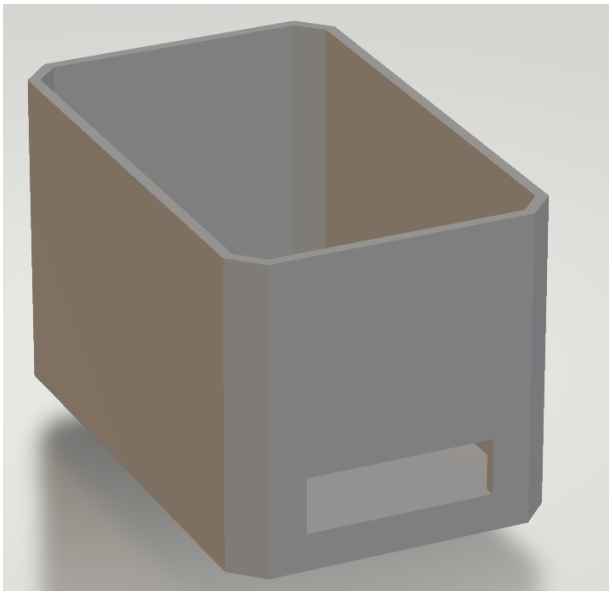


Fonte: do Autor.

22, da estrutura móvel. Nesse contexto, o motor de passo é responsável pelo movimento horizontal do sensor, o micro servo motor encarrega-se do deslocamento vertical do LIDAR, e o próprio sensor realiza a medição da distância entre ele e qualquer superfície em sua direção. O protótipo montado ficou conforme Figura 23.

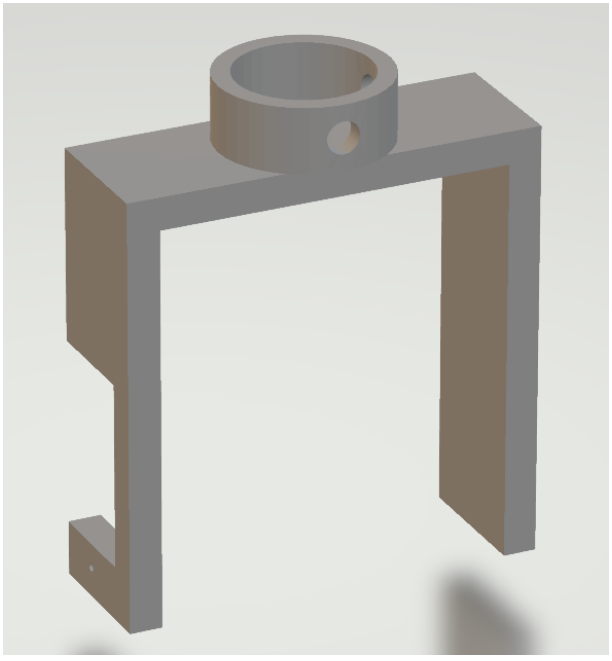
O motor de passo selecionado para este projeto é o Nema 17, amplamente empregado em aplicações de automação e robótica devido sua confiabilidade e podendo operar com um baixo passo. Possui um ângulo de passo padrão de 1,8 graus, ajustável para valores inferiores por meio de drivers como o A4988, capaz de reduzir até 1/16 do passo padrão. Com uma faixa de tensão nominal entre 12V e 24V, corrente nominal variando de 1,2 a 2,5 amperes por fase e um torque máximo de 0,4 Nm (Newtons-metro), dependendo da

Figura 20 – Estrutura Principal 3D.



Fonte: do Autor.

Figura 21 – Estrutura Móvel 3D.

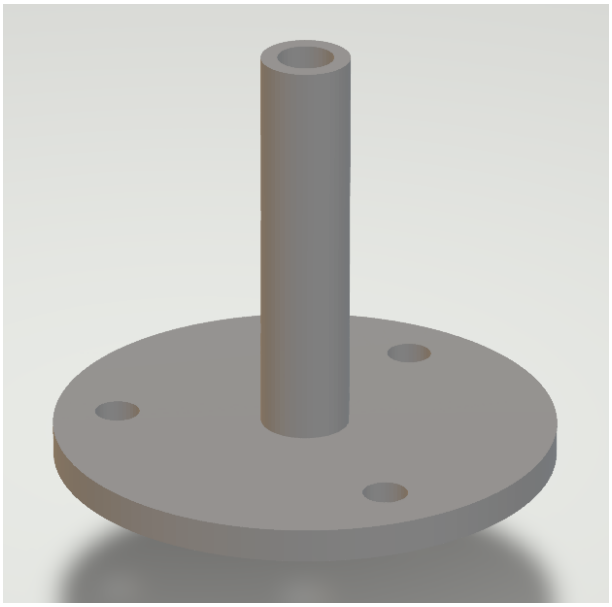


Fonte: do Autor.

corrente aplicada, o Nema 17 destaca-se por sua precisão, capacidade de posicionamento e torque suficiente para movimentar a estrutura móvel do projeto em questão, sendo a escolha ideal para essa aplicação específica.

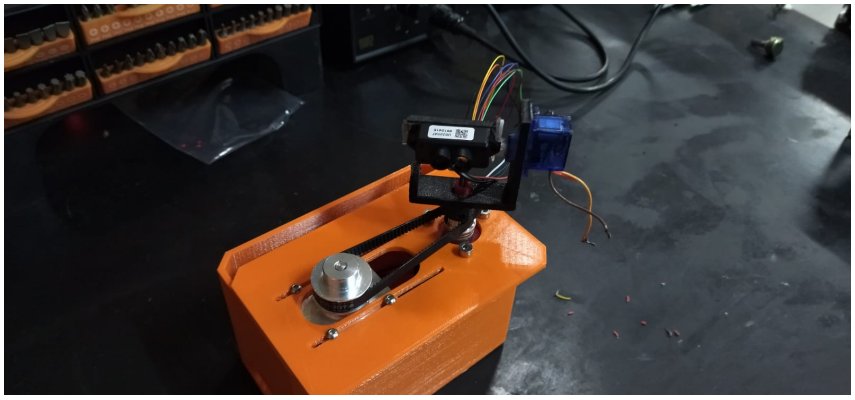
Conforme mencionado anteriormente, o motor de passo desempenha a função crucial de girar a estrutura móvel no eixo vertical, e, com o suporte de um driver, a resolução do motor foi ajustada para uma rotação de 360º em 800 pulsos elétricos. Teoricamente,

Figura 22 – Eixo Inferior da Estrutura Móvel 3D.



Fonte: do Autor.

Figura 23 – Protótipo Montado.



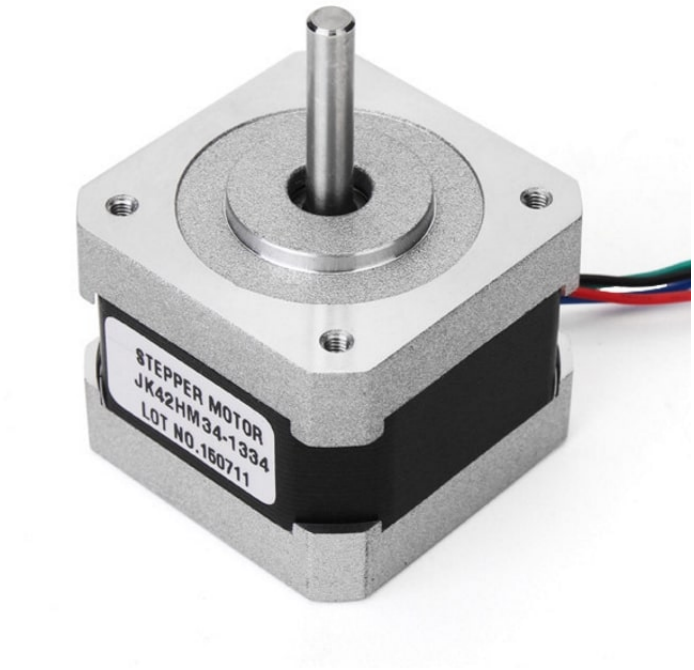
Fonte: do Autor.

isso resulta em um passo angular de 0,45º. A partir dessa informação, é possível estimar a distância entre dois pontos consecutivos, ambos correspondentes a um único passo angular, conforme detectado pelo LIDAR.

Assumindo que o feixe laser do LIDAR, ao formar um ângulo de 90º em relação a uma parede a uma distância de X metros, o próximo ponto no passo angular de 0,45º estará a uma distância d do primeiro ponto. Utilizando a regra do triângulo-retângulo, é possível expressar d como a Equação (8). Para uma distância X de 2 metros, os pontos consecutivos apresentarão uma distância d de 1,57 cm. Portanto, essa configuração proporciona uma distribuição eficiente de pontos na horizontal graças ao motor de passo.

$$d = X.\tan 0.45^{\circ} \tag{8}$$

Figura 24 – Motor de passo Nema 17.



Fonte: (HERO, s.d.)

O micro servo motor utilizado no protótipo é o SG90, que é um servo motor compacto e leve, amplamente utilizado em aplicações de modelismo, robótica, e outras áreas onde o espaço e o peso são críticos. Suas principais características incluem seu tamanho reduzido, peso de aproximadamente 9 gramas, operação de baixa tensão (geralmente entre 4,8V a 6V), torque moderado (em torno de 1,5 a 2,5 kg-cm) e uma faixa de rotação limitada em 180º. Eles são conhecidos pela precisão e capacidade de controle fino, sendo usados para controlar pequenos mecanismos e movimentos em aplicações onde o espaço é restrito, sendo essas as razões por sua escolha neste projeto.

Sobre a resolução do SG90, não foi encontrado nada definitivo em sua documentação. Portanto, para esse projeto foi utilizado o passo de 2º, que demonstrou resultados satisfatórios ao analisar a imagem da nuvem de pontos gerada. Aplicando a Equação (8), a distância d para X igual a 2 metros seria 6,98 cm.

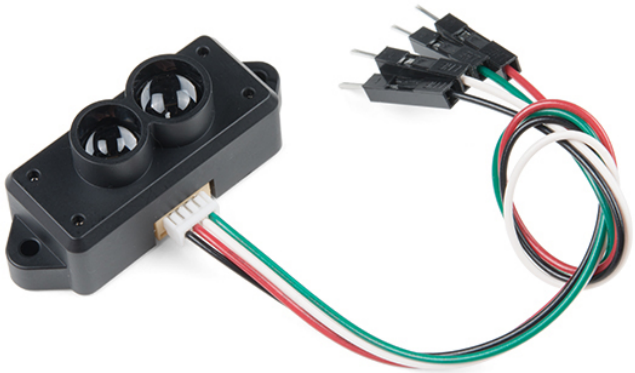
O LIDAR utilizado é o módulo TF Mini da Figura 26 que é um dispositivo de medição de distância sem contato que utiliza a tecnologia *Time-of-Flight* para fornecer medições precisas e em tempo real. O módulo opera com uma tensão de alimentação de 5V e uma corrente média de até 120mA, com um pico de corrente de 800mA. O módulo utiliza um protocolo de comunicação LVTTL (3.3V) e pode ser conectado a uma variedade de plataformas, incluindo dispositivos Arduino e derivados. O módulo é capaz de medir distâncias de pelo menos 30 centímetros e de até 12 metros com boa precisão e estabilidade, sendo sua precisão de ±4cm entre distâncias de 30cm à 6m e de ±6cm de 6m

Figura 25 – Micro servo motor SG90.

Fonte: (USINAINFO, s.d.)

até 12m. O TF Mini também possui uma taxa de frequência de até 1000Hz, permitindo medições rápidas e precisas em tempo real. Ele é compacto, de baixo custo e fácil de usar, tornando-o uma solução ideal para aplicações em robótica, automação industrial, drones, veículos autônomos e outras áreas.

Figura 26 – Sensor LIDAR TF Mini.

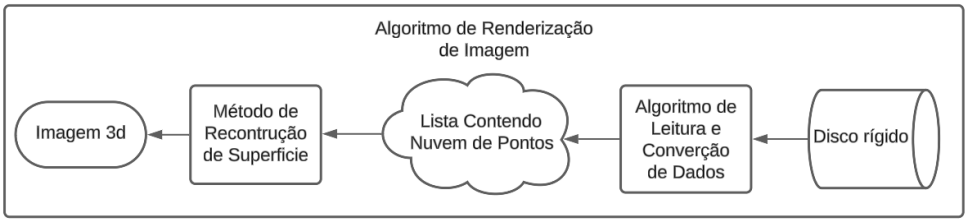


Fonte: (ELECTRONICS, s.d.)

3.3.1 Reconstrução de Superfície em uma Imagem 3D

Para começar processo de reconstrução de superfície é necessário, primeiro, ler os dados do disco e convertê-los em pontos cartesianos. Com os pontos pode-se aplicar um método de reconstrução de superfície para então gerar uma imagem 3D. Este processo está ilustrado no diagrama da Figura 27. Os métodos utilizados para os testes foram o BPA, *A-Shapes* e Poisson. Todos os algoritmos foram feitos em *python* e para a aplicação dos métodos de reconstrução de superfície em renderização da imagem, foi utilizado a biblioteca *Open 3D*.

Figura 27 – Diagrama do Algoritmo de Reconstrução de Imagem.



Fonte: do Autor

Conforme descrito no parágrafo anterior, com os dados coletados, primeiramente, é necessário convertê-los para um ponto no espaço usando as Equações (5), (6) e (7). Com isso podemos atribuir um vetor de pontos que represente o espaço analisado. O Código 3.1 representa o algoritmo deste processo, nele se observa que o arquivo é aberto em modo leitura, onde cada linha é lida e armazenada na váriavel *valores*. Essa variável é um vetor contendo dados do passo do servo motor, passo do motor de passo e distância lida pelo LIDAR. Esses dados são processados e transformados em coordenadas de um ponto no espaço. No final, esse ponto é armazenado numa lista usada para exibir os todos os pontos em uma imagem 3D ao final do processo.

```
# Abre o arquivo em modo de leitura
with open(caminho_arquivo, 'r') as arquivo:
    # Leia cada linha do arquivo
    for linha in arquivo:
        # Divide a linha em partes usando a v rgula como separador
        valores = linha.strip().split(',')

        # Verifica se a linha possui tr s valores
        if len(valores) == 3:
            dist = float(valores[0])
            phi = math.radians(float(valores[2]))
            theta = math.radians(float(valores[1]) * (180/steps))

            xValue = dist * math.sin(phi) * math.cos(theta)
            yValue = dist * math.sin(phi) * math.sin(theta)
            zValue = dist * math.cos(phi)
```

```
17
18              obj = {
19                  'x': float(xValue),
20                  'y': float(yValue),
21                  'z': float(zValue)
22              }
23
24              # Adiciona pontos na lista
25              objetos_lista.append(obj)
```

Código 3.1 – Converção dos dados em nuvem de pontos

A partir da nuvem de pontos é aplicado um método de reconstrução de superfície para gerar uma imagem 3D que represente o ambiente analisado. Para este projeto, foram escolhidos os três métodos apresentados nas Seções 2.1.8.1, 2.1.8.2, 2.1.8.4, ou seja, *Ball Pivoting*, Reconstrução de superfície por Poisson e *A-Shapes*. Para tal, foi utilizado a biblioteca Open 3D, que possui suporte para esses métodos.

O Código 3.2 representa o algoritmo do *Ball-Pivoting*. No algoritmo foi adicionado vetores normais à nuvem de pontos, atribuindo *point_cloud.normals*. Na imagem gerada deste método, a superfície só pode ser visualizada na vista contra a direção do vetor normal, portanto, a direção dos vetores foi definido do ponto obtido até a origem para que se possa ter uma vista de dentro do ambiente. No código também se observa que foi adicionado um vetor de raios de 4 elementos, eles representam o raio da esfera que percorre os pontos formando triângulos e a quantidade de tentativas para esse processo.

```
1  # Cria uma nuvem de pontos Open3D
2  point_cloud = o3d.geometry.PointCloud()
3
4  # Adiciona os arrays ao point_cloud
5  point_cloud.points = o3d.utility.Vector3dVector(np.column_stack((x_array
       , y_array, z_array)))
6
7  # Adiciona os vetores normais ao point_cloud
8  point_cloud.normals = o3d.utility.Vector3dVector(np.column_stack((
       vetor_x_array, vetor_y_array, vetor_z_array)))
9
10 # Vetor de raios da bola para 4 tentativas
11 radii = [16, 20, 22, 25]
12
13 # Cria nuvem de pontos com o m todo Ball Pivoting
14 rec_mesh = o3d.geometry.TriangleMesh.
       create_from_point_cloud_ball_pivoting(
15      point_cloud, o3d.utility.DoubleVector(radii))
16
17 o3d.visualization.draw_geometries([rec_mesh])
```

Código 3.2 – Algoritmo para gerar imagem 3D por Ball-Pivoting

No Código 3.3 se visualiza o algoritmo para gerar a imagem 3D pelo método de Poisson. Neste método, assim como *Ball-Pivoting*, também necessita de vetores normais que já estão inclusos em *point_cloud*, também definidos com a direção do ponto à origem. Neste algoritmo se observa que a malha foi criada a partir da chamada de *create_from_point_cloud_poisson*, essa função recebe a nuvem de pontos e um número representando o nível de detalhamento da malha. Logo abaixo do código anterior, se personaliza as densidades da malha em questões como cor e textura para ficar melhor de observar a profundidade da imagem.

```
# Constri malha pelo mtodo de Poisson a partir da nuvem de pontos
    com um detalhamento de nvel 15
with o3d.utility.VerbosityContextManager(o3d.utility.VerbosityLevel.
    Debug) as cm:
     mesh, densities = o3d.geometry.TriangleMesh.
    create_from_point_cloud_poisson(point_cloud, depth=15)

# Personaliza densidades de malha para melhor visualiza  o
densities = np.asarray(densities)
density_colors = plt.get_cmap('plasma')((densities - densities.min()) /
    (densities.max() - densities.min()))
density_colors = density_colors[:, :3]
density_mesh = o3d.geometry.TriangleMesh()
density_mesh.vertices = mesh.vertices
density_mesh.triangles = mesh.triangles
density_mesh.triangle_normals = mesh.triangle_normals
density_mesh.vertex_colors = o3d.utility.Vector3dVector(density_colors)

o3d.visualization.draw_geometries([density_mesh],
                                  zoom=0.664,
                                  front=[-0.4761, -0.4698, -0.7434],
                                  lookat=[1.8900, 3.2596, 0.9284],
                                  up=[0.2304, -0.8825, 0.4101])
```

Código 3.3 – Algoritmo para gerar imagem 3D por Poisson

O algoritmo para gerar a superfície por *A-Shapes* se encontra no Código 3.4. O método necessita de vetores normais, definidos com a direção do ponto à origem. No código se observa que é criado uma malha a partir da nuvem de pontos do objeto *TetraMesh_create_from_point_cloud*. A partir dessa malha, da nuvem de pontos e do parâmetro "a", se gera a superfície por *A-Shapes* pelo objeto *TriangleMesh_create_from_point_cloud_alpha_shape*. O parâmetro "*alpha*" que define o nível de detalhamento da imagem gerado, quando maior o valor mais detalhado a imagem fica.

```
# Adiciona os arrays ao point_cloud
point_cloud.points = o3d.utility.Vector3dVector(np.column_stack((x_array
    , y_array, z_array)))

```

```
# Gera uma malha a partir da nuvem de pontos
tetra_mesh, pt_map = o3d.geometry.TetraMesh.create_from_point_cloud(
    point_cloud)

# Gera superf cie pelo m todo A-Shapes
alpha = 12
mesh = o3d.geometry.TriangleMesh.create_from_point_cloud_alpha_shape(
    point_cloud, alpha, tetra_mesh, pt_map)
mesh.compute_vertex_normals()

o3d.visualization.draw_geometries([mesh], mesh_show_back_face=True)
```

Código 3.4 – Algoritmo para gerar imagem 3D por A-Shapes

4 RESULTADOS

Neste capítulo são mostrados os resultados obtidos pelo protótipo, através do processamento dos dados em imagens 3D. Abaixo se observa as especificações do computador em que se executou todos os algoritmos:

- Processador: AMD Ryzen 5 5600X *6-Core Processor* 3.70 GHz
- Placa de Vídeo: AMD Radeon RX 6600 GDDR6 8GB
- Memória RAM: 2x8GB DDR4 3200MHz.

Dos testes feitos, é primeiramente mostrado e analisado a imagem da nuvem de pontos de comparado com ambiente real. Nas seguintes seções são mostrados e analisados os resultados das técnicas de reconstrução por Poisson, BPA e *A-Shapes*.

4.1 AMBIENTE DE TESTES

Para os testes, foi limitado a rotação horizontal em 145º devido à demora para completar o escaneamento. O ambiente escaneado foi o quarto da Figura 28, que contém uma cama, uma janela coberta com cortinas, um ventilador de teto, um armário e outros objetos. Todo esse acumulo de móveis e objetos adicionam uma maior complexidade geométrica na superfície escaneada, assim servindo como um bom teste para a eficiência do protótipo e dos métodos de reconstrução de superfície. O protótipo foi colocado em uma banqueta de 1m de altura e posicionado próximo ao centro do quarto, assim dando uma visão mais ampla para o sensor escanear. O processo de escaneamento demorou cerca de 3 horas.

4.2 NUVEM DE PONTOS

A partir dos dados coletados pelo protótipo e convertidos para uma nuvem de pontos representativa do ambiente escaneado, nas Figuras 29 e 30 se encontram a nuvem renderizada em uma imagem 3D. Já na imagem gerada, se consegue passar um paralelo com a imagem real do ambiente, onde se consegue observar agrupamentos de pontos formando algo parecido com alguns móveis do quarto, como a cama, o armário, as cortinas e o próprio ventilador de teto. Este paralelo fica mais claro rotacionando a imagem no ambiente renderizado e com a superfície reconstruída, que será discutido nas seções posteriores. Na parte superior da cama e superior das paredes, se observam que a fileira de pontos horizontais estão mais afastados uns dos outros.

Esse afastamento dos pontos nos locais mencionados provavelmente estão relacionados a dois fatores principais. O primeiro fator que afeta todo o resultado se deve ao servo motor utilizado não ser tão preciso e confiável. Apesar de em seu *datasheet* informar que o motor rotaciona até 180º, isto não foi observado na prática para diferentes motores

Figura 28 – Quarto Mapeado.

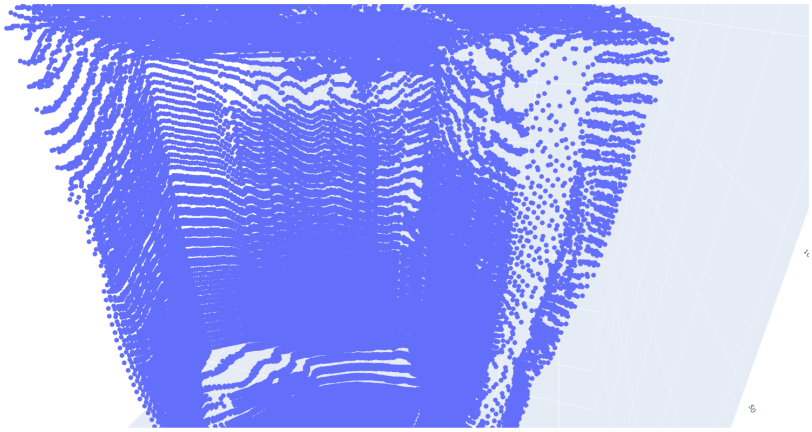
Fonte: do Autor

testados. O segundo fator é causado pela reflexão do *laser* com à superfície. Quanto mais agudo é o angulo do *laser* com a superfície, maior é o angulo entre o raio incidente e o raio refletido, já que os ângulos são iguais e espelhados conforme visto na Seção 2.1.2. Isto atrapalha a captação do raio refletido pelo receptor do LIDAR, causando um sinal fraco de baixa confiança.

4.3 RECONSTRUÇÃO DE IMAGEM POR POISSON

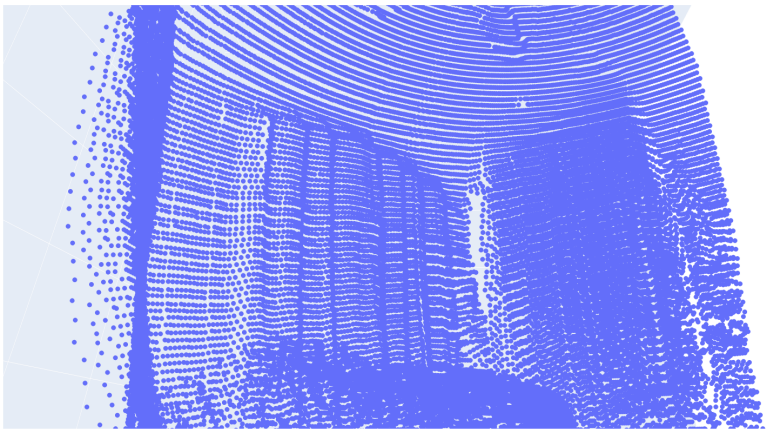
Na Figura 37a se observa o resultado da imagem gerada pelo método de reconstrução de superfície por Poisson a partir da nuvem de pontos gerada. A imagem por este método demorou cerca de 15 segundos para ser renderizada. Na Figura 32 se compara lado a lado o ambiente gerado com o real, nele se observa uma suavização nas bordas das superfícies geradas dos móveis. Observando ventilador, se vê que a superfície estava tendendo a renderizar suas aletas, mesmo em seus pontos cegos. Com isso, evidencia a vantagem desse método de conseguir lidar com dados ruidosos e incompletos.

Figura 29 – Perspectiva 1 da Nuvem de Pontos do Ambiente.



Fonte: Do autor.

Figura 30 – Perspectiva 2 da Nuvem de Pontos do Ambiente.



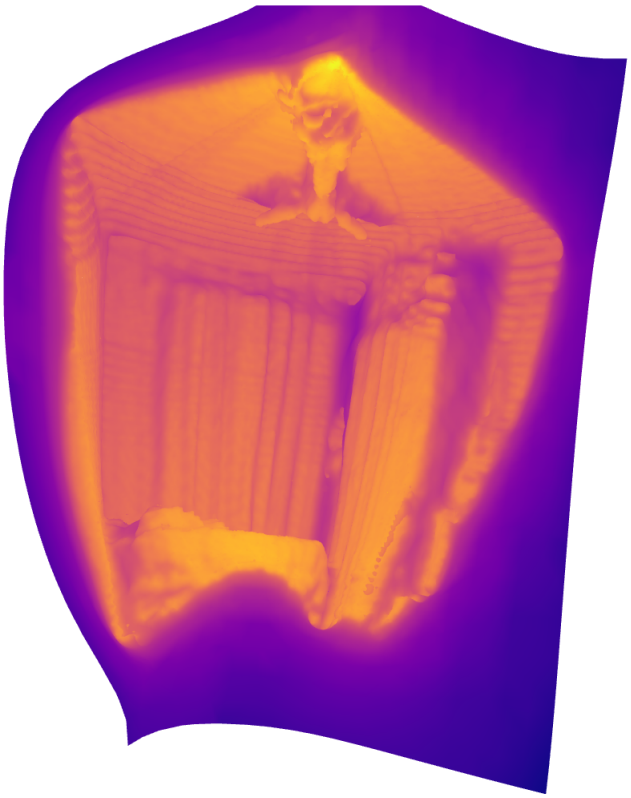
Fonte: Do autor.

4.4 RECONSTRUÇÃO DE IMAGEM POR *BALL-PIVOTING*

Na Figura 34a se ilustra a reconstrução realizada pelo método *Ball-Pivoting*. Este método demorou cerca de 25 minutos para renderizar a imagem, sendo que o Poisson demorou cerca de 1 minuto. Comparado o ambiente gerado com o real da Figura 34, a distinção entre os elementos construídos e o ambiente real é perceptível. No entanto, em contraste com a abordagem Poisson, a renderização apresenta algumas imperfeições, como superfícies em lugares indesejados e lacunas nos objetos. Essas falhas surgem em regiões com escassez de pontos, afastados uns dos outros e, portanto, fora do alcance do raio de vizinhança da esfera pivotada.

Para abordar integralmente essas lacunas, seria necessário aumentar o raio de vi-

Figura 31 – Ambiente reconstruído por Poisson.



Fonte: Do autor.

zinhança. No entanto, essa ampliação acarreta formação de mais triângulos, inclusive triângulos indesejados. Como evidenciado na Figura 35, a ampliação do raio resulta na criação excessiva de triângulos nas extremidades e cantos da superfície gerada, prejudicando a fidelidade da reconstrução do ambiente real.

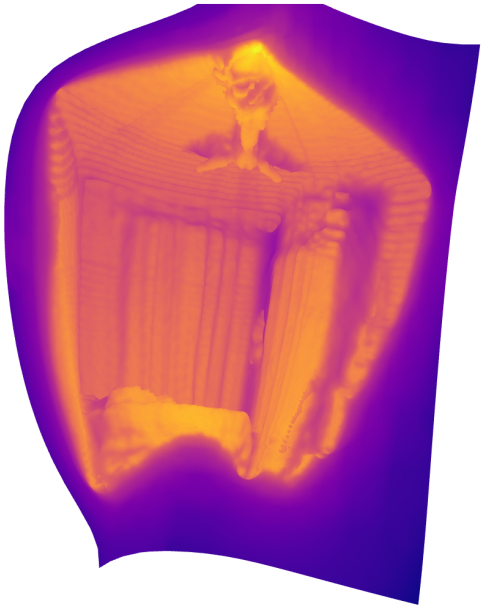
4.5 RECONSTRUÇÃO DE IMAGEM POR *A-SHAPES*

O resultado da reconstrução de superfície pelo método *A-Shapes* está representando na Figura 36 e 38. A renderização dessa imagem foi tão rápida quanto a pelo método de Poisson. Comparando com o ambiente real, Figura 37, assim como as imagens dos outros métodos, é possível relacionar objetos na imagem gerada com os objetos da imagem real do ambiente testado. Porém, fica evidente pela Figura 38 que o esse método não conseguiu preencher todos os espaços com superfície, deixando alguns buracos na imagem.

Numa tentativa de preencher esses buracos, foi aumentado o parâmetro "a", que resultou numa imagem sem buracos. Porém, nessa imagem formou superfícies em lugares indesejados, deixando a superfície total muito carregada. As Figuras 36 e 38 foram o melhor meio-termo entre preencher os buracos e não deixar a imagem com superfícies indesejadas.

Figura 32 – Comparação do ambiente reconstruído por Poisson e real.

(a) Ambiente Reconstruído.



Fonte: Do autor.

(b) Ambiente Real.

Fonte: Do autor.

Figura 33 – Ambiente reconstruído por *Ball-Pivoting*.

Fonte: Do autor.

Figura 34 – Comparação do ambiente reconstruído por *Ball-Pivoting* e real.

(a) Ambiente Reconstruído.

Fonte: Do autor.

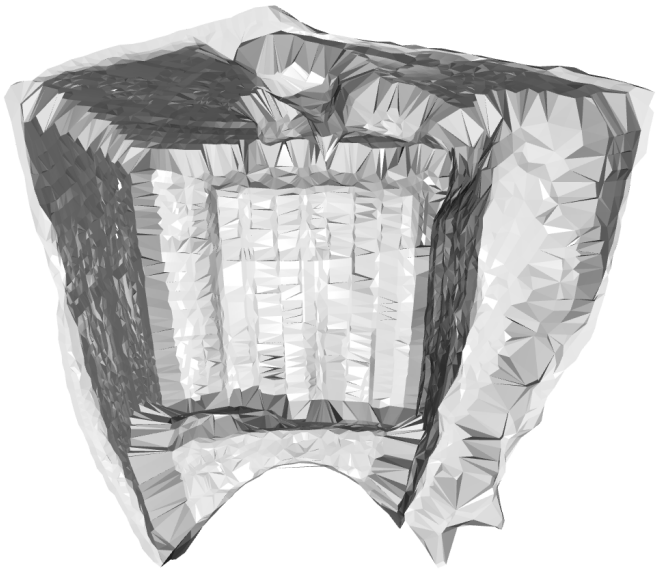
(b) Ambiente Real.

Fonte: Do autor.

Figura 35 – Ambiente por *Ball-Pivoting* com raio de vizinhança aumentado.

Fonte: Do autor.

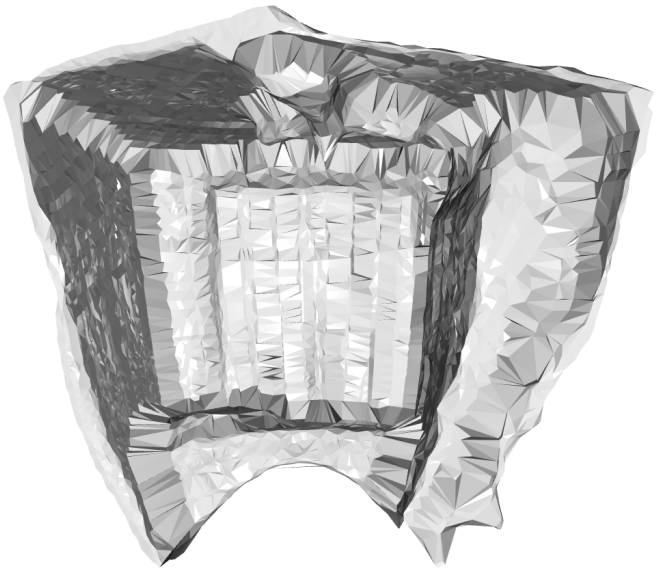
Figura 36 – Ambiente reconstruído por *A-Shapes*.



Fonte: Do autor.

Figura 37 – Comparação do ambiente reconstruído por *A-Shapes* e real.

(a) Ambiente Reconstruído.

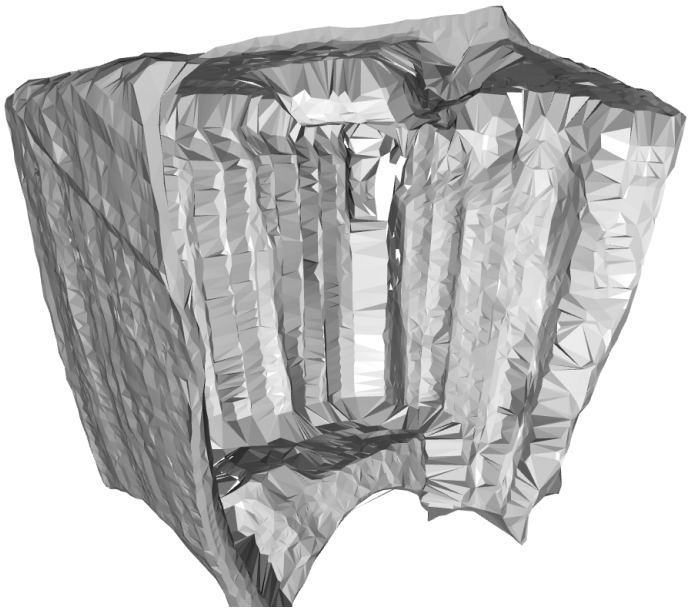


Fonte: Do autor.

(b) Ambiente Real.

Fonte: Do autor.

Figura 38 – Ambiente reconstruído por *A-Shapes* por outra perspectiva.



Fonte: Do autor.

5 CONCLUSÃO

É evidente o significativo impacto que a aplicação de LIDAR para escaneamento de ambientes tem no campo de mapeamento 3D. A utilização do LIDAR para mapear ambientes tridimensionais demonstra uma abordagem promissora para a criação de sistemas autônomos e inteligentes. A integração de tecnologias como o LIDAR, giroscópio e sensores de proximidade, aliada ao uso de algoritmos avançados, abre caminho para o desenvolvimento de soluções cada vez mais sofisticadas e eficientes.

Conforme observado no projeto, apesar de as imagens geradas conseguirem representar as formas do ambiente mapeado, ainda demonstra irregularidades e buracos. Esses problemas ocorrem devido à qualidade questionável do servo motor utilizado e da leitura comprometida do TF Mini para ângulos agudos, conforme discutido na Seção 4.2. O método de Poisson, em relação ao BPA e *A-Shapes*, se demonstrou ser o único satisfatório, por não deixar buracos e suavizar as irregularidades. Para melhorar esses resultados seria necessário escolher um servo motor mais confiável do que o utilizado neste projeto e realizar o mapeamento em diferentes pontos do ambiente para evitar pontos cegos e ângulos agudos.

Por fim, este documento ressalta a importância de comparar e avaliar a eficiência das técnicas de reconstrução de superfície utilizadas, evidenciando a busca por aprimoramento contínuo e a preocupação com a qualidade dos resultados obtidos. A integração de *hardware*, *software* e algoritmos de forma coesa e de baixo custo, como demonstrado nesse estudo, representa uma forma alternativa e acessível para solucionar demandas de mapeamentos internos.

REFERÊNCIAS

AGNIHOTRI, Nikhil. **Stepper Motor : Basics, Types and Working**. Acessado em: 29/03/2024. Disponível em: https://www.engineersgarage.com/stepper-motor-basics-types-and-working/.

BAPTISTA, Fabiana Tesine. **O Ensino de coordenadas polares através do software Geogebra**. 2017. Tese (Doutorado) – Universidade Estadual de Campinas.

BERNARDINI, Fausto; MITTLEMAN, Joshua; RUSHMEIER, Holly; SILVA, Claudio; TAUBIN, Gabriel. The ball-pivoting algorithm for surface reconstruction. **IEEE Transactions on Visualization and Computer Graphics**, v. 5, n. 4, p. 349–359, 1999.

BRITES FELIPE GONÇALVES E SANTOS, Vinicius Puga de Almeida. Motor de Passo. **PETele**, 2008.

CHENG, Yi; WANG, Gong Ye. Mobile robot navigation based on lidar. *In*: 2018 Chinese Control And Decision Conference (CCDC). [*S.l.*: *s.n.*], 2018. P. 1243–1246.

COOLIDGE, Julian Lowell. The Origin of Polar Coordinates. **American Mathematical Monthly**, v. 59, p. 78–85, 1952.

DENYSYUK, Pavlo; TESLYUK, Vasyl; CHORNA, Iryna. Development of mobile robot using LIDAR technology based on Arduino controller. *In*: 2018 XIV-th International Conference on Perspective Technologies and Methods in MEMS Design (MEMSTECH). [*S.l.*: *s.n.*], 2018. P. 240–244.

EDELSBRUNNER, H.; KIRKPATRICK, D.; SEIDEL, R. On the shape of a set of points in the plane. **IEEE Transactions on Information Theory**, v. 29, n. 4, p. 551–559, 1983.

ELECTRONICS, SparkFun. **TFMini - Micro LiDAR Module**. Acessado em: 03/06/2024. Disponível em: https://www.sparkfun.com/products/retired/14588.

EVANS, Martin; NOBLE, Joshua; HOCHENBAUM, Jordan. **Arduino em Ação**. [*S.l.*]: Novatec, 2013.

FREERTOS. **The FreeRTOS™ Kernel**. Acessado em: 29/03/2024. Disponível em: https://www.freertos.org/RTOS.html.

GREWAL, Harkishan; MATTHEWS, Aaron; TEA, Richard; GEORGE, Kiran. LIDAR-based autonomous wheelchair. *In*: 2017 IEEE Sensors Applications Symposium (SAS). [*S.l.*: *s.n.*], 2017. P. 1–6.

GRISETTI, Giorgio; STACHNISS, Cyrill; BURGARD, Wolfram. Improved techniques for grid mapping with Rao-Blackwellized particle filters. *In*: 1. IEEE Transactions on Robotics. [*S.l.*]: IEEE, 2007. v. 23, p. 34–46.

GUEDES, Andre. **Triangulação**. Acessado em: 03/03/2024. Disponível em: https://www.inf.ufpr.br/andre/geom/node8.html.

HARRAP, Rob; LATO, Matt. An Overview of LIDAR: collection to applications. *In*: p. 1–9.

HECHT, Eugene. **Optics, Fifth Edition**. [*S.l.*]: Pearson Education, 2017.

HERO, Maker. **Motor de Passo NEMA 17 1,33A 34mm para Impressora 3D**. Acessado em: 01/03/2024. Disponível em: https://www.makerhero.com/produto/motor-de-passo-nema-17-34mm-impressora-3d/.

KAZHDAN, Michael; HOPPE, Hugues. Screened Poisson Surface Reconstruction. **ACM Trans. Graph.**, Association for Computing Machinery, New York, NY, USA, v. 32, n. 3, jul. 2013. ISSN 0730-0301.

KOHLBRECHER, Stefan; VON STRYK, Oskar; MEYER, Johannes; KLINGAUF, Uwe. A flexible and scalable SLAM system with full 3D motion estimation. *In*: IEEE. 2011 IEEE International Symposium on Safety, Security, and Rescue Robotics. [*S.l.*: *s.n.*], 2011. P. 155–160.

LAPLANTE, Phillip A.; OVASKA, Seppo J. **Real-Time Systems Design and Analysis: Tools for the Practitioner**. Hoboken, New Jersey: IEEE Press, John Wiley & Sons, Inc., 2012.

MA, Wei; LI, Qingquan. An Improved Ball Pivot Algorithm-Based Ground Filtering Mechanism for LiDAR Data. **Remote Sensing**, v. 11, n. 10, 2019. ISSN 2072-4292.

MATTEDE, Henrique. **O que é Servo motor e como funciona?** Acessado em: 03/06/2024. Disponível em: https://www.mundodaeletrica.com.br/o-que-e-servo-motor-e-como-funciona/.

MENDES, Mariane. **O que é reflexão da luz?** Acessado em: 03/06/2024. Disponível em: https://brasilescola.uol.com.br/o-que-e/fisica/o-que-e-reflexao-luz.htm.

N.A. RAHIM, Tew Chin Keong S.A.A. Shukor. **Development of a Mobile Platform with IoT for LIDAR**. Malaysia, 2021.

PAVAN, Nadisson Luis; SANTOS, Daniel Rodrigues dos. UM MÉTODO AUTOMÁTICO PARA REGISTRO DE DADOS LASER SCANNING TERRESTRE USANDO SUPERFÍCIES PLANAS. **Boletim de Ciências Geodésicas**, Universidade Federal do Paraná, v. 21, n. 3, p. 572–589, jul. 2015. ISSN 1982-2170.

ROYO, Santiago; BALLESTA-GARCIA, Maria. An Overview of Lidar Imaging Systems for Autonomous Vehicles. **Applied Sciences**, v. 9, n. 19, 2019. ISSN 2076-3417.

SYSTEMS, Espressif. **ESP32 Series Datasheet v4.5**. [*S.l.*: *s.n.*]. https://www.espressif.com/documentation/esp32_datasheet_en.pdf. Acessado em: 16/03/2024.

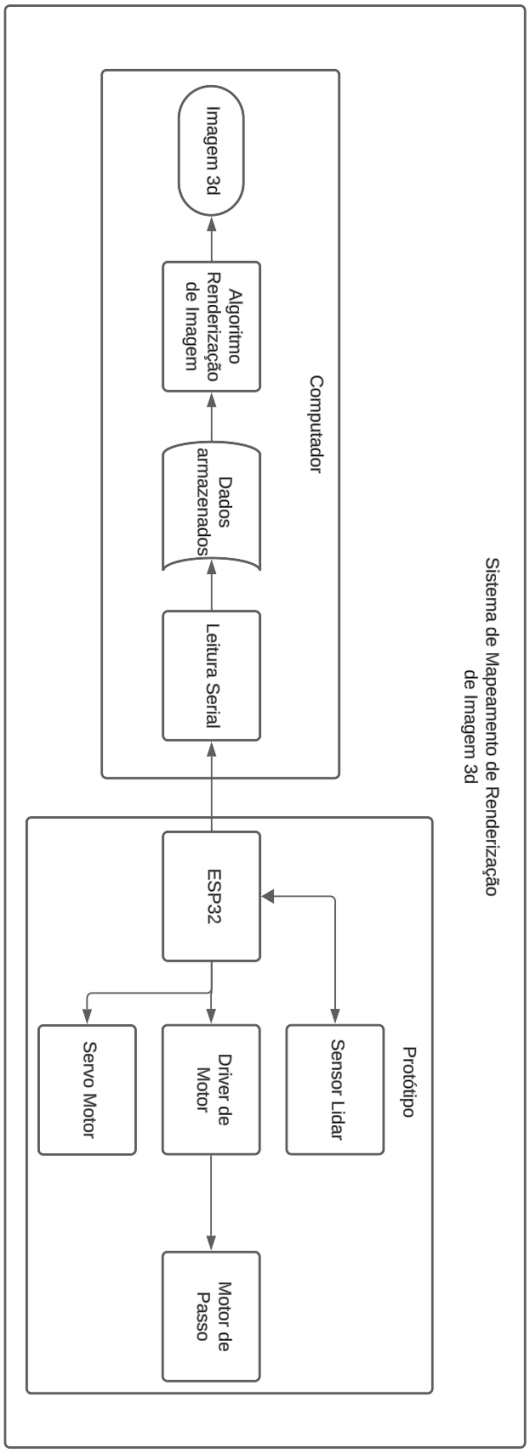
SYSTEMS, Espressif. **ESP32-DevKitC V4 Getting Started Guide**. Acessado em: 17/03/2024. Disponível em: https://docs.espressif.com/projects/esp-idf/en/stable/esp32/hw-reference/esp32/get-started-devkitc.html.

USINAINFO. **Micro Servo Motor 9g SG90 360° 1.6Kgf.cm Rotação Contínua**. Acessado em: 01/03/2024. Disponível em: https://www.usinainfo.com.br/servo-motores/micro-servo-motor-9g-sg90-360-16kgfcm-rotacao-continua-8508.html.

WIKIPEDIA. **Coordenaes polares**. Acessado em: 06/07/2024. Disponível em: https://ast.wikipedia.org/wiki/Coordenaes_polares.

Anexos

ANEXO A – Sistema de Mapeamento de Ambiente 3D



Fonte: Do autor.