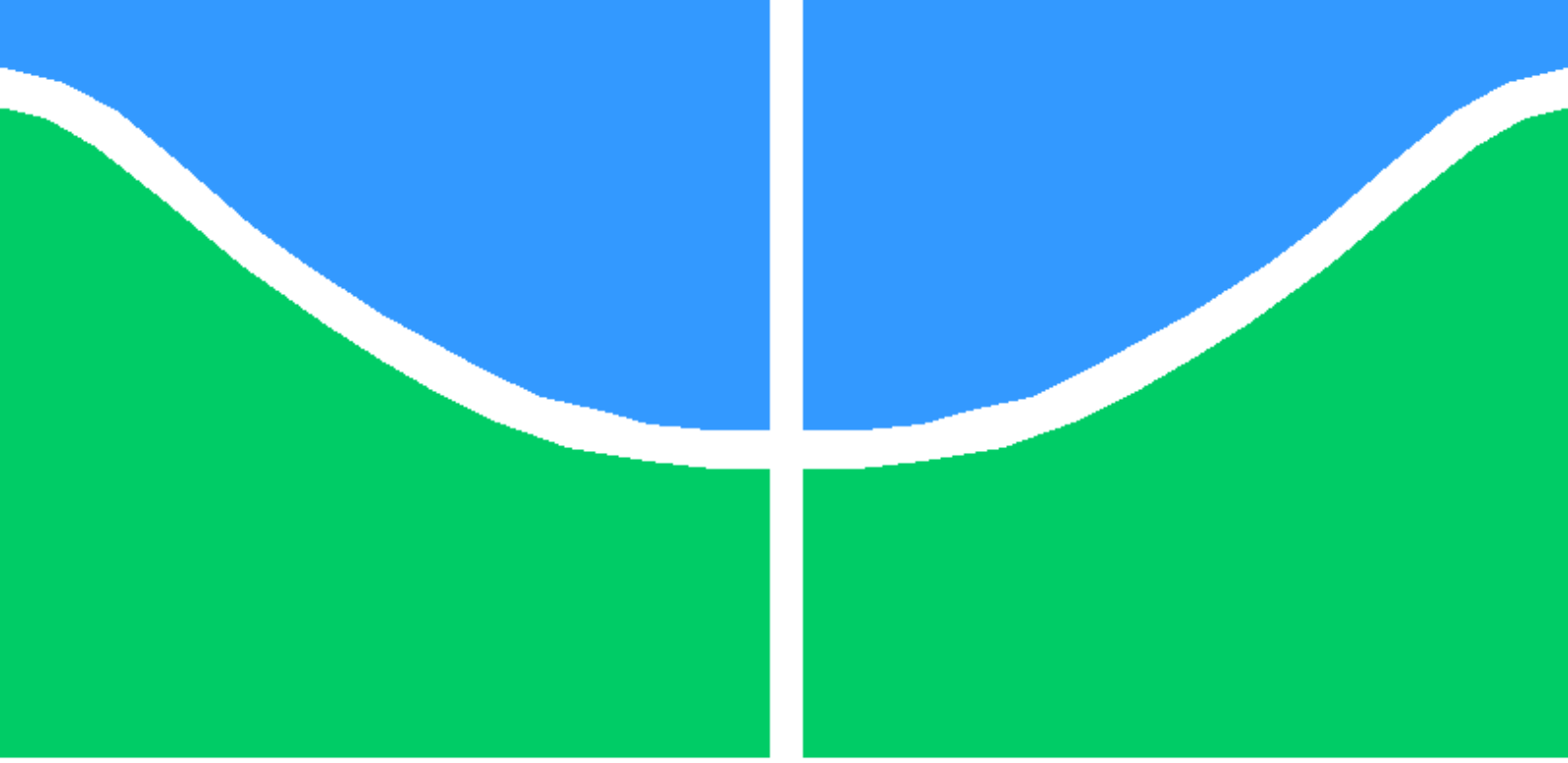

HARDWARE DE BASE PARA UM ESCÂNER 3D

JEREMY PAULE PEREIRA

TRABALHO DE CONCLUSÃO DE CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

FACULDADE DE TECNOLOGIA

UNIVERSIDADE DE BRASÍLIA

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

HARDWARE DE BASE PARA UM ESCÂNER 3D

JEREMY PAULE PEREIRA

Orientador: PROF. DR. DANIEL CHAVES CAFÉ, ENE/UNB

TRABALHO DE CONCLUSÃO DE CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

PUBLICAÇÃO PPGENE.DM - XXX/AAAA
BRASÍLIA-DF, 06 DE DEZEMBRO DE 2018.

UNIVERSIDADE DE BRASÍLIA
FACULDADE DE TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA ELÉTRICA

HARDWARE DE BASE PARA UM ESCÂNER 3D

JEREMY PAULE PEREIRA

TRABALHO DE CONCLUSÃO DE CURSO DE GRADUAÇÃO ACADÊMICO SUBMETIDA AO DEPARTAMENTO DE ENGENHARIA ELÉTRICA DA FACULDADE DE TECNOLOGIA DA UNIVERSIDADE DE BRASÍLIA, COMO PARTE DOS REQUISITOS NECESSÁRIOS PARA A OBTENÇÃO DO GRAU DE ALUNO EM ENGENHARIA ELÉTRICA.

APROVADA POR:

Prof. Dr. Daniel Chaves Café, ENE/UnB
Orientador

Claudia Barenco Abbas
Professor Avaliador

Eduardo Peixoto Fernandes da Silva
Professor Avaliador

BRASÍLIA, 06 DE DEZEMBRO DE 2018.

FICHA CATALOGRÁFICA
JEREMY PAULE PEREIRA
Hardware de base para um escâner 3D
2018xv, 147p., 201x297 mm
(ENE/FT/UnB, Aluno, Engenharia Elétrica, 2018)
Trabalho de conclusão de curso de Graduação - Universidade de Brasília
Faculdade de Tecnologia - Departamento de Engenharia Elétrica

REFERÊNCIA BIBLIOGRÁFICA

JEREMY PAULE PEREIRA (2018) Hardware de base para um escâner 3D. Trabalho de conclusão de curso de Graduação em Engenharia Elétrica, Publicação xxx/AAAA, Departamento de Engenharia Elétrica, Universidade de Brasília, Brasília, DF, 147p.

CESSÃO DE DIREITOS

AUTOR: Jeremy Paule Pereira
TÍTULO: Hardware de base para um escâner 3D.
GRAU: Aluno ANO: 2018

Jeremy Paule Pereira
SQS 416 Bloco H, apartamento 202, Brasilia, DF

Agradecimentos

Gostaria de agradecer principalmente a:

Daniel Chaves Café, meu orientador. Por ter me ajudado no caminho final da minha graduação.

Aos professores Claudia Jacy Barenco Abbas, Eduardo Peixoto Fernandes da Silva e Ricardo Zelenovsky. Por terem me ajudado esclarecendo minhas dúvidas sobres os assuntos abordados na minha tese.

À minha mãe. Por tudo.

À minha namorada, Maira Bravo Ramos. Por ter sempre ficado ao meu lado, mesmo nos momentos em que fiquei mais tenso.

Aos meus amigos Ruan Perassa Coêlho, Iago Pereira, Gabriel Melo e Miguel Pachá. Por terem me apoiado e ajudado quando precisei durante esse semestre.

Resumo

Por conta da intensa necessidade de catalogar o conhecimento humano, de mais em mais aparece a necessidade de guardar informações sobre artefatos passados. Com o grande avanço da robótica, a interação máquina-homem vem avançando e, com isso, surgindo novas necessidades de aprimoramento para essa interface. Essas são algumas das motivações que levaram à iniciação desse projeto. Por ser um projeto bem ambicioso, o sensato seria começar por algo menor e de mais fácil implementação. Para isso, o foco foi de criar um aparelho capaz de replicar um objeto existente no mundo físico e replicá-lo em um ambiente virtual. Sendo assim seria criado um escâner 3D. Sendo assim foi estudado como implementar um escâner 3D. Começando pela escolha do sensor de distância mais preferível para o projeto. Após a escolha do sensor de distância, um motor de passos foi escolhido para que fosse possível mover o ângulo de visão do objeto a ser analisado. Foi estudado também qual estrutura seria melhor para a implementação de um protótipo de escâner. Ao se implementar a interface entre o sensor e o motor, foi utilizado um Arduino MEGA e a interface entre o Arduino e a criação do arquivo de Nuvem de Pontos (Point Cloud) foi feita pela porta serial e processada em programas específicos.

SUMÁRIO

LISTA DE FIGURAS

Capítulo 1

Explicação do Problema

1.1 Contextualização

Nesse ano (2018) sofremos uma grande perda de registros históricos no Brasil. Ocorreu um incêndio no Museu Nacional do Rio de Janeiro[22]. Muitas obras de arte e registros históricos foram perdidos nesse incêndio por falta de manutenção e investimento. Por mais que não se possa fazer nada com os objetos destruídos, caso as esculturas ou achados arqueológicos tivessem sido escaneados, seria possível continuar estudos sobre eles e, até mesmo colocar réplicas à exposição.

Desde o início da civilização, o homem tem a necessidade de guardar informações de seu passado. Em museus temos registros do início da civilização humana até registros dos primeiros seres que viveram em nosso planeta. Um grande problema que enfrentamos é que muitos dos objetos que guardam nossa história tem pouca durabilidade, o que reduz a sua possibilidade de estudo. Esses objetos não podem ser realocados ou tocados. Sendo assim, para estudá-los é necessário estar em um ambiente controlado, que nem sempre é de fácil acesso para pesquisadores ou entusiastas que gostariam de conhecer mais sobre esses objetos. Através da reconstrução em ambiente virtual de artefatos arqueológicos ou paleontológicos, esses objetos seriam de fácil acesso a qualquer pessoa que possuísse alguma plataforma de visualização tridimensional.

Como citado em [8] o escaneamento tridimensional vem ganhando mais espaço em todo o mundo, inclusive na parte de pesquisa arqueológica. Métodos de catalogação de áreas em que existem grandes construções e achados arqueológicos, como: fotogrametria de curto alcance e medidas taqueométricas já podem ser substituídos por escaneamento 3D.

Para o estudo de grandes construções ou achados arqueológicos são utilizados diversos métodos. Dois métodos de coleta de dados para a gravação de informações sobre edifícios de herança cultural são a Taqueometria e a Fotogrametria. Essas duas formas de coleta de informações exigem mais tempo. A Taqueometria, por exemplo é eficiente quando se necessita medir a distância entre dois pontos. Para medir diversos pontos de uma estrutura

inteira são necessárias diversas horas. A Fotogrametria é uma forma útil de capturar formas geométricas para depois digitalizá-las, o seu problema é que uma câmera de alta qualidade é de difícil transporte, além de que se perde muita informação com esse método, a textura, por exemplo, não é analisada. Já com o escaneamento a laser esses pequenos detalhes já são possíveis de capturar [23], como mostrado na figura 1.1.

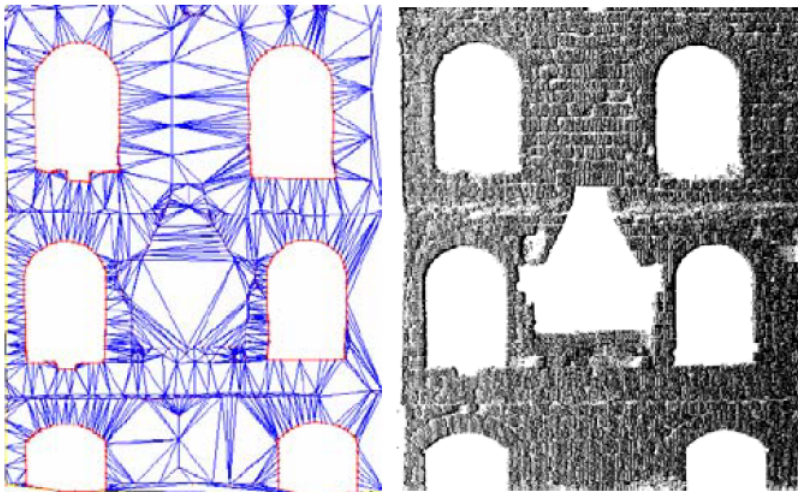

Figura 1.1: Comparação entre pontos triangularizados da fotogrametria (à esquerda) e a correspondente parede escaneada por laser.[23]

Por conta da quantidade de pontos que são capturados e pelo tempo necessário, o escaneamento a laser consegue capturar mais detalhes do que qualquer outra forma de captura de imagens tridimensionais.

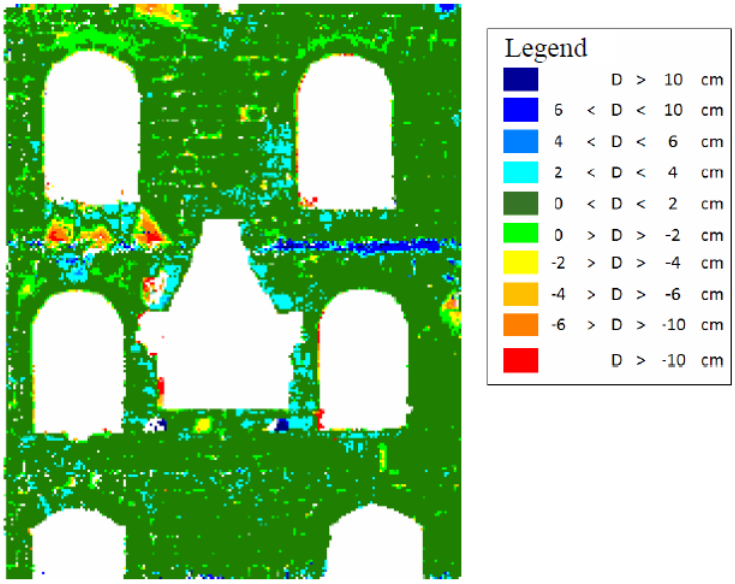

Figura 1.2: Relação entre a parede triangularizada e o escaneamento 3D (os pontos da triangularização foram considerados como o zero)[23]

Ainda se referindo a construções. O escaneamento 3D serve também para o estudo prévio de um ambiente em que será feita alguma obra de engenharia civil. Para esse caso, um escaneamento prévio pode ajudar na logística de como o projeto começará. [44].

Também há o caso de reconstrução em ambiente virtual de membros de pessoas para que sejam feitas órteses, como citado em[17]. Esse processo captura a forma do paciente e assim uma órtese pede ser feita sob medida de acordo com as necessidades de cada paciente e sem a necessidade do contato direto do gesso com o membro do usuário.

O projeto se baseou na ideia de criar uma máquina capaz de captar informações de localização de um objeto no plano real e leva-lo a um plano digital. Deixando o objeto capaz de passar por simulações sem comprometer sua estrutura física, ou de copiá-lo e recriá-lo em um curto período de tempo.

Além da capacidade de reconstruir objetos reais em ambientes digitais, seria possível armazenar informação sobre objetos frágeis para que possam ser guardados para a posterioridade, como achados arqueológicos e fósseis de animais já extintos. Assim essas informações não seriam perdidas tão facilmente com o tempo e poderiam ser compartilhadas facilmente sem requerer um transporte que poderia danificá-las.

1.2 Objetivo Geral

Construir um aparelho capaz de tomar medidas de distância sobre um objeto e poder recriá-lo em um ambiente virtual de forma rápida e que sua construção seja de baixo custo. Assim pequenos objetos poderiam ser rapidamente processados em ambiente digital, facilitando seu estudo para qualquer tipo de público.

1.3 Objetivos Específicos

Criar um código que possa capturar todos os dados de um objeto a ser analisado, interagindo apenas com o sensor de distância e dois motores de passo.

Criar um código que seja capaz de ler as informações enviadas à porta Serial e colocá-los em um arquivo compatível com uma biblioteca PCL.

Construir uma estrutura que suporte todos os equipamentos para as medidas serem tomadas. Suporte para Motor de Passos, Arduino, Sensor de Distância.

Estudar diferentes formas de implementar esse mecanismo.

Executar simulações com todos os componentes e gerar uma simulação PCL.

Capítulo 2

Embasamento Teórico

Este capitulo será dedicado a o estado atual dos estudos no âmbito de captura de informações através de sensores de distancia por tempo de voo está. Será abordado também como é a situação de estudos de Point Clouds, já que é um dos principais temas abordado nesse trabalho. Por fim será explicado melhor onde e como são usadas as técnicas e aparelhos de escaneamento tridimensional.

2.1 Sensores

Hoje já são diversas as formas de adquirir informação em três dimensões. Um método relativamente fácil é o de fotogrametria. Esse método necessita de mais de uma imagem bidimensional do objeto, ou espaço para que disso derive uma imagem tridimensional [41]. Ao analisar mais de uma imagem de um mesmo objeto é possível criar uma réplica tridimensional através de uma orientação de objeto, medidas de sua superfície e extração de características estruturais dele. Fotogrametria é bem precisa e comumente utilizada em cartografia, mapeamento, documentação tridimensional de achados arqueológicos e engenharia reversa. Um exemplo de uso da fotogrametria para estudo de terreno, pode ser visto a seguir na figura 2.1.

Outra forma é por sensores de proximidade. Esses sensores são capazes de determinar a presença de objetos próximos sem a necessidade de contato direto. Com esses sensores é possível recriar um objeto em um ambiente 3D coletando precisamente os dados da distância do objeto até o sensor. Existem diversas tecnologias para criar sensores de proximidade, dentre elas podemos encontrar tecnologias que se baseiam em: Sensores de Efeito Hall, Chaves de Indução de Proximidade, Sensores de Ultrassom e Sensores Óticos.

Sensores de Efeito Hall são sensores magnéticos muito eficazes para medir distância de metais. Sensores magnéticos conseguem detectar mudanças ou distúrbios em campos magnéticos criados. Chaves de Indução de Proximidade também são utilizadas para medir curtas distâncias entre metais. Sensores de indução geram ondas de rádio frequência em volta

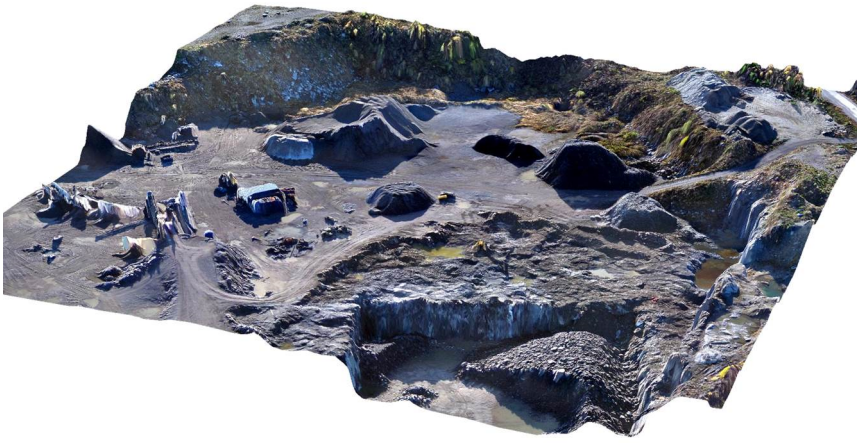

Figura 2.1: Terreno escaneado a cores[1]

de uma bobina. A indutância da bobina muda quando um objeto de metal é aproximado do campo. Na figura 2.2 podemos ver um exemplo de sensor de efeito Hall.

Figura 2.2: Sensor de Efeito Hall[46]

Sensores de ultrassom são úteis em distâncias médias para detectar objetos. Normalmente é composto por um transmissor e um receptor. O transmissor manda uma onda de som em certa frequência e o receptor deve detectar o retorno dessa onda. A distância é calculada através da velocidade da onda transmitida e o tempo que ela levou para retornar ao receptor. Na figura 2.3 temos um sensor de ultrassom.

Figura 2.3: Sensor de Ultrassom[37]

Por último temos os sensores óticos. Esses sensores também trabalham com um receptor e um transmissor. Ele usa a mesma forma de medição de distância dos sensores de ultrassom, mas por se basearem em ondas de luz, sua precisão é mais alta e seu tamanho reduzido.

Figura 2.4: Sensor Otico[42]

Outra tecnologia utilizada para gerar point clouds é a de câmeras baseadas no princípio de tempo de voo. Essa tecnologia possui a vantagem de adquirir dados por período de quadro, para obter um point clouds tridimensional a partir de um único ponto de perspectiva. Existem duas variações da forma de funcionamento de câmeras com o princípio tempo de voo. A primeira se baseia unicamente na medida do tempo de voo de um pulso de luz através de diodos de avalanche, o segundo método utiliza uma onda de luz modulada em amplitude recuperando a informação de distância medindo a diferença de fase entre o sinal enviado e o recebido [28]. Câmeras com sensores de tempo de voo usam tecnologia CMOS para a captação da informação. Elas trabalham com um arranjo de sensores CMOS que captam o pixel necessário para cada parte da imagem.

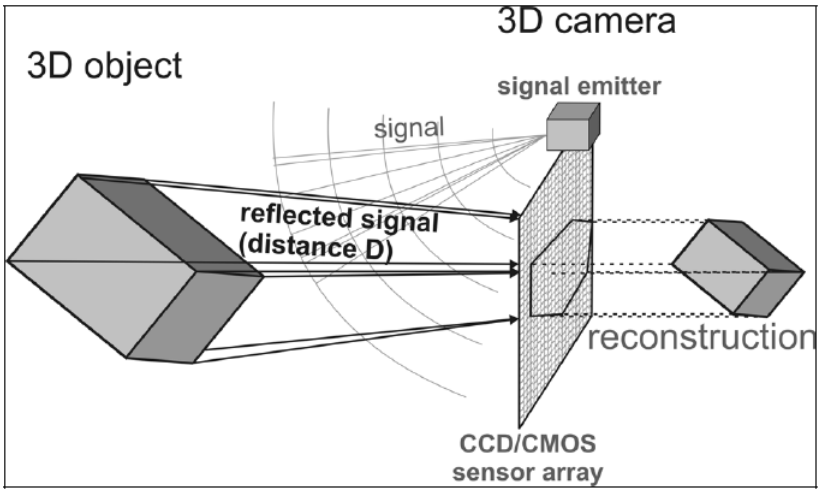

Figura 2.5: Principio de Funcionamento para Câmera CMOS[5]

A tecnologia LIDAR, que se baseia em Tempo de Voo, ou Time of Flight (ToF), é de grande utilidade para medição de distâncias, tanto em curto quanto em longo alcance. Seu nome vem do termo Light Detection and Ranging. Essa tecnologia usa, tanto o tempo de voo da onda de luz lançada, como o seu comprimento de onda no retorno, comparado com a

onda que foi enviada. Sensores LIDAR são usados em larga escala para o escaneamento de ambiente por veículos aéreos. Com o uso dessa tecnologia é possível mapear ambientes de forma rápida e com uma boa qualidade. Podemos ver os usos de sensores LIDAR na figura 2.6.

Na figura retirada pelo National Oceanic and Atmospheric Administration do U.S. Department of Commerce, podemos ver uma imagem mostrando como o escaneamento do ambiente ficou (parte da esquerda). Nela vemos que o relevo e profundidade pôde ser especificado pelo escaneamento. Também temos à forma como os dados foram coletados. Por um veículo aéreo sobrevoando a área desejada.

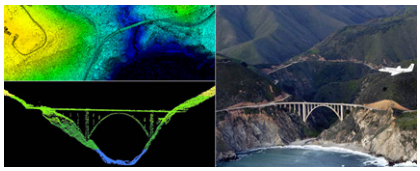

Figura 2.6: Escaneamento Aéreo 3D[43]

Em 2010 a Microsoft lançou ao mercado um sensor de movimento adaptado para se conectar à plataforma Xbox para jogos virtuais. Esse componente eletrônico permite que usuários da plataforma interajam com o ambiente virtual sem a necessidade de outros aparelhos conectados ao Xbox (controles). Isso é possível devido aos componentes do Kinect. Sua câmera RGB, seu sensor infravermelho de profundidade, seu microfone, seu processador e software. Com o uso de suas câmeras e seu sensor de profundidade, o Kinect é capaz de criar point clouds com uma alta qualidade, próxima a de sensores específicos.

Figura 2.7: Kinect para XBOX One[38]

Por conta de custos, o sensor escolhido para este trabalho foi o VL53L0X, um sensor de nova geração que utiliza a tecnologia ToF, ou tempo de voo. Ele é de proporções reduzidas, o que facilitará na tarefa de montar um escâner 3D. Por mais que suas proporções sejam reduzidas, sua acurácia é alta até mesmo em objetos que possuam superfícies reflexivas. O sensor pode medir distâncias absolutas até 2 metros, podendo ser aplicado em diversas situações. Ele é composto por uma SPAD (Single Photon Avalanche Diodes), um fotodetector que funciona por corrente de avalanche. Por conta dessas características espera-se que o sensor VL53L0X consiga trazer resultados satisfatórios.

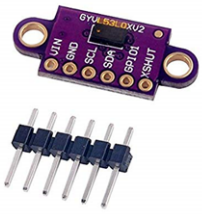

Figura 2.8: Sensor de distância VL53L0X[3]

2.2 Reconstrução 3D

Um método muito comum para reconstrução de objetos a partir de nuvens de pontos é a reconstrução de superfícies por proliferação de malha. Esses métodos criam uma superfície a partir de um triângulo inicial. O triângulo se propaga através da criação de outros triângulos ligados a ele, tendo que satisfazer alguns critérios geométricos[33].

Um algoritmo para reconstrução de nuvem de pontos consiste nos seguintes passos: cálculo de pontos vizinhos de cada ponto, construção do triângulo inicial, proliferação incremental da malha até a reconstrução completa da superfície.

Primeiramente, para se descobrir os pontos vizinhos, é necessário que sejam estabelecidos e analisados alguns critérios geométricos. Esses critérios são: proximidade, coplanaridade e regularidade.

A proximidade, como o nome já diz, é o cálculo dos pontos mais próximos ao redor do ponto analisado.

A coplanaridade é o critério de maior importância para a escolha do próximo triângulo a ser colocado na malha. Para isso trabalha-se com o conceito de vetor de proliferação[50]. Esse vetor é um vetor coplanar com o respectivo triânguloda frente de expansão e é perpendicular a própria aresta e se orienta para o seu exterior.

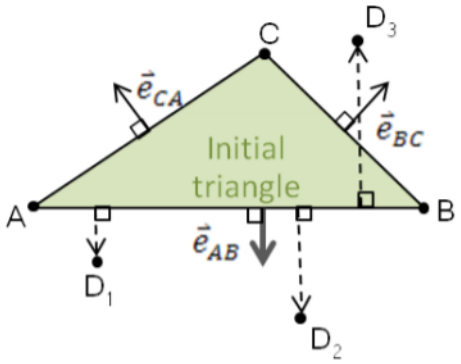

Figura 2.9: Vetores de Proliferação do Triângulo Inicial[33]

A regularidade da malha se relaciona com a criação de uma malha que de triângulos que são tendencialmente e desejavelmente equiláteros. Um triangulo mais regular, ou seja, que é

mais próximo de um equilátero, é o que a soma de seus cossenos internos seja maior.

Tendo definido os critérios de qualificação dos pontos da malha, o primeiro passo do algoritmo será o de encontrar, para cada ponto, uma vizinhança de pontos. O conjunto de pontos próximos é formado por uma reunião de subconjuntos. Exemplificando: dado um ponto A o primeiro subconjunto serão os X pontos mais próximos do ponto A. X é um valor que pode variar de acordo com as formas da malha a ser criada.

Tendo a vizinhança de cada ponto, pode-se então definir o triângulo inicial. O triângulo inicial deve ser tangencial à superfície curva imaginaria que interpola os pontos. Assim a reconstrução não ficara comprometida. A formação do triangulo inicial é dada da seguinte forma. Dado dois pontos, sendo eles os mais próximos de si em um conjunto de pontos, forma-se um segmento de reta conectando ambos. Dado o segmento de reta, encontra-se o ponto mais próximo desse segmento de reta. Estes três pontos formarão o candidato a triângulo inicial. Para que ele seja o triângulo inicial é necessário que todo o restante de pontos vizinhos aos pontos desse triângulo estejam definidos nos semiespaços definidos pelos vetores de proliferação de cada aresta.

Definido o triângulo inicial, é dado início ao processo de crescimento da malha. O principio que orienta o crescimento da malha é o de que a malha deve avançar no sentido de maior coplanaridade entre o vetor de proliferação do triangulo já formado com o plano do novo triângulo a ser formado. Assim a progressão da malha se da pelas regiões de menor curvatura.

Por fim, o que se tem é algo próximo ao do exemplo da figura 2.10.

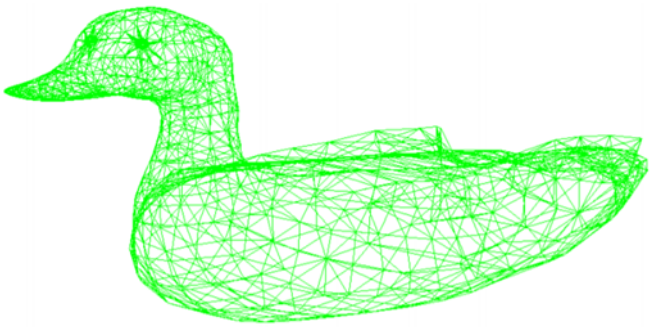

Figura 2.10: Exemplo de malha triangular[33]

2.3 Point Cloud

Um Point Cloud pode ser interpretado como um espaço cartesiano com três dimensões e, nesse espaço, existirão diversos pontos com suas coordenadas. Quando tomamos todas as medidas de um objeto, temos apenas os dados coletados através de sensores, para que possamos visualizar o objeto escaneado no ambiente virtual, os dados coletados precisam ser processados.

A geração de Point Clouds é relativamente simples. É apenas necessário que as coordenadas estejam especificadas e uma imagem tridimensional pode ser gerada através dela. O problema é que para interpretar as informações de um arquivo que contém um Point Cloud, devemos saber o que cada parte significa. A modelagem tridimensional não é algo novo, por isso existem diferentes formatos de arquivos de Point Clouds, criados por diferentes desenvolvedores para propósitos diferentes.

Para o caso de um escaneamento que leva em seus parâmetros dimensionais as informações de cor, temos um voxel. De forma simplificada, um voxel pode ser explicado como o objeto similar ao pixel, só que para um caso tridimensional. Sendo assim, um voxel terá uma informação de espaço e uma informação de cor. O voxel já é utilizado para o escaneamento de ambientes com a tecnologia LIDAR[25]. Utilizando o método de armazenamento de informação de cor, fica mais fácil de gerar um modelo digital do objeto. Sabendo as cores e suas tonalidades, é possível processar os dados para ter uma melhor noção da profundidade. Como é o caso do Kinect do XBOX. Esse hardware faz o uso de um sensor de distância e uma câmera para captar as cores[27]. Usando ambos juntos ele consegue fazer um processamento preciso de tridimensionalidade.

Os outros principais tipos de arquivos para armazenamento 3D são: PLY, um formato de arquivo poligonal criado na universidade de Stanford, outro processo é o STL, um formato de arquivo CAD para estereolitografia[40]. Os outros dois mais conhecidos são OBJ e X3D. O primeiro é um formato de arquivo criado pela Wavefront Technologies e, o segundo um arquivo padrão livre de royalties para representação de arquivos tridimensionais.

Para o projeto em questão foi escolhido o modelo PDC (Point Cloud Data). Esse formato é genérico criado para complementar os formatos já existentes que não possuíam suporte para PCL (Point Cloud Library). Além de ser genérico, esse formato também é compatível com MatLab.

A biblioteca matpcl é puramente feita de funções do MatLab e é uma interface especifica para trabalhar com ferramentas de PCL lendo ou escrevendo arquivos no formato PDC. Nessa biblioteca existem poucas funções de usuário. Elas são: savepcd(), loadpcd(), pclviewer() e lscpd().

A função savepcd() escreve uma matriz, tendo a opção de salvar com uma matriz de cores, um point cloud no formato ASCII PCD. Loadpcd() lê um arquivo PCD, seja em binário ou ASCII, e retorna uma matriz. Pclviwer() escreve uma matriz em um arquivo temporário e inicializa o programa pcl viewer para que o formato seja visualizado. Por fim, a função lscpd() mostra os atributos do arquivo PCD no diretório.

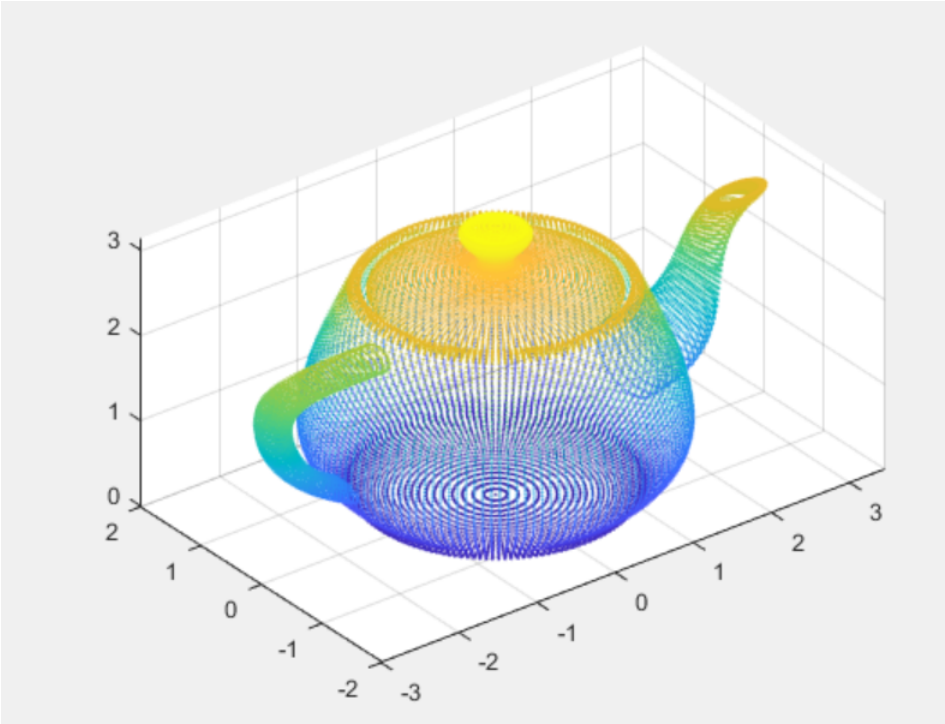

Figura 2.11: Exemplo de leitura de Point Cloud pelo MatLab

O formato de um arquivo PCD possui um cabeçalho com diversas informações sobre o arquivo. Um programa capaz de ler um arquivo PCD precisa de todas as informações do cabeçalho para processar de forma correta o arquivo.

Nesse formato, primeiramente vem o cabeçalho. Nele será especificado todo o formato do arquivo. Abaixo será especificada cada parte do cabeçalho:

- **Version:** Nesta parte do cabeçalho deve ser colocada a versão do arquivo que está sendo gravado. A representação da versão escrita deve estar no formato 0.x, sendo 'x' o número da versão sendo utilizada.

- **Fields:** Nesse espaço do cabeçalho deve ser especificado todos os campos que o Point Cloud pode ter. Ele é necessário já que um Point Cloud pode conter mais do que as informações de distância do objeto. É possível colocar também a cor, as normais da superfície, além de outros dados que podem ser importantes dependendo do uso.

- **Size:** Essa variável do cabeçalho especifica o tamanho que cada dimensão tomada em bytes.

- **Type:** Aqui é especificado o tipo de cada dimensão. Os valores que podem ser colocados são I para tipos signed como char, short e int. U é escolhido para representar tipos unsigned. F define os tipos float.

- **Count:** Essa variável serve para especificar quantos elementos cada dimensão do Point Cloud tem. Para caso estudado, todas as dimensões terão um único elemento.

- **Width:** Serve para especificar quantos pontos o Point Cloud deve ter por coluna. Também pode ser utilizado para especificar a quantidade total de pontos do Point Cloud.

- **Height:** Especifica a quantidade de colunas que o arquivo pode conter. O valor default dessa variável é 1.

- **Viewpoint:** Especifica a quantidade de pontos de captura de dados diferentes que podem ter no arranjo. A princípio não há muita utilidade para o projeto. Mais tarde, se for necessário utilizar mais de um sensor para aumentar a precisão de captura dos dados.

- **Points:** Especifica a quantidade de pontos que o Point Cloud terá.

- **Data:** É a ultima variável a ser colocada e ela especifica o tipo de dado que está presente no arquivo. Ela existe porque algumas versões suportam mais de um tipo de dado. A versão 0.7 suporta os dados do tipo ascii e binário.

As grandes vantagens do PDC como formato de Point Cloud são a sua capacidade de armazenar e processar dados organizados de Point Clouds. A possibilidade de ter dados armazenados no formato binário, que são o tipo de informação com maior velocidade de serem salvos na memória.

Com o aumento da necessidade de captura de imagens tridimensionais, de mais em mais surge a necessidade de uma alta qualidade em captura e processamento de arquivos contendo point clouds. Para isso surgem algoritmos que são mais eficientes em diferentes processos de processamento de point clouds para a imagem real.

2.3.1 Tipos de causadores de erros em nuvens de pontos

Para construir um bom modelo tridimensional de um objeto em um ambiente virtual, é necessário que os dados de sua captura tenham o menor número de erros possíveis. Por conta de fatores incontroláveis, sempre haverá uma quantidade de erros em um arquivo com dados para uma imagem em ambiente virtual [7]. O mais comum desses erros é o ruído, que não pode ser controlado, apenas reduzido. O ruído acarretará desvios nas medidas tomadas. Uma calibração eficiente evitará que o ruído cause desvios muito grandes.

Um outro causador de erros são pontos chamados de Ouliers, que no caso representam dados coletados que possuem valores aberrantes em relação aos outros pontos. Ocorrem muitas vezes por conta de problemas no sensor ou na estrutura física, que pode se mover de forma a causar pequenos erros. Fazendo uma correlação com os pontos já existentes é fácil apagar esses erros.

Erros por desalinhamento também são comuns e ocorrem muitas vezes por má calibração do aparelho ou o uso de outra calibragem para medidas que serão correlacionadas futuramente.

2.3.2 Tipos de entradas

Para um arquivo que contenha dados para uma imagem 3D é possível colocar diversos tipos de dados em sua entrada para que cada um seja processado para algum tipo de característica da imagem.

Colocar os dados referentes às normais da superfície escaneada no arquivo permite que mais detalhes sejam processados. As normais de superfície servem para ajudar a obter a orientação de para onde os próximos pontos estarão alocados. Isso ajuda a texturizar superfícies e determinar melhor detalhes curvos na imagem.

Um outro tipo de input que pode ser colocado em um arquivo de imagem 3D é a cor no formato RGB. Ter um sensor de cor ligado a uma câmera é muito útil para conseguir captar profundidade precisamente.

2.4 Caracterização dos componentes utilizados

2.4.1 Corpo do Scanner

A estrutura física do escâner será feita em madeira prensada, na forma de um cubo vazado, com apenas dois lados fechados. Em uma das faces será colocado o sensor, em uma esteira para que possa se mover na vertical. Ao centro estará a plataforma em que o objeto a ser escaneado será colocado. Esta plataforma será de forma circular e será ligada a um motor de passos responsável pela sua rotação.

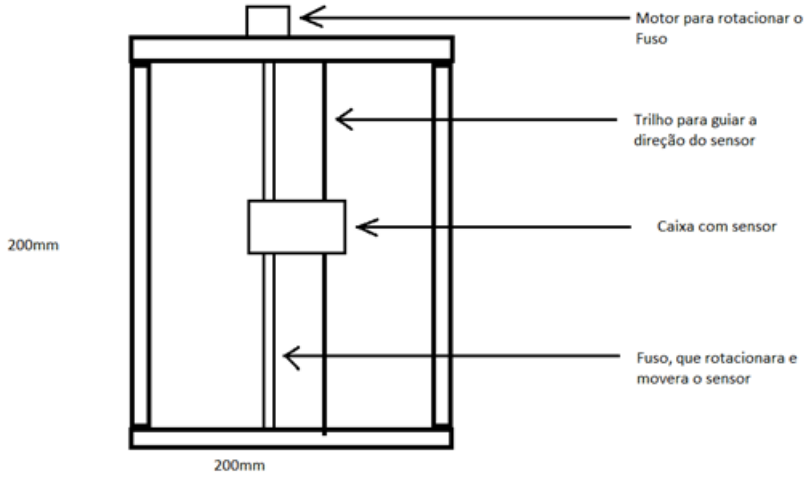

Figura 2.12: Esquema do Corpo do Scanner

2.4.2 Motor de Passos

O motor da base será um motor de passos NEMA 17. O NEMA 17 é um motor de passos hibrido e seu nome vem do fato de NEMA ser a associação de equipamentos elétricos e imagens médicas nos Estados Unidos. Essa associação prevê as normas que os motores de passo devem seguir.

Como já falado, o NEMA 17 é um motor híbrido. Os outros tipos de motores são os de Relutância Variável e os de Imãs Permanentes. Os motores do tipo relutância variável não são comuns pois não possuem imãs permanentes. Seu rotor é de ferro com um alto índice de pureza e seus campos magnéticos são formados pelos enrolamentos no estator e são alimentados por corrente continua. O circuito de controle desses motores é mais complexo por conta de não possuírem um imã permanente. Isso faz com que seu uso seja reduzido.

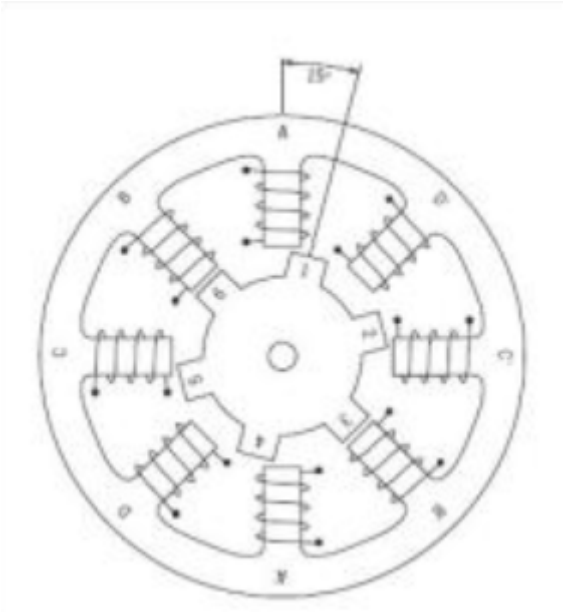

Figura 2.13: Núcleo do motor de passos de relutância variável[30]

Motores de Passo de Imã Permanente são mais baratos e com menor precisão. Seus ângulos de passo variam entre 7,5 e 15 graus por passo. Sendo assim são motores que possuem poucos passos (de 48 a 24 passos). Eles são comumente encontrados em periféricos de computadores. O nome é devido ao fato desses motores possuírem um imã permanente em seus circuitos. Isso simplifica seu circuito, mesmo que o deixando mais devagar e com menos torque, seu preço e velocidade de manufatura compensam isso.

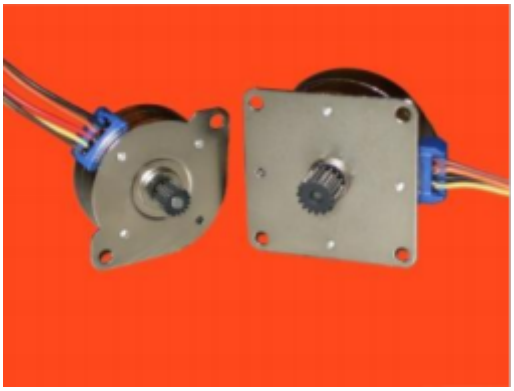

Figura 2.14: Motores de passo com imã permanente

Motores de Passo Híbridos, como o nome sugere, mesclam os princípios operacionais dos outros dois tipos de motores. Sendo assim são motores mais sofisticados. Eles possuem uma boa relação velocidade torque, além de terem um número grande de passos. Motores híbridos podem fazer passos de 3,6 a 0,9 graus por passo, ou seja, de 100 a 400 passos. O seu rotor é multidentado, como o motor de relutância variável, e possui dois imãs polarizados presos ao seu eixo. Isso permite que o motor tenha melhores características de torque e travamento. Sua alta precisão faz com que seja o tipo de motor mais utilizado em fresadoras e impressoras 3D.

O NEMA 17 foi então escolhido para o projeto por conta de sua quantidade de passos para completar uma volta. Sua capacidade de passos por volta de 200 passos, o que nos dá uma precisão de 1,8 graus por passo.

Já que procuramos uma boa precisão no escaneamento de um objeto, temos de ter a capacidade de tomar medidas muito próximas umas das outras.

Figura 2.15: Motor de passos Nema 17[26]

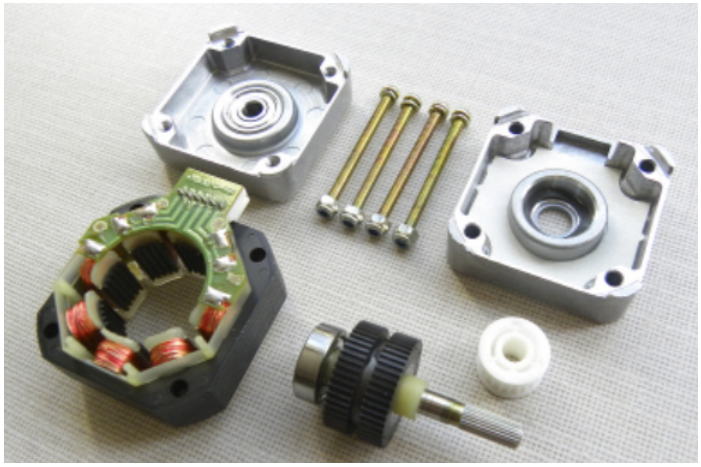

Figura 2.16: Estrutura interna do motor[32]

O motor deve ser alimentado com uma tensão entre 9 e 30 volts com uma corrente de até 2 amperes. Por conta a grande necessidade de energia para que o motor funcione, foi necessário o uso da entrada de alimentação externa do Arduino. A fonte escolhida foi um transformador 220V – 12V com capacidade máxima de corrente de 2,5A.

Figura 2.17: Fonte de alimentação para o Motor

Para se medir o motor de passos foi utilizado um transferidor de precisão de 180 graus e um pequeno pedaço de papel (cartolina). A cartolina ficou acoplada ao rotor do motor. Ela serviu como uma referência para a rotação do motor em relação ao transferidor.

O motor foi colocado sobre o transferidor e foi garantido que o motor ficasse nivelado em relação ao transferidor. Com a cartolina presa a seu rotor, o motor foi colocado para rotacionar. Por conta de o mínimo ângulo de rotação do rotor ser muito pequeno (1,8 graus), não foi possível observar se a rotação para conferir se este era o ângulo rotacionado.

Para conseguir conferir a precisão da rotação, o motor foi testado com ângulos maiores.

Os ângulos testados foram: 90, 60, 45, 30, 10 e 5 graus. Em todos a precisão foi bem satisfatória, pois não foi possível observar erros de 1 grau ou mais.

2.4.3 Controlador do Motor de Passos

Para facilitar o uso do motor, será usado um micro controlador especifico para motores de passo. O A4988 é um driver de micro passos para motores de passo. O modulo A4988 é o mais usado em fresadoras. Sendo assim seria a escolha lógica a se fazer para os primeiros testes.

O controlador A4988 facilita o controle do motor por conta dos diversos modos de operação que ele oferece para controle do motor. Por mais que pequeno, o Módulo Driver A4988 é extremamente complexo.

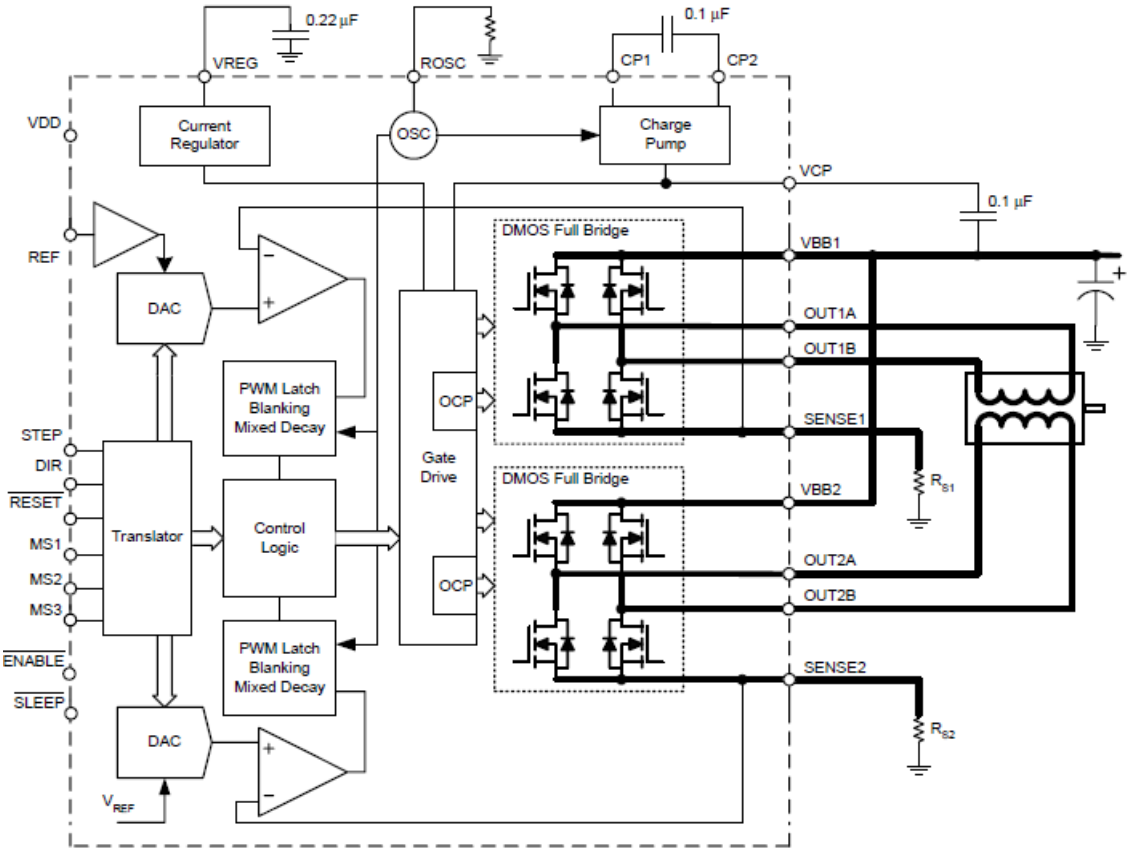

Figura 2.18: Diagrama de blocos funcional[2]

O chip que faz o controle do motor, o A4988, possui duas pontes H com transistores FET DMOS para controle de motores de passo bipolares.

O chip é versátil e resiliente. Possui proteção contra curtos em suas saídas e proteção contra excesso de temperatura.

O Driver permite cinco controles de passos. Passo completo, meio passo, um quarto de passo, um oitavo de passo e um dezesseis avos de passo. Esse tipo de controle é chamado de microstepping.

Com o controlador não precisamos fazer um controle preciso de como devem ser feitos os passos do motor, o controlador já o faz através de uma opção já escolhida previamente.

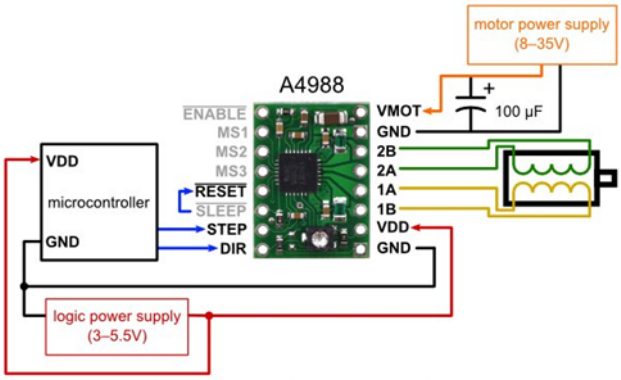

Figura 2.19: Ligação do Controlador A4988[31]

Pinos do A4988:

- **STEP:** Um sinal de subida fará com que o motor avance um passo.
- **DIR:** Define a direção de rotação do motor.
- **ENABLE:** Um sinal baixo ativa os drivers do motor.
- **RESET:** Desativa todos os drivers de saída dando um reset no chip.
- **SLEEP:** Desabilita os circuitos internos para economizar energia.
- **MS1, MS2, MS3:** Essas entradas selecionam o modo de microstepping que o motor seguirá.
- **OUT1A, OUT2A, OUT1B, OUT2B:** Pinos de saída do driver que são conectados diretamente ao motor.
- **SENSE 1 e SENSE 2:** Pinos de controle de corrente para as duas pontes H.

Resolução Micro-passo	MS3	MS2	MS1
Passo completo	0	0	0
meio passo	0	0	1
1/4 passo	0	1	0
1/8 passo	0	1	1
1/16 passo	1	1	1

Figura 2.20: Controle de microstepping[31]

2.4.4 Unidade de Processamento

Devida a complexidade do projeto se comparada à experiência com o tipo de hardware e à programação necessária. O Hardware para fazer o processamento dos dados coletados e a comunicação com o computador para o envio dos dados, foi o Arduino MEGA 2560.

Por conta de trabalhar com uma linguagem de alto nível e ser uma plataforma de trabalho relativamente comum, o Arduino reduz a complexidade do projeto.

O Arduino é conhecido por ser uma ferramenta versátil e possuir uma grande biblioteca de programas para as mais diversas funcionalidades. O Arduino, nesse projeto teria a função de passar os dados para a porta serial do computador, controlar os motores e tomar as medidas com o sensor de distância escolhido.

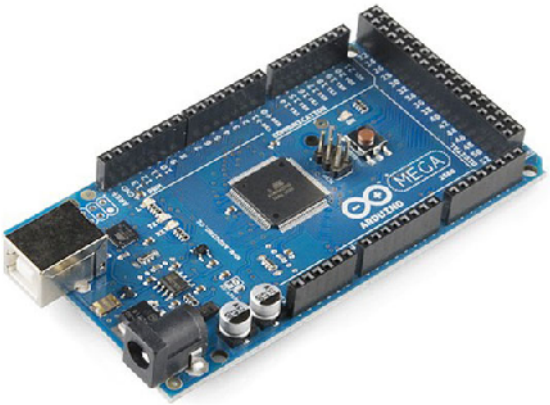

Figura 2.21: Arduino MEGA 2560[18]

A biblioteca de Arduino utilizada para a comunicação serial com o sensor de distância foi a Wire.h. Essa biblioteca permite a comunicação serial, de forma I2C. Essa comunicação é visível pois os pinos de ligação do sensor VL53L0X são o SDA e o SCL, Serial Data e Serial Clock. Nessa biblioteca temos as funções[4]:

- **begin():** Esta função inicializa a biblioteca Wire no Arduino. Normalmente é chamada apenas uma vez. Quando chamada, se especificado um endereço entre os parênteses, o Arduino será definido como escravo. Se não houver endereçamento, o Arduino será mestre.

- **requestFrom():** Essa é uma função de uso exclusivo do mestre. Ela é endereçada a um escravo e faz um pedido de um número determinado de bytes. Ela pode vir com um parâmetro de parada ou não. Caso não haja o parâmetro de finalização, o mestre, continuamente, estará requisitando os bytes do escravo.

  A função pode ser escrita da forma: Wire.requestFrom(address, quantity, stop), ou da forma Wire.requestFrom(address, quantity).

- **beginTransmission(address):** Começa a transmissão para o escravo de endereço determinado na função.

- **endTransmission():** Finaliza a transmissão para o dispositivo escravo e transmite os bytes que estavam enfileirados pela função write().

- **write():** Escreve os dados de um escravo, quando requisitado pelo mestre, ou enfileira os bytes para uma transmissão vinda do mestre para um dispositivo escravo.

- **available():** Retorna o número de bytes disponíveis para serem requisitados. Normalmente deve ser usada, por um mestre, após o uso da função requestFrom(). Pode ser usada por um escravo após o uso da função onReceive.

- **read():** Lê um byte transmitido por um escravo para o mestre depois da chamada da função requestFrom(). Também pode ler os bytes transmitidos de um mestre para um escravo.

- **setClock():** Essa função modifica o clock I2C do Arduino. O valor base do clock é de 100kHz. Nem todo valor é vâlido para ser definido. Valores comuns são: 100000(para modo normal) e 400000(para modo rápido). Alguns processadores também suportam as velocidades 1000000(1MHz) e 3400000(3,4MHz). Em caso de dúvidas, a documentação do processador deve ser estudada.

- **onReceive():** Registra uma função para ser utilizada quando o escravo recebe uma transmissão do mestre.

- **onRequest():** Registra uma função para ser utilizada quando o escravo recebe um pedido do mestre.

Para enviar as informações para o computador foi usada a função println() com o prefixo Serial. Tendo o endereço da porta serial definido, colocando essa função, o Arduino irá enviar para a porta serial tudo que for definido por esta função.

2.4.5 Sensor de Distância

O sensor de distância escolhido, como já falado, foi o vl53l0x da Adafruit. Ele é um sensor do tipo ToF que trabalha com luz não visível. O código para o uso do sensor, pelo Arduino, foi adaptado de códigos fontes encontrados no GitHub[13]. Sua capacidade é muito alta e suas formas de captação de medidas de distância são diversas. Para o projeto em si, estaremos utilizando pouco de seu potencial. Visando a simplificação do projeto, esperamos que seja possível usá-lo para captação única e fazer um loop para que seja possível repetir a medida.

2.4.5.1 Calibração

O sensor é de elevada complexidade e por conta disso, diversas calibrações são necessárias. Suas calibrações variam dependendo da distância em que o alvo se encontra a superfície do alvo e a iluminação do ambiente em que o sensor se encontra.

A maior parte das calibrações é feita na fabricação do sensor e só devem ser repetidas em circunstâncias específicas. Dependendo das condições da tomada de medida, existe a opção de fazer o carregamento de parâmetros específicos para aquela situação.

Após um reset, alguns passos devem ser tomados para que o sensor possa ser utilizado novamente. São elas: Inicialização e carregamento dos dados de calibração, Medição e Digital Housekeeping.

O Digital housekeeping é a última operação interna ao dispositivo que computa uma medição de distância e, dependendo de seu valor, a valida ou a rejeita.

Muitas das funções que são chamadas no período de inicialização do dispositivo precisam apenas ser chamadas e corretamente temporizadas. Elas são todas pré-programadas e o dispositivo as reconhece de fábrica.

Após a inicialização do dispositivo é possível efetuar uma ou mais medidas, dependendo de como o sensor será configurado.

Para sua inicialização, o dispositivo passa por algumas etapas em que chama funções internas. Na primeira etapa de inicialização chamamos duas funções: DataInit() e StaticInit(). Ambas as funções citadas realizam a inicialização interna do dispositivo, sendo que StaticInit só é chamada após a função DataInit ser executada. StaticInit é uma função que permite o carregamento de parâmetros específicos para um caso especifico que o sensor será utilizado.

Após a inicialização interna do sensor, ocorre a calibração dos diodos SPADs através da função PerformReferenceSPADManagement(). Essa calibração precisa ser feita apenas na etapa de fabricação do sensor e seus dados são armazenados no próprio sensor. Ela serve para guardar os dados de como os SPADs captam as ondas de luz para que a distância possa ser medida. Em outras palavras, as possíveis não linearidades e pequenos erros que podem vir de fabricação. Com esses dados coletados os erros e incertezas podem ser compensados.

Em seguida é feita uma calibração de temperatura, utilizando a função PerformRefCalibration(). Aqui é feita uma calibração em partes do sensor que são afetados por variações na temperatura externa, tais como VHV[47] e a calibração de fase do sensor. Esse tipo de calibragem funciona para ajustar a sensibilidade do sensor. Ela é feita durante o processo de fabricação e deve ser repetida caso a temperatura varie 8 graus Celsius da temperatura em que a última calibração foi feita.

Seguidamente é feita a calibração de Offset. Para executar essa calibração, é necessário que se use um alvo branco de 88% de refletividade a 10 centímetros do sensor, em um ambiente escuro. Atendendo as condições necessárias é chamada a função PerformOffset-

Calibration(). O valor de retorno dessa função é em micrômetros e é armazenada na memória Host.

Por fim ocorre a calibração para múltiplos caminhos, ou Cross Talk calibration. Esse tipo de calibração serve para compensar o erro quando o sinal de luz do sensor passa por algum vidro. Esse tipo de acontecimento pode afetar o resultado da distância medida. Sendo assim a calibração de Cross Talk fará uma compensação para o erro, como podemos ver na figura 2.21.

Figura 2.22: Compensação Crosstalk

Abaixo temos um diagrama de inicialização do dispositivo com os devidos tempos de atraso para a execução de cada função, na figura 2.22.

2.4.5.2 Modos de Funcionamento

O VL53L0X possui formas diferentes de coletar os dados de distância. Ele tem a opção de fazer uma medida e parar, ou de fazer diversas medidas, dando também a escolha de quanto tempo esperar entre as medidas a serem tomadas.

Dependendo da necessidade, o sensor, pode tomar medidas de diferentes formas. A primeira, e mais simples, é a função VL53L0XStartMeasurement(). Essa função permite ao sensor realizar uma medida, mas não guarda a informação. Ele é interessante de se usar em um sensor de distância para um acompanhamento em tempo real, já que ele não guarda os dados.

O segundo tipo de medida é o VL53L0XPerformSingleMeasurement(). Essa função realiza uma medida e guarda a informação coletada. Para saber se a informação sobre a distância foi tomada, as opções polling ou interrupção devem ser selecionadas.

Por fim temos a função VL53L0XPerformSingleRangingMeasurement(). Ela repete a segunda função apresentada anteriormente e, além de transmitir a informação, ela reseta a flag de interrupção, deixando assim o sensor pronto para tomar uma nova medida.

Caso o usuário esteja utilizando o sensor no modo continuo, basta usar a função VL53L0XStopMeasurement() para que as medidas parem de ser tomadas.

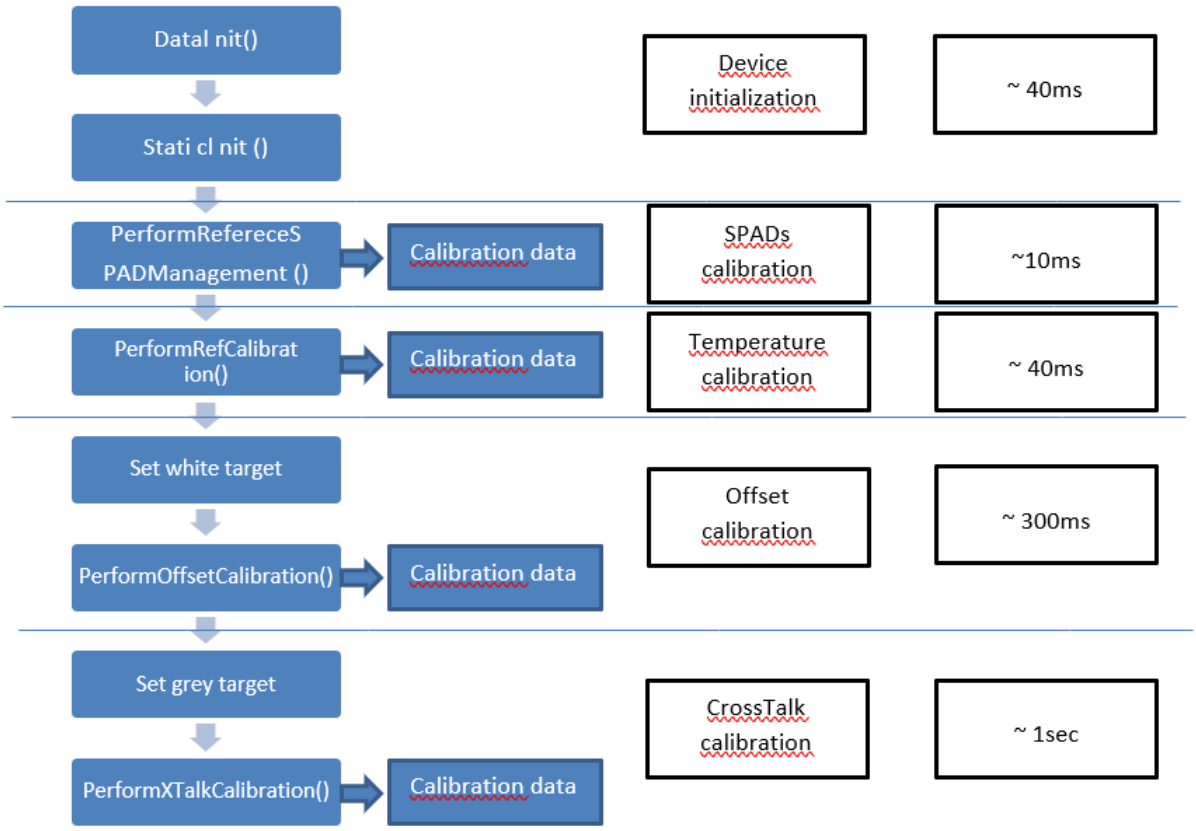

Figura 2.23: Diagrama de Inicialização

Sabendo que é possível começa a tomar medidas de diferentes formas e também e possível tomar medidas de forma continua ou pausada, as possibilidades para como executar uma medição utilizando o sensor se divide em quatro. Podemos tomar as medidas singulares, ou pausadas, por interrupção ou polling e podemos fazer o mesmo com a tomada de medidas continuas.

Para o tipo de medição contínua, como já dito antes, podemos fazer por polling ou por interrupção. A escolha entre esses dois tipos de método pode variar com o hardware ou a finalidade do projeto. Para o caso do escâner foi usado o método continuo por polling, que pode ser entendido pelo diagrama na figura 2.23.

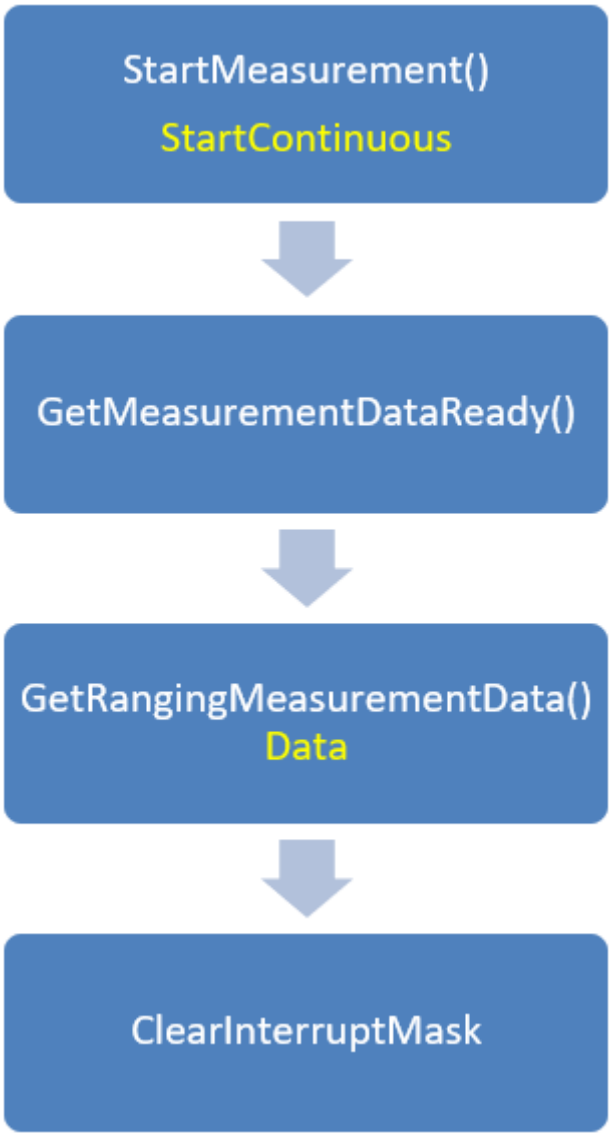

Figura 2.24: Diagrama de Medição Contínua por Polling

O caso da função ClearInterruptMask() é o de limpar a interrupção interna gerada para que o sensor realize a medida. Ela não está relacionada ao fato da medida ser ativada por polling ou interrupção. A função GetMeasurementDataReady() permite que o status da atual medida seja verificado (se ela foi tomada ou ainda está acontecendo). A função GetRanging-MeasurementData() retorna o valor da medida tomada.

O método por interrupção é bem parecido. Durante o processo só é adicionado um passo a mais no diagrama, o de limpar a flag de que houve a interrupção para que uma medida fosse tomada. Agora não há mais a necessidade de haver a checagem se houve ou não medida, por isso a função GetMeasurementDataReady() é suprimida. O diagrama da figura 2.24 exemplifica como se dá o sequenciamento das funções.

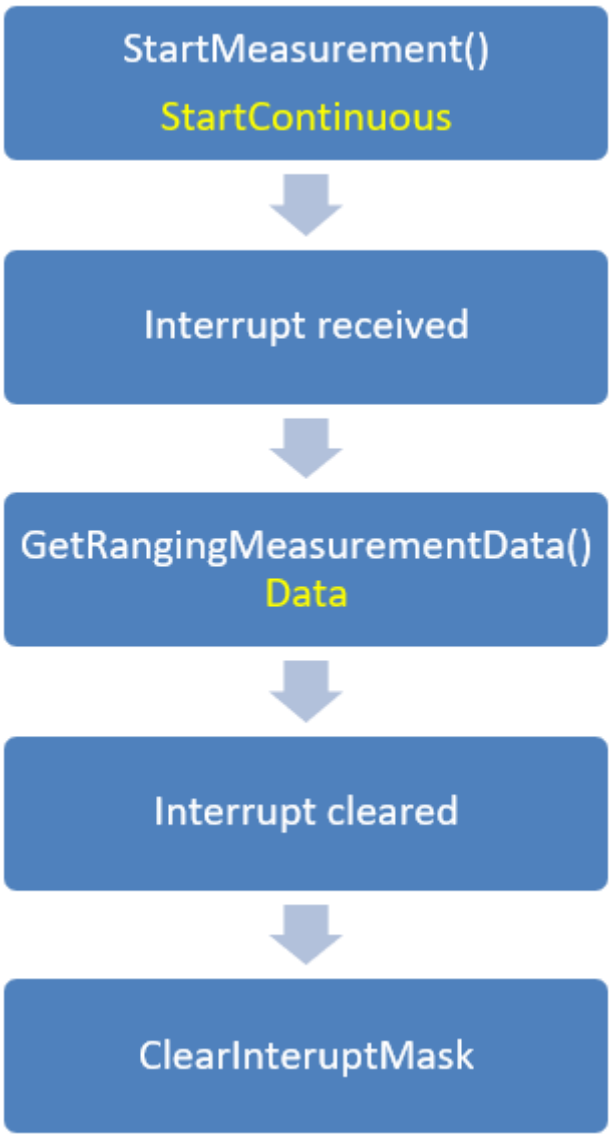

Figura 2.25: Diagrama de Medição Contínua por Interrupção

Para o caso do tipo de medição pausada seu polling pode ser feito de três formas diferentes e seu modo de interrupção é igual ao do método contínuo. A única diferença vem no modo de tomar medidas.

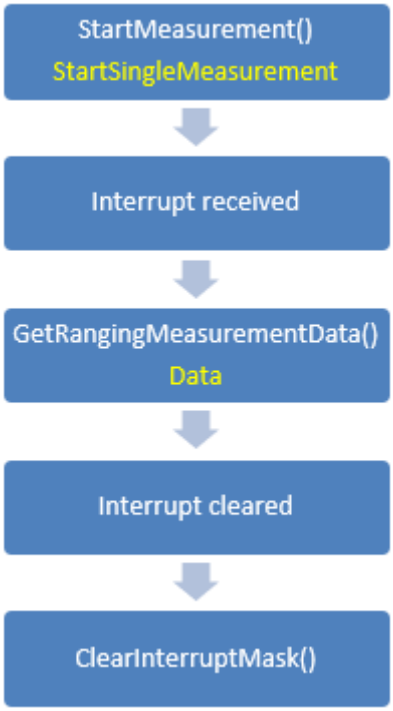

Figura 2.26: Diagrama de Medição Pausada por Interrupção

Como podemos ver seguindo o diagrama da figura 2.25, a única etapa adicionada é a do recebimento da interrupção e a mudança que ocorre na primeira etapa é a função que é chamada é a StartSingleMeasurement().

Para os modos de polling no método de medida pausada duas são feitas pela API do sensor e outra pode ser feita pelo controlador. As que são feitas pela API fazem o uso de funções internas que deixam o ciclo de tomada de medida automático. Caso seja desejado ter um maior controle sobre as funções chamadas é aconselhado usar o método de polling pelo controlador. Usando o método das API não temos acesso direto aos dados e às flags que estão ativas.

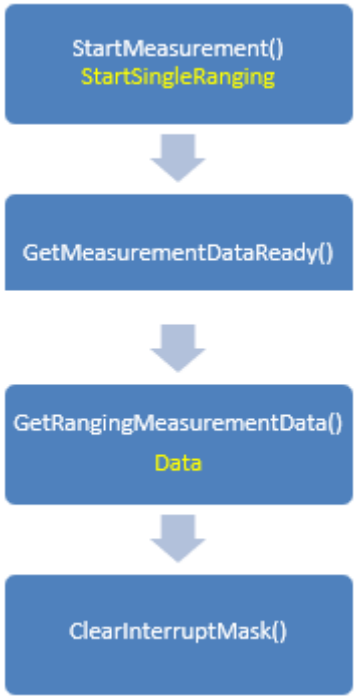

Figura 2.27: Diagrama de Medição Pausada por Polling Controlada pelo Host

Figura 2.28: Diagrama de Medição Pausada por Polling Controlada pela API

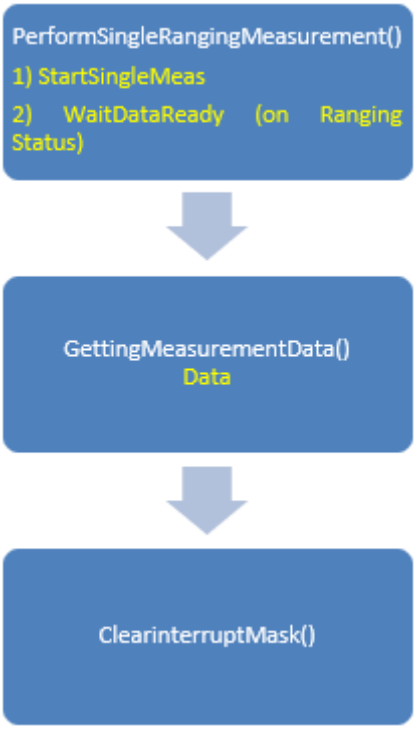

Figura 2.29: Diagrama de Medição Pausada por Polling Controlada pela API

2.5 Simulações Teóricas

Para realizar testes com o MatLab, alguns arquivos PCL foram abertos e simulados. Assim é possível ver diversas características de um arquivo PCL feito em um ambiente real.

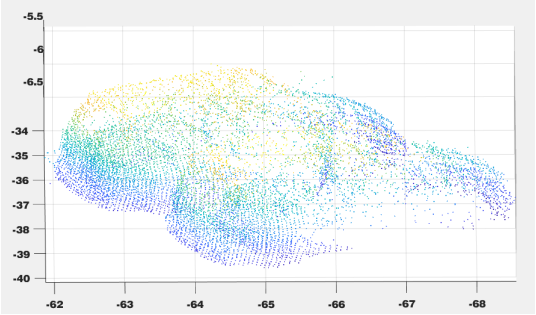

Figura 2.30: Imagem de Carros Escaneados

Como podemos ver na foto acima, o arquivo apresenta ruído. Isso torna mais complicado de visualizar a imagem que, no caso, são dois carros. Para que um arquivo PCL criado em um ambiente real possa ser simulado, é necessário que ele passe por filtragens e analises matemáticas, como o caso da Poisson Disk Sampling.

Na foto a seguir, por se tratar de um objeto menor, a qualidade ficou melhor já que com menos pontos já podemos identificar o objeto.

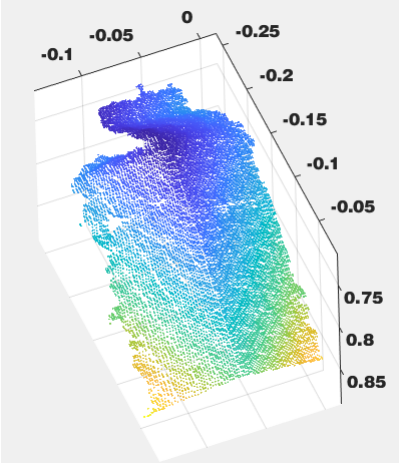

Figura 2.31: Caixa de Leite

Na figura a seguir, foi usado uma interface de cor, para que a imagem ficasse mais nítida.

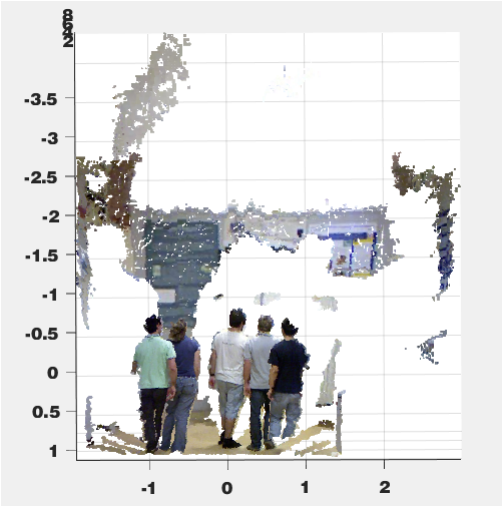

Figura 2.32: Pessoas em Sala

Como podemos ver, fica mais fácil de se analisar uma imagem quando, junto com as informações de distância, são colocadas as informações de cor. Através de programas específicos, a profundidades em um ambiente pode ser identificada através das sombras e luz.

Capítulo 3

Metodologia

3.1 Metodologia de Desenvolvimento

O projeto pôde ser dividido em diversas frentes. A parte que envolve o Motor de Passos e seu controlador, montagem da plataforma, o Sensor de Distância, a interface de Comunicação Serial e o processamento de arquivos PCD.

Para o controle do Motor de Passos foi necessário entender a biblioteca de controle do seu sensor, o A4988. O controlador possui uma grande complexidade por adicionar modos de controle ao motor. Como já falado, o motor tem capacidade de 200 passos, nos dando uma precisão de 1,8 graus por passo. Utilizando o controlador é possível chegar a fazer o sensor dar um oitavo de passo, deixando-o então com 1600 passos. Isso daria um grande aumento na precisão dos passos do motor para adicionar detalhes ao arquivo PCD, mas por ser um protótipo, esse aumento no número de passos foi julgado desnecessária.

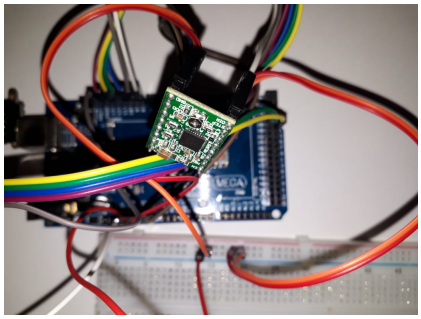

Figura 3.1: Driver Controlador para Motor de Passos A4988

A estrutura em que os componentes seriam colocados é uma estrutura cúbica, de madeira prensada. Suas faces são vazadas para que seja possível ver seu interior e mover objetos dentro dela. Ela foi montada na oficina do SG-11, na UnB. Além de todo esqueleto em madeira, também é necessário que exista uma base capaz de rotacionar. Essa base é encaixada no Motor de Passos, deixando fácil seu controle de rotação.

Figura 3.2: Estrutura externa do Escâner

O sensor de distância utilizado se inicializa sempre que for ligado, sendo assim não há a necessidade de preparar um processo de inicialização sempre que o sensor for ligado. A necessidade de uma inicialização acontece apenas quando a superfície do objeto a ser tomada a medida for espelhada ou se a temperatura do ambiente ao redor do sensor for pelo menos 8 graus Celsius maior, ou menor, do que a temperatura da última calibração.

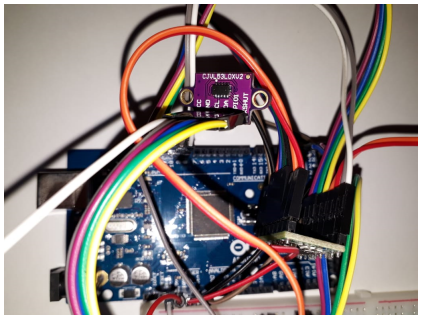

Figura 3.3: Sensor Conectado

Para que um arquivo PCD seja gerado, é necessário que ele apresente as características de tal. Isso é possível fazendo com que os dados lidos sejam transformados em um arquivo de texto que possa ser lido como se fosse um arquivo PCD. Isso se fez possível a partir da interface serial que inicializa um arquivo de texto com o cabeçalho necessário e lê os dados da porta serial e, logo em seguida os coloca no arquivo de texto.

3.2 Valores Tomados

Para garantir que fosse possível tomar medidas, foi calculado que a distância entre o sensor e o objeto a ser analisado deveria ser de no máximo 20 centímetros. Valores abaixo dessa distância possuíam um erro relevante e valores acima deixariam o escâner muito grande e então de difícil implementação.

Pelo fato de ser a implementação de um protótipo, não houve necessidade de escolher uma velocidade específica para a captação dos pontos. Levando em conta que não procuramos uma grande velocidade para digitalizar o objeto, não há necessidade de pegar um número mínimo de pontos. Sendo assim, a velocidade escolhida foi de 9600 bps para a leitura da Porta Serial. Com esse valor, mesmo sendo mais rápida que a velocidade de leitura do sensor, não há problema em analisar o mesmo ponto duas vezes. O problema se dá apenas se a leitura da porta fosse mais devagar que a do sensor, pois a leitura de pontos de distância seria perdida.

3.3 Simulações

Para que fossem tomados dados de distância e ângulo, o Arduino foi conectado ao computador e sua porta serial foi informada no código em C++ para a leitura da transmissão na porta. O sensor de distância e o driver de controle do motor estavam conectados ao Arduino.

O experimento foi feito em um ambiente com controle de luz e com o motor preso à mesa. Caso o motor fique sem encosto ou algo que o deixe imóvel, ele começará a se mover.

O sensor foi colocado a uma certa distância de uma barreira física e então o código foi inicializado. A distância foi tomada havendo em seus resultados pequenas imprecisões, muito provavelmente decorrentes de ruído. O motor não fornece erros em suas rotações e caso ele esteja bem fixado a uma superfície, a informação sobre o ângulo fornecido pela porta serial não apresentará erros.

Os dados coletados estão no próximo capítulo.

3.4 Sequenciamento de decisões

Para dar início ao projeto foi escolhido, como primeiro passo a caracterização e estudo aprofundado do sensor de distância. A escolha se deu por ser um componente novo e relativamente complexo para se trabalhar sem ter uma experiência prévia.

Primeiramente foi feito um estudo da biblioteca do sensor, para entender quais funções seriam necessárias para o projeto. Após o estudo das funções especificas foram executados testes para medir a precisão do sensor.

Com a precisão do sensor testada, o passo seguinte foi pensar em como seria a estrutura e como seus componentes se moveriam para que as medidas do objeto fossem possíveis de captar. Decidido então a forma da estrutura do escâner, seria necessário escolher e categorizar o motor que trabalharia na base do escâner.

O motor NEMA 17 atendia as necessidades do projeto devido à sua precisa e capacidade de carga, pois ele teria de ter força para mover a base e o objeto que fosse colocado sobre ela.

Tendo o motor e o sensor já preparados, seria necessário preparar a interface serial para o projeto. A interface foi criada em linguagem C++ por ser uma linguagem com boas bibliotecas para criação de arquivos. Com a interface serial ficou possível escrever em um arquivo de texto os dados enviados pelo Arduino referente aos seus componentes.

Capítulo 4

Resultados e Análise

Após a montagem do Hardware do circuito e da finalização do código, foi possível efetuar alguns testes mostrando que o código e o Hardware são funcionais.

4.1 Resultados Obtidos

Como uma forma de testar o sensor e de medir suas capacidades, alguns testes foram feitos. Por conta de se tratar de um sensor de luz, os testes foram feitos em um ambiente com luz controlada para que fosse possível ter melhor noção de se a iluminação ambiente poderia afetar as medições feitas.

O ambiente de teste foi um quarto com a iluminação sendo feita apenas por lâmpadas de diferentes intensidades. O sensor ficou apoiado em uma mesa escura com uma proteção plástica e, ao seu lado foi colocada uma régua para que pudesse ser feito uma medição precisa da distância do objeto colocado à frente do sensor e comparar com a distância medida pelo sensor.

Como podemos ver, os erros são presentes em todas as medidas. Sua magnitude se reduz para medidas pequenas. Nas medidas de 2 centímetros, os erros encontrados são grandes e podem propiciar um erro grande para a geração de um pointcloud. As medidas maiores, como as de 20 e 25 centímetros também apresentaram erro, mas tendo em vista que o sensor não está preso a uma plataforma, suas medidas de longas distâncias podem apresentar maiores erros.

Distância Real	2cm	4cm	5cm	8cm	10cm	15cm	17cm	20cm	25cm
Medida1	3,1cm	4,6cm	5,3cm	8,8cm	10,8cm	14,6cm	16,3cm	19,2cm	21,7cm
Medida2	3,1cm	4,7cm	5,7cm	8,7cm	10,5cm	14,8cm	16,0cm	19,2cm	21,5cm
Medida3	3,0cm	4,1cm	5,4cm	8,7cm	10,7cm	15,1cm	16,8cm	18,6cm	21,6cm
Medida4	2,8cm	4,3cm	5,3cm	8,5cm	10,9cm	15,0cm	17,0cm	19,0cm	21,8cm

Figura 4.1: Tabela de Teste para Precisão do Sensor

Os testes com o motor não tiveram resultados mensuráveis. Sabendo que o motor possui 200 passos e que não seria implementada a parte de microsteps através do controlador, a única variável trabalhada foi a velocidade de rotação.

O controlador do motor, por trabalhar com uma corrente de próxima de 2 ampères esquenta muito consideravelmente rápido. Por mais que o controlador tenha um controle de temperatura e proteção contra sobrecorrente, ele desligará se ficar muito quente. Sabendo que o motor executa uma volta a cada 20 segundo, aproximadamente, e supondo que ele tenha que fazer 100 revoluções, o controlador ficaria funcionando por pelo menos meia hora. Caso a temperatura não fosse levada em conta, o sensor desligaria e o escaneamento seria interrompido.

Foi possível obter uma resposta simultânea dos periféricos através da porta Serial. Tendo em vista que o programa para ler a porta já estava funcional, foi possível criar arquivos PCD em tempo real.

```
248 1.80 0.00
247 3.60 0.00
247 5.40 0.00
268 7.20 0.00
251 9.00 0.00
248 10.80 0.00
261 12.60 0.00
260 14.40 0.00
242 16.20 0.00
250 18.00 0.00
263 19.80 0.00
238 21.60 0.00
259 23.40 0.00
239 25.20 0.00
234 27.00 0.00
249 28.80 0.00
247 30.60 0.00
257 32.40 0.00
242 34.20 0.00
251 36.00 0.00
```

Figura 4.2: Dados Coletados da Porta Serial

Na figura 4.2 vemos como o arquivo de texto é gravado. A primeira coluna representa a distância entre o objeto e o sensor e é dada em milímetros. A segunda coluna é o ângulo atual do objeto em referência ao sensor. Por conta do motor de passos o ângulo máximo de variação é 1.8 graus. Por fim temos a coluna que representa a altura em que o objeto está sendo escaneado. A altura irá variar apenas quando o objeto tiver dado uma volta em torno do próprio eixo.

```
241 347.40 0.00
232 349.20 0.00
243 351.00 0.00
239 352.80 0.00
241 354.60 0.00
234 356.40 0.00
250 358.20 0.00
239 360.00 0.00
247 0.00 1.00
236 1.80 1.00
249 3.60 1.00
246 5.40 1.00
256 7.20 1.00
```

Figura 4.3: Variação da Altura ao Completar uma Rotação

Como mostrado na figura 4.3, ao completar uma volta sobre o próprio eixo, a altura será incrementada.

O sistema para de coleta de dados do escâner se baseia em um sistema cilíndrico. Essa escolha foi feita por conta da própria estrutura física do escâner. Como exemplificado na figura 4.4, o sistema é composto por uma base rotatória e, como já citado anteriormente, o sensor de distância deverá se mover na vertical.

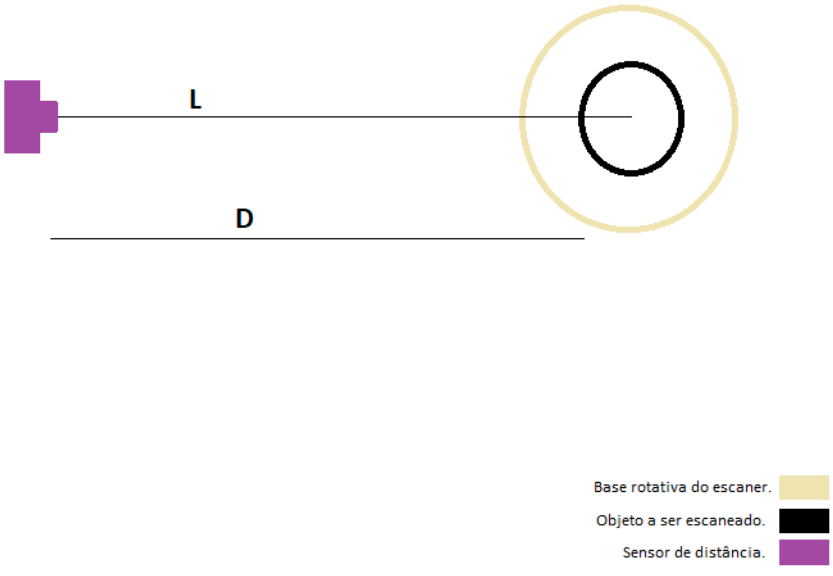

Figura 4.4: Posicionamento do Sensor em relação à Base

Sendo assim, nosso sistema possui duas dimensões de distância, as distâncias entre o sensor e o objeto e a distância vertical que o sensor incrementa para escanear, e uma dimensão angular, gerada pela rotação da base. Um sistema composto por essa combinação de dimensões caracteriza um sistema de coordenadas cilíndricas.

Para o processamento em coordenadas cilíndricas do projeto, devemos lembrar que a primeira coluna representa a distância entre o sensor e o objeto, mas não a dimensão do objeto. Sendo assim chamaremos essa distância entre o sensor e objeto de D, a distância entre o centro do objeto e o sensor de L e o raio do objeto de R. Senso assim, o raio que ocupa um ponto do objeto no momento em que é tomada sua distância terá o valor de:

$$R = L - D \tag{4.1}$$

Agora temos o raio do objeto em coordenadas cilíndricas. O ângulo e altura não precisam sofrer alterações para que os dados fiquem em coordenadas cilíndricas.

Por mais que com esses dados já seria possível processar uma imagem tridimensional através de um software específico para este tipo de sistema de coordenadas, o MatLab não processa dessa forma e nem esse arquivo não atende às características de um arquivo PCD.

Para deixar esse arquivo em um formato PCD, processável pelo MatLab, precisaríamos transformar as coordenadas cilíndricas em coordenadas cartesianas.

Isso se dá aplicando as funções seno e cosseno para a variável R. Sendo o seno e o cosseno do ângulo da rotação no instante da medida. Por exemplo.

Dada a seguinte matriz:

$$\begin{bmatrix} R1 & \theta 1 & Z1 \\ R2 & \theta 2 & Z2 \\ R3 & \theta 3 & Z3 \end{bmatrix}$$

A matriz acima é um exemplo de matriz em coordenadas cilíndricas. Temos nela um exemplo de três pontos. Para que essa matriz fique em um sistema de coordenadas cartesianas, temos de fazer o produto do raio de cada linha com o seno e o cosseno do ângulo da linha respectiva. Sabendo disso, a matriz acima, na forma cartesiana ficaria no formato:

$$\begin{bmatrix} R1cos(\theta 1) & R1sen(\theta 1) & Z1 \\ R2cos(\theta 2) & R2sen(\theta 2) & Z2 \\ R3cos(\theta 3) & R3sen(\theta 3) & Z3 \end{bmatrix}$$

A matriz acima é a representação no espaço cartesiano da matriz anterior a ela. Como podemos ver a álgebra necessária é bem simples, mas faz com que o arquivo com os dados diretamente coletados tenha que passar por um processamento matemático antes de poder

ser processado digitalmente para que a figura tridimensional seja acessível.

O processamento matemático será feito por outro programa. Sendo assim, após a coleta dos dados da porta serial, o arquivo gerado precisa ser processado em outro programa, o que faz com que não seja possível fazer todo o processo em paralelo.

```
function [ Mout ] = jeremy_converte( Min )
% converte de cilindrica para cartesiana

%  cada linha de Min tem o formato [raio angulo altura] ([rho theta z])
% cada linha de Mout tem o formato [x y z]


% cuidado!: a funcao do matlab pol2cart espera [angulo raio altura], por
% isso as trocas
    [A,B,C] = pol2cart(deg2rad(Min(:,2)),Min(:,1),Min(:,3));
    Mout = [A B C];
end
```

Figura 4.5: Transformação de Cilíndrico para Cartesiano

A plataforma atendeu todos os critérios de precisão esperados para que os periféricos conseguissem fazer uma boa leitura. Tanto o sensor de distância, quanto o motor seguem o que é especificado em seu datasheet.

Por mais que ainda existam pequenos erros no medidor de distância. Os erros em si são ínfimos e, por conta disso, não acarretarão problemas para o processamento do arquivo PCD.

Capítulo 5

Conclusão

5.1 Recapitulação dos Objetivos

O projeto foi idealizado com o intuito de criar uma máquina capaz de digitalizar objetos para que pudessem ser reconstruídos em um ambiente virtual. No ambiente virtual as imagens serviriam para estudos e até como base para uma impressora 3D gerar uma réplica.

Um pilar do projeto seria o escâner ser de baixo custo. Para isso os materiais usados deveriam ser os mais baratos possíveis, mas ainda alcançando um nível razoável de qualidade. Com um escâner a baixo custo com padrões mínimos de qualidade, seu alcance seria alto e muitas peças delicadas e de complexo transporte ficariam digitalmente acessíveis a um grande público.

Para que o preço fosse acessível, a criação do aparelho foi independente e as bibliotecas auxiliares de códigos são toda abertas para pesquisa. Assim temos um código próprio para utilizar no equipamento.

Após a interação entre o Arduino e seus periféricos foi necessária a criação da interface de comunicação serial. Ela permitiria que os dados fossem gravados em um arquivo que poderia, mais tarde, ser acessado.

Após a interação entre o Arduino e seus periféricos foi necessária a criação da interface de comunicação serial. Ela permitiria que os dados fossem gravados em um arquivo que poderia, mais tarde, ser acessado.

O tamanho físico de escâner é pequeno, para que seja de fácil transporte e tenha uma estrutura simples.

5.2 Dados Experimentais

Para garantir o funcionamento do foram feitos testes com os equipamentos. Foi possível concluir que os códigos e periféricos funcionam da forma desejada. Isso faz com que o projeto seja funcional logo que implementado.

O passo a passo para o início ao fim do processo começa por o objeto ser colocado em uma base giratória. Após todas as medidas serem tomadas, levando em conta que a distância que temos já é o raio do objeto, é necessário que ocorra o processamento de coordenadas cilíndricas para cartesianas. O processamento das coordenadas ocorre por outro programa, que não o de interface serial. Após o processamento das coordenadas, o arquivo gerado passará por outro programa que irá inserir o cabeçalho para definir o tipo do arquivo (PCD ou PLY).

5.3 Análise de Resultados

Na tabela da imagem 5.1 a seguir podemos ver o exemplo de coordenadas cartesianas e as coordenadas cilíndricas que as originaram.

247,878	7,78987	0
246,513	15,5093	0
245,904	23,2448	0
265,887	33,5893	0
247,91	39,2651	0
243,607	46,4706	0
254,714	56,9354	0
251,832	64,6594	0
232,391	67,5158	0
237,764	77,2542	0
247,452	89,0881	0
221,287	87,6136	0
237,698	102,861	0
216,254	101,761	0
208,496	106,234	0
218,2	119,957	0
212,603	125,733	0
216,992	137,707	0
200,153	136,024	0
203,063	147,534	0
203,07	157,517	0
194,169	160,631	0
172,526	152,102	0
194,635	182,774	0
172,534	172,534	0
175,929	187,345	0
159,376	180,777	0
158,081	191,087	0
145,872	188,057	0
136,366	187,692	0
134,338	197,672	0
134,493	211,926	0
129,297	218,628	0
122,365	222,582	0
111,228	218,297	0
103,464	219,873	0

248	1,8	0
247	3,6	0
247	5,4	0
268	7,2	0
251	9	0
248	10,8	0
261	12,6	0
260	14,4	0
242	16,2	0
250	18	0
263	19,8	0
238	21,6	0
259	23,4	0
239	25,2	0
234	27	0
249	28,8	0
247	30,6	0
257	32,4	0
242	34,2	0
251	36	0
257	37,8	0
252	39,6	0
230	41,4	0
267	43,2	0
244	45	0
257	46,8	0
241	48,6	0
248	50,4	0
238	52,2	0
232	54	0
239	55,8	0
251	57,6	0
254	59,4	0
254	61,2	0
245	63	0
243	64,8	0

Figura 5.1: Resultados processados para Coordenadas Cartesianas

Utilizando os pontos na forma cartesiana fica possível usar uma diversidade de programas capazes de ler os pontos e processar a imagem em um ambiente virtual.

5.4 Objetivos Alcançados

Foi possível criar toda a parte de programação do escâner e a interface de comunicação serial com o computador. O material utilizado para a montagem e criação do escâner foi de baixo custo e levando em conta que o processamento do arquivo pode ser feito pelo MatLab ou Octave.

Para ler o arquivo será necessário o uso de algum leitor de Point Cloud. O MatLab tem uma biblioteca para isso. Caso o MatLab não possa ser utilizado haverá a necessidade de encontrar outro programa capaz de fazê-lo ou criar um.

5.5 Projetos Futuros

Para projetos futuros utilizando esse trabalho como base, seria aconselhado alterar o formato da estrutura para uma que limite menos o tamanho do objeto a ser escaneado.

Um bom formato para a estrutura seria uma base em forma de ‘L’ em que o motor de passos ficará preso e terá acoplado em seu rotor a base giratória.

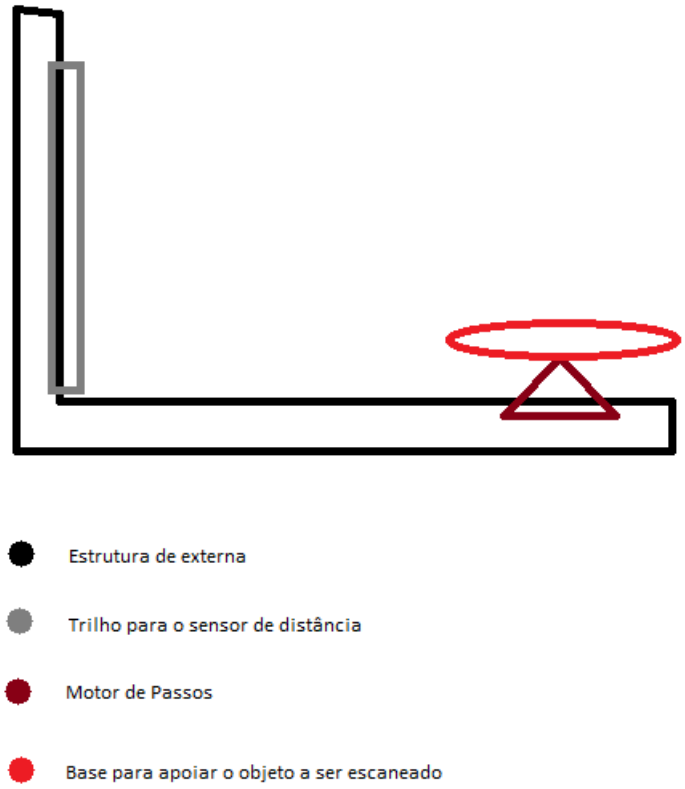

Figura 5.2: Modelo de Base para Projeto Futuro

Uma parte pouco explorada do projeto é o driver controlador do motor. Ele possui diversas funcionalidades que não foram exploradas. Sua capacidade de realizar microsteps pode ajudar na qualidade das imagens. Os microsteps são capazes de reduzir o ângulo mínimo de rotação, deixando a quantidade de pontos úteis para o processamento de uma imagem em um ambiente tridimensional.

Outra possibilidade não explorada é a de usar uma câmera que também consiga captar a cor do ponto em que ela mede a distância. Câmeras assim são mais caras, mas o XBOX Kinect é um exemplo de aparelho que realiza tal operação e é de mais fácil acesso.

Referências Bibliográficas

[1] agrimensordofuturo.com, *Experiências em fotogrametria com vant*, 2014.

[2] Allegro, *Dmos microstepping driver with translator and overcurrent protection*.

[3] Amazon, *Sodial(r) vl53l0x time-of-flight distance sensor gy-vl53l0xv2 module for arduino*, 2018.

[4] Arduino, *Wire library*, 2018.

[5] augmera, *Mobile ar hardware*, 2018.

[6] P. Axelsson, *Processing of laser scanner data algorithms and applications*.

[7] M. Berger and A. Tagliasacchi, *State of the art in surface reconstruction from point clouds*.

[8] W. Boehler and A. Marbs, *3d scanning and photogrammetry for heritage recording: A comparison*.

[9] ———, *3d scanning instruments*.

[10] R. Bridson, *Fast poisson disk sampling in arbitrary dimensions*.

[11] A. Cassinelli, S. Perrin, and M. Ishikawa, *Smart laser-scanner for 3d human-machine interface*.

[12] F. Chiabrando, R. Chiabrando, D. Piatti, and F. Rinaudo, *Sensors for 3d imaging: Metric evaluation and calibration of a ccd/cmos time-of-flight camera*.

[13] Git Community, *Adafruit_vl53l0x*, 2018.

[14] L. Cruz, D. Lucio, and L. Velho, *Kinect and rgbd images: Challenges and applications*.

[15] J. Demantké, C. Mallet, N. David, and B. Vallet, *Dimensionality based scale selection in 3d lidar point clouds*.

[16] J. Deschaud and F. Goulette, *Point cloud non local denoising using local surface descriptor similarity*.

[17] C. Dombroski, M. Balsdon, and A. Froats, *The use of a low cost 3d scanning and printing tool in the manufacture of custom made foot orthoses: a preliminary study.*

[18] electronicaembajadores.com, *Arduino mega 2560*, 2018.

[19] A. Ferneda, *Integração metrologia, cad e cam: Uma contribuição ao estudo de engenharia reversa*.

[20] C. Fischer, M. Suzete, A. Ramos, and F. Teixeira, *Digitalização 3d utilizando kinect e sistemas cad e cam para confecção de órtese de membro inferior*.

[21] C. Glennie, *Rigorous 3d error analysis of kinematic scanning lidar systems*.

[22] Globo, *O que se sabe sobre o incêndio no museu nacional, no rio*, 2018.

[23] P. Grussenmeyer, T. Landes, T. Voegtle, and K. Ringle, *Comparison methods of terrestrial laser scanning, photogrammetry and tacheometry data for recording of cultural heritage buildings*.

[24] M. Hoppen, J. Rossman, and R. Waspe, *A new method for interfacing 3d simulation systems and object-oriented geo data sources*.

[25] F. Hosoi, Y. Nakai, and K. Omasa, *Voxel tree modeling for estimating leaf area density and woody material volume using 3-d lidar data*.

[26] inventables, *Stepper motor - nema 17*, 2018.

[27] jameco.com, *How does the xbox kinect work*, 2018.

[28] T. Kahlmann, F. Remondino, and H. Ingensand, *Calibration for increased accuracy of the range imaging camera swissranger*.

[29] K. Khoshelham, *Accuracy analysis of kinect depth data*.

[30] Labdegaragem, *Motor de relutância variável*, 2018.

[31] Labegaragem, *Driver de controle de motor a4988*, 2018.

[32] _____, *Stepper motor - nema 17*, 2018.

[33] G. Leitão and A. Gomes, *Reconstrução de superfícies trianguladas a partir de nuvens de pontos sem restrições angulares*.

[34] P. Lohmann, A. Koch, and M. Schaeffer, *Approaches to the filtering of laser scanner data*.

[35] Z. Longyu, D. Haiwei, and S. Abdulmotaleb, *From 3d sensing to printing: A survey*.

[36] M. McCool, *Hierarchical poisson disk sampling distributions*.

[37] mercadolivre, *Sensor ultrassom hc-sr04 arduino*, 2018.

[38] Microsoft, *Microsoft kinect sensor for xbox one (black)*, 2018.

[39] G. Pandey, J. McBride, S. Savarese, and R. Eustice, *Automatic targetless extrinsic calibration of a 3d lidar and camera by maximizing mutual information*.

[40] pointclouds.org, *Documentation*, 2018.

[41] F. Remondino, *Heritage recording and 3d modeling with photogrammetry and 3d scanning*.

[42] reporteroindustrial, *Sensor óptico de borde de precisión, poscon 3d*, 2018.

[43] National Ocean Service, *What is lidar?*, 2018.

[44] Naai-Jung Shih, *The application of a 3d scanner in the representation of building construction site*.

[45] J. Smisek, M. Jancosek, and T. Pajdla, *3d scanning instruments (kinect)*.

[46] sparkfun, *Hall-effect sensor*, 2018.

[47] STMicroelectronics, *User manual*, STMicroelectronics, 2016.

[48] A. Voicu and L. Badita, *3d measuring of complex automotive parts using video-laser scanning*.

[49] A. Voicu, I. Gheorge, and L. Badita, *3d measuring of complex automotive parts using video laser scanning*.

[50] N. Wongwaen, S. Tiendee, and C. Sinthanayothin, *Method of 3d mesh reconstruction from point cloud using elementary vector and geometry analysis*.

[51] O. Wulf and B. Wagner, *Fast 3d scanning methods for laser measurement systems*.

[52] R. Zeleznik, H. Kenneth, and J. Hughes, *Sketch: An interface for sketching 3d scenes*.

.1 Apêndice

.1.1 Caracterização do Código

Devido ao fato de o projeto ter uma interface de comunicação entre o Arduino e outra unidade de processamento para salvar os dados, existem diferentes tipos de códigos para diferentes partes do projeto.

.1.2 Código Serial

Para que os dados sejam coletados e, mais tarde, processados, é necessária a implementação de uma interface serial capaz de estabelecer comunicação entre o Arduino e o Computador que estará armazenando os dados.

A interface foi estabelecida através de um programa em C++ capaz de ler informações enviadas pela porta serial e armazená-las em um arquivo de texto (.txt). Para isso foi necessário o uso da biblioteca fstream. Essa biblioteca permite a abertura e criação, tanto quanto o fechamento e formatação, de arquivos. Com o uso dessa biblioteca, o cabeçalho e a coleta de dados através da porta serial poderiam ser salvos em um arquivo que, mais tarde poderia ser lido por algum outro compilador e poderia ter seus dados processados.

O código começa criando um arquivo de texto que levará como parâmetros iniciais o cabeçalho necessário do arquivo para uma biblioteca de point cloud. Para isso foram criadas variáveis com os parâmetros desejados para nossos arquivos e depois da escolha do nome do arquivo, o cabeçalho é escrito no arquivo de texto.

```
//Declarando todo o conteudo do cabeçalho do arquivo
string version = "VERSION .7";
string fields = "FIELDS x y z";
string sizepcl = "SIZE 8 8 8";
string typepcl = "TYPE F F F";
string count = "COUNT 1 1 1";
string widthpcl = "WIDTH 213";
string height = "HEIGHT 1";
string viewpoint = "VIEWPOINT 0 0 0 1 0 0 0";
string points = "POINTS 100000000";
string datapcl = "DATA ascii";
```

Figura 4.1: Declaração das Strings de Cabeçalho

Figura 3: Declaração das Strings de Cabeçalho

Através do uso da biblioteca SerialPort, foi possível utilizar funções que leem os dados enviados à porta serial da máquina e os manipulam da forma como preferirmos. No caso o programa foi feito para que os dados adquiridos da porta serial fossem diretamente salvos em um arquivo de texto.

```
int i = 0;
while (i!=1) //Nesse ponto é criado o arquivo com todo o cabeçalho necessario
{
    if (inFile.is_open())
    {
        inFile << version << endl;
        inFile << fields << endl;
        inFile << sizepcl << endl;
        inFile << typepcl << endl;
        inFile << count << endl;
        inFile << widthpcl << endl;
        inFile << height << endl;
        inFile << viewpoint << endl;
        inFile << points << endl;
        inFile << datapcl << endl;

    }
    i++;
}
```

Figura 4.2: Escrita do cabeçalho no arquivo

Figura 4: Escrita das Strings de Cabeçalho no Arquivo

A biblioteca SerialPort ajuda a definir parâmetros para a leitura da porta serial. Ela pede de entrada apenas o nome da porta serial utilizada. Os outros parâmetros são definidos através de suas funções internas.

.1.3 Código Arduino

Para inserir a funcionalidade de dois periféricos (motor e sensor de distância), ambas as bibliotecas e variáveis precisariam ser inseridas.

Após todas as variáveis serem definidas e as bibliotecas chamadas, é necessário o setup da interface serial e a criação do objeto motor para que o programa o reconheça.

Por fim a função loop é definida para que as funções do motor e do sensor sejam chamadas.

```
#ifndef SERIALPORT_H
#define SERIALPORT_H

#define ARDUINO_WAIT_TIME 1000
#define MAX_DATA_LENGTH 255

#include <windows.h>
#include <stdio.h>
#include <stdlib.h>

class SerialPort
{
private:
    HANDLE handler;
    bool connected;
    COMSTAT status;
    DWORD errors;
public:
    SerialPort(char *portName);
    ~SerialPort();

    int readSerialPort(char *buffer, unsigned int buf_size);
    bool writeSerialPort(char *buffer, unsigned int buf_size);
    bool isConnected();
};

#endif // SERIALPORT_H
```

Figura 4.3: Biblioteca SerialPort (Arquivo .h)

Figura 5: Biblioteca SerialPort (Arquivo .h)

```
#include "Adafruit_VL53L0X.h"

//Parte do motor começa
#include <Arduino.h>
#include "BasicStepperDriver.h"
#include "A4988.h"
#define MOTOR_STEPS 200
#define RPM 120
#define rot 1.8
#define MICROSTEPS 1
#define DIR 8
#define STEP 9
#define ENABLE 13
#define stepper()
BasicStepperDriver mot(MOTOR_STEPS, DIR, STEP);
//Parte do motor acaba

Adafruit_VL53L0X lox = Adafruit_VL53L0X();
```

Figura 6: Bibliotecas e Variáveis para o Arduino

```
void setup()
  {
  Serial.begin(9600);
  mot.begin(RPM, MICROSTEPS);
  while (! Serial)
  {
    delay(1);
  }

  if (!lox.begin())
  {
    Serial.println(F("Failed to boot VL53L0X"));
    while(1);
  }
}
```

Figura 7: Setup do Arduino

```
void loop()
{
  VL53L0X_RangingMeasurementData_t measure;

  //Serial.print("Reading a measurement... ");
  lox.rangingTest(&measure, false); // pass in 'true' to get debug data printout!



  if (measure.RangeStatus != 4)
  {  // phase failures have incorrect data
    //Serial.print("Distance (mm): ");
    if (graus >= 360)
    {
    altura++;
    graus = 0;
    }
    Serial.print(measure.RangeMilliMeter);
    Serial.print(" ");
    Serial.print(graus);
    Serial.print(" ");
    Serial.println(altura);
  }

  else
  {
    Serial.println(0);
  }

  mot.rotate(rot);
  graus = graus + rot;


  delay(100);
}
```

Figura 8: Laço para Tomada de Medidas e Rotação do Motor