Leonardo dos Santos Duarte Silva, Pedro Arejano Scheunemann,
Gabriel Feijó Teixeira

# Escâner tridimensional de pequenos objetos que utiliza sensor de distância para a criação de uma nuvem de pontos

Rio Grande(RS)
2021

# 1

# Escâner tridimensional de pequenos objetos que utiliza sensor de distância para a criação de uma nuvem de pontos

**Leonardo dos Santos Duarte Silva, Pedro Arejano Scheunemann, Gabriel Feijó Teixeira**
**Orientador:** Prof. Carlos Rodrigues Rocha

## Resumo

Este documento exibe a construção de um escâner tridimensional responsável pela criação da nuvem de pontos de pequenos objetos. Tal feito se dá a partir da comunicação de dois softwares, um que realiza o cálculo e a criação dos pontos captados do objeto através de um sensor de distância, o qual é parte de uma estrutura móvel guiada por motores de passo síncronos que objetivam direcionar o sensor a todos ângulos do objeto, e o outro que reúne os pontos e permite visualizá-los juntos, formando o objeto. Como elemento principal é utilizada a placa Arduino Uno R3, a qual desempenha a função de integrar todos constituintes do sistema, além de controlar os motores de passo em conjunto com o processamento dos dados do sensor que são enviados para o software responsável pela criação da nuvem de pontos. Estes softwares foram feitos através de códigos elaborados nas linguagens de programação C++ e Python, e executados, respectivamente, no Ambiente de Desenvolvimento Integrado do Arduino e no Visual Studio Code.

**Palavras Chave:** Escâner tridimensional; Arduino; Sensor VL53L0X; Nuvem de pontos;

# Abstract

This document shows the construction of a three-dimensional scanner responsible for creating the point cloud of small objects. This feat takes place through the communication of two software, one that performs the calculation and creation of the points captured from the object through a distance sensor, which is part of a mobile structure guided by synchronous stepping motors that aim to direct the sensor to all angles of the object, and the other that gathers the points and allows viewing them together, forming the object. The main element is the Arduino Uno R3 board, which performs the function of integrating all components of the system, in addition to controlling the stepper motors together with the processing of sensor data that are sent to the software responsible for creating the point cloud. These softwares were made using codes elaborated in the programming languages C++ and Python, and executed, respectively, in the Arduino Integrated Development Environment and in Visual Studio Code.



**Keywords:** 3D Scanner; Arduino; VL53L0X Sensor; Point Cloud.

# 1 Introdução

Nos dias de hoje, com a crescente popularização de impressoras 3D, invento que permite a confecção de itens tridimensionais a partir da reprodução física de um modelo criado virtualmente, a demanda por tecnologias da engenharia reversa, ou seja, que possibilitem a reprodução digital de um modelo físico já existente, aumentou de forma vigorosa. Assim se deu o surgimento de escâneres 3D, os quais utilizam tecnologia de ponta para suprir esta exigência, e, por conta disso, possuem um custo de aquisição elevado. Porém, dependendo da aplicação do usuário, à vista da complexidade do objeto a ser escaneado, tendo ele dimensões que podem ser conferidas facilmente, escâneres de baixa resolução e principalmente mais baratos costumam ser suficientes. Nesse contexto, o presente trabalho tem como objetivo a construção de um escâner tridimensional de baixo custo idealizado a partir do interesse de que o produto sirva competentemente para criação da nuvem de pontos de pequenos objetos, para futura modelagem computacional, tanto para entusiastas como para interessados na área.

## 1.1 Justificativa

Esse projeto foi escolhido com o intuito de construir um escâner tridimensional com materiais de baixo custo e fácil acesso, visto que a demanda na área de tecnologia e desenvolvimento é grande e os produtos encontrados no mercado são muito caros. Nesse sentido, o trabalho visa facilitar o acesso desse dispositivo à usuários que pretendam se desprender de serviços de terceiros ou compra de equipamentos caros para conseguir usufruir dessa tecnologia, seja qual for sua aplicação.

## 1.2 Objetivos

O projeto tem por objetivo principal proporcionar um meio não custoso de construir um equipamento que proporcione um escaneamento para pequenos objetos de pouca complexidade para entusiastas ou interessados na área de escaneamento tridimensional. Para tanto, foram estabelecidas como metas:

- Projetar a parte estrutural do escâner, tanto os elementos móveis quanto os fixos, a fim de trazer estabilidade, velocidade e qualidade para todo o processo de escaneamento.
- Projetar o sistema eletrônico do escâner, interligando o sensor de distância e os motores de passo com a placa Arduino, com objetivo de se ter um controle preciso e harmônico desses elementos.
- Desenvolver o software tanto de controle dos motores de passo, comunicação e processamento dos dados do sensor no Arduino, quanto o software de aquisição destes dados para criação da nuvem de pontos em Python.

- Construir um protótipo do dispositivo, com base nos projetos realizados;
- Legitimar o projeto através de testes e outros pormenores;

# 2 Referencial Teórico

Neste tópico serão especificados os conceitos que servirão como base para o desenvolvimento do projeto, incluindo fundamentos e teorias que irão abranger os conteúdos mais importantes para construção do escâner tridimensional idealizado pelo grupo.

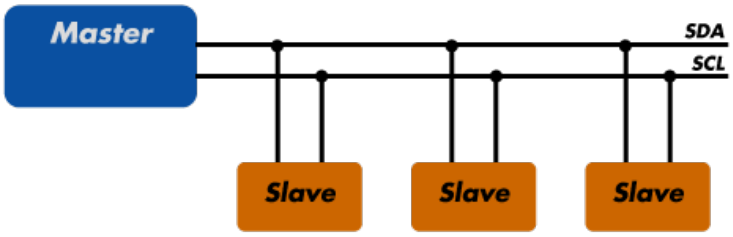
## 2.1 Sensor VL53L0X

Através da tecnologia de Tempo-De-Voo (*time-of-flight*) (LAKOVIć et al., 2019), a qual consiste em medir uma distância a partir da velocidade da luz e do tempo que ela demora entre sair de um emissor e chegar em um receptor, surge o sensor VL53L0X, o qual conta com um emissor e um receptor contidos em um pequeno invólucro de apenas 4.4mm x 2.4mm x 1.0mm (STMICROELECTRONICS, 2018) e permite medições precisas (resolução de 1mm) de obstáculos que estejam até 2 metros distantes. Sua configuração baseia-se basicamente em definir um *trade-off* entre velocidade de medição e precisão da mesma. Para isso, existem 4 diferentes modos de operação, são eles: o modo normal (*default mode*), o modo alta precisão (*high precision*), o modo longa distância (*long-range*) e o modo alta velocidade (*high-speed*). O primeiro, consegue medir objetos até 120cm com boa precisão (4%-6% de erro em lugares fechados e 6%-12% em lugares abertos) a uma velocidade média de 30ms por medição. O segundo, tem altíssima precisão, até 3% de erro à um custo de adicionais 170ms, assim, resultando em 200ms por medição para objetos até 1.2 metros distantes. O terceiro modo, por sua vez, permite medições de até 2 metros a uma velocidade de 33ms por medição com uma precisão igual ao modo normal. Por fim, o modo alta velocidade não tem a precisão como sua prioridade, sendo assim, consegue fazer medições a cada 20ms à objetos até 1.2m de distância com erro médio de 5% (STMICROELECTRONICS, 2018).

## 2.2 Protocolo I²C

O protocolo I²C (*Inter-Integrated Circuit*) (PATEL; GANDHI, 2019) é um protocolo de transferência de dados que ganhou muita relevância por, além de sua ampla aplicabilidade, ser de fácil uso e implementação. Ele consiste de apenas dois barramentos, o de dados (SDA, ou *Serial Data Line*) e o de clock (SCL, ou *Serial Clock Line*), e se baseia na hierarquia mestre-escravo, como visto na figura 1, com permissão de múltiplos mestres/escravos, onde apenas o primeiro consegue iniciar uma transferência no barramento e o segundo precisa responder (I²C..., 2016). Isso por si só já é uma vantagem em relação aos outros métodos, como UART (*Universal Asynchronous Receiver/Transmitter*) e SPI (*Serial Peripheral Interface*).

Figura 1 – Diagrama Comunicação I²C




## 2.3 Nuvem de Pontos

Nuvem de pontos é um conjunto de pontos tridimensionais que representam informações de objetos do mundo real de forma virtual (MEKURIA; LI; TULVAN, 2016). A representação desses dados tridimensionais pode ser utilizada em vários cenários, desde realidade virtual e mapeamento por meio de dispositivos móveis à escâneres de artefatos históricos e impressoras 3D. Essa técnica é amplamente utilizada para captação de dados de escâneres 3D, visto que, para que eles sejam armazenados, é necessário apenas preencher uma lista de valores com tuplas, sendo cada uma delas o conjunto (x, y, z). No entanto, esse não é o produto final para a produção de modelos 3D. Nesse sentido, entram técnicas de *Mesh* para juntar todos pontos e formar uma figura única.

## 2.4 Motores de Passo

Os motores de passos são dispositivos eletromecânicos que rotacionam seu eixo a partir de pequenos incrementos angulares originados por pulsos elétricos aplicados em uma sequência específica nas bobinas do motor. O deslocamento do eixo ocorre pelo fato de que a energização das bobinas cria um campo magnético intenso que atua no sentido de se alinhar com os dentes do rotor, o qual é magneticamente ativo (BRITES; SANTOS, 2008). Assim, ao polarizar de forma adequada as bobinas, pode-se configurar o modo de operação do motor, seja por passo completo com alto torque (*Full-Step*), passo completo com baixo torque (*Wave-Step*) ou meio passo (*Half-Step*). Abaixo estão tabelas 1, 2 e 3 para melhor visualização da sequência de pulsos específicas referente a cada modo de operação mencionado. O número 0 (nível lógico baixo) indica fase não acionada, e o número 1 (nível lógico alto) indica fase acionada.

Tabela 1 – Full-step

| passo | fase 1 | fase 2 | fase 3 | fase 4 |
|---|---|---|---|---|
| 1 | 1 | 1 | 0 | 0 |
| 2 | 0 | 1 | 1 | 0 |
| 3 | 0 | 0 | 1 | 1 |
| 4 | 1 | 0 | 0 | 1 |

Tabela 2 – Wave-step

| passo | fase 1 | fase 2 | fase 3 | fase 4 |
|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 |
| 2 | 0 | 1 | 0 | 0 |
| 3 | 0 | 0 | 1 | 0 |
| 4 | 0 | 0 | 0 | 1 |

Tabela 3 – Half-step

| passo | fase 1 | fase 2 | fase 3 | fase 4 |
|---|---|---|---|---|
| 1 | 1 | 0 | 0 | 0 |
| 2 | 1 | 1 | 0 | 0 |
| 3 | 0 | 1 | 0 | 0 |
| 4 | 0 | 1 | 1 | 0 |
| 5 | 0 | 0 | 1 | 0 |
| 6 | 0 | 0 | 1 | 1 |
| 7 | 0 | 0 | 0 | 1 |
| 8 | 1 | 0 | 0 | 1 |

Vale comentar também que os motores de passo podem ser unipolares, quando estes possuem dois enrolamentos por fase, sendo um para cada sentido de corrente, ou bipolares, quando têm apenas um enrolamento por fase. Esse tipo de motor é normalmente empregado em aplicações que demandam alta precisão, quando é necessário controlar fatores como ângulo de rotação, velocidade e sincronismo. No projeto foram utilizados dois motores de passo unipolares, modelo 28BYJ-48, que podem ser alimentados com 5V e consomem baixa corrente. Estes motores possuem um ângulo de passo de 11,25° (32 passos por revolução), para uma sequência de 4 passos, ou 5.625° (64 passos por revolução), para uma sequência de 8 passos. Porém, como possui mecanismos de redução e a razão da variação de velocidade é de 1/64, então para uma sequência de 4 passos são necessários 2048 passos para dar uma volta, e para uma sequência de 8 passos são necessários 4096. Isso implica que para os modos de operação *full-step* e *wave-step* é preciso 2048 passos para completar uma volta e para o modo *half-step* é preciso 4096 voltas.

### 2.4.1 Drivers para motores de passo

Um driver para motor de passo é um dispositivo responsável por receber os pulsos de sinal gerados pelo controlador e realizar o chaveamento dos componentes de potência para fornecer a corrente necessária em cada fase a fim de dar movimento ao motor. Em suma, o driver é o dispositivo intermediário entre o controlador e o motor de passo. É através do controlador, conforme a determinação do usuário, que os pulsos de sinal são enviados na sequência desejada para energizar as bobinas do motor de acordo com o modo de operação pretendido, e é assim que se tem um controle inteligente de um motor de passo. No projeto foram utilizados drivers modelo ULN2003. O interessante é que estes drivers possuem 4 LEDs que indicam o acionamento de cada fase do motor.

## 2.5 Arduino Uno R3

Arduino Uno R3 é uma placa de prototipagem eletrônica que contém um microcontrolador Atmega328P, que faz o papel de processador principal, além de alguns periféricos como: 14 pino digitais *I/O* (dos quais 6 pode ser usados com saída PWM), 6 entradas analógicas, um cristal oscilador de 16 MHz, conector de energia, conector USB fêmea do tipo B (que também possui a função de alimentar o Arduino quando não há uma fonte externa conectada), barramento ICSP e um botão de reset. Vale comentar também que além do processador principal, a placa é composta por um processador USB, função validada a um microcontrolador ATmega16U2, que é responsável por fazer a comunicação do Arduino com o PC através da porta USB. Ele é necessário, pois o processador principal do Arduino (ATmega328P) não suporta conexão direta com uma porta USB. Dessa forma, o processador USB converte os dados da USB do PC para um sinal serial (UART), e este sim pode ser lido pelo processador principal. O ATmega328P é o "cérebro"do Arduino UNO e, resumidamente, podemos dizer que ele tem três funções: receber, enviar e interpretar os sinais da serial que vêm do processador USB ATmega16U2, executar o software que está programado nele e interagir diretamente com os shields e elementos externos, realizando acionamento de dispositivos e leitura de sensores (FURLAN, 2018). A placa pode ser facilmente conectada à um computador via conexão USB e programada via IDE (*Integrated Development Environment*, ou Ambiente de Desenvolvimento Integrado) utilizando uma linguagem baseada em C/C++.

## 2.6 Manipulação das portas digitais do Arduino

Para determinar os estados dos pinos de entrada e saída do Arduino o microcontrolador Atmega328p possui registradores onde esses dados são armazenados. Ao chamar as funções de entrada e saída fornecidas pela biblioteca padrão do Arduino, como a *pinMode()* e *digitalWrite()*, o que é feito é nada mais que modificar tais registradores. Entretanto, a manipulação direta destes registradores permite que a leitura e escrita nesses pinos sejam muito mais rápidas com códigos eficientemente mais leves (PICORETI, 2017).

Os pinos de entrada e saída do Arduino são divididos em PORTs, um conjunto de pinos que podem ser configurados ou acessados de uma só vez, conforme a tabela 4.

Tabela 4 – PORTs

| | |
|---|---|
| PORTB | representa os pinos digitais de 8 a 13 |
| PORTC | representa os pinos analógicos de 0 a 5 |
| PORTD | representa os pinos digitais de 0 a 7 |

Cada PORT é controlado por 3 registradores, que são também variáveis globais pré-definidas na linguagem do Arduino. Cada registrador possui uma função diferente, em consonância com a tabela 5 a seguir.

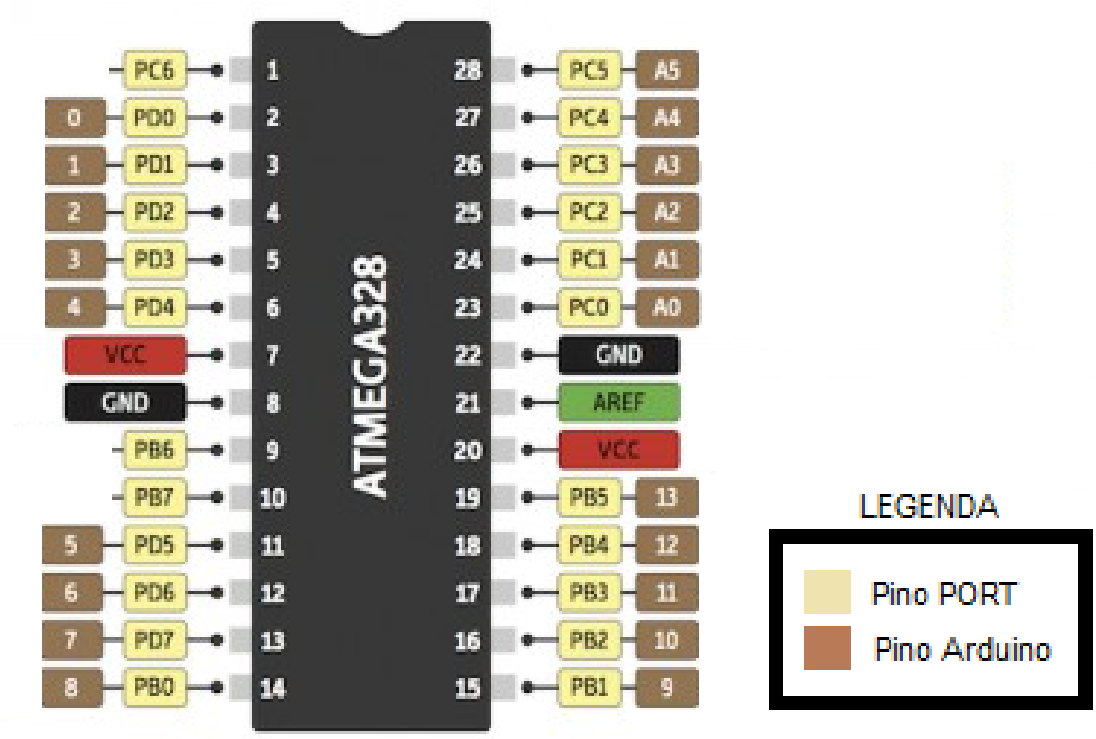
Tabela 5 – Funções de cada registrador

| | |
|---|---|
| PORTx | registrador de dados, são responsáveis por definir um pino como nível lógico alto (bit = 1) ou nível lógico baixo (bit = 0) |
| DDRx | registrador de direção, são responsáveis por determinar se os pinos de um determinado PORT se comportarão como entrada (bit = 0) ou saída (bit = 1) |
| PINx | registrador do endereço de entrada do pino, são responsáveis por guardar o estado lógico de um pino |

sendo x igual a B, C ou D, referente ao PORT a ser manipulado

De modo geral, todos registradores do Atmega328P possuem tamanho de 1 byte. Portanto, para manipular quaisquer registradores e, consequentemente, controlar os estados dos pinos *I/O*, através da *IDE* do Arduino, basta atribuir um valor em forma de 1 byte antecedido pela letra B à variável global referente ao registrador. Vale mencionar que cada bit controla o estado do seu respectivo pino P$xn$ (sendo $x$ referente ao PORT, e $n$ referente a posição do bit dentro do byte) conforme o mapeamento do chip Atmega328p, mostrado na figura 2. Por exemplo, o bit menos significativo - situado na posição 0 - dos registradores PORTB, DDRB e PINB controla o pino PB0, referente ao pino digital 8 do Arduino. Se a intenção é defini-lo como saída e ativá-lo, basta atribuir o byte B00000001 à váriavel DDRB e B00000001 à váriavel PORTB, como na listagem 1.

Figura 2 – Mapeamento do chip Atmega328P



Listing 1 – Demonstração da manipulação do pino digital 8 do Arduino

```
void setup() {
    // Forma convencional
    pinMode(8, OUTPUT);
    // Forma utilizada
    DDRB = B00000001;
}
void loop() {
    // Forma convencional
    digitalWrite(8, HIGH);
    // Forma utilizada
    PORTB = B00000001;
}
```

# 3 Projeto do Protótipo

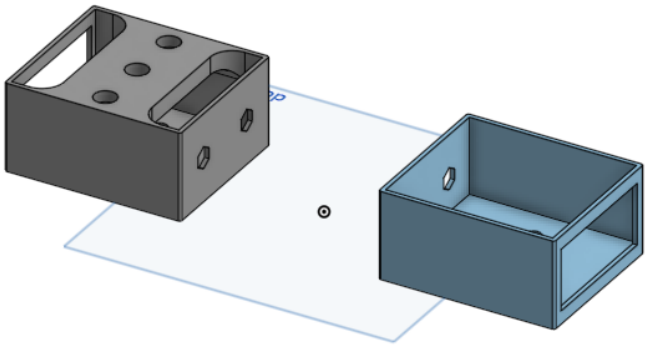
Para desenvolver o projeto era preciso ter o enfoque na movimentação do sensor que fornece as medições para a construção da nuvem de pontos e do objeto a ser escaneado. Logo, foi pensado em construir uma plataforma que tivesse duas partes, uma para a base giratória, onde é colocado o objeto a ser escaneado, e outra para deslocar o sensor no sentido vertical. Para o movimento do sensor foi pensado em utilizar uma vara rosqueada presa ao eixo de um motor de passo que faria uma peça contendo o sensor se deslocar ao longo da vara com o auxílio de porcas e hastes lisas, sendo as últimas para fornecer estabilidade ao escaneamento. A base giratória também receberia um motor de passo para realizar o seu movimento, sendo ambos os motores controlados simultaneamente pela placa Arduino, através da sua *IDE*. A estrutura foi pensada para oferecer estabilidade, velocidade e precisão para o escâner e recebe algumas peças desenhadas em um Software de design 3D feitas para impressão 3D, o OnShape (INC., 2021). Este programa é primordial para a realização do projeto, pois além de vários usuários poderem modificar um projeto ao mesmo tempo, é possível ter muita precisão nas medidas das peças desenhadas. Essa qualidade que o software oferece facilita o encaixe das partes e componentes, pois tudo será feito sob medida. Dito isto, para melhor entendimento do que foi exposto, ao longo deste tópico será descrito as etapas iniciais do projeto.

## 3.1 Peças desenvolvidas no CAD

Como informado anteriormente, a estrutura tem duas partes, então começou-se desenhando duas caixas pequenas que são as bases da estrutura e que receberão os motores. Elas possuem uma face aberta para entrada destes. A primeira caixa contém três furos verticais, os dois da ponta são onde as hastes responsáveis por estabilizar o sensor são fixadas e o do meio é onde a vara rosqueada vai se conectar ao eixo do motor. Os dois furos horizontais são para fazer a junção entre as duas caixas. Já a segunda caixa tem apenas os dois furos

horizontais com a mesma funcionalidade já citada. Esta caixa recebe o motor que gira o disco, onde é colocado o objeto. Ambas as caixas possuem as dimensões de 7 cm x 8 cm x 5 cm. Na Figura 3 é possível observar as partes desenvolvidas.

Figura 3 – Desenho das Caixas

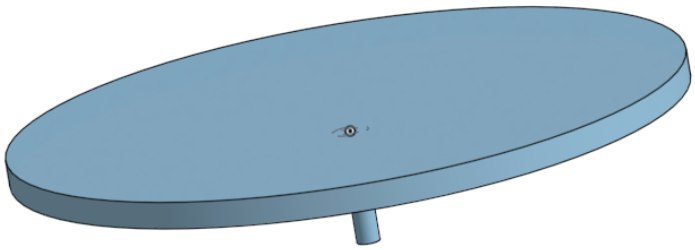


Também foi desenhada outra peça que será responsável pelo deslocamento do sensor no eixo vertical. Esta peça contém cinco furos, sendo três verticais e dois horizontais. Dentre os verticais, o central receberá duas porcas que serão acopladas a vara rosqueada, e os outros dois receberão as hastes que garantem a estabilidade do processo de escaneamento e fixação da própria peça. Já os furos horizontais foram feitos para receber dois parafusos que encaixarão no sensor, como mostra a Figura 4. Essa parte mede 7cm x 3 cm x 5 cm.

Figura 4 – Peça do sensor.

Por último foi desenvolvido o disco que recebe o objeto a ser escaneado, o qual tem 15 cm de diâmetro e 5mm de altura. Também possui um pino em sua base de 1,5 cm de altura que serve para encaixar na mangueira que ligará o disco ao eixo do motor. A figura 5 mostra o desenho:

Figura 5 – Disco Giratório.



## 3.2 Peças da estrutura móvel

Como já foi exposto anteriormente, para movimentar a peça do sensor foi pensado em utilizar uma vara rosqueada conectada com o eixo do motor de passo para quando o eixo do motor girar, a peça movimente-se pela vara através de duas porcas presas no interior dela. Não menos importante, também foram utilizadas duas hastes lisas, que ficam presas tanto na caixa do motor quanto na peça, as quais impedirão o movimento desta peça junto ao giro da vara rosqueada.

As hastes mencionadas tem 17 cm de altura, 0.7 cm de diâmetro e são de metal. Já a vara rosqueada tem 20 cm de altura e 0.4 cm de diâmetro. Para fazer a conexão entre o eixo do motor e a vara foi pensado em utilizar uma mangueira. Também decidiu-se fixar algumas das partes do circuito em uma tábua de madeira com cola e parafusos, a fim de uma estrutura do escâner melhor arranjada e maior organização das conexões elétricas.

## 3.3 Fonte de alimentação

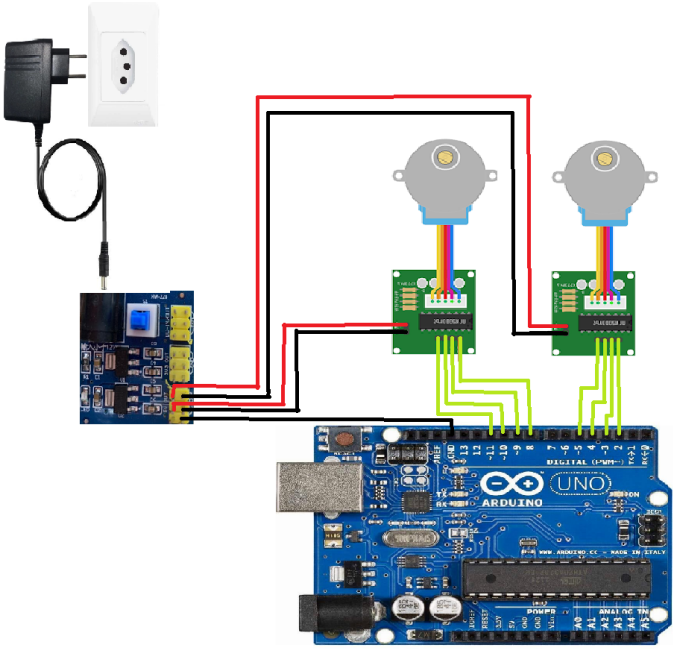
Para a energização do motores será utilizada uma fonte bivolt com saída de 9 volts e plug P4 conectada a um módulo regulador de tensão *step-down* AMS1117. A principal função deste módulo step-down é, de modo geral, converter um nível de tensão inicial em outro nível secundário, no caso 9v para 5v. É relevante mencionar que embora fosse possível utilizar o Arduino para alimentar o circuito, já que este fornece 5 volts justamente para alimentação de circuitos externos, não era desejado sobrecarregar a placa, dessa forma evitando quaisquer possíveis problemas internos.

## 3.4 Circuito dos motores de passo

Os motores de passo, modelo 28BYJ-48, foram conectados aos drivers ULN2003, que possuem um conector JST de 5 vias que se encaixa perfeitamente nos 5 fios do motor. O driver possui 6 pinos, sendo que 2 deles devem ser ligados ao 5v e GND, provenientes do regulador de tensão AMS1117. O Arduino também deve ter seu GND conectado ao GND deste regulador de tensão. Os outros pinos do driver, os quais são as entradas dele (IN1, IN2,

IN3, IN4), foram conectados às portas digitais 2, 3, 4, 5 (referente a PORTD) e 8, 9, 10, 11 (referente a PORTB) do Arduino, dito que são 2 drivers para 2 motores. Por fim, o Arduino é conectado ao computador com cabo USB. Na figura 6 está representado o esquemático e ligações do circuito dos motores de passo com a fonte de alimentação.

Figura 6 – Esquemático dos motores de passo com alimentação




## 3.5 Código

Seguindo os conceitos de programação orientada à objetos (STROUSTRUP, 1988) e conceitos de código limpo (MARTIN, 2008), no nosso código completo foram escritas funções, classes e comentários. Além disso, todo código do sistema foi desenvolvido pensando na sua manutenibilidade e fácil leitura para pessoas não experientes na programação. O sistema se baseia em dois arquivos principais, um que é enviado ao Arduino e outro que roda na máquina do usuário. O primeiro, visa movimentar os motores de passo de maneira sincronizada para que seja possível medir, através do sensor de profundidade, calcular e enviar os os milhares de pontos X, Y e Z do objeto inteiro à porta Serial. O segundo, por sua vez, consiste na leitura da mesma porta, manipulação dos dados obtidos e geração da nuvem de pontos a partir deles.

## 3.6 Controle dos motores de passo na IDE Arduino

O software desenvolvido pelo grupo define duas listas com 4 conjuntos de byte cada, que representam, respectivamente, a sequência de pulsos nas portas digitais do Arduino conectadas às entradas dos drivers ULN2003 e, consequentemente, dos motores. Estes conjuntos de bytes são criados para manipular os registradores de forma mais prática, bastando

atribuí-los às váriaveis globais PORTx/DDRx/PINx em uma lógica sequencial. Já que os motores são pequenos e a estrutura relativamente pesada, foi escolhido configurá-los no modo *full-step*. Como já foi dito, nesse modo, o motor 28BYJ-48 necessita de 2048 passos para dar uma volta completa. Portanto, para fazer o motor conectado ao driver na PORTD do Arduino realizar uma volta no sentido horário, há a necessidade de realizar 4 passos ciclicamente 512 vezes, sendo eles: B00001100, B00011000, B00110000, B00100100, dando pulsos na sequência: PD2(pino 2) e PD3(pino 3), PD3(pino 3) e PD4(pino 4), PD4(pino 4) e PD5(pino 5), PD5(pino 5) e PD2(pino 2). Isso é mostrado na listagem 2. Já para fazer o motor conectado ao driver na PORTB do Arduino realizar uma volta no sentido horário há a necessidade de realizar estes 4 passos 512 vezes ciclicamente: B00000011, B00000110, B00001100, B00001001, dando pulsos na sequência: PB0(pino 8) e PB1(pino 9), PB1(pino 9) e PB2(pino 10), PB2(pino 10) e PB3(pino 11), PB3(pino 11) e PB0(pino 8).

Listagem 2 – Controle motor de passo pela PORTD

```
void FullStepSensor(bool clockwise = true) {

    // Matriz dos bytes das fases do motor
    byte matrix[4] = {B00001100, B00011000, B00110000, B00100100};

    for(int j = 0; j < 4; j++) {
    // Atraso de tempo entre as fases em milisegundos
    delay (phase_delay);
    sensorSteps += 1;
        if(clockwise) {
          PORTD = matrix[j];
        }
        else {
          PORTD = matrix[3 - j];
        }
    }
    // Reset das portas
    PORTD = B00000000;
}
```

## 3.7 Configurando o Sensor VL53L0X

Para que a comunicação $I^2C$ ocorra, são utilizadas duas bibliotecas: Wire e VL53L0X. A primeira, permite que seja feita a comunicação do sensor ao Arduino através do protocolo I²C, por meio do comando *Wire.begin()*, o qual faz com que o Arduino se junte ao barramento I²C como mestre, já que nenhum parâmetro foi passado na função. A segunda biblioteca, por sua vez, é responsável por configurar e manipular o sensor. Nesse sentido, visto que o barramento do Arduino está preparado para receber esse tipo de comunicação, a biblioteca consegue endereçar o sensor dentro do barramento e distinguir sua informação de outros dispositivos que também pudessem estar ali conectados. A biblioteca VL53L0X permite mudar

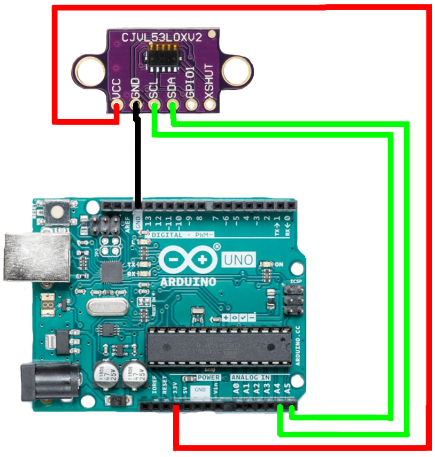
diversas configurações do sensor, desde o tempo que se deve esperar entre uma medição e outra até o tempo que irá demorar cada uma delas, com o fim de poder definir o *trade off* velocidade x precisão. Como o trabalho consiste em criar um arquivo para impressão 3D, o sensor foi configurado com esse fito. Assim, através da função *setMeasurementTiming-Budget(50000)*, definiu-se o tempo que levaria uma medição. O padrão do sensor é 33ms, porém, para maior acurácia, sem que se perdesse velocidade, foi escolhido 500ms. Para fins de segurança, foi definido, através da função *setTimeout(500)*, 500ms como o período no qual operações de leitura serão abortadas se o sensor não estiver pronto. Além disso, para a configuração do sensor, escolheu-se como modo de leitura o modo contínuo, através da função *startContinuos()*, a qual faz com que o sensor realize o máximo de medições possíveis. A configuração do sensor pode ser vista na listagem 3.

Listagem 3 – Configuração Sensor

```
#include <VL53L0X.h>

VL53L0X sensor;

void setup() {
    // (...)
    // Configurando Sensor VL53L0X
    sensor.init();
    sensor.setTimeout(500);
    sensor.setMeasurementTimingBudget(50000);
    sensor.startContinuous();
}
```

A conexão com o Arduino é feita de maneira bem simples, como pode ser visto na figura 7, há apenas a necessidade de conexão dos pinos comum (GND) e de entrada de tensão (VIN) do Sensor ao microcontrolador. Além disso, há a necessidade de conectar os pinos SCL e SDA, para comunicação $I^2C$, às portas destinadas a isso no Arduino. No caso do modelo UNO, os pinos para esse fim são, respectivamente, o A4 e o A5.

Figura 7 – Circuito Sensor VL53L0X.



É importante ressaltar que o sensor conta com 3 modos de operação manipuláveis pela API, sendo eles: *Single Ranging*, a qual cada medição é feita separadamente apenas quando uma função da API é chamada, *Continuous Ranging*, a qual as medições acontecem a todo tempo, assim que acaba uma outra já começa, e, por fim, o modo *Timed Ranging*, onde as medições ocorrem em ciclos predefinidos.

## 3.8 Geração da nuvem de pontos

Para a obtenção e manipulação dos dados obtidos pelo sensor VL53L0X, foram criadas classes e funções. Primeiramente foi criada uma função, *measure3DPoint*, que retorna um objeto do tipo Ponto, classe criada pelo grupo com o fim de facilitar a manipulação dos pontos, a partir de cálculos trigonométricos. Cada um deles (x, y, z) é obtido de maneira diferente: o primeiro, é obtido através da expressão:

$$cos(inRadians) * (75 - lastDistance)$$

o segundo, através da expressão:

$$-1 * sin(inRadians) * (75 - lastDistance)$$

Nos dois casos, *inRadians* representa quanto, em radianos, a base rotacionou e 75 a distância do centro da base rotatória até o sensor. *lastDistance*, por sua vez, é o valor da última medição feita pelo sensor, em milímetros. Por fim, o terceiro ponto, o z, é obtido através da seguinte expressão:

$$sensorSteps/512$$

Onde *sensorSteps* é o número de passos que o motor que move o sensor deu e 512 representa quantos passos são necessários para que o sensor se mova 1mm. Depois de mensurados e calculados, como visto na listagem 4, os 3 pontos da coordenada, são transmitidos, via comunicação Serial, para o software Python, o qual, através da biblioteca PyVista, junta todos esses pontos e cria um arquivo de extensão *.vtk*, sendo esse a nuvem de pontos final.

# 4 Construção de Um Protótipo

Neste tópico será mostrado o processo de montagem do protótipo com as peças que foram desenhadas já impressas e os componentes eletrônicos juntos, expondo a junção de todos elementos do projeto, seja a parte estrutural quanto a eletrônica. Além disso, serão apresentadas algumas dificuldades que apareceram ao longo da construção, fora os testes realizados com os motores para verificar os movimento das peças com eles e as conexões elétricas.

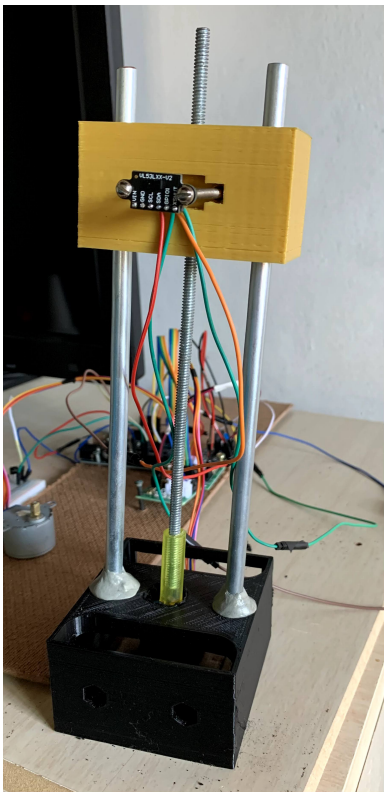
Listagem 4 – Geração da nuvem de pontos

```
VL53L0X sensor;

// Retorna um Point contendo os 3 pontos (x,y,z) encapsulados
Point measure3DPoint() {

  lastDistance = sensor.readRangeContinuousMillimeters();

  float inRadians = baseSteps * 0.7 * 3.14159265 / 180;
  float x = cos(inRadians) * (75 - lastDistance);
  float y = -1 * sin(inRadians) * (75 - lastDistance);

  float z = sensorSteps / 512;

  Point point(x, y, z);

  return point;
}
```

## 4.1 Plataforma

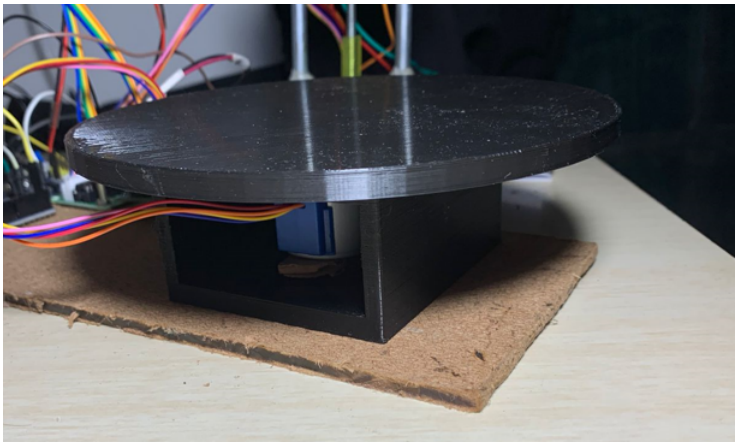
Depois de desenvolver todas peças nos softwares e esperar a sua fabricação, foi possível dar início a construção do escâner. A primeira parte montada foi a que recebe o sensor pois era a mais suscetível a falhas. Ao começar os encaixes, foi encontrado apenas um problema para a construção dela. O empecilho era com as hastes que ficam presas na caixa, elas estavam um pouco soltas e não garantiam estabilidade alguma para a peça do sensor se deslocar ao longa da vara rosqueada. Para solucionar o problema, foi colocado resina epóxi nas bases em contato com a caixa para fixá-las. Também é valido mencionar que para estabelecer o sensor em sua peça foram utilizados dois parafusos que ficam em suas laterais e atravessam a a sua peça, garantindo certa estabilidade. Na Figura 8 é possível observar a construção.

Figura 8 – Parte Montada.



Depois de terminar a construção da primeira parte, avançou-se para confecção da base giratória. A montagem da base foi rápida, primeiro fixou-se o motor na caixa, depois conectou-se o disco ao eixo do motor com uma mangueira e por fim posicionou-se esta parte de frente à outra. A construção desta porção do projeto não teve complicações para o encaixe, mas as alturas entre as duas caixas não estava satisfatória para o escaneamento, então na hora da medição foram utilizadas suportes na parte inferior da base que gira. Outro ponto importante que o desnível das peça afeta é a conexão entre as caixas que foi citada anteriormente mas que acabou não sendo possível utilizar. Na Figura 9 é possível observar a construção, mas sem o suporte.

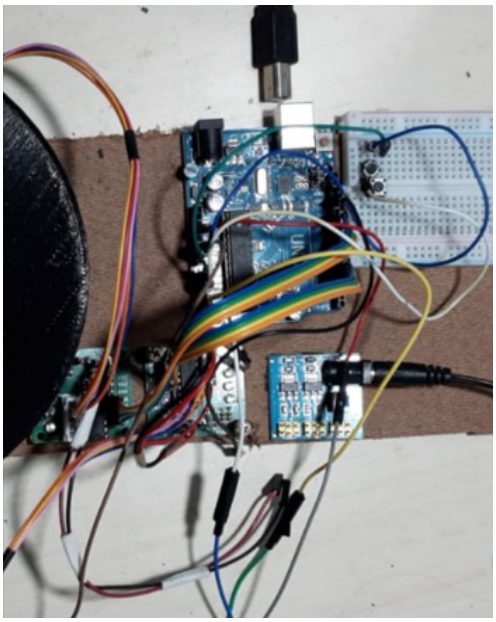
Figura 9 – Base Giratória.



## 4.2 Conexões Elétricas

Para conseguir conectar todas os componentes às suas respectivas fontes de tensão e controle, foram utilizados Jumpers e uma Protoboard. A escolha da Protoboard, embora não muito atraente esteticamente, foi necessária pois tinha-se o receio de utilizar uma placa de circuito impressa que poderia danificar as conexões dos drivers ou até mesmo estragar alguns dos motores com a soldagem errada das trilhas, e como são equipamentos que o custo não é tão baixo, foi preferível utilizar a Protoboard. Na figura 10 está a vista de cima das conexões.

Figura 10 – Conexões Vistas de Cima.

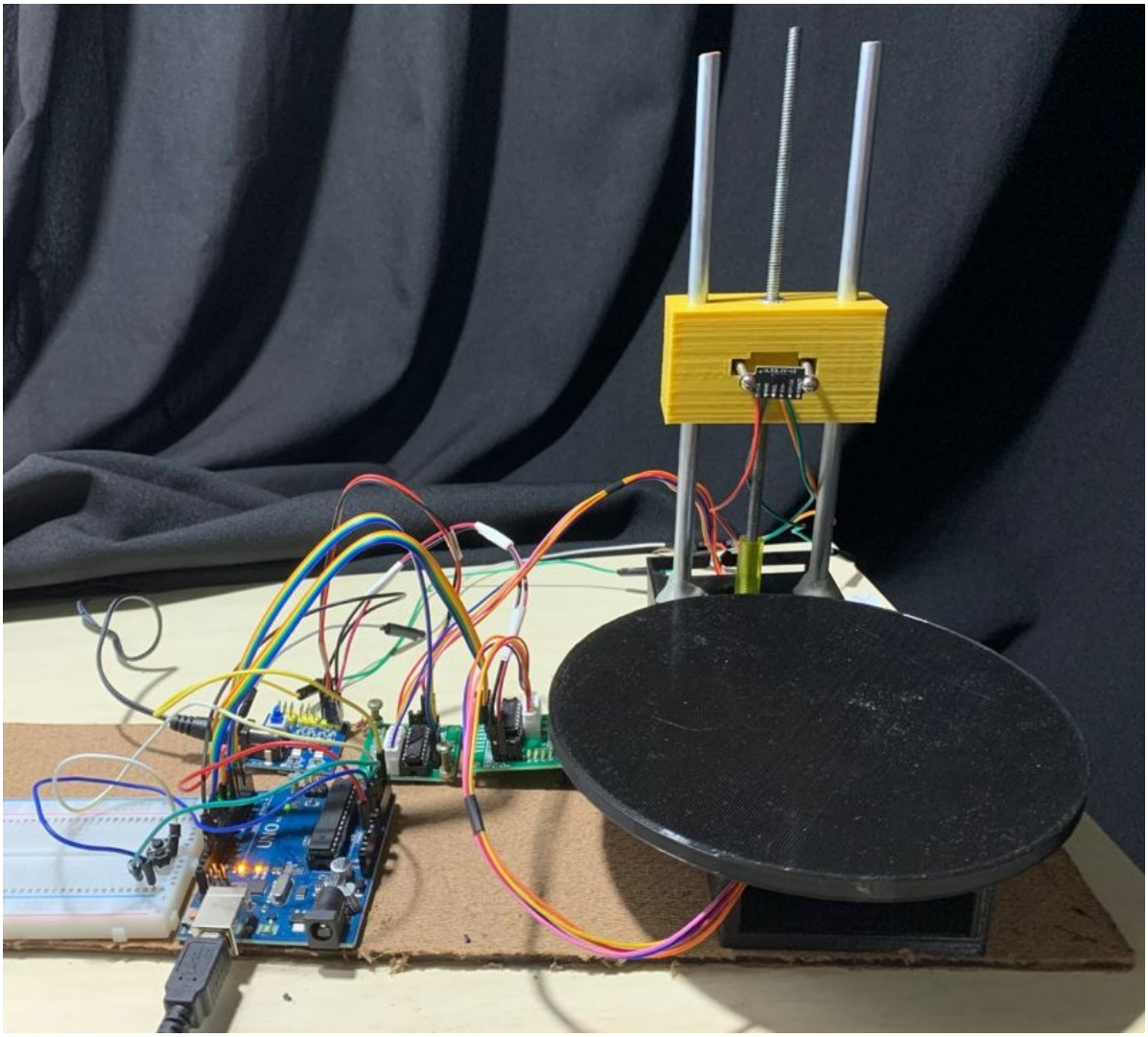


## 4.3 Testes finais do protótipo

Ao finalizar as construções do projeto, foram testados os motores de passo para analisar o comportamento das peças e também para descobrir a relação entre a quantidade de giros do eixo do motor, que está acoplada a vara rosqueada, e a medida que move verticalmente, em milímetros, a peça que contém o sensor. Assim, foi pensado em fazer um ciclo de: dar um giro completo no objeto, fazendo o respectivo motor de passo girar uma volta, isso com o sensor em uma altura inicial, e depois, fazer o outro motor de passo, o que move o sensor, girar uma volta também, o que de acordo com testes realizados pelo grupo, fazem o sensor subir aproximadamente 1 mm. O ciclo se repete, com outro giro completo no motor da base, girando outra vez o objeto, e depois, o outro motor gira mais uma vez, incrementando mais 1 milimetro na altura, e assim se sucede até o fim do objeto. Dessa forma, o sensor vai captando todos os pontos do objeto para a criação da nuvem de pontos e os envia pela porta Serial ao programa Python. Quando for o fim do objeto, o motor de passo percorre toda altura acumulada no sentido contrário, fazendo a peça retornar a altura inicial. Uma forma de aumentar a resolução é diminuindo o incremento da altura, porém, o

tempo de escaneamento aumentará proporcionalmente. Por fim, é importante salientar que para o processo de escaneamento começar é necessário o acionamento de um botão conectado à placa Arduino e colocado para facilitar a ação. Na Figura 11 é possível observar as conexões prontas.

Figura 11 – Protótipo



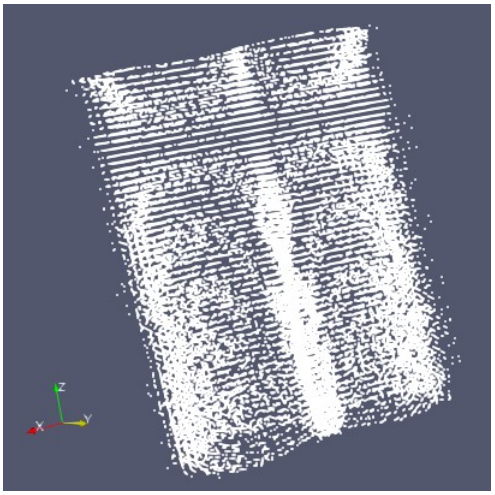
# 5 Experimentação e Resultados

Os resultados obtidos pelo grupo após o escaneamento de objetos de pouca complexidade, como caixas de baterias e esmaltes de unha, não foram satisfatórios quanto o esperado. Apesar do sensor conseguir captar os pontos, alguns ruídos e a falta de um software de pós-processamento dos dados resultaram em uma nuvem de pontos aquém do desejado. Além disso, a inconsistência dos objetos físicos que fazem parte da plataforma somada a falta de suavidade oferecida pelas peças quando o escâner está em funcionamento geraram inconveniências relacionadas a instabilidade tanto do sensor subindo e descendo quanto do objeto girando. Como principal exemplo, a vara rosqueada por apresentar imperfeições em sua

ranhuras acaba diminuindo a qualidade do escaneamento pelo fato da porca não realizar um movimento suave ao decorrer de seu trajeto, assim captura-se diversos pontos indesejados do ambiente e perde-se precisão nas medições. Todos esses pequenos detalhes geram erros na criação da nuvem de pontos dos objetos, visto que é um trabalho delicado e minucioso. Mas, apesar dessas falhas, que podem ser corrigidas em projetos futuros com o acerto dos detalhes, o projeto não fica para trás. Como exemplo de um escaneamento feito, abaixo segue a figura 12 do objeto de referência escaneado, e na figura 13, e sua respectiva nuvem de pontos.

Figura 12 – Objeto de Referência


Figura 13 – Nuvem de pontos gerada



# 6 Conclusão

Este projeto foi desenvolvido com o fim de permitir que entusiastas ou interessados na área de escaneamento 3D pudessem construir um escâner tridimensional com materiais

de baixo custo e fácil acesso. No entanto, problemas como a instabilidade oferecida pela estrutura ao sensor e nenhuma limpeza dos dados obtidos, captando não só pontos reais, mas também ruídos, fizeram com que o objetivo não fosse totalmente alcançado, visto que apesar de ser possível captar pontos, a qualidade da nuvem criada ainda peca. Nesse sentido, ressalta-se que esse é um trabalho em andamento, deixando para trabalhos futuros aperfeiçoamentos no que tange tanto a obtenção dos pontos e criação da nuvem de pontos, quanto à limpeza de dados, como a retirada de ruídos. Além disso, aperfeiçoamentos na construção da estrutura e na precisão dimensional da montagem são necessários.

Todavia, a oportunidade de desenvolver esse projeto, embora em condições atípicas, devido a pandemia, foi primordial para a finalização do curso de Automação Industrial. Apesar de que não tenha-se atingido todos os objetivos aguardados, o produto final, à vista do aprendizado gerado, satisfaz as expectativas criadas. O conhecimento exposto sobre diferentes aspectos da automação e computação para construção do escâner, além da elucidação das imperfeições a serem consertadas que geraram os erros obtidos, sustentam futuros projetos e servem como um ótimo subsídio de estudo, recompensando todo trabalho feito.

# Referências

BRITES, Felipe Gonçalves; SANTOS, Vinicius Puga de Almeida. *Motor de Passo*. Dissertação (Mestrado) — Universidade Federal Fluminense - Engenharia de Telecomunicacoes, Niterói, 2008.

FURLAN, Gustavo Ambrozini. *Esquema Elétrico do Arduino - O Guia Definitivo*. 2018. Disponível em: <https://www.circuitar.com.br/tutoriais/esquema-eletrico-do-arduino-o-guia-definitivo/index.html>. Acesso em: 27/08/2021.

INC., Onshape. *Onshape: product development platform*. 2021. Disponível em: <https://www.onshape.com/en/platform>. Acesso em: 20/05/2021.

I²C - Everything you need to know. 2016. Disponível em: <https://www.mikroe.com/blog/i2c-everything-need-know>. Acesso em: 31/08/2021.

LAKOVIć, Nikola et al. Application of low-cost vl53l0x tof sensor for robot environment detection. In: *2019 18th International Symposium INFOTEH-JAHORINA (INFOTEH)*. [S.l.: s.n.], 2019. p. 1–4.

MARTIN, Robert C. *Clean Code: A Handbook of Agile Software Craftsmanship*. 1. ed. [S.l.]: Prentice Hall PTR, 2008. v. 1. 431 p. (Robert C. Martin Series, v. 1).

MEKURIA, R; LI, Z; TULVAN, C. Evaluation criteria for pcc (point cloud compression). Geneva, 2016. Disponível em: <https://mpeg.chiariglione.org/standards/mpeg-i/point-cloud-compression/evaluation-criteria-pcc>. Acesso em: 22/08/2021.

PATEL, Prachi Talati Dr. Sagar; GANDHI, Ssaniya. *Design of I²C Protocol*. Dissertação (Mestrado) — Charotar University of Science and Technology, Índia, 2019.

PICORETI, Rodolfo. *Entrada e saída - Manipulando registradores*. 2017. Disponível em: <https://portal.vidadesilicio.com.br/entrada-e-saida-manipulando-registradores/>. Acesso em: 27/08/2021.

STMICROELECTRONICS. *VL53L0X World's smallest Time-of-Flight ranging and gesture detection sensor*. [S.l.], 2018.

STROUSTRUP, B. What is object-oriented programming? *IEEE Software*, v. 5, n. 3, p. 10–20, 1988.