Universidade de São Paulo
Instituto de Matemática e Estatística
Bacharelado em Ciência da Computação

Bruno Rafael Aricó

MAC0499 - Trabalho de Formatura
LIDARO
Um Radar Laser aberto e de baixo custo

São Paulo
Outubro de 2018

LIDARO
Um Radar Laser aberto e de baixo custo

Monografia final da disciplina
MAC0499 – Trabalho de Formatura Supervisionado.

Supervisor: Roberto Hirata Jr

São Paulo
Outubro de 2018

Agradecimentos

Este trabalho é em honra a todos aqueles que me apoiaram na elaboração deste projeto, amigos e familiares.

Agradeço a todos.

Resumo

Será apresentado neste trabalho o desenvolvimento de um radar laser multi propósito criado utilizando ferramentas de fabricação digitais, tais como impressora 3D e cortadora laser e placas de prototipagem rápida. Todo o desenvolvimento foi orientado para se ter baixo custo e construído com partes que são facilmente encontráveis no mercado brasileiro. Tanto seu hardware quanto software são abertos e livres, desenvolvidos nas linguagens Python e Arduíno, com a implementação voltada para o microcontrolador ESP32.

Palavras-chave: Scanner-3D, Lidar, ESP32, Arduino.

Abstract

In this monography the development and implementation of a Lidar builded using digital fabrication tools like a 3D printer, laser cutter and prototyping boards such as Arduino will be presented.
The development is marked by the characteristics of open source and hardware, low cost and parts that are easily found in the Brazilian market.
The software are implemented in Python and Arduino laguage, with Firmware oriented to the microcontroller ESP32.
Keywords: Lidar, ESP32, Scanner-3D, Arduino.

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Motivação

O ser humano tem por natureza sua vivência intrinsecamente ligada ao espaço tridimensional e tanto a visão, quanto o cérebro adaptaram-se a esse ambiente desde a infância através da interpretação dos estímulos visuais, auditivos e sensórios externos. Por exemplo, a posição dos olhos favorece a visão estereoscópica (a percepção de espaço 3D) onde com base nela temos a sensação da profundidade do ambiente. Além disso, sensores como o aparelho vestibular localizado no ouvido nos ajudam a ter noções do posicionamento do corpo o espaço.

Como essas percepções são naturais e intuitivas, não nos atentamos conscientemente aos desafios que são fornecer e interpretar esses mesmos estímulos para um robô móvel. Além disso, mesmo em nossos sistemas perceptores humanos temos "falhas"no tratamento dos sinais que nos são fornecidos, como é o caso das ilusões de óptica (como da Figura 1.1).

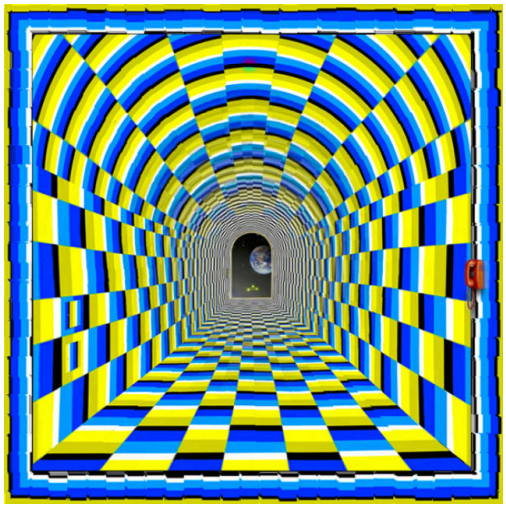

Figura 1.1: *Ilusão de óptica clássica onde o estímulo da imagem nos gera uma falsa percepção de profundidade*

Fonte: httpz://magens.mdig.com.br

Por essas e outras razões, a pesquisa e desenvolvimento de sensores é importante. A prática tem mostrado que os sistemas de localização não podem confiar em apenas um tipo de sensor, mas na combinação de muitos sensores.

1.1 Proposta

O objetivo deste trabalho foi o de desenvolver um lidar de software aberto e com estrutura e componentes facilmente encontráveis no mercado. Para isso foi utilizada peças impressas em impressora 3D, tornando-as reprodutíveis, e hardware não específico como sensores de distância de aplicação genérica, como os utilizados em pinças robóticas e microcontroladores comuns.

A construção do protótipo envolveu desde o design em CAD de todas as peças impressas ao estudo do melhor plástico a ser utilizado para a impressão, como vemos no capitulo 3 e o design da placa de circuito e o estudo dos elementos de hardware a serem utilizados no capitulo 4.

Também esta descrito o funcionamento e ideias adotadas na implementação do firmware e software no capítulo 5 e testes de desempenho em diversas condições no capítulo 6.

O desenvolvido neste trabalho utiliza-se fundamentalmente do princípio da reflexão luminosa e do tempo de viagem da luz desde sua emissão até a recepção por um sensor, conhecido por ToF (*Time of Flight*).

Em vista de o projeto ser open-source, o nome Lidaro refere-se ao acrônimo de origem do nome Lidar unido a terminação "o"de "Open"fazendo referência a sua implementação aberta, sendo um dos poucos Lidares que possuem a funcionalidade de scan 3D com esta característica. Este é o primeiro projeto de lidar de código e hardware aberto, com tecnologia ToF, implementado no mundo.

Capítulo 2

Tecnologias Lidar

Em diversas áreas do conhecimento como engenharias, arquitetura e geologia, por exemplo, há a necessidade de se obter medidas de algum ambiente ou objeto. A tecnologia Lidar (um acrônimo para **Li**ght Ra**dar**) tem sido cada vez mais usada devido a necessidades de se obter uma medida precisa e que facilite uma representação virtual de tais ambientes e objetos. Nesse sentido, há uma grande diversidade de dispositivos diferentes que podem ser utilizados para esta finalidade, porém a base de todas é essencialmente a mesma, uma fonte de luz coerente, mais precisamente, o Laser e um sensor para medir sua reflexão.

2.1 Aplicações da tecnologia Lidar

A aplicação da reconstrução de cenas ou objetos virtualmente é empregada desde a década de 70 onde foi utilizada com lasers altimétricos (Veja um exemplo na Figura 2.1) a bordo da espaçonave da missão Apollo 15 da NASA (*National Aeronautics and Space Administration*) para realizar o levantamento topográfico de algumas regiões da Lua. A acurácia do equipamento na época era suficiente para ser disparado a uma distância de 30km da superfície e ter 10m de resolução. O sucesso e a importância dos dados obtidos foi tão grande que essa mesma técnica foi utilizada nas missões Apollo subsequentes.

Na década de 80, ela também foi utilizada para o perfilamento da atmosfera e oceano quanto a sua composição baseando-se na absorção e reflectância de um laser; este foi o caso do LITE (*Lidar In-Space Technology Experiment*) que orbitou a Terra dentro do ônibus espacial por nove dias, obtendo dados da distribuição e composição de nuvens, partículas na forma de aerossóis dispersas pela atmosfera e poluentes. Com a miniaturização de seus

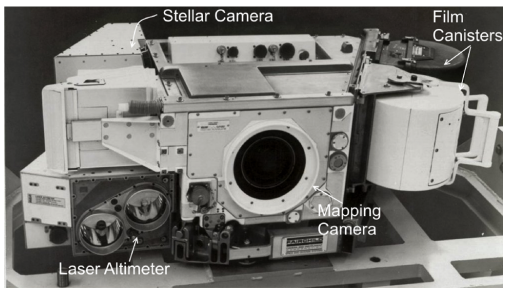

Figura 2.1: *Fotografia do altímetro laser utilizado na missão apolo 15.*

Fonte: http://apollo.sese.asu.edu

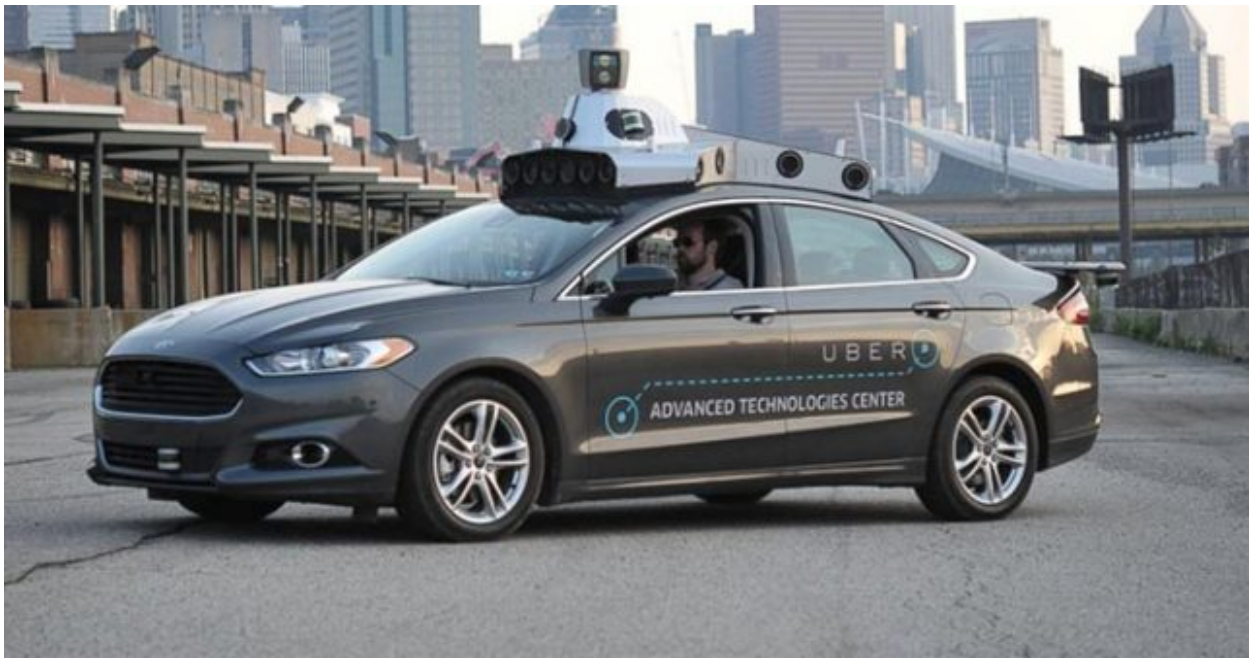

Figura 2.2: *Volvo XC90, modificado pela Uber para ser um carro autônomo, destaque para o Lidar sobre o teto do veículo*

Fonte: http://motorabc.com

componentes até o final da década de 90 já seria possível embarcá-lo em um avião para fazer experimentos semelhantes.

Com a miniaturização principalmente do elemento emissor laser, pode-se então transformar o Lidar em um equipamento mais portátil e encontrar-se novos empregos para a tecnologia como, por exemplo, na manutenção e preservação de construções históricas. A ideia era capturar os dados e gerar um modelo tridimensional virtual de sua estrutura para aplicações em agrimensura, fazendo levantamento topográfico terrestre por meio de aviões de pequeno porte ou drones com Lidar embarcado.

Uma aplicação em destaque, para a qual esta tecnologia vem sendo direcionada são veículos autônomos (Figura 2.2) pois, devido a grande resolução do modelo gerado e a precisão que ele possui, é possível se detectar obstáculos e planejar a movimentação do veículo com antecedência, além de poder se utilizar do ambiente virtual amostrado para se criar um mapeamento da área.

2.2 Os diversos tipos de sensores Lidar

Existem diversos tipos de arquiteturas de implementação de Lidar, em sua essência todas utilizam uma fonte luminosa coerente para obter as medidas, mas a forma como fazem a estimação de distância é diferente entre si. Assim como também existem arranjos que permitem a descrição do ambiente em duas dimensões, existem outros que permitem a virtualização em três dimensões.

Aplicações específicas podem exigir apenas a descrição do ambiente em duas dimensões, como é o caso dos robôs de faxina doméstica (ex. Roomba), onde necessitam saber apenas a planta baixa da casa para identificar a posição de paredes ou móveis, como em outras,

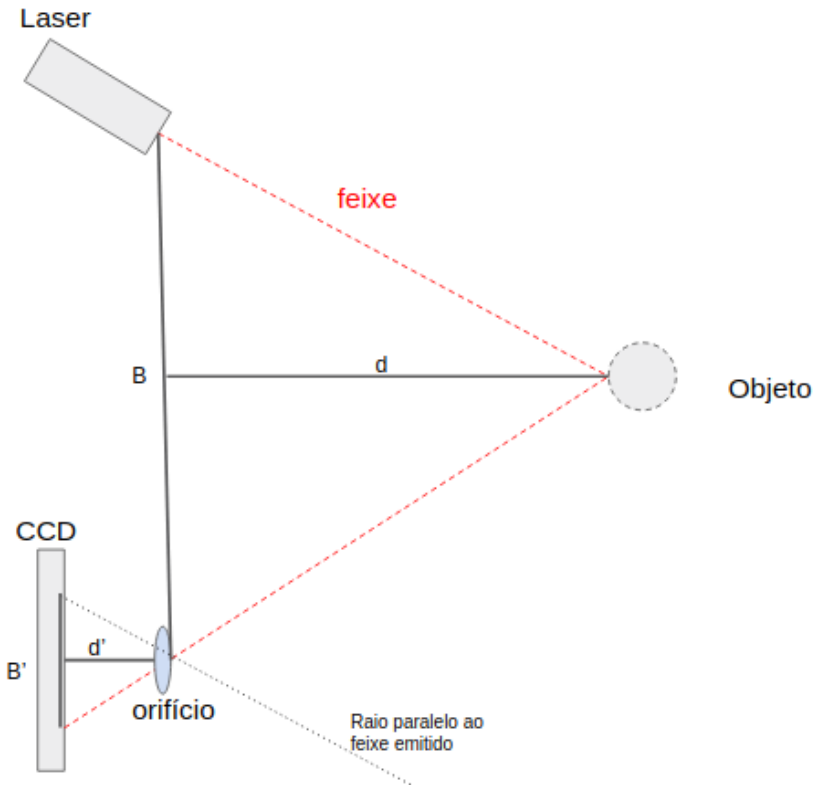

Figura 2.3: *Lidar por triangulação, calcula a distância* d *pela semelhança entre os triângulos definidos por (B,d) e (B',d')*

as orientadas para arquitetura necessitam de uma visualização em três dimensões de um cômodo de uma residência.
Nas próximas subseções apresentamos alguns dos diversos tipos de sensores existentes.

2.2.1 Lidar por triangulação

As implementações deste tipo de equipamentos estão em constante evolução, desde técnicas que utilizam a geometria do posicionamento de um objeto utilizando lasers e CCD's (*Charge-Coupled Device*) como é o caso do Lidar de Triangulação (Fig. 2.3), o qual utiliza medidas conhecidas entre a fonte de emissão laser (Laser na Fig. 2.3) até o orifício da lente (B na Fig. 2.3), e entre o sensor CCD e a lente (d' na Fig. 2.3), além do tamanho do sensor (B' na Fig. 2.3), para estimar a distância até o objeto (Objeto na Fig. 2.3) por semelhança entre os triângulos definidos por (B, d) e (B', d').

2.2.2 Lidar com sensor ToF

Outro tipo de implementação usual de Lidar utiliza sensores do tipo ToF (*Time of Flight*) que determina a distância até o obstáculo utilizando o tempo que um pulso de luz leva para ir e retornar até o sensor (Fig. 2.4), como é o caso do desenvolvido neste trabalho. Este é o tipo de Lidar mais abundante no mercado por sua relação de custo benefício.

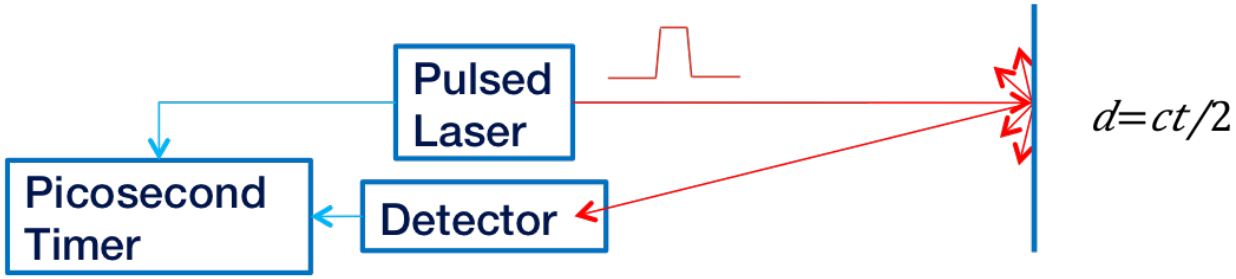

Figura 2.4: *Modelo em alto nível dos elementos de um sensor ToF*

Fonte: http://sensl.com

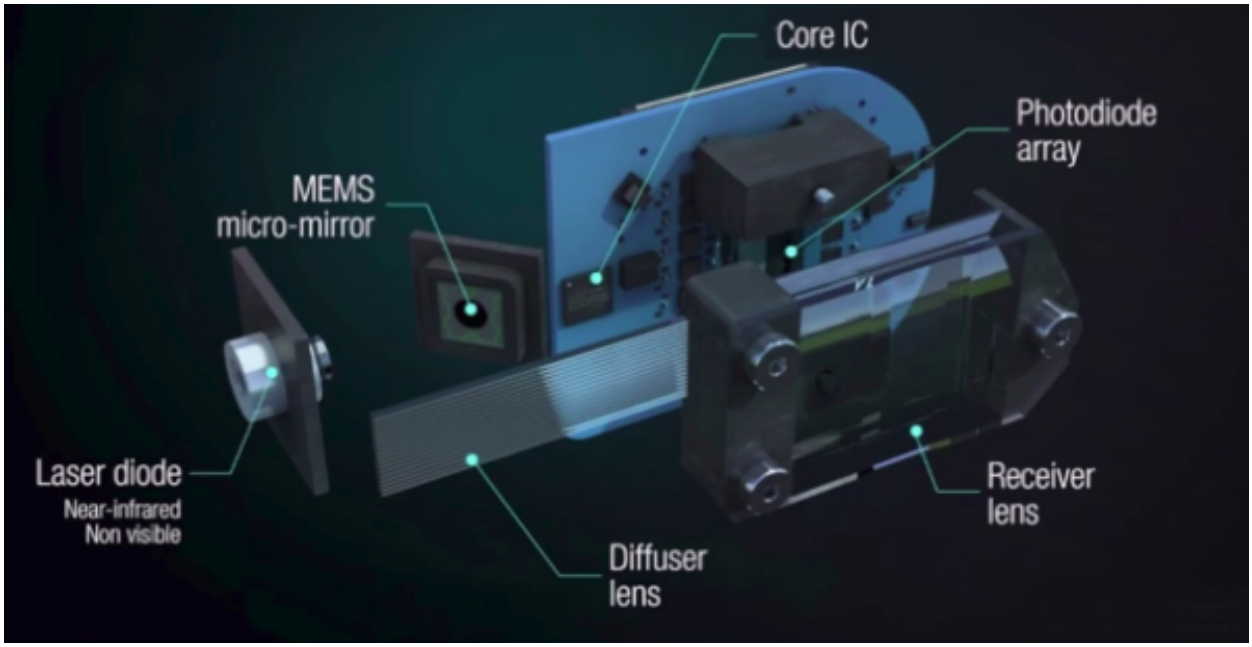

Figura 2.5: *Descrição dos componentes que constituem um modelo de Lidar estado sólido, ênfase para a falta de peças móveis.*

Fonte: https://leddartech.com/

2.2.3 Lidar Estado Sólido

O estado da arte é o chamado Lidar de estado sólido (Fig. 2.5), que usa uma topologia de funcionamento dos que possuem sensores ToF, mas tem como diferencial usar de MEMS (*Micro Eletronical Mechanics Systems*)(Fig. 2.6). No lugar de peças grandes se movimentando para a orientação do laser no alvo, esse tipo de tecnologia apresenta um micro espelho montado em um chip de silício que faz a reflexão e orientação do laser no alvo.

Esse tipo de Lidar tem como principais características seu tamanho reduzido, durabilidade e baixa vulnerabilidade a perda de calibração.

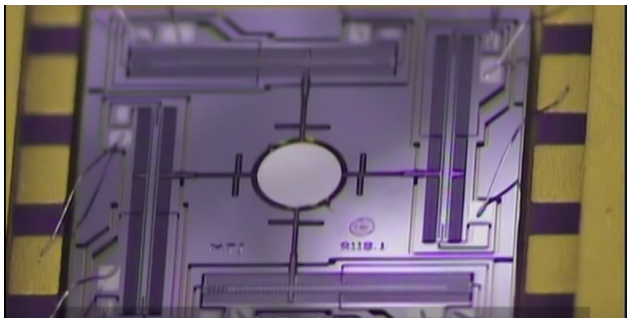

Figura 2.6: *Imagem de macroscopia de um espelho montado em chip de silício, junto com seus atuadores de orientação eletromagnéticos*

2.3 Princípios de funcionamento do sensor ToF

O laser (*light amplification by stimulated emission of radiation*) é um dispositivo que produz uma radiação eletromagnética (luz) de um comprimento de onda muito bem definido, colimada, e com velocidade de propagação no ar de aproximadamente $c = 2.9972 \times 10^8 m/s$. Essas características são muito convenientes para o uso na estimação de distâncias e, pelo comprimento de onda ser muito pequeno (da ordem de centenas de nanômetros) dá precisão suficiente para capturar pequenos detalhes.

O Lidar apresentado neste trabalho utiliza um sensor do tipo ToF o qual opera de modo semelhante a um sonar, onde uma fonte dispara pulsos sonoros cronometradamente em direção a um determinado alvo ao qual se deseja medir a distância até a fonte. Os pulsos propagam-se pelo meio (ar, ou líquido) até atingir o alvo, refletir e ser captado por um sensor próximo da fonte. O tempo de eco, isto é, o tempo que o pulso levou para ir e voltar é usado para determinarmos a distância (levando-se em conta a velocidade do som nesse meio). O mesmo principio se aplica para o ToF, mas ao invés da utilização do som, utiliza-se a luz.

O som poderia ser usado para esta aplicação também, mas há diversos problemas quando desejamos resolução muito elevada nas medições (por causa do comprimento de onda de uma onda sonora). Isso mesmo quando são utilizadas frequências elevadas nos pulsos, como o ultrassom, pois efeitos intrinsecamente predominantes em ondas mecânicas são notados, como difração. Esses efeitos também existem no caso da luz, mas eles são muito menores para essa finalidade.

Ao se utilizar um comprimento de onda muito menor e com velocidade de propagação muito elevada, surgem novos desafios como: qual tipo de material na fabricação do sensor é capaz de reagir a quantidade de luz refletida na intensidade que ela retorna do alvo, em um intervalo de tempo o suficiente para que não interfira no valor da medição da distância.

Capítulo 3

Lidaro

Neste capítulo apresentamos a proposta de um Lidar projetado totalmente no âmbito deste TCC.

3.1 Por que Lidaro?

O nome Lidaro refere-se ao acrônimo de origem do nome Lidar unido a terminação "o" de "Open", fazendo referência a sua implementação aberta. Este é um dos poucos Lidares que possuem a funcionalidade de varredura 3D com esta característica.

Além disso, do ponto de vista de custo e facilidade de implementação, sem peças específicas, ele é o primeiro no mundo com tecnologia ToF.

3.2 Visão geral do Lidaro:

A arquitetura do hardware projetado é composta de 3 partes principais (Fig. 3.1:

- Disco rotativo: parte do Lidar responsável pela orientação no eixo correspondente a altura em coordenadas esféricas. Neste projeto, o disco possui liberdade para girar 360 graus.
- Torre: parte na qual o disco é fixado, responsável pela orientação do Lidar no eixo azimutal. Ele tem liberdade de rotação de 180 graus.
- Base: parte fixa na qual é fixado o servo que suporta a torre. Dentro da base é onde se localiza a maior parte da eletrônica do projeto incluindo o microcontrolador principal

Desta maneira, o Lidaro, por essência, possui 2 graus de liberdade (em coordenadas esféricas, Fig. 3.2):

- o eixo azimutal controlado pelo servo motor;
- o eixo da altitude controlado pelo motor "brushless"(sem escovas).

Sua estrutura é montada em plástico para engenharia conhecido por PLA (polylactide), próprio para impressão em 3D; este material foi escolhido para conferir a resistência mecânica necessária para a medição levando-se em consideração o baixo custo e a acessibilidade.

O sensor ToF é posicionado sobre o disco principal do Lidar para possibilitar seu direcionamento para toda a abóbada de posições na esfera de alcance (intervalo onde as medidas do sensor são confiáveis).

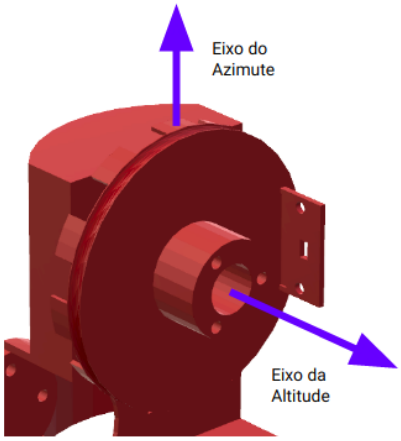

Figura 3.1: *Eixos de rotação do Lidaro*

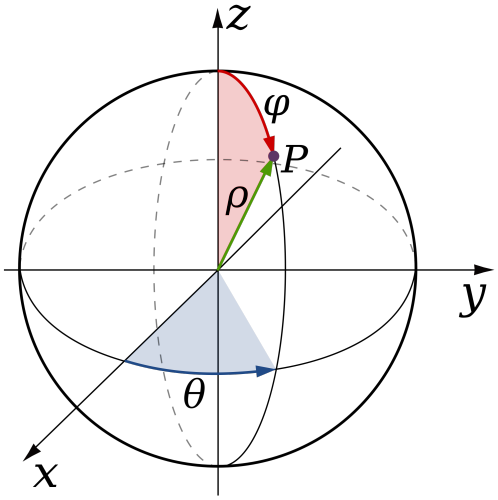

Figura 3.2: *Representação em coordenadas esféricas dos ângulos de altitude ρ e de azimute θ*

Fonte: https://commons.wikimedia.org

A aquisição de dados de posicionamento é feito de forma diferente para cada um dos eixos do Lidar. No eixo de controle da altitude temos um codificador rotativo e no eixo de controle do azimute usamos o controle de orientação do servo motor. Mais detalhes sobre a implementação serão apresentados em seções posteriores.

Operacionalmente, o Lidaro é controlado por uma interface gráfica desenvolvida no contexto deste TCC e que pode ser executada em um computador O computador comunica-se com o Lidaro por WiFi e transmite os comandos para liga-lo, desliga-lo, definir o modo de operação da varredura (3D ou 2D), a velocidade para os eixos, e a representação dos pontos coletados. Ele também possui a funcionalidade de seu software interno (firmware) poder ser atualizado remotamente.

3.3 Estrutura

Toda a modelagem da estrutura foi pensada de forma a se otimizar em termos de espaço e peso para que sua aplicação fosse multipropósito, desde uso em drones, até em pequenos veículos terrestres.

O projeto inicial foi baseado num protótipo já existente de um projeto denominado LIDAR360 (1) (Fig. 3.3), o qual tinha a capacidade de detectar objetos apenas no plano horizontal e voltado para aplicações de robôs móveis. A estrutura do projeto do LIDAR360 era composta de diversas peças impressas em 3D como suporte para o rolamento, o rotor que dá suporte ao sensor, o anel com os dentes do codificador ("encoder") e o suporte para o motor.

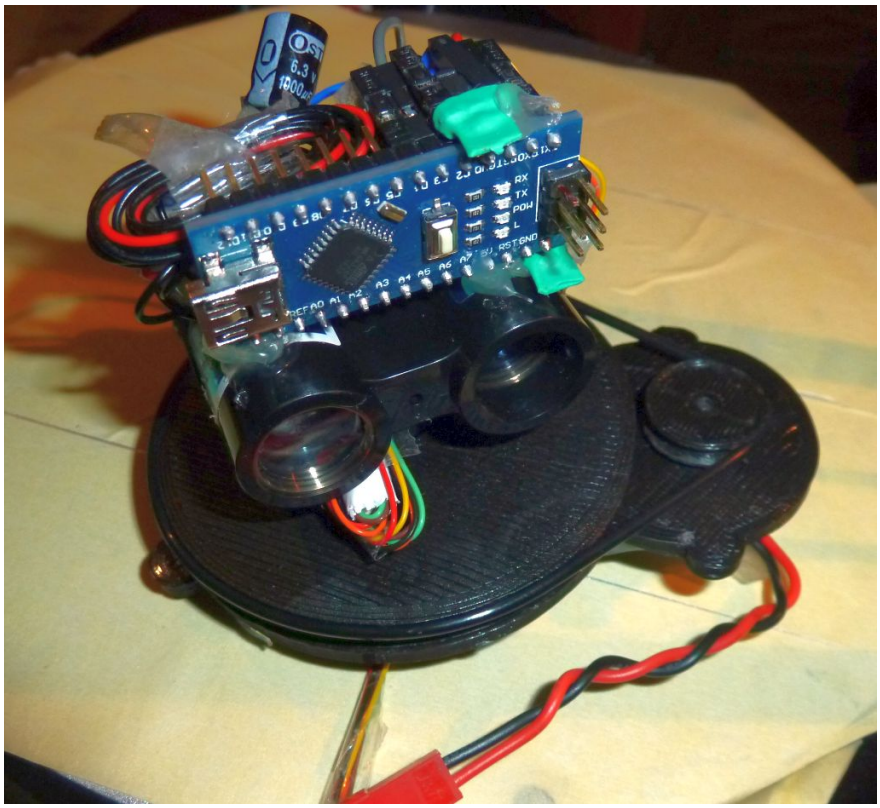

Figura 3.3: *Modelo base para a estrutura*

Fonte: http://grauonline.de

Como parte das mudanças do projeto original para o Lidaro de forma a fazer a amostragem da nuvem de pontos em volume, foi necessária a introdução de dois eixos de rotação. Além disso, usamos um sensor de menor custo em comparação ao do projeto original cujo o formato é diferente do vl53l0x.

Essas adaptações levaram a uma modificação completa do design de todas as peças (Fig. 3.4).

O disco rotativo (Fig. 3.5) teve de ser redimensionado para que o novo sensor se alocasse de uma forma conveniente sobre ele. Outra mudança foi em relação a disposição dos dentes do codificador rotativo, originalmente localizados na parte fixa do Lidar, levando o sensor infravermelho a ser posicionado no disco rotativo. Consequentemente, a fiação deste sensor teve de ser passada pelo acoplamento rotativo e o balanceamento desta peça, fundamental para o bom funcionamento do Lidar, foi dificultada.

A solução encontrada para melhorar o balanceamento foi a de posicionar os dentes do encoder no disco rotativo e o seu respectivo sensor infravermelho na torre do Lidar. A quantidade de dentes do encoder rotativo também foi alterada para possibilitar um c ontrole com maior precisão da velocidade de rotação deste eixo obtendo uma leitura mais precisa da orientação do sensor ToF.

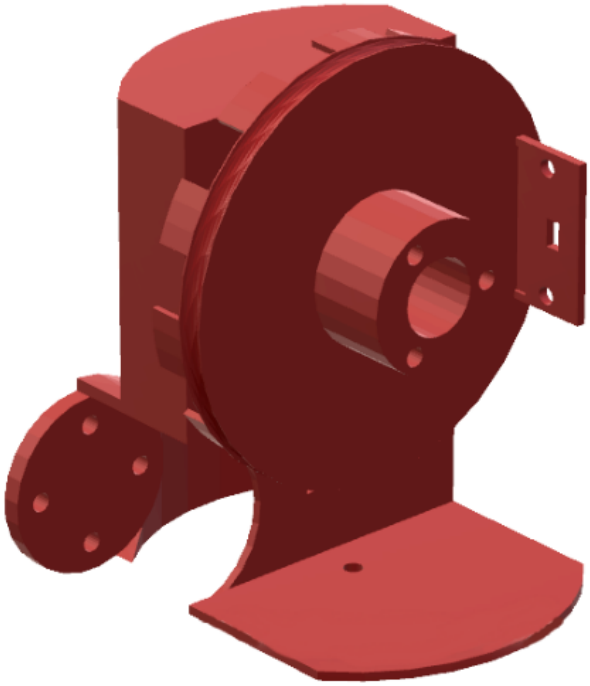

Figura 3.4: *Modelo 3D do Lidaro*

O motor de rotação do disco foi trocado por um brushless ao invés do original brushed para que tenhamos um maior torque e uma velocidade de rotação mais constante.

O posicionamento do motor na estrutura, antes fixo na base, devido a acrescentarmos um segundo eixo de rotação à estrutura foi movido para a Torre (Fig. 3.6), mantendo assim o mesmo artifício de transmissão do torque por meio de correia.

Uma polia de 9 mm de diâmetro foi desenhada para ter encaixe perfeito no eixo do motor e transmitir a rotação para o disco principal numa taxa de redução de aproximadamente 7.5:1.

A torre é posicionada aparafusada sobre um servo-motor mg995, que está fixado sobre a parte imóvel do Lidar, esta sendo uma caixa vazada que também contém parte da eletrônica de controle utilizada.

Todas as peças construídas para este Lidar foram fabricadas utilizando uma impressora 3D DIY(*Do it yourself*) (Apêndice A) construída durante o ano de 2016/2017 com a finalidade de ajudar no processo de desenvolvimento deste TCC.

O material de impressão escolhido também não foi ao acaso. Segundo (2), e observando os ensaios de tração dos plásticos mais utilizados para a impressão 3D (Fig. 4.2), PLA (poliácido láctico), ABS (Acrilonitrila Butadieno estireno) e PETG (Politereftalato de etileno glicol). Conforme o gráfico de força vs deformação, percebemos que o mais rígido de todos os plásticos, a temperatura ambiente, é o PLA, sendo ele também aquele que possui o menor custo por quilograma.

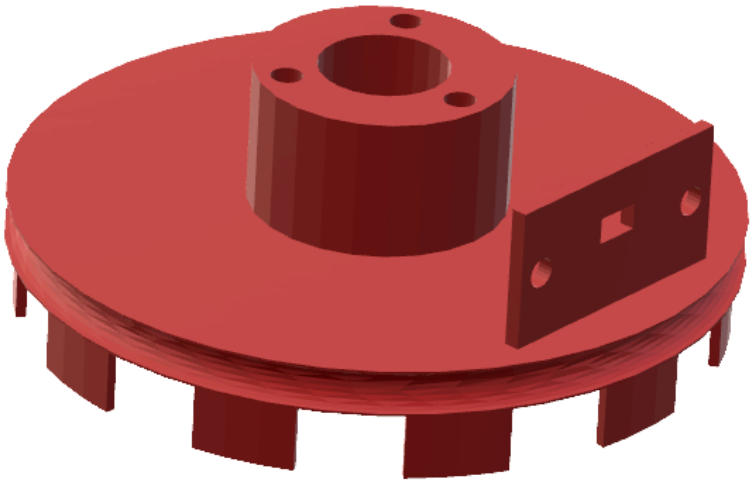

Figura 3.5: *Rotor do Lidaro em representação 3D, ênfase para o suporte do sensor e dentes do encoder*

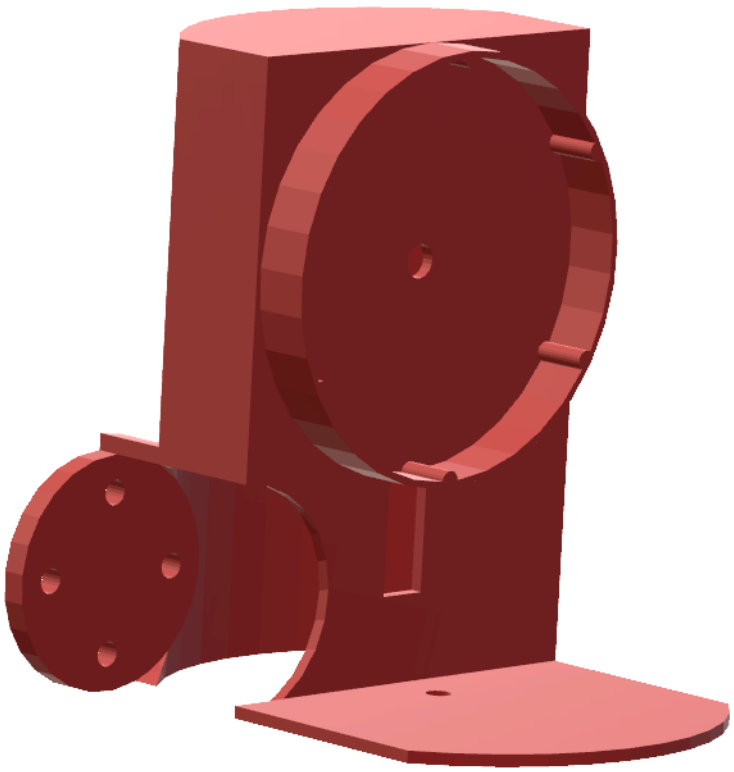

Figura 3.6: *Torre de suporte para o rotor do Lidaro, ênfase para, em seu centro o orifício de fixação do sensor IR*

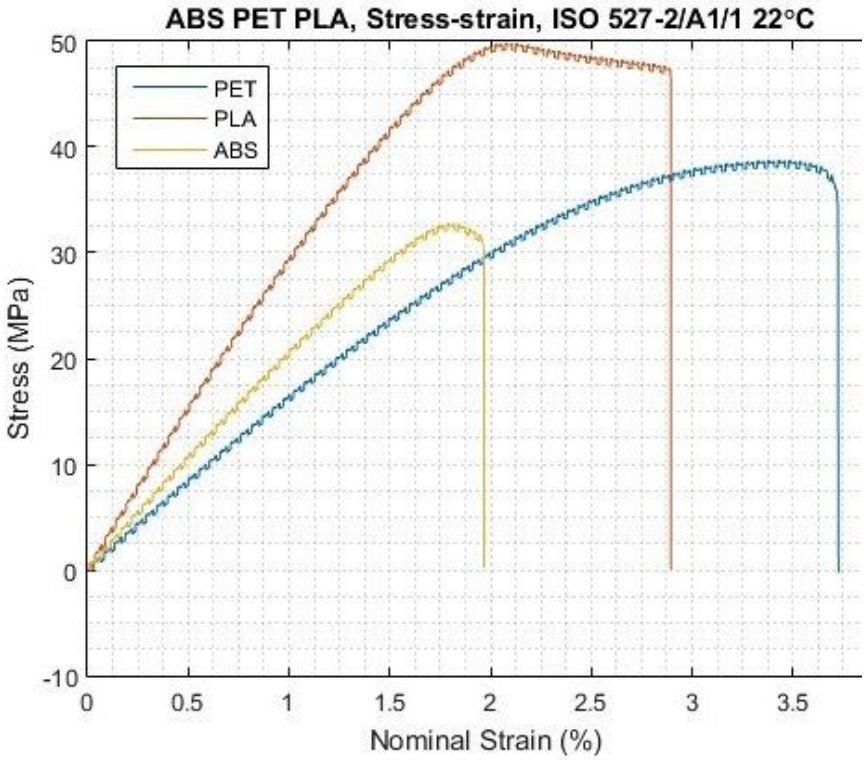

Figura 3.7: *Gráfico do ensaio de tração do PLA, ABS e PETG*

Fonte: https://www.tudosobreimpressoras3d.com.br

Capítulo 4

Lidaro: Hardware

Neste capítulo serão apresentados os elementos relacionados ao hardware. Dedicada especificamente a explicitar a motivação das escolhas de cada componente e também uma visão geral do funcionamento de cada um dos elementos eletrônicos.

4.1 Elementos de Hardware:

Referenciados aqui estão os diversos componentes que constituem o hardware do Lidaro:

- Sensor vl53l0x(3)
- Módulo IR com encoder OPB660N(4) com Comparador LM393(5)
- Módulo Acelerômetro MPU6050(6)
- Placa de desenvolvimento Heltec-ESP32(7)
- Fonte Gaiola 12V-5A
- Servo-motor MG995(8)
- Motor brushless A2212(9)
- ESC (*electronic speed controller*) 30A
- Acoplamento rotativo srm12-06a
- Filamento PLA para impressora 3D
- Rolamento 6807zz
- Cabo multivias
- Parafusos M3
- Porcas M3
- Placa de fenolite
- Percloreto de Ferro
- Conectores macho e fêmea de 3 e 4 terminais
- Elástico para dinheiro

4.1.1 Sensor ToF

Os sensores ToF disponíveis no mercado acessível a público comum é limitado, principalmente quando é levado em consideração o custo. Na prospecção feita para encontrar o sensor ideal para a construção deste projeto foram investigados os sensores vl53l0x(3), vl53l1x(10), vl6180x(11), TF02(12), lite v3(13).

	Preço (R$)	**Taxa de amostragem(Hz)**	**Distância máxima (cm)**	**Disponível no Brasil**
vl53l0x	55.00	50	200	sim
vl53l1x	80.00*	50	400	não
vl6180x	16.00*	10	60	não
Lite V3	522.00*	500	4000	não
TF02	400.00*	100	2200	não

Tabela 4.1: *Tabela de comparação de características dos sensores ToF escolhidos para a prototipagem do Lidar. Para os valores indicados por * foi feita a conversão direta do valor do dólar cotado a R$4.05*

Analisando os dados da tabela 4.1, a maioria dos sensores não se podem ser encontrados em território brasileiro, com exceção apenas do vl53l0x. Assim quando feita uma análise da relação de custo *vs* distância *vs* taxa de amostragem temos que o mais indicado para o projeto é o vl53l0x.
Muitos dos Lidares implementados em projetos de hardware aberto que usam alguma placa semelhante a Arduino usam o Lite V3 por ser o sensor mais recomendado para esta aplicação. Na sequencia temos o TF02, pois tanto a taxa de amostragem quanto a distância máxima são ótimas, porém com custos altíssimos.
Os sensores da série vlXXXXx são utilizados para aplicações diferentes de escaneamento, como é o caso da medida de distâncias lineares para abertura e fechamento de pinças de braços robóticos, assim como em detectores de gestos.

Especificações do sensor vl53l0x

Figura 4.1: *Sensor VL53L0X usado no Lidaro, o CI está montado sobre o uma placa para facilitar as ligações*

Fonte: https://arduinotech.dk

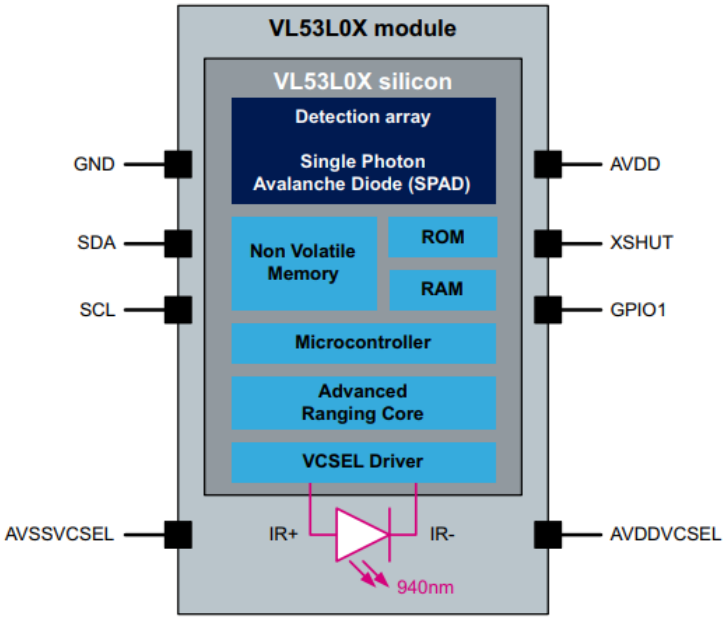

Figura 4.2: *Topologia dos elementos funcionais implementados in sílico*

Fonte: datasheet do vl53l0x

Este sensor utiliza pulsos de luz na faixa do espectro correspondente ao infravermelho (cerca de 940nm), com uma distância efetiva de medição de até 2 metros. É implementado em uma única pastilha de silício com diversos elementos funcionais tais como o detector de fótons, na arquitetura SPAD(*Single Photon Avalanche Diode*), unidade de ROM(*Read only Memory*), RAM(*Random Access Memory*) e EEPROM(*Eletrically Eraseble Programmable Read only Memory*), um microcontrolador, um setor dedicado ao controle do laser denominado por ARC(*Advanced Ranging Core*), o driver de disparo do laser, o VCSEL (*Vertical Cavity Surface Emitting Laser*) Driver , e o diodo laser em si. O fato de possuir emissor e receptor em um único encapsulamento é o que o torna um dispositivo altamente compacto quando em relação aos outros sensores candidatos que não são da série vlXXXXx.
A tecnologia do detector, conhecida como SPAD(14), define uma classe de detectores luminosos que possuem resolução da ordem de picosegundos para a detecção da chegada de fótons, onde esses podem ser pequenamente quantizado, tornando assim este um detector de alta sensibilidade. Enquanto que o emissor é um laser da categoria de VCSEL(15), essa implementação nos garante que a divergência do feixe de luz seja menor do que as outras implementações possíveis para lasers em silício. Isso nos oferece uma resolução maior para o objeto detectado pelo sensor.
O microcontrolador no conjunto implementado é responsável pelas operações em conjunto de emissor e receptor e também pelo envio de dados pelo protocolo I2C (*Inter-Integrated Circuit*).
Por existir esse microcontrolador também temos a disposição uma API (*Application Programming Interface*) pela qual podemos enviar comandos que definem o modo de operação do CI; operações como de calibração e inicialização do sensor, inicio e parada da coleta de dados, qual a acurácia desejada e o modo de amostragem (disparo único, onde uma única medida é feita, disparo continuo onde logo após uma medida feita outra já é realizada na sequência e o modo *delay*, onde se escolhe intervalos periódicos para o disparo do sensor).
A topologia da comunicação entre o *host* (Figura 4.3), o microcontrolador principal, um ESP32, e o VL53L0X pode ser representado por este diagrama presente no próprio datasheet

do sensor.

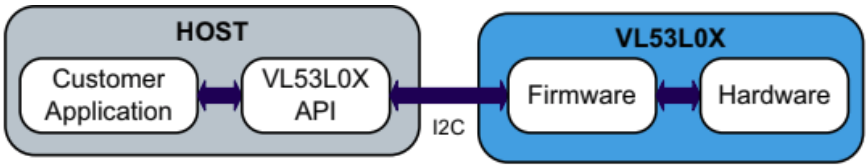

Figura 4.3: *Estrutura de comunicação entre o sensor e o ESP32 (Host)*

Fonte: datasheet vl53l0x

Limitações para o projeto:

Como comentado na seção anterior, para a escolha desse sensor no projeto foram levadas em considerações questões como custo-beneficio. Como o VL53L0X é um sensor relativamente barato para as capacidades que possui, foi a escolha ideal perto de outros disponíveis no mercado. No entanto ele possui algumas limitações para finalidades práticas, mas não para validação do modelo do Lidar construído. As limitações do sensor escolhido são:

- Comprimento de onda do laser: O comprimento de onda do laser é abundante no espectro normal da luz solar, medidas em campo aberto ou com excesso de iluminação podem ser influenciadas.

- Amostragem: Dada a frequência de amostragem máxima desse sensor (50 Hz), a velocidade de rotação do Lidar não pode ser muito elevada. Para assim conseguir uma densidade de pontos na representação que seja interessante para as aplicações em geral. Assim para aplicações que exijam alta velocidade no mapeamento se tornariam inviáveis com esse tipo de sensor.

- Distância: O sensor garante uma boa resolução das medidas para distâncias menores que 2.0 metros, portanto para aplicações onde se deseja realizar a virtualização de ambientes com diâmetro máximo igual ou superior a essa distância este sensor não é indicado.

Tendo essas limitações em vista para este sensor, de forma a tornar factível de se conseguir ampliar suas capacidades, o rotor principal foi construído de forma independente, assim sendo possível acoplar um outro modelo de sensor mais robusto e com um limite de medição maior. Também foi utilizado o protocolo de comunicação I2C, o qual é amplamente comum em sensores deste tipo, facilitando também a integração no software.

4.1.2 O microcontrolador, ESP32

Figura 4.4: *Integrado do microcontrolador ESP32*

Fonte: http://digikey.ca

Figura 4.5: *Placa de Prototipagem rápida utilizada da Heltec*

Fonte: http://ojisanseiuchi.com

O Microcontrolador utilizado é um ESP32 (Figura 4.4) da fabricante Espressif, a bordo da placa da fabricante Heltec (Figura 4.5). Nesse dispositivo são embarcados interfaces para comunicação Bluetooth, WiFi e Lora. No desenvolvimento deste projeto foi utilizado apenas o WiFi para comunicação.
A bordo desta placa existe também um display OLED (*organic light-emitting diode*) de 0.96 polegadas que pode ser utilizado como debug local. Esta placa nos oferece uma interface de comunicação USB-Serial por onde enviamos para a ROM do microcontrolador o binário compilado do firmware.
O microcontrolador possui arquitetura de 32 bits e dispõe de dois núcleos de processamento, operando ambos a clock máximo de 80MHz. Em termos de memória, possui 448 kBytes de memória ROM, 520 kBytes de memória RAM e 4 Mb de memória Flash. Dispõe de 48 pinos de entrada e saída (GPIO), onde qualquer um deles pode ser configurado como pino de interrupção e com exceção dos pinos de 34 a 39 todos podem exercer a função de gerar pulsos PWM (*Pulse Width Modulation*).

O ambiente de programação utilizado para este microcontrolador foi a IDE para desenvolvimento Arduino, onde ele foi programado na linguagem Arduino, a qual é essencialmente da linguagem C++ com algumas modificações.

A escolha deste microcontrolador

O principal motivo que levou a escolha deste microcontrolador para o protótipo final, foi a existência de WiFi a bordo, unido ao fato de ser multi-core, o que tornaria muito mais eficiente. Com essas características implementação pode então compensar em parte o fato da baixa frequência de amostragem do vl53l0x. Testes também foram realizados com um Arduino Nano, mas a necessidade de um módulo externo para comunicação sem fio e sua menor capacidade de processamento tornaram inviável pela relação de custo benefício.

4.1.3 O sensor acelerômetro e giroscópico MPU6050

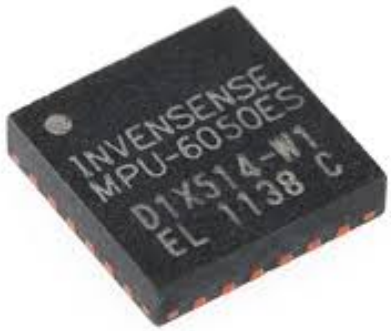

Figura 4.6: *Integrado do MPU6050*

Fonte: https://www.boxelectronica.com

O sensor MPU6050 (Figura 4.6) é uma fusão de acelerômetro e giroscópio em um único encapsulamento de silício, concedendo a ele 6 graus de liberdade. Possui também, por fins de calibração interna, um sensor de temperatura.

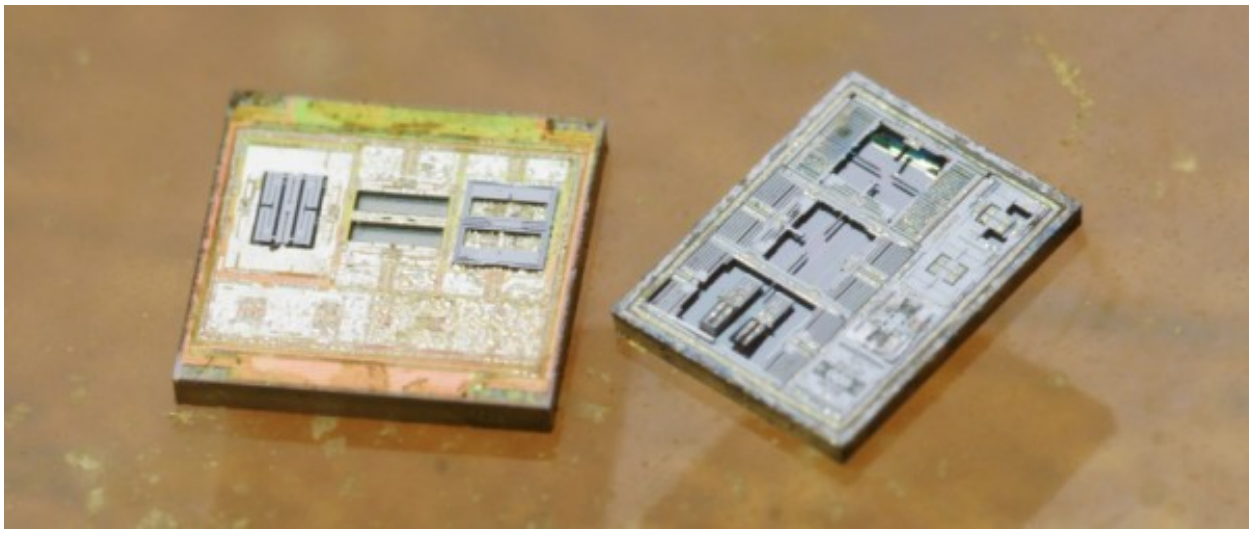

Figura 4.7: *Imagem da implementação interna do CI MPU6050, onde podemos ver a implementação eletro-mecãnica do dispositivo acelerômetro e giroscópio*

Fonte: http://www.electronicslab.com

Classificado por sendo um sensor do tipo Sistema microeletromecânico (MEMS), em seu interior possui pequenas partes móveis (Figura 4.7) que dada a aceleração ao qual é submetido gera pequenas alterações de corrente, que são mapeadas em unidades de aceleração.
Na mesma pastilha de silício ainda possui um DMP (*Digital Motion Processor*) responsável por processar internamente as variações de correntes resultante dos MEMS e fornecer os dados de aceleração linear e angular diretamente em sua saída.
Esses valores são devolvidos pelo DMP através de 6 ADC's de 16 bits cada, sendo um para cada grau de liberdade, onde por meio do controle da resolução desses ADC's consegue-se ajustar a sensibilidade do acelerômetro e giroscópio a movimentação.
Para facilitar sua integração com outros microcontroladores, possui disponível como protocolo de comunicação o I2C. Esta característica se encaixa às necessidades deste projeto, pois assim como o vl53l0x, também utiliza-se deste protocolo reduzindo significadamente a complexidade do Hardware.
A necessidade de se utilizar um sensor acelerômetro no Lidar surge principalmente de se obter a inclinação da superfície sobre o qual ele está situado, ausentando-o de um nivelamento manual.
O sensor MPU6050 foi o escolhido para o uso dentre o universo de sensores inerciais existente, devido a sua facilidade de uso. É muito comum em projetos com Arduinos, sendo barato e também facilmente encontrado no mercado, tanto brasileiro quanto internacional.

4.1.4 A escolha dos Motores

Quatro tipos de motores foram selecionados como possíveis candidatos para movimentarem os eixo de orientação do sensor.
São eles os motores do tipo, Brushed (escovado), Brushless (não escovado), servo-motor e motor de passo. Cada um deles apresenta, geralmente, características particulares.

- **Motor Brushed**: O motor do tipo escovado é conhecido por ser um dos motores mais utilizados em aplicações gerais que se necessitam de torque.

No entanto este tipo de motor tem a sua potência mássica baixa [1], sendo ainda ele um motor com baixo rendimento energético efetivo. Como vantagem o seu baixo preço e a facilidade no uso sendo ele um motor bifásico simples, necessitando apenas de um pequeno driver de potência para ser controlado por um microcontrolador.

Figura 4.8: *Exemplo de motor Brushed usado em teste de validação*

Fonte: http://ifuturetech.org

- **Motor Brushless**: Os motores do tipo brushless apresentam uma elevada potência mássica quando comparados com os outros tipos de motores aqui citados como candidatos. Sendo este bastante utilizado em aeromodelismo e drones, tem como o porém de serem do tipo trifásico, necessitando de um driver específico para controlá-lo(16): um controlador eletrônico de velocidade, que essencialmente é um microcontrolador que dispara transistores drivers de potência, para a ativação das bobinas do motor no momento correto. Este dispositivo eletrônico é conhecido comumente como ESC.

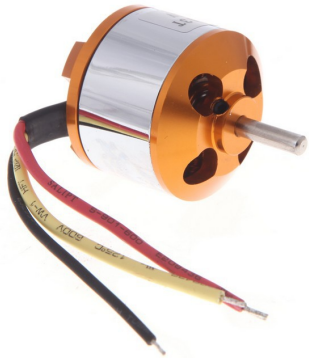

Figura 4.9: *Motor Brusheless utilizado na implementação do Lidaro*

Fonte: http://robojax.com

- **Servo-motor**: São tipo de motores utilizados por possuir elevado torque, no entanto usualmente a baixa velocidade quando comparado a motores do tipo brushed ou brushless. Tem como vantagem, além do elevado torque também oferecem o posicionamento de seu eixo com a precisão de alguns graus. Os modelos comumente encontrados aplicáveis ao escopo do projeto são também utilizados em aeromodelismo, sendo uma ótima opção por terem pouca massa e serem acessíveis tanto em facilidade de se encontrar como em preço.

[1] por potência mássica se entende a relação entre potência e massa

Figura 4.10: *Servo motor mg995 utilizado na implementação do Lidaro*

Fonte: http://moviltronics.com.co

- **Motores de passo**: Sobre este tipo de motor pode-se dizer ser o meio termo entre o brushed e o servo-motor quanto a sua funcionalidade. Possui sua movimentação na resolução de décimo de graus onde ao mesmo tempo pode conferir certa velocidade a sua movimentação. Tendo como desvantagem a sua massa elevada e quando comparado ao servo motor um preço também maior por unidade de torque.

Figura 4.11: *Motor de passo 28byj-48 candidato para ser usada no Lidaro*

Fonte: https://www.protosmart.com.br

Por fim, visando as finalidades deste projeto, para a movimentação do eixo de altitude foi escolhido um motor do tipo brushless após testes com motor do tipo brushed e este não apresentar a potência necessária para a movimentação e apresentar elevada dissipação térmica.
Optou-se pelo modelo A2212, que apresenta uma ótima relação de RPM/volt, lembrando-se da restrição de que os eixos não podem girar tão rapidamente para compensar a baixa taxa de amostragem do sensor, para garantir uma resolução de pontos amostrados por grau aceitável. Porém com esta escolha abriu-se mão de se obter diretamente do motor a informação de posicionamento do sensor. A solução encontrada para isso foi a implementação de um encoder rotativo para o controle da posição deste eixo.
Para o eixo azimutal, foi escolhido o servo-motor mg995. Por se levar em conta principalmente o preço, a velocidade de rotação que deveria também ser lenta, pelos mesmos motivos já citados para o eixo de altitude e pela facilidade de adquirir-se informações de posicionamento por inferência direta do motor.

4.1.5 Acoplamento rotativo

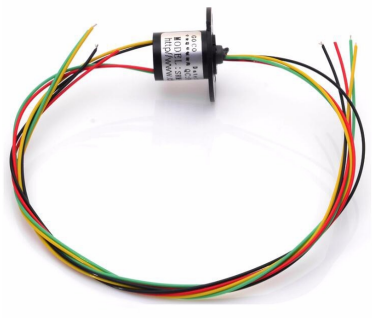

Figura 4.12: *srm12-06a, o modelo de acoplamento rotativo escolhido*

Fonte: http://pinterest.com

A peça eletro-mecânica chave para o funcionamento do lidar é o acoplamento rotativo. Instalado internamente ao rotor, por meio do qual saem os fios que comunicam o sensor ToF que está no disco do lidar com o microcontrolador. Esta peça é chave pois ela evita que a devido ao giro ocorra a torção do fio até que o mesmo se rompa. Este tipo de componente foi validado em um teste de longa duração de giro constante para se obter a certeza que devido aos contatos por escovas não houvessem ruídos intrínsecos desse atrito que influenciassem na comunicação por meio do protocolo I2C. Da mesma forma que foi validada a continuidade do sinal por meio de um osciloscópio (modelo DIY) DSO138 montado a partir de um kit (Figura 4.13), confirmando o baixo nível de ruído, sendo o mesmo não detectado nem na escala mínima de variação de tensão, em milivolts.

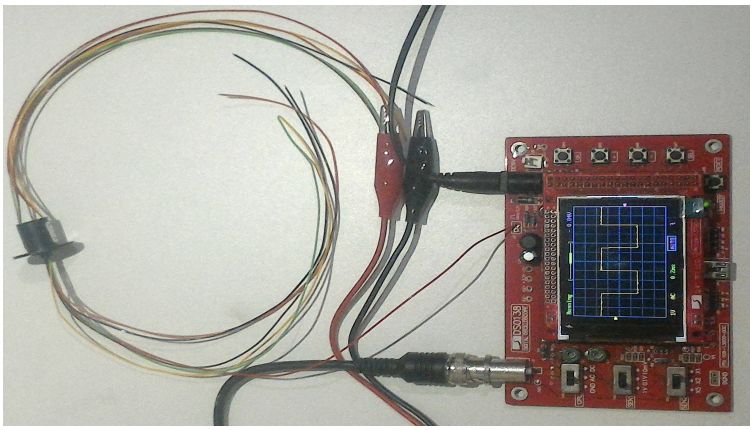

Figura 4.13: *Teste executado com o osciloscópio no acoplamento rotativo*

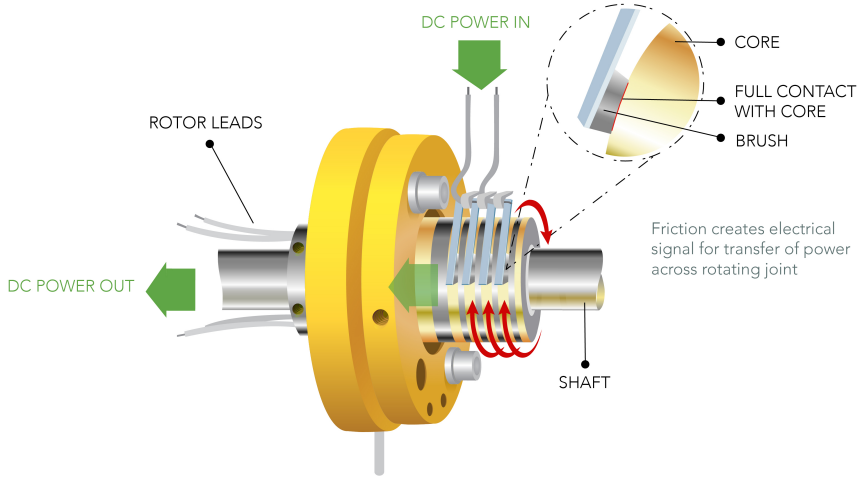

Figura 4.14: *Representação dos elementos funcionais de um acoplamento rotativo tipo tambor como o utilizado*

Fonte: https://www.element14.com

Internamente (Figura 4.14) ele é composto de um tambor montado sobre rolamentos onde há 8 trilhas circunferenciais isoladas de cobre. Individualmente sobre cada uma dessas trilhas escovas de material condutivo resistente a desgaste são postas, garantindo contato e ao mesmo tempo a movimentação circular.
Com esta montagem se garante a movimentação e, ao mesmo tempo, o contato dos fios na parte móvel e estática.

4.1.6 Fonte de Energia

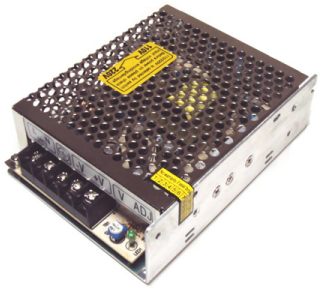

Figura 4.15: *Fonte Tipo Gaiola*

Fonte: Fonte: https://dx.com

Como fonte de energia para alimentar o sensores, motores e microcontrolador do Lidaro temos como principal opção para uma aplicação estática próxima a pontos de acesso de energia o uso de uma fonte chaveada tipo Gaiola (Figura 4.15), que pode prover 12V @ 10A.

Como segunda opção temos uma bateria de polímero de lítio. Típica de uso em drones do tipo 3S, com tensão nominal de 11.1V. O modelo de bateria utilizado (Figura 5.1) é uma com capacidade de carga de 2200mAh, por uma questão de custo. A qual fornece energia

Figura 4.16: *Bateria do tipo LiPo utilizada*

Fonte: https://www.rstore.in

para mais de 5 varreduras de 1 minuto [2]. Apesar do aumento do custo da bateria ser superior a fonte ela fornece praticidade para operar o Lidaro de forma completamente remota.

4.1.7 Placa de Circuito

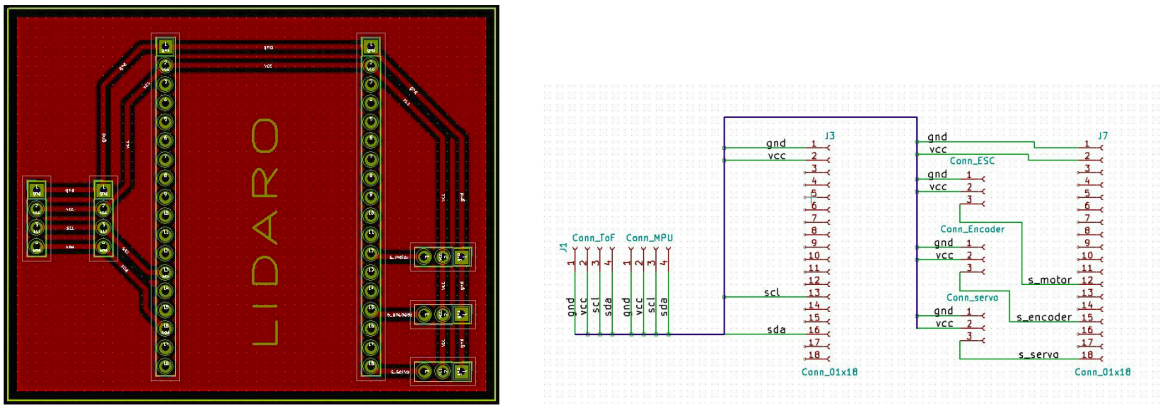

Figura 4.17: *Design da Placa de Circuito impresso e seu esquemático*

A placa de circuito do Lidaro (Figura 4.17), é essencialmente um shield para conectar os sensores mais facilmente à placa do ESP32. Nela temos os soquetes onde encaixamos os chicotes de cabos provenientes dos sensor vl53l0x, MPU6050, encoder e também os chicotes de ambos os motores. A alimentação para todos os componentes do circuito vem diretamente do regulador de tensão interno do ESC. Esta placa foi modelada no software Kicad. Por consistir de uma placa com apenas em uma face contendo trilhas foi implementada usando a técnica de fabricação caseira de placas de circuitos impresso. Nesta tecnica as trilhas são desenhadas a mão com caneta de permanente recobrindo o cobre com um filme protetor

[2] Valor obtido por testes, ainda não se foi realizados o experimento de tempo de funcionamento total para exaustão total da bateria

de tinta. Na sequência é imersa em solução a 70% em concentração de percloreto de ferro, onde então ocorre uma reação de oxirredução e temos a extração do cobre das regiões não desejadas da placa mantendo apenas as regiões de trilhas condutivas.

Capítulo 5

Lidaro: Software

Este capítulo tem sua organização dividida em duas partes. A que compete ao firmware, que é o código que sera embarcado no microcontrolador e o software, que se refere ao código executado no computador que controlará remotamente o Lidaro.

5.1 Firmware

O Lidaro possui como principais features:

- Configuração fácil para conexão com rede WiFi
- Scan bidimensional
- Scan tridimensional
- Envios de dados do Scan por meio de WiFi
- Controle de velocidade de rotação de ambos os Eixos
- Acelerômetro/giroscópio embutido para compensar desnivelamentos
- Atualização do firmware por meio de OTA (*Over the Air*), ou seja remotamente
- Plotagem dos pontos escaneados em tempo real

5.1.1 Linguagem e estrutura

A estrutura de código orientada para microcontroladores possui uma estrutura particular, pois por se tratarem de códigos embarcados onde muitas vezes possuem apenas o mínimo de interface. São rotinas autônomas e remotas que respondem a alguns comandos, geralmente programas que ficam executando em loop e/ou que são definidos por uma máquina de estados. Para facilitar a compreensão desse funcionamento os programas desenvolvidos em linguagem Arduíno possuem duas funções principais: a função `setup()` e a função `loop()`. A `setup()`, como o próprio nome diz, prepara o hardware para a execução do código principal e na função `loop()` é onde se encontra o trecho de código que executa a ação principal. Essas duas funções pela natureza da linguagem são executadas sequencialmente sempre, primeiro a `setup()` e em seguida a `loop()`. A IDE de programação utilizada foi a Arduíno IDE pela principal vantagem de facilitar o ambiente de programação como a inclusão de bibliotecas externas e possuir integrada a praticidade de executar *cross compilation* e por enviá-lo para a placa por meio de comunicação USB-Serial. A compilação

cruzada é necessária neste caso, pois o código gerado é para um controlador terceiro, no caso o ESP32, e não para o sistema nativo onde foi compilado.

5.1.2 Visão Geral sobre o firmware

Em uma descrição alto nível do firmware, temos essencialmente as classes principais:

- *Serial(17)*: por meio da qual se inicializa uma comunicação USB com algum dispositivo serial, por onde são enviados dados para debug, a taxa de transferência de dados foi definida como sendo de 115200 *baud*.

- *Wire(18)*: Responsável por iniciar o barramento do protocolo I2C e realizar a comunicação com os dispositivos contidos nesta. Em nível físico, o barramento esta definido nos pinos GPIO 5(SDA) e 4(SCL) da placa de desenvolvimento utilizada.

- *WiFi(19)*: Classe responsável por definir o modo de funcionamento do WiFi dentro do protocolo estabelecido. Esta classe tem como classe cliente a *WiFiClient*, que é responsável por instanciar um cliente TCP (*Transmission Control Protocol*) para envio de dados a um determinado IP *Internet Protocol*. Por conveniência a porta de rede utilizada para realizar a comunicação TCP foi definida por 8090.

- *WiFiUDP*: Classe que gerencia a comunicação por meio de UDP. A porta de rede utilizada para comunicação UDP (*User Datagram Protocol*) foi definida também por conveniência como sendo a 8091.

- *Servo(20)*: Classe da qual cada um dos motores, tanto o servo-motor MG995 quanto o brushless A2212, são instancias. Em nível físico o motor brushless esta associado ao pino GPIO 25 (Sinal) e o MG995 ao pino GPIO 21 (Sinal).

- *ArduinoOTA(21)*: Classe que detecta a presença de atualização de firmware e as executa remotamente, utiliza como base o protocolo FTP(*File Transfer Protocol*).

- *vl53l0x(22)*: Esta classe faz uso da comunicação I2C para realizar o acesso ao sensor vl53l0x. Por meio dela se define os modos de operação possíveis para o sensor, como por exemplo tempo de *timeout* para o retorno do sinal do laser, sensibilidade do SPAD e o modo de operação. Neste projeto o modo escolhido foi o contínuo, que permite a maior taxa de amostragem em comparação com o modo disparo único.

- *MPU6050(23)*: A função desta classe é realizar a comunicação entre o sensor giroscópio e acelerômetro MPU6050. Por meio dela se define a sensibilidade requerida do sensor, modo de operação do DMP, calibração, entre varias outras funcionalidades. Ela também faz uso do protocolo I2C.

De modo geral temos na inicialização, dentro da função `setup()`, a definicão dos protocolos utilizados, tanto serial, I2C e WiFi. A configuração do modo de operação de cada um dos sensores, instância da classe que executa a atualização OTA, a definição da porta utilizada pelo sensor encoder e a definição das propriedades da instância de cada um dos motores.

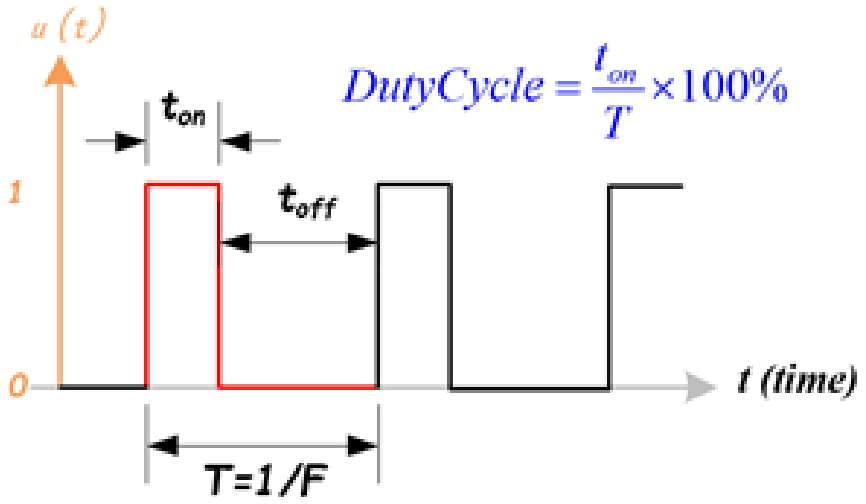

Figura 5.1: *Pulso PWM e equação que se usa para calcular o duty cycle*

Fonte: https://airsupplylab.com

Cabe citar aqui que o motor brushless é controlado de modo análogo a um servo-motor, já que essa classe Servo abstrai o controle de elementos PWM (*Pulse Width Modulation*), que é principio básico de funcionamento de ambos os motores (24).
O PWM é definido como um trem de pulsos onde por meio da razão entre o tempo que o sinal permanece em alto e em baixo, o que é conhecido por ciclo de trabalho ou *duty cycle*, se faz o controle dos respectivos atuadores.
No caso do servo uma razão de 0% de *duty cycle* fornece-nos o posicionamento do motor a 0 grau e a taxa de 100% na posição limite superior, que nesse caso é 180 graus. Para o motor brushless é análogo, levando-se em consideração que como temos o ESC interfaceando no hardware entre este motor e o ESP32 e para seu funcionamento exige uma razão miníma de aproximadamente 14% de *duty cycle*.

Ainda dentro da função `setup()` é enfática a referência a três funções específicas que são chave para o funcionamento do firmware: Existe a declaração de dois tipos de interrupções, uma delas por hardware, associada ao pino GPIO 14, onde esta ligado o fio de sinal do sensor IR pertencente ao encoder definida pela função `attachInterrupt()` e a definição de uma interrupção por *timmer* em `setup_timer()`. Uma explicação mais profunda sobre a importância dessas funções será dada na subseção encoder.
Outra função importante é a `xTaskCreatePinnedToCore()`. Nela definimos que duas *threads* serão executadas em paralelo: por meio desta função associamos ao núcleo número 1 do ESP32 a execução da função `coreTask()` enquanto que no núcleo número 0 por padrão é executada a função `loop()`.
É ressaltado que os microcontroladores tem por função principal executar ações de controle de hardwares secundários e não necessariamente de realizar processamento de grande volumes de dados, devido a seu baixo poder de processamento quando comparado com um microprocessador. Desta forma os valores amostrados são enviados em sua forma bruta para

o servidor, com apenas o processamento prévio da interpretação dos pulsos elétricos dos sensores para valores nas unidades metros e graus.
No firmware outra função a ser destacada é a `get_inclination()`, a qual mede a inclinação do Lidaro em relação a normal gravitacional, podendo fazer a compensação da inclinação da superfície sobre a qual esta apoiado. A cada inicio de varredura é feito uma leitura dos giroscópios para verificar e compensar qual a sua inclinação instantânea.

Multicore

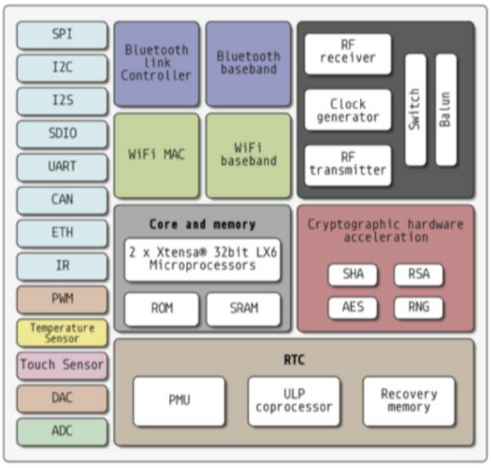

Figura 5.2: *Representação da estrutura funcional implementada em sílicio do ESP32, pode-se perceber a existência de dois núcleos Xtensa 32bits LX6*

Fonte: https://instructables.com

Como definido na função `setup()` o firmware executado é *multiThread*, cada uma das *threads* é conferia a um dos núcleos do ESP32, sendo este *Dual-Core*.
A estratégia da divisão das tarefas foi de otimizar as leituras do sensor VL53L0X sendo a sua taxa de amostragem intrínseca a única limitante e não o processamento delas, para tanto a funcionalidade dos núcleos foi definida da seguinte forma:
O núcleo de número 1 que executa a função `coreTask()` é o definido como responsável por fazer a comunicação com o computador. A ele é conferida a tarefa de fazer o envio de dados para o computador servidor, esta é a função mais crítica em consumo de tempo, por esse motivo é executada em paralelo.
No núcleo número 0 onde é executada a função `loop()`, é realizada a verificação dos comandos enviados do computador servidor para o Lidaro, a solicitação de amostras vindas do sensor ToF e a execução da movimentação do eixo azimutal pelo controle do movimento de oscilação do servo-motor.

O Encoder

O Encoder implementado no Lidaro é uma das partes mais críticas do funcionamento que garante a precisão dos pontos amostrados.
Uma vez que o rotor do eixo de controle de altitude é colocado em movimento os dentes localizados em sua periferia passam a obstruir e desobstruir periodicamente a o feixe de luz IR deste sensor, gerando assim em sua saída um sinal periódico. O período deste sinal é proporcional a velocidade de rotação e a distância entre os dentes.
No design do rotor criado o espaçamento e o tamanho de cada um dos dentes angularmente

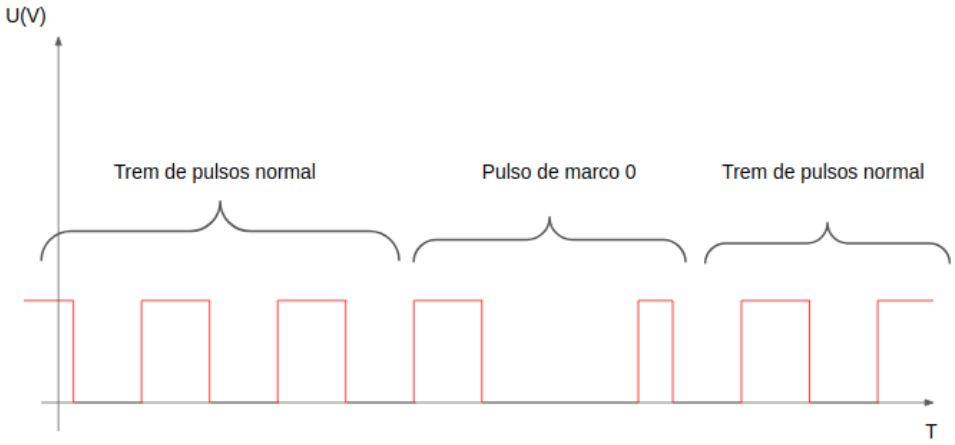

Figura 5.3: *Representação de amostra de trens de pulso recebidos pelo sensor IR do encoder, pulsos de duty cycle 50% representam pulsos normais e o que a relação é 25% representa o instante que o dente diferenciado passa pelo encoder marcando o ponto de uma rotação completa*

é equivalente a 12 graus.
Neste trem de pulsos gerados em cada uma das mudanças de estado de alto para baixo (quando o rotor transita do estado na qual interrompe o feixe para quando não o interrompe) e de baixo para alto é disparada uma interrupção de hardware. Esta interrupção atua imediatamente sobre núcleo número 1 para executar o trecho de código correspondente a função `timming()`, na qual se determina, por meio de um dente especial no encoder que provoca um pulso de menor duração (Figura 5.3), se uma volta completa foi realizada. Simultaneamente calcula-se discretamente, com base no tempo entre as interrupções, velocidade angular do rotor.
O diferencial do funcionamento desta implementação de encoder esta justamente na precisão de seu posicionamento. Um encoder normal com a resolução dos dentes de 12 graus como é o caso deste poderia apenas identificar variações a cada borda de dente, que significaria uma transição de estado de alto para baixo ou de baixo para alto, tendo assim a resolução de apenas 12 graus.
Caso o sensor obtivesse uma medida em meio a um dente ele não teria resolução para afirmar com certeza sobre este posicionamento. Para prosseguir com esta implementação então optou-se por fazer uma estimação baseada na velocidade angular, para isso utilizamos uma interrupção temporal. Com base na velocidade angular calculada a cada dente do encoder, estimamos de quantos em quantos milissegundos o rotor avançaria 1 grau e assim disparando por meio desta interrupção temporal a rotina que incrementa a contagem, a função `estimation()`, até que a próxima interrupção por hardware aconteça.
Utilizando esta técnica temos a resolução aumentada de 12 para 1 grau.

Protocolo TCP

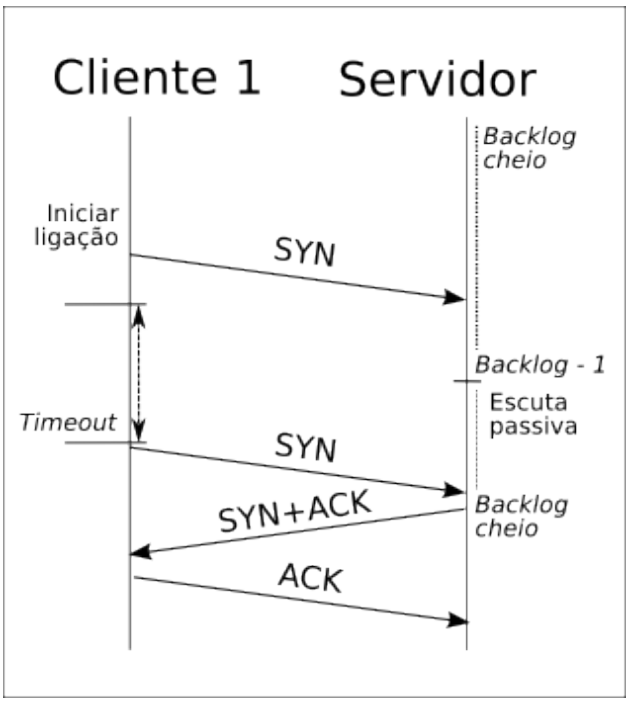

Figura 5.4: *Handshake do procolo TCP, com as respectivas flags que são trocadas entre cliente e servidor no processo*

Fonte: https://commons.wikimedia.org

O protocolo de transmissão de dados chamado de TCP (25), tem como sua principal vantagem a garantia do recebimento dos dados enviados devido a forma como é feita a entrega destes pacotes. Garantindo que o cliente (remetente) tenha a confirmação do recebimento da informação pelo lado do servidor (destinatário), assim caso o destinatário não o tenha recebido o cliente é avisado.
Para se ter essa garantia o cabeçalho deste pacote enviado é maior, possuindo todas as *flags* de identificação solicitadas para o funcionamento deste protocolo. Devido a isso seu tamanho em bytes é aumentado e obtemos um *overhead* maior na conexão devido ao *handshake* (Figura 5.4) mais longo.
Esse pacote foi o escolhido para o envio dos dados amostrados pelo Lidaro pela garantia que nenhuma amostra será perdida ou enviada indevidamente devido ao protocolo. E como o sensor possui uma velocidade de amostragem inferior ao *overhead* do protocolo é justificado seu uso para isso.

Protocolo UDP

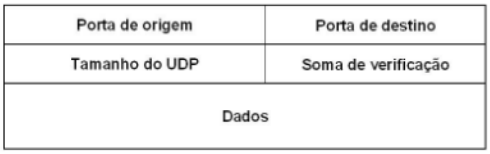

Figura 5.5: *Representação alto nível de um pacote UDP, que em comparação ao TCP é muito mais simplificado*

O protocolo de comunicação UDP é um dos mais simples implementados para a camada de transporte da rede para a troca de mensagens, mas devido a sua simplicidade perde-se garantias, como a da entrega da mensagem ou que a mensagem não chegue corrompida. Dessa forma é usualmente recomendado para o envio de pequenas mensagens. Como os comandos trocados entre o Lidaro e o computador servidor para configuração são curtas e esporádicas então este protocolo é ideal.

Lista de Comandos

Os comandos são categorizados como mensagens enviadas do Lidaro para o servidor e do servidor para o Lidaro, na tabela 5.1 temos a descrição de quais são estes:

Tabela 5.1: *Setlist de comandos*

Função	Sintaxe	Origem	Descrição
Mensagens de valores amostrados	"<Ang_azimutal><Ang_altitude><distancia>"	*Lidaro*	Sendo os valores Ang_azimutal o ângulo em graus do posicionamento do eixo do azimute, Ang_altitude o ângulo em graus do posicionamento do eixo de altitude, distância a distância amostrada pelo sensor ToF.
Mensagem de valor de calibração	"c <Offset_angX><Offset_angY><Offset_angZ>"	*Lidaro*	Sendo **Offset_angX**, **Offset_angY**, **Offset_angZ** os ângulos correspondentes a inclinação do Lidaro em relação a horizontal, vertical e em relação a seu eixo de rotação vertical, ou como conheciados angulos de *Yaw*, *Pitch* e *Roll*.
Mensagens On/Off	"o00<value>"	*Servidor*	Sendo **value** igual a 1 para ligado e 0 para desligado
Mensagem de velocidade do eixo de altitudes	"v<value>"	*Servidor*	Sendo *value* qualquer um dos seguintes valores: 0 (parado), 1(lento), 2(médio), 3(rápido), 4(muito rápido), 5(insano).
Mensagem de velocidade do eixo azimutal	"a<value>"	*Servidor*	Sendo *value* qualquer valor entre 0 e 5, seguindo a mesma escala de velocidade definida para o eixo da altitude

Conectividade

Figura 5.6: *Portal de Conexão, da esquerda para a direita, imagem inicial do portal e imagem de configuração do WiFi e do endereço do computador que executará a interface*

Todos os dados enviados pelo Lidaro são por meio da rede WiFi, dessa forma para se fazer a conexão com a rede ao executar a sua inicialização faz-se o uso da biblioteca `wifimanager`. Por ela se verifica todas as redes WiFi disponíveis e, caso ele já não tenha sido pré configurado para uma das redes existentes, o Lidaro se inicia no modo estação, sendo ele o provedor de uma rede WiFi cujo ssid é "LIDARO_CONF". Ao se conectar nessa rede um portal de configuração é aberto automaticamente, ou do contrário também é possível acessá-lo uma vez conectado na respectiva rede pelo IP "192.168.1.4". Uma vez

na página de configuração pode-se escolher configurar a rede manualmente pela opção sem scan e automaticamente scaneando as redes locais e selecionando uma delas. Outro campo que também deve ser preenchido é o "ip server"que identifica o IP do computador que estará rodando o software com a interface para o Lidaro.

5.2 Software

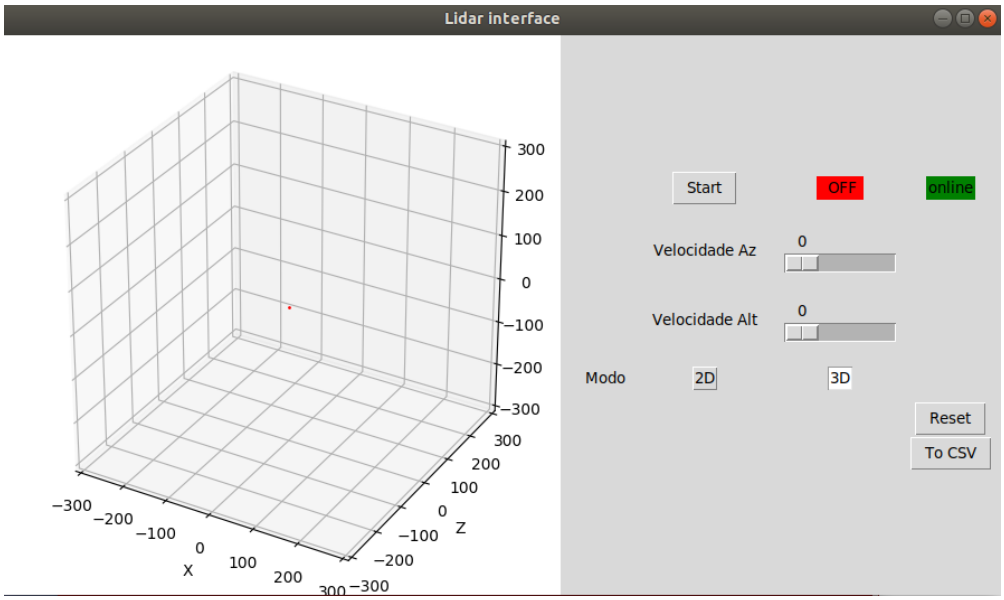

Figura 5.7: *Interface gráfica de operação do Lidaro*

Para executar medições usando o Lidaro pode-se fazer o uso de conexões via socket diretamente na porta 8091 (UDP) para enviar os comandos na sintaxe já descrita e receber na porta 8090 os dados das medições também na forma padronizada descrita na tabela 5.1. Alternativamente a esse modo, foi implementada para fazer uma abstração a esse tipo de conexão um software que possui uma interface gráfica, para o controle do Lidaro e visualização dos pontos recebidos em uma janela de plot.

5.2.1 Visão geral sobre o Software

O software para a comunicação com o Lidaro foi implementado utilizando Python3 como linguagem e Tk como interface gráfica.
Neste software (Figura 5.7) temos utilidades básicas para a operação do Lidaro como comandos para liga-lo e desliga-lo, fazer a seleção de velocidade para seus eixos, selecionar o modo de scan que se deseja realizar, se 2D ou 3D, disposição de visualização dos pontos plotados e exportação dos pontos no formato CSV (*Comma Separated Values*), todas essas operações executadas nos respectivos botões da tela de comando. Também existe uma *label* de verificação do estado de conexão do Lidaro com o computador que indica se ele esta online ou offline.
O motor por trás da interface pode ser definido essencialmente por 2 threads principais, a que plota os dados na tela e a responsável por administrar a comunicação recebendo e enviando os dados.
A thread de comunicação de dados é a que faz a gestão das portas de TCP e UDP usadas pelo Lidaro, assim como também assume o papel de aplicar as conversões do sistema de

coordenadas esférico para cartesiano pelas seguintes equações seguindo o padrão americano:

$$\begin{cases} x = r\ cos(\theta)\ sen(\phi) \\ y = r\ sen(\theta)\ sen(\phi) \\ z = r\ cos(\phi) \end{cases}$$

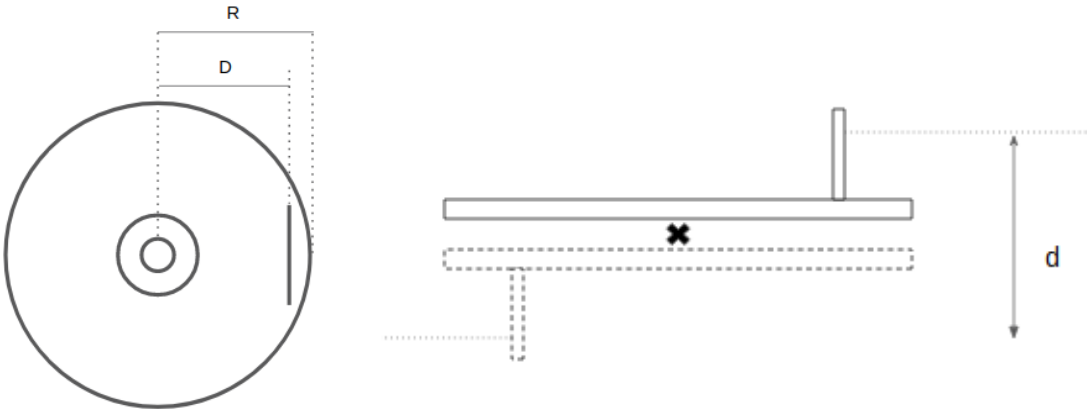

Figura 5.8: *Ao fazer a consideração da distância em relação ao ponto de 0 do Lidaro tem que se levar em consideração a distância do eixo de rotação à posição do sensor, assim como a sua movimentação orbital, que leva o laser a se deslocar em d de sua posição inicial a final*

Onde temos ϕ, θ e r como as coordenadas esféricas e x, y e z sendo as coordenadas cartesianas.
A conversão entre o sistemas é essencial para conseguirmos a visualização dos dados igual temos no espaço real.
Além desta conversão de coordenadas também é aplicada uma correção quanto ao posicionamento do sensor em relação ao raio do rotor (Figura 5.8).
O Lidaro é provido de um sensor acelerômetro a partir do qual se obtém os dados de inclinação da superfície sobre o qual ele esta apoiado. Essa informação de inclinação é dada em termos de inclinação dos eixos X, Y e Z em graus, definidos como $\hat{x}$, $\hat{y}$ e $\hat{z}$.
Para aplicar a correção da inclinação nos pontos amostrados é aplicada a matriz de rotação em 3D definida como:

$$\begin{bmatrix} cos(\hat{y}) * cos(\hat{z}) & cos(\hat{x}) * sin(\hat{z}) + sin(\hat{x}) * sin(\hat{y}) * cos(\hat{z}) & sin(\hat{x}) * sin(\hat{z}) - cos(\hat{x}) * sin(\hat{y}) * cos(\hat{z}) \\ -cos(\hat{y}) * sin(\hat{z}) & cos(\hat{x}) * cos(\hat{z}) - sin(\hat{x}) * sin(\hat{y}) * sin(\hat{z}) & sin(\hat{x}) * cos(\hat{z}) + cos(\hat{x}) * sin(y) * sin(\hat{z}) \\ sin(\hat{y}) & -sin(\hat{x}) * cos(\hat{y}) & cos(\hat{x}) * cos(\hat{y}) \end{bmatrix}$$

Pelo software também se consegue definir o modo de operação do Lidar, se deseja fazer um scan bidimensional, apenas no plano XY, ou se tridimensional. Para fazer a seleção do modo tem-se disponível na interface um seletor para isso. A diferença elementar entre os modos de operação está entre quais dos eixos são ativados.
Para o modo de operação tridimensional ambos os eixos de rotação, o azimutal e o de altitude são ativados e para o modo bidimensional apenas o eixo azimutal rotaciona, sendo para esse motivo um ajuste manual do posicionamento do eixo de altitude para selecionar em qual ângulo o eixo da altitude será apontado.
Os dados de ambos os scans podem ser exportados para um arquivo CSV se utilizando do botão da interface, o arquivo de saída vai possuir 4 colunas sendo elas no formato:

```
<Número da amostra> <X> <Y> <Z>
```

Ou seja, com os valores já convertidos para as coordenadas cartesianas e com a correção dos ângulos pelo acelerômetro.

Capítulo 6

Testes

6.1 Teste estrutural

A estrutura do Lidaro foi construída montada sobre a fonte de alimentação (Figura 6.1) com a finalidade de torna-lo mais compacto, mas também sendo possível o remover para que seja alimentado por meio da bateria como já foi afirmado anteriormente:

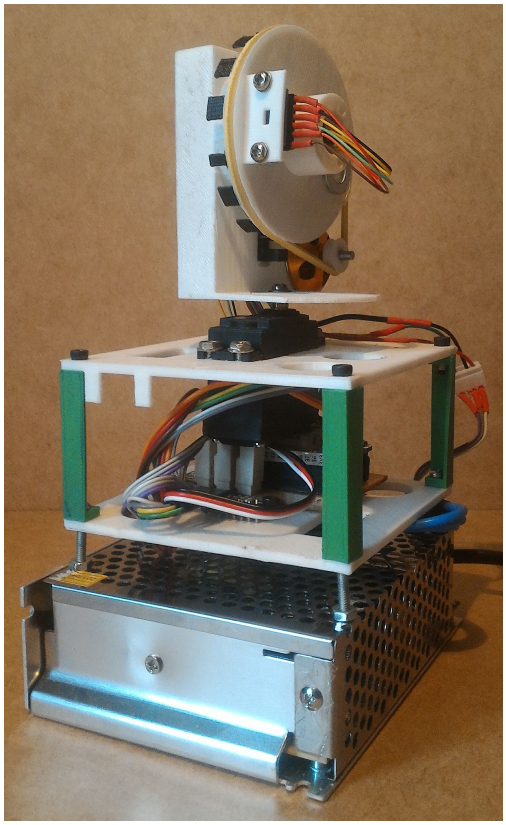

Figura 6.1: *Estrutura final do Lidaro*

Na tabela 6.1 temos as velocidades mínimas e máximas para a execução do scan para cada um dos eixos, essas velocidades relacionadas com a taxa de amostragem do sensor igual a 50Hz nos darão a relação de o quão esparsa será nossa nuvem de pontos por rotação de cada eixo.

Tabela 6.1: *Velocidade de rotação máxima e mínima de cada eixo*

	Min(graus/segundo)	**Max(graus/segundo)**
Velocidade Eixo Azimutal	20	80
Velocidade Eixo Altitude	720	1260

Sendo o eixo de altitude o mais rápido, neste é onde temos a maior limitação. Em sua rotação mais lenta conseguimos uma resolução de 1 amostra a cada 18 graus. Considerando o valor correspondente a tangente de 18 graus, a medida que o obstáculo detectado é mais afastado do Lidaro mais esparsa é a nuvem de pontos.

6.1.1 Cenário de testes globais

Para avaliar as capacidades performáticas do Lidaro na virtualização de ambientes foram criados quatro tipos de cenários, dois para validar distâncias superiores a 500 mm e dois para validar distâncias inferiores a essa.

Inferiores a 500 mm

Figura 6.2: *Setup do cenário de medição da caixa*

Foram selecionados dois objetos de formas variadas para se fazer a virtualização do interior dos mesmo, as formas escolhidas foram a circular e a retangular, para isso foi utilizado um balde e uma caixa de poliestireno como cenários. A caixa de poliestireno sendo um paralelepípedo com dimensões de 215x335x265 mm, a qual possui uma das faces de 215x335 mm aberta.
O Lidaro foi posicionado no interior desta caixa com ela na posição vertical (Figura 6.2) a aproximadamente 150mm da parede do fundo e equidistante as paredes laterais. Este teste teve por objetivo avaliar a precisão com o qual as distâncias seriam lidas sendo elas pequenas, em um objeto relativamente não reflexivo, tal como é o poliestireno, e a precisão que ângulos retos são captados. Foi executado neste ambiente o scan no modo bidimensional e tridimensional, e os resultados foram muito próximos do esperado tanto por questões de aproximação das medidas quanto na resolução do objeto.
No modo bidimensional (Figura 6.3), a varredura foi executada na maior velocidade disponível para o para o eixo azimutal e com o sensor manualmente posicionado a 90 graus com vertical, voltado para o interior da caixa.
O período de execução desta varredura foi de 14 segundos, tempo o suficiente para o sensor percorrer 180 graus, ou seja, todo o seu percurso disponível.
Podemos perceber que a correção da posição do sensor sobre o disco do rotor é funcional, e que a resolução dos valores amostrados nos permitem visualizar o perfil da caixa com apenas algumas aberrações nos ângulos e nos valores mais extremos.

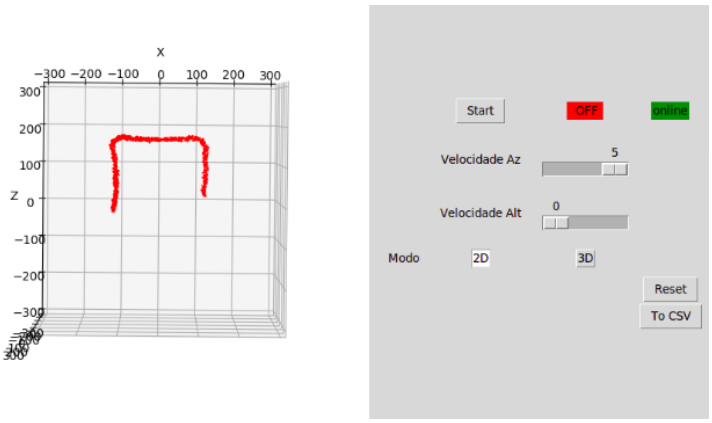

Figura 6.3: *Scan bidimensional da caixa*

No scan tridimensional (Figura 6.4), feito a partir da mesma posição obtivemos uma nuvem de pontos mais esparsa, como era de se esperar, no entanto que ainda assim nos promove a visualização da caixa. O scan executado foi realizado a velocidade 2 para o eixo azimutal e velocidade 1 para o eixo de altitude, levando um tempo de aproximadamente 1 minuto para se completar, nos fornecendo uma boa resolução.
Podemos perceber uma maior concentração dos pontos nas proximidades de onde o Lidaro foi posicionado, assim como a detecção de sua própria estrutura na posição onde ele foi colocado. Na vista lateral da imagem pode-se perceber a ausência da parede faltante assim como um espalhamento de pontos ruidosos correspondente a abertura.

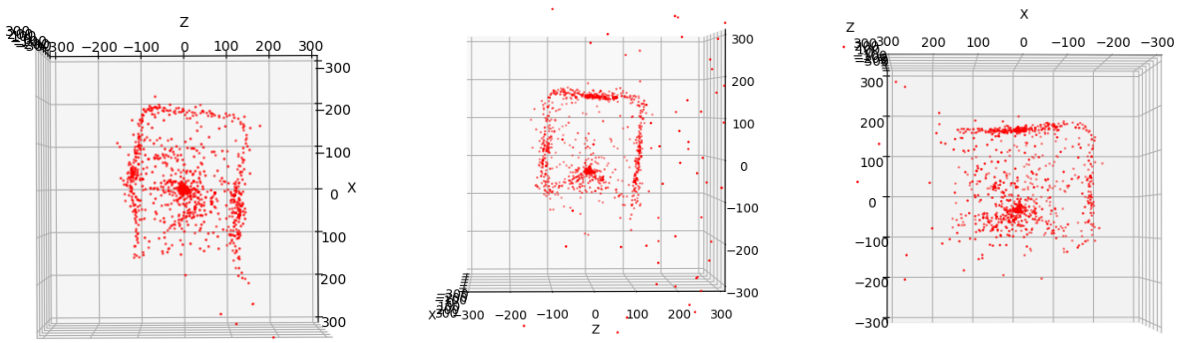

Figura 6.4: *Resultado do scan 3D da caixa, da esquerda par a direita, visto de cima, frente e lado*

Figura 6.5: *Setup do cenário de medição do balde, o Lidaro está localizado dentro do balde*

Como segunda prova para pequenos objetos, foi feita a varredura também no interior de um balde, o objetivo desse teste seria para validar a detecção de arcos e o comportamento do Lidaro frente a objetos pouco mais reflexivos, como por exemplo o plástico. O balde utilizado tem por formato aproximado a um cilindro com base de 330mm com altura de 330mm, possui uma constricção do diâmetro para 220 mm a 280 mm de altura (Figura 6.5). O scan para este objeto foi realizado a velocidade 5 para o eixo azimutal e 1 para o eixo de altitudes, levando ao tempo de varredura de aproximadamente 14 segundos. A escolha dessas velocidades foi buscando observar se haveriam deformações axiais na forma virtualizada.

No resultado (Figura 6.6) podemos perceber que o diâmetro se manteve constante, sem muitas deformações mesmo com possíveis reflexões, tendo obtido resolução o suficiente para detectar a constricção do balde. É possível mais uma vez se perceber o posicionamento do Lidaro na imagem pela região com maior densidade de pontos.

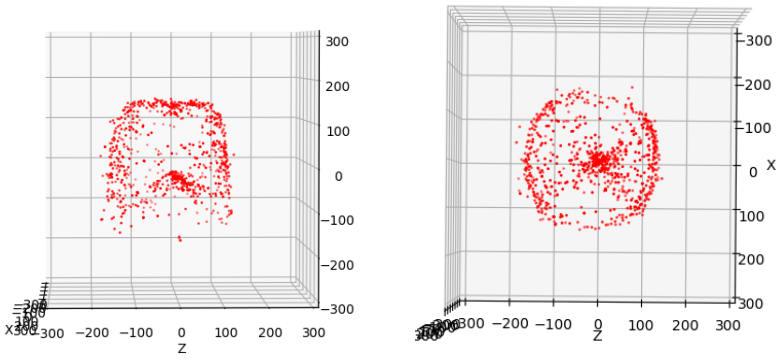

Figura 6.6: *Representação do balde virtualizado, da esquerda para a direita, vista lateral e vista superior*

Superiores a 500 mm

Para se realizar os testes a longa distância, consideradas aqui superiores a 500 mm, foram criados dois cenários com diferentes aspectos para se avaliar a resolução do sensor para este caso.

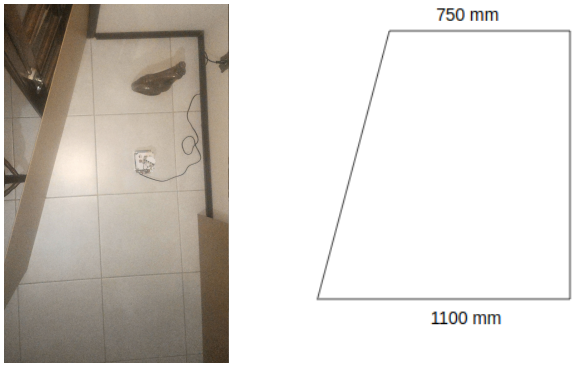

Figura 6.7: *Cenário em forma de trapézio, com suas dimensões, criado para executar os testes*

No primeiro teste foi montado um pequeno cenário na forma de um trapézio (Figura 6.7) sendo que a sua base maior foi retirada, deixando o cenário aberto em um de seus lados para se observar o comportamento nesta condição. Também foi posicionada uma estátua vazada a uma certa distância do Lidaro para avaliar se a mesma seria detectada.
Nesta configuração foi feita uma análise comparativa entre os resultados da virtualização variando-se a velocidade de rotação dos eixos, com o Lidaro foi posicionado a aproximadamente 1000 mm da base menor e feito um scan bidimensional a velocidade de rotação do eixo azimutal em 1 e 5 (Figura 6.8). Em ambos os scans é possível se perceber que a resolução foi suficiente para detectar a estátua e também se é perceptível que com o aumento da velocidade houve uma maior esparcidade nos pontos.

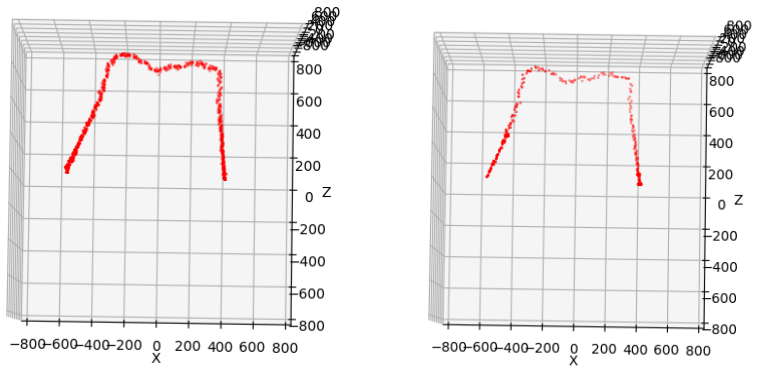

Figura 6.8: *Da esquerda para a direita, mapas virtualizados a velocidade 1 e 5*

Para as varreduras comparativas tridimensionais foram feitos 3 testes: um com velocidade de ambos os eixos em 1 (Figura 6.9), outro com velocidade do eixo azimutal em 5 e o de altitude em 1 (Figura 6.10) e por último com todas as velocidade em 5 (Figura 6.11). O tempo que cada um dos scans levaram foram de 1 minuto, 18 segundos e 19 segundos, respectivamente.
Percebe-se claramente em todos as virtualizações a ausência da parede retirada do cenário e que a medida que a velocidade de scan aumenta em ambos os eixos a detecção da estátua foi ficando prejudicada, mas ainda assim com alguns pontos dela ainda capturados.

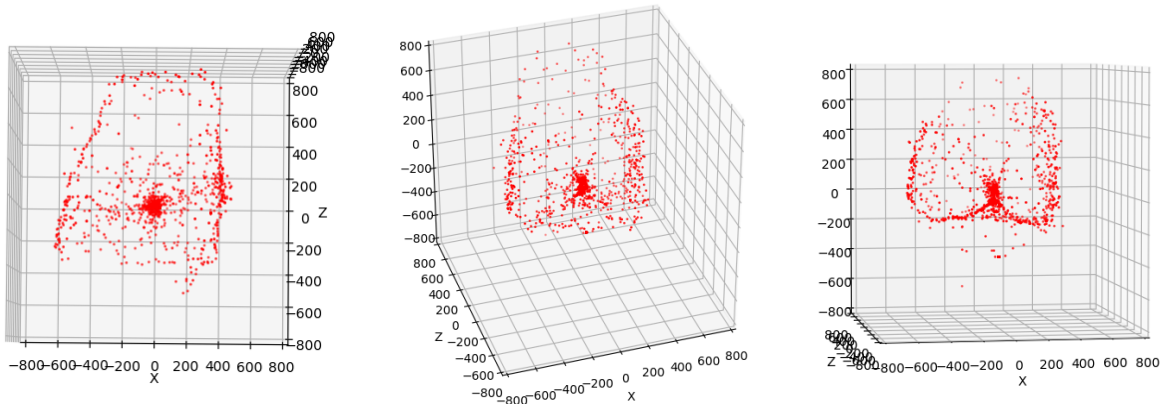

Figura 6.9: *Virtualização 3D feita com velocidade de ambos os eixos a velocidade 1*

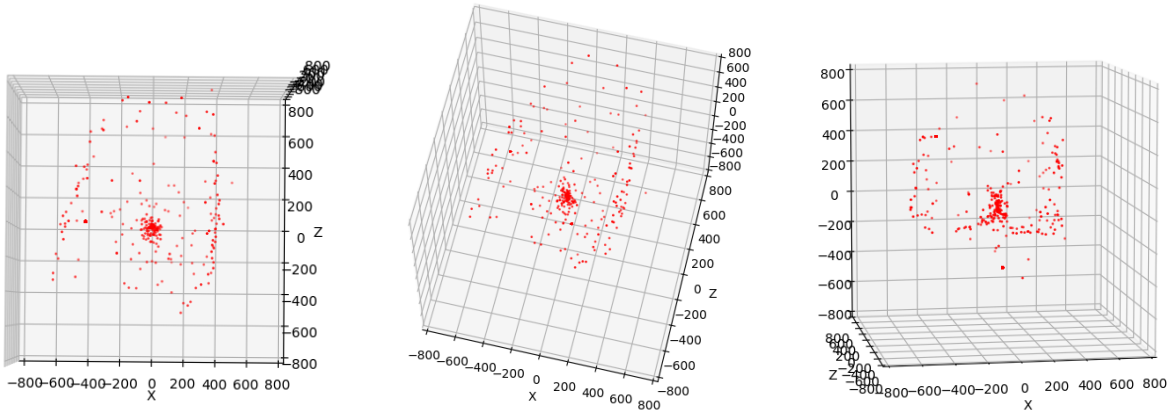

Figura 6.10: *Virtualização 3D feita com velocidade azimutal 5 e velocidade do eixo de altitude 1*

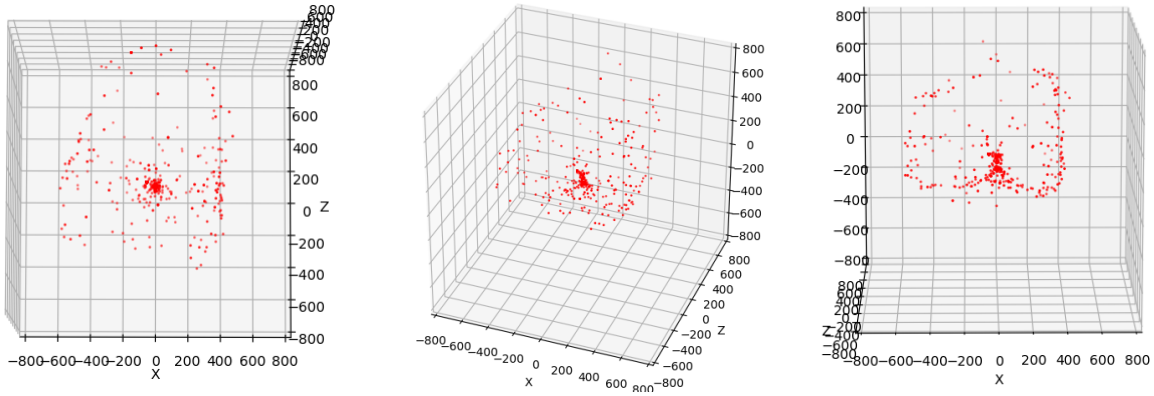

Figura 6.11: *Virtualização 3D feita com velocidade azimutal 5 e velocidade do eixo de altitude 5*

Como segundo teste de longa distância foi criado um cenário (Figura 6.12) onde haviam 3 obstáculos com propriedades interessantes, o primeiro deles a mesma caixa de poliestireno já citada, o balde também já citado e um banco de madeira com 4 pés finos. A ideia de usar esses obstáculos é para se ter a referência do comportamento do sensor frente a objetos circulares (balde), retangulares (caixa) e finos (pés do banco).

Figura 6.12: *Cenário com os obstáculos*

Como resultado podemos observar que o Lidaro teve resolução suficiente para detectar todos os obstáculos, com ênfase para o detalhe do decréscimo da precisão das medidas feitas com o aumento da distância, perceptível no scan 2D. Comparativamente, tanto a virtualização 2D e 3D nos forneceram dados bastante consistentes em relação ao posicionamento dos obstáculos.

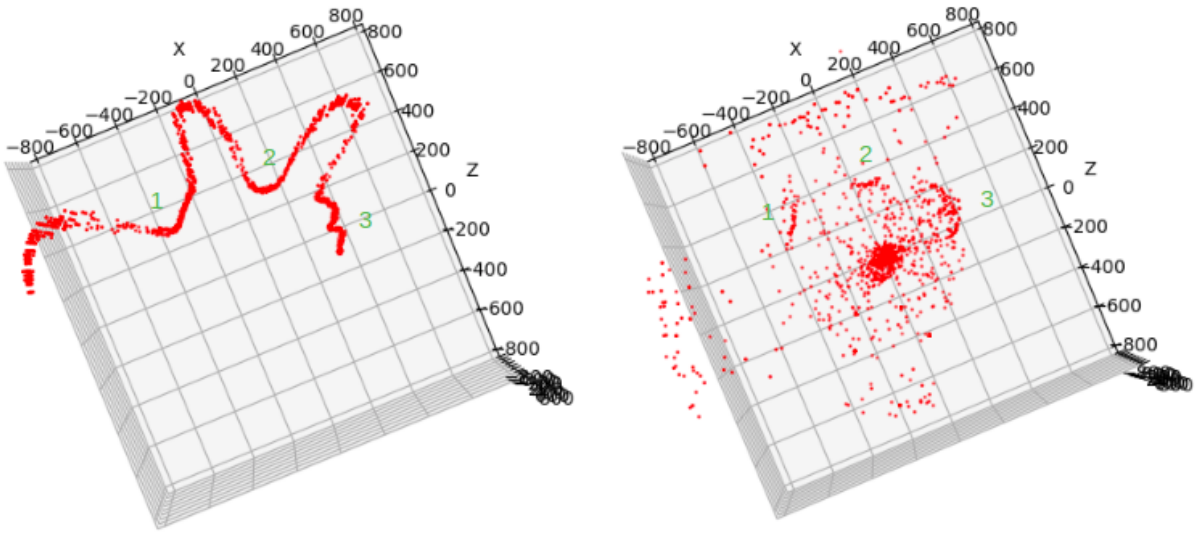

Figura 6.13: *Virtualização do ambiente de obstáculos, 2D e 3D vista de cima, respectivamente, da esquerda para a direita. Estão enumerados os obstáculos como 1 - caixa, 2 - balde, 3 - banco*

Capítulo 7

Conclusões

7.0.1 Adversidades encontradas

Durante a implementação deste projeto várias adversidades foram enfrentadas, as quais levaram a diversas modificações tanto estruturais e de hardware quanto de software/firmware. Algumas das principais serão listadas aqui.

- Utilização de um microcontrolador atmega32U4(26): Foi cogitado inicialmente a implementação do Lidaro utilizando um Arduino pro micro, o qual possui um microcontrolador atmega32U4. Porém devido a sua baixa velocidade de processamento, pela ausência de dois núcleos quando comparado com o ESP32, e a ausência de placas no mercado que possuíssem WiFi embutido, o que exigiria um módulo extra apenas para esta função, o tornaram a solução não ideal para a implementação, tanto do ponto de vista de rentabilidade quanto de praticidade.
  Todo o desenvolvimento inicial do projeto foi feito sobre esse microcontrolador, o teste de sensores e de controle dos motores, pois o código é compatível.

- Motor brushed ao invés de brushless: Testes foram feitos com um motor do tipo brushed por ser uma ótima opção do ponto de vista de facilidade de obtenção desse tipo de motores, no entanto devido ao torque necessário para girar o rotor a dada velocidade que é necessária. O mesmo não possuía potência o suficiente, sendo então exigida uma corrente muito elevada sobre suas bobinas internas fazendo-o esquentar demasiadamente. Para a correção desses problemas, considerando-se a relação de custo benefício, foi escolhido então o motor brushless A2212.

- Interrupções de FPU: Durante a implementação das rotinas de interrupção correspondentes a precisão dada ao encoder, que originalmente foram testadas com sucesso no microcontrolador atmega32U4, ao serem portadas para o ESP32 causavam a reinicialização do microcontrolador.
  Após observar a documentação do ESP32, isso ocorria devido ao uso do tipo `float` dentro desta rotina. Por natureza o tipo `float` é processado na unidade de ponto flutuante do ESP32, que possui um hardware dedicado para isso *in silico* que torna estas operações mais rápidas, no entanto que exigem uma interrupção para que as mesmas sejam executadas. Assim, quando uma interrupção de ponto flutuante era chamada dentro de outra interrupção esta operação não é suportada pelo ESP32. Portanto para que esta mesma rotina fosse compatível com este microcontrolador foi necessário o uso do tipo `double` que é processada por software.

- Servo 9g(27) ao invés de mg995: Por uma relação de custo inicialmente considerou-se usar o servo-motor do modelo 9g que possui uma menor relação de torque que o

mg995 para movimentar a torre do Lidaro no entanto ela apresentava um erro demasiadamente grande na leitura de seu posicionamento. Isso era causado devido ao *momentum* necessário para girar a torre por sua massa e pela resistência apresentada pelos fios de suas conexões. Assim, portanto, o servo mg995 tornou-se a opção mais viável.

7.0.2 Melhoramentos Futuros:

Diversos melhoramentos podem ainda ser feitos no Lidaro e em sua interface de operação, além da criação de filtros para o processamento dos dados de saída. Estruturalmente ocorre ainda reverberação acústica dentro do corpo da torre, o que faz com que seu barulho em funcionamento de scanner 3D seja elevado. Como o motor brushless possui um intervalo de frequências sonoras bem definida de funcionamento pode-se ser feito um estudo acústico para que dentro da cavidade da torre a amplificação desta frequência seja diminuída, assim como otimizações estruturais para torná-lo mais robusto. Uma aplicação que começou ser desenvolvida para o Lidaro é um pequeno veículo autônomo (Figura 7.1) sobre o qual ele poderia ser montado para se auto dirigir dentro de algum ambiente fazendo o que se é conhecido por SLAM (*Simultaneous Localization and Mapping*), mas foi descontinuada por fugir ao escopo deste trabalho.

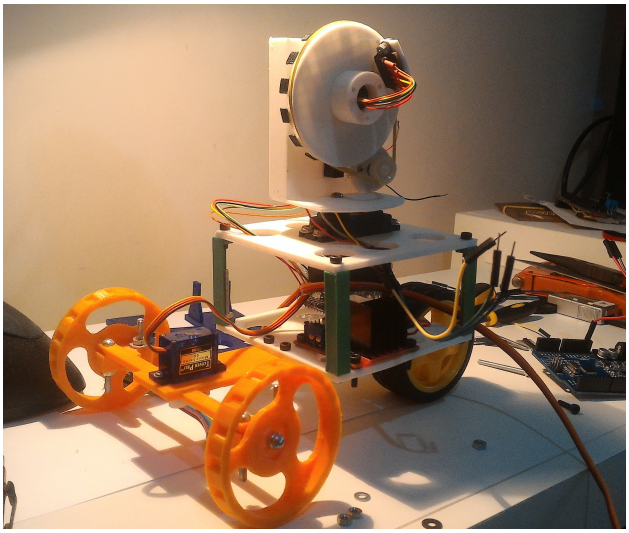

Figura 7.1: *Protótipo do carro iniciado onde o Lidaro seria para fazer SLAM*

Os dados provenientes do Lidar possuem ruídos e uma solução para isso seria a implementação de filtros para estes, tal como pode-se ser utilizado o filtro estatístico para *outliers* (28), no qual baseado na nuvem de pontos, observando a vizinhança de um ponto pode-se dizer se ele é considerado pertencente ou não ao conjunto. Assim esse filtro também podendo ser adicionado a interface ou para fazer um tratamento a posteriori.

7.0.3 Repositório

Todos os arquivos com fimware, software da interface, arquivos CAD para a impressão 3D e construção da placa de circuito podem ser encontrados no repositório do Github do projeto.

Apêndice A

Cianofícea: A Impressora 3D DIY de baixo custo

Figura A.1: *Cianofícea, a impressora 3D*

O projeto deste Lidar foi construído inteiramente utilizando ferramentas de fabricação digital e prototipagem rápida, por isso se entende principalmente utilizando impressão 3D. A impressora 3D utilizada (Figura A.1), do tipo cartesiana, foi construída com a finalidade de auxiliar a prototipagem deste projeto. Toda a estrutura, desde os *frames* de sustentação inferiores, corpo e rotor do Lidaro foram modelados e impressos em 3D. O projeto de construção da impressora foi baseado em um projeto de hardware e software abertos de um modelo chamado de Graber I3, o qual teve algumas adaptações do modelo original com finalidades de otimização como serão citadas a seguir.
O projeto foi construído focado no baixo custo e peças encontradas com facilidade em território brasileiro.
Toda a estrutura física da impressora foi construída em madeira MDF (*Medium-Density*

Fiberboard) de 6.0mm de espessura, barras rígidas, de ferro inox comumente utilizadas em máquinas CNCs (*Computer Numerical Control*) gerais e fusos estriados utilizados em para construção civil.
A parte mais crítica da construção com relação ao baixo custo são os motores do tipo Nema 23, que são do tipo nema 17(29), estes são encontrados por preços elevados em território nacional, desta forma a opção foi importá-los, assim como toda a parte eletrônica de fornecedores da china.
Como microcontrolador principal, foi utilizado um Arduino Mega com o *firmware* Marlin que foi modificado para as restrições desta impressora e modificações subsequentes. Como driver de motores foi utilizada uma ramps 1.4 como indicada no projeto original. Os elementos aquecedores utilizados foram o de uma extrusora para a impressora *open-source* do tipo Prussa assim como o modelo de mesa aquecida escolhida.
Como decisão de projeto, para facilitar a construção, a mesa aquecida foi definida como fixa, sem parafusos para a regulagem de altura, mas garantindo um bom alinhamento horizontal inicial, pequenas imperfeições são corrigidas com a implementação do BAL (*Bed Auto Leveling*).
As otimizações feitas no projeto inicial foram focadas em produtividade e precisão da impressora como a remoção do motor de injeção de filamento na extrusora, que originalmente é colocado sobre o eixo X, e o colocando na parte traseira, utilizando um método de injeção do filamento no bico aquecido denominado *Bowden*, nesta forma o filamento é conduzido até o bico por meio de um tubo de PTFE (politetrafluoretileno) diminuindo a massa e, por consequência, a inércia da cabeça de impressão melhorando assim o acabamento da peça e aumentando a velocidade de movimentação deste eixo.
O nivelamento automático também foi habilitado de forma que não fosse mais necessária a calibração e alinhamento da mesa da impressora, para isso um sensor do tipo indutivo foi colocado na cabeça de impressão o qual faz a amostragem de pontos ao longo da mesa, gerando assim o plano com sua inclinação correspondente e o compensando por meio de software. Mais informações da construção podem ser encontradas na página de desenvolvimento da impressora.

Referências Bibliográficas

[1] Lidar360. http://grauonline.de/wordpress/?page_id=1233. Accessed: 2018-09-30. 11

[2] L Chilson. The difference between abs and pla for 3d printing. 2013. 12

[3] ST Semiconductors. *World's smallest Time-of-Flight ranging and gesture detection sensor*. 15, 16

[4] TT Electronics. *Slotted Optical Switch*. 15

[5] Texas Instruments. *LMx93-N, LM2903-N Low-Power, Low-Offset Voltage, Dual Comparators*. 15

[6] Invensense. *MPU-6000 and MPU-6050 Product Specification Revision 3.4*. 15

[7] Espressif. *ESP32 Series Datasheet*. 15

[8] TowerPro. *MG995 High SpeedMetal Gear Dual Ball Bearing Servo*. 15

[9] Rhydolabz. *A2212/13T TECHNICAL DATA*. 15

[10] ST Semiconductors. *A new generation, long distance ranging Time-of-Flight sensor based on ST's FlightSense^{TM} technology*. 16

[11] ST Semiconductors. *Proximity and ambient light sensing (ALS) module*. 16

[12] Benewake. *TF02 Specification*. 16

[13] Garmin. *Operation Manual and Technical Specifications*. 16

[14] Richard Walker Robert K. Henderson Edoardo Charbon, Matt Fishburn and Cristiano Niclass. Spad-based sensors. 2013. 17

[15] Inc Princeton Optronics. Vertical-cavity surface-emitting laser technology. none. 17

[16] Brushless dc motor control made easy. http://ww1.microchip.com/downloads/en/appnotes/00857a.pdf. Accessed: 2018-10-06. 22

[17] Serial. https://www.arduino.cc/reference/en/language/functions/communication/serial. Accessed: 2018-10-06. 30

[18] Wire. https://www.arduino.cc/en/Reference/Wire. Accessed: 2018-10-06. 30

[19] Wifi. https://www.arduino.cc/en/Reference/WiFi. Accessed: 2018-10-06. 30

[20] Servo. https://www.arduino.cc/en/Reference/Servo. Accessed: 2018-10-06. 30

[21] Arduinoota. https://github.com/espressif/arduino-esp32/tree/master/libraries/ArduinoOTA. Accessed: 2018-10-06. 30

[22] vl53l0x. https://github.com/pololu/vl53l0x-arduino. Accessed: 2018-10-06. 30

[23] Mpu6050. https://github.com/jrowberg/i2cdevlib/tree/master/Arduino/MPU6050. Accessed: 2018-10-06. 30

[24] Siddharth Singh. Pulse-width modulated signal generation for control of servo motor on simplest microprocessor. 2014. 31

[25] Tarik Eltaeib. Tcp/ip protocol layering. 2015. 34

[26] atmega32u4. https://www.microchip.com/wwwproducts/en/ATmega32u4. Accessed: 2018-10-14. 47

[27] 9g. http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf. Accessed: 2018-10-14. 47

[28] Wei Jiang Hancong Liu, Sirish Shah. On-line outlier detection and data cleaning. *Computers and Chemical Engineering*, 2004. 48

[29] pbclinear. *Stepper Motor NEMA 17*. 50