UNIVERSIDADE FEDERAL DE SANTA MARIA
CENTRO DE TECNOLOGIA
CURSO DE ENGENHARIA DE COMPUTAÇÃO

Eugênio Piveta Pozzobon

DESENVOLVIMENTO DE UMA PLATAFORMA EQUATORIAL DE BAIXO CUSTO PARA ASTROFOTOGRAFIA

Santa Maria, RS
2022

Eugênio Piveta Pozzobon

DESENVOLVIMENTO DE UMA PLATAFORMA EQUATORIAL DE BAIXO CUSTO PARA ASTROFOTOGRAFIA

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia de Computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Engenheiro de Computação**.

ORIENTADOR: Prof. Rafael Concatto Beltrame

Santa Maria, RS
2022

Eugênio Piveta Pozzobon

DESENVOLVIMENTO DE UMA PLATAFORMA EQUATORIAL DE BAIXO CUSTO PARA ASTROFOTOGRAFIA

Trabalho de Conclusão de Curso apresentado ao Curso de Engenharia de Computação da Universidade Federal de Santa Maria (UFSM, RS), como requisito parcial para obtenção do grau de **Engenheiro de Computação**.

Aprovado em 23 de junho de 2022:

Rafael Concatto Beltrame, Dr. (UFSM)
(Presidente/Orientador)

Fernando Mariano Bayer, Dr. (UFSM)

Marcelo Serrano Zanetti, Dr. (UFSM)

Santa Maria, RS
2022

DEDICATÓRIA

Dedico este trabalho a todos os astrofotógrafos e aos apaixonados pela observação celeste.

AGRADECIMENTOS

Agradeço à minha Família por todo o suporte, carinho, cuidado e educação que me proporcionaram desde o meu nascimento.

Agradeço ao meu orientador, professor Rafael Beltrame, por toda a paciência, suporte, atenção e dedicação com o projeto. Por fornecer equipamento, custeio, ajudar com a captura e edição das imagens; sempre pensando no melhor lugar e momento para captar registros fantásticos. Foram 18 meses de muito aprendizado, conversa, empatia e cooperação.

Agradeço ao professor Fernando Bayer por realizar a impressão 3D de componentes e aos técnicos do NUPEDEE - UFSM pela fabricação de algumas peças projetadas neste trabalho.

Agradeço a todos, também, que colaboraram de alguma forma para a revisão de algum trecho desse documento, em especial ao meu orientador e às minhas amigas Tainá Lersh e Gabriela Bonugli.

Por fim, agradeço ao Engenheiro Ricardo Batista pela cooperação na avaliação e feedback para este trabalho, além do Astrofotógrafo Alberlan Barros por divulgar esse projeto.

"Se você tem uma maçã e eu tenho uma maçã e formos trocar essas maçãs, você e eu ainda teremos uma maçã cada um. Mas se você tem uma ideia e eu tenho uma ideia e formos trocar essas ideias, cada um de nós terá duas ideias."

(George Bernard Shaw)

RESUMO

DESENVOLVIMENTO DE UMA PLATAFORMA EQUATORIAL DE BAIXO CUSTO PARA ASTROFOTOGRAFIA

AUTOR: Eugênio Piveta Pozzobon
ORIENTADOR: Rafael Concatto Beltrame

Atualmente, a astrofotografia é realizada por grandes telescópios e também por um crescente número de astrofotógrafos amadores, os quais contam com diversas limitações. A principal reside no fato de que objetos celestes, de forma geral, demandam elevados tempos de exposição. Infelizmente, caso a câmera esteja imóvel sobre um tripé, o movimento de rotação da Terra não permite que os astros sejam expostos ao sensor por muito tempo, pois as imagens acabam borradas. Por esse motivo, é necessário o uso de uma ferramenta que movimente a câmera no sentido de rotação aparente do céu, compensando esse movimento e permitindo um registro fotográfico de alta qualidade. Para isso, existem inúmeras ferramentas comerciais para o rastreamento do céu, porém, todas elas são comercializadas no hemisfério Norte e com um custo expressivo para o brasileiro médio. Então, a fim de simplificar e reduzir o custo associado à astrofotografia, tornando-a mais acessível, objetiva-se neste trabalho desenvolver uma plataforma equatorial para astrofotografia que seja portátil, robusta, precisa, de fácil configuração e utilização, de peso e volume compatíveis com tripés fotográficos, e com custo inferior à soluções comerciais existentes. A plataforma tem como diferencial um aplicativo *mobile* que auxilia a configuração da plataforma para a obtenção de registros fotográficos. O sistema final passou por testes de rastreamento (erro de velocidade) e de vibração para garantir que a montagem estaria adequada para o uso, e também com testes em campo, onde foram obtidas fotografias da Via-Láctea.

Palavras-chave: Astrofotografia. Plataforma Equatorial. Sistema Eletrônico. Aplicativo Android. Bluetooth.

ABSTRACT

DESENVOLVIMENTO DE UMA PLATAFORMA EQUATORIAL DE BAIXO CUSTO PARA ASTROFOTOGRAFIA

AUTHOR: Eugênio Piveta Pozzobon
ADVISOR: Rafael Concatto Beltrame

Currently, astrophotography remains practiced by large telescopes and also by astrophotographers that have countless challenges. The main one relies on the fact that celestial bodies, in general, demand long exposure times. Unfortunately, despite the camera being on a tripod, the rotational movement of the Earth doesn't allow the stars to be exposed to the sensor for a long time, resulting in blurry images. For this reason, it is necessary to use a tool that moves the camera in the apparent rotation direction of the sky, compensating for this movement and obtaining a high-quality photographic record. For this, there are numerous commercial tools for tracking the sky. However, all of them are commercialized in the Northern hemisphere and with an exorbitant cost for the average Brazilian. So, in order to simplify and reduce the cost associated with astrophotography, making it accessible, the objective of this work was to develop an equatorial platform for astrophotography that is portable; robust; precise; easy to set up and use; of weight and volume compatible with photographic tripods; and at a lower cost than commercial solutions. The platform's differential is a mobile application that makes it easy to use these tools for the configuration of the platform to obtain photographic records. The final system has been passed into vibration tests to ensure that the assembly would be suitable for use. Also, the system got tested with long exposure photography.

Keywords: Astrophotography. Equatorial Mount. Electronic System. Android Application. Bluetooth.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇAO

Desde os primeiros registros datados, civilizações já observavam o céu e, desse modo, podiam contabilizar a passagem do tempo, identificar as estações do ano, os ciclos sazonais de chuvas e/ou secas, etc. Assim, por exemplo, era possível realizar um planejamento mais assertivo acerca da melhor época de plantio e colheita de diferentes culturas. Cada civilização tinha seu meio e técnica de observação. Por exemplo, em 4000 a.C., os povos da Mesopotâmia utilizavam zigurates para observar o céu noturno; já em 2500 a.C., foi construída a estrutura de pedras Stonehenge, na Inglaterra, para registrar os solstícios. Foi somente em 1609 d.C. que Galileu conseguiu aperfeiçoar e utilizar um telescópio refrator para observar os planetas e as estrelas pela primeira vez (HELERBROCK, c2021).

A astrofotografia consiste no emprego de técnicas fotográficas para registrar objetos astronômicos, como planetas, estrelas, galáxias, nebulosas, etc. A primeira astrofotografia é datada de 1840 e é um registro da Lua (JÚNIOR, 2016). Desse período em diante, a fotografia teve um papel muito importante na observação celeste ao possibilitar o registro do céu para análise científica. Essa técnica foi evoluindo gradualmente de forma que, por volta de 1960, já havia equipamentos que permitiam realizar registros mais concretos e eficientes, possibilitando fotos com mais definição e nitidez (SANTOS, 2010).

No fim do século XX, telescópios maiores e mais complexos, instalados na Terra ou em torno de sua órbita no espaço (como o telescópio espacial Hubble), ampliaram a capacidade da ciência em estudar fenômenos astronômicos ou mesmo a origem do próprio universo (SANTOS, 2010). Paralelamente, as ferramentas amadoras para astronomia (como telescópios de baixo custo), ou mesmo para astrofotografia (câmeras de custo mais acessível e equipamentos para rastreamento do céu) continuaram a ser desenvolvidas, de forma que um grande número de astrônomos e, especificamente, astrofotógrafos amadores continuassem exercendo seu *hobby* ou mesmo contribuindo à ciência.

Nesse sentido, o desenvolvimento de equipamentos acessíveis voltados ao público amador tem um papel fundamental para despertar o interesse pela ciência em cada vez mais pessoas. No entanto, os principais desafios de se fotografar galáxias, nebulosas, etc., é que esses corpos, de forma geral, demandam elevados tempos de exposição (JÚNIOR, 2016). Infelizmente, o movimento de rotação da Terra não permite que os astros sejam expostos ao sensor por muito tempo, pois as imagens ficariam borradas. Por esse motivo, é necessário o uso de uma ferramenta que movimente a câmera (acoplada ou não a um telescópio) no sentido de rotação aparente do céu, compensando o movimento e permitindo que o sensor da câmera possa receber luz por longos períodos de tempo (de minutos a horas) (JÚNIOR, 2016). Assim, consegue-se obter um registro fotográfico de alta qualidade e com um baixo custo computacional de pós-processamento.

Por isso, existem inúmeras ferramentas comerciais para o rastramento do céu vol-

tadas ao público amadador, como Nyx Track, SkyGuiderTM Pro, PolarieTM, entre outras (conforme Figura 1.1). Porém, todas elas são comercializadas no hemisfério norte e com um custo excessivo para brasileiro médio, considerando taxa de câmbio, taxas de importação e frete. Assim, de modo a contribuir à popularização da astrofotografia no Brasil, incentivando cada vez mais jovens a seguirem na carreira científica, propõe-se o desenvolvimento de uma plataforma equatorial de baixo custo para astrofotografia.

Figura 1.1 – Exemplo de solução comercial empregada em astrofotografias

Fonte: (CORPORATION, c2021).

2 DESENVOLVIMENTO TEÓRICO

2.1 ASTROFOTOGRAFIA

A astrofotografia é um ramo da astronomia e da fotografia que combina toda a ciência envolvida na documentação e registro de estrelas, constelações, planetas, meteoros, etc., com a arte da fotografia. Dentro da astrofotografia, existem variantes de estilo, como fotografias planetária, solar e céu profundo (ANDOLFATO, 2017), que se diferem pelo objeto alvo de registro. Além disso, existem diferenças entre a astrofotografia praticada profissionalmente por cientistas, em grandes telescópios, da praticada por amadores. Porém, ambas as atividades são importantes e se complementam.

As fotografias capturadas por telescópios profissionais possuem vantagens como uma grande ampliação, além de foco e definição, devido aos grandes espelhos que compõe suas montagens. Contudo, isso se torna um problema para a captura de imagens mais amplas, que são registradas, em sua maioria, por astrofotógrafos amadores (ANDOLFATO, 2017). E mesmo em nível considerado amador, é preciso de equipamentos os quais, no Brasil, custam um preço que acaba afastando uma boa parcela da população dessa prática.

2.1.1 Equipamentos

Além de uma câmera e uma lente, existem alguns equipamentos periféricos que são fundamentais para a prática da astrofotografia: tripé e disparador remoto (intervalômetro) para a câmera. O tripé é responsável por manter a câmera estável durante o registro das estrelas; o Disparador tem a função de operar a câmera remotamente para evitar que o operador faça a câmera tremer ao apertar algum botão e/ou também permitir a utilização do modo *Bulb* das DSLR. O modo *Bulb* consiste em permitir um controle total do tempo de exposição pelo operador (LOCHUN et al., 2015).

2.1.1.1 Câmeras

As câmeras digitais possuem sensores digitais de imagem que substituem o filme das máquinas fotográficas mais antigas. O sensor CMOS pode ter diversos tamanhos físicos e uma densidade de *pixels* por polegada (dpi) diversa entre os modelos. Normalmente, quanto maior for o sensor físico, mais qualidade terá a imagem final, mesmo mantendo-se

o número de mega-píxels do sensor (ALVES, 2018).

Existem diversos modelos de câmeras para fotografia: compacta, super-zoom, *mirrorless*, DLSR e por fim as câmeras em celulares. As câmeras *mirrorless* atualmente são equiparáveis às DSLR e ambas são as mais usadas para astrofotografia, possibilitam trocar lentes e podem ser utilizadas para capturar de céu noturno. Além disso, possibilitam uma série de configurações em modo manual que não são possíveis em câmeras semi-profissionais, compactas ou de super-zoom (LOCHUN et al., 2015).

2.1.1.2 Lentes

A lente é um equipamento acoplado no corpo da câmera, sendo responsável por focalizar a luz que invade o sensor. As lentes podem ser rígidas no corpo da câmera, no caso de modelos semi-profissionais e compactos; ou podem ser removíveis para o caso de modelos profissionais. Nesse último caso, as lentes removíveis são itens que podem ser obtidos por escolha do fotógrafo e existe uma variedade de modelos.

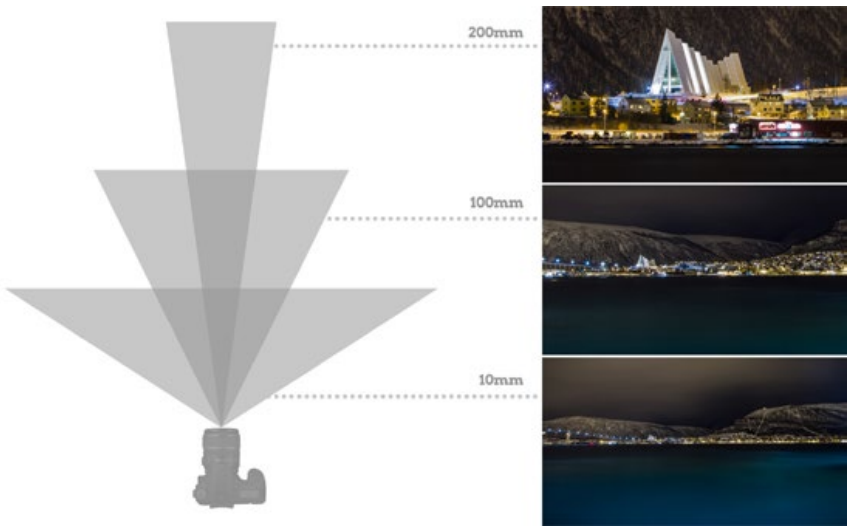
Esses modelos podem ser lentes fixas ou *zoom*. O primeiro é um modelo de lente que possui uma distância focal fixa. Já as lentes *zoom* permitem uma variação na distância focal, o que acaba gerando o *zoom* óptico (REGINA, 2013).

2.1.1.2.1 Distância Focal

A distância focal de uma lente é um fator mensurado em milímetros e é o que determina seu ângulo de abertura de visão. Quanto maior for a distância focal, mais fechado será o ângulo de visão, gerando um zoom. Do contrário, quanto menor for a distância focal, maior será o ângulo de visão e consequentemente menor será o zoom da lente. A Figura 2.1 ilustra essa relação da distância focal (REGINA, 2013).

Em um contexto de astrofotografia, lentes mais abertas são úteis para capturar a Via-Láctea (Figura 2.2), pois a via láctea cobre o céu completamente, e somente com lentes abertas é possível encaixar toda a galáxia. Para fotografias de constelações, nebulosas e planetas distantes da Terra, é necessário uma lente mais fechada, que possibilite o enquadramento com o zoom necessário, conforme o tamanho do astro a ser fotografado. (Figura 2.3)

Figura 2.1 – Efeito de zoom gerado pela variação da distância focal



Fonte: (REGINA, 2013).

Figura 2.2 – Fotografia da Via Láctea com lente *zoom* em configuração de 4mm

Fonte: Autor.

Figura 2.3 – Fotografia de Júpiter e as Luas de Galileu com lente *zoom* em configuração de 205mm

Fonte: Autor.

2.1.2 Exposição

A exposição de uma imagem se refere à quantidade de luz captada pelo sensor da câmera. Uma imagem muito clara é considerada superexposta, um caso onde o sensor recebeu muita luz. Ao contrário, uma imagem subexposta é uma fotografia escura que recebeu pouca luz. Existem 3 parâmetros configuráveis em uma câmera profissional, que são determinantes para a exposição e também para a qualidade da foto final: tempo de exposição, abertura da lente e sensibilidade do sensor. (VIERO, 2018). De forma geral, conseguir a exposição ideal é o principal desafio da astrofotografia de céu profundo (ANDOLFATO, 2017).

2.1.2.1 Tempo de Exposição

Para captar uma imagem, a câmera possui um obturador, que controla a passagem de luz em direção ao sensor interno que capta a imagem. Uma fotografia de longa exposição significa que a câmera permaneceu com obturador aberto, permitindo a passagem de luz por um longo intervalo de tempo (LOCHUN et al., 2015). Porém, não é possível utilizar de longas exposições em alguns casos, pois a imagem pode sair "borrada"(Figura 2.4). Num cenário de fotografia em esporte, uma pessoa em uma corrida precisa ser fotografada em uma fração de segundo e uma paisagem, ao contrário, pode ser capturada durante mais de um segundo se a câmera estiver imóvel em um tripé.

Figura 2.4 – Impacto do tempo de exposição (em segundos) na captura de objetos em movimento

Fonte: Adaptado de (VIERO, 2018).

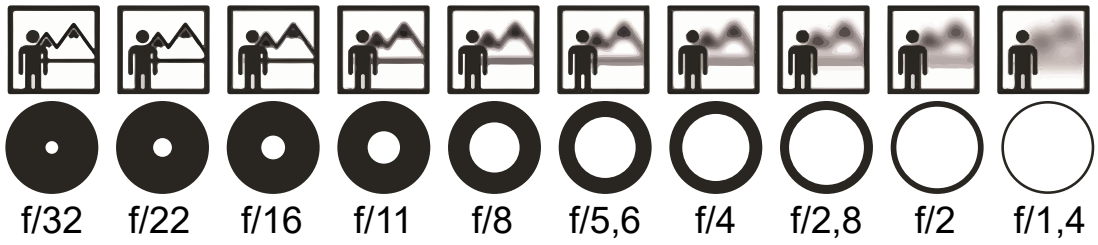
2.1.2.2 Abertura

A abertura é diâmetro do diafragma da lente, que permite a passagem de luz para o sensor (Figura 2.5). Isso determina um valor "f/número". Um baixo "f/número" como f/1.8, indica um alto valor de abertura e significa que a câmera irá receber mais luz (LOCHUN et al., 2015). A abertura também impacta na profundidade de campo, como demonstra a Figura 2.6, o que significa que um valor baixo também apresenta o ônus da dificuldade focalizar.

Figura 2.5 – Variações de abertura de uma lente

Fonte: Adaptado de (VIERO, 2018).

Figura 2.6 – Impacto da abertura na profundidade de campo



Fonte: Adaptado de (VIERO, 2018).

2.1.2.3 Sensibilidade (ISO)

O ISO (International Organization for Standardization), é um padrão internacional para a sensibilidade do sensor das câmeras. Essa sensibilidade também é configurável no sistema da câmera no momento da fotografia. Um valor baixo de ISO significa que o sensor precisa de mais tempo de exposição para captar mais luz, ao mesmo tempo, que reduz o ruído na imagem. (Figura 2.7) Um valor de ISO alto implica que a imagem final terá muito ruído, mas possibilita que ela seja registrada com um baixo tempo de exposição (LOCHUN et al., 2015). O ruído agregado pelo ISO também acaba prejudicando a fotografia, reduzindo o contraste e a saturação das imagens, o que também pode levar a posterização, que é o comprometimento total da fotografia, pois a foto perde resolução e criam-se falhas nos *pixels* da imagem.

Figura 2.7 – Variações do ISO e o ruído agregado

Fonte: Adaptado de (VIERO, 2018).

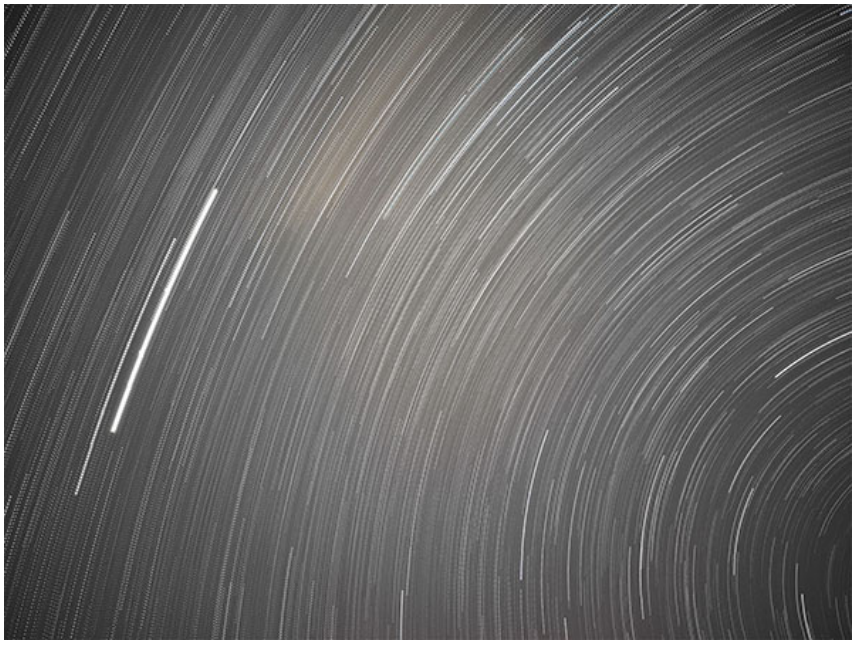
2.1.3 Formatos de Arquivos

As câmeras profissionais possibilitam salvar as imagens em diferentes formatos de arquivos, que inclui formato RAW, JPG *(Joint Photographic Experts Group)* ou ambos. Os arquivos JPG são uma versão reduzida dos formatos RAW, onde se aplica um algoritmo de compressão de imagens que acaba gerando perca de informações. Desse modo, arquivos RAW possuem a informação completa do sensor, sem nenhum tipo de compactação e acabam sendo muito grandes, mas permitem uma pós-produção mais precisa que acaba resultando em uma imagem com mais qualidade e detalhes (LOCHUN et al., 2015).

2.1.4 Rastro de Estrelas

O movimento de rotação da terra gera um movimento aparente no céu. Ao realizar uma fotografia de longa exposição, esse movimento será visível, criando o efeito de rastro de estrelas ou *star trail* (Figura 2.8).

Figura 2.8 – Fotografia com a captura de um *star trail*



Fonte: Autor.

2.1.4.1 Tempo de Exposição Máximo

Existe um limite de tempo máximo para que uma câmera, fixada em um tripé, permaneça captando luz sem que ocorra o fenômeno de *star trail*. Esse tempo depende de vários fatores, sendo os dois principais, a distância focal da lente e a posição da estrela.

A distância focal é importante pois uma lente com um longo comprimento amplia a imagem, da mesma forma que amplia o rastro das estrelas. Do contrário, lentes com ângulo mais aberto, de menor comprimento focal, tornam as coisas menores, incluindo o movimento das estrelas, e isso permite um tempo de exposição maior (COVINGTON, 2006).

A distância de um astro até a linha do equador celestial é chamada de declinação estrelar, sendo medida em graus. Quanto menor for essa distância, mais rápido a estrela aparenta se movimentar no céu. Seja a declinação simbolizada por σ, e a distância focal nomeado F, em milímetros , uma equação capaz de aproximar o limite do tempo de exposição (t_{max}) é dada em (2.1) (COVINGTON, 2006). Existem ainda outras metodologias que buscam aproximar o tempo máximo de exposição.

$$t_{max} = \frac{343}{F\cos(\sigma)} \ \ [s] \tag{2.1}$$

2.1.4.1.1 Regra dos 500

A regra dos 500 é uma equação que se baseia apenas distância focal, e o cálculo do tempo máximo é dado em (2.2). É uma regra simples, mas que permite uma aproximação razoável sobre o tempo máximo de exposição. Existem variantes dessa regra que alteram a constante no numerador, como a regra dos 300 ou a regra dos 400 (COX, 2021).

$$t_{max} = \frac{500}{F} \ \ [s] \tag{2.2}$$

2.1.4.1.2 Regra NPF

A regra NPF é uma evolução da equação (2.1), que considera múltiplos fatores para recalcular a constante do numerador (COX, 2021). O tempo de exposição máximo é calculado pela equação 2.3 com base na abertura da lente (N), na distância focal (F), no tamanho em micrômetros do sensor da câmera (p), na declinação da estrela para onde a câmera será apontada (σ), e um fator de multiplicação (k). O fator k é mantido em 1, porém pode ser aumentado até 3 para obter imagens mais nítidas e contrastantes (COX, 2021).

$$t_{max} = k \cdot \frac{16,9N + 0,1F + 13,7p}{F\cos(\sigma)} \ \ [s] \tag{2.3}$$

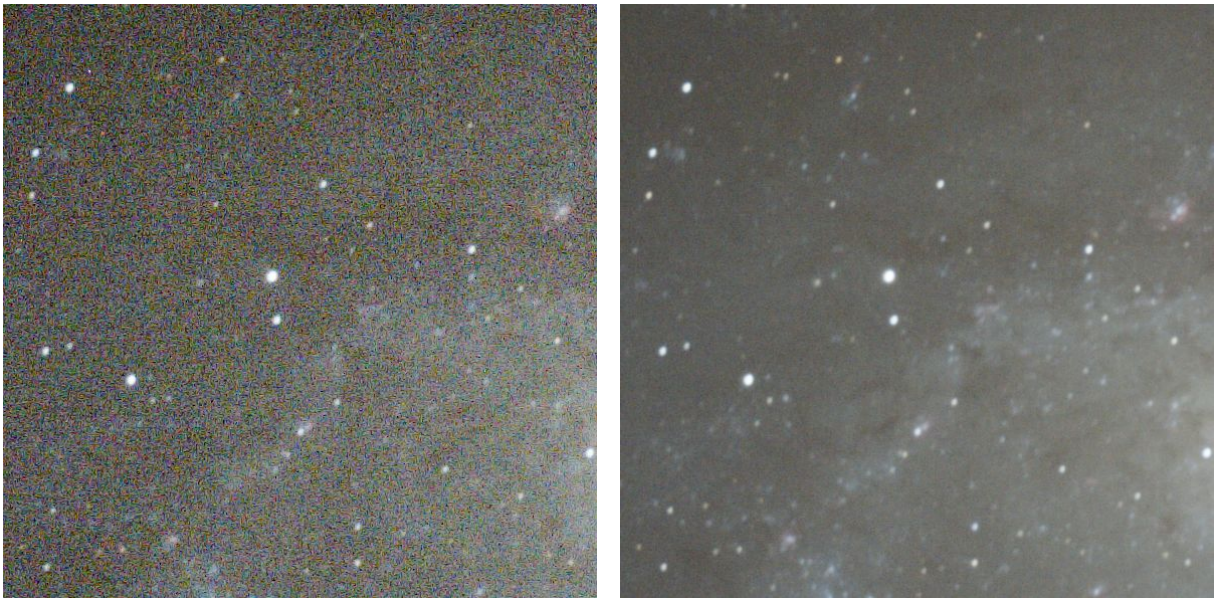
2.1.5 Empilhamento de Fotos

O fenômeno de *star trail* gera a necessidade do uso de ferramentas para compensar o movimento da Terra e permitir uma fotografia de longa exposição sem a criação de rastro. Essa compensação pode ser feita por *software*, realizando-se o empilhamento de fotos de curta exposição (ANDOLFATO, 2017).

O empilhamento consiste na junção de múltiplas imagens capturadas com a câmera montada em um tripé ou em uma montagem equatorial motorizada, que possibilita o somatório da luz capturada com essas fotos. Esse método de processamento é relevante para a astrofotografia e possibilita a redução de ruído usando imagens de calibração (LOCHUN et al., 2015). Existem inúmeros programas capazes de realizar esse processo como *Deep Sky Stacker*, *Sequator*, entre outros.

A combinação das imagens no pós-processamento não gera uma imagem mais luminosa ou colorida, o objetivo da combinação é o aumento da Relação Sinal Ruído (SNR). A única forma de gerar uma imagem final com mais luz e cores é realizando uma sequência de fotografias com maior tempo de exposição (DEEP SKY STACKER, c2021a). As figuras 2.9a e 2.9b comparam o resultado de uma imagem que passou pelo processo de empilhamento.

Figura 2.9 – Efeito da combinação de imagens. (a) Imagem Original e (b) Empilhamento de 32 imagens



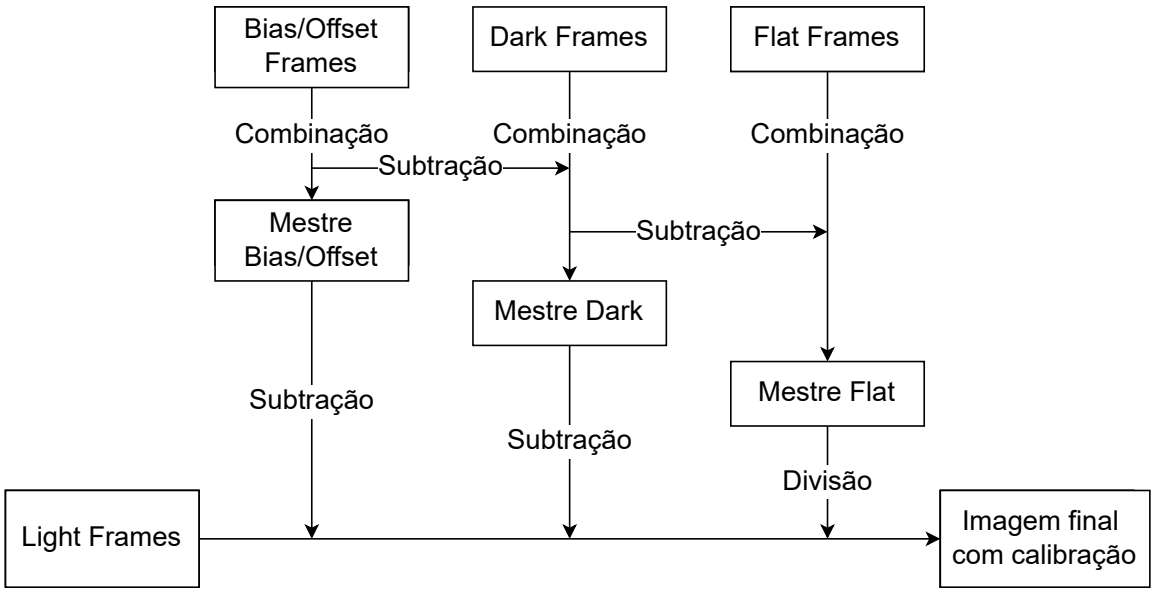
(a) (b)

Fonte: (DEEP SKY STACKER, c2021a).

2.1.5.1 Imagens de Calibração

As fotografias registradas sobre um alvo celeste são chamadas de *Light Frames* e estas podem ser empilhadas como escrito anteriormente. No entanto, é possível realizar um processo de calibração do empilhamento, fornecendo imagens de calibração (DEEP SKY STACKER, c2021b). O processo é feito combinando fotos chamadas de *Dark Frames*, *Bias Frames*, *Flat Frames* e *Dark Flat Frames* (não muito utilizado). Essas imagens são extras e precisam ser fotografadas com a câmera em condições específicas e posteriormente adicionadas ao *software* durante o processo de empilhamento (DEEP SKY STACKER, c2021a). O resultado entregue pelo *software* será uma imagem final calibrada, que irá passar por um processo simplificado no diagrama da Figura 2.10.

Figura 2.10 – Diagrama do processo de calibração do empilhamento sem o uso de *Dark Flat Frames*



Fonte: Adaptado de (DEEP SKY STACKER, c2021a).

2.1.5.1.1 *Dark Frames*

Os *Dark Frames* são fotografias que indicam ao software a localização do sinal de ruído das fotografias. São necessárias de 10 a 20 fotos com a lente tampada para criar a calibração, as quais devem necessariamente ser fotografadas com ISO, tempo de exposição e condições ambientais iguais aos *Light Frames*.(DEEP SKY STACKER, c2021b)

2.1.5.1.2 *Bias (Offset) Frames*

Os *Bias/Offset Frames* são usados para remover sinais de ruído na leitura do sensor da câmera. Essas fotografias devem ser capturadas no menor tempo de exposição possível, com lente tampada, na mesma configuração de ISO dos *Light Frames*. São necessárias cerca de 10 a 20 fotos para que a calibração funcione adequadamente. A temperatura da câmera não é um fator relevante nesse caso (DEEP SKY STACKER, c2021b).

2.1.5.1.3 *Flat Frames*

Flat Frames são imagens de calibração capturadas com mesmo ISO e abertura dos *light frames* colocando uma superfície branca (como uma folha de papéu, por exemplo) na frente da lente, incidindo luz na folha. Elas têm o objetivo de indicar a vinheta da lente (escurecimento nas bordas da imagem), além da distribuição não uniforme de luz provocada por pó ou riscos na lente. Novamente, são necessários de 10 a 20 imagens (DEEP SKY STACKER, c2021b).

2.1.6 Métodos de Rastreamento

Tendo em vista o limite do tempo de exposição e o movimento de rotação da Terra, discutidos na seção 2.1.4.1, os *softwares* de empilhamento possuem algoritmos que compensam a rotação das estrelas, rotacionando as imagens fotografadas no sentido oposto, e realizando o empilhamento dessas imagens após esse ajuste das fotos. Esse método compensa o ruído, mas como os tempos de exposições são curtos, torna-se mais difícil obter cor e contrastes nos objetos celestes. Isso só é possível capturar aumentando o tempo de exposição.

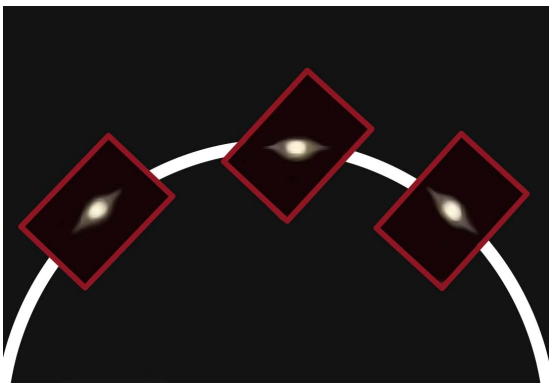
Então, para realizar astrofotografias de longa exposição, é necessário o uso de um rastreador físico que movimenta a câmera no sentido de rotação aparente das estrelas, garantindo que não ocorrerá o efeito de *star trail*. Existem dois métodos de rastreamento: Alt-Azimutal e Equatorial (LOCHUN et al., 2015).

2.1.6.1 Alt-Azimutal

Uma montagem Alt-Azimutal funciona movendo uma câmera ou um telescópio por meio dos eixos vertical e horizontal, alterando o azimute e a altitude simultaneamente. Isso requer um sistema com dois motores para realizar o rastreamento, o que torna essa

montagem mais cara e complexa. Além disso, para astrofotografias, essa montagem acaba não sendo indicada pois ela não consegue compensar a rotação aparente dos astros, que também é gerado pelo movimento de rotação da terra (Figura 2.11) (LOCHUN et al., 2015).

Figura 2.11 – O enquadramento não rotaciona com o astro na montagem Alt-Azimutal

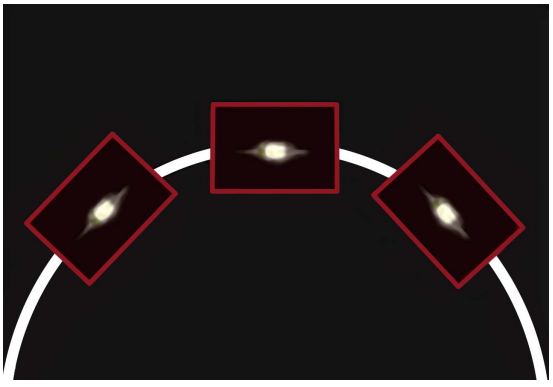


Fonte: Adaptado de (LOCHUN et al., 2015).

2.1.6.2 Equatorial

Ao contrário do modelo de montagem comentado na seção anterior, uma montagem equatorial consegue compensar a rotação aparente dos astros (Figura 2.12) e, por esse motivo, é a melhor opção de mecanismo para realizar astrofotografias. Isso ocorre pois essa construção realiza o movimento da câmera de forma circular, na mesma velocidade de rotação aparente da Terra, após alinhar o eixo de altitude junto com o meridiano polar (eixo norte-sul) (LOCHUN et al., 2015). Esse sistema requer somente um motor, porém, precisa também de um método acurado de alinhamento com o meridiano, este que será explorado na próxima seção.

Figura 2.12 – O enquadramento se mantêm constante na montagem equatorial

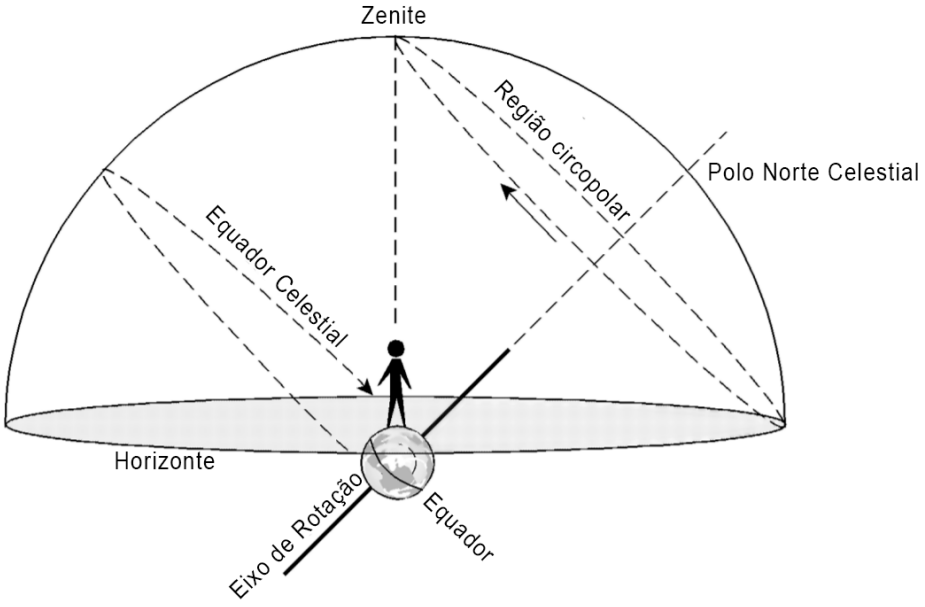


Fonte: Adaptado de (LOCHUN et al., 2015).

2.2 PLATAFORMAS EQUATORIAIS

Plataformas Equatoriais são mecanismos que se baseiam em uma montagem equatorial para realizar o rastreamento das estrelas no céu. Existem modelos para telescópios e outros específicos para astrofotografia, sendo este último o foco deste trabalho. Essa montagem requer o alinhamento do eixo de rotação da plataforma com o eixo de rotação celeste que, para uma pessoa localizada no hemisfério norte, será o polo norte celestial, e para alguém no hemisfério sul, será o polo sul celestial (Figura 2.13).

Figura 2.13 – Projeção do eixo de rotação da terra no céu marca os polos Norte/Sul; a projeção da linha do equador determina o Equador celestial.



Fonte: (MACK, 2008).

A posição do polo norte/sul celestial, no céu, depende da posição geográfica (latitude) da pessoa/equipamento de observação e é alinhado com o eixo de rotação da terra. Depois que o eixo da plataforma está alinhado com o polo celeste, ela começa a rotacionar no sentido e mesma taxa da rotação do planeta, compensando o movimento aparente do céu.

2.2.1 Modelos de Montagem *Barn Door*

Barn Door é um modelo de montagem que se caracteriza por possuir duas bases acopladas, onde uma é fixada no tripé do fotógrafo, e a outra é móvel, fixando a câmera que será rotacionada para acompanhar o movimento aparente do céu (Figura 2.14). Seu funcionamento é igual a abertura de uma porta com dobradiças (DAVID, K., 2015).

Figura 2.14 – Exemplo de Montagem *Barn Door*

Fonte: (SERONIK, 2007).

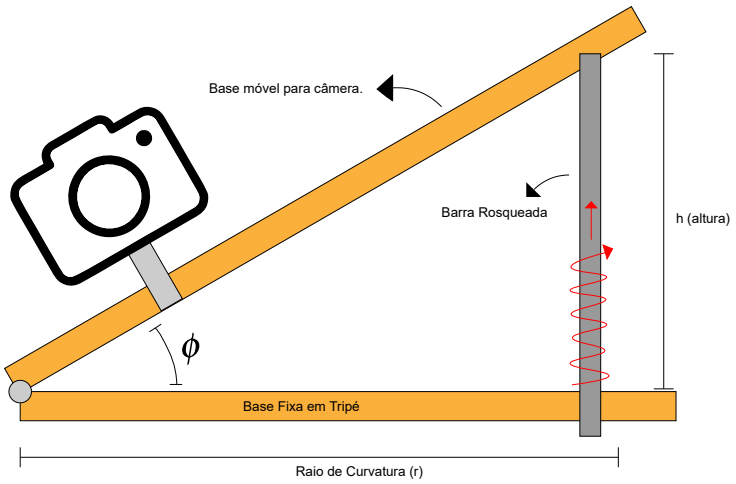
Esses modelos são tradicionalmente conhecidos pela comunidade de astrofotografia por serem de baixo custo, pois é possível automatizar o movimento da câmera sem a necessidade de um motor potente e caro. Além disso, comparando a modelos de montagem usados em sistemas comerciais, um sistema *Barn Door* pode ser facilmente modificado ou reparado. Entretanto, perdem para modelos comerciais no quesito transportabilidade e precisão (DAVID, K., 2015). A transportabilidade é a facilidade com que o equipamento pode ser transportado de um ponto ao outro, isso leva em conta o formato da estrutura, volume e peso.

Dentre os modelos de *Barn Door*, os mais comuns são: montagem com braço simples; montagem com braço Duplo e montagem curva. A primeira (Figura 2.15) é composta por uma base fixa conectada a base da câmera que é movida por um eixo perpendicular à parte fixa. Esse sistema acaba tendo algumas limitações para manter uma variação constante no ângulo de rotação da câmera, que, apesar da elevação do eixo ser constante, o ângulo de rotação não é, o que acumulará erro de rastreamento. Isso gera erros de rastreamento que limitam o uso desse modelo para exposições com, no máximo, 15 minutos (TROTT, 1989).

Trot (1989) desenvolveu um mecanismo de braço duplo (Figura 2.16) que consegue aumentar para até 1h o tempo máximo de exposição, em comparação com o modelo anterior. Contudo, tem como desvantagem a complexidade do sistema, que exige várias peças para montagem e configuração.

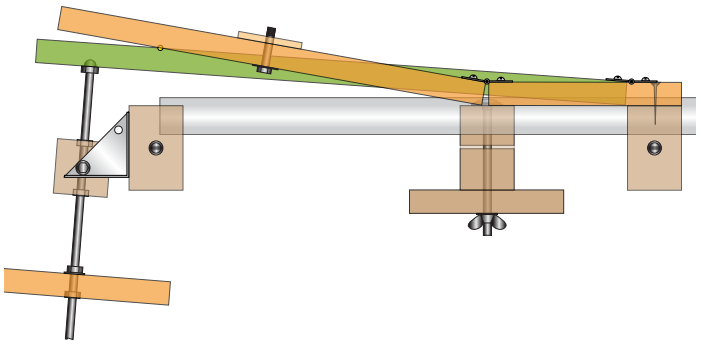
Por fim, a montagem curva é composta por uma barra roscada curva, que não possui o problema de compensação de velocidade, pois ela acompanha a curvatura do movimento da plataforma. A dificuldade dessa montagem provém da curvatura do eixo de rotação, que pode trazer questões relacionadas à estabilidade devido à necessidade de

Figura 2.15 – *Barn Door* com braço simples



Fonte: Adaptado de (TROTT, 1989).

Figura 2.16 – Vista lateral do mecanismo de Braço Duplo



Fonte: Adaptado de (TROTT, 1989).

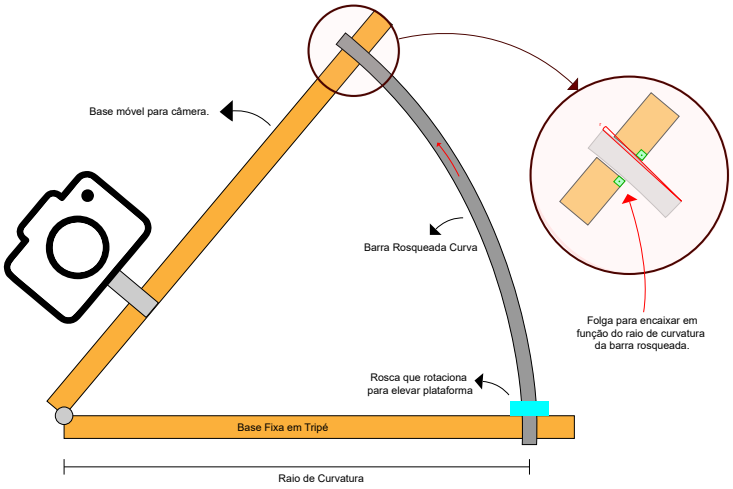
folga nas bases (Figura 2.17). Outro detalhe é que a barra não pode ser rotacionada. A movimentação deve ser feita através de uma rosca que é rotacionada na base, e realiza o deslocamento da barra roscada para cima ou para baixo (DAVID, K., 2015).

Comparando os atributos dos modelos de *Barn Door*, a montagem curva é a mais apropriada para o projeto proposto, visto que não demanda a construção de um mecanismo demasiadamente complexo. Dessa forma, ela reduz a quantidade de materiais e processos, bem como a lógica de controle do motor, minimizando ainda mais os custos, ao passo que potencializa o resultado do projeto.

2.2.2 Métodos de Alinhamento Polar

O alinhamento com o polo norte/sul celeste é fundamental para a execução da astrofotografia através de uma plataforma equatorial, e erros podem comprometer o funci-

Figura 2.17 – Modelo de Montagem Curva e o problema da folga no mecanismo



Fonte: Adaptado de (JONES, 1980).

onamento do sistema. Quanto mais bem alinhada está a plataforma, maior será o tempo de exposição que ela conseguirá obter sem gerar rastros nas estrelas. Para realizar esse alinhamento, existem dois métodos: Localizar a estrela Polar que indica a posição do polo celeste e/ou utilizar de instrumentação para posicionar a plataforma nos valores de azimute e inclinação corretos, dada a posição da plataforma no planeta.

2.2.2.1 Localização da Estrela Polar

Existem duas formas de alinhamento por meio da localização no céu da estrela polar. A primeira delas é através um *laser*, que é alinhado com a estrela polar para indicar o alinhamento (Figura 2.18). No entanto, esse não é um método extremamente confiável, pois a estrela polar não é exatamente centrada no polo norte celeste, dessa forma o alinhamento não fica preciso. As principais vantagens desse método são a facilidade e o baixo custo que é, em média, na ordem de US$ 10.[1]

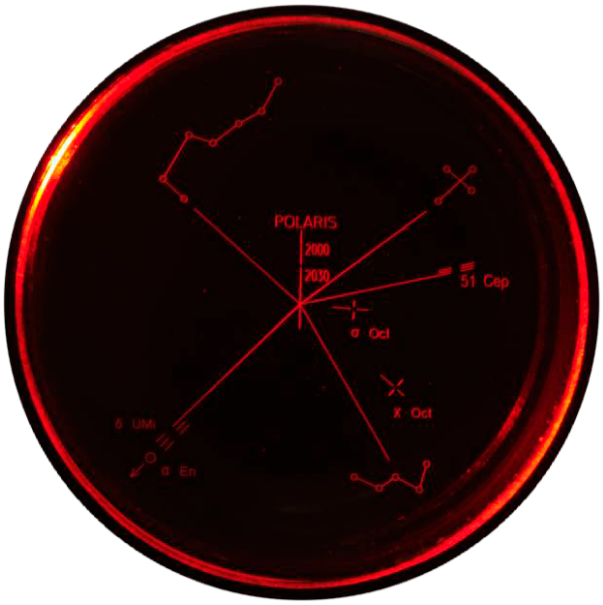
O segundo método envolve uma luneta que permita a identificação das constelações que indicam o polo Norte/Sul (Figura 2.19). No hemisfério Norte, procura-se a estrela Polaris; no hemisfério Sul, busca-se a constelação de Sigma Octantis e o Cruzeiro do Sul. Ele é um método bem confiável, porém, no hemisfério Sul, isso normalmente é mais difícil, pois essas constelações são de alta magnitude. Isso significa que possuem um baixo brilho, tornando-as difíceis de serem localizadas no céu. As lunetas buscadoras têm um custo variado, são normalmente comercializadas fora do Brasil e tem um custo de no mínimo US$ 80 .[2]

[1] Considerando modelos pesquisados em agosto de 2021
[2] Considerando modelos pesquisados em agosto de 2021

Figura 2.18 – Alinhamento por *laser*

Fonte: (DYER, 2019).

Figura 2.19 – Visor de uma luneta para astrofotografia, contendo marcadores para alinhar as estrelas



Fonte: Adaptado de (BRESSER, c2021).

Ambos os métodos possuem a desvantagem de requerer um céu limpo na região polar (Figura 2.20) e baixos níveis de poluição luminosa para identificar as estrelas, porém, apresentam a simplicidade como vantagem.

Figura 2.20 – Alinhamento impossível sem colaboração do tempo limpo

Fonte: (EGAN, c2021b).

2.2.2.2 Instrumentação

Através de instrumentos de medição, é possível realizar o alinhamento da plataforma equatorial separando o processo em duas etapas. Na primeira etapa é feito o ajuste do azimute para alinhar a plataforma com o polo norte geográfico. Posteriormente, ela deve ser inclinada até o ângulo referente ao polo norte celeste que é dado pelo ângulo da latitude do local onde a plataforma está localizada, ajustando a elevação.

2.2.2.2.1 Ajuste de Azimute

O ajuste do azimute pode ser realizado com uma bússola ou um magnetômetro que possibilite indicar o polo norte magnético. No entanto, o polo norte geográfico possui uma diferença com o polo norte magnético devido à oscilação do campo magnético do planeta. Essa discrepância é chamada de declinação magnética.

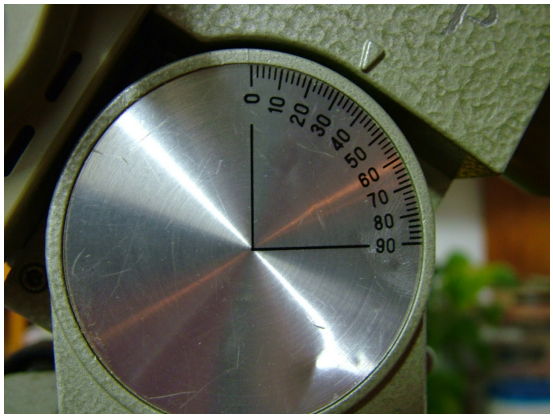
Além disso, devido à inconsistência do campo magnético, o valor da declinação é diferente para cada localização ao redor do globo terrestre, mas pode ser calculada usando modelos magnéticos globais que são o resultado de pesquisas com sensores em satélites e apresentam valores com acurácia de 0,5 graus (INFORMATION, c2021).

2.2.2.2.2 Ajuste de Elevação

O ajuste de elevação para uma determinada latitude pode ser feito através de marcações de ângulo em um eixo ou transformador (Figura 2.21), ou com sensores de posição inercial (acelerômetro e giroscópio) (OBSERVATÓRIO LUPUS, 2014). É evidente que o

primeiro método não é preciso, pois depende obrigatoriamente de um bom processo de manufatura e calibração, além da falta de precisão para ângulos de latitude com valores decimais.

Figura 2.21 – Marcação da latitude para ajuste de elevação



Fonte: (OBSERVATÓRIO LUPUS, 2014).

2.2.2.3 Método Drift

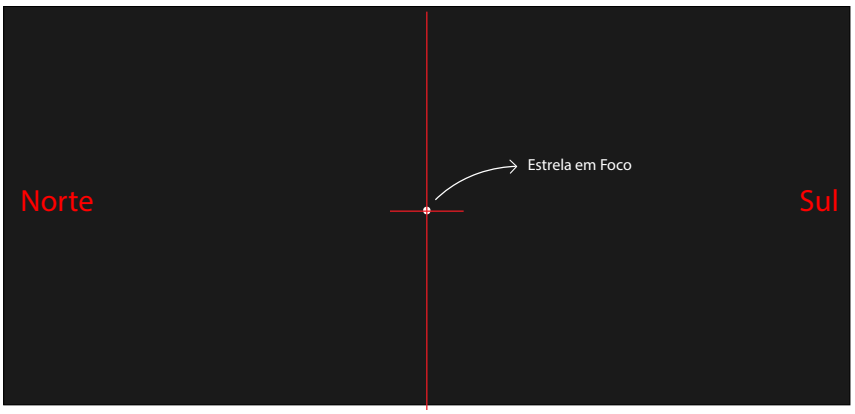
O alinhamento da plataforma com o polo celeste é fundamental para uma longa exposição usando uma lente com grande comprimento focal. Se a montagem não estiver corretamente alinhada, ela não impedirá o surgimento de rastro na fotografia por um longo período de tempo (LOCHUN et al., 2015).

Plataformas que são alinhadas com uma luneta polar conseguem atingir normalmente 180 segundos de exposição para lentes de baixo comprimento focal (baixo *zoom*). Se mais tempo for necessário ou for usada uma lente com muita ampliação, então o método *Drift* de alinhamento é mandatório, ainda que, não é possível realizar esse ajuste de precisão sem que a plataforma já tenha sido previamente alinhada (LOCHUN et al., 2015).

O método consiste em localizar, primeiramente, uma estrela brilhante o suficiente para visualizar no visor da câmera, e que esteja na linha do equador polar. Habilitando uma linha de grade no visor, o usuário deve alinhar a estrela escolhida de forma que fique centralizada no cruzamento da grelha (Figura 2.22) e, então, ligar o rastreador, observando se haverá movimentação da estrela para algum dos lados do visor. Na sequência, o usuário deve apontar para uma estrela ao leste ou oeste, no horizonte, e observar para qual lado (esquerda/direita ou Norte/Sul) do visor haverá fuga (Figura 2.23). O lado (esquerda ou direita) que indica Norte ou Sul depende para onde a câmera estiver apontando e a orientação dela (LOCHUN et al., 2015).

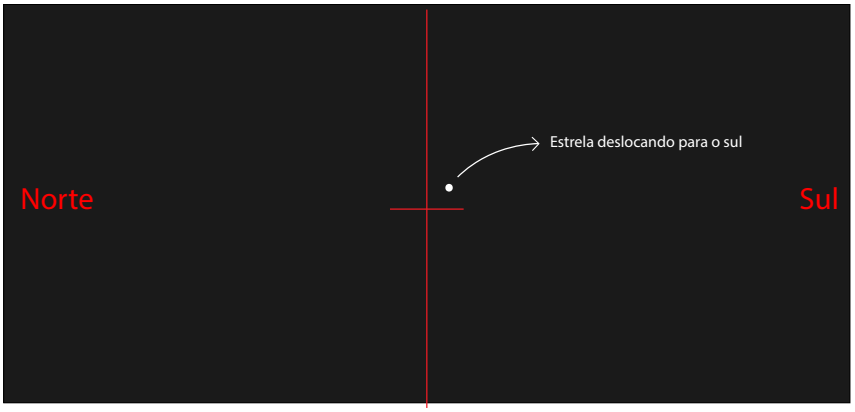
Dependendo para qual lado do visor (norte/sul) houver fuga, deverá ser feito um ajuste específico no alinhamento, que também é diferente para cada hemisfério do pla-

Figura 2.22 – Estrela centralizada na grelha do visor da câmera



Fonte: Adaptado de (OBSERVATÓRIO LUPUS, 2014).

Figura 2.23 – Estrela apresentando fuga para a direita que, neste caso, aponta para o Sul



Fonte: Adaptado de (OBSERVATÓRIO LUPUS, 2014).

neta. O fotógrafo deve saber para qual lado do visor é norte ou sul, pois será isso que determinará a correção para movimentar a plataforma (Tabela 2.1). A primeira estrela ajuda a ajustar o azimute da plataforma e a segunda colabora para uma elevação correta. Além disso, a precisão do ajuste dependerá do tempo decorrido para a estrela sair da marcação do visor. Esse processo deve ser repetido, com tentativa e erro, até que a estrela permaneça parada pelo tempo mínimo desejado pelo fotógrafo para a lente que estiver usando (LOCHUN et al., 2015).

É importante ressaltar que para esse método ser possível, o tripé onde a plataforma é montada deve possuir *knobs* para ajuste mais acurado, como na cabeça de tripé da Figura 2.24, que possui um *knob* para movimentação somente de azimute, outro para movimento livre, e um terceiro para um ajuste mais preciso de ambas as posições. Além dessa, existem cabeças para filmagem que permitem um controle ainda mais suave e preciso, sendo extremamente recomendadas para esse método (Figura 2.25).

Tabela 2.1 – Ajustes do método *Drift* para cada caso

Localização da estrela	Lado da Fuga	Correção (Hemisfério Sul)	Correção (Hemisfério Norte)
Meridiano	Norte	Azimute para Leste	Azimute para Leste
	Sul	Azimute para Oeste	Azimute para Oeste
Leste	Norte	Altura para cima	Altura para baixo
	Sul	Altura para baixo	Altura para cima
Oeste	Norte	Altura para baixo	Altura para cima
	Sul	Altura para cima	Altura para baixo

Fonte: Adaptado de (OBSERVATÓRIO LUPUS, 2014).

Figura 2.24 – Cabeça de Tripé com possibilidade para ajuste com precisão

Fonte: (Optison, c2021).

Figura 2.25 – Cabeça de Tripé com possibilidade para ajuste com mais precisão

Fonte: (Manfrotto, c2021).

2.2.3 Soluções Comerciais Existentes

Existem inúmeras soluções comerciais para o problema proposto, porém, todos usam uma luneta ou um *laser* como método de alinhamento polar. Esses produtos também diferem em especificações e orçamentos. A Tabela 2.2 ilustra alguns sistemas, dentre os disponíveis no mercado, comparando suas funcionalidades.

Contudo, na realidade brasileira, o preço mostrado passaria ainda por impostos,

Tabela 2.2 – Comparativo das Soluções de Mercado

	Nyx Tracker	iOptron	Vixen Optics	SkyWatcher
Preço (US$)	115	299	399	299
Carga Máxima (kg)	2.25	3	2	3
Erro periódico (arcsec)	115	100	50	50
Volume (cm^2)	155	490	323	220
Peso (kg)	0,4	1,15	0,79	0,72
Alinhamento	*Laser*	*Polar Scope*	*Polar Scope*	*Polar Scope*

Fonte: Adaptado de (EGAN, c2021a).

tornando a compra mais inviável. O Nyx Tracker (Figura 2.26) é o único da lista que possui uma estrutura de *Barn Door*, e é o sistema mais acessível, porém o método de alinhamento é de difícil execução no hemisfério sul.

Figura 2.26 – Nyx Tracker

Fonte: (EGAN, c2021a).

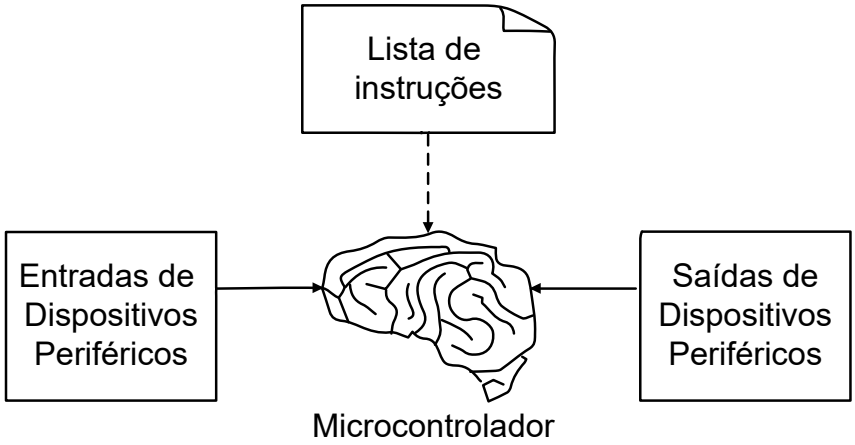
2.3 OBJETIVOS

Pelo *benchmark* exposto, fixaram-se como objetivos o desenvolvimento de uma solução robusta, visualmente elegante, e que se aproxime das propriedades do modelo comercial mais acessível, com o custo inferior a US$115 . Além disso, deve ter como diferencial um aplicativo que permita uma fácil interação do usuário com o sistema, facilitando o processo de configuração e alinhamento polar.

2.4 SISTEMAS EMBARCADOS

Um sistema embarcado é definido como um conjunto de periféricos controlados por um microcontrolador ou microprocessador (Figura 2.27) de uma forma oculta ao usuário final. O "cérebro"funciona por meio de um *software*, que é um conjunto de instruções elaborado por um programador e é executado em *hardware* (RUSSEL, 2010).

Figura 2.27 – Visualização conceitual de um sistema embarcado.



Fonte: Adaptado de (RUSSEL, 2010).

Os periféricos são o conjunto de dispositivos conectados ao controlador para executar uma leitura de dados ou ação externa. Sensores, botões e interfaces são exemplos de dispositivos periféricos para leitura de dados. Motores, LEDs (Diodos Emissores de Luz) e atuadores lineares são exemplos que executam uma ação ordenada. As conexões destes periféricos com os controladores são construídas por meio de condutores e semicondutores como resistores, capacitores, transistores, diodos, etc. (RUSSEL, 2010).

2.5 PERIFÉRICOS

No contexto do projeto, propõe-se um sistema embarcado que utiliza sensores para realizar a leitura da posição geográfica e inercial da plataforma equatorial para guiar o usuário na sua configuração. Após esse ajuste, o usuário pode ativar um atuador que irá realizar o rastreamento do movimento aparente das estrelas no céu.

Dessa forma, é necessário a leitura de Sensores de Posição Inercial (IMU) e um sinal de GPS (Sistema de Posicionamento Global). É preciso também um motor de passo que servirá como o atuador da plataforma.

2.5.1 Motor de Passo

O motor de passo é um tipo de atuador comumente utilizado em aplicações de controle, como impressoras 3D, máquinas CNC, entre outros sistemas que requerem uma boa precisão. Além disso, os motores de passo também possuem outros pontos fortes: não tem escovas internas que geram a necessidade de manutenção, evitando desgastes; não são dependentes da carga aplicada, mantendo a velocidade de rotação constante desde que o torque necessário não exceda os limites; giram em passos que são incrementados de forma constante durante a sua aplicação, possibilitando um controle preciso sem a necessidade de um sensor com *feedback*; e, por fim, conseguem manter o sistema imóvel quando estacionados (JONES, c2004).

Existem dois tipos mais comuns de motores de passo (Figura 2.28): unipolar e bipolar, que se diferenciam pela forma em que as bobinas são conectadas e energizadas para movimentar o rotor. No motor unipolar, cada bobina é controlada individualmente; no motor bipolar, as bobinas são agrupadas e controladas em conjunto para ganhar desempenho. (ADVANCED MICRO SYSTEM INC., c2021b).

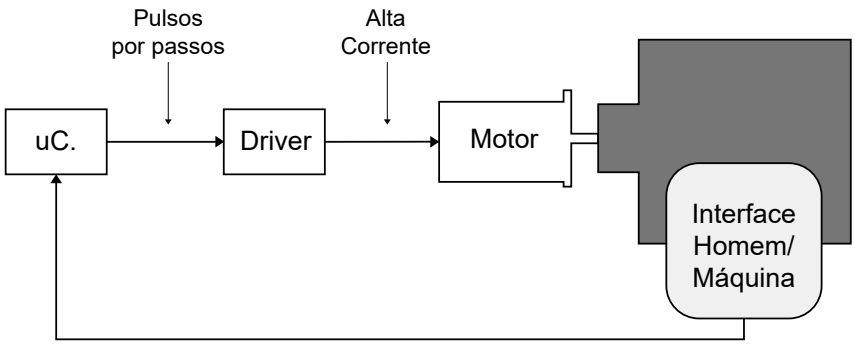
Figura 2.28 – Modelos de motores: (a) um motor unipolar, (b) motor bipolar

Fonte: Adaptado de (ADVANCED MICRO SYSTEM INC., c2021b).

2.5.1.1 Driver

Para que a corrente flua pelas bobinas e seja conduzida por um microcontrolador, é preciso um *driver* que realize o chaveamento pois, normalmente, microcontroladores não conseguem fornecer corrente para energizar um motor de passo. Os *drivers* são circuitos integrados com transistores e atendem ao princípio de funcionamento dos tipos de motores. O diagrama do controle de um motor pode ser simplificado como demonstra a Figura 2.29, onde o microcontrolador — após receber um sinal da máquina, ou de um usuário — atua em um *driver*, enviando pulsos pelas suas saídas. O driver recebe esses sinais e atua diretamente no motor, permitindo a passagem de corrente e operando a máquina.

Figura 2.29 – Diagrama de controle de um motor de passo



Fonte: Adaptado de (ADVANCED MICRO SYSTEM INC., c2021b).

2.5.1.2 Torque

Torque é o momento da força que um motor consegue aplicar no seu eixo de rotação. Normalmente, motores de passo tem diferentes valores de torque que os caracterizam para cada situação de aplicação e são descritos pelos fabricantes (JONES, c2004). Além disso, é preciso manter-se atento ao consumo necessário: quanto maior o torque exigido do motor, maior será a corrente que ele irá demandar sob a carga aplicada. Têm-se os seguintes tipos de torque em motores de passo.

- *Holding torque* – Torque mínimo para girar o eixo do motor enquanto os enrolamentos são energizados.
- *Detent torque* – Torque necessário para tirar o eixo da inércia quando os enrolamentos não são energizados.
- *Pull-in torque* – Torque máximo suportado para iniciar a rotação dos eixos, sem perder passos, e em uma aceleração constante.
- *Pull-out torque* – Torque máximo suportado quando o motor já estiver rotacionando em velocidade constante.

2.5.1.3 Tamanho do Passo

Cada sinal que é enviado ao motor faz com que o rotor gire um passo. Esse tamanho de passo é uma característica de cada atuador e define a resolução mínima de movimento angular que o eixo consegue realizar. Essa característica é importante pois quanto maior a precisão da aplicação, maior deve ser o número de passos no motor. Por exemplo, um motor que contenha 200 passos consegue ter uma precisão de $360^\circ \div 200 = 1,8^\circ$.

Apesar da precisão dos motores, é possível que ocorram erros ao rotacionar em cada passo. Esses problemas derivam de falhas mecânicas inerentes aos mecanismos internos do motor, não são cumulativas e oscilam de 3 a 5% de erro no deslocamento (ADVANCED MICRO SYSTEM INC., c2021b).

2.5.1.4 Formas de Controle

Existem três diferentes abordagens para realizar o chaveamento das bobinas de um motor de passo usando *drivers*: *Full Step*, *Half Step* e *Microstep*. Essas técnicas são aplicáveis a qualquer tipo de motor.

2.5.1.4.1 Full Step

Essa é a forma mais simples de controlar o motor: para cada pulso do controlador, uma bobina é ativada completamente e isso é alternado ao longo do tempo (Tabela 2.3). Nesse modo, o motor consome mais corrente, oferece mais torque e gera mais ruído. Além disso, considerando o exemplo de motor da seção anterior, com 200 passos, o controlador irá precisar emitir 200 pulsos no *driver* para que o motor dê um giro de 360 graus (ADVANCED MICRO SYSTEM INC., c2021b).

Tabela 2.3 – Sequência de acionamentos das bobinas do motor pelo microcontrolador no modo *Full Step*

Passo	B.1	B.2	B.3	B.4
1	1	0	0	0
2	0	1	0	0
3	0	0	1	0
4	0	0	0	1

Fonte: (MURTA, 2016).

2.5.1.4.2 Half Step

Um controle *Half Step* proporciona uma redução no ruído e ganho de eficiência, porém reduz o torque máximo do motor. Essa abordagem funciona subdividindo as etapas de controle no modo *Full Step*, consumindo 400 pulsos do controlador que acionam 2 bobinas para cada pulso, com um sinal intermediário acionando somente uma bobina (Tabela 2.4) (ADVANCED MICRO SYSTEM INC., c2021b).

Tabela 2.4 – Sequência de acionamentos das bobinas do motor pelo microcontrolador no modo *Half Step*

Passo	B.1	B.2	B.3	B.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

Fonte: (MURTA, 2016).

2.5.1.4.3 Microstep

Uma abordagem de *Microstep* busca suavizar ainda mais o controle do motor para reduzir o ruído e ganhar eficiência ao custo de perder torque. Esse método subdivide o controle em ainda mais etapas, requerindo *drivers* especializados que podem energizar as bobinas de forma suave ao longo do tempo.

2.5.2 GPS

O GPS (*Global Positioning System*) é um sistema de navegação por rádio que recebe informações via satélite para determinar a posição global de um objeto (JAUCH; SILVA; PAZ, 2014). Atualmente, todo *Smartphone* ou aparelho celular contém um circuito capaz de realizar a leitura do sinal GPS e determinar sua localização atual. No entanto, a precisão da localização depende de um bom sinal dos satélites, podendo ter poucos metros de erro.

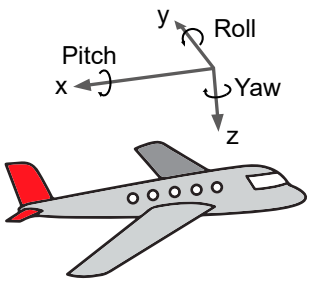
O valor retornado pelo GPS pode ser dado em coordenadas geográficas ou em UTM (*Universal Transversa de Mercator*) que é um sistema de coordenadas planas. As coordenadas geográficas podem ser expressadas em: graus-minutos-segundos ($ggg^o mm$”ss,s); graus-minutos-decimais ($ggg^o mm,m$”); e graus-decimais (ggg,g^o) (JAUCH; SILVA; PAZ, 2014).

2.5.3 Sensores de Posição Inercial

A posição de uma plataforma qualquer pode ser descrita por um sistema de coordenadas. Um padrão muito comum é o sistema RPY (*Roll, Pitch, Yaw*) que expressa a posição por meio dos 3 ângulos posicionais do objeto (Figura 2.30) (ZANONI, 2012). Os 3

ângulos são relativos a um dos 3 eixos *x*, *y* ou *z* e podem ser traduzidos, respectivamente, para ângulo de rolagem (roll), arremesso (pitch) e guinada (yaw).

Figura 2.30 – Sistema de ângulos *Roll(ϕ)-Pitch(θ)-Yaw(ψ)* para um avião



Fonte: Adaptado de (ZANONI, 2012).

Para aferir esses ângulos, em um sistema qualquer, é necessário o uso de sensores de posição inercial (IMU). Existem 3 modelos principais que são comumente usados de forma integrada para fornecer o máximo de precisão possível: Acelerômetro, Giroscópio e Magnetômetro. Todos esses serão obrigatórios para desenvolvimento do projeto proposto.

2.5.3.1 Acelerômetro

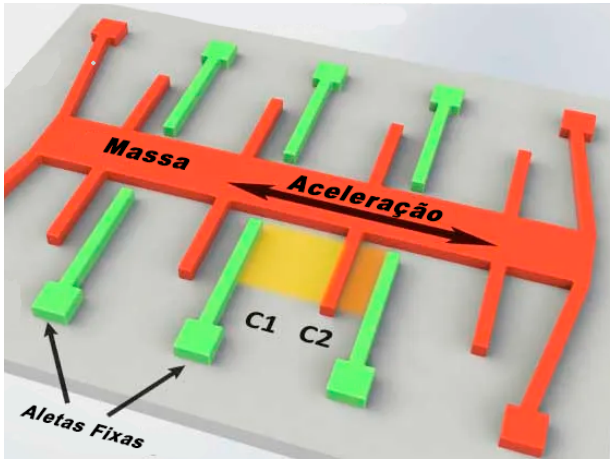
Acelerômetros são sensores que medem a aceleração de um sistema linearmente em relação a um eixo. Todas as forças envolvidas são mensuradas, como, por exemplo, a aceleração da gravidade (ZANONI, 2012), e o valor é dado em G ou (m/s^2). Por meio da força da gravidade, é possível decompor os vetores de força medindo a aceleração nos 3 eixos principais, e por meio disso calcular os ângulos inerciais de um objeto.

Um acelerômetro digital é um Sistema Micro Eletromecânico (MEMS) que funciona através da medição da capacitância entre duas aletas das estruturas internas, como demonstra a Figura 2.31. A alteração na capacitância corresponde à aceleração com que o bloco de massa interno se move (NEDELKOVSKI, 2016).

2.5.3.2 Giroscópio

Giroscópios são sensores capazes de medir a taxa da variação angular de um corpo que gira em torno de um eixo e é mensurado em $(^\circ/s)$ (ZANONI, 2012). Um giroscópio microeletrônico consiste em um ressonador, que vibra dentro de um espaço entre bases de apoio, detectando uma alteração na ressonância de acordo com o efeito de aceleração angular (Figura 2.32). Para obter o ângulo inercial de um objeto, considerando um giroscópio para cada eixo de rotação, é preciso integrar (somar) os valores de velocidade angular

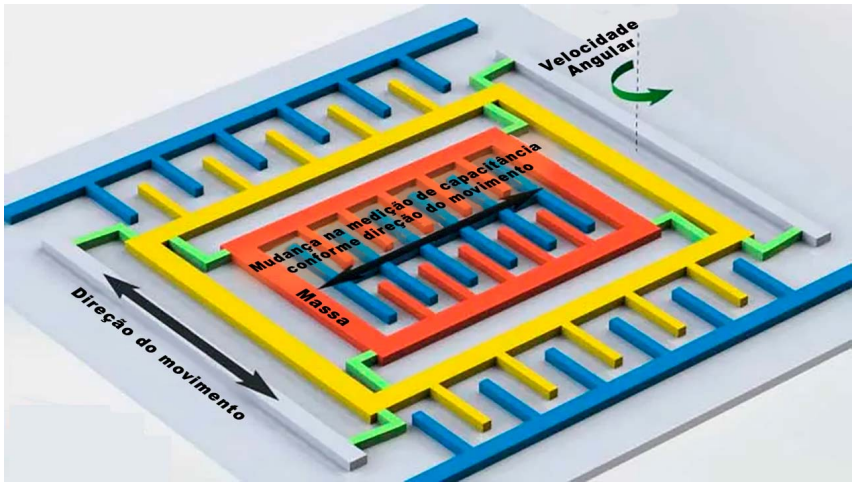
Figura 2.31 – Visualização de um acelerômetro digital: a massa laranja se move, alternando a distância de sua aleta com as partes em verde, oscilando a capacitância C1 e C2



Fonte: Adaptado de (NEDELKOVSKI, 2016).

medidos ao longo do tempo (LAGE, 2016).

Figura 2.32 – Vista interna de um giroscópio digital



Fonte: Adaptado de (NEDELKOVSKI, 2016).

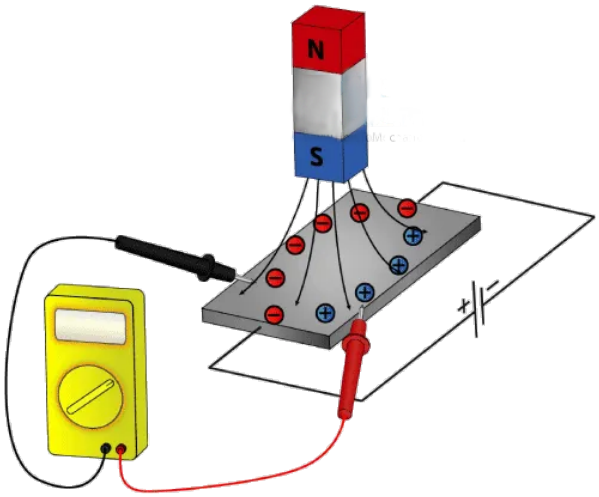
2.5.3.3 Magnetômetro

Magnetômetros são sensores capazes de medir a intensidade do campo magnético em um eixo. Quando três magnetômetros são unidos para medir essa intensidade, um eixo cada, é possível determinar a direção e o sentido do campo magnético (MOURA, 2013). Esses sensores são utilizados em pesquisas para verificar a intensidade do campo magnético terrestre, por exemplo. É por meio desse IMU que é possível construir uma bússola digital para realizar o alinhamento de um sistema para o polo Norte/Sul magnético

do planeta.

Os magnetômetros digitais (MEMS) utilizam do Efeito Hall para medir a diferença de potencial gerada pelo caminho de passagem dos elétrons em um objeto metálico (Figura 2.33). Alguns outros sensores usam do princípio da magneto-resistência, onde alguns metais (Ferro e Níquel) alteram sua resistência elétrica quando expostos a um campo magnético (NEDELKOVSKI, 2016).

Figura 2.33 – Efeito Hall aplicado para medição de campo magnético



Fonte: Adaptado de (NEDELKOVSKI, 2016).

2.6 PROCESSAMENTO DE DADOS

Infelizmente, os dados brutos fornecidos pelos sensores IMU, descritos nas seções anteriores, precisam passar por etapas de pós processamento, seja em *hardware* ou *software*, para que estejam livres de informações não desejadas, como ruídos. Além disso, é preciso transformar os dados de aceleração, velocidade e intensidade de campo magnético para determinar corretamente os ângulos de uma plataforma no sistema RPY.

A partir do entendimento dos erros comuns em sensores IMU, é possível descrever boas práticas de análise e filtragem desses sinais. Uma abordagem secundária é realizar um pós processamento para combinar os valores de aceleração, velocidade angular e intensidade de campo magnético para que um sensor consiga compensar os erros dos demais, e obter um resultado mais preciso (WOODMAN, 2007).

2.6.1 Erros comuns em sensores IMU

Sensores Inerciais não são totalmente exatos, independentemente do seu padrão de qualidade e podem carregar erros que comprometem um sistema de navegação. É preciso investigar a fonte dos erros, e compensar-los (ZANONI, 2012).

Erros gerados por imperfeições nos sensores são determinísticos e podem ser identificados e corrigidos na sua calibração. Mas além desses, existem outros distúrbios aleatórios que devem ser levados em conta no processo: *Bias*, Ruído Branco e Efeitos de Temperatura.

- ***Bias*:** Sensores IMU possuem um erro de *offset* que é chamado de *bias*. Esse erro acompanha o valor bruto do sensor e pode gerar incongruências nas medições de ângulos. Em um giroscópio, quando a velocidade angular é integrada para calcular o ângulo de posição resultante, todos os erros ao longo do tempo acabam sendo somados e agregados no valor final (WOODMAN, 2007).
- **Ruído Branco:** esse tipo de erro é um ruído que acompanha o sinal de saída e pode ter origem no ambiente ou circuito do IMU. É um ruído que interfere em todo espectro de frequência e pode ser minimizado utilizando filtros passa-baixa, passa-faixa ou passa-alta (ZANONI, 2012).
- **Efeitos de temperatura:** Todo acelerômetro e giroscópio é suscetível a uma não linearidade no efeito *bias* devido a flutuações na temperatura. Contudo, é possível realizar calibrações para que o erro seja minimizado e isso requer um sensor de temperatura acoplado com o IMU (WOODMAN, 2007).

2.6.2 Filtragem de sinais

Um dos primeiros passos no processamento de dados é limpar o sinal original fornecido pelo sensor, que agrega erros somando sinais que se originam em frequências de leitura indesejadas. Para isso, é preciso determinar qual a frequência de oscilação máxima do sistema e cortar qualquer dado que seja lido em uma frequência superior a de corte. Se esses valores não forem eliminados, os erros podem ser somados nas próximas etapas de pós processamento, acumulando e prejudicando o correto funcionamento do sistema (YANG; BAJENOV; SHEN, 2017).

A maneira mais comum de implementar esse processamento é com um filtro passa-baixa, passa-faixa, ou passa-alta. Esses filtros são capazes de manter apenas o somatório de sinais dentro das frequências desejadas, rejeitando as demais em outras frequência.

Filtro passa-baixa é o nome dado a um circuito eletrônico, ou algoritmo de processamento, que limita a amplitude dos sinais em frequências maiores que a frequência de

corte, e mantém os demais inalterados. A atenuação para cada frequência varia de acordo com as configurações adotadas no projeto do filtro, onde quanto maior a ordem do filtro, maior será a atenuação nas frequências fora da banda passante.

Um filtro passa-altas, ao contrário, limita a amplitude dos sinais em frequências abaixo da frequência de corte. E o filtro passa-faixa é a combinação dos dois filtros anteriores, para permitir a passagem de sinais em faixas de frequências especificadas no projeto.

Por meio desses filtros, portanto, é possível minimizar o impacto do ruído branco e eliminar possíveis distorções e vibrações que são indesejadas na leitura dos sensores (YANG; BAJENOV; SHEN, 2017). A frequência de corte e a ordem do filtro devem ser determinados pelo projetista de forma a garantir que os dados desejados não sejam afetados negativamente.

2.6.3 Cálculo de *Pitch* e *Roll*

Uma vez que os sinais de acelerômetro e giroscópio tenham sido corretamente filtrados, é possível calcular os ângulos inerciais. Isso pode ser realizado de forma independente, usando os sensores de forma separada.

Uma vez que o acelerômetro esteja imóvel, e realizando a leitura somente da aceleração gravitacional, é possível calcular a posição inercial do sensor com os vetores de aceleração nos 3 eixos. Então, seja A_x, A_y e A_z os dados relativos a cada eixo, as equações (2.4) e (2.5) correspondem, respectivamente a, o cálculo dos ângulos *pitch* (θ) e *roll* (ϕ) (PATRÃO, 2015).

$$\theta_A = \arctan\left(\frac{-A_x}{\sqrt{A_y^2 + A_z^2}}\right) \tag{2.4}$$

$$\phi_A = \arctan\left(\frac{A_y}{A_z}\right) \tag{2.5}$$

O giroscópio, por outro lado, fornece os dados da taxa de variação angular dos 3 ângulos que se deseja obter. Dessa forma, a leitura dada pelo giroscópio é a derivada dos ângulos inerciais e, portanto, para calculá-los, basta realizar um somatório ao longo do tempo. Isso é descrito nas equações (2.6) e (2.7), considerando G_θ e G_ϕ os dados de cada eixo do giroscópio (PATRÃO, 2015).

$$\theta_G = \int G_\theta dt \tag{2.6}$$

$$\phi_G = \int G_\phi dt \tag{2.7}$$

2.6.3.1 Filtro Complementar

O Filtro Complementar é uma combinação linear na qual se define um peso para cada variável de entrada. Neste caso, um peso para o acelerômetro e outro para o giroscópio, em que a soma desses pesos deve ser igual a 1 (LAGE, 2016).

A fusão de sensores é a solução para a combinação das faixas de precisão de ambos os sensores, a fim de um compensar a imprecisão do outro. O giroscópio é responsável pela medição precisa em comportamentos de alta frequência e o acelerômetro pela medição em baixas frequências (OLIVEIRA WALDRI DOS S. ; GON CALVES, 2017).

Então, é preciso definir a frequência de corte e os intervalos de medição dos sensores para calcular o valor da constante associada ao peso de cada sensor no cálculo final. Assim, para obter os ângulos de rotações, *pitch* (θ) e *roll* (ϕ), usa-se as equações (2.8) e (2.10), sabendo que $K_A + K_G = 1$ (LAGE, 2016).

$$\theta = K_A \cdot \theta_A + K_G \cdot \theta_G \tag{2.8}$$

$$\phi = K_A \cdot \phi_A + K_G \cdot \phi_G \tag{2.9}$$

Onde, seja f_d a frequência de divisão desejada para os dados de acelerômetro e giroscópio e dt o intervalo entre as amostra de dados:

$$K_G = \frac{\frac{1}{f_d}}{\frac{1}{f_d} + dt} \tag{2.10}$$

2.6.4 Magnetômetro como uma Bússola

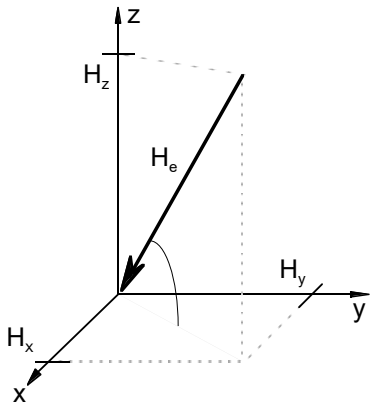
O campo magnético terrestre pode ser lido, em qualquer posição no planeta, como um vetor que aponta em direção ao polo norte magnético. Um magnetômetro pode ler os valores relativos aos seus 3 eixos e determinar o sentido do campo, como demonstrado na Figura 2.34. A intensidade desse campo pode ser determinada pela equação (2.11) (HONEYWELL, 1995).

$$H_e = \sqrt{H_x^2 + H_y^2 + H_z^2} \tag{2.11}$$

Todo sistema de orientação com o campo magnético terrestre requer um sensor magnetômetro triaxial em conjunto com um sistema de medição inercial (Figura 2.35). O cálculo posicional da orientação magnética combina os dados dos 3 eixos de medição.

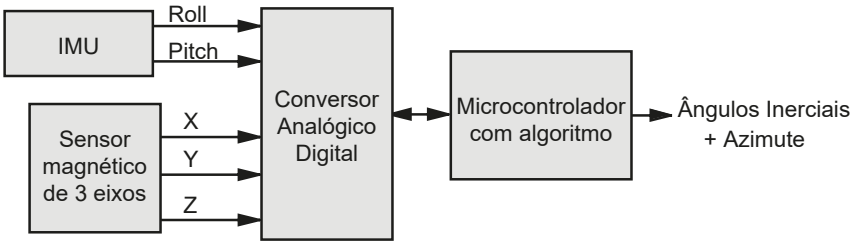
Em uma situação de perfeitas condições, onde o sensor está alinhado com o plano horizontal e sem ruídos, é possível calcular o ângulo azimutal ψ pela equação (2.12),

Figura 2.34 – Vetor de direção do campo magnético



Fonte: Adaptado de (HONEYWELL, 1995).

Figura 2.35 – Diagrama de leitura dos dados inerciais com magnetômetro



Fonte: Adaptado de (CARUSO, 2000).

onde Y_h e X_h são os componentes horizontais da medição de campo magnético com o magnetômetro, como ilustra a figura 2.36. O plano horizontal é definido pelo momento onde os dados de *pitch* (θ) e *roll* (ϕ) são nulos (CARUSO, 2000).

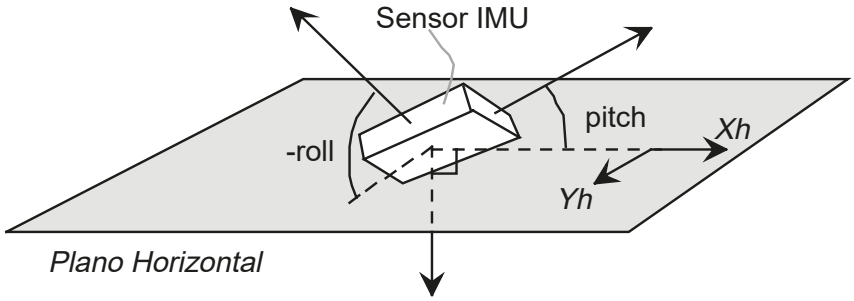
$$\psi = \arctan\left(\frac{Y_h}{X_h}\right) \tag{2.12}$$

Em uma situação diferente, onde a plataforma se encontra inclinada (Figura 2.36), é preciso decompor e planificar os vetores. Isso é feito com base nos dados de *pitch* e *roll* obtidos pelo acelerômetro e giroscópio. Após, reaplica-se os valores na equação (2.12) para determinar o azimute (CARUSO, 1997). As equações (2.13) e (2.14) demonstram a planificação para cada componente horizontal, onde: M_x, M_y, M_z são os valores de campo magnético, para cada eixo, aferidos com o uso do magnetômetro.

$$X_h = M_x\cos(\theta) + M_y\sin(\phi)\sin(\theta) - M_z\cos(\theta)\sin(\phi) \tag{2.13}$$

$$Y_h = M_y\sin(\theta) + M_z\sin(\phi) \tag{2.14}$$

Figura 2.36 – Planificação dos dados de um sistema inclinado



Fonte: Adaptado de (CARUSO, 2000).

2.6.4.1 Distorções do campo magnético

A precisão de sensores magnetômetros pode ser afetada pela presença de materiais ferrosos na proximidade, ou qualquer perturbação do campo magnético gerada por sistemas elétricos próximos. Para que o sensor consiga realizar uma leitura correta, é necessário calcular o quanto de perturbação está afetando a leitura do sensor.

Quando um componente ferroso é posicionado em um campo magnético uniforme, isso gera uma perturbação denominada *hard-iron*. Porém, alguns materiais não magnéticos como ferro e níquel geram um erro chamado de *soft-iron*. Ambos afetam o campo magnético de forma diferente, e precisam ser compensados com estratégias de calibração, conforme à equação (2.15), onde: Mn é uma matriz de erros de desalinhamento na montagem entre os eixos do sensor e do objeto; S_x, S_y, S_z são fatores de escala; SI é a matriz composta de constantes para remover o erro *soft-iron*; R_x, R_y, R_z são os valores crus do sensor, e O_x, O_y, O_z são os *offsets* causados pela interferência *hard-iron* (KUNCAR; SYSEL; URBANEK, 2016).

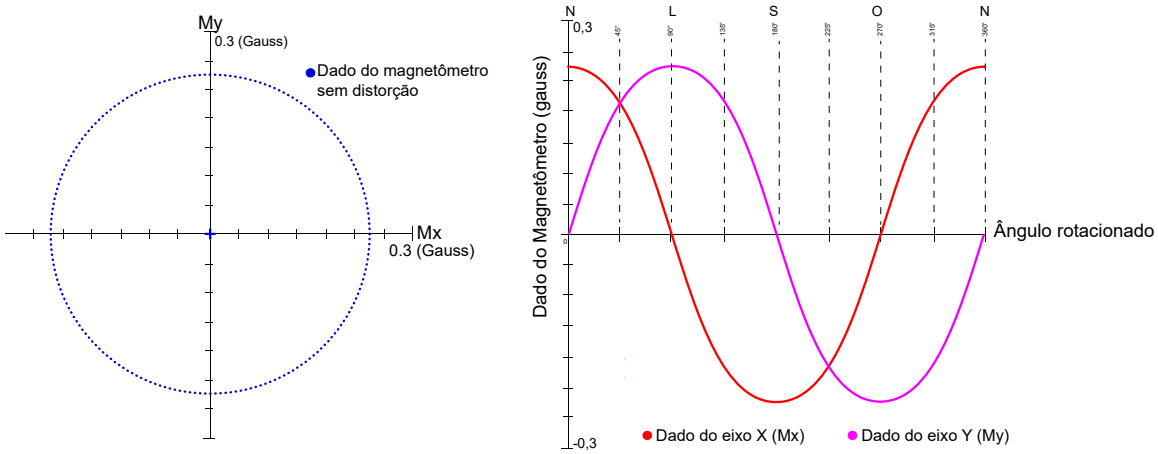
$$\begin{bmatrix} M_x \\ M_y \\ M_z \end{bmatrix} = [Mn]_{3x3} \cdot \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & S_z \end{bmatrix} \cdot [SI]_{3x3} \cdot \begin{bmatrix} R_x + O_x \\ R_y + O_y \\ R_z + O_z \end{bmatrix} \tag{2.15}$$

Todas as constantes de calibração devem ser obtidas antes de fazer uso do sensor. Porém, os métodos para encontrar as constantes e aplicá-las podem divergir. Para este trabalho, concentrou-se em aplicar as calibrações no plano horizontal, pois, na prática, utilizar uma calibração nos 3 eixos poderia ser demasiado complexo não só para o controlador embarcado mas também implicaria em mais etapas de calibração e uma maior complexidade de usabilidade para o usuário final.

2.6.4.2 Calibração contra erros hard e soft-iron

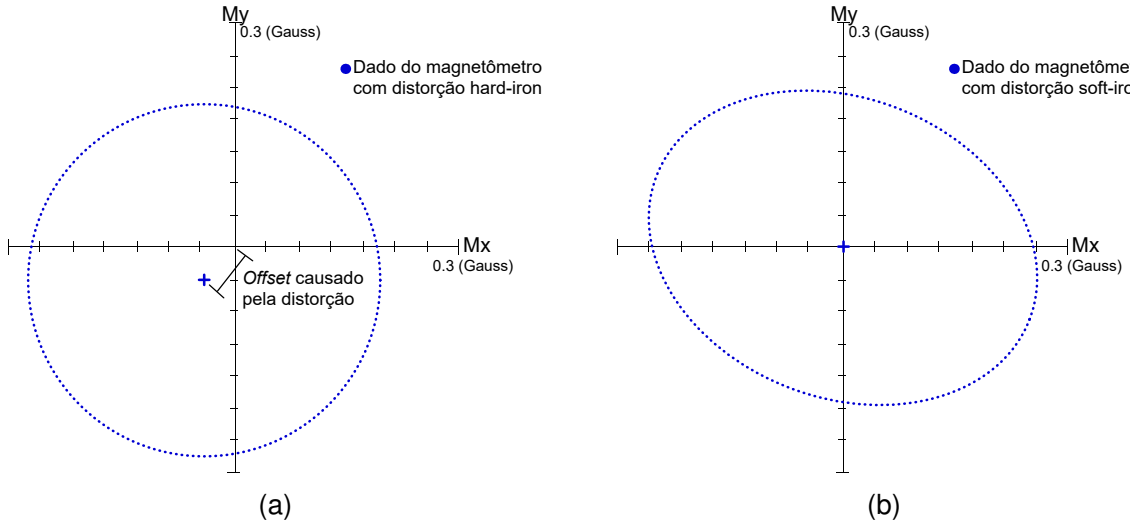
Quando um conjunto de medições é realizar rotacionando um magnetômetro $360°$, é possível obter um mapa planificado do campo magnético ao redor do sensor. Na Figura 2.37 é visível um campo sem interferências, formando um círculo com os dados coletados. Quando uma interferência *hard-iron* ocorre, o campo magnético é alterado com um *offset* (Figura 2.38a) que é simples de ser detectado por um controlador. Diferentemente, a interferência *soft-iron* manipula o campo e distorce o círculo, formando uma elipse (Figura 2.38b). Quando ambas as distorções são combinadas, o campo é lido como uma elipse fora de centro (KONVALIN, C., 2009).

Figura 2.37 – Leitura de dados ideal de um magnetômetro ao rotacioná-lo $360°$



Fonte: Adaptado de (VECTORNAV, c2021).

Figura 2.38 – Impacto das interferências *hard-iron* (a) e *soft-iron* (b) na leitura do magnetômetro em rotação de $360°$



Fonte: Adaptado de (VECTORNAV, c2021).

O método mais simples de ser implementado é compensando somente a distorção *hard-iron*. Para isso, é preciso determinar os *offset* da distorção O_x, O_y, O_z medindo a distância da extremidade ao centro do círculo, para cada eixo. Isso é demonstrado respectivamente nas equações (2.16) e (2.17), onde os valores mínimos e máximos são os menores e os maiores valores obtidos da leitura crua do magnetômetro, enquanto é rotacionado 360° (KONVALIN, C., 2009).

$$O_x = \frac{x_{max} + x_{min}}{2} \tag{2.16}$$

$$O_y = \frac{y_{max} + y_{min}}{2} \tag{2.17}$$

Porém, esse algoritmo não elimina a distorção *soft-iron*. Para isso, pode-se usar os mesmos valores mínimos e máximos mencionados anteriormente. E, a partir deles, trabalhando somente no plano horizontal, é possível determinar os fatores de escala S_x e S_y, além dos *offsets* O_x e O_y, como demonstram as equações (2.18), (2.19), (2.20) e (2.21). Os fatores de escalas, se calculados menores que 1, devem ser aproximados para 1 (CARUSO, 1997).

$$S_x = \frac{(y_{max} - y_{min})}{(x_{max} - x_{min})} \tag{2.18}$$

$$S_y = \frac{(x_{max} - x_{min})}{(y_{max} - y_{min})} \tag{2.19}$$

$$O_x = \left[\frac{(x_{max} - y_{min})}{2} - x_{max} \right] \cdot S_x \tag{2.20}$$

$$O_y = \left[\frac{(y_{max} - y_{min})}{2} - y_{max} \right] \cdot S_y \tag{2.21}$$

Dessa maneira, apenas com foco no plano horizontal, e desconsiderando erros de desalinhamento na montagem, a equação (2.15) é simplificada em (2.23) e (2.23); a matriz de compensação do erro *soft-iron* é substituída somente pelos fatores de escala. E, assim, o azimute pode ser determinado pela equação (2.24) .

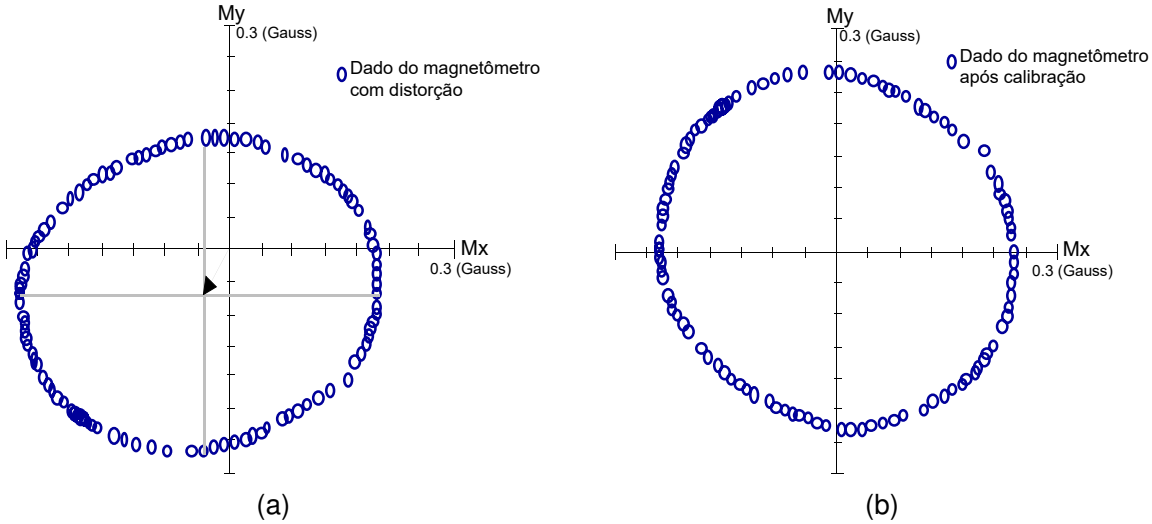
$$\begin{bmatrix} M_x \\ M_y \end{bmatrix} = \begin{bmatrix} S_x & 0 \\ 0 & S_y \end{bmatrix} \cdot \begin{bmatrix} R_x + O_x \\ R_y + O_y \end{bmatrix} \tag{2.22}$$

$$\begin{bmatrix} M_x \\ M_y \end{bmatrix} = \begin{bmatrix} S_x \cdot (R_x + O_x) \\ S_y \cdot (R_y + O_y) \end{bmatrix} \tag{2.23}$$

$$\psi = \arctan\left(\frac{M_y}{M_x}\right) \tag{2.24}$$

Esse algoritmo já foi aplicado por (CARUSO, 1997) no processo de calibração de magnetômetros dentro de veículos, para remover a distorção provocada pelos materiais do carro e do motor. A Figura 2.39 demonstra a efetividade desse procedimento.

Figura 2.39 – Aplicação da calibração: (a) sinal original lido, (b) sinal lido após aplicação da calibração



Fonte: Adaptado de (CARUSO, 1997).

Por fim, é crucial cuidar a instalação do sensor, que deve estar o mais afastado possível de motores, materiais magnéticos e ferrosos; principalmente ferro e níquel. Se isso não for possível, que seja determinada a distorção gerada pelos componentes para realizar o *offset*. Porém, mesmo com calibração, não é possível eliminar a distorção de medições em função de campos magnéticos oscilantes (CARUSO, 2000).

E, além disso, ressalta-se que as distorções magnéticas ocorrem conforme o sistema é posicionado em locais diferentes, e cada local deve ter sua própria calibração. Se o sensor é reinstalado em outra orientação no mesmo local, então também será necessário uma nova calibração. Somente após todas essas etapas é possível identificar o correto sentido do norte magnético terrestre (CARUSO, 2000).

2.7 PROTOCOLOS DE COMUNICAÇÃO SERIAL

O sensores e atuadores explicados nas seções anteriores podem se comunicar com os controladores através de protocolos de comunicação específicos. Esses modelos funcionam como uma linguagem para acessar as informações, sendo fundamentais para a diagramação do sistema eletrônico em *hardware* e *software*. Por meio dessas redes, um controlador consegue realizar a leitura dos dados armazenados no registrador de um sensor, por exemplo, ou escrever uma nova calibração em outro local da memória (VALDEZ

JONATHAN; BECKER, 2015).

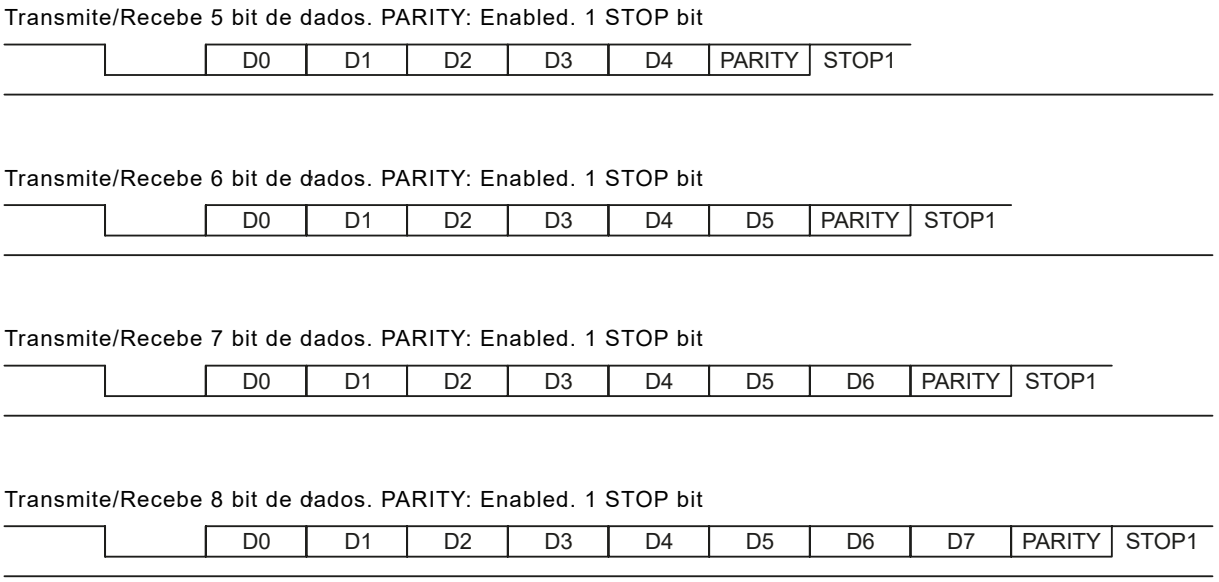
2.7.1 UART

O protocolo UART (*Universal Asynchrounous Receiver/Transmiter*) é uma forma de comunicação baseada em padrões industriais. Nesse protocolo há somente duas linhas: receptor (RX) e transmissor (TX) que utilizam de uma taxa de transmissão referência (*baud rate*) para que haja o correto processamento bit a bit do que é comunicado via protocolo (TEXAS INSTRUMENTS, 2010).

As mensagens enviadas possuem limite na velocidade que depende do *clock* do controlador que está sendo utilizado. Além disso, baseado nas configurações dos dispositivos, as mensagens são enviadas no seguinte pacote (Figura 2.40):

- 1 *START bit*
- 5, 6, 7, ou 8 *data bits*
- 1 *PARITY bit* (opcional)
- 1, 1.5, ou *2 STOP bits*

Figura 2.40 – Exemplos de comunicação UART



Fonte: Adaptado de (TEXAS INSTRUMENTS, 2010).

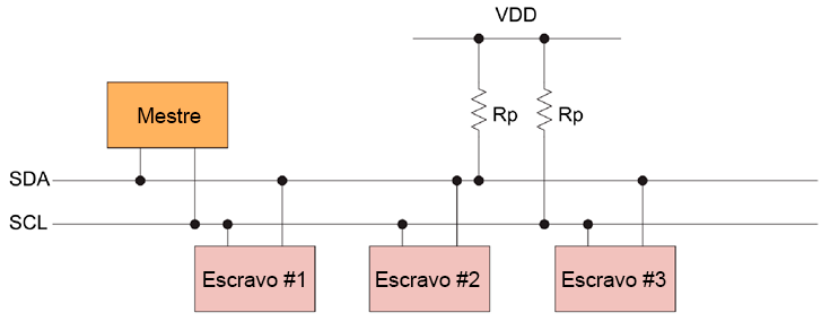
Os pinos de comunicação UART são conectados ligando-se o pino RX de um dispositivo no TX do outro. O projeto do *hardware* dessas linhas de comunicação é um pouco

mais livre, mas recomenda-se manter a indutância tão baixa quanto possível, além de rotear as trilhas de comunicação afastadas das fontes de ruído (PETERSON, 2020).

2.7.2 I2C

A rede I2C (*Inter-Integrated Circuit*) é um protocolo padrão e popular, usado para comunicação entre mestre(s) e escravo(s) (Figura 2.41). Para a comunicação ocorrer, o mestre deve ordenar quando e o que será transmitido pelos escravos. Nesse procedimento, o mestre sempre especifica o endereço do escravo que ele deseja ler ou escrever alguma informação nos seus registradores, esses endereços são um valor de 0 a 127 e dois escravos não devem partilhar o mesmo valor (VALDEZ JONATHAN; BECKER, 2015). A rede permite diversos modos de comunicação, que se diferenciam pela velocidade da transmissão de dados. Os mais comuns são de 100 kbit/s (*Standard Mode*), 400 kbit/s (*Fast Mode*) ou 3.4 Mbit/s(*Fast Mode Plus*). Contudo, nem todos os dispositivos que aceitam o protocolo I2C irão suportar todos os modos obrigatoriamente.

Figura 2.41 – Topologia de uma rede I2C com 1 mestre e 3 escravos conectados às linhas SDA e SCL, ambas com resistores de *pull-up*



Fonte: Adaptado de (AFZAL, c2021).

Para funcionar, o protocolo I2C precisa de duas linhas: *Serial Data* (SDA) e *Serial Clock* (SCK ou SCL) (Figura 2.41). Essas linhas precisam estar conectadas com a alimentação do sistema, que deve ser de 5V, por meio de um resistor *pull-up*. O valor mínimo dele é calculável com a equação (2.25), considerando V_{OL} e I_{OL} a tensão e corrente máxima para que os circuitos consigam realizar a leitura de um nível lógico válido (ARORA, 2015).

$$R_P(min) = \frac{V_{CC} - V_{OL}(max)}{I_{OL}} \tag{2.25}$$

Contudo, existe um valor limite para o resistor, que é determinado pelo total da capacitância entre as linhas. Esse fator altera o tempo de alternância entre um nível lógico alto e baixo, que deve estar dentro de padrões da rede. Esse tempo máximo depende da taxa de transmissão, ou modo da rede. Assim, $t_r = 1000ns$ (*Standard Mode*), $t_r = 300ns$

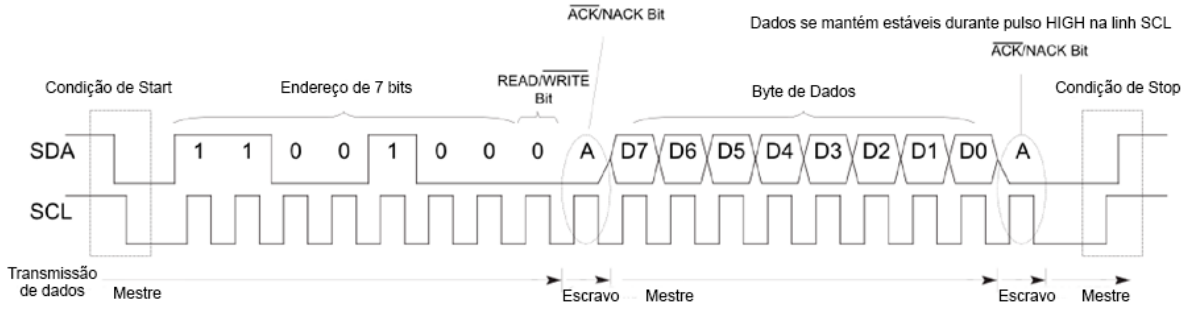
(*Fast Mode*) e $t_r = 120ns$ (*Fast Mode Plus*). Esse valor máximo é determinado pela equação (2.26) (ARORA, 2015).

$$R_P(max) = \frac{t_r}{0,8473 \times C_b} \tag{2.26}$$

A transmissão de dados se inicia quando a rede está em *idle*, ou seja, ambas as linhas devem estar em nível lógico alto após uma condição de STOP (nível lógico baixo). Dessa maneira, considerando dois cenários possíveis, a comunicação pode ser prosseguida das seguintes formas: o mestre deseja escrever ou alterar dados no escravo; o mestre deseja ler dados de um escravo (VALDEZ JONATHAN; BECKER, 2015).

1. Para o primeiro caso (Figura 2.42):
    - Mestre transmite uma condição de *START* e envia o endereço do escravo com quem quer se comunicar;
    - Mestre envia os dados;
    - A transferência se encerra com um sinal de *STOP* do mestre.
2. Para o segundo caso (Figura 2.43):
    - Mestre transmite uma condição de *START* e envia o endereço do escravo com quem quer se comunicar;
    - Mestre solicita uma leitura de dados no registrador do escravo;
    - Escravo envia os dados para o Mestre;
    - Mestre encerra a transmissão com sinal de *STOP*.

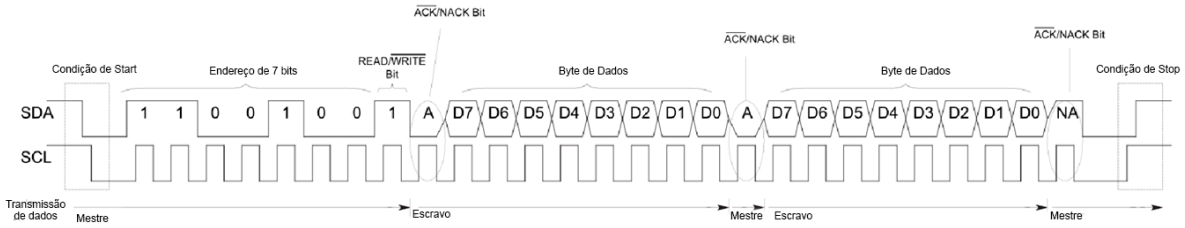
Figura 2.42 – Transmissão de dados do mestre para o escravo



Fonte: Adaptado de (AFZAL, c2021).

As condições de *START* e *STOP* são descritas sempre pelo mestre. Essas duas condições são situações específicas na rede, pois são o único momento da transmissão

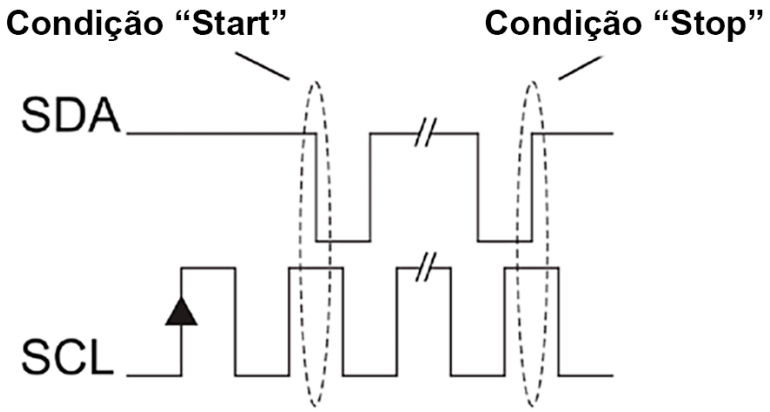
Figura 2.43 – Requisição de dados do mestre ao escravo



Fonte: Adaptado de (AFZAL, c2021).

em que a linha SDA é alterada enquanto o *clock* na linha SCK está em nível lógico alto. No caso de uma situação de *START*, a linha SDA alterna do nível lógico alto para o baixo, e o inverso ocorre na situação de *STOP* (Figura 2.44) (AFZAL, c2021).

Figura 2.44 – condições de *START* e *STOP* no padrão I2C



Fonte: Adaptado de (AFZAL, c2021).

Para cada conjunto de dados enviados, um bit ACK (*acknowledgement*) é enviado na sequência — lido como um nível lógico baixo — pelo receptor da mensagem, comunicando o correto recebimento e permitindo uma nova transmissão. Esse bit é enviado imediatamente após a comunicação, e o transmissor deve liberar a linha SDA para que o receptor possa enviar a mensagem (VALDEZ JONATHAN; BECKER, 2015).

Devido ao controle de impedância e capacitância, recomenda-se que o roteamento físico desses sinais, em uma placa de circuito, não seja longo. Quanto maior a velocidade de comunicação, menor deve ser a distância entre os dispositivos I2C, afastando-os também de sistemas que possam causar interferência eletromagnética (PETERSON, 2020).

2.7.3 Bluetooth

Bluetooth é um protocolo de comunicação sem-fio e de curto alcance. Esse sistema é uma tecnologia adequada para sistemas embarcados por apresentar um baixo custo e consumo, provendo fácil conectividade, sobretudo nas versões mais recentes do protocolo (TAKAHASI, 2003).

O fator baixo consumo aumenta a economia de energia e limita o alcance da conexão a depender da classe do dispositivo. Para um dispositivo Classe 3, com 1 mW de potência, o alcance será de até 1 m; classe 2, com até 10 mw, alcance de 10m; classe 3, com até 100 mW, total de 100 m (TIER, 2019).

A tecnologia *wireless bluetooth* especifica um modelo de comunicação síncrono entre somente um par de dispositivos com a banda de 2.4 GHz a 2.5 GHz dividida igualmente entre eles. Esse padrão já é geralmente usado para sistemas de localização, transferência de áudio e dados de forma geral (MULLER, 2000).

Nesse protocolo, cada dispositivo possui um endereço MAC (*Media Access Control* ou Controle de Acesso de Mídia) que distingue-os, sendo gravado fisicamente no *hardware*. Aquele que solicita a comunicação passa a ser o mestre de uma relação mestre/escravo, que também é *full-duplex* (TAKAHASI, 2003).

Quando dois ou mais dispositivos se conectam via *Bluetooth*, um deles será o mestre e será formada uma *Bluetooth Wireless Personal Area Network* (BT-WPAN). Será classificada como *piconet* para até 8 dispositivos conectados e *scatternet* caso duas redes *piconet* se comuniquem por meio de um dispositivo escravo de cada uma (TIER, 2019).

2.7.3.1 Bluetooth Low Energy

Bluetooth Low Energy (BLE) é uma versão da tecnologia otimizada para baixo consumo e é atualmente muito utilizado em *smartphones*. Para que a tecnologia funcione, otimizando a comunicação, é definido um protocolo para diferentes perfis com atributos genéricos de dispositivos (GATT do inglês *Generic Attribute Profile*). Os perfis GATT correspondem a serviços e características (BLUETOOTH, 2019).

Os serviços servem para caracterizar os conjuntos de dados, por exemplo um sensor de batimento cardíaco. A característica do perfil define uma propriedade ou configuração da informação. Em uma analogia, o GATT é como entrar em uma sala de armazenamento com vários armários e cada armário com várias gavetas. Nesse caso, os serviços são os armários e as gavetas são as características com as informações dos dispositivos (DEVZONE, 2018).

Na comunicação, todas essas informações são identificadas por um código único, ou *Universally Unique ID* (UUID). Esse é um valor que identifica o tipo de dispositivo com o qual um *smartphone*, por exemplo, está se comunicando, e descreve todas as informa-

ções mencionadas anteriormente. Existem padrões UUID de 16 bits como para o caso do sensor de batimento cardíaco. Mas para dispositivos que não seguem um padrão, existem UUIDs de 128 bits que servem para caracterizar serviços específicos em aplicações únicas (DEVZONE, 2018). Por meio desses UUIDs que um dispositivo consegue reconhecer qual o tipo de dado que está sendo comunicado, e assim o protocolo pode se otimizar para melhor atender à demanda da comunicação.

2.8 INTERFACE GRÁFICA

Para que um sistema embarcado se torne amigável para um usuário comum, é preciso uma interface gráfica que sintetize as informações e forneça formas de controle do usuário para a plataforma. Exemplos de interfaces podem ser simples botões, LEDs ou também um aplicativo para *smartphone*. Contudo, independente do nível de complexidade, toda interface deve se ater a princípios em sua elaboração, para que se tornem entendíveis pelo maior número possível de usuários.

2.8.1 Princípios e Diretrizes

Os princípios e as diretrizes comumente utilizados em interfaces humano-computador giram em torno dos seguintes tópicos: correspondência com as expectativas dos usuários; simplicidade nas estruturas das tarefas; equilíbrio entre controle e liberdade do usuário; consistência e padronização; promoção da eficiência do usuário; antecipação das necessidades do usuário; visibilidade e reconhecimento; conteúdo relevante e expressão adequada; e projeto para erros (BARBOSA et al., 2021). Esse conjunto de princípios são conhecidos como heurística de Nielsen, pois são aplicáveis em qualquer sistema, independente de casos específicos.

2.8.1.1 Visibilidade dos status do sistema

O sistema deve sempre manter o usuário atualizado sobre as condições de operação com uma taxa de atualização condizente para a informação. Ao informar o status da bateria, por exemplo, o usuário do *smartphone* consegue predizer quanto tempo de uso ainda terá e irá conseguir manejar sua interação com base nessa previsibilidade (NIELSEN, 2020).

2.8.1.2 Comunicar-se com o mundo real

O Projeto tem que se comunicar com o usuário na língua do usuário. Se um brasileiro não sabe inglês, ele "ficará perdido"nos Estados Unidos. Da mesma forma, o desenvolvedor não pode assumir que o usuário entenderá o aplicativo somente pelo fato do desenvolvedor ter feito algo que ele próprio entenda. É sempre recomendado conferir a linguagem do sistema com um conjunto grande de pessoas para evitar mal entendidos.

Quando o usuário não entende a língua do sistema, ele se sente afastado e irá deixar de usar a plataforma. É interessante que a plataforma tenha *designs* semelhantes com objetos do mundo real, dessa forma, o usuário se sente "contemplado"e consegue facilmente fazer a conexão entre o mundo real e a plataforma (KALEY, 2018).

2.8.1.3 Liberdade de Controle do Usuário

Por vezes, a pessoa que está realizando um processo em um sistema pode cometer um engano. Esse evento pode levar a situações de erro que não devem comprometer a experiência. Por isso, os usuários precisam de uma “saída de emergência” claramente marcada para sair do estado indesejado. Isso reduz a sua ansiedade e o medo de errar, pois ele sabe que os erros podem ser contornados (BARBOSA et al., 2021).

2.8.1.4 Consistências e Padrões

É importante que o sistema mantenha uma consistência entre suas telas, ou mesmo em grandes plataformas, ou seja, que os múltiplos programas tenham o mesmo padrão, com funções localizadas no mesmo lugar, com nomes similares e com um *disign* similar. Exemplo disso são as telas dos aplicativos do Google Docs: todos possuem o mesmo estilo de menu. Idem para o Microsoft Office.

A consistência também se estende aos ícones. O ícone que representa um botão, por exemplo, é importante que seja consistente em estilo com os demais. Eles podem ser mais preenchidos, *clean*, neutros ou suaves. O que importa nesse caso, é que sejam todos padronizados (HARLEY, 2014).

2.8.1.5 Prevenção de erros

Uma forma de prevenção é oferecer sugestões numa caixa de pesquisa, por exemplo. Em situações de rotina, como disparar um lembrete, a tela de criação pode oferecer

uma sugestão padrão de um modelo que faça sentido para o usuário. Para evitar corrupção de dados pelo usuário durante o cadastro, é possível sugerir ao usuário o preenchimento de números de forma truncada, fazendo o pós-processamento para ler o número corretamente.

2.8.1.6 Relembrar o usuário é mais fácil do que o usuário relembrar

Quando o usuário precisa repensar sobre algo incomum na memória, ele despende muito tempo. Então, quando a plataforma exige uma lembrança do usuário para entender algo, isso limita a experiência e incorre em perda de tempo ou confusão.

Por isso, é mais interessante realizar a exigência com uma possível sugestão de resposta correta. A recognição de algo é muito mais prático para a mente humana, pois ao mostrar para o cérebro algo relacionado com o que precisa lembrar-se, dispara-se a memória de forma mais efetiva. Dar uma pista para o cérebro é mais eficiente do que simplesmente perguntar sem oferecer nada (BUDIU, 2020).

2.8.1.7 Torne o sistema flexível e eficiente

Atalhos, personalização e customização. Com esses fatores é possível melhorar a usabilidade para aqueles que não são mais novatos no *software* e isso ajuda a manter esses usuários ativos. Um fotógrafo experiente, que está acostumado com os atalhos de teclado nos aplicativos da Adobe, teria muita dificuldade se o teclado viesse a falhar, pois a mente já assimilou os atalhos mais usados e eles fazem diferença na velocidade com que o profissional interage com o *software* (LAUBHEIMER, 2020).

2.8.1.8 Tenha um projeto minimalista

Um projeto é minimalista significa usar elementos simples num arranjo onde desenho e a interface combinem de forma agradável sem chamar a atenção de forma desnecessária, colaborando com que o usuário foque somente naquilo que é necessário (NIELSEN, 2020).

2.8.1.9 Ajude o usuário a entender e se recuperar de erros

O usuário precisa entender quando o sistema não está funcionando e como fazê-lo voltar à normalidade. As mensagens de erro devem ser expressas de uma forma simples, indicando o possível problema e a solução. Cores vermelhas e pretas ajudam a demonstrar o sinal de erro para o usuário (LAUBHEIMER, 2015).

2.8.1.10 Tire dúvidas e documente o sistema

Existem duas formas de ajudar o usuário e tirar suas dúvidas. A primeira é de forma proativa, onde a aplicação guia o usuário para se familiarizar com a interface. Outra forma é por uma seção com perguntas e respostas, a qual ajuda os usuários a se tornarem mais independentes com a aplicação, resolvendo seus próprios problemas e filtrando os casos que precisam de suporte para a equipe técnica da plataforma (JOYCE, 2020).

2.8.2 Android

Para desenvolver o aplicativo de controle que o usuário final irá utilizar no processo de alinhamento da plataforma, será usado o *framework Android*, da *Google*.

2.8.2.1 Ambiente de Desenvolvimento

O *Android Studio* é o ambiente de desenvolvimento integrado oficial para a criação de aplicativos *Android* e é baseado no *IntelliJ IDEA*. Ele oferece uma série de Recursos que possibilitam a confecção de um aplicativo: Sistema de compilação flexível baseado em *Gradle*; Um emulador rápido com suporte a vários recursos; ambiente unificado que possibilita o desenvolvimento para qualquer dispositivo *Android*, incluindo relógios e televisões; integração com *GitHub* para *backup* e documentação do código; entre outras funções que possibilitam analisar o desempenho de um aplicativo em tempo real, bem como fazer *updates* (GOOGLE DEVELOPERS, 2021).

2.8.2.2 Linguagens de Programação

Existem soluções de desenvolvimento *Android* mais *user-friendly* como *APP Inventor* ou *Kodular*, porém, essas interfaces não garantem ao desenvolvedor um pleno controle

do aplicativo, e muitas vezes acabam limitando o projeto da interface. Por isso, usar linguagens de programação nativas é uma abordagem mais interessante para aplicativos mais completos. É possível criar aplicativos com diversas linguagens, mas somente duas são nativas e permitem realizar aplicações que podem usar de todo o poder de processamento de um *smartphone*: Java e *Kotlin*.

Em 2017, *Kotlin* foi definido pela Google como sendo a principal linguagem de desenvolvimento *Android*. Ela é muito mais nova que Java, sendo desenvolvida pela JetBrains. A grande motivação de se usar *Kotlin* para o desenvolvimento reside no fato de ser uma linguagem segura para prevenção de objetos nulos, operando em paralelo com qualquer código em Java e dando opções de co-rotinas. Além disso, ao comparar dois códigos com a mesma função, um escrito em Java e o outro em *Kotlin*, o segundo pode ser até 40% mais compacto, o que implica em uma linguagem mais concisa e compreensível entre desenvolvedores. A desvantagem de se usar *Kotlin*, para este trabalho, é somente a falta de uma comunidade grande, comparando com Java, o que limita o suporte para eventuais problemas de desenvolvimento (REDKA, 2021).

Dentro do ambiente de desenvolvimento usa-se também linguagem de arquivos XML para a criação de interfaces gráficas (*layouts*), bem como a escrita dos vetores, animações, arquivos de configuração do aplicativo e temas de *layout*. Para armazenamento de dados dentro do aplicativo, normalmente usa-se um banco de dados que é operado com códigos de consulta *SQL*.

2.8.2.3 Material Design

Existem uma série de diretrizes de projeto fornecidas pela Google para guiar o desenvolvimento de aplicativos *Android*. Essas informações são fornecidas principalmente pela biblioteca *Material Design*, que fornece pacotes facilmente implementáveis de *layouts* para aplicações responsivas e padronizadas.

A biblioteca colabora com o desenvolvedor fornecendo ícones, tipografia, cores e componentes gráficos que trazem uma imersão para o usuário de forma simples e minimalista. Os *designs* se inspiram no mundo real, facilitando a comunicação com o usuário (MATERIAL FUNDATION, 2021).

2.8.3 Usuários

O objetivo deste trabalho possui um nicho muito específico de usuários: astrofotógrafos. Dentro desse grupo de pessoas, é possível abstrair uma série de expectativas e necessidades a respeito desse sistema e, mais especificadamente, de uma interface. Para

entender o ponto de vista dos possíveis usuários é preciso criar pesquisas buscando uma coleta de dados relevantes para o projeto.

No entanto, é importante destacar que o termo "usuário" não diz respeito somente a um astrofotógrafo que realmente irá usar a plataforma. Ele pode ser não só um usuário ativo (primário), ou um ocasional (secundário), mas também pessoas entusiastas que acompanham a astrofotografia e são chamados de *stakeholders* (BARBOSA et al., 2021).

A partir da análise feita com a coleta de dados inicial, é possível determinar as exigências de *software* (e também *hardware*) que implicam na melhor usabilidade da plataforma equatorial.

3 DESENVOLVIMENTO DA PLATAFORMA

Para desenvolver a plataforma é necessário um projeto estrutural em conjunto com o projeto eletrônico da Placa de Circuito Impresso (PCI) que irá embarcar os componentes eletrônicos.

3.1 PROJETO ESTRUTURAL

Dentre as variações de modelo Barn-Door que foram abordadas na seção 2.2.1, optou-se pelo modelo de rosca curvada, que é mais simples e barato dentre as outras opções. A montagem de braço simples foi descartada pois ela requer que o motor tenha um *feedback* do ângulo da plataforma para regular sua correta velocidade, encarecendo o sistema tornando-o mais complexo. Ao contrário, a montagem de braço duplo não possui esse último problema, porém requer uma montagem mais elaborada e com mais componentes mecânicos, deixando a desejar no fator simplicidade.

3.1.1 Requisitos de Projeto

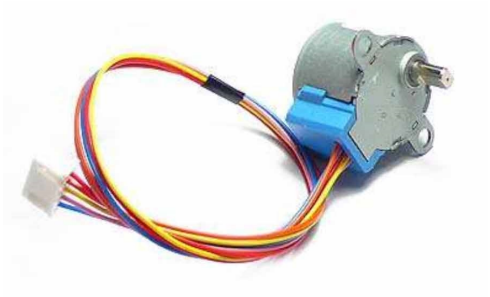
Apesar da montagem curva ter suas vantagens, ela também apresenta características estruturais que requerem atenção durante o projeto. Então, destacam-se abaixo os requisitos mais relevantes para este trabalho:

- Mover a câmera com uma velocidade angular constante de $0,2507^\circ/min$;
- Suportar pelo menos 2kg de peso total em rastreamento;
- Ter peso máximo de 1kg sem a câmera;
- Possuir vibração – em rastreamento e nos 3 eixos; inerciais – que não seja perceptível pelo sensor da câmera.

3.1.2 Motor 28BYJ-48

O motor selecionado foi o modelo 28BYJ-48 (Figura 3.1), por ser de baixo custo e funcionar com tensão de alimentação de 5V. A importância disso será abordada na próxima

Figura 3.1 – Motor de Passo 28BYJ-48



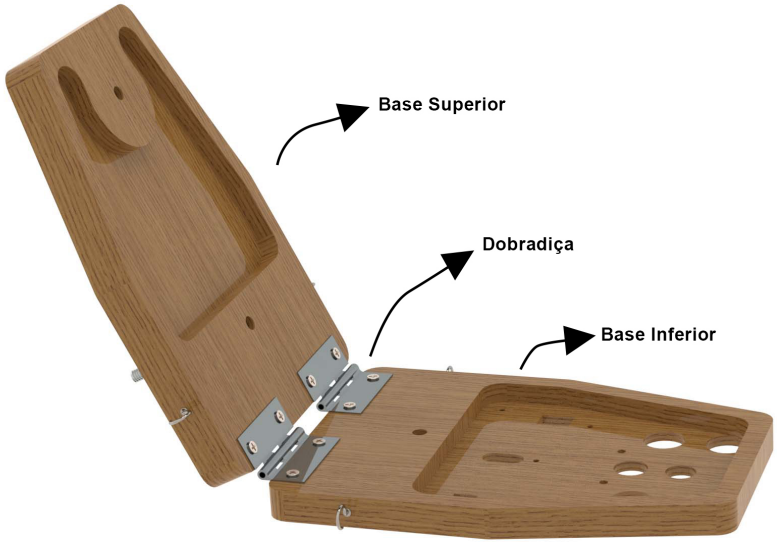
Fonte: Adaptado de(WELTEN HOLDINGS, c2022).

seção. O Motor é unipolar, possui 4 fases e 2048 passos em modo *Full Step*; o *pull in* torque é de $29,43\ mN.m$ (WELTEN HOLDINGS, c2022).

3.1.3 Estrutura Principal

A estrutura principal é a parte responsável por abrigar todos os componentes, fixar a câmera e também ser fixada no tripé fotográfico. Ela foi projetada com MDF 15 mm pois é um material economicamente mais viável que metais, e consegue prover resistência ao sistema. O projeto das peças nomeadas de base superior e base inferior se conectam com uma dobradiça conforme demonstrado na Figura 3.2.

Figura 3.2 – Encaixe das bases superior e inferior



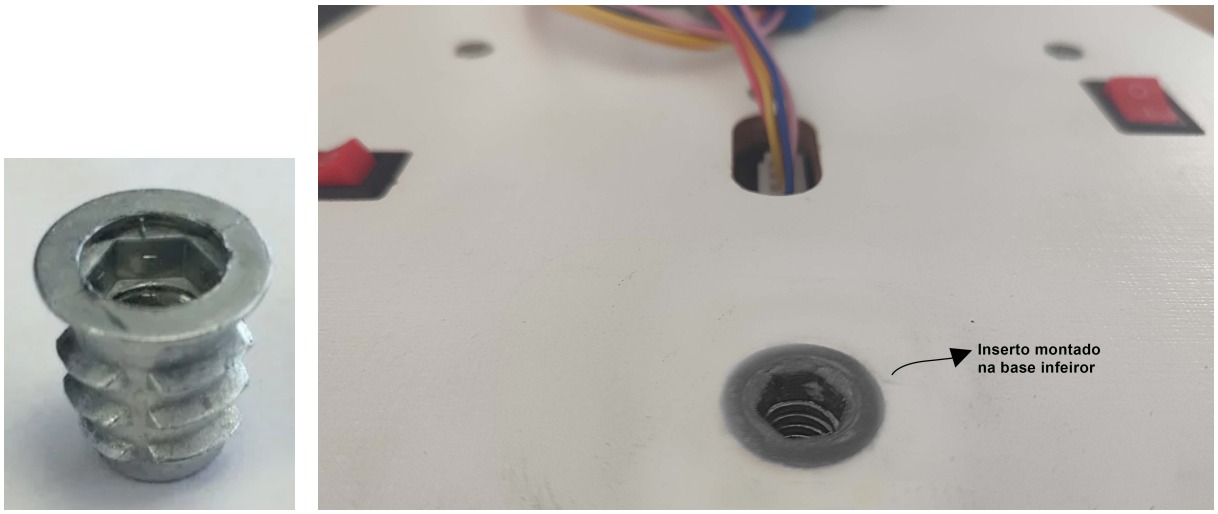
Fonte: Autor.

É possível reparar que ambas as peças possuem um rebaixamento interno, que não é passante, e tem como função alocar todos os componentes. No entanto, esse tipo de

corte é mais caro do que um passante fabricado à laser, por exemplo, e por isso sugere-se que os componentes eletrônicos sejam simplificados em projetos futuros.

A base superior fixa a câmera, e a base inferior é fixada no tripé fotográfico. Essa fixação da base inferior é realizada usando um inserto roscado para madeira de 1/4" (Figura 3.3a), que fica instalado como demonstra a Figura 3.3b. A fixação da câmera na base superior é realizada usando um *Ball-Head*. No CAD, isso foi representado usando um modelo de *Ball-Head* simplificado (Figura 3.4), porém o sistema consegue suportar qualquer modelo, uma vez que a fixação desses componentes é feita por meio de uma rosca padrão de 3/8".

Figura 3.3 – Fixação no tripé: (a) Inserto e (b) Montagem



(a) (b)

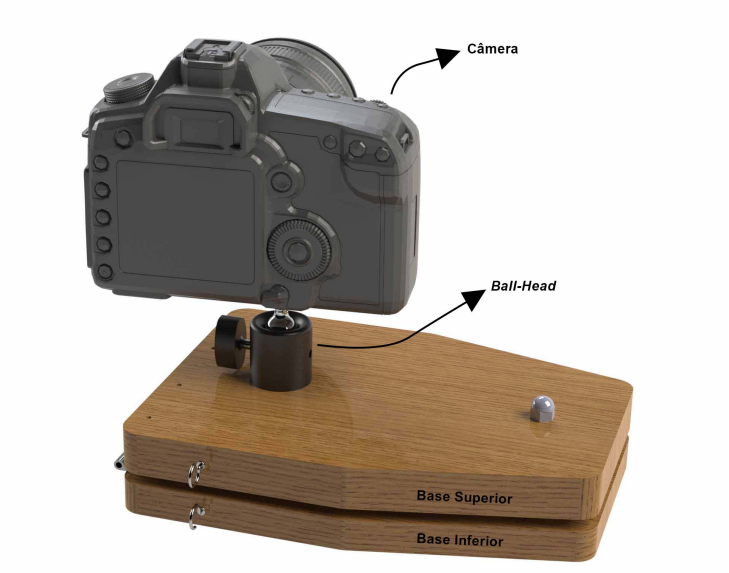
Fonte: Autor.

A estrutura e os componentes foram posicionados de forma que o projeto fosse compacto. O motor de passo é fixado na base inferior. Conecta-se em seu eixo uma engrenagem que irá transmitir o torque para movimentar a plataforma. O sistema de transmissão foi projetado para funcionar com duas engrenagens, que buscam suavizar o movimento do motor, reduzindo sua velocidade e aumentando o torque. A relação adotada no projeto foi 2:5[1], onde a engrenagem menor é montada no motor e a maior transmite momento para o eixo curvado. A Figura 3.5 demonstra a montagem da transmissão, com uma vista de corte na engrenagem menor e base da plataforma; a engrenagem menor está em verde, e a maior está em vermelho.

Com base nas dimensões extraídas dessa montagem, foi avaliado matematicamente se o motor possuirá o torque necessário para rotacionar o sistema de transmissão. Para isso, se considera o cenário onde a câmera emprega maior força contra a plataforma, que é com o sistema totalmente fechado, com a base superior nivelada. As dimensões

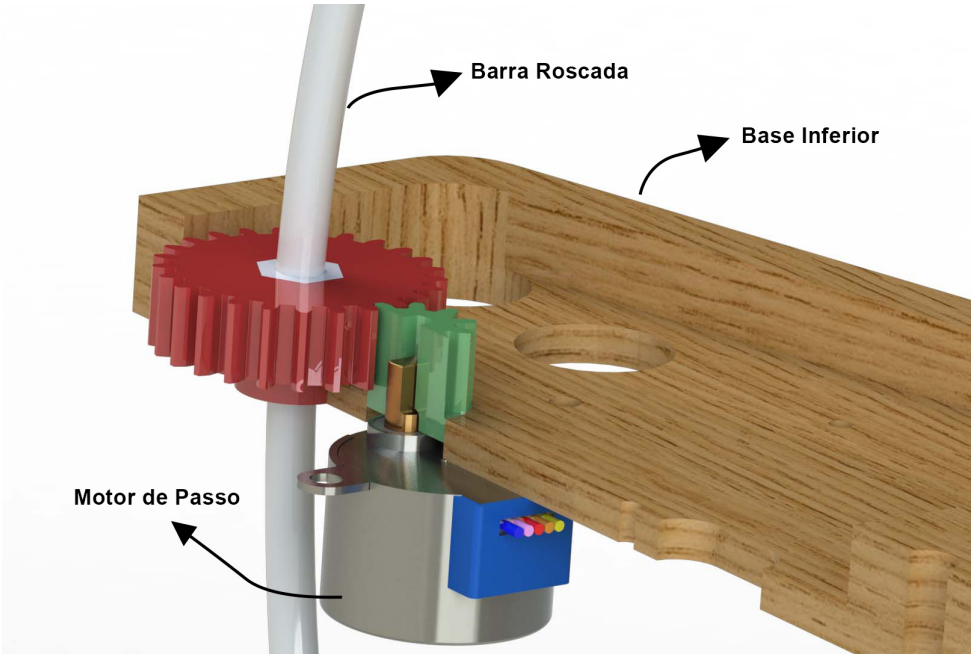
[1] A seleção de uma relação de 2:5 nas engrenagens se motivou durante a realização dos cálculos a serem explicados a seguir, para que o motor trabalhe em uma velocidade e torque adequados.

Figura 3.4 – Montagem da câmera no Ball Head



Fonte: Autor.

Figura 3.5 – *Render* que demonstra a transmissão do torque do motor para elevar a plataforma

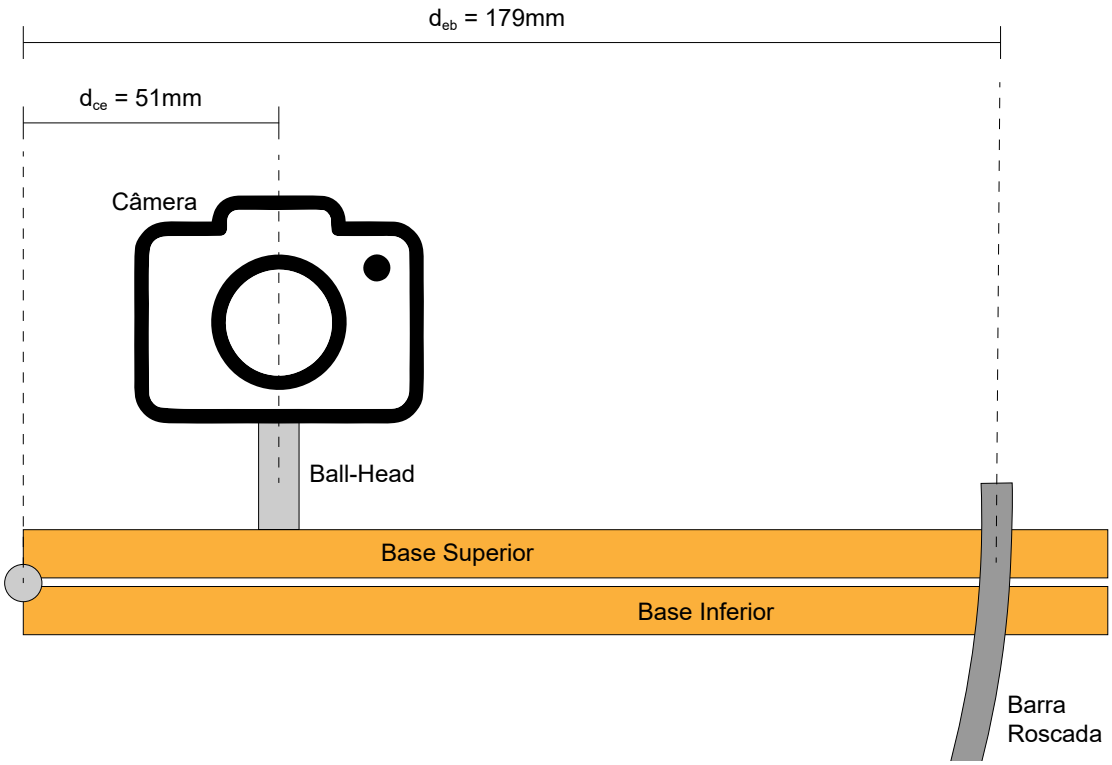


Fonte: Autor.

e variáveis usadas constam na Figura 3.6. O torque exercido no eixo curvo, pela câmera e equipamentos, é dado na equação (3.1), considerando: massa da câmera somada aos suportes e *ball-head* $m = 2,5\ kg$; aceleração da gravidade $g = 9,81\ m/s^2$; distância $d_{ce} = 51\ mm$ entre câmera e eixo de rotação da plataforma.

$$\tau_c = m\ g\ d_{ce} = 1,370\ Nm \tag{3.1}$$

Figura 3.6 – Representação da montagem com as dimensões para cálculos



Fonte: Autor.

Dessa forma, a força F_{barra} que a barra roscada deve exercer para mover a câmera é dada na equação (3.2). Considerando, também, que a distância d_{eb} entre a dobradiça e a barra roscada é de $179\ mm$.

$$F_{barra} = \frac{\tau_c}{d_{eb}} = 7,6\ N \tag{3.2}$$

Então, o torque empregado na engrenagem maior, que está em contato com a barra roscada é dado na equação (3.3). Essa equação considera o passo da rosca na barra roscada $p = 1\ mm$; e uma eficiência de rotação e igual a $0,15$ (PACIFIC BEARING COMPANY, 2017). Esse último fator deve-se pelo atrito na transmissão em uma rosca sem lubrificação, e ele pode sofrer alterações de acordo com o fabricante. Como esse valor é incerto e depende da montagem, considerou-se o menor valor obtido em catálogos de fabricantes.

$$\tau_{eng2} = \frac{F_{barra} \cdot p/(2\pi)}{e} = \frac{9,2 \cdot 0,00159}{0,1} = 8,134 mN.m \tag{3.3}$$

Após isso, considerando a relação de 2:5 adotada entre as duas engrenagens, calcula-se o torque máximo do motor, τ_{max}, com a equação (3.4). Além disso, se adici-

ona um coeficiente de segurança de 1,5 (ADVANCED MICRO SYSTEM INC., c2021a). Dessa forma, têm-se um torque máximo necessário igual a $5,85\ mN.m$, que respeita o limite de $34,3\ mN.m$

$$\tau_{max} = \tau_{eng1} = \tau_{eng2} \cdot (2/5) \cdot 1,5 = 5,85\ mN.m \tag{3.4}$$

A velocidade do motor também foi calculada com a equação (3.5). Ela considera que a rotação completa da câmera, com a plataforma, seria de $23\ h$ e $56\ min$; o raio de rotação do sistema (r_{sis}) é de $179\ mm$, medindo do eixo da dobradiça até a barra roscada; o passo da barra roscada M6 que é de $1\ mm$; e a relação entre as engrenagens, de 2:5. O valor calculado de $1,96\ rpm$ é adequado pois respeita a relação de torque máximo *versus* velocidade do motor[2]

$$V(rpm) = \frac{2 \cdot \pi \cdot 0,179/0,001}{23 \cdot 60 + 56} \cdot \frac{5}{2} = 1,96\ rpm \tag{3.5}$$

3.1.4 Engrenagens

Tendo em vista a confirmação do dimensionamento dado anteriormente, foi realizado o projeto das engrenagens. Devido à customização do projeto, optou-se por fabricá-las com impressão 3D. Dentro disso, considerando os materiais mais comuns utilizados nas impressoras (PLA, ABS e PETG), optou-se pelo PLA em função de sua resistência mecânica e baixo custo (FACUNDO, 2021).

Para a modelagem, foram calculadas as dimensões de projeto para engrenagens de dentes retos, que são adequadas para prototipagem com impressão 3D. Essas dimensões tem como base o diâmetro externo de uma das engrenagens; o ângulo de pressão, que é igual para ambas; e o número de dentes (FACUNDO, 2021).

O ângulo de pressão adotado é o padrão de mercado: $\theta = 20°$; e o diâmetro externo (d_e) da engrenagem menor foi definido tendo como base o espaço máximo que poderia ocupar na plataforma, que é de $16\ mm$. Definiu-se também que a engrenagem menor terá 10 dentes (Z) e, consequentemente, pela relação 2:5, a maior terá 25 dentes.

Além disso, módulo da engrenagem é um valor que deve ser igual em ambas, e é utilizado como parâmetro para o cálculo das demais medidas de projeto. Com a equação (3.6,) calculou-se $m = 1,33$. Com ele, define-se também que a espessura mínima das engrenagens é de $b = 8\ mm$; calculada pela equação (3.7). Dessa forma, para o projeto, foi determinado que a espessura seria de $b = 10\ mm$, encaixando na profundidade do

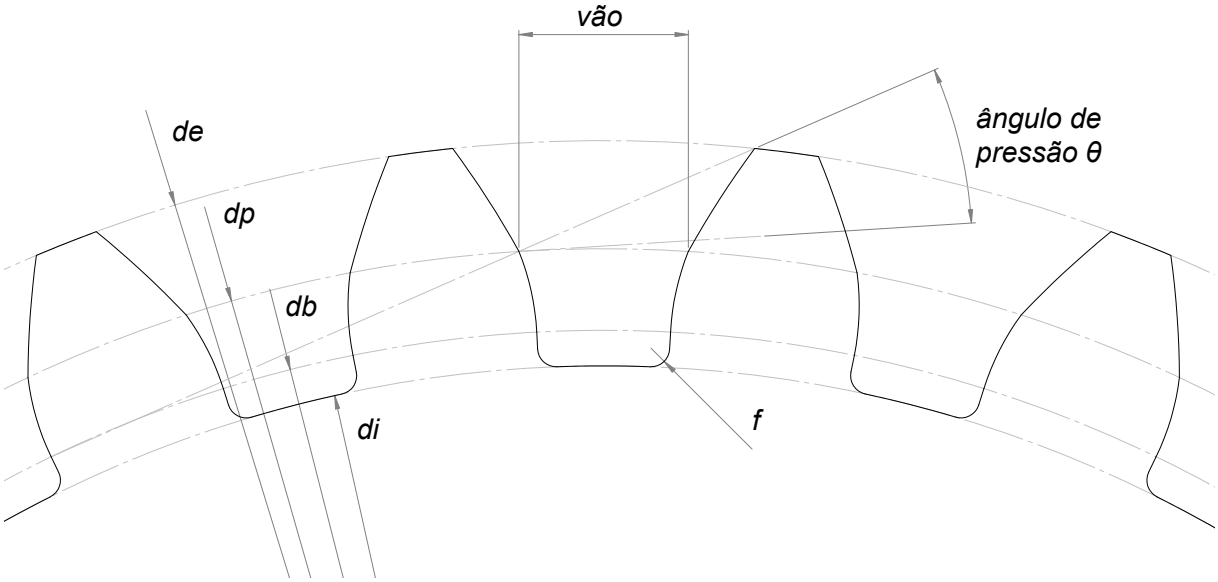
[2]Essa curva de torque pela velocidade não existe oficialmente. Em muitos fóruns essas curvas são discutidas e medidas experimentalmente. Neste *link* consta uma das curvas extraídas e que demonstram a aplicabilidade dessa relação de torque e velocidade adotada, considerando que $2\ rpm$ implicam em 143 *half-steps* por segundo: <https://forum.arduino.cc/t/28byj-48-5v-stepper-100-rpm/575338>

rebaixo feito na base inferior da plataforma. Assim, a Tabela 3.1 sintetiza os demais valores calculados, que estão exemplificados na Figura 3.7 (KIPEL, 2018).

$$m = \frac{d_e}{Z+2} \tag{3.6}$$

$$b > 6cdotm \tag{3.7}$$

Figura 3.7 – Dimensões de projeto para engrenagens



Fonte: Adaptado de (KIPEL, 2018).

Tabela 3.1 – Cálculos das dimensões para engrenagens

Medida	Eng. menor (mm)	Eng. maior (mm)	Equação
Diâmetro Externo (d_e)	16,00	36,02	(3.8)
Diâmetro Primitivo (d_p)	13,33	33,35	(3.9)
Diâmetro de Base (d_b)	12,53	31,34	(3.10)
Diâmetro Interno (d_i)	10,45	30,46	(3.11)
Passo (P)	4,19	4,19	(3.12)
Espessura do Dente (e_d)	2,09	2,09	(3.13)
Vão (v)	2,09	2,09	(3.13)
Filete (f)	0,22	0,22	(3.14)

Fonte: Autor.

$$d_e = m \times (Z+2) \tag{3.8}$$

$$d_p = m \times Z \tag{3.9}$$

$$d_b = dp \times \cos\theta \tag{3.10}$$

$$d_i = de - (2,166 \times m) \tag{3.11}$$

$$P = m\pi \tag{3.12}$$

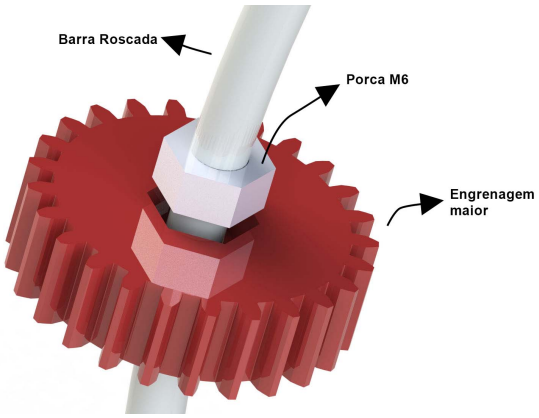
$$v = e_d = \frac{P}{2} \tag{3.13}$$

$$f = \frac{m}{6} \tag{3.14}$$

3.1.5 Barra Roscada

A barra roscada é o componente que realiza a elevação da base superior, recebendo o movimento da engrenagem maior através de uma porca sem travante, fixada na engrenagem. A porca é inserida na engrenagem, como ilustra a Figura 3.8, e se mantêm por meio de um colante *TeekBond*.

Figura 3.8 – Vista explodida da engrenagem maior com a porca e a barra roscada



Fonte: Autor.

Essa barra é comercializável somente em formato reto, sem curvatura. O processo de dobra pode ser feito de forma artesanal ou com equipamentos especializados. Para o projeto, fez-se a dobra de forma manual, corrigindo imperfeições até que o resultado ficasse o mais próximo possível do projetado. Essa conferência é realizada com o gabarito

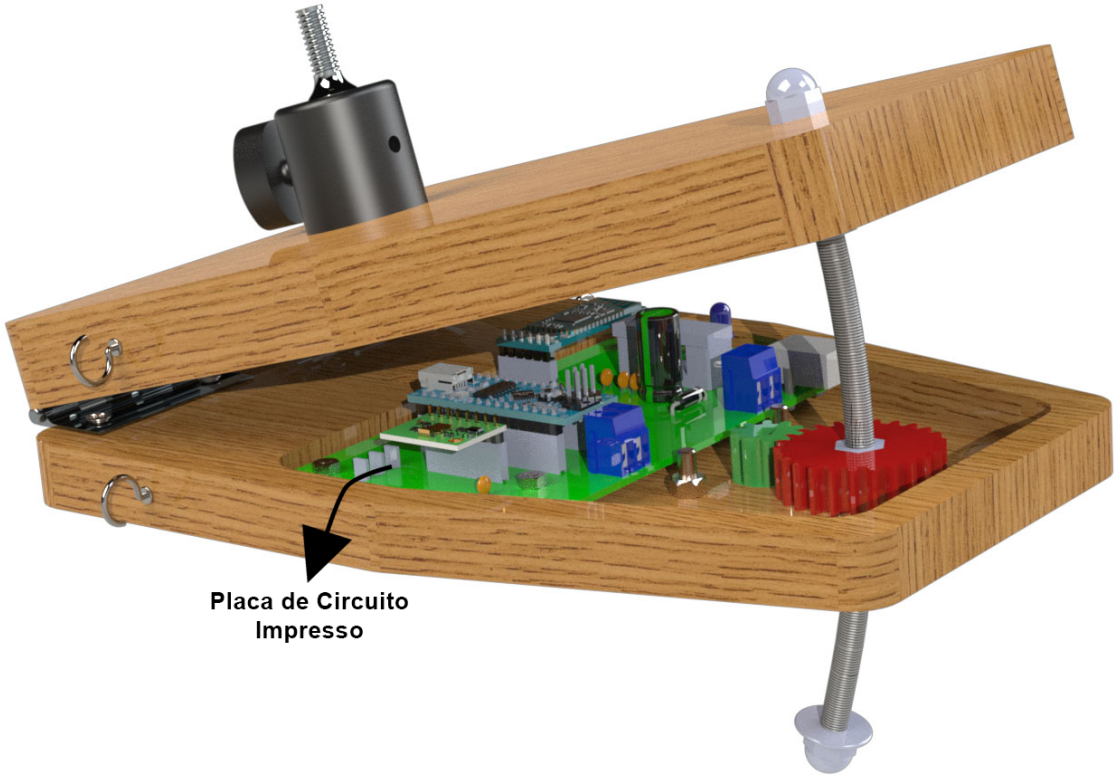
do Apêndice A.1.

Por isso, destaca-se que barra roscada pode se tornar o principal causador de problemas na estabilidade do sistema. A curvatura pode gerar instabilidade no contato com a engrenagem e, se não for fabricada de maneira precisa, pode criar certas incertezas no funcionamento.

3.1.6 Placa de Circuíto Impresso

Para finalizar o projeto mecânico, foi inserido um modelo 3D da placa de circuito impresso, garantindo que os componentes mecânicos e eletrônicos estão projetados corretamente, alinhando furos de fixação e de botões. A Figura 3.9 demonstra como ficou o projeto final.

Figura 3.9 – Render do projeto final



Fonte: Autor.

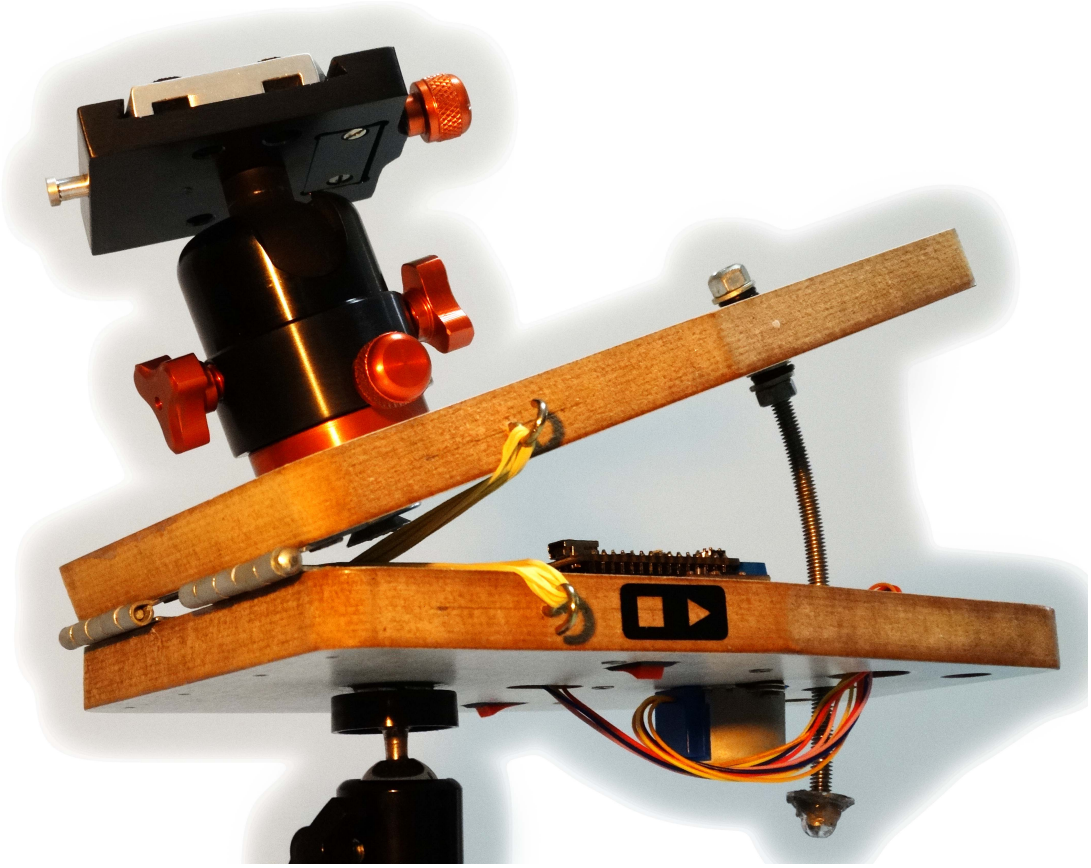
3.2 MONTAGEM

Após a fabricação e montagem da plataforma, o resultado pode ser visualizado nas Figuras 3.10, 3.11, 3.12 e 3.13. Nela, é possível reparar alguns detalhes como o *Ball-Head* Maxxi Grua utilizado para fixar a câmera na base superior. Além disso, foi inserido na montagem física um conjunto de 4 elásticos que colaboram para amortecer a vibração do sistema, que será detalhada na Seção 5.1.3.

Figura 3.10 – Vista lateral da montagem final do sistema

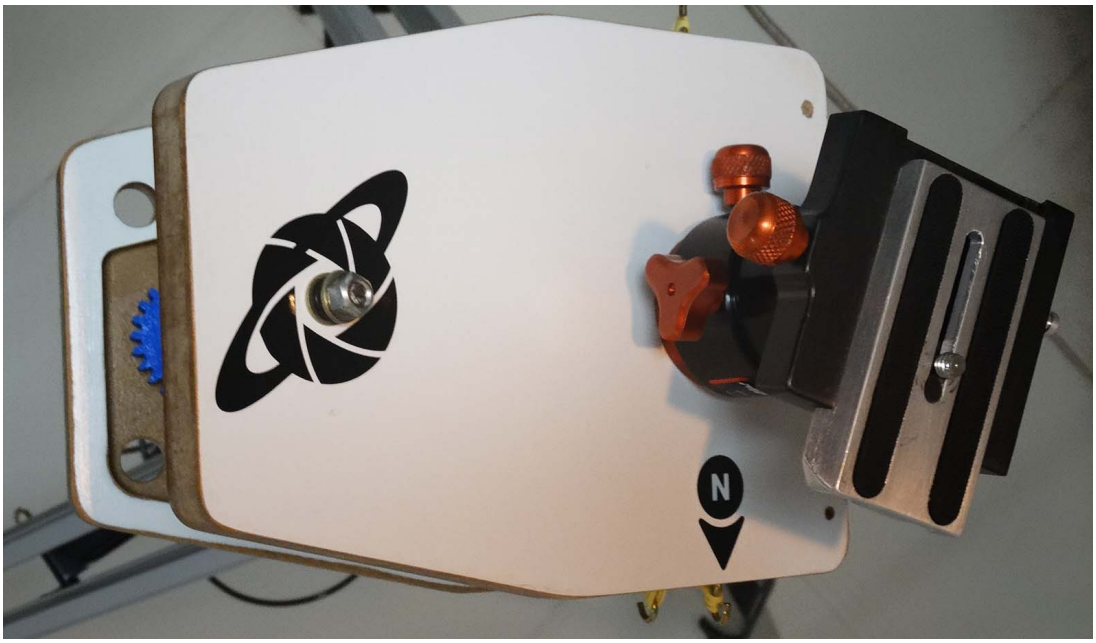
Fonte: Autor.

Figura 3.11 – Vista lateral da montagem em um tripé



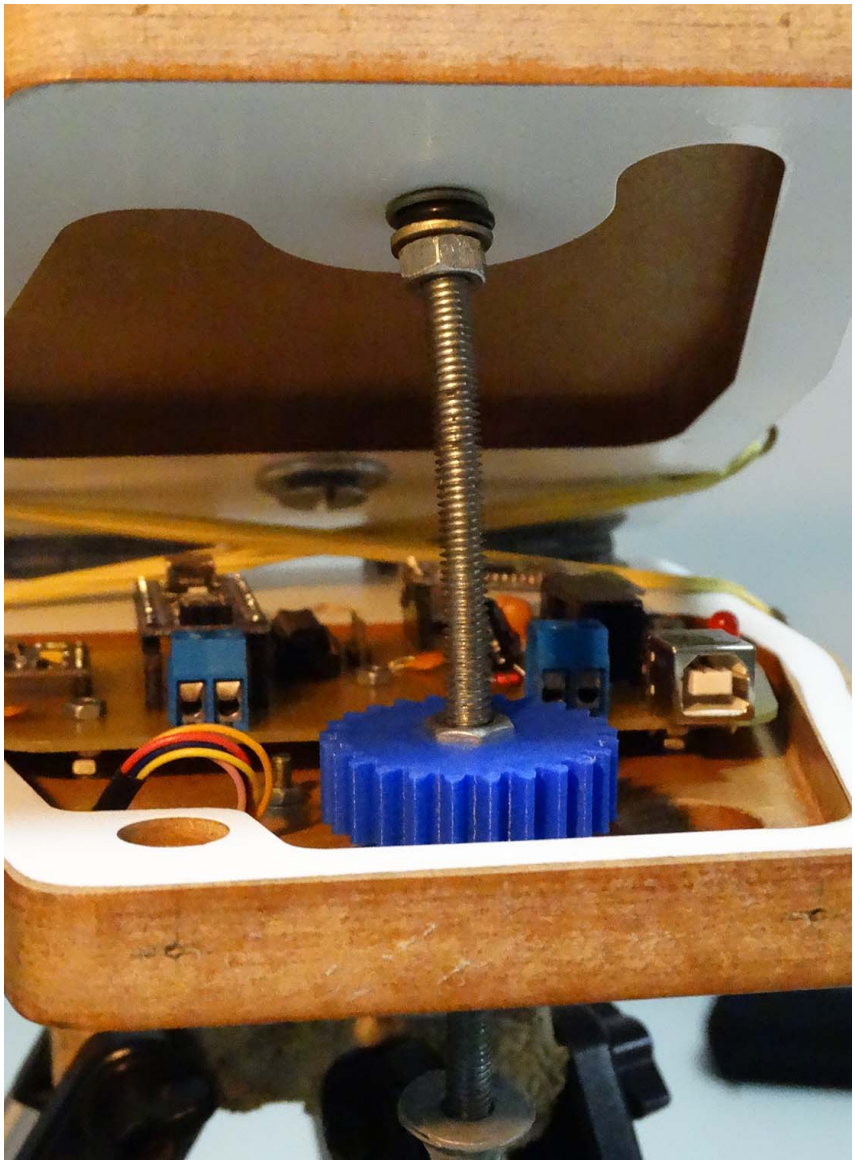
Fonte: Autor.

Figura 3.12 – Vista superior da montagem



Fonte: Autor.

Figura 3.13 – Vista frontal da montagem



Fonte: Autor.

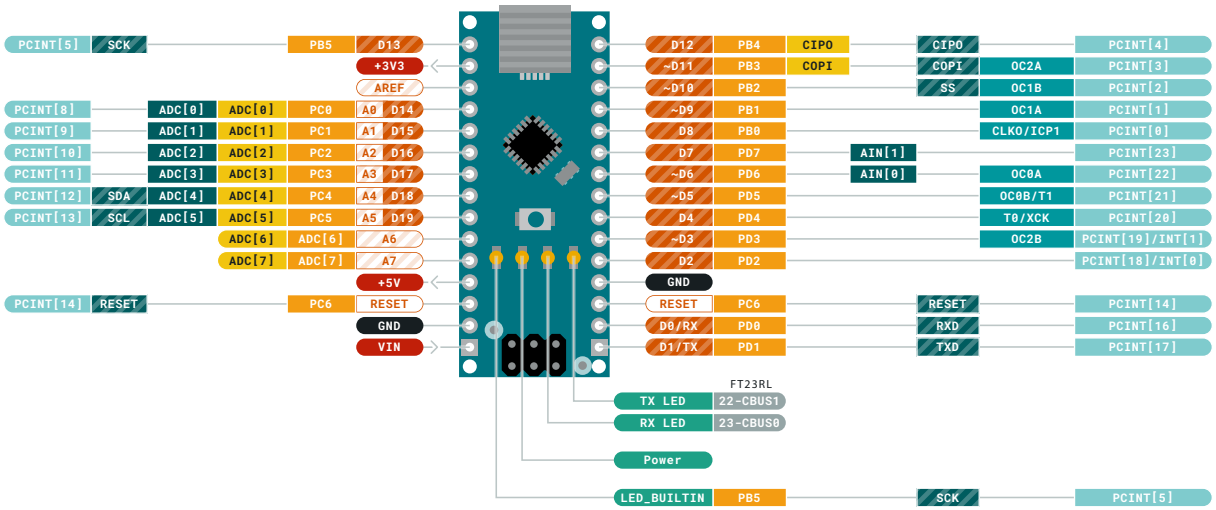
3.3 HARDWARE ELETRÔNICO

O *hardware* que foi apresentado em CAD na última subseção embarca o Arduíno, *driver* do motor de passo, botões de controle – um para energizar o sistema e outro para ativar o rastreamento –, além dos módulos Bluetooth e de sensores.

3.3.1 Arduíno Nano

Para realizar o controle do sistema, foi empregado o controlador ATMEGA328P com um Arduíno Nano (Figura 3.14). Esse controlador tem somente um núcleo de processamento com *clock* de 16MHz, além de pinos de comunicação I2C, UART, entre outros (ARDUINO, 2021). Ele foi escolhido por ter uma ampla documentação na internet, uma variedade de bibliotecas, e ser um dos mais acessíveis do mercado, principalmente no Brasil.

Figura 3.14 – Pinagem do Arduíno Nano

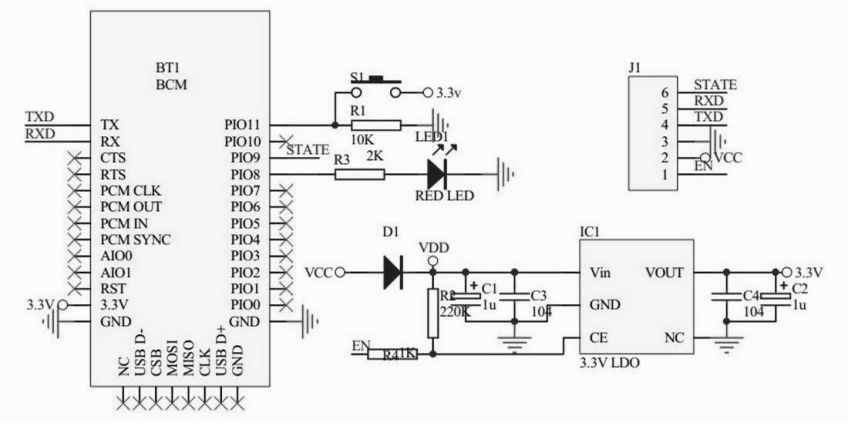


Fonte: Adaptado de (ARDUINO, 2021).

3.3.2 Módulo Bluetooth HC05

Para a comunicação Bluetooth, foi escolhido o módulo HC05, que utiliza protocolo Serial UART de comunicação e consequentemente é compatível com o controlador ATMEGA328P. O módulo, mostrado na Figura 3.15, possui os 2 pinos RX e TX de comunicação, VCC e GND de alimentação, e ainda outros 2 pinos: STATE e KEY/EN. O pino STATE indica que o módulo está pareado com algum outro dispositivo. O segundo é utilizado para permitir que o controlador altere parâmetros internos do sistema Bluetooth (GUANGZHOU HC INFORMATION TECHNOLOGY CO., 2011). O módulo embarca o *chip* HC05, LEDs indicadores, além de componentes passivos de alimentação e funcionamento da rede UART.

Figura 3.15 – Módulo HC05 e suas conexões



Fonte: Adaptado de (GUANGZHOU HC INFORMATION TECHNOLOGY CO., 2011).

3.3.3 Módulo de Sensores GY87

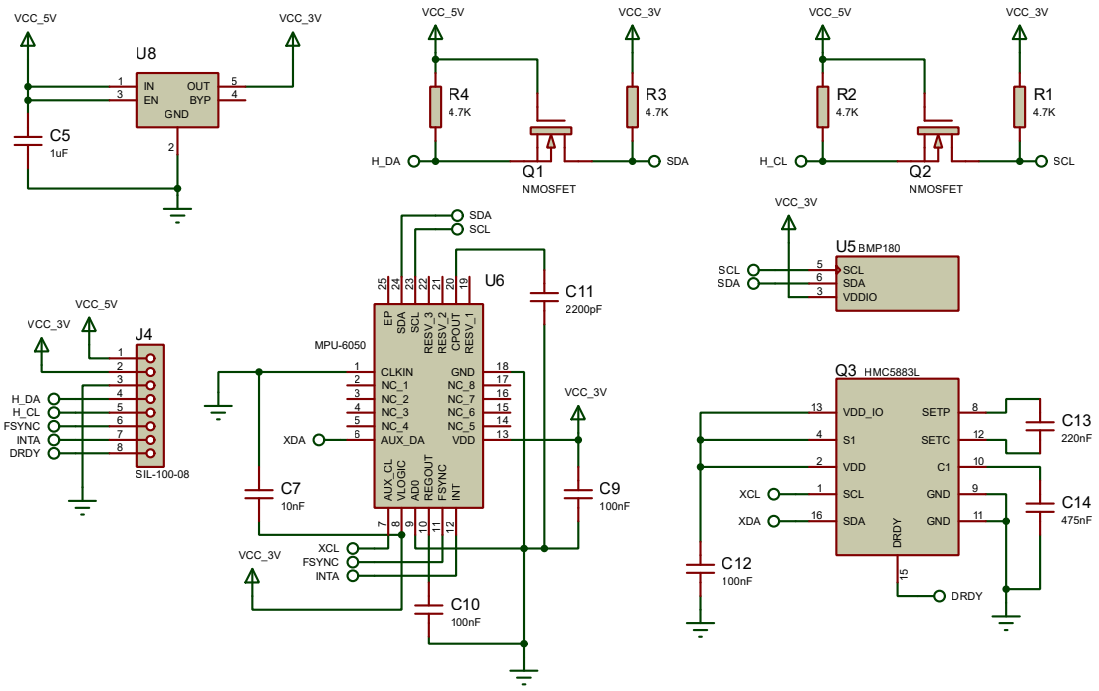
O sensoreamento da plataforma, para guiar o alinhamento equatorial, é realizado pelo conjunto de sensores MPU6050 – acelerômetro e giroscópio – e HMC5883 – magnetômetro –, pois ambos podem ser conectados facilmente em um Arduino usando um módulo GY87. A conexão é realizada com rede I2C e o circuito do módulo é mostrado na Figura 3.16.

É possível reparar na Figura 3.16 que a rede I2C do módulo interliga-se somente com as conexões do MPU6050 (H_CL e H_DA), e no módulo HMC5883 essa comunicação ocorre com os pinos I2C auxiliares do MPU6050 (AUX_DA e AUX_CL). Então, para que o HMC seja conectado diretamente com a rede I2C do módulo, é preciso configurar o MPU para que ele internamente interconecte os pinos H_CL com AUX_CL e H_DA com AUX_DA. Essa configuração é realizada no registrador 55 do MPU6050, setando-o em 1 (INVENSENSE, 2013). Além disso módulo já conta com os resistores da rede I2C e os capacitores de filtro de alimentação do circuíto.

3.3.4 *Driver* ULN2003

Para controle do motor de passo foi escolhido o *driver* ULN2003, que normalmente é comprado junto com o motor. Com ele, é possível realizar controle *full* e *half step*, operando com 5 V e sendo compatível com um Arduíno. O *driver* consegue controlar até 8 bobinas e, por isso, somente 4 portas de entrada e saída serão ocupadas. Esse *driver* é composto por um *array* de 8 transistores *Darlington*, onde o comando de cada transistor é enviado por um pino do controlador e a saída do transistor é conectada em uma fase do motor. Dessa forma, o *driver* consegue prover a corrente necessária para o funcionamento adequado do rotor (TEXAS INSTRUMENTS, 2019).

Figura 3.16 – Diagrama esquemático do módulo GY87



Fonte: Adaptado de (SMART PROTOTYPING, c2022).

3.3.5 Alimentação do Sistema

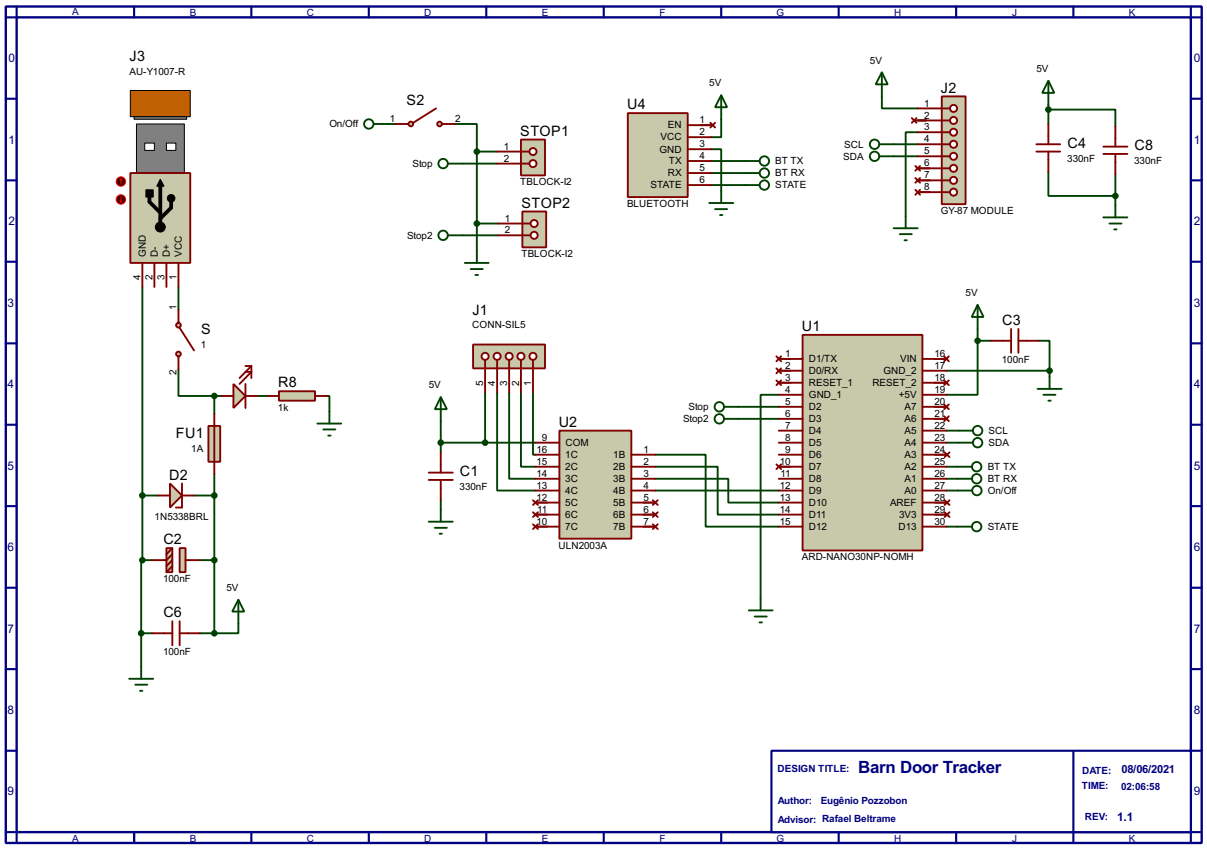
Optou-se por realizar a alimentação do sistema eletrônico com um *powerbanks* de 5 V, com um circuito de proteção entre o *powerbank* e os demais componentes. Esse circuito utiliza um fusível e um diodo *Zenner*. O fusível tem a função de impedir um surto de corrente: quando há um curto no sistema, por exemplo, o fusível abre, interrompendo a passagem de energia. O diodo Zenner 1N5338BRL selecionado trabalha com uma faixa de tensão de 5.1 V; se a tensão se elevar acima desse ponto de operação, o diodo entrará em condução, impedindo que os componentes da placa queimem pelo aumento de tensão (SEMICONDUCTOR COMPONENTS INDUSTRIES, 2013).

Fazer com que a alimentação do sistema seja de 5V torna ela compatível com as portas USB de celulares. Dessa forma, o equipamento pode ser utilizado longe de qualquer fonte de energia elétrica, sendo possível mantê-lo funcionando somente com um celular e um cabo de alimentação, por exemplo.

3.3.6 Diagrama Elétrico Completo

Todos esses fatores foram considerados no esquemático da PCB que está na Figura 3.17.

Figura 3.17 – Diagrama elétrico completo



Fonte: Autor.

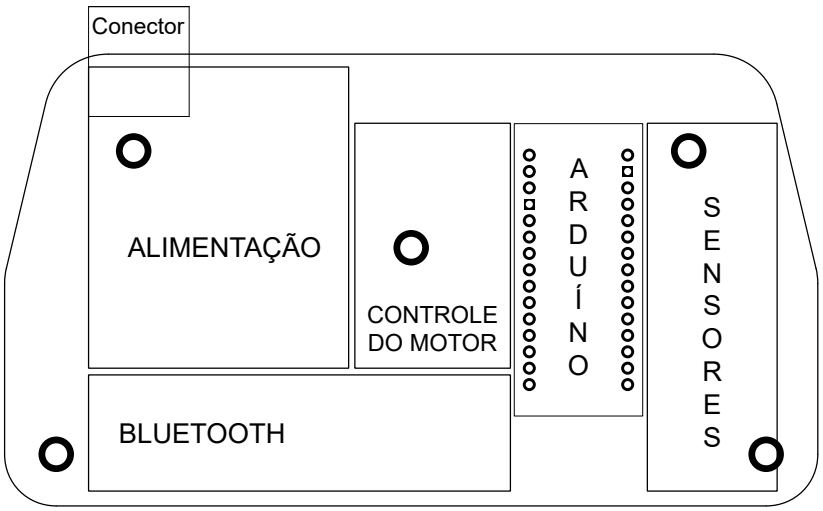
3.3.7 *Layout* da PCI

O *layout* foi desenvolvido utilizando o espaço reservado na base inferior e nos locais de fixação, respeitando as limitações dos protocolos de comunicação e recomendações de desenvolvimento para os componentes. São cinco pontos de fixação, dos quais quatro localizam-se em cada canto, e o quinto foi instalado no centro, onde planejou-se deixar o conector do motor de passo. O conector USB do tipo B e botões de controle foram posicionados próximos dos cantos da placa. Esses posicionamentos garantem que a placa não irá fletir quando o usuário estiver montando-a ou realizando um acionamento.

Além desse fator, os componentes foram organizados para serem agrupados em 4 zonas: alimentação, dados, controle e comunicação *Bluetooth* (Figura 3.18). Buscou-se manter a zona de dados o mais distante possível do sistema *Bluetooth*, e o mais próximo possível do Arduíno, para reduzir a impedância das linhas SDA e SCL do protocolo I2C dos sensores.

Foi criado um plano de terra apenas na parte inferior da placa, onde os componentes são soldados. Assim, a placa pode ser fabricada em modos mais simples e artesanais. Por isso, também, manteve-se uma distância de $0,75\ mm$ entre trilhas, e dos componentes

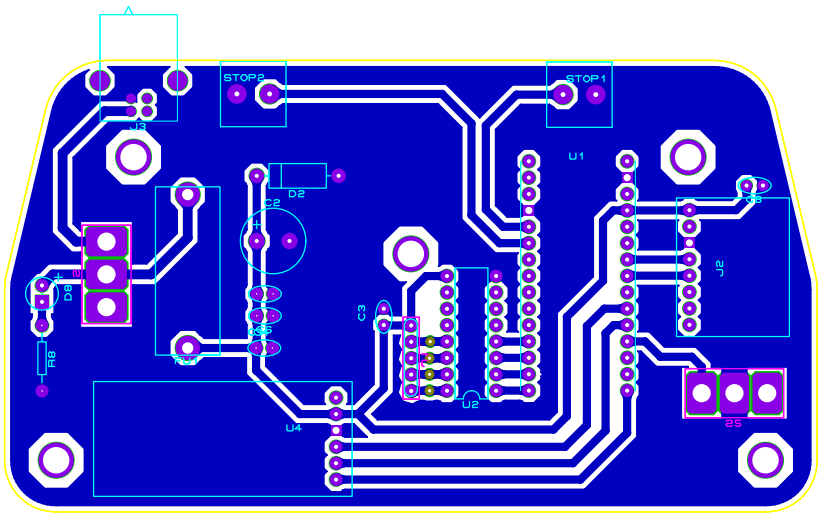
Figura 3.18 – Regiões da PCB



Fonte: Autores.

com o plano de terra, visto que distâncias menores podem incorrer em curtos durante o processo de fabricação. A Figura 3.19 mostra o projeto da PCI.

Figura 3.19 – Projeto da placa de circuito impresso



Fonte: Autor.

3.3.8 Fabricação do Circuito

A placa foi fabricada com CNC e os componentes, todos PTH (*Pin Through Hole*), foram soldados com ferro de solda e estanho. Após esses processos, foi aplicado um verniz isolante de PCI – Isotec ® – da Implastec, para protegê-la contra umidade, uma vez

que a plataforma será usada em ambientes noturnos, ficando exposta ao orvalho e sereno.

3.4 SOFTWARE EMBARCADO

As instruções do ATMEGA328p foram desenvolvidas em C++ na IDE do Arduino, e são o *software* que irá embarcar o microcontrolador[3]. Foram utilizadas bibliotecas adquiridas no GitHub[4] para os sensores, comunicação e controle do motor. Elas foram adaptadas e organizadas para simplificar o código, tornando-o mais eficiente.

A eletrônica tem como principais funções realizar o controle do motor, ler e processar os dados dos sensores inerciais, enviando os dados dos ângulos calculados para o aplicativo[5]. Além disso, deve processar o controle dos botões pelo usuário, tanto os botões físicos, como os botões virtuais do aplicativo, cuja alternância de estado é processada na comunicação Bluetooth.

O processamento dos dados dos sensores inerciais foi implementado com os algoritmos descritos na seção 2.6.3, para o cálculo de *Pitch* e *Roll*; e seção 2.6.4 para algoritmo de cálculo do ângulo azimutal com o magnetômetro. As informações de calibração do magnetômetro foram armazenadas na EEPROM do ATMEGA328P, onde foram utilizados 12 dos 1024 bytes disponíveis (ATMEL CORPORATION, 2015).

O módulo de sensores usa comunicação I2C, enquanto o módulo Bluetooth utiliza comunicação UART. O código que implementa a comunicação com esses módulos está descrito nas bibliotecas, respectivamente. O algoritmo dessas bibliotecas está em consonância com o descrito nas seções 2.7.1 e 2.7.2.

[3] O código está disponível no Github neste *link*: <https://github.com/Eugenio-Pozzobon/EasyTracker-Astro/tree/master/src/main>

[4] As licenças para uso dessas bibliotecas foram respeitadas em conjunto com o licenciamento do projeto

[5] As informações sobre o processamento e protocolo de comunicação implementados estão na seção 4.2.7

4 DESENVOLVIMENTO DO APLICATIVO

Para realizar o desenvolvimento de um aplicativo, é preciso analisar a necessidade dos usuários, determinando os casos de uso do sistema. A partir disso, é possível prototipar a interface de forma prévia, combinando as heurísticas de *design* comentadas na seção 2.8. Após essas etapas, é possível implementar o aplicativo *Android*.

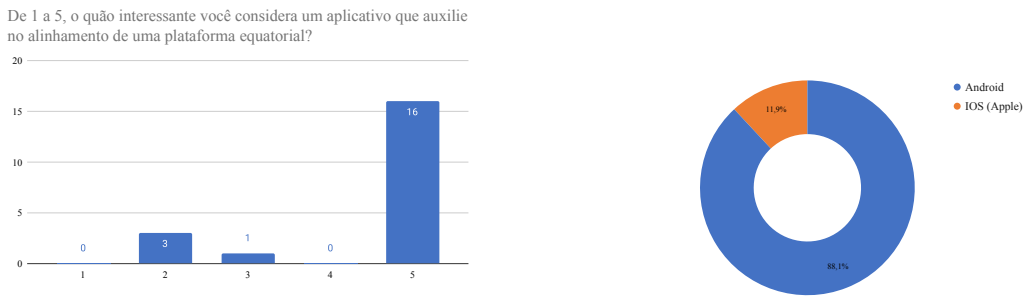
4.1 NECESSIDADES DOS USUÁRIOS

O primeiro passo do desenvolvimento foi buscar entender o que os usuários de uma plataforma equatorial precisam, e como o aplicativo pode contribuir para sanar as necessidades. Para investigar isso, foi desenvolvido um formulário de pesquisa, destinado a astrofotógrafos que já utilizaram, ou não, uma plataforma de astrofotografia. Essa pesquisa foi divulgada por meio de parceiros em mídias sociais e obteve 68 respostas.

A pesquisa foi desenvolvida buscando responder a duas perguntas gerais: "qual o nível de interesse e necessidade de uma plataforma equatorial?", e "qual o tipo de fotografia que os astrofotógrafos almejam com essas técnicas?". Essas perguntas permitem identificar, no público, se a ideia é realmente válida, qual o nível de precisão que eles precisam, e se requerem recursos para utilizar métodos de alinhamento avançado (como o método *t*, abordado na seção 2.2.2.3). Na mesma oportunidade, verificou-se qual o tipo de sistema operacional o público usa, escolaridade, entre outras informações relevantes.

Os resultados evidenciaram a necessidade do aplicativo e a relevância do projeto, demonstrado pela Figura 4.1a. Concomitantemente, constatou-se que os usuários *Android* dominam uma grande parcela do público, como demonstra a Figura 4.1b.

Figura 4.1 – Resultados do Formulário de Pesquisa com Usuários. (a) Interesse dos astrofotógrafos no aplicativo. (b) Sistema operacional usado



(a) (b)

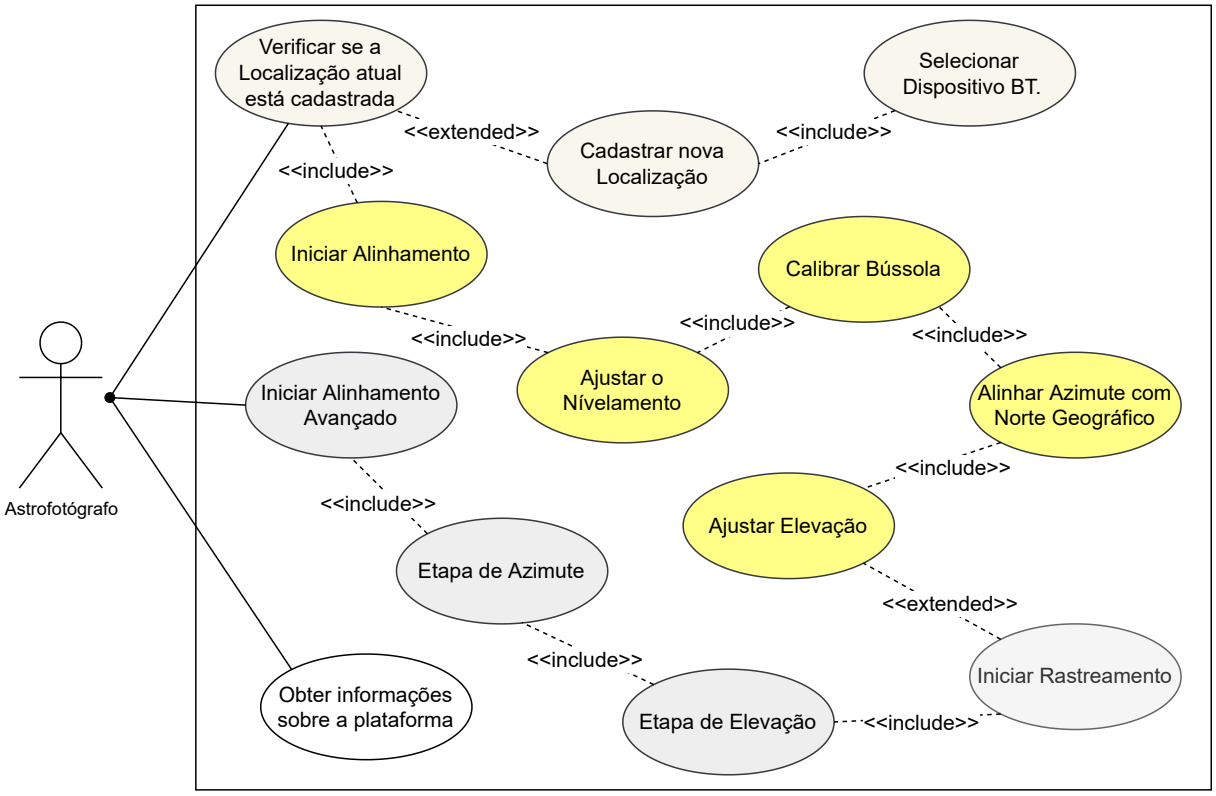
Fonte: Autor.

4.1.1 Casos de Uso

Em função das necessidades elencadas pelos usuários de uma plataforma equatorial, pode-se inferir possíveis casos de uso do aplicativo. Cada caso gera um fluxo de atividade alternativo, com cada passo que o usuário possivelmente realizará dentro do software. Em geral, têm-se três cenários.

No primeiro caso, o fluxo de atividades diz respeito ao usuário que deseja alinhar a plataforma. Um segundo fluxo faz referência a um usuário avançado, que deseja realizar um alinhamento por método *Drift*. Por fim, um fluxo comum é referente ao usuário que deseja buscar informações no aplicativo, sobre fotografia, astrofotografia ou o projeto da plataforma como um todo. Cada cenário pode ser visualizado no diagrama geral de casos de uso [1](Figura 4.2).

Figura 4.2 – Diagrama de Casos de Uso



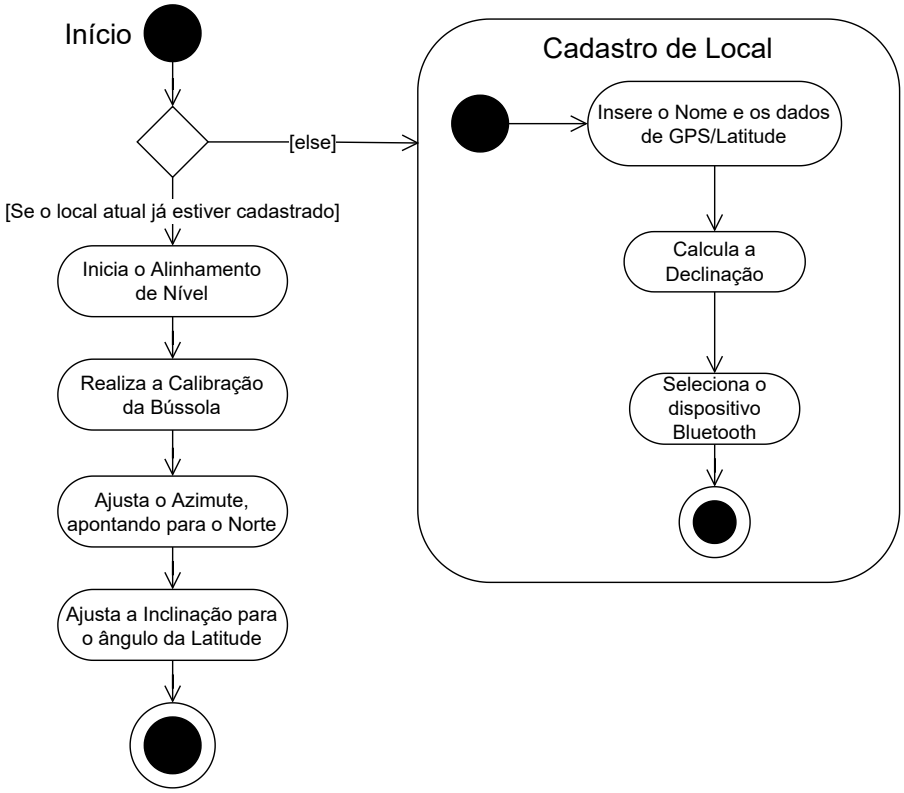
Fonte: Autor.

[1]No diagrama, quando uma ação está conectada usando o termo "*include*", significa que a próxima ação é obrigatória para o fluxo do aplicativo. Quando a conexão é feita com o termo "*extended*", significa que a ação é opcional.

4.1.1.1 Fluxo de atividades

Com base nesses três casos, determina-se qual a sequência de atividades que os usuários deverão executar para obter o que desejam. A primeira delas, de alinhamento, possui uma variação para novos usuários no aplicativo, pois estes não possuem nenhum local cadastrado no sistema, e precisam passar por telas receptivas de aprendizado. Mas, após realizar o cadastro de uma localização, o fluxo de atividade de alinhamento segue a mesma ordem: nivela-se a plataforma, ajusta-se o azimute com o polo Norte/Sul e, inclina-se a plataforma no ângulo da latitude. Esse cenário pode ser visualizado no fluxo de atividades da Figura 4.3.

Figura 4.3 – Fluxo de Atividade Principal

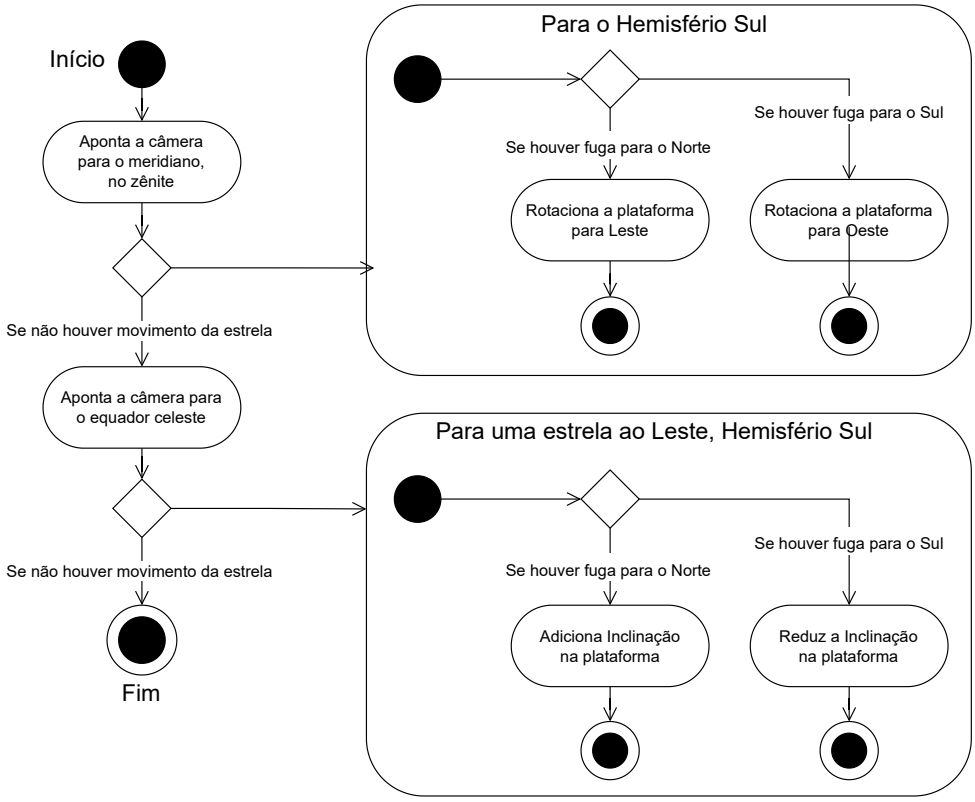


Fonte: Autor.

O segundo caso de uso, envolvendo alinhamento avançado com método *Drift*, pode ser sintetizado em um tutorial passo a passo. Primeiro para realizar o ajuste preciso de azimute, e depois para ajustar a elevação. O diagrama da Figura 4.4 demonstra o processo.

Por fim, o último caso – e mais simples –, que diz respeito a um usuário buscando informação, pode ser sintetizado em duas telas: uma com perguntas frequentes; e outra com informações mais complexas, redirecionando para os repositórios digitais e documentos.

Figura 4.4 – Fluxo de Atividade para alinhamento com Método *Drift*



Fonte: Autor.

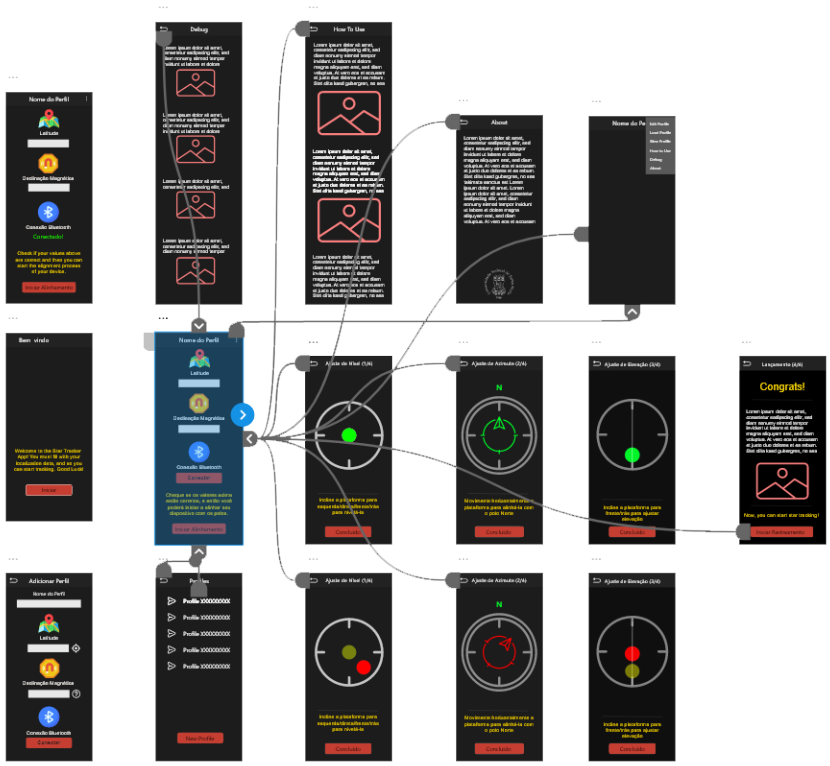
4.1.2 Prototipação das Telas

O fluxo da atividade principal, envolvendo configuração do aplicativo e alinhamento da plataforma, foi esboçado usando o Adobe XD (Figura 4.5). Isso foi realizado para determinar quais elementos de interface seriam usados para o usuário monitorar o sistema e coordenar o seu fluxo de uso. No entanto, ressalta-se que essa etapa funciona apenas como um passo inicial, e não aborda de forma técnica problemas que podem surgir no código. Ou seja, trata-se de um esboço, que apenas guiará a consistência do projeto como um todo.

De forma geral, optou-se por usar ícones para chamar a atenção do usuário para as variáveis de posição, declinação magnética e status da conexão *Bluetooth*. Como somente esses três dados são fundamentais para o uso da plataforma, foi possível criar um *layout* padrão de perfil para cada localização (Figura 4.6).

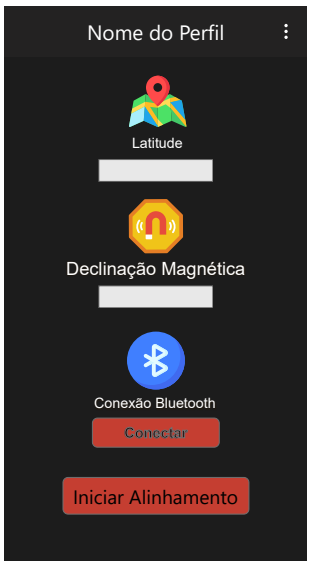
Além das telas de dados, o fluxo da atividade de alinhamento da plataforma foi projetado para guiar o usuário em um passa-a-passo simplificado. Porém, todas as telas foram desenhadas para manterem o mesmo padrão de interface, tornando a experiência mais fluída e compreensível. As demais situações foram pós implementadas diretamente durante o desenvolvimento do sistema *Android*, sendo apenas abstraídas durante essa primeira etapa.

Figura 4.5 – Prototipação das Telas no Adobe XD



Fonte: Autor.

Figura 4.6 – Esboço de tela padrão para o perfil de dados de localização



Fonte: Autor.

4.2 IMPLEMENTAÇÃO DO APLICATIVO

Diante do planejamento de interface (UI) e experiência (UX) de usuário feito anteriormente, incia-se o processo de implementação do aplicativo *Android*. Três problemas

precisam ser discutidos: recursos de *design* e experiência do usuário empregados; dados a serem coletados do usuário; e arquitetura de desenvolvimento. Antes de abordar essa discussão, será demonstrado a parte gráfica do *software*.

4.2.1 Interface e Experiência

A interface gráfica do aplicativo foi desenvolvida considerando a identidade visual planejada para o projeto, e baseando-se em elementos prontos da biblioteca Material Design.

4.2.1.1 Identidade Visual

A identidade visual foi montada com cores alaranjadas, e misturando elementos como o diafragma de uma câmera e os anéis de Saturno. O aplicativo foi nomeado como EasyTracker Astro e a identidade mostrada na Figura 4.7 é aplicada em todos os materiais[2].

Figura 4.7 – Base da Identidade Visual

Fonte: Autor.

4.2.1.2 Elementos e Cores

A biblioteca Material unifica padrões e elementos para geração de recursos *mobile*, e por isso buscou-se sempre utilizá-la para inserção de componentes. Assim, todos os campos de entrada de dados do usuário, na criação do perfil de localização, são campos de entrada providos por ela. As cores seguem um padrão do modo escuro da Material. As cores que não são padronizadas pela biblioteca, como em ícones, são baseadas na

[2]Neste *link* é possível acessar o Guia da Identidade do Projeto: <https://github.com/Eugenio-Pozzobon/EasyTracker-Android-App/tree/master/Identidade%20Visual>

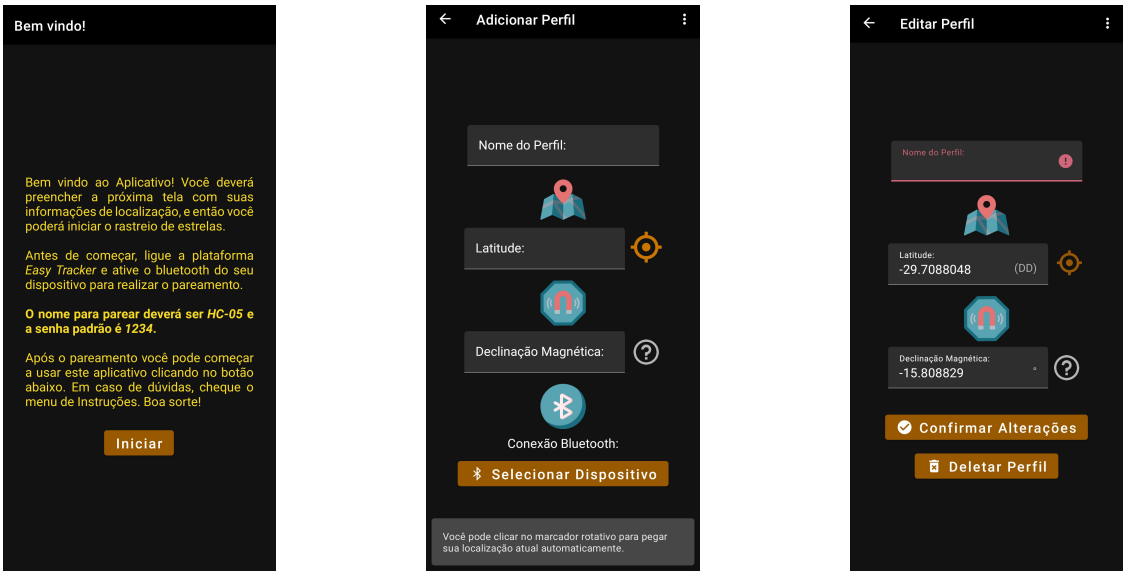
identidade visual do projeto. O código de cada cor é configurado no projeto em um arquivo[3] que informa o código hexadecimal das cores utilizadas.

Com base nesses elementos, cores, e nas telas do protótipo esboçadas na etapa da subseção 4.1.2, implementou-se o aplicativo. As telas serão abordadas nas próximas sub-seções na ordem da experiência de um novo usuário. [4]

4.2.1.3 Primeiro Uso

Um novo usuário da plataforma é reconhecido pelo sistema e para ele aparece uma mensagem inicial de boas vindas, e que informa a ele sobre a necessidade de pareamento com o sistema eletrônico antes de usar o App. Na sequência, o usuário deve registrar uma localização nova de onde ele usará a plataforma (Figura 4.8)

Figura 4.8 – Telas Iniciais



(a) Boas vindas (b) Inserção de dados (c) Erro: faltou o nome

4.2.1.4 Manipulação de Perfis de Localização

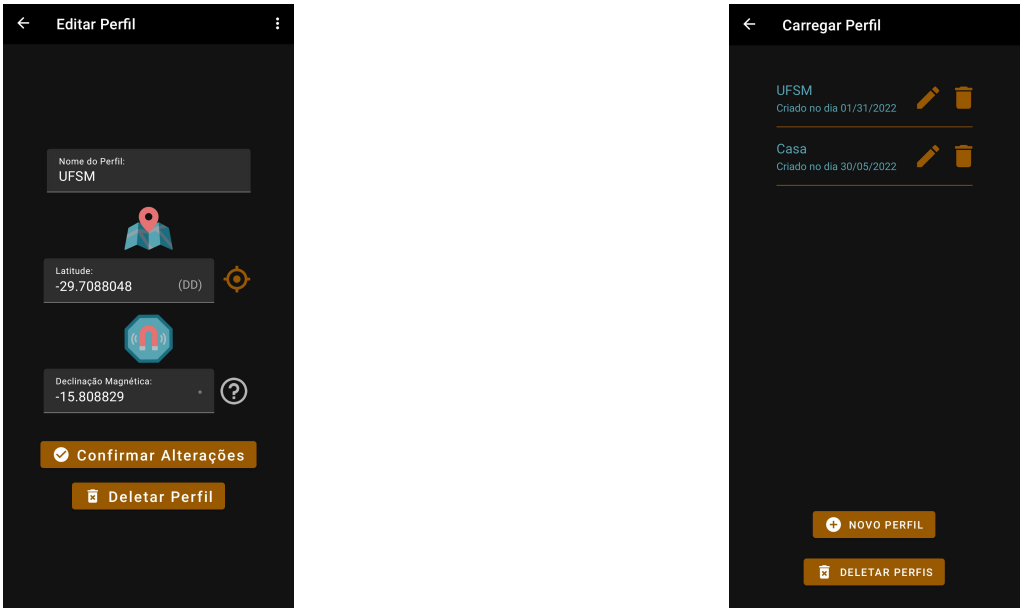
Após configurado uma primeira vez, o aplicativo irá abrir sempre na tela de perfil, carregando dados da última utilização, ignorando as telas anteriores. Ainda é possível que o usuário edite as informações neste perfil, ou adicione novos, edite os novos, e carregue

[3] O arquivo de configuração pode ser acessado aqui: <https://github.com/Eugenio-Pozzobon/EasyTracker-Android-App/blob/master/app/src/main/res/values/colors.xml>

[4] A interface e o uso do aplicativo podem ser melhor visualizados neste vídeo tutorial: <https://youtu.be/MnpiZsJ5V1E>

outros perfis no aplicativo (Figura 4.9). O usuário ainda pode apagar todos os perfis de localização, mas ele é obrigado a criar um perfil para usar o aplicativo.

Figura 4.9 – Telas de perfis



(a) Editar perfil

(b) Carregar perfil

4.2.1.5 Botão de Conectar em 3 estágios

O botão de conexão Bluetooth oscila em 3 estados que indicam o estado da conexão. (Figura 4.10)

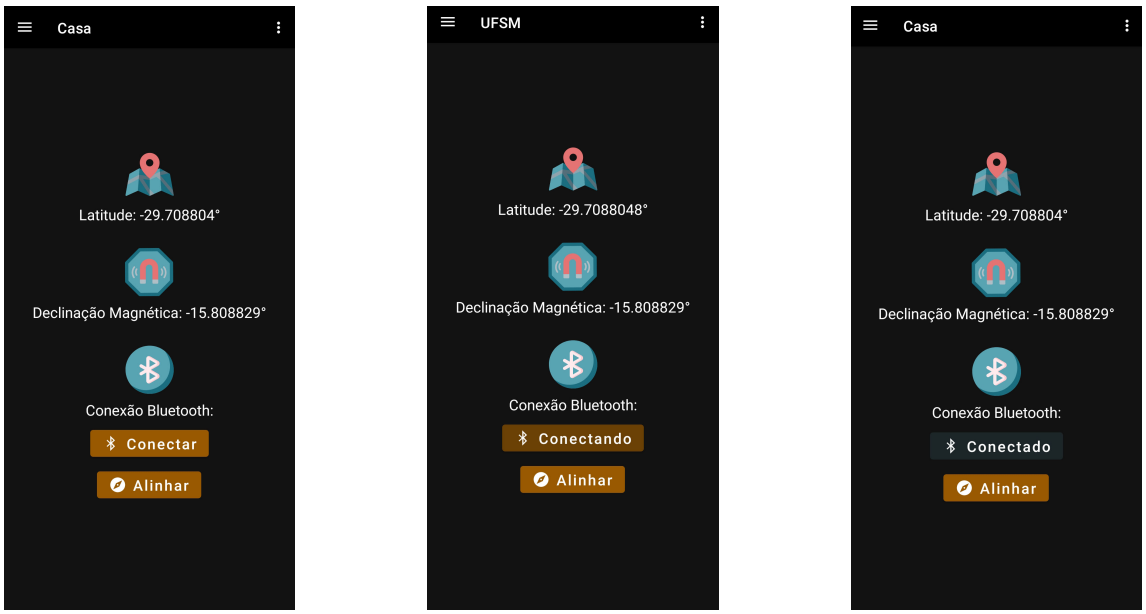
4.2.1.6 Realização do Alinhamento

As 4 telas de alinhamento consistem de 3 telas guias e uma tela final (Figura 4.11) com dicas e instruções, e com um controle de ligar ou desligar o rastreamento da plataforma. O usuário deve realizar o alinhamento pelas 3 telas tendo como base a vista superior da plataforma que está presente nas telas, guiando para nivelar e alinhar a plataforma com o polo celeste.

4.2.2 Outras Telas

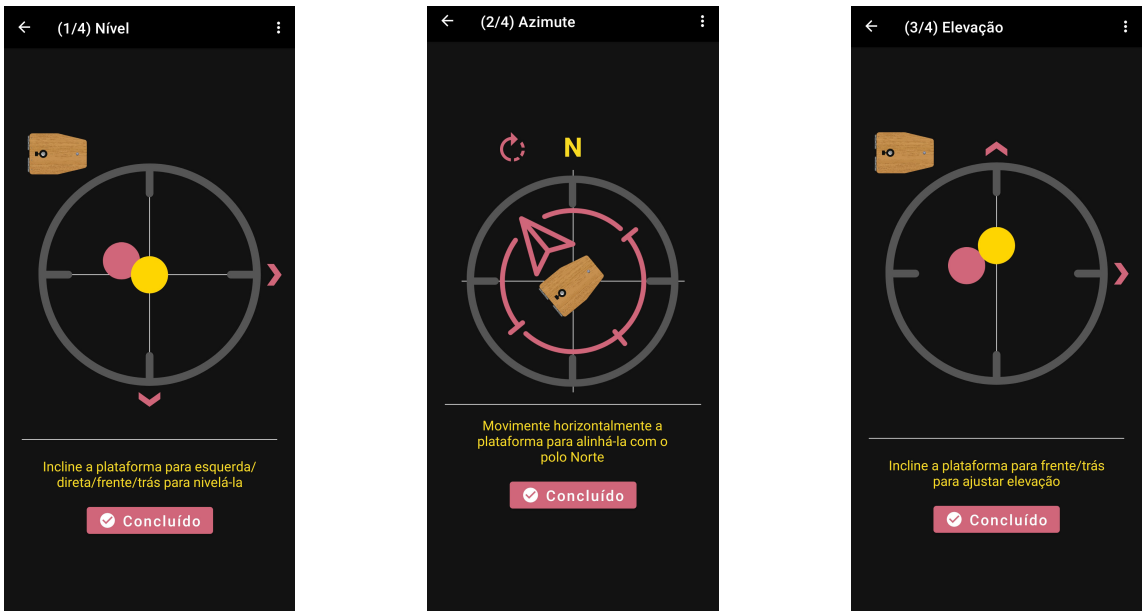
Ainda existem telas com guias e tutoriais (Figura 4.12a), que pode ser expandida com mais questões, telas sobre o aplicativo (Figura 4.12b), que redirecionam para *links*

Figura 4.10 – Conexão Realizada



(a) Conexão não iniciada (b) Conexão em andamento (c) Conexão concluída

Figura 4.11 – Telas de Alinhamento



(a) Nível da plataforma (b) Azimute da Plataforma (c) Elevação da plataforma

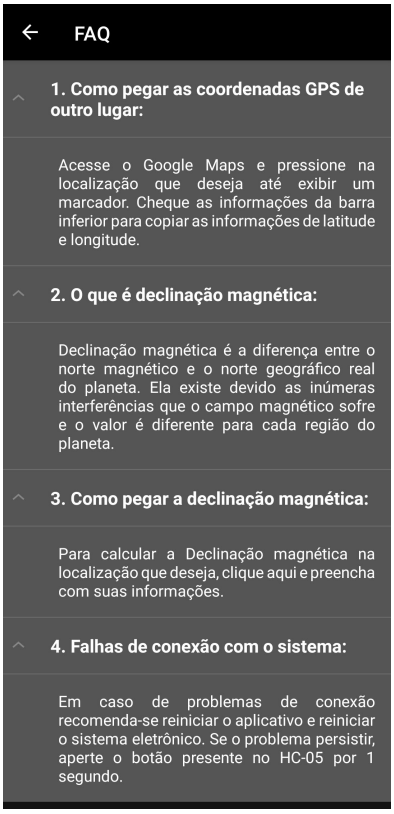
externos de documentação *open-source.*

4.2.3 Menus

O sistema ainda é composto por menus. O menu lateral (Figura 4.13a) pode ser acessível somente na tela inicial da atividade que é a tela com o perfil principal. O menu

Figura 4.12 – Telas Informativas

(a) Tela de informações



(b) Questões comuns

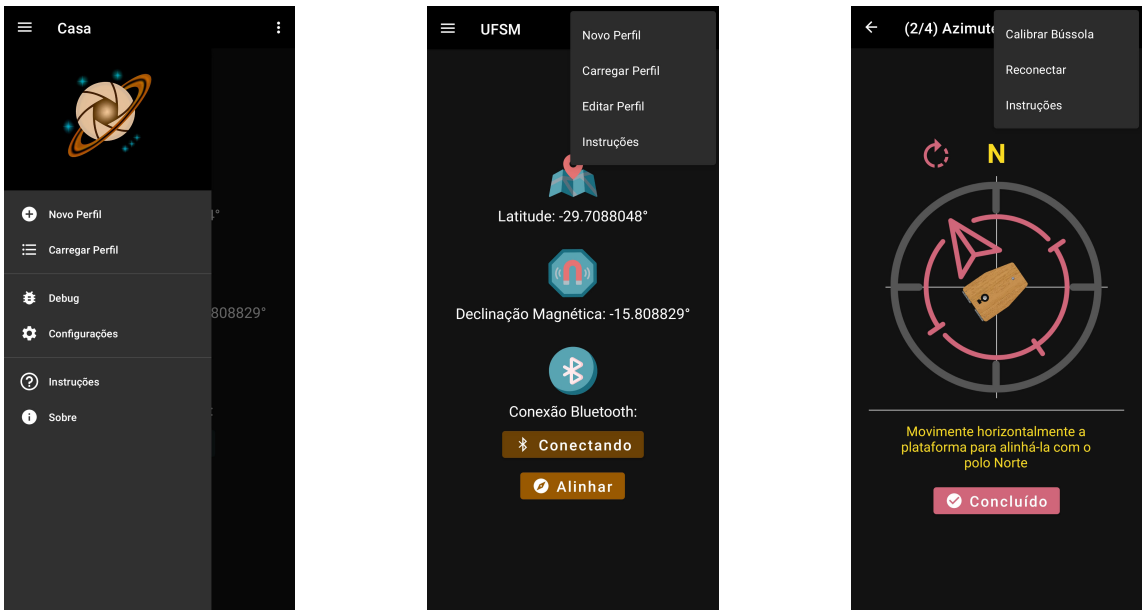
superior é configurável e foi melhorado trazendo soluções específicas para cada tela, de forma geral, oferecendo a possibilidade de acessar a seção de dúvidas sobre o sistema. Os principais menus constam na tela inicial (Figura 4.13b) e na tela de bússola (Figura 4.13c).

4.2.4 Requisitos de Sistema

A plataforma da Google possui uma série de versões e é atualizada anualmente para os usuários finais. Porém, na realidade, o que faz um aplicativo funcionar são as inúmeras bibliotecas utilizadas durante o seu desenvolvimento, e que são atualizadas com mais frequência pela gigante de buscas. Essa realidade impõe aos desenvolvedores uma necessidade de, constantemente, se manterem atualizados e de atualizarem o código de seus aplicativos.

Por conta dessas atualizações, existem funções de sistemas que só são disponíveis a partir de versões específicas do *Android*, que é decorrente da constante evolução já mencionada. Então, por exemplo, quando um *smartphone* se encontra no *Android* 10, do ponto de vista do desenvolvedor isso implica que esse *smartphone* trabalha com uma API

Figura 4.13 – Menus



(a) Menu de Navegação (b) Menu de perfil (c) Menu da bússola

Android número 29. Consequentemente, ele terá algumas funções que não existem para o *Android* 9 (API 28).

Então, ao desenvolver um aplicativo *Android*, o programador deve estabelecer uma versão mínima para o qual ele irá prover suporte, e uma versão *target*, que é a versão oficial em que o aplicativo operará com todos os seus recursos padrão. Normalmente, o desenvolvedor define a versão *target* para ser a mais recente oferecida pela Google.

Para o caso do sistema proposto, a versão 5 do Android é a mínima para o qual é possível oferecer suporte. Isso motiva-se pois é no *Android Lolipop* que a Google passa a incorporar à tecnologia *Bluetooth Low Energy* no sistema. Além disso, segundo as estatísticas oferecidas pelo *Android Studio*, os *Smartphone Android* com versões 5, ou superior, totalizam 98% dos dispositivos usados no mundo[5].

Ou seja, oferecer suporte para *smartphones* com versões inferiores à *Lolipop* implica em possíveis problemas de gerenciamento de energia, relacionados a sistemas *bluetooth*. Isso gera uma série de possíveis *bugs* e problemas, delimitando o *trade-off* entre oferecer o aplicativo para mais 2% dos usuários ou ter de lidar com a possibilidade desses problemas. Essa definição é informada diretamente ao sistema compilador do aplicativo, *Graddle*, no arquivo de construção (*build*) do software, como demonstra a Figura 4.14 [6].

Então, a partir da definição de quais subsistemas do celular serão usados junto com o aplicativo, entende-se quais permissões são necessárias obter do usuário. Neste caso, em função da necessidade de uso do GPS e o *Bluetooth*, é preciso que o usuário

[5] Informações obtidas no dia 1 de Fevereiro de 2022.

[6] O arquivo de *build* pode ser acessado aqui: <https://github.com/Eugenio-Pozzobon/EasyTracker-Android-App/blob/master/app/build.gradle>

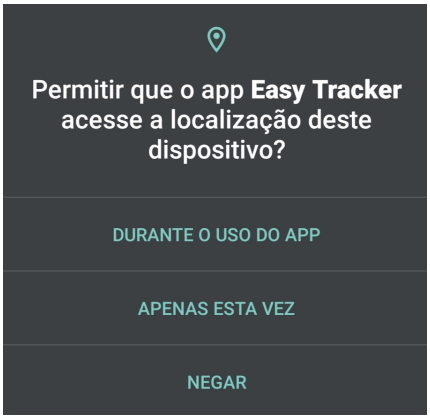
Figura 4.14 – Configuração principal do compilador, no arquivo *build.graddle*

```
...
android {
        ...
        defaultConfig {
                minSdkVersion 21
                targetSdkVersion 31
                ...
        }
}
```

Fonte: Autor.

permita serviços de localização e comunicação. Esses serviços são definidos no Arquivo de Manifesto[7], – onde são declaradas todas as atividades e recursos que serão usados pelo aplicativo – e, quando forem requeridos pela aplicação, a plataforma *Android* se encarregará de solicitar ao usuário a permissão, por meio de um pop-up (Figura 4.15).

Figura 4.15 – Sistema Android solicitando permissão ao usuário



Fonte: Autor.

Ademais, requisitos de espaço de armazenamento interno e memória RAM são inerentes à qualquer aplicativo. Nesse caso em específico, buscou-se simplificar ao máximo a base de dados para que o aplicativo seja o mais leve possível, também eliminando possíveis questões relacionadas a privacidade de dados do usuário.

Por isso, os únicos dados armazenados pela plataforma são aqueles que compõem a classe *Profile*, que armazena os dados do perfil de localização: latitude do local, declinação magnética do local, nome do perfil e data em que as informações foram armazenadas. Além disso, uma variável que armazena o endereço MAC do dispositivo *Bluetooth*, e uma *flag* que indica se o perfil está sendo usado pelo usuário, também foram armazenadas. O código na Figura 4.16 demonstra a criação da classe para o banco de dados[8].

[7] O arquivo de manifesto completo pode ser acessado aqui: <https://github.com/Eugenio-Pozzobon/EasyTracker-Android-App/blob/master/app/src/main/AndroidManifest.xml>

[8] O arquivo de manifesto completo pode ser acessado aqui: <https://github.com/Eugenio-Pozzobon/EasyTracker-Android-App/blob/master/app/src/main/AndroidManifest.xml>

Figura 4.16 – Código usado para estabelecer a base de dados

```
@Entity(tableName = "profile_table")
data class Profile(
        @PrimaryKey(autoGenerate = true)
        var profileId: Long = 0L,
        @ColumnInfo(name = "profile_name")
        var profileName: String = "",
        @ColumnInfo(name = "last_profile")
        var lastProfile: Boolean = false,
        @ColumnInfo(name = "gps_data")
        var gpsData: String = "",
        @ColumnInfo(name = "declination")
        var declination: String = "",
        @ColumnInfo(name = "bluetooth_mac")
        var btAddress: String = "",
        @ColumnInfo(name = "start_time_milli")
        val startTimeMilli: Long = System.currentTimeMillis()
)
```

Fonte: Autor.

4.2.5 Dados e Privacidade

Hoje em dia, a privacidade é um dos tópico muito relevante e é discutida por qualquer empresa de tecnologia. Com essa preocupação, contratos e políticas de privacidade fazem parte dessa discussão, e, por isso, é necessário que seja descrito da forma mais clara possível, para o usuário, quais e como seus dados serão usados.

Então, para o projeto, elaborou-se uma política de privacidade destacando o uso de dados *Bluetooth* e GPS. As demais informações coletadas não são considerados sensíveis[9].

4.2.6 Arquitetura do Funcionamento

A arquitetura de desenvolvimento de um aplicativo consiste na forma como a base de dados irá se comunicar com a interface de usuário, assim como buscar resolver problemas de acesso a dados em variáveis que estão sendo atualizadas em *threads* concorrentes. Existem inúmeras arquiteturas consolidadas, porém a Google recomenda oficialmente o padrão *Model-View-ViewModel* (MVVM) para *Kotlin* e, por isso, foi usado neste software.

No MVVM, o *Model* é a parte do sistema onde as classes que representam objetos de dados são definidas. A *View* consiste na atividade que está sendo mostrada na tela ao usuário, também é responsável por capturar as ações que o usuário executa ao clicar na

[9] A política de privacidade pode ser conferida na íntegra neste *link*: <https://docs.google.com/document/d/e/2PACX-1vSsbAxDjI8h1yLdCQBjsGnjhXyjDytVgF9mRqp6a1HK85VqDc5i_ZdO71KILV9Ekomr6r_xwSlvMC2R/pub>

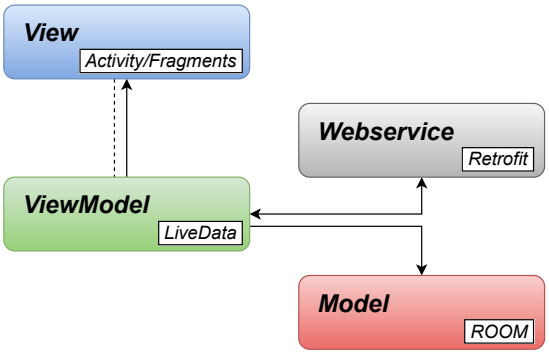
tela. O *ViewModel*, é a classe que conecta a *View* com a Base de Dados, onde são declaradas as regras de negócio (SOLUTIONS, c2022). Nele, novas *threads* são disparadas, concorrentemente, para modificar base de dados de acordo com essas regras estabelecidas pelo desenvolvedor, assim como pode se comunicar com a *View*, informando quando uma alteração deve ser realizada na tela. Essa arquitetura pode ser simplificada ao diagrama da Figura 4.17, onde o *Model* é sempre modificado por meio do *ViewModel* e a *View* pode disparar ações no *ViewModel* quando o usuário pressiona um botão ou altera um campo de texto, por exemplo, ao passo que as modificações da *View* são ordenadas pelo *ViewModel*, por meio das regras de negócio.

Figura 4.17 – Arquitetura MVVM genérica.

Fonte: (SOLUTIONS, c2022).

No desenvolvimento *Android*, em *Kotlin*, o *Model* é normalmente uma Base de Dados *ROOM*, que é o nome da biblioteca que permite criar e manipular Bancos de Dados com *querys* SQL. O *ViewModel* é uma classe específica que coordena as *Views* com regras de negócio. As *Views* são declaradas como *Activity* ou *Fragments* (SOLUTIONS, c2022). A aplicação da arquitetura MVVM dentro do ambiente Android é descrita na Figura 4.18, considerando também o uso de APIs remotas para obter acesso a dados em nuvem, que pode ser realizado com a biblioteca *Retrofit*, mas que não foi utilizado neste trabalho.

Figura 4.18 – Arquitetura MVVM aplicada ao sistema *Android*, em *Kotlin*

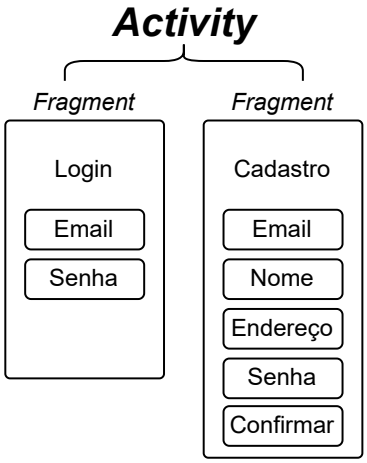


Fonte: Adaptado de (SOLUTIONS, c2022).

A *Activity* consiste na Atividade que está sendo executada. Um aplicativo pode possuir várias atividades internamente, que se conectam de acordo com regras de interface, exemplo: um usuário acessa uma atividade restrita a telas de *login*, onde ele faz um cadastro ou preenche suas informações de *login*; quando ele realiza o *login*, uma nova atividade é iniciada com as próximas telas do aplicativo carregadas. O *Fragment* consiste em cada tela exibida dentro de uma *Activity*. No exemplo anterior, a *Activity* de *login* pode conter um

Fragment para o cadastro inicial e outro *Fragment* para o usuário inserir suas credenciais (e-mail e senha) de acesso à aplicação (Figura 4.19).

Figura 4.19 – Exemplo de *Fragments* dentro de uma mesma *Activity*



Fonte: Autor.

O *ViewModel* se conecta com a *View* por meio de *databinding*. Com ele, é possível conectar um dado que é mostrado na *View* diretamente ao *ViewModel*, assim como conectar funções executadas no pressionar de um botão. Isso é possível com a Classe de dados *MutableLiveData*, que diferencia-se de outras variáveis pois é possível disparar funções somente quando ela é alterada. Com isso, o sistema não precisa monitorar a todo instante quando um dado é editado, de maneira automática e economizando processamento.

A *View* é construída em um arquivo XML, onde é definido o *layout* e como os componentes se comportarão na tela do usuário. Neste arquivo, a instância de dados – que irá se comunicar por *databinding* – é criada, referenciando-a ao *ViewModel*, como demonstra o código da Figura 4.20. Dessa maneira, é possível conectar um texto diretamente à uma variável da regra de negócios, ou também conectar um botão à uma função (Figura 4.21). Essas variáveis e funções são declaradas dentro da classe de *ViewModel* referente ao *fragment* onde está sendo referenciada, simplificado na Figura 4.22.

4.2.7 Protocolo de Comunicação *Bluetooth*

Dentro dos componentes de arquitetura, o *Bluetooth* se encontra fora de qualquer estrutura. Ele funciona com *threads* separadas, mas sua classe é necessariamente atrelada à uma *Activity* e ao contexto de um *Fragment*. Entre outras palavras, o *Bluetooth* do celular só se comunica com o aplicativo se estiver associado aos elementos da tela. Por esse motivo, estabeleceu-se que toda a seção principal, de perfil de localização e alinhamento, consistiria em uma atividade única. Caso o aplicativo fosse dividido em mais

Figura 4.20 – Declaração do *ViewModel* dentro do *layout*, no *Fragment* que cria um novo perfil de localização

```
<layout>
        ...
        <!-- Disponibiliza os dados do ViewModel para realizar databinding. -->
        <data>
        <variable
        name="newProfileViewModel"
        type="com.epp.easytracker.newprofile.NewProfileViewModel" />
        </data>
</layout>
```

Fonte: Autor.

Figura 4.21 – Código dentro do *layout* de um *fragment* para conectar um texto à uma variável, e um botão à uma função que aciona o *ViewModel* diretamente

```
<layout>
        ...
        <!-- Campo de declinação magnética.
                Quando o usuário editar,
                esse dado irá direto para o ViewModel -->
        <com.google.android.material.textfield.TextInputEditText
        android:text="@={newProfileViewModel.magDeclination}"
        android:textSize="18sp"
        />

        <!-- Quando o botão for pressionado,
                o ViewModel será sinalizado -->
        <Button
        android:onClick="@{() -> newProfileViewModel.onStartConnection()}"
        />
        ...
</layout>
```

Fonte: Autor.

Activitys, o sistema necessariamente iria desconectar o *Bluetooth* quando elas fossem alternadas. Em termos práticos, o *Bluetooth* iria reconectar com a eletrônica em cada alternância de telas, criando uma preocupação indesejada para o usuário.

A conexão é realizada usando o endereço MAC do dispositivo com que deseja-se comunicar. Para a comunicação iniciar, indica-se ao sistema, qual o serviço *Bluetooth* adotado pela eletrônica, ou seja, seu UUID. Para o caso do chip HC-05 empregado, o código como o da Figura 4.23 realiza a conexão.

Para este trabalho, a comunicação do *smartphone* com a plataforma equatorial é sempre iniciada pelo celular. O aplicativo envia mensagens de confirmação da comunicação em uma taxa de 2Hz. Com isso, a eletrônica sabe que pode enviar mensagens para o aplicativo e quais informações enviar. Caso o aplicativo não envie mensagens ou não haja conexão, a eletrônica não permanece utilizando sistema *Bluetooth* e assim economiza

Figura 4.22 – Código do *ViewModel*, que se conectará com a *View* por *Databinding*

```
package com.epp.easytracker.newprofile

//declaração do viewModel
class NewProfileViewModel(...) : AndroidViewModel(application) {
        ...
        // variável mutável para ser acessada por databinding
        var magDeclination = MutableLiveData<String>()

        // função pública para ser acessada via databiding
        // informa ao viewmodel que o botão de
        // conexão bluetooth foi pressionado
        fun onStartConnection(){
                viewModelScope.launch {
                        _startConnection.value = true
                }
        }
}
```

Fonte: Autor.

energia.

A comunicação estabelecida como padrão ocorre no cenário de alinhamento, onde o aplicativo envia o *char* ’0’ como confirmação da comunicação, e a plataforma transmite os dados dos ângulos inerciais. Quando é preciso alterar o estado da eletrônica para que inicie a calibração ou rastreamento, outro *char* é enviado. Então a plataforma envia ao celular uma mensagem com dois *char*, confirmando a alteração de estado. Isso é realizado da seguinte forma, para cada cenário:

- Iniciar rastreamento: *smartphone* envia o *char* ’s’, plataforma envia string "s,s\n"
- Parar o rastreamento: *smartphone* envia o *char* ’n’, plataforma envia string "n,n\n"
- Calibrar a bússola: *smartphone* envia o *char* ’c’, plataforma envia string "c,c\n"

Ao receber essas strings, o aplicativo altera *flags* dentro da classe *bluetooth*. Essas variáveis são declaradas como *MutableLiveData* e indicam diretamente à *View* quando houve uma alteração na plataforma, confirmando ao usuário que foi iniciado o rastreamento ou a calibração. Quando o *smartphone* envia o valor ’0’ e a plataforma não está em estado de rastreamento ou calibração, a comunicação segue o padrão.

Os dados recebidos na comunicação passam por um *buffer* de 230 *bytes*, e o aplicativo tem acesso aos dados armazenados nele após ter sido inteiramente preenchido ou se houver passado o *timeout* do sistema Android. O tempo para a eletrônica enviar 230 *bytes*, somado a latência do *Bluetooth*, implicam no tempo total de intervalo (*delay*) entre as informações serem captadas pelos sensores e aparecerem na tela para usuário.

Figura 4.23 – Código simplificado para estabelecer a base de dados

```
...
private lateinit var mmDevice: BluetoothDevice // dispositivo
private lateinit var mmInStream: InputStream // buffer de entrada
private lateinit var mmOutStream: OutputStream // buffer de saída
private lateinit var mmAdapter: BluetoothAdapter
lateinit var mmSocket: BluetoothSocket
private val myUUID: UUID? =
        UUID.fromString("00001101-0000-1000-8000-00805F9B34FB")
fun connectThread(context: Context) {
        val bluetoothManager =
                context.getSystemService(Context.BLUETOOTH_SERVICE)
                as BluetoothManager

        // checa se o dispositivo suporta comunicação Bluetooth
        // e se ela está ativada
        if (bluetoothManager.adapter != null) {
                mmAdapter = bluetoothManager.adapter
                if (!mmAdapter.isEnabled) {
                        throw Exception(
                                "Bluetooth adapter não encontrado ou acionado!")
                }
                ...
                // Declara o dispositivo pelo seu endereço MAC
                mmDevice = mmAdapter.getRemoteDevice(DeviceMAC)
                // Realiza a Conexão
                mmSocket = mmDevice.createRfcommSocketToServiceRecord(myUUID)
                ...
                try {
                        mmSocket.connect()
                        // Aciona os Buffers I/O de dados bluetooth
                        mmInStream = mmSocket.inputStream
                        mmOutStream = mmSocket.outputStream

                } catch (e: IOException) {
                        ...
                }
        }
        ...
}
```

Fonte: Autor.

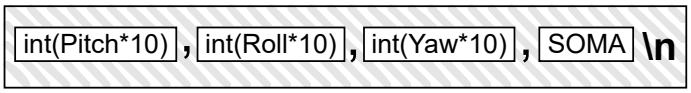
Como as mensagens são enviadas pela plataforma em 9600 bits por segundo, existe uma defasagem de, pelo menos, $230 \times 8/9600 = 192$ milissegundos.

Assim, constata-se que, quanto mais rápido o sistema for capaz de enviar as informações, mais rápido o *buffer* será preenchido. Se o *buffer* não é preenchido, observou-se que o *Android* libera os dados após um período limite (*timeout*). Além disso, outro fator relevante é o tamanho do pacote. Quanto menor for o pacote de dados contendo os ân-

gulos inerciais do *EasyTracker*, mais atualizações de tela serão possíveis no aplicativo, promovendo maior fluidez da interface.

Quando o usuário está em processo de alinhamento, e solicita dados dos ângulos para a plataforma, a Eletrônica é responsável por realizar todos os cálculos com os dados do IMU e enviar para o *smartphone* os três ângulos inerciais juntamente com uma chave de validação. Esses 4 valores são enviados entre vírgulas, e com uma quebra de linha no final da mensagem, com o caracter "\n", como demonstra a representação na Figura 4.24. Além disso, para compactar a mensagem, os valores em ponto flutuante que armazenam os ângulos na eletrônica são convertidos para um valor inteiro. Para não perder precisão, o valor é multiplicado por 10 antes de ser convertido. Com isso, o tamanho do pacote passa a ser de $2+1+2+1+2+1+2+1=12$ *bytes*. Logo, resta ao aplicativo: extrair, validar e mostrar os dados do *buffer* ao usuário.

Figura 4.24 – Pacote de dados transmitido no cenário de alinhamento



Fonte: Autor.

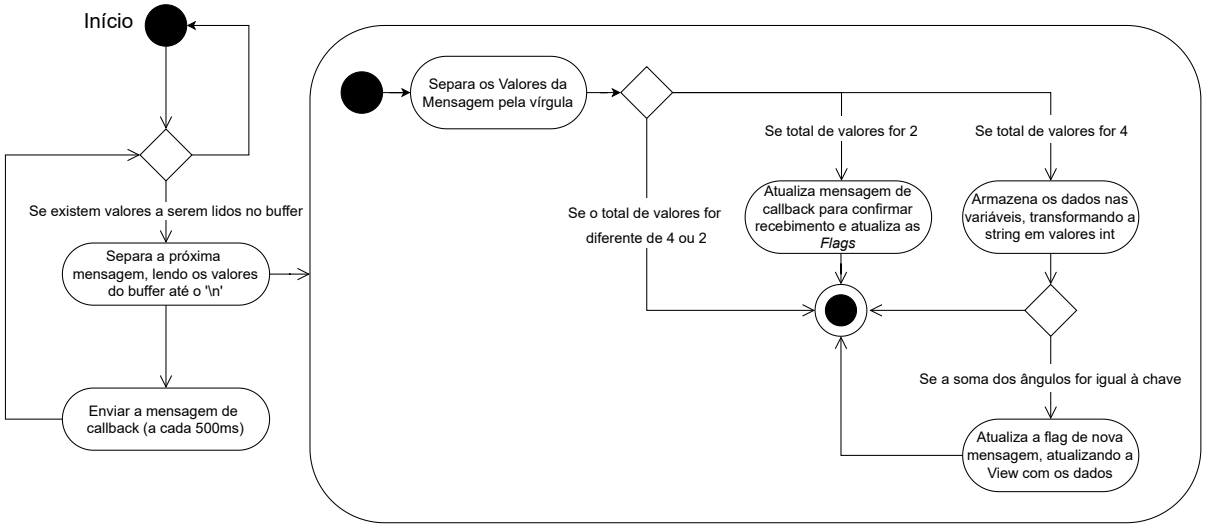
A extração dos dados é feita por meio da vírgula e do caracter de final de linha. Com o fim de linha separam-se os pacotes dentro do *buffer*, e os valores dentro de cada pacote são separados com o caractere de vírgula. A validação mencionada é necessária pois a comunicação *bluetooth* não garante a integridade dos dados enviados na mensagem, que são interpretados como uma *String*. Isso é realizado de duas formas: conferindo se a estrutura da *String* de dados está correta, e se os valores estão corretos.

A verificação da estrutura é realizada comparando se a quantidade de valores recebidos é de 4 elementos; na sequência, a verificação de dados é feita com o quarto dado do vetor, que é calculado na eletrônica como a soma dos 3 primeiros valores. No aplicativo é feito o mesmo processo: somam-se os 3 primeiros valores e essa soma é comparada com o quarto valor enviado pela eletrônica. Se o dado calculado for igual à chave, significa que os dados mantiveram sua integridade e podem ser mostrados ao usuário. Caso contrário, são descartados. Esse processamento é descrito no diagrama da Figura 4.25.

Todos esses dados são processados dentro da classe *Android* de Serviço *Bluetooth*, onde são declaradas duas *flags*, uma indicando o status da comunicação, e outra indicando se um novo dado foi processado. A comunicação dessa *thread* com os *fragments* é realizado utilizando a classe *MutableLiveData* para monitorar as *flags*, similar ao processo de *databinding*. Assim, é possível atualizar os itens mostrados na tela, ou também determinar quando um erro de comunicação ocorreu por falha de conexão.

No caso de erro de comunicação, o aplicativo chama a atenção do usuário por meio do *popup* na Figura 4.26, informando que não está mais conectado e questionando se o

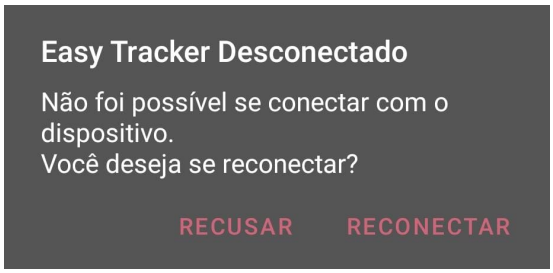
Figura 4.25 – Diagrama de processamento empregado no lado do Aplicativo



Fonte: Autor.

usuário deseja se reconectar. Assim, novamente, processamento é economizado, pois a interface irá atualizar somente quando houver mudanças no status da comunicação, ou se existirem novos dados a serem recebidos.

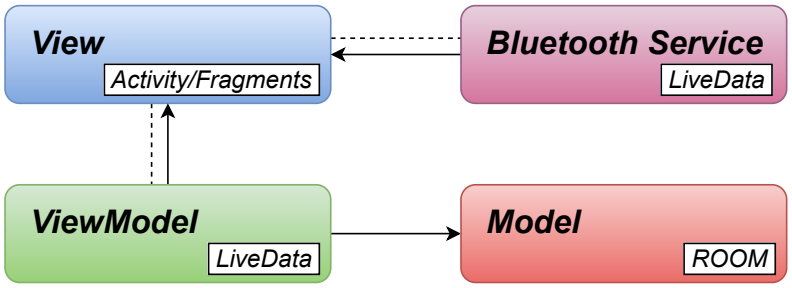
Figura 4.26 – *Popup* padrão para reconectar com a Eletrônica



Fonte: Autor.

Portanto, a arquitetura final do sistema pode ser demonstrada na Figura 4.27. A estrutura de comunicação *Bluetooth* possui suas próprias regras de negócio e se comunica direto com a *View* para modificar as informações mostradas ao usuário. O *ViewModel* controla apenas as demais regras, englobando navegação entre os *Fragments* e base de dados.

Figura 4.27 – Arquitetura final do aplicativo



Fonte: Autor.

5 RESULTADOS OBTIDOS

Para validar o sistema é preciso testar os requisitos mecânicos, de custo e de desempenho do aplicativo. Então, foram realizados uma sequência de testes em campo, como demonstra as Figuras 5.1a e 5.1b. Além disso, a plataforma foi enviada para um astrofotógrafo e Engenheiro que conduziu testes usando seu equipamento fotográfico.[1]

Figura 5.1 – EasyTracker em testes

(a)

(b)

5.1 REQUISITOS MECÂNICOS

Os objetivos mecânicos da plataforma foram alcançados com ressalvas. A plataforma montada totalizou 716 g sem o *Ball Head* da câmera, estando dentro da faixa de peso comparando com dispositivos similares no mercado. Ademais, a impressão 3D não se comportou como o esperado; somado a isso, existe um erro de inconsistência e vibração motivado por falha de fabricação na barra roscada que deve ser analisado.

[1] Relatório do astrofotógrafo Ricardo Batista pode ser acessado neste *link*: <https://drive.google.com/file/d/1CYGOFqtZQYYRpUol3SJYC5zx-3VPIXVx/view?usp=sharing>

5.1.1 Impressão 3D

A engrenagem menor é o elo mais fraco no sistema de transmissão. A primeira versão da impressão 3D acabou sofrendo uma rachadura por não suportar o torque do motor. Na Figura 5.2 é possível verificar que o trinco se forma perpendicularmente à engrenagem, evidenciando o motivo de ruptura.

Figura 5.2 – Engrenagem com rachadura em evidência

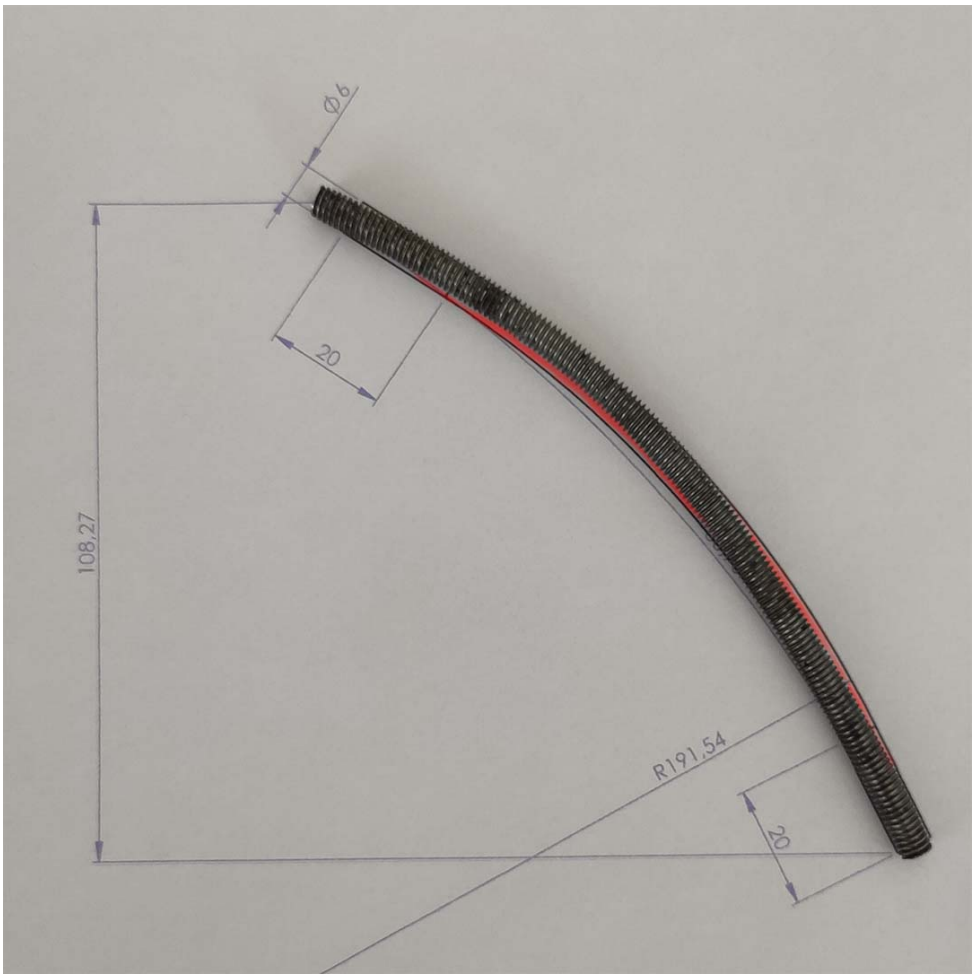
Fonte: Autor.

Além disso, quando a engrenagem apresentou a rachadura, o peso do sistema totalizava mais de 2,5 kg com o *Ball-Head* Maxxi Grua somado à câmera CANON 6D MARK II com uma lente 70-200 mm f/2.8. Contudo, este acontecimento não invalida o projeto pois ocorreu com um peso de rastreamento superior à meta proposta. Por isso, recomenda-se cautela em situações onde o equipamento se aproxima ou ultrapassa a marca de 2,5 kg de peso total.

5.1.2 Barra roscada

Além do problema com a engrenagem, o movimento de elevação da barra roscada se mostrou inconstante em certos pontos. Essa variação reflete a não idealidade da curvatura dessa barra roscada. A Figura 5.3 torna claro a discrepância entre como deveria ser, e como ficou. Esse erro de fabricação gerou inconsistência em certos pontos no funcionamento da plataforma, os quais serão demonstrados na próxima seção.

Figura 5.3 – Erro de fabricação demarcado em vermelho, onde a barra roscada apresenta uma curvatura que varia em relação ao desenho técnico



Fonte: Autor.

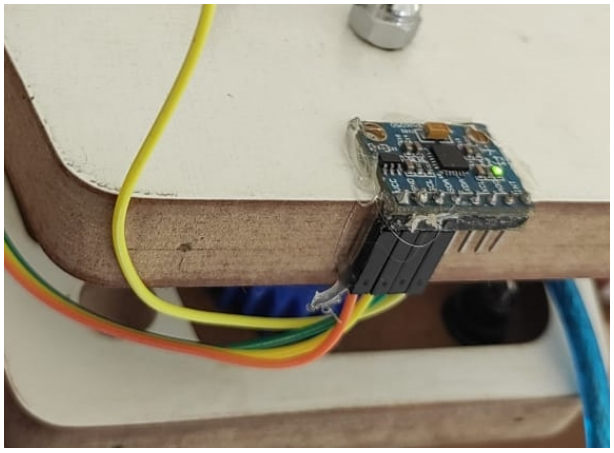
5.1.3 Análise de Consistência

A velocidade e vibração da plataforma foram analisados por meio de um acelerômetro instalado na base superior, como demonstra a Figura 5.4. Com esse teste, foi possível comparar diferentes cenários de uso de elementos amortecedores no sistema e o seu impacto na consistência e vibração. Os dados foram obtidos em uma taxa de 100 Hz, utilizando um Arduíno e comunicação serial com um *notebook* que armazenou os dados [2].

Então, os dados obtidos no primeiro teste – que constam na Figura 5.5 –, demonstram que existia uma inconstância na velocidade angular do sistema, além de um excesso de vibração. Além disso, com os dados da Figura 5.6, confirmou-se que existe a necessidade de elementos para amortecer a vibração. Ressalta-se, ainda, que existem pontos de

[2] O sistema de captação de dados com o sensor está documentado neste repositório do Github: <https://github.com/Eugenio-Pozzobon/Vibration-Analyzer>

Figura 5.4 – Acelerômetro instalado na base superior



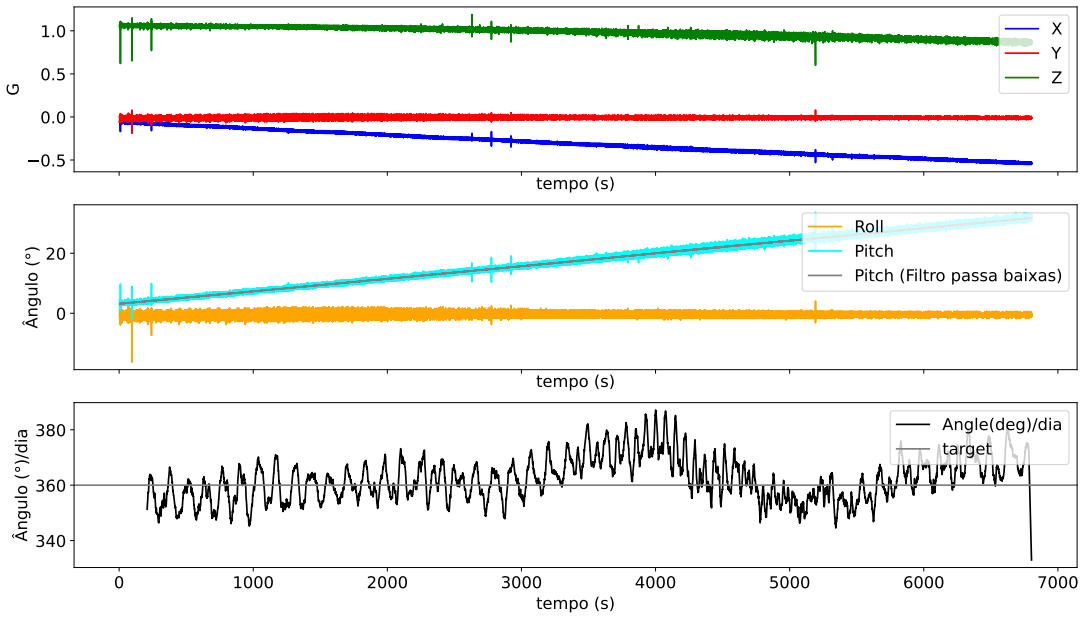
Fonte: Autor.

intensificação da vibração e, pelo período temporal onde acontecem, é possível inferir que são decorrentes do erro de fabricação do eixo curvado.

Figura 5.5 – Primeiro teste do sistema

Fonte: Autor.

Figura 5.6 – Teste do sistema com elementos amortecedores



Fonte: Autor.

5.2 CUSTO TOTAL DO SISTEMA

O custo dos materiais e processos de fabricação estão descritos na Tabela 5.1 [3], com a qual pode se constar um total de R$ 305,00 envolvendo materiais e fabricação de componentes [4]. Comparando ao valor do NyxTech – que possui uma construção e materiais similares, com um preço de US$ 110 – o EasyTracker conseguiu atingir a meta de custo. Nessa tabela, o custo referente ao MDF refere-se a seção de uma chapa de MDF necessária para fabricar as peças. Os componentes eletrônicos foram comprados em um *e-commerce* internacional, resultando nesse valor para a época. Outros Materiais diz respeito à peças como parafusos, dobradiças e material para o acabamento geral da plataforma.

Tabela 5.1 – Descrição aproximada dos custos do sistema

Item	Custo (R$)	Custo (US$)
MDF 15mm	25,00	5,06
Usinagem	100,00	20,24
PCB	50,00	10.12
Componentes Eletrônicos	95,00	19.23
Outros Materiais	35,00	7,09
Total	305,00	61,74

Fonte: Autor.

5.3 ANÁLISE DE DESEMPENHO DO APLICATIVO

O *software* pode ser avaliado com relação ao uso e implementação das heurísticas de desenvolvimento, com relação ao desempenho em *hardware* – consumo de processamento e memória –, e ainda pode ser avaliado com *feedbacks* dos usuários.

5.3.1 Interface Gráfica

Quando à interface e recursos, é possível avaliar de forma qualitativa como as 10 Heurísticas de Nielsen foram utilizadas. Os status cruciais para funcionamento são todos visíveis: o *status bluetooth* do sistema é visível por meio do estado do botão de conexão na tela de visualização do perfil. Nas telas de alinhamento, se o sistema está alinhado, os ícones ficam esverdeados, caso contrário ficam avermelhados.

[3] Alguns valores são aproximados, pois foram doações da Universidade

[4] Considerando taxa de câmbio para o dia 5 de maio de 2022, quando 1 dólar vale 4,94 reais.

Outrossim, as cores e os elementos de ícones são similares com o que os usuários estão acostumados, como os ícones nas telas de perfis de localização que fazem sentido para o contexto e são minimalistas. Além disso, as telas de alinhamento se assemelha a uma bolha de nível onde o acelerômetro é empregado. Na tela de alinhamento com o polo magnético, o sistema emula uma bússola.

Nessas mesmas telas de alinhamento, os usuários podem controlar livremente o fluxo das telas de alinhamento. Dessa forma conseguem conferir alguma informação que possam ter ignorado, assim como podem navegar para telas de suporte. Não obstante, a consistência desses padrões nas telas de alinhamento também foi consolidado nas telas de perfis. As ações de criar, editar, e visualizar um perfil usam por base a mesma tela, facilitando para o usuário entender como utilizar o aplicativo.

Além disso, a prevenção de erros é feita de forma ativa, relembrando o usuário de, por exemplo, calibrar a bússola ao entrar na tela de alinhamento azimutal, ou também com um *popup* para atualizar os dados de declinação magnética em localizações que foram armazenadas e não foram atualizadas por mais de 2 meses. A prevenção também ocorre nas entradas de dados: onde o usuário precisa inserir valores de latitude e declinação, que devem ser valores decimais (Figura 5.7). No entanto, eles podem ser escritos em graus, minutos e segundos, mas o usuário não vai conseguir inserir dessa forma, pois, o teclado numérico não permite essa formatação.

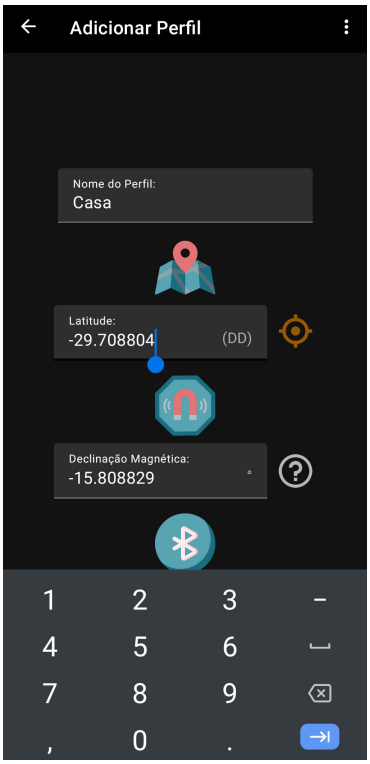


Figura 5.7 – O teclado só permite uso de vírgula e ponto, não permitindo aspas e sinal de grau (°)

Dessa forma, o desenvolvimento da interface gráfica do aplicativo conseguiu atingir

um estágio compatível com o estado da arte atual relativo a criação de aplicativos Android. As heurísticas foram respeitadas e a implementação é fluída, os itens estão dispostos respeitando uma hierarquia visual que está em conformidade com a identidade do projeto e os recursos do Sistema Android. [5]

5.3.2 Desempenho

O aplicativo foi lançado na loja de aplicativos Android via Google Play Console. O Console de desenvolvedor realiza testes automáticos com alguns dispositivos e avalia o desempenho. Os resultados indicam que o aplicativo pode ter problemas de lentidão em *hardwares* mais antigos e com menor capacidade de memória e armazenamento; mas, ao mesmo tempo, não detectou nenhum problema em sistemas mais recentes (Figura 5.8).

Figura 5.8 – Resultados apresentados no Google Play Console

Fonte: Autor.

Considerando que o dispositivo com dificuldade no processamento é defasado, é razoável considerar que o aplicativo não trará problema para a maior parcela de dispositivos atualizados.

[5] A interface e o uso do aplicativo podem ser melhor visualizados neste vídeo tutorial: <https://youtu.be/MnpiZsJ5V1E>

5.4 COMPARATIVO GERAL

Os três pontos avaliados anteriormente indicam eventuais problemas que afetaram a consistência do sistema e projeto como um todo. No entanto, é necessário verificar ainda se, apesar desses entraves, o projeto consegue cumprir com os objetivos gerais: se o método de alinhamento é tão confiável quanto os métodos tradicionais de luneta e laser; assim como se as especificações de erro estão dentro da margem comercial. Dessa maneira, dois erros podem ser avaliados: o erro inerente ao sistema mecânico, nomeado de erro periódico; e o erro relativo ao processo de alinhamento.

5.4.1 Erro Periódico

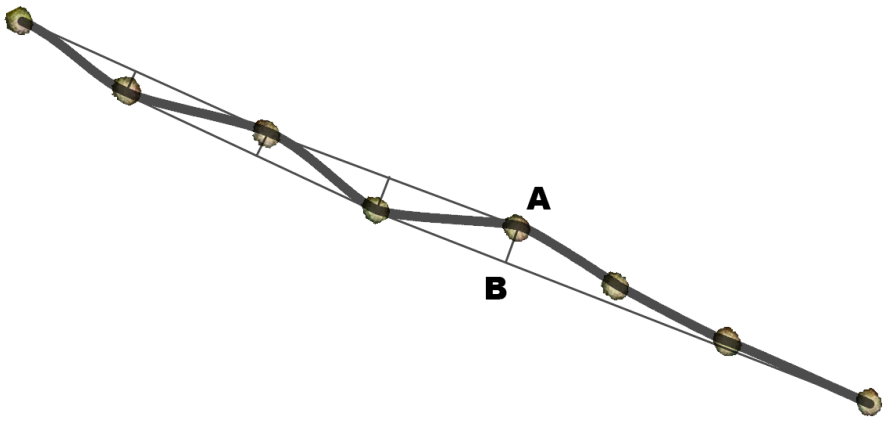
O erro periódico mensura a oscilação do sistema de transmissão, gerada por variações na montagem e encaixe dos componentes. O período desse erro é definido pelo tempo total para que o motor de passo realize uma rotação. O erro é medido em arco-segundos (arcsec), que é uma unidade que irá avaliar o quanto uma estrela alternou sua posição pelo visor da câmera, em uma fração de graus, independente da lente utilizada. O teste que mensura esse erro é realizado apontando a câmera para qualquer alvo no céu, desalinhando o rastreador propositalmente e realizando uma fotografia de longa exposição (EGAN, c2022).

Então, apontou-se a câmera para Júpiter, utilizando a Câmera DSC-HX300, com comprimento focal em $d = 192\ mm$. Assim foi possível extrair a oscilação do posicionamento da estrela ao longo de 8 fotografias captadas em intervalos de 10 s. Empilhando as imagens, têm-se o resultado da Figura 5.9. Usando um *software* editor de imagens, é possível desenhar sobre esses pontos uma forma de onda. O erro periódico é igual à distância de pico a pico medida em arc segundos. Essa distância é medida inicialmente em pixel e, posteriormente convertida para arcsec com as equações (5.1) e (5.2), sabendo que essa câmera tem um tamanho de pixel (px_s) igual a $1,19\ \mu m$ e sendo d_{AB} a distância em pixel pico a pico da onda. Esse procedimento foi realizado também com outros equipamentos e com diferentes alvos, e a média de erro periódico calculado foi igual $64\ arcsec$. Os resultados documentados podem ser conferidos no Apêndice C.1.

$$ang_{Res}\ (arcsec/px) = \frac{px_s\ (\mu m) \cdot 206,3}{d\ (mm)} \tag{5.1}$$

$$Erro_{periodico}\ (arcsec) = d_{AB}\ (px) \cdot ang_{Res}\ (arcsec/px) \tag{5.2}$$

Figura 5.9 – Pontos obtidos com as 8 imagens sobrepostas que foram captadas com intervalos de 10 s



Fonte: Autor.

5.4.2 Erro de Alinhamento

Um alinhamento bem feito está diretamente relacionado ao tempo de exposição máximo de uma foto registrada com a câmera montada no EasyTracker. Segundo o astrofotógrafo Ricardo Batista, em seu relatório, o iOptron SkyGuider Pro consegue proporcionar até 4 *stops* extras de exposição, com um alinhamento pela luneta. Isso significa dizer que o tempo total de exposição que é possível atingir com rastreamento é de $t_{max} = t_{atual} \cdot 2^{stop}$; onde o t_{atual} é o tempo utilizado para fotografias sem rastreamento. Em um exemplo, com uma câmera e lente onde a regra NPF deduza um tempo máximo de exposição de 3s, um ganho de 4 stops implica em $3 \times 2^4 = 48s$

Para determinar quantos "*stops*" é possível atingir, é necessário avaliar o erro de rastreamento da plataforma ao longo de um período de tempo. Com base nesse erro, determina-se o tempo máximo de exposição para que a estrela não passe de um limite de tolerância especificado pela Regra NPF (seção 2.1.4.1.2). Essa tolerância indica quantos *pixels* uma estrela pode mover-se na captura da imagem, sem que isso seja um rastro indesejável. Assim, com o erro determinado, compara-se o tempo pela regra NPF e o tempo de rastreamento máximo estimado pelos valores empíricos, calculando quantos *stops* extras são possíveis de se obter com o alinhamento da plataforma.

Então, em diferentes condições e câmeras, foram realizadas diversas capturas, anotando o tempo de intervalo entre elas. Todas as fotografias foram realizadas com o rastreador ligado e alinhado. Com elas, foi possível determinar quantos *pixels* a estrela deslocou no período de tempo observado, obtendo, com isso, a velocidade angular de erro. Com a equação (5.1) utilizada na seção anterior, determina-se a resolução angular da câmera, e pode-se então determinar quantos arcsec/s de erro acumula-se durante o rastreamento do EasyTracker.

Com a regra NPF, então, determina-se o tempo de exposição correto para que não haja rastro, $tmax_{NPF}$. Sabendo a resolução angular da câmera e a velocidade angular da Terra em arcsec/s calculado na equação (5.3), determina-se a tolerância do rastro de *pixels*, $NPF_{tol.}$, da regra NPF com a equação (5.4).

$$V_{Terra} = \frac{360 \cdot 3600}{23 \cdot 3600 + 55 \cdot 60 + 4} = 15,04 \; arcsec/s \tag{5.3}$$

$$NPF_{tol} = \frac{tmax_{NPF} \;\; [s] \cdot V_{terra}}{ang_{Res} \;\; [arcsec/px]} \;\; [px] \tag{5.4}$$

Dessa maneira, calcula-se o tempo máximo, t_{max}, de rastreamento com a equação (5.5) sabendo: o erro de rastreamento do sistema, V_{erro}; a tolerância de *pixels*, NPF_{tol}; e a resolução angular da câmera, $ang_{Res} \;\; [arcsec/px]$. Esse resultado, quando comparado ao tempo dado pela regra NPF, determina quantos *stops* de exposição é possível obter com o EasyTracker.

$$t_{max} = \frac{NPF_{tol} \cdot ang_{Res}}{V_{erro}} \, [s] \tag{5.5}$$

Nos testes mais recentes, possuindo prática com o equipamento, foi possível obter até $4,46$ *stops* de exposição; o que significa dizer um ganho de 21,9 vezes no tempo de exposição da foto. O Apêndice C.2 contém uma tabela registrando as medidas obtidas com cada captura realizada em testes.

Os testes conduzidos pelo astrofotógrafo levaram à marca de 4 *stops*, ou 16 vezes de ganho no tempo de exposição. Também foi possível verificar que o método de alinhamento com o aplicativo é muito mais ágil quando comparado aos tradicionais, podendo ser realizado em até quatro minutos. Isso é quatro vezes mais rápido comparando com o tempo necessário para alinhamento pela luneta do iOptron SkyGuider Pro e executado por um profissional experiente.

6 CONSIDERAÇÕES FINAIS

Os objetivos do trabalho foram elaborados com base em *benchmark* comercial, e foi possível vencer as expectativas criando um protótipo economicamente viável, com massa total dentro da média comercial, que ao mesmo tempo é mais amigável com o público-alvo. Dentre os sub-objetivos de projeto, o erro do motor é melhor do que dispositivos comerciais e os limites de peso suportado também se encontram dentro da margem esperada para o porte do protótipo. A Tabela 6.1 sintetiza a comparação do sistema com modelos comerciais, demonstrando os resultados atingidos.

Tabela 6.1 – Comparativo das Soluções de Mercado

	EasyTracker	Nyx Tracker	iOptron	SkyWatcher
Preço (US$)	61,74[1]	115	299	299
Carga Máxima (kg)	2,5	2,25	3	3
Erro Periódico (arcsec)	64	115	100	50
Peso (kg)	0,72	0,4 0	1,15	0,72
Alinhamento	Sensores e App.	*Laser*	*Polar Scope*	*Polar Scope*

Fonte: Adaptado de (EGAN, c2021a).

O aplicativo, por outro lado, é o grande diferencial. Com ele foi possível simplificar todo o processo de alinhamento e os *feedbacks* do usuário convidado demonstram sua eficiência em realizar um alinhamento bom com um curto período de tempo. Essa abordagem inovadora também foi motivo de atenção na mídia: a Revista Arco da UFSM[2], o G1[3] e a RBS TV[4] divulgaram o trabalho em suas plataformas. Com ela, foi possível captar imagens como as Figuras no Apêndice D.

Assim, o projeto passou a ser alvo de muitos interessados em obter um protótipo. No entanto, como não é necessariamente uma obrigação ou desejo dos autores fazer do EasyTracker um produto comercial, e pensando no futuro do desenvolvimento deste, optou-se por manter o projeto como *Open-Source*.

6.1 PROJETO *OPEN-SOURCE*

Então, o EasyTracker foi integralmente disponibilizado no GitHub para que possa ser reproduzido por quem se interessar e seja melhorado pela comunidade. Durante as

[2]Link da reportagem na revista Arco: <https://www.ufsm.br/midias/arco/a-arte-de-fotografar-o-espaco/>

[3]Link da reportagem no G1: <https://g1.globo.com/rs/rio-grande-do-sul/noticia/2021/12/30/pesquisadores-da-ufsm-criam-dispositivo-que-ajuda-fotografos-amadores-na-pratica-da-astrofotografia.ghtml>

[4]Link da reportagem no Jornal do Almoço da RBS TV: <https://globoplay.globo.com/v/10171682/>

etapas em que houve contato com o público, foi possível perceber que existem diversos outros usos para o conceito que foi proposto. E, com isso, a melhor forma de fazer com que essas variações do trabalho realmente venham a florescer, é tornando o projeto público, acessível e bem documentado.

Contudo, apesar de livre, ressalta-se que o projeto de *software* embarcado e Android estão licenciados sob proteção da GPLV3. A licença foi aplicada em cada arquivo de *software*, como consta em suas diretrizes. O trabalho, portanto, não pode ser reproduzido sem que os autores sejam mencionados. Além disso, tudo aquilo que for produzido com base neste material, também deve ser disponibilizado de forma *open-source*, e com a mesma licença GPL.

A documentação desse projeto, que muitas vezes foi referenciada neste documento em notas de rodapé, está protegida sob a licença *Creative Commons* (CC BY-SA). Com ela, essa documentação pode ser redistribuída e alterada, desde que os autores recebam os devidos créditos (COMMONS, c2022). Todos os arquivos licenciados carregam a marca dessa licença.

Por fim, o projeto eletromecânico está apenas disponível de forma *online* e sem licença. As leis internacionais de patente referente a componentes mecânicos impedem que este projeto possa ser patenteado, uma vez que já está disponível na internet para amplo acesso. Mesmo que fosse de interesse depositar uma patente, não seria possível pois os projetos e modelos de rastreadores *Barn-Door*, já estão publicados desde 1980.

6.2 SUGESTÕES PARA TRABALHOS FUTUROS

Finalmente, o projeto não termina com a elaboração deste documento. Existem muitos diferenciais na parte eletrônica e de *software* que podem ser expandidos para outros projetos. Por exemplo, é possível reaproveitar o projeto, remodelando-o para ser usado em conjunto com sistemas de rastreamento profissionais.

Contudo, para aqueles que desejam desenvolver e montar o seu próprio EasyTracker, melhorias podem ser implementadas, ou mesmo customizações podem ser criadas. O inserto da base inferior, usado para fixação no tripé, por exemplo, pode ter variações dependendo do padrão empregado no tripé que a pessoa utiliza, assim como o *ball-head* de fixação da câmera. Recomenda-se ainda utilizar um padrão 3/8" uma vez que este fixador é mais resistente que o 1/4", e consegue prover uma melhor fixação à plataforma.

Para continuações do projeto, é interessante uma evolução do código no Arduíno e do aplicativo. A proposta de apoio na interface para realização do método *Drift* – discutida e projetada, respectivamente, nas seções 2.2.2.3 e 4.1.1.1 – ainda pode ser implementada no aplicativo. Para a eletrônica, recomenda-se a substituição do Arduíno por um ESP32 com *bluetooth* integrado.

O ESP32 é um microcontrolador voltado para sistemas IOT (*Internet of Things*), capaz de realizar *multithread* com seu *chip dual-core* (ESPRESSIF SYSTEM, c2022). Assim, é possível aplicar uma *thread* para leitura do sensor com comunicação *bluetooth*, e outra para o controle do motor de passo; ambas são funções que não podem funcionar totalmente em simultâneo com um Arduíno.

Com essa alternativa de controlador, é possível simplificar a placa, tendo apenas o ESP32, alimentação e o sensor. O conjunto de instruções programados em C/C++ pode ser reaproveitado pois esse controlador também é programado com a mesma linguagem de programação do Arduíno.

REFERÊNCIAS BIBLIOGRÁFICAS

ADVANCED MICRO SYSTEM INC. **How to Size a Motor**. [S.l.], c2021. Disponível em: <https://www.amci.com/industrial-automation-resources/plc-automation-tutorials/how-size-motor/>. Acesso em: 16 jul. 2021.

____. **Stepper Motor System Basics**. [S.l.], c2021.

AFZAL, S. **I2C Primer**. Analog Devices, c2021. Disponível em: <https://www.analog.com/en/technical-articles/i2c-primer-what-is-i2c-part-1.html>. Acesso em: 13 de novembro de 2021.

ALVES, V. **Tipos de máquinas fotográficas**. 4. ed. [S.l.], 2018. 54 p. Disponível em: <https://fotografiadicas.com.br/wp-content/uploads/2018/07/Tipos-de-M%C3%A1quinas-Fotogr%C3%A1ficas-Fotografia-Dicas-4a-Ed.pdf>. Acesso em: 12 jul. 2021.

ANDOLFATO, R. **Astrofotografia Prática: O guia da Fotografia do Universo**. Brasil: Clube de Autores, 2017. 454 p.

ARDUINO. **Pinout Nano**. [S.l.], 2021. Disponível em: <https://content.arduino.cc/assets/Pinout-NANO_latest.pdf>.

ARORA, R. **I2C Bus Pullup Resistor Calculation**. 1. ed. [S.l.], 2015. 8 p. Disponível em: <https://sites.ifi.unicamp.br/soares/files/2017/03/Pull-up-slva689.pdf>.

ATMEL CORPORATION. **ATmega328P**. [S.l.], 2015. Disponível em: <https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf>.

BARBOSA, S. D. J. et al. **Interação Humano-Computador e Experiência do Usuário**. [S.l.]: Autopublicação, 2021.

BLUETOOTH. **Bluetooth Core Specification**. [S.l.], 2019. Disponível em: <https://www.bluetooth.org/docman/handlers/downloaddoc.ashx?doc_id=478726>. Acesso em: 12 jul. 2021.

BRESSER. **BRESSER Polar Finder Scope for StarTracker**. Bressr group of companies, c2021. Disponível em: <https://www.bresser.de/en/Astronomy/BRESSER-Polar-Finder-Scope-for-StarTracker-PM-100.html>. Acesso em: 11 ago. 2021.

BUDIU, R. **Memory Recognition and Recall in User Interfaces.** Nielsen Norman Group, 2020. Disponível em: <https://www.nngroup.com/articles/recognition-and-recall/>. Acesso em: 25 de junho de 2021.

CARUSO, M. Aplication of magnetoresistive sensors in navigation systems. In: . [S.l.]: SAE SP-1220, 1997. p. 15–21.

____. Applications of magnetic sensors for low cost compass systems. In: **IEEE 2000. Position Location and Navigation Symposium (Cat. No.00CH37062)**. [S.l.: s.n.], 2000. p. 177–184.

COMMONS, C. **Sobre as Licenças**. Creative Commons, c2022. Disponível em: <https://br.creativecommons.net/licencas/>. Acesso em: 22 de dezembro de 2021.

CORPORATION iOptron. **SkyGuiderTM Pro camera mount full package**. IOPTRON, c2021. Disponível em: <https://www.ioptron.com/product-p/3550.htm>. Acesso em: 20 fev. 2021.

COVINGTON, M. A. **Astrophotography for the Amateur**. [S.l.]: Cambridge University, 2006.

COX, S. **00 Rule vs NPF Rule**: Shutter speed for astrophotography. photographylife, 2021. Disponível em: <https://photographylife.com/500-rule-vs-npf-rule>. Acesso em: 13 jul. 2021.

DAVID, K. **Astrophotography Part 3 of 6: Making a Barn Door Tracker**. Pentax Foruns, 2015. Disponível em: <https://www.pentaxforums.com/articles/photo-articles/astrophotography-part-3-barn-door-tracker.html>. Acesso em: 10 ago. 2021.

DEEP SKY STACKER. **How to create better images**. [S.l.], c2021. Disponível em: <http://deepskystacker.free.fr/english/theory.htm>. Acesso em: 12 jul. 2021.

____. **Light, Dark, Flat, Bias... What are they and how to create them?** [S.l.], c2021. Disponível em: <http://deepskystacker.free.fr/english/faq.htm#lightdarkflatoffset>. Acesso em: 12 jul. 2021.

DEVZONE. **Bluetooth low energy Services, a beginner's tutorial**. Nordic Semicondctor, 2018. Disponível em: <https://devzone.nordicsemi.com/guides/short-range-guides/b/bluetooth-low-energy/posts/ble-services-a-beginners-tutorial>. Acesso em: 13 de novembro de 2021.

DYER, A. **Testing the MSM Tracker**. The Amazing Sky, 2019. Disponível em: <https://amazingsky.net/2019/08/22/testing-the-msm-tracker/>. Acesso em: 10 ago. 2021.

EGAN, M. **Nyx Tech**. Nyx Tech, c2021. Disponível em: <https://nyxtech.us/>. Acesso em: 29 mar. 2021.

____. **Tips and Tutorials**. Nyx Tech, c2021. Disponível em: <https://nyxtech.us/pages/tips-tutorials>. Acesso em: 9 ago. 2021.

____. **What is periodic error and how does the Nyx Tracker perform?** Nyx Tech, c2022. Disponível em: <https://nyxtech.us/pages/nyx-tracker-faq#periodicerror>. Acesso em: 29 abr. 2022.

ESPRESSIF SYSTEM. **ESP32 Datasheet**. [S.l.], c2022. Disponível em: <https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf>. Acesso em: 10 mai. 2022.

FACUNDO, A. **Best 3D printing Material/filament for Gears!** 3D Solved, 2021. Disponível em: <https://3dsolved.com/best-3d-printing-filament-for-gears/>. Acesso em: 2 de dezembro de 2021.

GOOGLE DEVELOPERS. **Android Studio**: Guia do usuário. [S.l.], 2021. Disponível em: <https://developer.android.com/studio/intro>. Acesso em: 16 mar.2021.

GUANGZHOU HC INFORMATION TECHNOLOGY CO. **HC Serial Bluetooth Products: User Instructional Manual**. [S.l.], 2011. Disponível em: <https://www.rcscomponents.kiev.ua/datasheets/hc_hc-05-user-instructions-bluetooth.pdf>. Acesso em: 8 mai. 2022.

HARLEY, A. **Icon Usability.** Nielsen Norman Group, 2014. Disponível em: <https://www.nngroup.com/articles/icon-usability/>. Acesso em: 25 de junho de 2021.

HELERBROCK, R. **História da Astronomia**. Brasil Escola, c2021. Disponível em: <https://brasilescola.uol.com.br/fisica/historia-astronomia.htm.> Acesso em: 20 fev. 2021.

HONEYWELL. **COMPASS HEADING USING MAGNETOMETERS**. [S.l.], 1995. 2 p.

INFORMATION, N. C. F. E. **Magnetic Field Calculators**. National Oceanic and Atmospheric Administration, c2021. Disponível em: <https://www.ngdc.noaa.gov/geomag/calculators/magcalc.shtml#declination>. Acesso em: 27 jul. 2021.

INVENSENSE. **MPU-6000 and MPU-6050 Product Specification**. [S.l.], 2013. Disponível em: <https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf>.

JAUCH, F. E.; SILVA, L. d. S.; PAZ, O. L. da. **SISTEMA DE POSICIONAMENTO GLOBAL - GPS**. [S.l.], 2014. Disponível em: <https://acervodigital.ufpr.br/bitstream/handle/1884/37756/APOSTILA%20GPS%202014.pdf>. Acesso em: 15 jul. 2021.

JÚNIOR, S. D. C. **Introdução à astrofotografia**. Network for Astronomy School Education, 2016. Disponível em: <http://sac.csic.es/astrosecundaria/complementario/pt/actividades/instrumentos/astrofotografia.pdf>. Acesso em: 20 fev. 2021.

JONES, D. W. **Stepping Motors Fundamentals**. [S.l.], c2004. 22 p. Disponível em: <http://www.t-es-t.hu/download/microchip/an907a.pdf>. Acesso em: 15 jul. 2021.

JONES, E. Some experiments with curved bolt drives. **Sky and Telescope Magazine**, 1980.

JOYCE, A. **Help and Documentation**. Nielsen Norman Group, 2020. Disponível em: <https://www.nngroup.com/articles/help-and-documentation/>. Acesso em: 25 de junho de 2021.

KALEY, A. **Match Between the System and the Real Worl**. Nielsen Norman Group, 2018. Disponível em: <https://www.nngroup.com/articles/match-system-real-world/>. Acesso em: 25 de junho de 2021.

KIPEL, L. R. **A CONSTRUÇÃO DE ENGRENAGENS CILÍNDRICAS COM O AUXÍLIO DE EQUAÇÕES EM SOFTWARES CAD/CAE**. 2018. 96 f. Monografia (Trabalho de Conclusão de Curso) — Curso de Engenharia Mecânica do Centro Universitário UNIFACVEST, Lages, 2018.

KONVALIN, C. **Compensating for Tilt, Hard-Iron, and Soft-Iron Effects**. Fierce Electronics, 2009. Disponível em: <https://www.fierceelectronics.com/components/compensating-for-tilt-hard-iron-and-soft-iron-effects>. Acesso em: 6 dec. 2021.

KUNCAR, A.; SYSEL, M.; URBANEK, T. Calibration of triaxial accelerometer and triaxial magnetometer for tilt compensated electronic compass. In: SILHAVY, R. et al. (Ed.). **Automation Control Theory Perspectives in Intelligent Systems**. Cham: Springer International Publishing, 2016. p. 45–52. ISBN 978-3-319-33389-2.

LAGE, V. N. **DESENVOLVIMENTO DE UM CONTROLADOR PID PARA ESTABILIZAÇÃO DE UMA PLATAFORMA COM DOIS GRAUS DE LIBERDADE**. 2016. 96 f. Monografia (Trabalho de Conclusão de Curso) — Curso de Engenharia de Controle e Automação,Universidade Federal de Ouro Preto., Ouro Preto, 2016.

LAUBHEIMER, P. **Preventing User Errors**. Nielsen Norman Group, 2015. Disponível em: <https://www.nngroup.com/articles/slips/>. Acesso em: 25 de junho de 2021.

____. **Flexibility and Efficiency of Use**. Nielsen Norman Group, 2020. Disponível em: <https://www.nngroup.com/articles/flexibility-efficiency-heuristic/>. Acesso em: 25 de junho de 2021.

LOCHUN, K. et al. **The Complete Guide to Astrophotography.** [S.l.]: BBC Sky at Night Magazine, 2015.

MACK, J. E. **Star Watch**: The amateur astronomers guide to finding observing and learning about over 125 celestial objects. London: White Crow Books, 2008. v. 1, 304 p.

MATERIAL FUNDATION. **Material Design**: Introduction. [S.l.], 2021. Disponível em: <https://material.io/design/introduction>. Acesso em: 16 mar. 2021.

MOURA, R. S. **DESENVOLVIMENTO DE UM SISTEMA DE POSIÇÃO ESPACIAL INERCIAL**. 2013. 150 f. Monografia (Trabalho de Conclusão de Curso) — Curso de Engenharia Elétrica com ênfase em Eletrônica, Escola de Engenharia de São Carlos, São Carlos, 2013.

MULLER, N. J. **Bluetooth Demystified**. [S.l.]: McGraw-Hill, 2000. v. 1, 396 p.

MURTA, J. G. A. **Guia completo do Motor de Passo 28BYJ-48 e Driver ULN2003**. Eletrogate, 2016. Disponível em: <https://blog.eletrogate.com/guia-completo-do-motor-de-passo-28byj-48-driver-uln2003>. Acesso em: 22 de novembro de 2021.

NEDELKOVSKI, D. **What is MEMS? Accelerometer, Gyroscope & Magnetometer with Arduino**. How to Mechatronics, 2016. Disponível em: <https://howtomechatronics.com/how-it-works/electrical-engineering/mems-accelerometer-gyrocope-magnetometer-arduino/>. Acesso em: 22 de novembro de 2021.

NIELSEN, J. **Usability Heuristics for User Interface Design.** Nielsen Norman Group, 2020. Disponível em: <https://www.nngroup.com/articles/ten-usability-heuristics/>. Acesso em: 25 de junho de 2021.

OBSERVATÓRIO LUPUS. **Alinhamento pelo Método Drift**. Observatório Astronômico Lupus, 2014. Disponível em: <http://observatoriolupus.blogspot.com/2014/10/alinhamento-pelo-metodo-drift-derivacao.html?m=1>. Acesso em: 26 jul. 2021.

OLIVEIRA WALDRI DOS S. ; GON CALVES, E. N. ImplementaÇÃo em c: Filtro de kalman, fusÃo de sensores para determinaÇÃo de Ângulos. **ForScience**, v. 4, 2017.

PACIFIC BEARING COMPANY. **World Class LEAD SCREW Assemblies**. [S.l.], 2017. Disponível em: <https://pages.pbclinear.com/rs/909-BFY-775/images/Product-Catalog-Lead-Screw-Technology.pdf>.

PATRÃO, S. M. C. **Ferramenta de Teste e Validação para Algoritmos de Fusão Sensorial**. 2015. 90 f. Dissertação (Mestrado em Instrumentação Biomédica) — Instituto Superior de Engenharia de Coimbra, Coimbra, 2015.

PETERSON, Z. **I2C vs. SPI vs. UART: How to Layout These Common Buses**. Altium, 2020. Disponível em: <https://resources.altium.com/p/i2c-vs-spi-vs-uart-how-layout-these-common-buses>. Acesso em: 13 de novembro de 2021.

REDKA, M. **Kotlin vs. Java**: Which programming language to choose for your android app. MLSDev, 2021. Disponível em: <https://mlsdev.com/blog/kotlin-vs-java>. Acesso em: 16 mar. 2021.

REGINA, C. **Aprenda a Fotografar em 7 Lições**. [S.l.], 2013. 18 p.

RUSSEL, D. **Introduction to Embedded Systems: Using ANSI C and the Arduino Development Environment**. [S.l.]: Morgan e Claypool, 2010. 276 p.

SANTOS, N. S. **A astrofotografia e sua importância para a astronomia**. Educação Espacial, 2010. Disponível em: <https://educacaoespacial.files.wordpress.com/2010/10/a-astrofotografia-e-sua-importncia-para-a-astronomia.pdf>. Acesso em: 19 fev. 2021.

SEMICONDUCTOR COMPONENTS INDUSTRIES. **5 Watt Surmetic 40 Zener Voltage Regulators**. [S.l.], 2013. Disponível em: <https://br.mouser.com/datasheet/2/308/1N5333B_D-2309169.pdf>.

SERONIK, G. A tracking platform for astrophotography. **Sky and Telescope Magazine**, 2007.

SMART PROTOTYPING. **GY-87 Schematic**. [S.l.], c2022. Disponível em: <https://www.smart-prototyping.com/image/data/2_components/Arduino/100960%20MPU6050/01.jpg>.

SOLUTIONS, D. **MVVM and its implementation in Android**. Digital Solutions Consulting GmbH, c2022. Disponível em: <https://digital-solutions.consulting/blog/mvvm-and-its-implementation-in-android/>. Acesso em: 22 de janeiro de 2022.

TAKAHASI, C. K. **Arquitetura de Mobilidade Bluetooth**. 2003. 116 f. Dissertação (Mestrado em Ciência da Computação) — Universidade Federal de Pernambuco, Recife, 2003.

TEXAS INSTRUMENTS. **KeyStone Architecture**: Universal asynchronous receiver/transmitter (uart). 1. ed. Dallas, 2010. 51 p. Acesso em 8 nov. 2021. Disponível em: <https://www.ti.com/lit/ug/sprugp1/sprugp1.pdf?ts=1636387120360>.

____. **ULN200x, ULQ200x High-Voltage, High-Current Darlington Transistor Arrays**. [S.l.], 2019. Disponível em: <https://www.ti.com/lit/ds/symlink/uln2003a.pdf>.

TIER, E. G. **DESENVOLVIMENTO DE SISTEMAS DE MONITORAMENTO EMPREGANDO TECNOLOGIA LORA EM UM PROTÓTIPO TIPO FORMULA SAE**. 2019. 120 f. Monografia (Trabalho de Conclusão de Curso) — Curso de Engenharia de Controle e Automação, Universidade Federal de Santa Maria, Santa Maria, 2019.

TROTT, D. Double arm barn-door drive. **Sky and Telescope Magazine**, p. 436–441, 1989.

VALDEZ JONATHAN; BECKER, J. **Understanding the I2C Bus**. 1. ed. [S.l.], 2015. 8 p.

VECTORNAV. **MAGNETOMETER: HARD E SOFT IRON CALIBRATION**. [S.l.], c2021. Disponível em: <https://www.vectornav.com/resources/inertial-navigation-primer/specifications--and--error-budgets/specs-hsicalibration>. Acesso em: 11 dec. 2021.

VIERO, E. **Os 3 pilares da fotografia**. Eduardo e Mônica, 2018. Disponível em: <https://www.eduardo-monica.com/new-blog/iso-velocidade-abertura-exposicao-fotografia>. Acesso em: 5 de julho de 2021.

WELTEN HOLDINGS. **28BYJ-48 – 5V Stepper Motor**. [S.l.], c2022. 1 p.

WOODMAN, O. J. **An introduction to inertial navigation**. [S.l.], 2007. 37 p. Disponível em: <https://www.cl.cam.ac.uk/techreports/UCAM-CL-TR-696.pdf>. Acesso em: 16 jul. 2021.

YANG, W.; BAJENOV, A.; SHEN, Y. Improving low-cost inertial-measurement-unit (imu)-based motion tracking accuracy for a biomorphic hyper-redundant snake robot. **Robotics and Biomimetics**, v. 4, n. 1, p. 16, Nov 2017. Disponível em: <https://doi.org/10.1186/s40638-017-0069-z>.

ZANONI, F. D. **Modelagem e Implementação do Sistema de navegação para um AUV**. 2012. 116 f. Dissertação (Mestrado em Engenharia de Controle e Automação) — Escola Politécnica da Universiade de São Paulo, São Paulo, 2012.

APÊNDICE A – DESENHO TÉCNICO DAS PEÇAS

A.1 – BARRA DE ELEVAÇÃO

APÊNDICE B – BILL OF MATERIALS (BOM)

BILL OF MATERIALS (BOM)

Descrição	Família	Versão	Tipo	Código
28BYJ-48 Stepper Motor	Eletrônica	01	Peça	BDT_E_P_01_v01
Botão batente	Eletrônica	01	Peça	BDT_E_P_02_v01
Batente	Eletrônica	01	Peça	BDT_E_P_03_v01
Barra Roscada	Global	01	Fixador	BDT_G_F_05_v01
Engrenagem menor	Base inferior	01	Peça	BDT_I_P_06_v01
Engrenagem maior	Base inferior	01	Peça	BDT_I_P_07_v01
Parafuso Motor	Base inferior	01	Peça	BDT_I_P_08_v01
Mola de tração	Global	01	Fixador	BDT_G_F_09_v01
Gancho para mola	Global	01	Fixador	BDT_G_F_10_v01
Base Inferior	Base inferior	01	Peça	BDT_I_P_11_v01
Dobradiça base	Base inferior	01	Peça	BDT_I_P_12_v01
Base superior	Base superior	01	Peça	BDT_S_P_13_v01
Pino Dobradiça	Global	01	Peça	BDT_G_P_14_v01
Dobradiça topo	Base superior	01	Peça	BDT_S_P_15_v01
Montagem Inferior	Base inferior	01	Montagem	BDT_I_A_16_v01
Montagem Superior	Base superior	01	Montagem	BDT_S_A_17_v01
Base Mini Tripé	Base superior	01	Peça	BDT_S_P_18_v01
Globo - Mini Tripé	Base superior	01	Peça	BDT_S_P_19_v01
Trava Mini Tripé	Base superior	01	Peça	BDT_S_P_20_v01
Mini Tripé	Base superior	01	Montagem	BDT_S_A_21_v01
Porca Martelo 1/4"	Global	01	Peça	BDT_G_P_22_v01
PCB Montada	Eletrônica	01	Montagem	BDT_E_A_23_v01
Gancho para madeira	Global	01	Fixador	BDT_G_F_24_v01
Porca M6 Comum	Global	01	Fixador	BDT_G_F_25_v01
Porca M6 Acabamento	Global	01	Fixador	BDT_G_F_26_v01
PCB	Eletrônica	01	Peça	BDT_E_P_27_v01
HC05	Eletrônica	01	Peça	BDT_E_P_28_v01
IMU	Eletrônica	01	Peça	BDT_E_P_29_v01
Arduino Nano	Eletrônica	01	Peça	BDT_E_P_30_v01

APÊNDICE C – DOCUMENTAÇÃO DE TESTES

C.1 – TESTES DE ERRO PERIÓDICO

Câmera:	Pixel Size (um):
SONY DSC	1,19
CANON 6D M II	6,54

PERIODO (rpm) 0,76

PERIODO (s) 45,45

Foto	Câmera	FL (mm)	distância (px)	erro (arcsec)
DSC01630	SONY DSC	68	17	61
DSC01681	SONY DSC	192	61	78
DSC01681	SONY DSC	192	64	81
DSC01681	SONY DSC	192	45	57
DSC01681	SONY DSC	192	41	53
_MG_1432	CANON 6D M II	105	4	51
_MG_1432	CANON 6D M II	105	5	66
_MG_1432	CANON 6D M II	105	5	66

Média 64

Foto	Referência	Câmera	N	FL (mm)	x (px)	y (px)	tempo(s)	distância (px)	arcsec/s	arcsec/px	t_max(NPF)	npf tolerance	t_max (s)	stops
DSC01594	Theta Scorp	SONY DSC	0,36	4	2627	2105	417,42	6,08	0,89	61,37	7,65	1,87	128,65	4,10
DSC01595	Theta Scorp	SONY DSC	0,36	4	2626	2096	703,96	15,13	1,32	61,37	7,65	1,87	87,21	3,38
DSC01596	Theta Scorp	SONY DSC	0,36	4	2625	2092	901,55	19,24	1,31	61,37	7,65	1,87	87,86	3,39
DSC01597	Theta Scorp	SONY DSC	0,36	4	2625	2087	1083,18	24,19	1,37	61,37	7,65	1,87	83,96	3,31
DSC01598	Theta Scorp	SONY DSC	0,36	4	2625	2083	1195,50	28,16	1,45	61,37	7,65	1,87	79,59	3,23
								Média	1,27				93,45	3,48
DSC01600	Theta Scorp 2	SONY DSC	0,36	4	2620	2078	285,99	2,24	0,48	61,37	7,65	1,87	239,77	5,60
DSC01601	Theta Scorp 2	SONY DSC	0,36	4	2619	2073	369,80	7,28	1,21	61,37	7,65	1,87	95,23	3,53
DSC01602	Theta Scorp 2	SONY DSC	0,36	4	2619	2072	957,10	8,25	0,53	61,37	7,65	1,87	217,58	5,33
DSC01603	Theta Scorp 2	SONY DSC	0,36	4	2618	2058	1033,52	22,20	1,32	61,37	7,65	1,87	87,26	3,38
								Média	0,88				159,96	4,46
DSC01611	Jupiter	SONY DSC	0,36	4	2170	1530	405,30	14,14	2,14	61,37	5,68	1,39	39,93	2,65
DSC01612	Jupiter	SONY DSC	0,36	4	2163	1523	604,47	24,04	2,44	61,37	5,68	1,39	35,03	2,48
DSC01613	Jupiter	SONY DSC	0,36	4	2149	1518	1018,61	38,01	2,29	61,37	5,68	1,39	37,33	2,56
DSC01614	Jupiter	SONY DSC	0,36	4	2143	1513	1256,61	45,80	2,24	61,37	5,68	1,39	38,22	2,59
								Média	2,28				37,63	2,57
_MG_9957	110 Herculis	CANON 6D M II	0,36	50	3452	2335	78,20	12,17	4,20	26,98	2,15	1,91	12,25	2,39
_MG_9958	110 Herculis	CANON 6D M II	0,36	50	3462	2337	156,40	22,36	3,86	26,98	2,15	1,91	13,33	2,49
_MG_9959	110 Herculis	CANON 6D M II	0,36	50	3471	2341	234,60	32,02	3,68	26,98	2,15	1,91	13,97	2,55
_MG_9960	110 Herculis	CANON 6D M II	0,36	50	3480	2345	312,80	41,76	3,60	26,98	2,15	1,91	14,28	2,58
_MG_9961	110 Herculis	CANON 6D M II	0,36	50	3488	2346	391,00	49,73	3,43	26,98	2,15	1,91	14,99	2,64
								Média	3,75				13,76	2,53
_MG_9923	N/I 14mm	CANON 6D M II	0,36	14	2986	1702	78,20	2,24	2,76	96,37	6,93	1,72	60,20	2,95
_MG_9924	N/I 14mm	CANON 6D M II	0,36	14	2988	1705	156,40	5,83	3,59	96,37	6,93	1,72	46,17	2,58
_MG_9925	N/I 14mm	CANON 6D M II	0,36	14	2989	1707	234,60	8,06	3,31	96,37	6,93	1,72	50,09	2,69
_MG_9926	N/I 14mm	CANON 6D M II	0,36	14	2990	1710	312,80	11,18	3,44	96,37	6,93	1,72	48,16	2,64
_MG_9927	N/I 14mm	CANON 6D M II	0,36	14	2991	1713	391,00	14,32	3,53	96,37	6,93	1,72	47,01	2,60
_MG_9928	N/I 14mm	CANON 6D M II	0,36	14	2993	1716	469,20	17,89	3,67	96,37	6,93	1,72	45,15	2,55
								Média	3,38				49,46	2,67

Estrela Ref	x (px)	y (px)	Foto
Theta Scorp	2628	2111	DSC01593
Jupiter	2180	1540	DSC01610
Theta Scorp 2	2621	2080	DSC01599
110 Herculis	3440	2333	_MG_9956
N/I 14mm	2985	1700	_MG_9922

Câmera:	Pixel Size (um):
SONY DSC	1,19
CANON 6D M II	6,54

APÊNDICE D – FOTOGRAFIAS COM O EASYTRACKER

Figura D.1 – Primeiro teste realizado com CANON EOS 6D Mark II, lente 50mm, f/2.2, 4 imagens de 30 s empilhadas

Figura D.2 – Outra imagem da Via-Láctea com a CANON EOS 6D Mark II, lente 50mm, f/2.8, 6 imagens de 78 s empilhadas