UNIVERSIDADE FEDERAL DO PARANÁ

CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

SISTEMA DE AUTOMAÇÃO PARA PEQUENOS TELESCÓPIOS

CARLOS ALBERTO LERO NETO

LEONARDO YUJI FUIGUTI

CURITIBA

2024

SISTEMA DE AUTOMAÇÃO PARA PEQUENOS TELESCÓPIOS

Trabalho apresentado no curso de graduação de engenharia elétrica na Universidade Federal do Paraná.

Área de conhecimento: Engenharia Elétrica

Orientador: Prof. Dr. Carlos Marcelo Pedroso

CURITIBA

2024

RESUMO

Devido à rotação da Terra em torno do seu próprio eixo, cria-se uma ilusão de que os astros estão se movendo durante a observação. Para resolver esse problema é necessário que se tenha um sistema para que o telesópio possa acompanhar o movimento aparente dos corpos celestes. Este trabalho propõe um sistema automático que calcula a velocidade de rotação dos motores do telescópio com base em suas coordenadas, garantindo que siga a rotação da Terra para neutralizar a velocidade relativa. O software de controle será instalado em um hardware embarcado, possibilitando o acesso e configuração remota através da rede Ethernet ou Wi-Fi. Também serão feitas aquisições de imagens com o uso de um dispositivo de captura de imagem conectado ao sistema através de uma interface USB. O sistema como um todo será uma plataforma para futuros projetos na área.

Palavras-chave: Telescópio automático; Rastreamento;Controle remoto; Rede Wi-Fi

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CCD	*charge-coupled device*
USB	*Universal Serial Bus*
TCP	*Transmission Control Protocol*
GPIO	*General Purpose Input/Output*
PWM	*Pulse Width Modulation*
RPM	*Rotação por minuto*
DIR	*Direção*
VCC	*Tensão em corrente contínua*
EN	*Enable*
HTTP	*Hypertext Transfer Protocol*
IP	*Internet Protocol*
FIFO	*First In, First Out*
GND	*Ground*
SUID	*Set User ID*
PAD	*Payload Dinâmico*
LSB	*Least Significant Bit*
LSByte	*Least Significant Byte*

LISTA DE SÍMBOLOS

z	número de dentes de uma engrenagem
i	relação de transmissão
n	velocidade de rotação em RPM
w	velocidade ângular
r	raio primitivo
θ	ângulo de pressão
m	módulo
d_p	diâmetro primitivo
d_b	diâmetro de base
d_e	diâmetro externo
d_i	diâmetro interno
l	largura lateral da engrenagem
α	ângulo do dente
r	raio da curvatura do dente entre o diâmetro externo e primitivo
r_1	raio da curvatura do dente entre o diâmetro primitivo e de base
r_3	curvatura na base do dente
f	coeficiente de Grant para determinar r
f'	coeficiente de Grant para determinar r_1
a	altura da cabeça
T	período

SUMÁRIO

1 INTRODUÇÃO

O desenvolvimento de um sistema rastreador automático para pequenos telescópios surge da necessidade de aprimorar a observação astronômica, eliminando as limitações impostas pelo rastreamento manual. A automação deste processo facilita a observação contínua e habilita a aquisição de imagens através de câmeras digitais.

O contínuo avanço tecnológico nas redes de computadores tem viabilizado a criação de uma interconexão entre diversos dispositivos espalhados pelo globo terrestre. Essa interconectividade abre novas fronteiras para a exploração astronômica, possibilitando a comunicação com telescópios posicionados em áreas remotas. Esse cenário permite a obtenção de imagens e vídeos de alta resolução, mesmo quando o observador está fisicamente distante do telescópico.

Com o sistema realizando observações com acompanhamento automático, é necessário efetuar ajustes na posição angular do telescópio, uma vez que as estrelas "se movem" ao longo do dia devido à rotação da Terra em torno de seu próprio eixo e essa rotação da Terra no sentido oeste-leste levam aproximadamente 23 horas, 56 minutos e 4 segundos para completar uma rotação completa [1].

Esse movimento aparente das estrelas cria a ilusão de que estão se movendo pela constante alteração na posição do observatório devido à rotação terrestre. Este fenômeno representa um desafio para o sistema em questão, que visa acompanhar e observar uma região específica do espaço. Para lidar com o movimento aparente dos objetos celestes, é necessário ajustar os dois eixos de orientação do telescópio, cujas velocidades de ajuste variam de acordo com a posição do observador na superfície do planeta e estão relacionadas à rotação.

Atualmente, uma variedade de projetos, como o KStars [2] e o Stellarium [3], viabilizam o controle automatizado de telescópios para acompanhar o movimento aparente dos objetos celestes. No entanto, essas soluções operam em conjunto com diversos modelos de telescópios, como o GOTO, que requerem um sistema mecânico adicional. Assim, esses projetos realizam apenas uma parte do processo de rastreamento, necessitando de integração com outras soluções para uma execução completa do rastreamento.

Com o desenvolvimento deste projeto, será possível estabelecer um framework

aberto, visto que das inúmeras alternativas disponíveis no mercado para a automatização do telescópio nenhuma disponibiliza o código fonte nem o projeto do sistema mecânico. A partir disso o presente projeto habilita desenvolvimentos futuros com a adição de novas funcionalidades, como por exemplo, o acompanhamento de objetos rápidos em baixa órbita (como a Estação Espacial Internacional) ou o uso de técnicas de processamento de imagem para o aprimoramento do sistema de guiagem.

Nesse contexto, este trabalho propõe o desenvolvimento de um sistema embarcado para pequenos telescópios, visando permitir que usuários controlem remotamente o dispositivo para direcionar sua observação para áreas específicas do céu. Esse sistema proporciona diversos ganhos, sendo uma delas a melhora no acompanhamento dos objetos, uma vez que substitui a calibração da angulação em que o telescópio aponta de manual para automática, tornando o processo mais preciso. O projeto também oferece maior acessibilidade possibilitado pela introdução do sistema de rede Wi-Fi. Além disso, o projeto viabiliza a captura de imagens para astrofotografia, ampliando as possibilidades de registros astronômicos. Por fim, esse sistema abre novas oportunidades no estudo de pesquisa no campo de rastreamento e aquisição de imagens astronômicas, fomentando avanços nesta área.

Além dessa seção introdutória, a presente monografia está organizada em sete capítulos. O Capítulo 2 apresenta os objetivos do projeto. As informações técnicas fundamentais para a compreensão do projeto são detalhados no Capítulo 3. No Capítulo 4 são apresentados os recursos a serem utilizados bem como os métodos para o desenvolvimento do projeto, enquanto que o capítulo 5 demonstra os resultados atingidos. As orientações para a futuros trabalhos do projeto se encontra no capítulo 6. As conclusões são apresentados no Capítulo 7.

2 OBJETIVOS

2.1 Objetivos gerais

O projeto tem como objetivo criar um sistema de rastreamento de objetos astronômicos que permita a configuração e o envio de imagens pela internet. A implementação visa não apenas acompanhar precisamente os objetos astronômicos, mas também simplificar a operação para facilitar a utilização por parte dos usuários.

2.2 Objetivos específicos

1. Analisar os tipos de montagem disponíveis e escolher o mais adequado para o projeto.

2. Implementação do sistema mecânico selecionado.

3. Capturar e armazenar imagens de uma região do espaço desejado utilizando dispositivo de captura de imagem e um hardware de sistema embarcado.

4. Implementação de um sistema de controle dos motores para acompanhar o movimento aparente das estrelas devido a rotação da Terra.

5. Implementação da comunicação utilizando Protocolo TCP-IP para permitir a configuração e envio de imagens entre o usuário e o sistema via rede WI-FI.

2.3 Metodologia

O projeto consiste em um sistema de automação do telescópio através do controle da velocidade de rotação dos dois motores a partir da informação da localização onde será feita a observação astronômica. A introdução destes mecanismos envolve a integração de uma rede Wi-Fi com protocolo TCP-IP, a estrutura física do telescópio capturando a imagem, a câmera recebendo essa imagem, os dois motores acompanhando o movimento aparente das estrelas e, por fim, um *hardware* para sistemas embarcados coordenando todos esses processos.

A primeira etapa é analisar as alternativas de construção da montagem física. Uma vez escolhida e implementada, será necessário implementar um software de controle do

sistema de localização e acompanhamento, que será realizado no sistema embarcado em linguagem C.

A segunda etapa deste trabalho consiste na comunicação entre o telescópio e o sistema embarcado. O telescópio será apontado para uma região do céu onde se encontram os objetos que serão observados e então um sensor de captura de imagem irá receber essa imagem. Esses dados serão transmitidos do sensor para o sistema embarcado, onde são processados e armazenados.

Com a captura e armazenamento da imagem realizadas, torna-se fundamental a transmissão destes dados. Para isso, será implementado um servidor TCP no sistema, habilitando o acesso remoto via Wi-Fi para configuração e transmissão de imagens. Através do acesso remoto na rede, o usuário informa a localização onde será realizada a observação astronômica e a partir dessas coordenadas o sistema irá realizar o cálculo das velocidades de rotação dos motores no eixo horizontal e vertical.

Ao unificar a tecnologia de captura de imagem, comunicação em rede e calibração dos motores, este projeto automatiza a observação astronômica proposta inicialmente. Deste modo o projeto confere um sistema dinâmico e flexível de modo que o telescópio possa acompanhar as estrelas independente da sua localização além de proporcionar maior acessibilidade com a implementação da comunicação em rede. No diagrama da Figura 1 pode ser observado a estrutura do projeto completo.

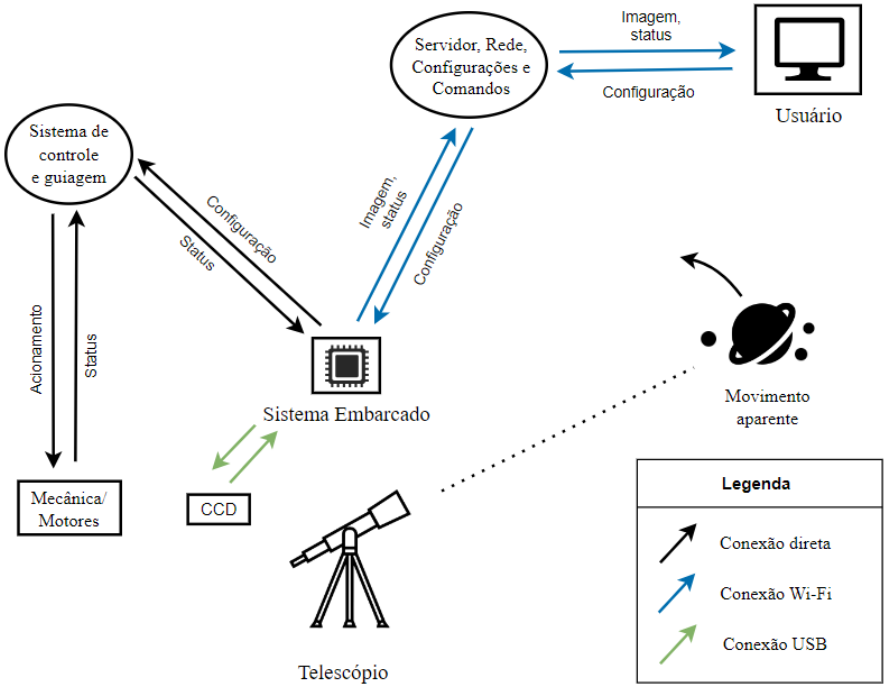


Figura 1 – Diagrama do projeto do telescópio

3 REVISÃO BIBLIOGRÁFICA

3.1 Telescópio

Um telescópio é um instrumento óptico que amplia a imagem de objetos distantes, utilizando lentes ou espelhos para coletar e focalizar a luz. Ele é amplamente utilizado na astronomia para observar corpos celestes como estrelas, planetas e galáxias.

Na astronomia, existem três tipos principais de telescópios: refratores, refletores e catadióptricos. Neste projeto, será utilizado um telescópio refletor, que inclui subtipos como o Telescópio de Newton, Telescópio Cassegrain, Telescópio de Gregory e Telescópio Schiefspiegler. O modelo que será utilizado ao longo do projeo é um telescópio newtoniano.

Inicialmente a luz passa para o tubo até ser refletida por um espelho côncavo localizado na base do tubo. Esse espelho reúne e focaliza os raios de luz em um ponto de convergência. A luz refletida pelo espelho côncavo é novamente refletida por um pequeno espelho plano, denominado espelho secundário, redireciona a luz para uma abertura lateral do tubo, onde está localizada a ocular [4].

Os espelhos utilizados nesse tipo de telescópio oferecem várias vantagens sobre as lentes, principalmente em telescópios de grande diâmetro. Como os espelhos podem ser suportados por trás, eles não sofrem o arqueamento que afeta as grandes lentes, eliminando um dos problemas ópticos. Além disso, a luz refletida por um espelho não sofre dispersão como a luz que passa por uma lente, o que praticamente elimina os problemas de cromática, comuns em telescópios refratores. Construir grandes espelhos é geralmente mais barato e tecnicamente mais viável do que fabricar grandes lentes, pois os espelhos podem ser feitos em segmentos menores e montados juntos, enquanto as lentes grandes precisam ser fabricadas inteiras, aumentando os custos e a complexidade [4].

Na Figura 2 pode ser observado o esquemático do sistema de espelhos. $E1$ representa o espelho côncavo e $E2$ o espelho reto. O ponto F representa o ponto focal do espelho $E1$.

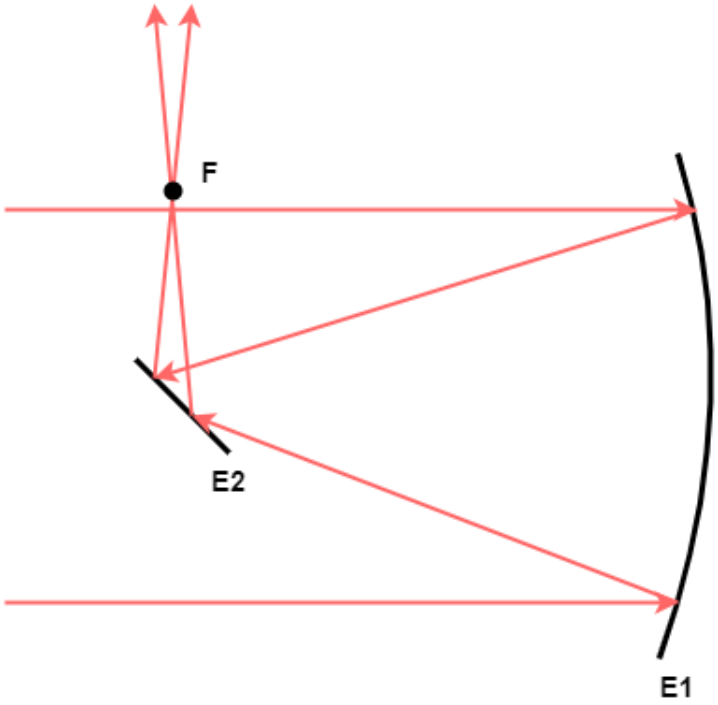


Figura 2 – Funcionamento de um telescópio newtoniano

3.2 Tipos de montagem

Para a observação dos astros no céu, quanto maior for a ampliação do telescópio, menor será o campo de visão, logo uma parte menor do céu estará visível ao observador, sendo assim o astro permanecerá por menos tempo em seu campo de visão. Isso ocorre devido ao efeito de rotação terrestre, sendo então necessário a escolha de um tipo de montagem a ser utilizada no telescópio, para que o astro em questão sempre fique no campo de visão do observador.

A montagem a ser utilizada no telescópio deve ser escolhida cautelosamente, pois ela deve proporcionar movimentos suaves tanto nos eixos verticais quanto nos horizontais, deve ser rígido e não permitir vibrações que prejudique a visualização dos astros, além de permitir que o tubo do telescópio fique equilibrado [5].

A primeira montagem é a azimutal, onde a recentralização ocorre com a ajuda de dois eixos [6], portanto ela necessita da utilização de dois motores. A segunda montagem é a equatorial, onde a recentralização ocorre com a ajuda de apenas um eixo [6], sendo necessário a utilização de um motor. Devido a isso, o alinhamento do segundo eixo do telescópio deve ser feito manualmente, com a base de rotação do telescópio sendo inclinado

de acordo com o ângulo da posição em relação ao equador. Além disso, a montagem deve ser orientada para o Sul/Norte verdadeiro, de modo que simule a colocação do sistema exatamente sobre o equador.

Também é preciso destacar a necessidade de uma estrutura firme de modo que não haja vibração durante o registro dos objetos pelo telescópio [6]. Caso a montagem não seja rígida, a imagem capturada será prejudicada devido às oscilações causadas pela falta de consistência mecânica da estrutura, tornando o processo ineficiente.

Para o desenvolvimento do projeto será utilizada a montagem do tipo azimutal, tendo em vista que o objetivo é o seguimento de astros com o usuário se conectando ao telescópio de forma remota. Portanto, como essa montagem não necessita de um ajuste manual ela será a que mais se encaixa nos objetivos do projeto. Dentro desse tipo de montagem existem inúmeros subgrupos, sendo que dos materiais disponíveis para a construção do tipo da montagem e por se tratar de um telescópio pequeno, o tipo que mais se adequa ao projeto foi a montagem de garfo.

Uma das maiores vantagens desse tipo de montagem é o fato de não ser necessário o uso de contrapeso, além de ser um tipo de montagem mais compacta e ideal para telescópios de pequeno porte. Esse tipo de montagem consiste em um suporte motorizado que segura o tubo com dois braços como se fosse uma pinça. Tanto os braços como a base dessa montagem são controladas por um motor de passo. Como qualquer montagem, ela possui alguns pontos fracos, sendo a primeira dela que o telescópio é muito suscetível a balançar no eixo polar, além de precisar a ativação dos dois motores ao mesmo tempo para fazer o rastreamento dos astros, assim como a utilização de braços firmes e rígidos. Essa montagem pode ser vista na Figura 3.

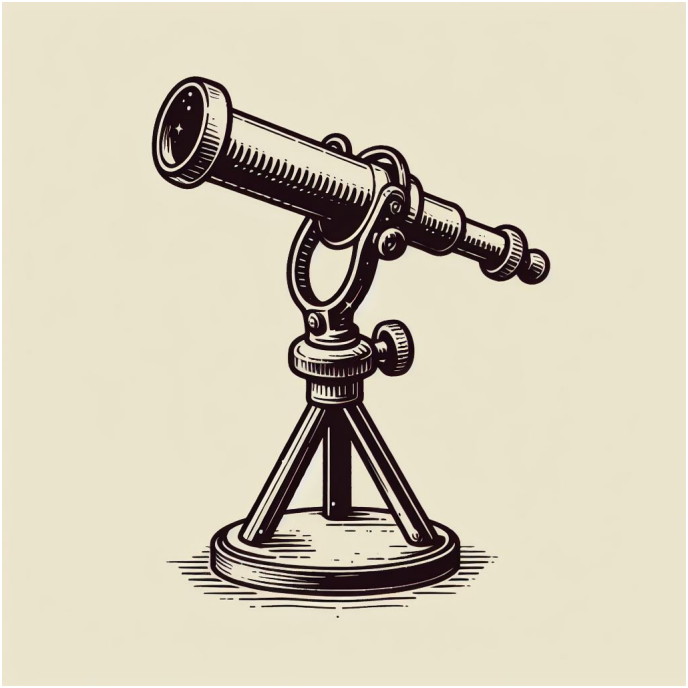


Figura 3 – Montagem de Garfo.

3.3 Mecânica

O sistema mecânico desenvolvido neste projeto utiliza dois motores de passo responsáveis pela rotação dos eixos vertical e horizontal do telescópio. Adicionalmente, será empregado um conjunto de engrenagens cilíndricas de dentes retos, que auxiliará na execução desse movimento. Tais engrenagens são amplamente utilizadas em sistemas que necessitam alterar a velocidade de rotação e transmitir potência entre diferentes eixos. Esse tipo de sistema oferece uma relação de velocidade precisa e é adequado para aplicações que envolvem altas potências [7].

Uma das principais razões para o uso de engrenagens de dentes retos é a capacidade de proporcionar uma velocidade de rotação uniforme. Para assegurar essa característica, é necessário que os dentes das engrenagens sejam projetados conforme a lei fundamental do engrenamento, que determina que a normal comum no ponto de contato entre duas engrenagens deve sempre passar por um ponto fixo situado na linha que conecta os centros das engrenagens [7]. Esse ponto fixo, denominado ponto de passo (*pitch point*), encontra-se na tangente entre os círculos primitivos. A Figura 4 ilustra a normal comum e o ponto fixo P. Os pontos O_1 e O_2 indicam os centros das engrenagens, o ponto A corresponde à área de contato entre os dentes, r_1 e r_2 são os raios primitivos, e n_1 e n_2 representam as

velocidades em RPM.

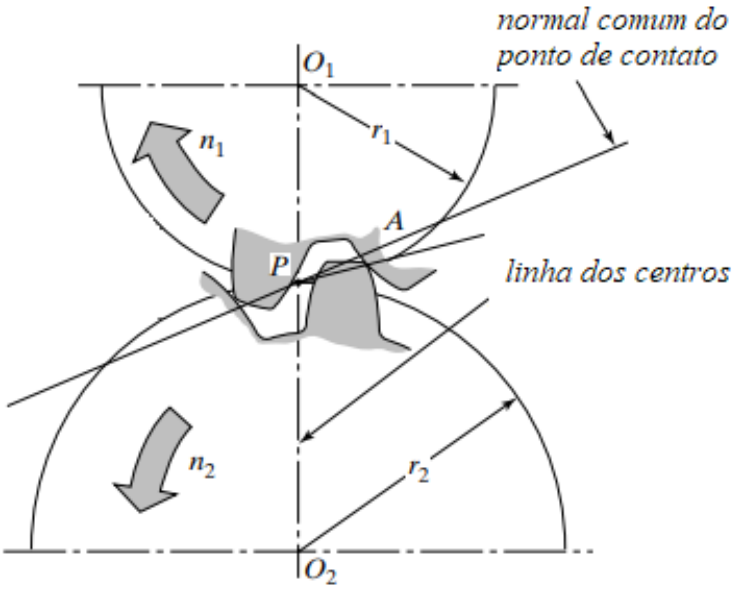


Figura 4 – Lei do engrenamento para transmissão de duas engrenagens

Fonte: adaptado de [7]

A normal comum é determinada pela reta perpendicular à linha tangente comum formada pelas superfícies dos dentes no ponto de contato entre as engrenagens. Essa reta tangencia simultaneamente os círculos de base das duas engrenagens em contato. Esta linha, juntamente com a reta perpendicular à linha dos centros, define o ângulo de pressão [7]. Este conceito é ilustrado na Figura 5. A reta tt é tangente formada pelas superfícies dos dentes no ponto de contato, e a reta nn a perpendicular de tt. O ponto A é o ponto de contato entre as engrenagens e P o ponto onde as circunferências do raio primitivo se tangenciam. Já θ é o ângulo de pressão.

Um dos principais parâmetros da engrenagem é o raio primitivo e ele é fundamental para entender o funcionamento das engrenagens. Ele representa o raio do círculo hipotético que toca o ponto de contato entre duas engrenagens. Quando as engrenagens estão em contato, esses círculos primitivos rolam um sobre o outro, o que determina a velocidade e o movimento de cada engrenagem. Esse conceito é essencial para projetar sistemas de engrenagens eficientes e precisos.

Uma das relações mais relevantes dentro de um sistema de engrenagens é a relação

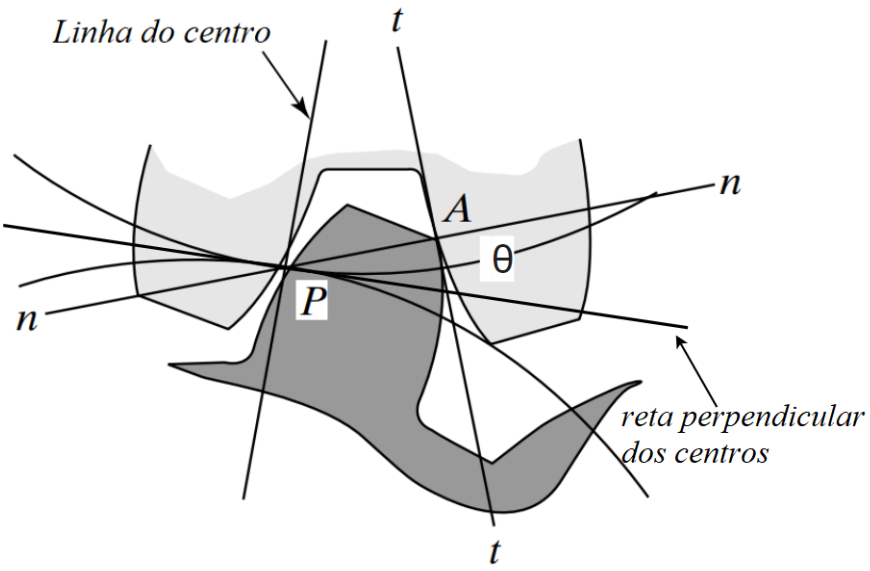


Figura 5 – Ângulo de pressão.

Fonte: adaptado de [7]

de transmissão dada por:

$$i = \frac{w_2}{w_1} = \frac{n_2}{n_1} = \frac{z_1}{z_2} = \frac{r_1}{r_2} \tag{3.1}$$

onde i é a relação de transmissão, w representa a velocidade angular, n a velocidade em RPM, z o número de dentes, r o raio primitivo da engrenagem. O índice 1 se refere a primeira engrenagem, enquanto que o índice 2 se refere a segunda engrenagem.

Utilizando (3.1) é possível a determinação de parâmetros de uma engrenagem específica, tais como o número de dentes, velocidade angular e raio primitivo, com base na relação e nos parâmetros da outra engrenagem à qual está conectada.

Para o funcionamento correto do sistema de engrenagens, é necessário que atenda alguns critérios: as engrenagens tenham um mesmo módulo e ângulo de pressão.

Para o projeto dessas engrenagens é necessário conhecer os parâmetros envolvidos para sua construção. O primeiro parâmetro a ser apresentado será o módulo (m). Ele é uma referência para o dimensionamento da engrenagem, associado diretamente com o tamanho da engrenagem, sendo utilizado para determinar o valor do diâmetro primitivo. Na Tabela 1 podem ser vistos alguns valores padrões de módulos utilizados [7].

Tabela 1 – Valores padrão de módulo

módulo																		
0,3	0,5	1	1,25	1,5	2	2,5	3	4	5	6	8	10	12	16	20	25	32	40	50

Outro parâmetro relevante no projeto de engrenagens é o ângulo de pressão (θ). Esse ângulo determina a dimensão do raio de base que será utilizado para definir a curvatura dos dentes da engrenagem. Os valores mais comuns para esse parâmetro são $14\frac{1}{4}^{\circ}$, 20° e 25° [7].

Engrenagens cilíndricas de dentes retos são caracterizadas por possuírem dentes lineares dispostos paralelamente ao eixo de rotação. Para projetar esse tipo de engrenagem, é fundamental compreender alguns diâmetros de circunferências que compõem a peça. O principal deles é o diâmetro primitivo (d_p), que corresponde a uma circunferência teórica equivalente a uma roda de fricção. Quando duas engrenagens estão em contato, os círculos primitivos podem ser considerados como representações de uma transmissão por atrito entre dois cilindros. Esse parâmetro é essencial, pois serve como base para calcular outros diâmetros e a altura dos dentes. O valor de d_p pode ser determinado pela seguinte equação:

$$d_p = mz \tag{3.2}$$

onde z representa o número de dentes e m o módulo da engrenagem.

O segundo é o diâmetro de base (d_b), parâmetro do círculo imaginário utilizado para a projeção das curvaturas dos dentes da engrenagem, determinado por:

$$d_b = d_p \cos(\theta) \tag{3.3}$$

Por fim, tem-se o diâmetro externo (d_e) e interno (d_i) que correspondem ao limite superior e inferior dos dentes da engrenagem respectivamente denotados por:

$$d_e = m\,(z + 2) \tag{3.4}$$

$$d_i = m\ \ (z - 2,5) \tag{3.5}$$

A largura lateral de uma engrenagem pode ser qualquer valor que obedeça a relação:

$$(\frac{20}{6})m \leq l \leq 50 \tag{3.6}$$

Já o ângulo do dente pode ser calculado por:

$$\alpha = \frac{90}{m} \tag{3.7}$$

É importante ressaltar que as expressões apresentadas em (3.2)-(3.7) foram apresentados por Almeida, et all [8] e a unidade dos parâmetros de distâncias são dados em mm e os ângulos em graus. A Figura 6 ilustra os principais parâmetros de uma engrenagem. Os parâmetros a, b e h representam a altura da cabeça, altura de raíz e a altura do dente respectivamente.

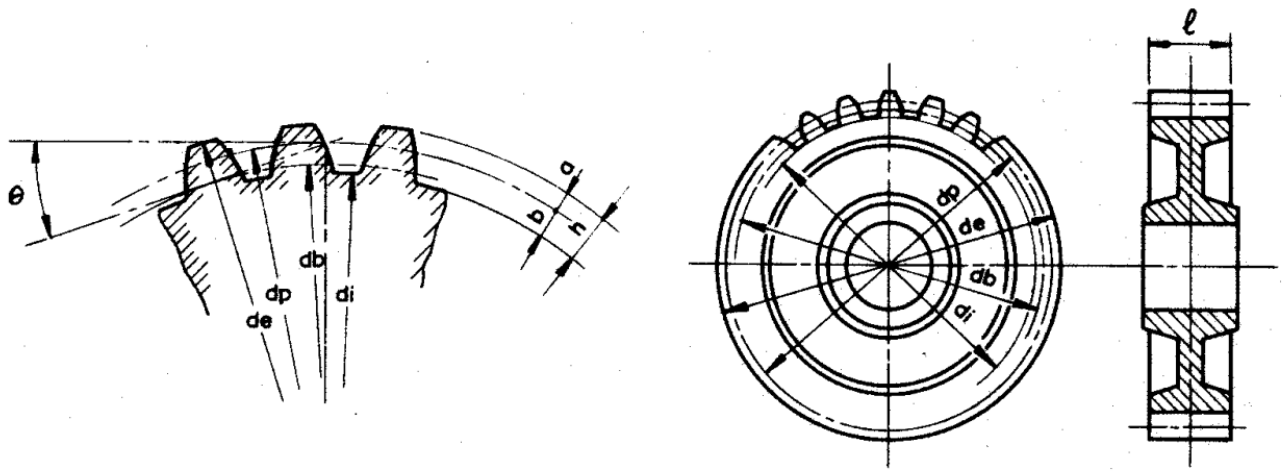


Figura 6 – Parâmetros da engrenagem.

Fonte: adaptado de Provenza [9]

Para o traçado aproximado das curvas dos dentes de uma engrenagem, em [9] foram aprersentadas três expressões para determinar o raio de curvatura. Isso pode ser observado em (3.8), (3.9) e (3.10). Esses raios podem ser observados na Figura 7. A primeira expressão é dada por:

$$r = fm \tag{3.8}$$

onde r é o raio do dente entre o diâmetro externo e o diâmetro primitivo, f coeficiente que pode ser consultado na Tabela 2, m é o módulo.

A segunda expressão é dada por:

$$r_1 = f'm \tag{3.9}$$

onde r_1 é o raio do dente entre o diâmetro primitivo e o diâmetro de base, f' coeficiente que pode ser consultado na Tabela 2.

A terceira expressão é dada por:

$$r_3 = \frac{m}{6} \tag{3.10}$$

onde r_3 é o raio no extremo inferior do dente.

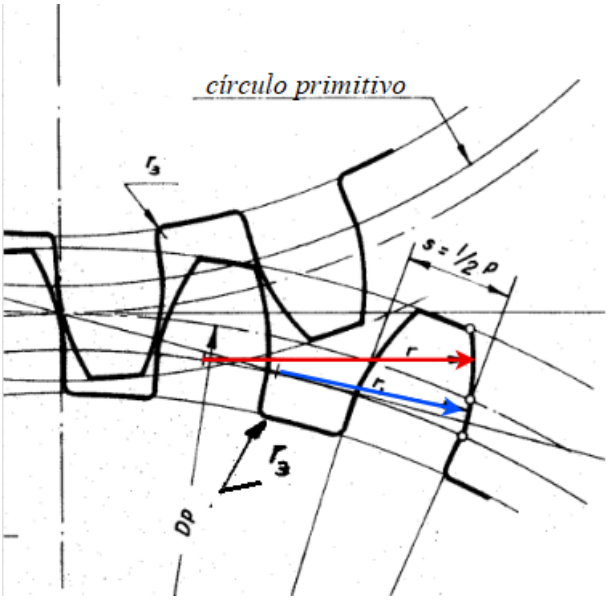


Figura 7 – Raio da curvatura dos dentes

Fonte: baseado por Provenza [9]

Os coeficientes f e f' dados em (3.8) e (3.9) também são apresentados por Provenza [9], mostrados em função do número de dentes. Esses valores podem ser consultados na Tabela 2.

Um dos fenômenos que pode prejudicar o sistema de transmissão é a interferência. Esse fenômeno ocorre quando o círculo externo de uma engrenagem intersecta a normal

Tabela 2 – Coeficientes f e f'

z	f	f'	z	f	f'	z	f	f'
8	2,10	0,45	21	3,41	1,92	34	4,33	3,09
10	2,28	0,69	22	3,49	2,06	35	4,36	3,16
11	2,40	0,83	23	3,57	2,15	36	4,45	3,23
12	2,51	0,96	24	3,64	2,24	37-40	4,20	
13	2,62	1,09	25	3,71	2,33	41-45	4,63	
14	2,72	1,22	26	3,78	2,42	46-51	5,06	
15	2,82	1,34	27	3,85	2,50	52-60	5,74	
16	2,92	1,46	28	3,92	2,59	61-70	6,52	
17	3,02	1,58	29	3,99	2,67	71-90	7,72	
18	3,12	1,69	30	4,06	2,76	91-120	7,78	
19	3,22	1,79	32	4,20	2,93	121-180	13,38	
20	3,32	1,89	33	4,27	3,01	181-360	21,62	

comum fora do segmento da reta que tangencia os dois círculos de base [7]. Quando isso acontece, resulta em uma transmissão que não segue a lei do engrenamento, o que pode prejudicar a transmissão.

Para uma compreensão mais clara, será apresentado um caso de interferência na Figura 8. Nessa figura, pode-se observar a normal comum, o ponto de contato P entre os círculos primitivos, os pontos C e D onde a normal comum tangencia o círculo de base de cada engrenagem, os centros O_1 e O_2 e o ângulo de pressão θ. O contato do dente ocorre no ponto B. Percebe-se que o ponto B está fora do segmento da reta CD, resultando em um caso de interferência.

A interferência pode ser evitada projetando as duas engrenagens de modo que obedeça a seguinte relação:

$$4kz_2 + 4k^2 \leq z_1^2 sin(\theta)^2 + 2z_1 z_2 sin(\theta)^2 \tag{3.11}$$

onde z_2 é o número de dentes da coroa, z_1 representa o número de dentes do pinhão, θ o ângulo de pressão e k a relação entre a altura de cabeça e o módulo dado por $k = \frac{a}{m}$

Como normalmente em (3.11) a altura da cabeça (a) é igual ao módulo, assume-se que $k = 1$. Para alguns casos específicos em que não é igual ao módulo, basta encontrar a altura da cabeça através da relação a seguir:

$$a = \frac{d_e - d_p}{2} \tag{3.12}$$

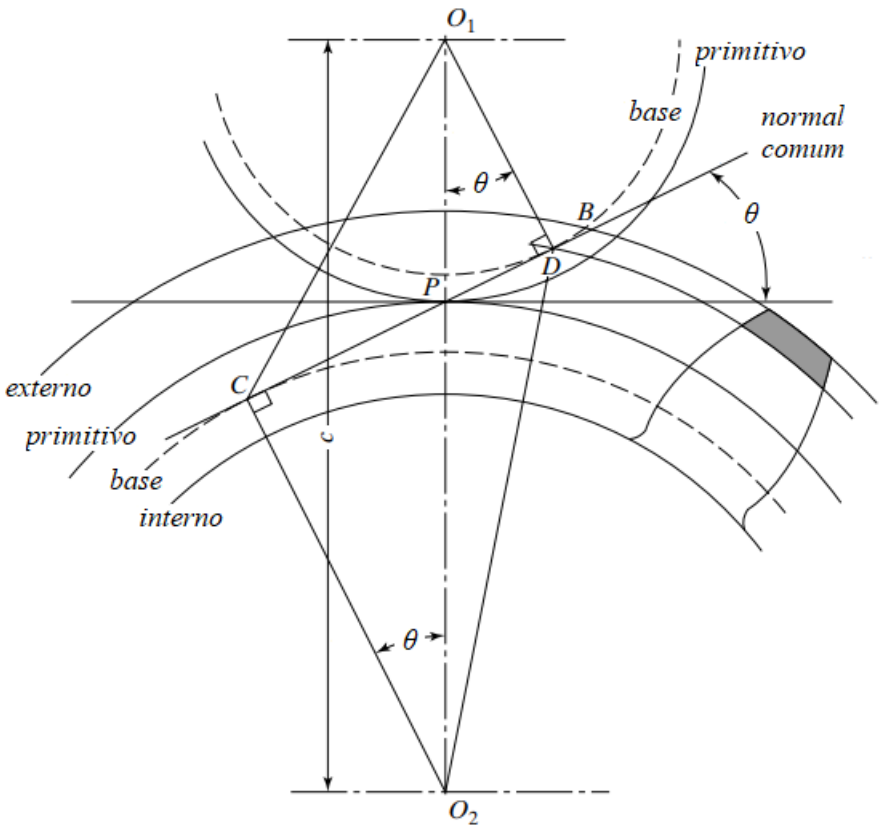


Figura 8 – Interferência entre duas engrenagens.

Fonte: adaptado de Wilson Sadler [7]

Outro tipo de engrenagem além das cilíndricas com dentes retos, são as engrenagens cônicas com dentes retos. Enquanto a engrenagem cilíndrica tem a projeção do seu perfil a partir de um diâmetro primitivo, as engrenagens cônicas são projetados a partir do cone primitivo. Um sistema de duas engrenagens cônicas tem os eixos de rotação se cruzando, deferente da cilíndrica onde os eixos eram paralelos. Normalmente o ângulo formado entre esses eixos é de 90°, mas pode variar conforme a aplicação. A superfície de contato entre duas engrenagem desse tipo se dá através de um cone rolante

A grande vantagem desse tipo de engrenagem é a possibilidade de transmissão de potência em eixos que se intersectam com um ângulo, normalmente de 90°. Isso possibilita a redução de espaços em um sistema de transmissão, já que ela permite orientar os eixos em diferentes angulações. O problema desse das engrenagens cônica é que elas tem limitação de velocidade para sua operação, dado que quando aplicado uma velocidade alta, o sistema de transmissão pode ficar susceptível à ruídos e vibrações.

Para a projeção da engrenagem cônica, é necessário conhecer os parâmetros tanto

do pinhão, quando da coroa, pois os parâmetro de uma das engrenagens influenciam nos parâmetros da outra. Assim como nas engrenagens cilíndricas de dentes retos, inicialmente é definido o número de dentes, o módulo e o ângulo de pressão. Outro parâmetro importante para as engrenagens cônicas é o ângulo formado entre os eixos(λ).

Com esses parâmetros determinados deve-se calcular o diâmetro primitivo pela equação (3.2). Então a partir do diâmetro primitivo e do número de dentes da coroa e do pinhão, é possível determinar o comprimento do cone primitivo e o ângulo formado por esse cone pela expressão (3.13) e (3.14) respectivamente. Na Figura 8 pode ser observado os parâmetros das engrenagens cônicas.

$$L = \sqrt{\frac{d_{p1}^2}{4} + \frac{d_{p2}^2}{4}} \tag{3.13}$$

$$\delta_1 = \arctan\frac{z1}{z2} \qquad \delta_2 = \arctan\frac{z2}{z1} \tag{3.14}$$

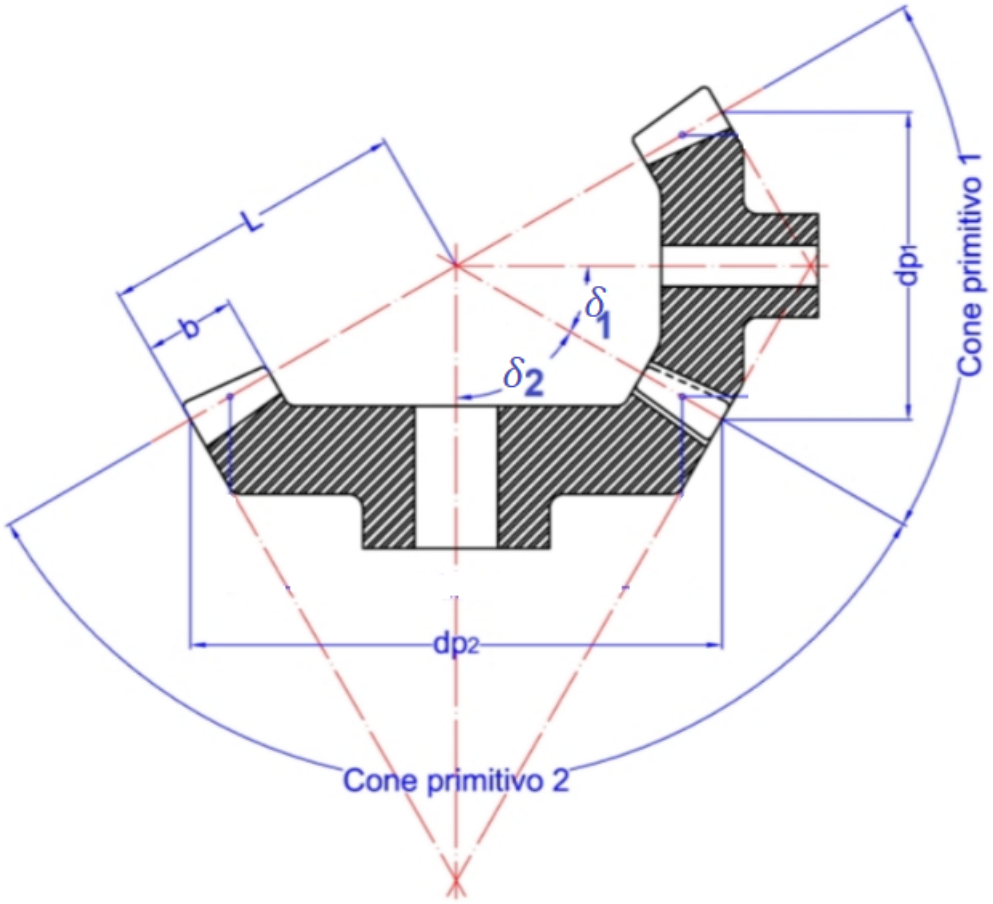


Figura 9 – Parâmetros de uma engrenagem cônica.

Fonte: adaptado de J. C. d. Almeida [8]

Outros parâmetros importantes para a projeção dessa engrenagem é a projeção de largura de face, que pode ser calculada como um terço do comprimento do cone e também a projeção do diâmetro externo que pode ser calculado com a equação (3.15).

$$d_{c1} = d_{p1} + 2m\cos\delta_1 \qquad d_{c2} = d_{p2} + 2m\cos\delta_2 \tag{3.15}$$

3.4 TMC2208

O TMC2208 é um driver utilizado para o controle de motores de passo. Ele é amplamente utilizado em projetos que envolvem motores de passo devido à sua eficiência, baixo ruído e capacidade de operação suave.

O TMC2208 possui dois modos principais de operação: StealthChop e SpreadCycle. O StealthChop proporciona uma operação silenciosa e suave, sendo ideal para aplicações

de baixa velocidade que exigem precisão, como impressoras 3D ou telescópios, onde o ruído é uma preocupação. Já o SpreadCycle oferece maior torque e desempenho em altas velocidades, com controle mais dinâmico do motor, sendo adequado para situações que requerem mais força ou respostas rápidas.

Escolheu-se o *driver* TMC2208 operando no modo StealthChop devido à sua capacidade de fornecer uma operação silenciosa e suave, ideal para o movimento preciso de telescópios. Como o sistema proposto precisa garantir uma movimentação constante e precisa dos motores, evitando vibrações e ruídos que possam interferir na captura de imagens de corpos celestes, o StealthChop é a melhor escolha. Além disso, esse modo é otimizado para baixas velocidades, exatamente o perfil de operação que um telescópio exige para seguir o movimento aparente das estrelas, já que o acompanhamento precisa ser contínuo e sem solavancos, evitando qualquer distorção ou perda de foco durante a observação e captura de imagens.

O TMC2208 possui uma série de pinos de conexão para controlar o motor de passo e configurar suas funcionalidades. Ele se conecta ao motor de passo bipolo através de quatro pinos principais de fase A e B. Esses pinos se conectam diretamente às bobinas do motor de passo. O *driver* controla o fluxo de corrente através desses pinos para mover o motor. Além disso o *driver* também possui os pinos STEP, que recebe pulsos para mover o motor, onde cada pulso corresponde a um movimento de um *microstep*, e DIR, que define o sentido de rotação do motor. Por fim, também será utilizado o pino de *enable* (EN) que é utilizado para ativar ou desativar o driver e os pinos VCC e GND que servem para a conexão a uma fonte de alimentação.

3.5 Apache

O Apache HTTP Server ou Apache, como é comumente conhecido, é um servidor HTTP de código aberto, o que significa que pode ser modificado e distribuído livremente para sistemas operacionais modernos como o Windows e UNIX. O Apache é um projeto da The Apache Software Foundation e foi lançado em 1995. Esse projeto tem como premissa oferecer um servidor seguro, eficiênte e extensível que forneça serviços HTTP [10].

Ele é um dos servidores mais populares e tem um papel essencial na arquitetura de servidores de internet, responsável pela hospedagem de sites, páginas e aplicações webs

implementadas de forma segura, flexível e estável [10].

O Apache trabalha mediando a comunicação entre os navegador e os arquivos hospedados no servidor, processando as requisições HTTP e entregando as páginas web ou arquivos. Para isso, primeiramente é obtido o endereço IP do domínio buscado [10].

3.6 Escalonamento Linux

O Linux classifica as threads em três categorias distintas para o escalonamento:

1. FIFO em tempor real

2. Escalonamento circular em tempo real

3. Tempo compartilhado

Nenhuma dessas classes de threads é considerada realmente de tempo real, pois não permite especificar limites de tempo nem oferece garantias de execução. Essas classes apenas possuem uma prioridade superior em relação aos threads da classe padrão de tempo compartilhado.

Os threads FIFO em tempo real possuem a maior prioridade e, uma vez em execução, não são preemptados, a menos que um novo thread FIFO de tempo real, com prioridade mais alta, esteja pronto para execução.

O sistema FIFO (First In, First Out) é uma estrutura de dados essencial em diversos cenários de sistemas computacionais. Ele adota o princípio de que o primeiro elemento inserido na fila será o primeiro a ser removido, mantendo uma ordem rígida de processamento ou transmissão de dados. FIFO é amplamente utilizado para comunicação entre processos, em sistemas operacionais, redes de computadores e na programação de sistemas embarcados.

No escalonamento FIFO, o processo em execução é completado do início ao fim, sem interrupções. Quando um novo processo chega enquanto outro ainda está em execução, ele é posicionado em uma fila de espera, aguardando sua vez para ser processado pela CPU. Essa fila organiza os processos pela ordem de chegada, garantindo que cada um seja atendido na sequência de sua entrada na fila. [11]

3.7 Threads

Uma thread é uma maneira de dividir um processo ou tarefa de um programa em múltiplas partes que podem ser executadas simultaneamente, permitindo que diferentes atividades sejam realizadas de forma concorrente.

O que as threads trazem para o modelo de processo é a capacidade de realizar múltiplas execuções dentro do mesmo ambiente, mantendo um alto nível de independência entre elas. Executar múltiplas threads em paralelo em um único processo equivale a ter diversos processos rodando de forma simultânea em um computador. [11]

4 MATERIAIS E MÉTODOS

4.1 Materiais utilizados

A seguir, apresenta-se a lista dos recursos necessários, que inclui tanto os materiais físicos quanto as ferramentas de software para o desenvolvimento do projeto:

- Impressora 3D FDM (engrenagens, suporte para motor, caixa do raspberry)
- 2 x Motor de passo NEMA17
- 2 x Caixa de redução
- 2 x TMC2208
- 1 x Telescópio MEADE ETX90
- 1 x Tripé
- 1 x Suporte para rotação do telescópio em 2 eixos
- 1 x Raspberry pi 4b
- 1 x Buscador
- 1 x Dispositivo de captura de imagem
- 1 x Cabos para microHDMI para HDMI
- 1 x Monitor
- 1 x Cartão micro SD
- 1 x fonte de alimentação com entrada USB-C
- 1 x fonte de alimentação CC
- limguagem C
- Compilador do C
- SolidWorks
- Sistema Operacional Linux (Ubuntu)

- Apache2
- ffmpef e v4l2 comunicação e configuração da câmera

4.2 Desenvolvimentos de peças

O objetivo do projeto é seguir o movimento da Terra com precisão. Para garantir esse acompanhamento preciso, o telescópio deve ter uma velocidade angular correspondente ao movimento da Terra em torno de seu próprio eixo. Enquanto a Terra gira em um sentido, o telescópio deve se mover no sentido contrário. Isso resultará em uma velocidade angular nula, criando a ilusão de que o objeto celeste está estático quando observado diretamente pelo telescópio.

Para realizar o cálculo da velocidade angular de rotação da Terra foi utilizado o tempo que era necessário para uma volta completa (23 horas, 56 minutos e 4 segundos). Com esse valor, foi possível chegar a uma velocidade angular $7,29.10^{-5} rad/s$ ou $6,963.10^{-4} RMP$. Essa velocidade representa o limite máximo de rotação do motor necessário para acompanhar o movimento aparente dos corpos celestes. No entanto, dado que o projeto busca uma aplicação geral, inclusive aqueles fora da linha do equador, a velocidade de rotação do motor poderá assumir um valor menor que $6,963.10^{-4} RMP$, já considerado um valor extremamente baixo.

A fim de diminuir a velocidade de rotação do motor, será implementado um sistema de engrenagens. Ao aumentar a relação de transmissão, obtém-se maior precisão no sistema. Por exemplo, se o motor de passo tiver um ângulo de passo de $1,8°$ com uma relação de transmissão de $i = 10$, a precisão do sistema será de $0,18°$. Outro fator que contribui para a implementação das engrenagens é o fato de que esse sistema oferece maior estabilidade, uma vez que o design dos dentes é projetado para reduzir a vibração, proporcionando uma transmissão suave e uniforme.

Inicialmente será feito o dimensionamento das engrenagens que serão utilizadas. Para isso, deve-se escolher alguns parâmetros iniciais. Para o projeto das engrenagens foi selecionado um módulo de $m = 1$ e um ângulo de pressão $\theta = 20°$. Com isso determinado, manipulou-se (3.11) de modo que isolasse o número de dentes da coroa (z_2) em função do

número de dentes do pinhão z_1 resultando em (4.1).

$$z_2 \leq \frac{z_1^2 sin(\theta)^2 - 4k^2}{4k - 2z_1 sin(\theta)^2} \tag{4.1}$$

A curva de (4.1) pode ser observada na Figura 10. Avaliando os pontos gerados, percebe-se que o número mínimo de dentes do pinhão que gera a maior relação de transmissão foi de $z_1(min) = 17$, que foi escolhido para uso no projeto. Arbitrariamente, adotou-se um número de dentes da coroa de $z_2 = 120$, para que não resultasse em uma engrenagem muito grande.

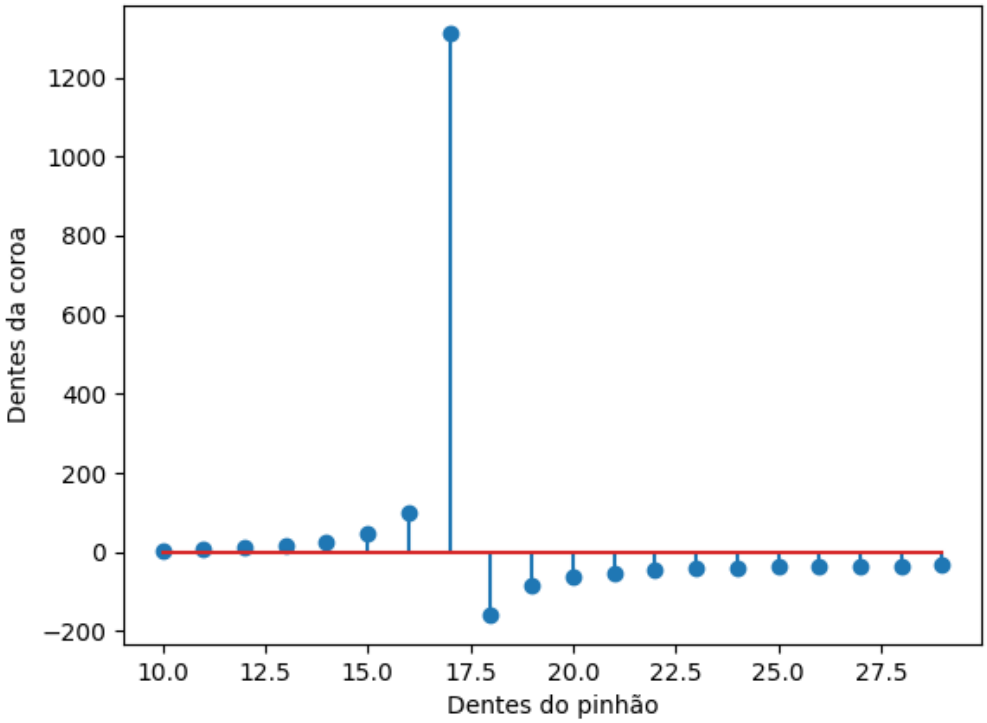


Figura 10 – Máximo de dentes de uma coroa para evitar interferência

Utilizando (3.2)-(3.10) foram calculados os parâmetros das engrenagens a serem utilizados. Na Tabela 3 são apresentados esse valores calculados.

Na sequência foi feito o cálculo da relação de transmissão para o sistema de engrenagens utilizando (3.1). O sistema inclui duas engrenagens de tamanhos diferentes e uma caixa de redução, então a relação de transmissão equivalente será dada pela multiplicação da relação da caixa de transmissão pela relação das engrenagens. A caixa de

Tabela 3 – Parâmetros das engrenagens cilíndricas calculadas calculados

z=17				**z=120**			
d_p	$17mm$	α	$5,294°$	d_p	$120mm$	α	$0.75°$
d_e	$19mm$	r	$3,02mm$	d_e	$122mm$	r	$7,78mm$
d_i	$14,5mm$	r_1	$1,58mm$	d_i	$117,5mm$	r_1	$7,78mm$
d_b	$15,975mm$	r_3	$0,167mm$	d_b	$112,763mm$	r_3	$0,167mm$
l	$10mm$	-	-	l	$10mm$	-	-

redução foi implementada para garantir uma construção compacta com tamanho pequeno quando comparado ao tamanho do tubo do telescópio. Ela é um dispositivo mecânico utilizado para diminuir a velocidade de rotação de um eixo motor para o eixo de saída, sendo esse processo realizado por um conjunto de engrenagens que localizada em seu interior.

O esquemático completo do sistema de engrenagens para a rotação do eixo horizontal, que corresponde ao eixo em que os braços que apoia o telescópio, pode ser visto na Figura 11.

Já para o sistema de rotação do eixo vertical, eixo esse que corresponde à base onde o telescópio e os braços estão fixados, foi projetada outra estrutura. Utilizou-se engrenagens que já vinham junto com o suporte de rotação do sistema, além de implementar duas engrenagens cônicas e uma caixa de redução com um intuito de se ter um mecanismo mais compacto. Os número de dentes das engrenagens incluídas com o suporte são de 18 e 206 dentes.

Para as engrenagens cônicas utilizadas nesse sistema, foram feitos o cálculos para projetá-las. Foram utilizados as expressões (3.2) e (3.13)-(3.15) com os resultados sendo mostrados na Tabela 4. Para essas engrenagens, optou-se por utilizar um módulo de 1 e um ângulo de pressão de 20°, assim como na engrenagem cilíndrica, além de utilizar os mesmos parâmetros para ambas engrenagens, uma vez que elas foram utilizadas apenas para reduzir o tamanho do sistema de transmissão.

Tabela 4 – Parâmetros das engrenagens cônicas calculados

z=20			
d_p	$20mm$	L	$14.14mm$
σ	$45°$	d_c	$21,41mm$
b	$4,71mm$	–	–

O esquemático completo do sistema de engrenagens para a rotação do eixo vertical

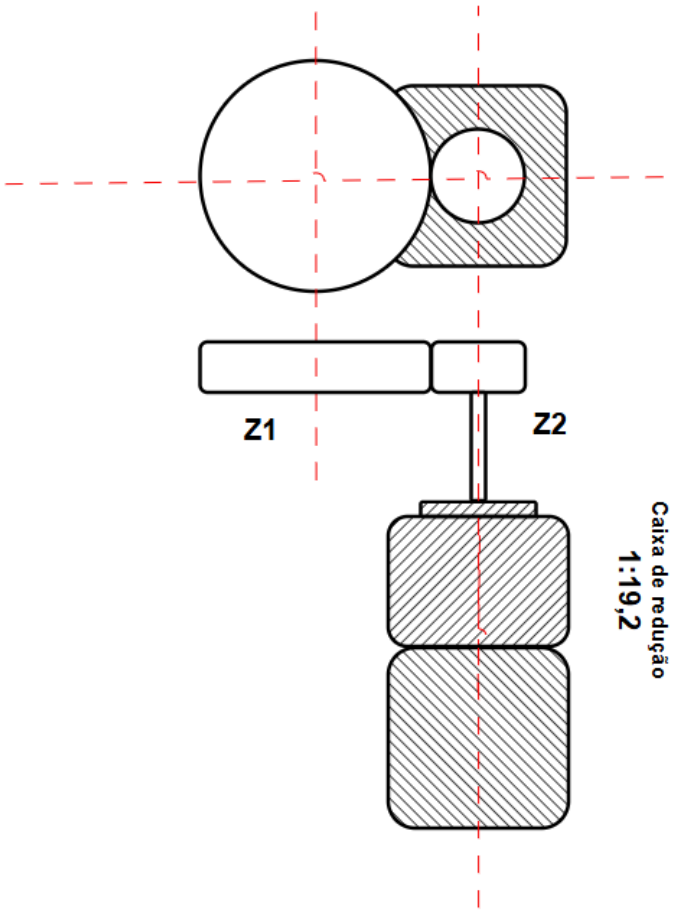


Figura 11 – Sistema de engrenagem do eixo horizontal

pode ser visto na Figura 12. z_1 e z_2 são as engrenagens que já estavam incluídos junto com a base enquanto z_3 e z_4 são as engrenagens cônicas projetadas.

Para o desenvolvimento e a criação das engrenagens foi utilizado o Solidworks, software utilizado para a criação de designs e modelos 3D. Os resultados obtidos podem ser observados nas Figuras 13 e 14.

Com os modelos 3D das engrenagens pronto, o arquivo foi configurado no UltimakerCura, que é um aplicativo que converte modelos 3D em instruções para comunicação com a impressora 3D, dividindo o modelo em camadas finas e gerando o caminho exato que o bico da impressora seguirá para criar a peça. Por se tratar de engrenagens que possuem pequenos detalhes, a impressora foi configurada para imprimir em velocidade reduziada, de modo a garantir a qualidade da peça. Por fim, o material escolhido para compor a peça foi o PLA com a impressora Hurakan da marca BigTreeTech.

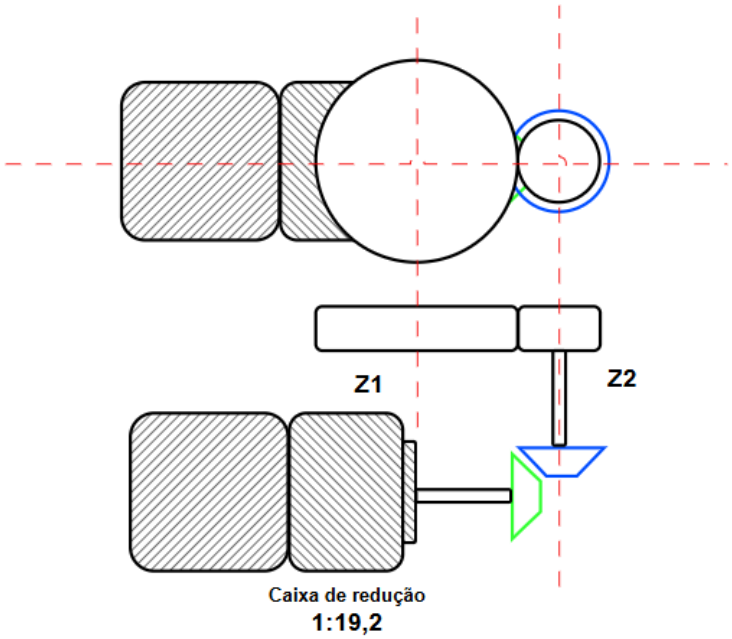


Figura 12 – Sistema de engrenagem do eixo vertical

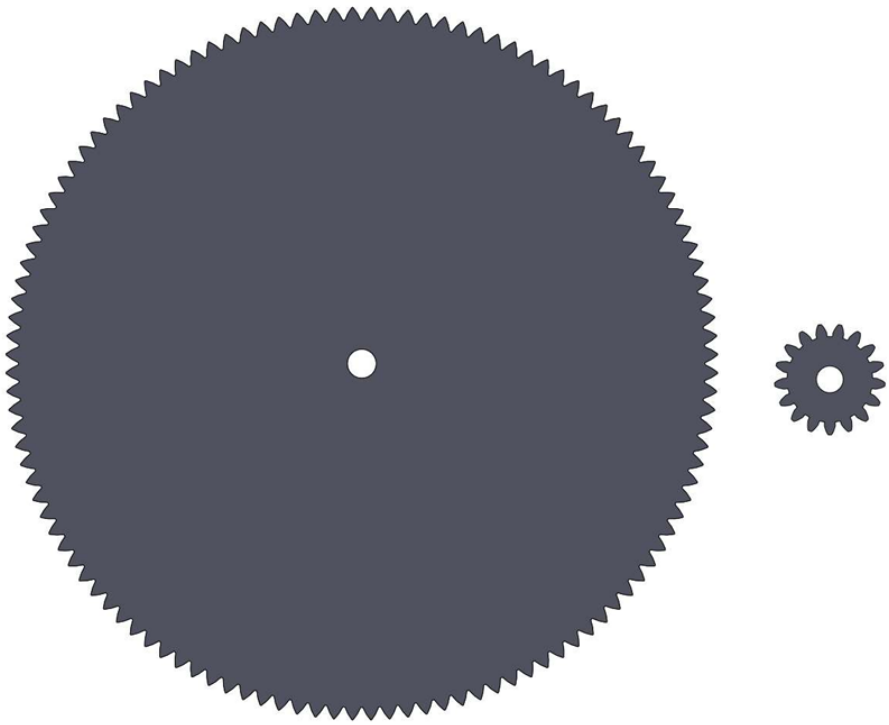


Figura 13 – Engrenagens projetadas no SolidWorks

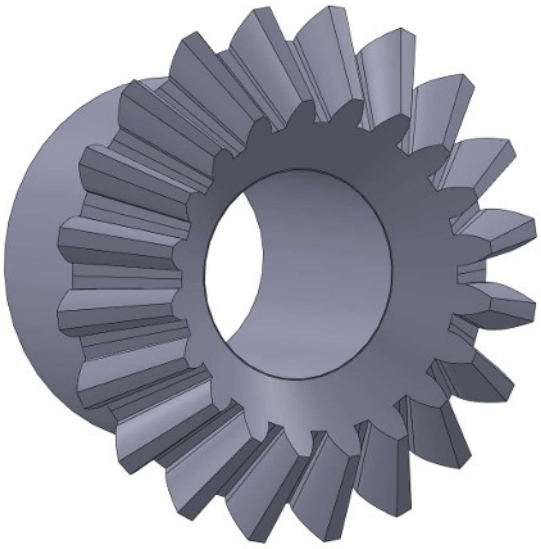


Figura 14 – Engrenagens projetadas no SolidWorks

Com todas as engrenagens projetadas, seria necessário projetar agora outras peças para acoplar esse sistema de transmissão à montagem do projeto. Para o eixo vertical, não foi necessário nenhuma peça adicional, utilizando apenas a estrutura já existente como pode ser visto na Figura 15. Já para o eixo horizontal foi projetado um suporte para apoiar o motor e o redutor em um dos braços da montagem. Esse suporte seria encaixado entre a base giratória e o braço da montagem e o seu modelo 3D pode ser visto na Figura 16.

Para corrigir o desnível causado pelo suporte, foi projetado uma outra peça com a mesma espessura para ser colocado entre o suporte giratório e o braço da montagem. O modelo 3D dessa peça pode ser observado na Figura 17. Essa peça foi projetada para acomodar o motor de passo acoplado na caixa redutora, além de incluir um espaço vazio para poder realizar o desacoplamento do motor do sistema de engrenagens, possibilitando o ajuste manual do eixo. Para realizar esse acoplamento e desacoplamento do motor, será utilizado um gancho com um furo, assim como observado na Figura 18. O gancho possibilita a retirado do motor da engrenagem, destravando o eixo horizontal, permitindo que seja feito o ajuste manual. Para prende-lo novamente, basta mover o motor para frente com o gancho e após chegar no final, pode travá-lo com uma haste cilíndrica no furo existente no gancho. A planta da montagem do eixo horizontal está sendo ilustrado na Figura 19.

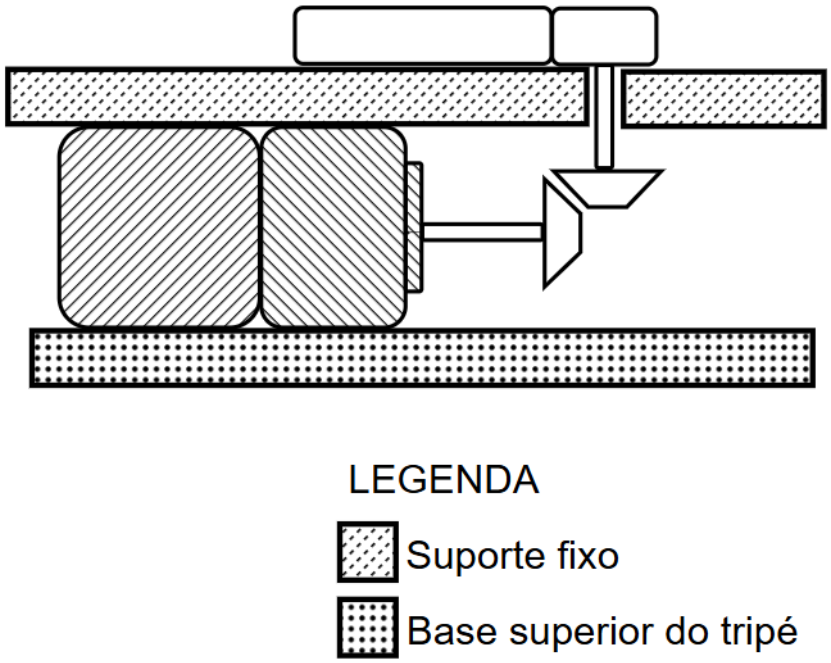


Figura 15 – planta da montagem do eixo vertical

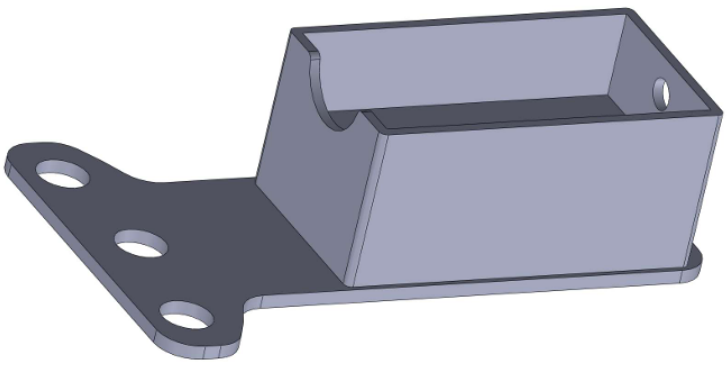
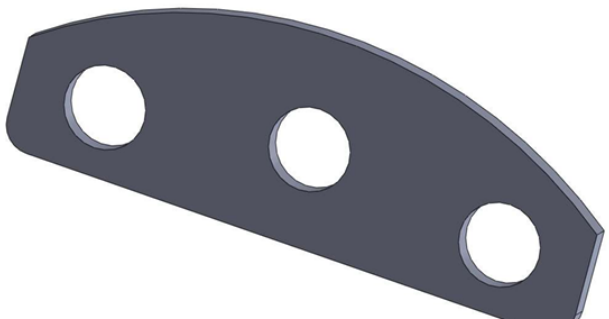


Figura 16 – Suporte para o motor



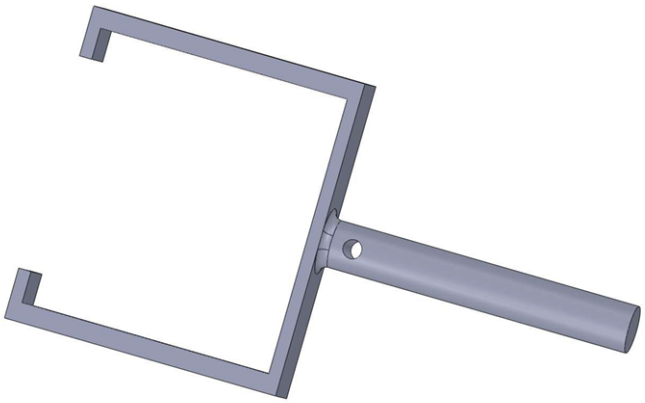


Figura 18 – Gancho para acoplar e desacoplar o motor do sistema

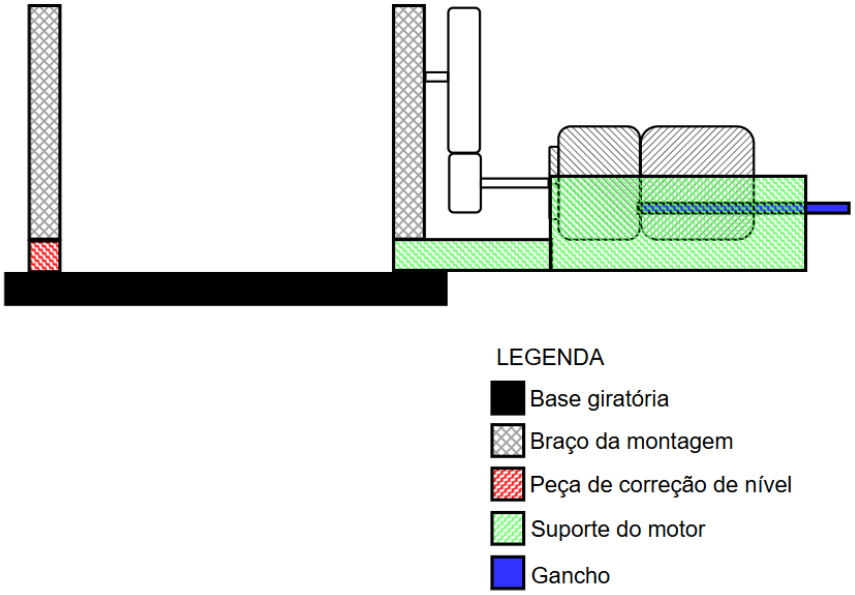


Figura 19 – Planta da montagem do eixo horizontal

A última peça impressa, foi um *case* para proteção do raspberry. O seu modelo 3D está ilustrado nas Figuras 20 e 21. Observa-se pela figura que existem alguns cortes nas duas peças. Elas são para entradas/saída do dispositivo e também para ventilação do *hardware*.

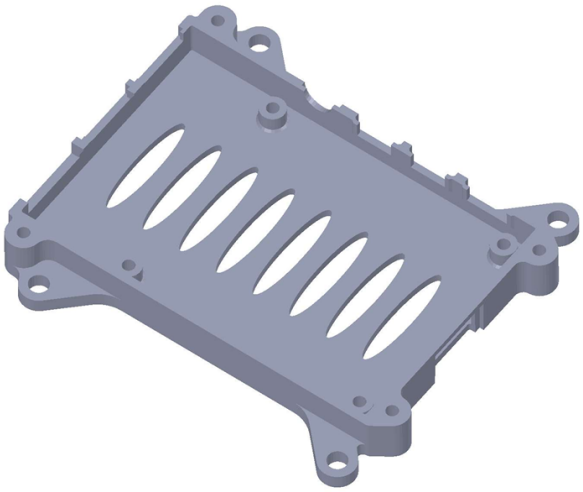


Figura 20 – Case inferior do raspberry

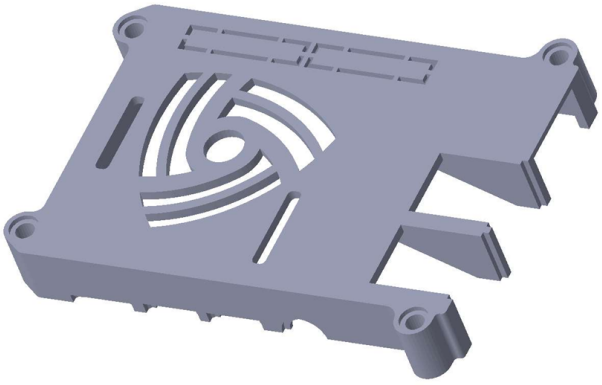


Figura 21 – Case superior do raspberry

Na Figura 22 pode ser observado todos os componentes criados.

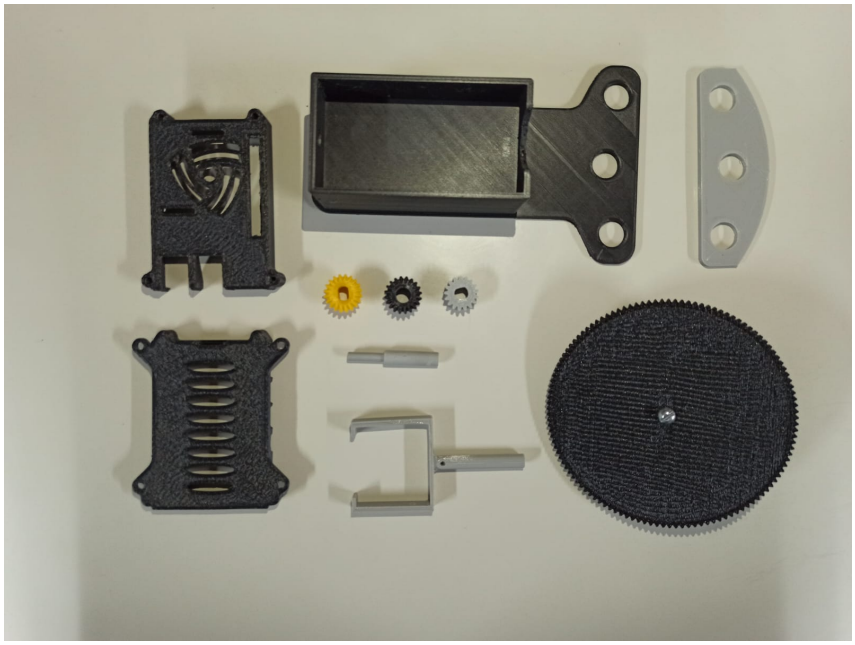


Figura 22 – Peças desenvolvidas com uma impressora 3D

4.3 Montagem do sistema mecânico

Para a montagem do telescópio, foram utilizadas suas peças originais bem como algumas outras peças projetadas, com ajustes realizados conforme necessário. Inicialmente, o motor da base foi substituído por um motor de passo, a fim de oferecer maior variedade de velocidades, já que o motor original proporciona apenas uma velocidade angular, correspondente à rotação terrestre, além disso os motores de passo apresentam movimentos mais precisos. É importante ressaltar que as engrenagens da base foram mantidas, pois foram projetadas especificamente para este telescópio e se encaixavam perfeitamente. Já para a rotação do eixo horizontal, foi implementado o sistema de engrenagens das Figuras 11 e 19.

O motor selecionado para integrar o sistema mecânico do projeto é o NEMA 17, modelo 17hs4401, devido ao seu ângulo de passo relativamente pequeno (1,8 graus por passo), o que proporciona alta precisão de movimentação. Além disso, o motor é compacto e robusto, características importantes para aplicações em sistemas automatizados, como telescópios.

Para o controle do motor, utilizou-se o módulo controlador de motores de passo TMC2208. Este controlador foi escolhido por diversas vantagens: ele permite movimentos mais suaves ao motor, reduzindo vibrações e ruídos indesejados, fundamentais para reduzir

oscilações mecânicas que possam surgir a partir das vibrações do motor. Além disso, o TMC2208 possui recursos de segurança integrada, como proteção contra sobrecorrente e curto-circuito, evitando danos ao microcontrolador (Raspberry Pi 4) e garantindo a estabilidade operacional do telescópio.

No aspecto elétrico, o circuito foi projetado para alimentar o sistema com uma fonte de bancada ajustada a $12V$, que atende às especificações do módulo TMC2208. A alimentação de 12V é fornecida ao terminal VM (alimentação dos motores), conforme indicado no esquemático (23). Capacitores eletrolíticos de $100uF$ foram adicionados próximos aos pinos de alimentação de cada módulo para reduzir oscilações e ruídos elétricos provenientes da fonte de alimentação, garantindo estabilidade na operação dos motores.

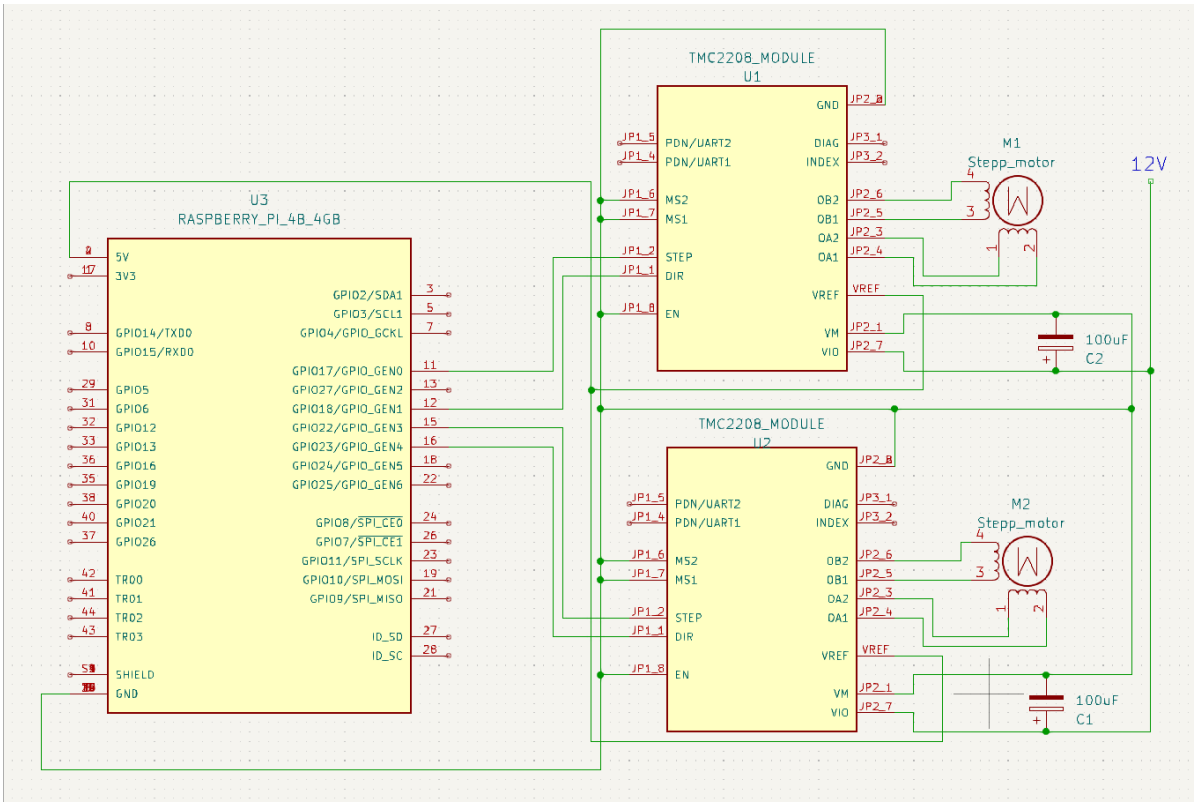


Figura 23 – Esquemático do circuito de controle dos motores

Os sinais de controle do motor são gerados pelo Raspberry Pi 4 e enviados aos pinos STEP e DIR do TMC2208. O sinal STEP determina os pulsos de passo do motor, enquanto DIR define a direção do movimento (horário ou anti-horário). O pino EN é utilizado para habilitar ou desabilitar o driver do motor, permitindo maior controle sobre o consumo energético do sistema.

Adicionalmente, os modos de operação do TMC2208, como o microstepping (passos fracionados), são configurados por meio dos pinos MS1 e MS2, que podem ser conectados a tensões lógicas altas ou baixas, dependendo da resolução desejada. No presente projeto, os pinos foram configurados para operar com microstepping de 1/8 com os dois pinos conectados ao Terra, melhorando a suavidade do movimento do motor.

Um dos motores será localizado na base para rotacionar o eixo vertical, enquanto o outro motor ficará próximo ao braço da montagem para rotacionar o eixo horizontal. Para fixar o motor na lateral, será utilizado o suporte que foi impresso da Figura 16. Nesse suporte, também será implementado um mecanismo de desacoplamento do motor, permitindo a rotação manual do telescópio. No sistema mecânico da base, correspondente ao eixo de rotação vertical, está embutido um sistema de desacoplamento por parafuso. Dessa forma, o pinhão não terá contato com a coroa, retirando o motor do sistema de transmissão, permitindo o ajuste manual da direção.

Na Figura 24 pode ser vista a montagens do sistema mecânico para a rotação do eixo vertical, enquanto que a montagem do sistema mecânica do eixo horizontal está representada na Figura 25.

Figura 24 – Montagem do sistema mecânico vertical

O *hardware* para sistemas embarcados escolhido foi o Raspberry Pi 4 Model B. Essa seleção foi baseada em sua alta capacidade de processamento, possuir um módulo integrado para conectividade Wi-Fi, portas GPIO (*General Purpose Input/Output*) que podem ser configuradas para agir tanto como entrada quanto saída digital, que serão

Figura 25 – Montagem do sistema mecânico horizontal

usadas para o controle do motor. Além disso ele também possui entrada Ethernet, HDMI (*High-Definition Multimidia Interface*) e suporte para câmera USB (*Universal Serial Bus*). Optar por um hardware embarcadao com configurações de alto desempenho foi uma decisão estratégica visando futuras implementações, tais como a identificação automática da direção do movimento aparente ou processamento digital de imagem.

Para controlar a velocidade de rotação do motor, foi selecionado o *driver* TCM2208. Ele desempenha o papel crucial de direcionar o sentido de rotação além de controlar a velocidade de rotação. A velocidade do movimento é controlada por um trem de pulsos digitais gerada por um dos pinos do *raspberry*. O *driver* recebe os sinais enviados pelo Raspberry Pi e chaveia a tensão de alimentação, variando o ciclo de trabalho do pulso, para o motor de passo, de modo a induzir suas bobinas.

Para que o *raspberry* possa capturar e processar as imagens, será utilizada uma câmera conectada ao sistema por meio de uma interface USB. A escolha se deu pela simplicidade de integração com o *hardware* para sistemas embarcados, além de possuir suporte nativo para programação pelo sistema operacional.

4.4 Software

A interface com o usuário é realizada através de um navegador (web browser). Com esse objetivo, foi escolhido o Apache [10] como servidor. A programação da interface foi

realizada com páginas HTML e programas escritos em PHP

Foi desenvolvida uma interface web para a interação do usuário com o sistema. A Figura 26 representa o *design* da interface desenvolvida. Como pode ser vista, a interface possui botões para realizar o controle manual do direcionamento do telescópio, uma área de configuração da câmera e uma área para o sistema entrar no modo onde os motores giram para acompanhar o movimento aparente dos astros no céu.

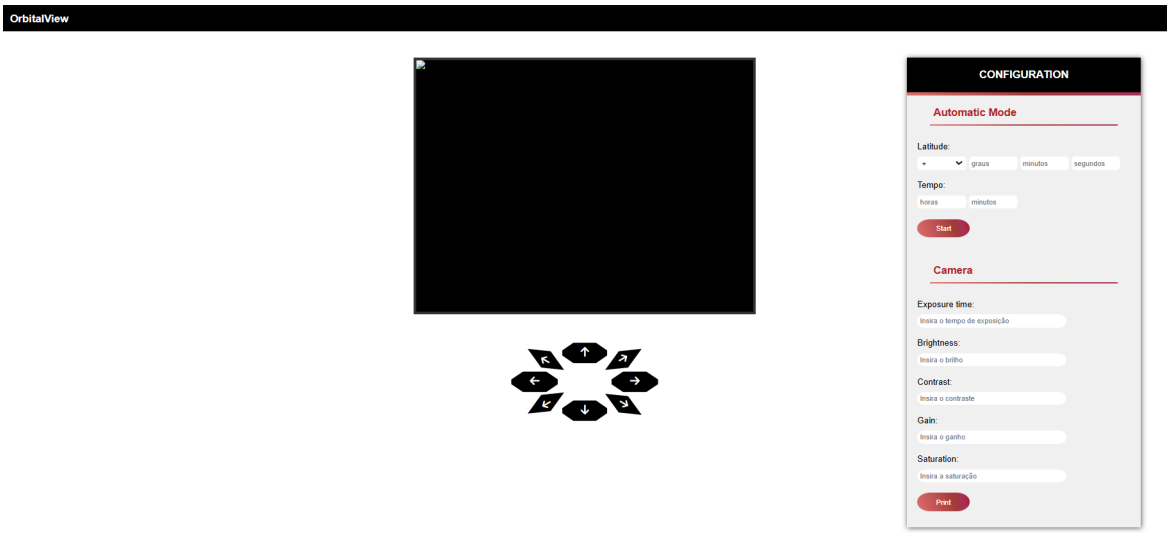


Figura 26 – Interface de comunicação do usuário com o sistema

Ao iniciar qualquer um dos botões de direção (representados pelas setas na figura 26) na interface, os comandos são enviados para um *script* PHP. Esse *script* recebe os comandos, combinado-os em uma única *string*, gravando-o em um PIPE anônimo, permitindo que o programa de controle do motor possa acessá-los posteriormente. Além disso, o *script* PHP que executa um kill -HUP no processo de controle do motor, fazendo com que ele execute o comando do PIPE. Quando o usuário cessa a interação com o botão de direção, outro comando é enviado para manter os motores em estado estático.

No formulário destinado ao controle automático da interface web, denominado *Automatic Mode*, o usuário encontrará diversos campos disponíveis para preenchimento. Após inserir os dados necessários, deve-se pressionar o botão *start* para que as informações sejam enviadas por um *script* PHP para o PIPE de comunicação e envia um sinal SIGHUP. Este comando sinaliza que o sistema irá operar no modo de acompanhamento do movimento

de rotação da Terra, utilizando parâmetros como latitude e tempo de monitoramento fornecidos pelo usuário.

Outra funcionalidade do botão *start* é que, ao ser pressionado, o formulário completo incluindo o próprio botão *start* e os botões de direção, são ocultados, já que esses elementos não são necessários no modo automático. Em substituição ao formulário, será exibido um botão *stop*, que permite encerrar o modo automático e retornar ao modo de repouso, utilizando o mesmo método de escrita no PIPE e envio do sinal SIGHUP. Quando o botão *stop* é acionado, os botões de direção e o formulário do modo automático reaparecem, enquanto o botão *stop* é ocultado.

Para o controle do motor foi utilizada a linguagem de programação C devido a sua alta eficiência e controle de baixo nível, características essenciais para o controle de motores e comunicação com o microprocessador *Raspberry Pi*. Além disso, a linguagem permite uma ótima gestão de recursos.

A função principal do programa é responsável por inicializar e gerenciar o sistema como um todo, incluindo a configuração dos recursos, a espera por comandos e a execução das tarefas apropriadas para o controle dos motores. Inicialmente, o semáforo utilizado para sincronização das *threads* é configurado, garantindo que ele comece em estado bloqueado. Em seguida, o ambiente de trabalho para controle de GPIO é preparado, e os pinos necessários para operação dos motores são configurados como saídas.

O *loop* principal do programa entra em execução contínua, aguardando a liberação do semáforo após a recepção de um comando. Quando isso ocorre, diferentes comportamentos podem ser acionados com base no tipo de comando recebido. Para comandos que envolvem o controle dos motores, são criadas *threads* que executam a função responsável por gerar os pulsos de controle. Essas *threads* ajustam os pinos de direção e passo para cada motor de acordo com os parâmetros enviados, podendo controlar individualmente os motores horizontal e vertical, ou ambos simultaneamente, dependendo da necessidade.

Para comando de controle automatico, baseados em coordenadas e tempo, o programa calcula a direção e a duração apropriadas para movimentação dos motores, simulando um acompanhamento de objetos astronômicos. Durante a execução, o tempo decorrido é monitorado e, ao atingir o limite especificado, as *threads* responsáveis pelo movimento dos motores são encerradas. Além disso, constantemente é monitorado se o

comando de stop (0x01) foi executado, para quando identificá-lo, parar a rotação dos motores e entrar no estado de repouso.

O funcionamento do código é ilustrado na Figura 27. Inicialmente, o motor encontra-se em repouso. Quando um botão de direção é pressionado, o motor começa a girar na direção correspondente, e o sistema realiza verificações contínuas para determinar se o botão permanece pressionado. Caso a condição seja verdadeira, o motor mantém o movimento na direção especificada; caso contrário, retorna ao estado de repouso. É importante ressaltar que, na interface de controle desenvolvida, a ativação de um botão direcional vertical ou horizontal resulta no acionamento exclusivo do motor correspondente à direção selecionada. Por outro lado, ao pressionar um botão associado a uma direção diagonal, ambos os motores são simultaneamente acionados, garantindo o movimento combinado necessário para atender à trajetória desejada.

Adicionalmente, ao acionar o botão *send*, os motores ajustam o movimento do telescópio para acompanhar a velocidade angular da rotação terrestre, permanecendo nesse estado até que o botão *stop* seja pressionado, momento em que o movimento é interrompido.

No caso do Controle Automático, é implementada uma condição de *Time Out*. Quando o usuário opta por operar o sistema no modo automático, é necessário inserir o tempo durante o qual o telescópio realizará o acompanhamento do objeto. Caso esse tempo seja ultrapassado, os motores retornam automaticamente ao estado de repouso.

Por fim, considerando que o telescópio será operado em uma área remota e o usuário precisará interagir com o sistema à distância, foi configurado um serviço no *systemd*. Esse serviço garante que o *script* que realiza o controle do motor vertical e horizontal do projeto seja executado automaticamente durante a inicialização do sistema operacional Linux, assegurando que o telescópio esteja pronto para uso assim que o sistema operacional for ligado, sem a necessidade de intervenções manuais. Considerando que a biblioteca utilizada para o controle dos pinos GPIO exige permissões de superusuário, foi atribuido permissão SUID para ser executado pelo usuário como se ele fosse o superusuário. Essa configuração garante que o programa tenha acesso irrestrito aos recursos necessários para operar corretamente.

Para garantir o acompanhamento contínuo dos astros, o código responsável pelo

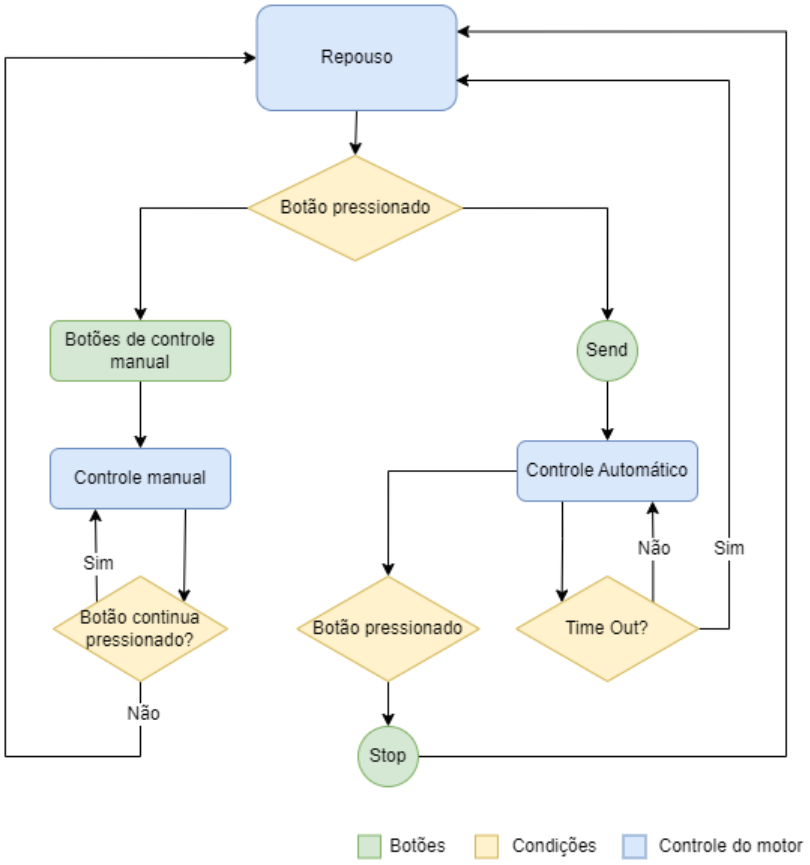


Figura 27 – Funcionamento entre Interface Web X script do motor

controle do motor deve operar ininterruptamente, pois qualquer interrupção resultaria na perda do objeto observado. Para evitar esse problema, o código deve rodar com prioridade em tempo real. Essa configuração foi integrada junto ao *script* de inicialização, assegurando que o controle dos motores receba os recursos necessários do sistema para manter o acompanhamento do objeto.

Por fim, a Figura 28 ilustra o funcionamento do sistema de comunicação desenvolvido para o projeto. Inicialmente, o usuário interage com a Interface Web, onde pode selecionar os modos de operação do telescópio e ajustar as configurações da câmera. Quando um dos modos de operação do telescópio é escolhido, os dados inseridos na Interface Web são enviados para um *script* em PHP. Esse *script* escreve as informações em um PIPE de comunicação inter-processos. Posteriormente, o *script* de controle dos motores lê os dados presentes no PIPE, iniciando assim o movimento do telescópio conforme as instruções recebidas.

Caso a configuração da câmera seja escolhida, os dados são enviados para o script

da câmera, onde os parâmetros escolhidos pelo usuário são configurados através do v4l2 e a captura da imagem realizada pelo ffmpeg. Caso o usuário escolha salvar a foto gerada, ela será enviada para a Interface Web, onde o usuário poderá fazer a sua observação.

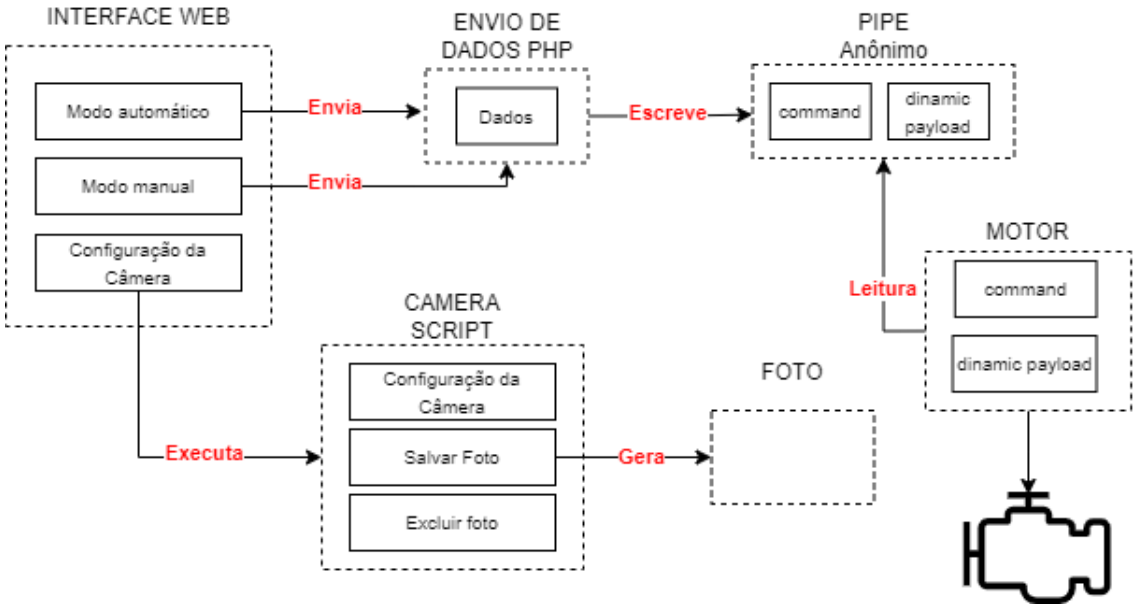


Figura 28 – Sistema de comunicação entre os códigos

4.5 Comunicação entre processos

Para configurar os modos de operação do motor serão utilizados programas escritos em linguagem C e PHP. Assim, é necessário um meio de comunicação entre esses programas para que possam interagir entre si. Para isso, optou-se pelo uso de um PIPE anônimo (FIFO), que intermediará a comunicação entre processos, considerando que todos os arquivos e processos envolvidos estão sendo executados na mesma máquina. Essa escolha oferece diversas vantagens relacionadas à simplicidade, desempenho e adequação às necessidades do sistema.

Os *named pipes*, são mais fáceis de configurar e utilizar quando comparados com os sockets, uma vez que não há necessidade de lidar com conceitos adicionais como endereços IP ou portas, que são uma configuração essencial quando se usa os *sockets*. Em um ambiente onde os processos estão confinados ao mesmo sistema, os PIPES anônimos permitem uma comunicação direta e eficiente com menos sobrecarga de configuração.

Além disso, os PIPES anônimos operam diretamente no sistema de arquivos, eliminando a sobrecarga associada à abstração de rede, tornando os ”arquivos” FIFOs

ideais para sistemas simples e de alto desempenho, como o controle de motores ou telescópios automatizados.

Outra vantagem oferecida por essa interface de comunicação é que elas são suportadas por sistemas baseadas em Unix/Linux e podem ser integradas com deferentes linguagens de programação como a linguagem C ou PHP utilizados nesse trabalho, possibilitando uma conexão direta entre dois processos de linguagens diferentes sem a necessidade de bibliotecas específicas ou um aplicativo externo.

Foi desenvolvido um protocolo, estruturando a comunicação de forma mais organizada. Também vale ressaltar que os dados enviados serão do tipo *string*, uma vez que a interpretação desse tipo de dado é a mesma na linguagem C e PHP, não sendo necessário uma conversão. Para parâmetros numéricos como latitude, é feita a conversão de *string* para um inteiro no programa que irá utilizá-lo.

A estrutura de comunicação consiste basicamente em dois campos sendo o primeiro campo para comandos e o segundo campo seria para *payload* dinâmico, na qual este último campo pode variar o significado e a quantidade de campos conforme o comando inserido. A estrutura geral desse protocolo pode ser observado na Figura 29. A figura ilustra duas áreas sendo o primeiro campo correspondente à área de comando com 2 bytes. Para determinar o comando enviado através desse campo, será feito uma comparação bit a bit, uma vez que esse método agiliza o processo de verificação.

Percebe-se que pela Figura 29 não seria necessário que o campo de comandos houvesse 2 bytes, pois não há mais do que 8 comandos, porém, essa quantidade de bytes foram reservadas pensando na extensão deste projeto, para quando for implementado mais comandos, como a localização do telescópio por GPS, processamento de imagens entre outros.

Para caso o LSB (*Least Significant Bit*) do LSByte (*Least Significant Byte*) correspondente ao campo de COMANDO estiver com valor de 2, a área de *payload* dinâmico (PAD) será composto por 2 campos, sendo que o primeiro campo terá 1 *byte* indicando o motor que será acionado e o segundo campo será também de 1 *byte* indicando a direção que irá girar. Ao ser lidos esses dados no programa em que for processá-lo, irá inicialmente realizar a conversão para um inteiro para que assim possam realizar a comparação bit a bit

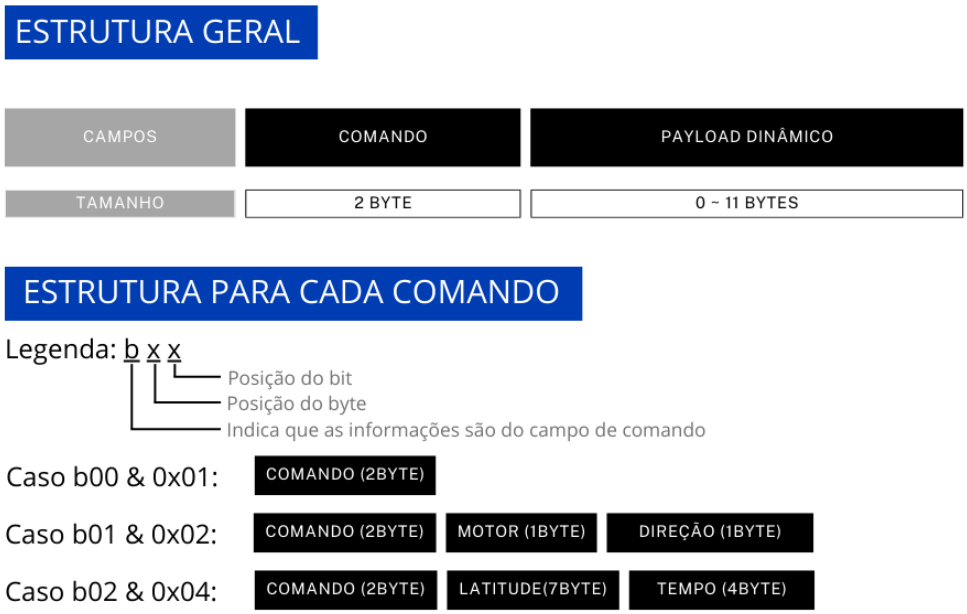


Figura 29 – Protocolo de comunicação

para cada caso. Esse comando será enviado quando for requisitado para realizar o controle remoto da direção do telescópio, possibilitando rotacionar nos eixos vertical e horizontal.

A estrutura geral dos campos de PAD para este caso pode ser visto na Figura 30. Na figura são ilustrados os campos de motor e direção. Para quando o campo de motor tem valor igual a 1, apenas o motor x que rotaciona o eixo horizontal é acionado, enquanto para um valor de 2 o motor y que rotaciona o eixo vertical será o único acionado. Para o caso em que o campo assume um valor de 4, ele aciona os dois motores ao mesmo tempo. Para o campo de direção, foi configurado 4 estados como pode ser visto na figura.

Já para o caso do segundo LSB do LSByte correspondente ao campo de COMANDO for igual a 4, o PAD será composto por 2 campo sendo que o primeiro terá 7 byte que corresponde à latitude (1 *bytes* para sinal, 2 *bytes* para graus, 2 *bytes* para minutos e 2 *bytes* para segundos) e 4 bytes para o campo de tempo de acompanhamento (2 *bytes* para horas e 2 *bytes* para minutos). Ao receber esse campos, o programa irá converté-los para um inteiro. Esses parâmetros são utilizados para calcular a velocidade que cada motor deve girar e o número de passos em que o motor irá girar. O comando em questão será

Figura 30 – Campos do payload para o caso de controle do direcionamento

utilizado quando já estiver concluído o direcionamento do telescópio, de modo que esteja apontando para a região de interesse de observação, para assim inicializar a observação com o sistema acompanhando o movimento aparente.

A estrutura geral dos campos de PAD para este caso pode ser visto na Figura 31. Na figura são ilustrados os campos de sinal, graus, minutos e segundos correspondentes a latitude e os campos de horas e minutos correspondentes ao tempo em que será feito acompanhamento do movimento aparente. Com a informação da latitude será determinado a velocidade de rotação de cada eixo de modo que o sistema possa acompanhar a rotação da Terra em torno do seu próprio eixo e o tempo será utilizado para encerrar o processo de acompanhamento. O sinal da latitude define em qual sentido o eixo y deve girar.

Figura 31 – Campos do payload para o caso de controle automático

4.6 Considerações e fórmulas

Ao ser utilizada a interface, o usuário deve seguir algumas regras. Primeiro, para os campos de entrada de texto, o usuário deve obedecer os limites de cada campo. Por exemplo, para o campo latitude, ele não deve passar de 90 graus além de inserir apenas valores do tipo inteiro, sem vírgula. Para os campos de minutos, segundos da latitude e campos de minuto do tempo existe um limite de até 60 sendo que para o campo de horas do tempo existe um limite de 24 horas.

Além disso, o usuário deve inserir exatamente dois dígitos para cada campo do modo automático (graus, minutos, segundos, horas de tempo e minutos de tempo), pois foram reservados 2 *bytes* para cada campo no protocolo de comunicação utilizado neste projeto.

Já para o campo de entrada dos parâmetro da câmera, o usuário deve obedecer os valores permitidos para o v4l2 configurar. Por exemplo, para a *webcam* utilizada neste projeto, o limite é de 1 até 5000 que equivale à $1us$ e $5ms$ respectivamente. Então o

usuário só poderá inserir valores dentro dessa faixa, sendo que o formato deve ser inteiro. O mesmo vale para os outros 4 campos.

Para verificar os limites de configuração, pode-se utilizar o seguinte comando no terminal, substituindo XX de videoXX pelo número correspondente à câmera:

```
v4l2-ctl -d /dev/videoXX --list-ctrls
```

Sabe-se que a rotação da Terra em torno do seu próprio eixo ocorre do leste para o oeste, sendo que a latitude representa o ângulo entre o plano de rotação da Terra e a normal da superfície que tangencia a Terra. A latitude para o globo pode ser visto na Figura 32.

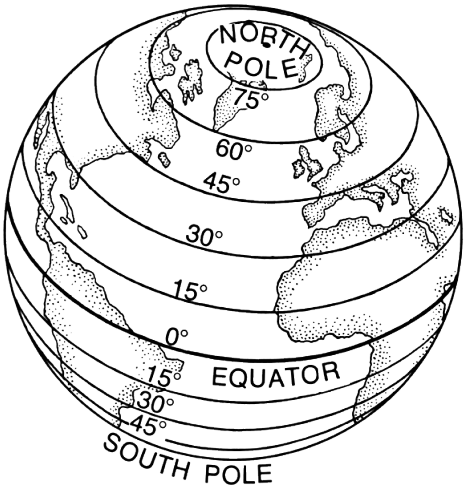


Figura 32 – Sistema de comunicação entre os códigos

Fonte: https://commons.wikimedia.org

Então significa que para quem está exatamente na latitude 0°(linha do equador), as estrelas se ”movem” apenas no plano leste oeste normal à superfície, enquanto que para uma coordenada com latitude diferente, apresentará uma direção da estrela com inclinação igual da latitude. Para melhor entendimento, está sendo ilustrado essa situação na Figura 33.

O cálculo da velocidade será baseado nas relações de engrenagem, passo do motor, velocidade de rotação da Terra e a latitude onde se localiza o telescópio. O motor utilizado, tem um passo de 1,8, porém ao ser utilizado o *driver*, esse passo é reduzido para 1,8/8,

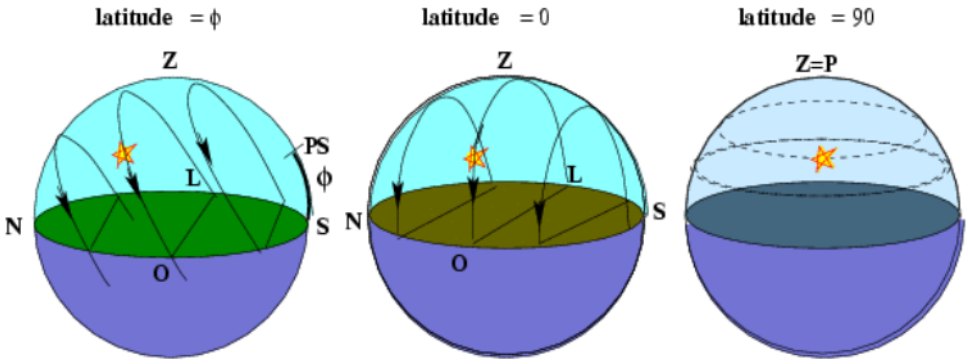


Figura 33 – Sistema de comunicação entre os códigos

Fonte: https://www.if.ufrgs.br

alteração necessária para aumentar a precisão do sistema. sabe-se que a velocidade ângular pode ser obtida pela expressão (4.2).

$$w = \frac{\Delta\theta}{T} \tag{4.2}$$

Sabe-se também que a velocidade do motor pode ser determinado pela expressão (**??**), então essas duas expressões podem ser igualadas e então, isolando o período para cada passo, temos a expressão (4.3):

$$T = \frac{\Delta\theta z_2}{i_r w_1 z_1} \tag{4.3}$$

onde o índice $_1$ é para os parâmetros relacionados à coroa acoplada no telescópio e $_2$ é ao pinhão.

Sabe-se que a velocidade de rotação da Terra pode ser dado por $360/((23.60 + 56)60 + 4)$ enquanto i_r tem um valor de 19,2. Porém, essa velocidade da Terra só é válida para caso o plano de rotação for paralelo ao plano de rotação da Terra. Logo os tempos para cada eixo do motor pode ser calculador pela expressão (4.4) e (4.5)

$$w_x = w\cos\alpha,$$

$$w_y = w\sin\alpha,$$

$$T_x = \frac{\Delta\theta z_{2x}}{i_r w_x z_{1x}}, \tag{4.4}$$

$$T_y = \frac{\Delta\theta z_{2y}}{i_r w_y z_{1y}}. \tag{4.5}$$

onde w é a velocidade de rotação da Terra, o índice $_x$ é para os parâmetros relacionados motor x e $_y$ ao motor y.

As expressões (4.4) e (4.5) só são válidas se o plano de rotação da Terra contiver à região que se quer observar e o telescópio, caso contrário essas expressões não são válidas. Para esclarecer isso, a Figura 34 ilustra 2 situações onde no primeiro caso tanto a estrela quanto o telescópio estão contidos no plano de rotação enquanto no segundo caso apenas um dos dois ficam contidos no mesmo plano, sendo que foi considerado uma latitude igual à 0°. Para ambos os casos a estrela se move na direção perpendicular à superfície, uma vez que sua latitude é zero. Para o primeiro caso, a estrela irá se mover e o telescópio irá conseguir acompanhá-lo. No entanto, para o segundo caso em que o telescópio teve que direcionar, de modo que não aponta mais para a direção leste oeste, passará a ter outro plano de rotação que não irá mais acompanhar o objeto direcionado.

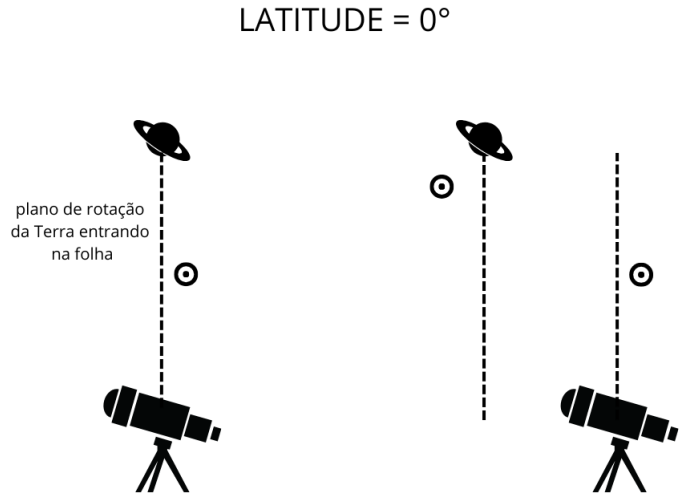


Figura 34 – Exemplo de caso onde as expressões (4.4) e (4.5) são válidas e inválidas

Para facilitar o entendimento, foi feito, para o mesmo exemplo, um desenho vetorial como pode ser visto na figura 35. Percebe-se que ao realizar a rotação do telescópio para fazer o direcionamento para a região de interesse, cria-se um ângulo θ entre essa direção e o vetor de direção da rotação. Ao observar a figura, percebe-se que tem um caso semelhante à quando tem uma latitude diferente de 0 onde a velocidade está decomposta em dois eixos com relação à direção do telescópio. Então para o sistema acompanhar essa estrela, deve-se acrescentar à latitude esse ângulo que existe entre a direção do telescópio e a direção de rotação da Terra.

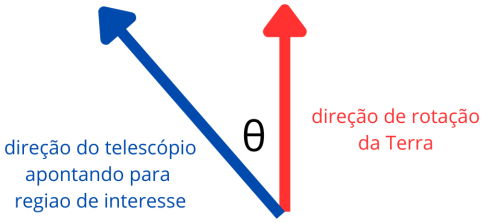


Figura 35 – Vetores de rotação da Terra e dadireção em que o telescópio aponta

5 RESULTADOS

Neste capítulo, serão apresentados os resultados obtidos durante o desenvolvimento do projeto, acompanhados de uma análise comparativa com os objetivos previamente estabelecidos.

Inicialmente, foram estabelecidos cinco objetivos principais dentro do projeto. O primeiro deles envolveu o estudo das diferentes configurações de montagem para o sistema de rotação que podiam ser implementados para pequenos telescópios. Considerando que o projeto se destina a telescópios de pequeno porte e aproveita peças já disponíveis, optou-se pela montagem de garfo, onde esse tipo de montagem equatorial é caracterizado por sua operação baseada em dois eixos de rotação, oferecendo os movimentos necessários para o acompanhamento dos astros no céu.

Com a montagem definida, foi realizada a aquisição de todas as peças necessárias para o desenvolvimento da estrutura mecânica. Algumas peças, como o suporte giratório e engrenagens, foram reaproveitados das peças que já acompanhavam o telescópio, contribuindo para a redução de custos e otimização de recursos. As demais engrenagens necessárias foram projetadas utilizando o *software* de modelagem 3D SolidWorks e posteriormente produzidas com a utilização de uma impressora 3D. Além disso, o sistema contou com a utilização de caixas redutoras, fundamentais para aumentar a precisão do movimento e assegurar que o sistema tenha a precisão necessária para realizar o acompanhamento dos objetos.

Para o protótipo do sistema, foi utilizado um tripé de fotografia no qual foram acoplados tanto o telescópio quanto o sistema de motores. No entanto, como o suporte não suporta cargas pesadas, ele não oferece uma boa estabilidade necessária para garantir o desempenho ideal do sistema.

Outro fator que impactou negativamente a precisão do sistema foi o do conjunto de engrenagens cônicas. Devido a limitações nas impressoras 3D utilizadas, não foi possível atingir uma impressão de alta resolução, o que resultou em um acabamento impreciso das peças. Como consequência, o sistema de engrenagens cônicas apresenta um certo atrito, fazendo com que o movimento não seja suave, prejudicando o controle e afetando a precisão.

Foram realizados testes de velocidade do motor para determinar a máxima velocidade que o sistema consegue atingir. A maior velocidade registrada foi de 1125 $graus/s$. Entretanto, ao testar essa mesma velocidade com o redutor, observou-se que o valor obtido foi de 30,5 $graus/s$, uma diferença significativa em relação ao valor teórico esperado, que seria 58,6 $graus/s$ (calculado como 1125/19,5).

Além disso, foram realizados testes em duas velocidades distintas com e sem o redutor. Para o motor isolado, os valores obtidos mostraram-se próximos dos resultados teóricos. Já no sistema composto por motor e redutor, verificou-se um atraso de aproximadamente $5s$ por volta em relação ao valor teórico. Durante os testes, utilizou-se um transferidor acoplado ao motor para medir os deslocamentos angulares em graus, enquanto o tempo foi registrado com o auxílio de um cronômetro.

A montagem utilizada para os testes está ilustrada na Figura 36, enquanto a Tabela 5 apresenta a comparação entre as velocidades medidas e calculadas. Observou-se que, em velocidades mais baixas, o sistema apresenta maior precisão, necessitando apenas de pequenos ajustes. Vale ressaltar que as velocidades testadas foram projetadas para o modo de direcionamento do telescópio, onde se requerem velocidades mais altas. Contudo, o principal objetivo do projeto é o acompanhamento de astros, que ocorre em velocidades muito menores. Assim, a discrepância nas velocidades configuradas e medidas no modo de direcionamento não representa um problema para o trabalho.

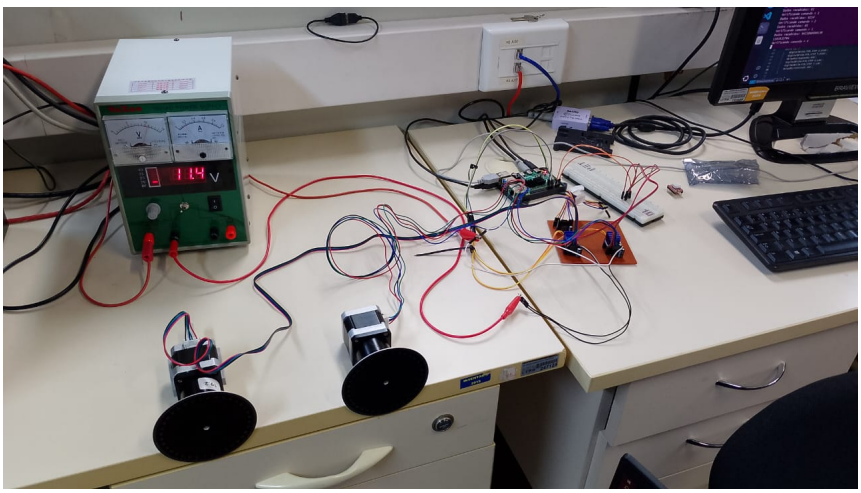


Figura 36 – Montagem para a medição das velocidades do motor

Para a função do acompanhamento da rotação da Terra, também foi utilizado o mesmo *script*. Foi feito um teste apenas dos motores acoplados nos redutores de 1 para

Tabela 5 – Comparação entre velocidade real e calculada

Motor		**Motor com redutor**	
vel. configurada	vel medido.	vel. configurada	vel. medido
1125	1091	$58,59$	$30,5$
225	$223,7$	$11,72$	$10,26$

19,2 com um tempo de duração de 45 minutos e latitude de 45° e 1 horas com latitude de 0 °. Vale a pena ressaltar que cada motor irá girar com velocidades diferentes uma vez que a relação de transmissão é diferente para cada eixo, já que a relação das engrenagem do eixo vertical é de $i_v = 19,2\frac{206}{18}$ e do eixo horizontal é $i_h = 19,2\frac{120}{17}$. Na tabela 6 pode ser visto quantos graus deveria ter girado em cada redutor e quanto girou na prática.

Tabela 6 – Comparação entre o deslocamento angular real e calculado

-	**Motor X**		**Motor Y**	
config	$\Delta\theta$ calculado [°]	$\Delta\theta$ medido [°]	$\Delta\theta$ calculado [°]	$\Delta\theta$ medido [°]
$45min$ 45°	$56,3$	$55,5$	$91,28$	$91,2$
$1h$ 0°	106	110	0	0

Outro *software* essencial para o projeto é responsável pela configuração da câmera e pela captura de uma foto utilizando essas configurações. Para desenvolvê-lo, foi necessário, primeiramente, identificar quais parâmetros poderiam ser ajustados e os limites de cada um. Inicialmente, analisou-se os parâmetros do sensor CCD previsto para o projeto. No entanto, constatou-se que esse sensor não permitia ajustar o tempo de exposição, uma limitação crítica, considerando que essa configuração é fundamental para capturar imagens adequadas no ramo de astrofotografia.

Diante disso, optou-se por utilizar uma webcam como solução temporária, permitindo o desenvolvimento de um ambiente de configuração que será adaptado futuramente para um dispositivo de captura com maior flexibilidade. Na Figura 37, estão apresentados os parâmetros configuráveis da webcam, assim como seus limites. Apesar de ser possível ajustar o tempo de exposição, o valor máximo disponível (5 ms) ainda é insuficiente para astrofotografia. Contudo, como mencionado, o foco inicial é criar um ambiente funcional de configuração que poderá ser utilizado com equipamentos mais avançados posteriormente.

A interface, ilustrada na Figura 26, foi implementada utilizando PHP, oferecendo um ambiente onde o usuário pode controlar a direção do movimento por meio de botões com ícones de setas. Após posicionar o telescópio na direção desejada, o usuário pode

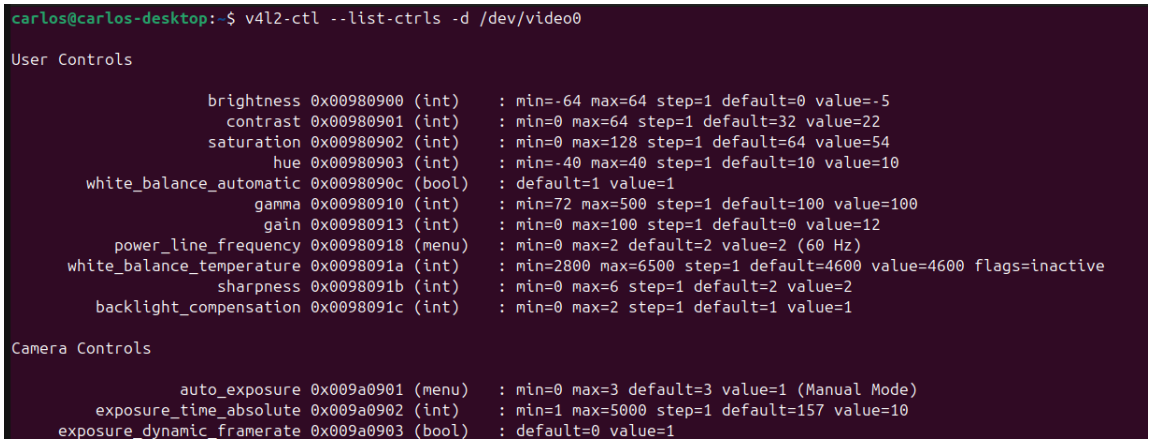


Figura 37 – Configuração de câmera

iniciar a observação inserindo a latitude correspondente à localização do telescópio, bem como o tempo previsto para o acompanhamento. Com esses dados configurados, basta clicar no botão *Start* para que o sistema processe as informações e inicie o acompanhamento dos corpos celestes.

Durante a observação, o usuário tem a opção de capturar imagens utilizando a seção de câmeras, localizada logo abaixo da área de configuração do telescópio para o modo automático. Nessa interface, é possível ajustar os parâmetros desejados para a fotografia antes de realizar a aquisição da imagem. Após capturar a imagem, ela é exibida na tela, logo acima dos botões de controle de direção.

Por fim, foram capturadas algumas imagens com a *webcam*, ajustando diferentes propriedades. As imagens obtidas com variações no tempo de exposição e contraste estão apresentadas na Figura 38. Nessas capturas, utilizou-se uma sala como cenário, em vez de estrelas ou regiões do céu.

A etapa de validação do sistema, que consistiria em capturar fotografias do céu, não foi realizada neste trabalho devido a fatores operacionais e climáticos. Embora o dispositivo de captura utilizado apresente limitações em sua configuração, seria viável conduzir essa validação direcionando o dispositivo para objetos astronômicos de maior brilho, como a Lua ou Marte. Nesse caso, seria possível comparar as fotografias capturadas em diferentes momentos para avaliar o deslocamento do objeto dentro do quadro, verificando a precisão do sistema em manter o alinhamento ao longo do tempo.

Entretanto, a realização dessa etapa dependeu diretamente de condições climáticas

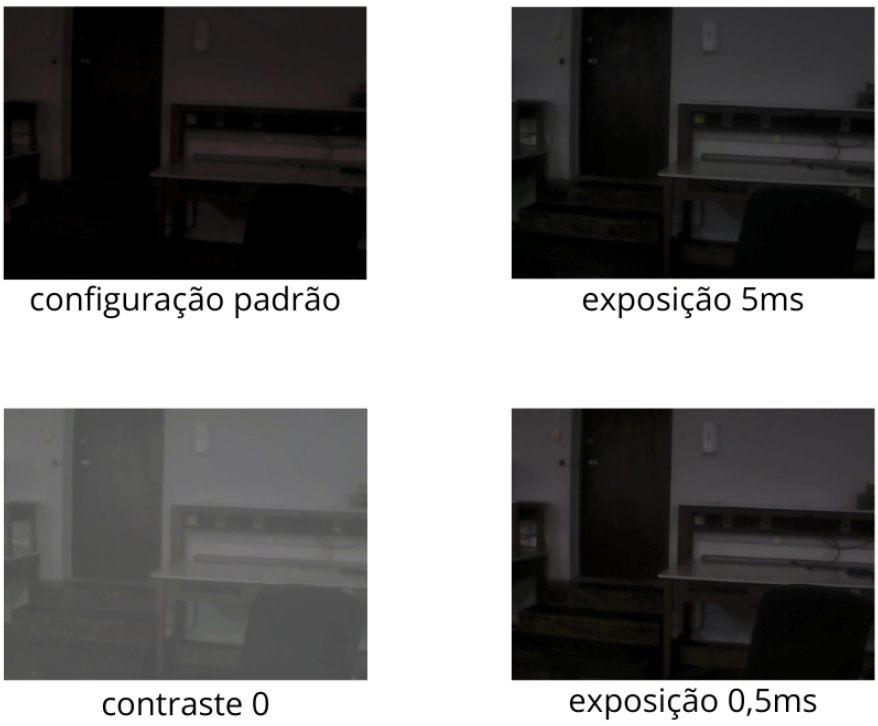


Figura 38 – Fotos com diferentes parâmetros de configuração

favoráveis, ou seja, de céus limpos e ausência de nebulosidade. Quando o sistema estava pronto para realizar os testes e validações, predominou um clima desfavorável para esta etapa, com dias seguidos de chuva ou céu nublado, o que dificultou a realização desta última etapa. Além disso, durante os testes preliminares, ocorreu a falha de um dos módulos de acionamento do motor, comprometendo o funcionamento do sistema e inviabilizando a continuidade dessa etapa de validação.

Embora a etapa de validação com fotografias não tenha sido executada, sua realização futura é essencial para garantir a precisão do sistema no rastreamento de objetos astronômicos. Essa etapa permitiria não apenas verificar possíveis deslocamentos, mas também identificar ajustes necessários para melhorar o desempenho em cenários reais.

6 Projetos futuros

Para projetos futuros é necessário implementar um sistema em que calcule automaticamente o ângulo deslocado com a implementação de sensores ou até mesmo calcular através do tempo e a velocidade de rotação programada, sendo que para aplicar este último método seria necessário que a velocidade programada fosse exatamente igual à velocidade real do sistema.

Além disso, a aquisição de um dispositivo de captura de imagem adequado para realizar astrofotografia também é um requisito fundamental para o funcionamento pleno do projeto, além da projeção das engrenagens cônicas com um material mais resistente ou a aquisição da engrenagem com materiais metálicos.

A partir da implementação do sistema de medição e a aquisição dos dispositivo de captura e as engrenagens, deve-se verificar a velocidade de rotação dos motores bem como a foto gerada com as configurações alteradas. Com essa verificação feita, seria necessário realizar a validação do sistema como um todo, de modo que se deve direcionar o sistema para uma região de observação e então, em alguns períodos de tempo, gerar uma imagem dessa região verificando se o sistema de fato está conseguindo compensar o movimento de rotação da Terra. Vale ressaltar que o sistema deve estar apontando para o sentido leste oeste, para que a medição do deslocamento angular tenha como referência essa direção. Os itens que devem ser verificados são:

- Velocidade de rotação real x programada.
- Constância da velocidade ao longo do tempo.
- Influência dos passos na imagem final obtida pela ampliação no telescópio
- Acompanhamento do movimento aparente sem direcionar o sistema.
- Acompanhamento do movimento aparente mudando a direção leste oeste.
- Variação dos parâmetros da câmera
- Comparação das imagens capturadas ao longo do tempo

Além dessas implementações, outras funcionalidades podem ser adicionadas para melhorar o desempenho do sistema. Uma proposta seria o desenvolvimento de um *software* de processamento de imagem, com o objetivo de aprimorar a nitidez das capturas e calcular o vetor de deslocamento da imagem. Com base nesse vetor, seria possível determinar a direção de rotação necessária para o sistema e ajustar a velocidade do motor de forma precisa.

Adicionalmente, uma função de *stacking* de imagens poderia ser implementada. Essa técnica permitiria dividir um tempo de exposição prolongado em várias capturas menores, combinando-as posteriormente em uma única imagem. Isso resultaria em uma maior qualidade de captura sem os desafios associados a exposições muito longas.

Outra funcionalidade relevante seria a inclusão de um sistema de monitoramento de temperatura próximo à câmera, aliado a um mecanismo de resfriamento. Isso ajudaria a reduzir os ruídos térmicos, especialmente durante fotos de longa exposição, garantindo resultados mais limpos e detalhados.

O projeto pode ser integrado ao banco de dados do *Stellarium*, permitindo que o usuário escolha o objeto e então o telescópio realize a busca automatizada pelos astros. Essa integração elimina a necessidade de ajustes manuais por meio da interface Web, facilitando a localização dos corpos celestes de forma mais prática.

7 Conclusão

Combinando princípios de engenharia elétrica, computação embarcada e design mecânico, o projeto oferece uma solução inovadora e acessível que atende às necessidades tanto de astrônomos amadores quanto de pesquisadores profissionais. Ao longo deste trabalho, foram discutidos os desafios da observação astronômica decorrentes do movimento aparente dos corpos celestes causado pela rotação da Terra. Para isso, a proposta deste trabalho consiste no desenvolvimento de um sistema que possa acompanhar o movimento aparente das estrelas.

Um dos principais pontos positivos do projeto é a integração de tecnologias para alcançar um sistema eficiente e confiável. A utilização de motores de passo controlados por drivers TMC2208 garantiu precisão para velocidades baixas no movimento do telescópio, características essenciais para evitar oscilações e garantir a estabilidade das imagens capturadas.

Outro ponto de destaque é a interface de controle baseada em tecnologias web. A integração do servidor Apache com scripts em PHP permitiu o desenvolvimento de uma interface acessível, capaz de ser operada remotamente por meio de redes Wi-Fi. Essa abordagem amplia a acessibilidade do sistema, permitindo que usuários manipulem o telescópio de locais remotos sem necessidade de presença física, o que é particularmente vantajoso para observações em regiões com pouca interferência luminosa ou climática.

Adicionalmente, para velocidades mais altas o telescópio não apresenta essa precisão, porém para esse projeto é indiferente, tendo em vista que o objetivo central é o acompanhamento dos astros. O uso de engrenagens projetadas com precisão no software SolidWorks e fabricadas em impressoras 3D possibilitou um sistema mecânico, que embora seja de baixo valor aquisitivo e de fácil implementação, acabou não proporcionando um desempenho favorável ao sistema, visto que as engrenagens estavam se desgastando devido ao alto torque exigido pelo sistema.

Uma das etapas em que o projeto não obteve sucesso foi a validação do sistema, devido às condições climáticas desfavoráveis no momento dos testes. Essa etapa é essencial para confirmar se o sistema projetado realmente atende à problemática apresentada na introdução. Em projetos futuros, a realização dessa validação será imprescindível para

verificar se o desempenho do sistema corresponde aos objetivos propostos ou se ajustes serão necessários para alcançar os resultados esperados.

Embora o sistema tenha demonstrado um bom desempenho em muitos aspectos, alguns pontos negativos e limitações foram observados durante o desenvolvimento e testes:

1. Limitação na Escalabilidade do Sistema de Engrenagens. Embora o sistema mecânico seja funcional, a dependência de engrenagens impressas em 3D pode comprometer a durabilidade em aplicações de longo prazo ou em condições extremas, como temperaturas muito altas ou baixas. Substituir o materiais utilizado por outro mais resistentes, como ABS ou nylon reforçado, poderia aumentar a robustez do sistema.

2. Dependência de Configurações Manuais. Atualmente, a calibração inicial do telescópio (entrada de latitude e tempo de observação) precisa ser feita manualmente pelo usuário. A inclusão de sensores GPS e bússola digital poderia automatizar essa etapa, tornando o sistema mais intuitivo e fácil de usar.

3. Falta de Processamento de Imagens Integrado. Embora o sistema permita capturar imagens de corpos celestes, não há ferramentas integradas para processar ou melhorar essas imagens. Adicionar algoritmos de processamento, como ajustes automáticos de brilho e contraste, ou até mesmo funções de alinhamento astronômico, poderia elevar a qualidade dos resultados obtidos.

4. Ausência de Modo de Operação Portátil. O projeto depende de uma fonte de alimentação fixa, o que limita sua portabilidade. A implementação de uma bateria recarregável ou outro sistema de alimentação móvel aumentaria a versatilidade do telescópio.

Somado a tudo isso, houve problemas com equipamentos utilizados para esse projeto como o dispositivo de captura, que embora não possuíssem especificações adequadas para a prática da astrofotografia, eles são suficientes para realizar a etapa de testes e validações do sistema.

Apesar dos desafios enfrentados durante o desenvolvimento do projeto, é importante destacar que o trabalho realizado serve como uma base sólida para a implementação de futuras funcionalidades. A estrutura inicial, composta pelo sistema mecânico projetado,

os scripts de controle dos motores, a configuração da câmera e a interface que conecta o sistema ao usuário, oferece uma base para expansões e melhorias no sistema.

As expectativas para o futuro do projeto são amplas, reforçando seu papel como uma plataforma versátil e extensível. Entre as possibilidades futuras, destaca-se a implementação de algoritmos de processamento de imagem para identificar e acompanhar automaticamente corpos celestes em movimento, o que agregaria valor significativo para aplicações em astrofotografia e pesquisa científica.

Outra expansão potencial envolve o rastreamento de objetos de alta velocidade, como satélites de baixa órbita, incluindo a Estação Espacial Internacional. Essa aplicação exigiria ajustes na lógica de controle e na velocidade dos motores, mas se beneficiaria diretamente da infraestrutura de hardware e software já desenvolvida. Além disso, a adição de sensores GPS e giroscópios poderia permitir a automação completa do posicionamento inicial do telescópio, eliminando a necessidade de entrada manual de coordenadas.

No campo da automação, o sistema demonstra como tecnologias relativamente acessíveis podem ser utilizadas para resolver problemas complexos de forma eficaz. A escolha por motores de passo silenciosos, comunicação por FIFO e controle por *raspberry* não apenas garantiu o cumprimento dos objetivos, mas também demonstrou a viabilidade técnica e econômica da proposta.

Este trabalho para a automação de telescópios também incentiva a democratização do acesso à astronomia, trazendo a exploração do universo para mais perto de todos.

APÊNDICE A - CÓDIGOS: Interface Web

.1 Interface Web: index.php

```
<?php
// Lida com a requisi  o AJAX para verificar o arquivo
if (isset($_GET['check_image'])) {
    $image_path = 'img/img.png'; // Caminho relativo ao script PHP

    if (file_exists($image_path)) {
        echo json_encode(['exists' => true]);
    } else {
        echo json_encode(['exists' => false]);
    }
    exit; // Encerra o script ap s retornar a resposta
}
?>

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0
    ">
    <title>OrbitalView</title>
    <link rel="stylesheet" href="css/menu.css">
    <link rel="stylesheet" href="css/container.css">
    <link rel="stylesheet" href="css/btn.css">
    <link href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome
    /6.0.0/css/all.min.css" rel="stylesheet">

</head>
<body>

    <!--Menu Horizontal-->
    <header>
        <div class="menu">
            <div class="logo">OrbitalView</div>
        </div>
    </header>
```

```
35
36      <!--Organizando em 3 colunas-->
37      <div class="container">
38          <div class="edge-column"></div>
39
40          <div class="middle-column">
41
42              <div class="video-container">
43                  <img id="webcam-stream" src="/TCC/telecop/video_stream"
    alt="Webcam Stream">
44              </div>
45
46              <div class="btnBox" id="btn-box">
47                  <div class="btnRow">
48                      <button class="btn clip1" id="left-up" onmousedown="
    directionCmd('02','4','8')" onmouseup="stopCmd('01')"><i class="fas
    fa-arrow-right"></i></button>
49
50                      <button class="btn" onmousedown="directionCmd
    ('02','1','1')" onmouseup="stopCmd('01')"><i class="fas fa-arrow-up"
    ></i></button>
51
52                      <button class="btn clip1" id="right-up" onmousedown=
    "directionCmd('02','4','1')" onmouseup="stopCmd('01')"><i class="fas
    fa-arrow-right"></i></button>
53                  </div>
54
55
56
57                  <div class="btnRow">
58                      <button class="btn left btnSpace" onmousedown="
    directionCmd('02','2','4')" onmouseup="stopCmd('01')"><i class="fas
    fa-arrow-left"></i></btuton>
59                      <button class="btn right btnSpace" onmousedown="
    directionCmd('02','2','1')" onmouseup="stopCmd('01')"><i class="fas
    fa-arrow-right"></i></button>
60                  </div>
61                  <div class="btnRow">
62                      <button class="btn clip1" id="left-down" onmousedown
```

```
    ="directionCmd('02','4','4')" onmouseup="stopCmd('01')"><i class="fas
     fa-arrow-right"></i></button>
63
64                  <button class="btn" onmousedown="directionCmd
    ('02','1','4')" onmouseup="stopCmd('01')"><i class="fas fa-arrow-down
    "></i></button>
65
66                  <button class="btn clip1" id="right-down"
    onmousedown="directionCmd('02','4','2')" onmouseup="stopCmd('01')"><i
     class="fas fa-arrow-right"></i></button>
67              </div>
68
69          </div>
70
71          <script>
72          function directionCmd(cmd, motor, dir) {
73              var xhttp = new XMLHttpRequest();
74              xhttp.open("GET", "direction.php?cmd=" + cmd + "&motor="
     + motor + "&dir=" + dir, true);
75              xhttp.send();
76          }
77
78          function stopCmd(cmd) {
79              var xhttp = new XMLHttpRequest();
80              xhttp.open("GET", "direction.php?cmd=" + cmd, true);
81              xhttp.send();
82          }
83          </script>
84
85
86      </div>
87
88      <div class="edge-column">
89          <div class="config-box">
90              <div class="title-box">CONFIGURATION</div>
91              <div class="line"></div>
92
93              <h1 class="sub-title">Automatic Mode</h1>
94              <div class="line sub-line"></div>
```

```
95                <div class="form" id="view-start">
96                    <label for="oi">Latitude:</label>
97                    <select name="signal" id="signal" class="sendBox
    mini">
98                        <option value="1">+</option>
99                        <option value="0">-</option>
100                   </select>
101                   <input type="text" id="deg" name ="deg" placeholder=
    "graus" class="sendBox mini">
102                   <input type="text" id="min" name ="min" placeholder=
    "minutos" class="sendBox mini">
103                   <input type="text" id="sec" name ="sec" placeholder=
    "segundos" class="sendBox mini">
104
105                   <label for="oi">Tempo:</label>
106                   <input type="text" id="time_hour" name ="time_hour"
    placeholder="horas" class="sendBox mini">
107                   <input type="text" id="time_min" name ="time_min"
    placeholder="minutos" class="sendBox mini"> <br>
108                   <button class="view" onclick="automatiCmd('04')">
    Start</button>
109               </div>
110
111               <div class="form" id="stop-com" style="display:none;">
112                   <button class="view" onclick="control_mode()" id="
    btn-stop">Stop</button>
113               </div>
114
115               <script>
116
117               function control_mode(){
118                   document.getElementById('btn-box').style.display = "
    block";
119                   document.getElementById('view-start').style.display
    = "block";
120                   document.getElementById('stop-com').style.display =
    "none";
121
122               }
```

```
123                 function automatiCmd(cmd) {
124                     document.getElementById('btn-box').style.display = "
    none";
125                     document.getElementById('view-start').style.display
    = "none";
126                     document.getElementById('stop-com').style.display =
    "block";
127 
128                     var signal = document.getElementById('signal').value
    ;
129                     var deg = document.getElementById('deg').value;
130                     var min = document.getElementById('min').value;
131                     var sec = document.getElementById('sec').value;
132                     var T_hour = document.getElementById('time_hour').
    value;
133                     var T_min = document.getElementById('time_min').
    value;
134 
135 
136                     // Criar a requisi  o para enviar os dados via GET
137                     var xhttp = new XMLHttpRequest();
138                     var url = "automatic.php?cmd=" + cmd
139                     + "&signal=" + encodeURIComponent(signal) + "&deg="
    + encodeURIComponent(deg) +
140                             "&min=" + encodeURIComponent(min) +
141                             "&sec=" + encodeURIComponent(sec) +
142                             "&T_hour=" + encodeURIComponent(T_hour) +
143                             "&T_min=" + encodeURIComponent(T_min);
144                     xhttp.open("GET", url, true);
145                     xhttp.send();
146 
147                 }
148                 </script>
149 
150 
151                 <h1 class="sub-title">Camera</h1>
152                 <div class="line sub-line"></div>
153                 <div class="form">
154                     <label for="exposure">Exposure time:</label>
```

```
155                     <input type="text" id="exposure" name="exposure"
    placeholder="Insira o tempo de exposi  o" class="sendBox big">
156
157                     <label for="brightness">Brightness:</label>
158                     <input type="text" id="brightness" name="brightness"
     placeholder="Insira o brilho" class="sendBox big">
159
160                     <label for="contrast">Contrast:</label>
161                     <input type="text" id="contrast" name="contrast"
    placeholder="Insira o contraste" class="sendBox big">
162
163                     <label for="gain">Gain:</label>
164                     <input type="text" id="gain" name="gain" placeholder
    ="Insira o ganho" class="sendBox big">
165
166                     <label for="saturation">Saturation:</label>
167                     <input type="text" id="saturation" name="saturation"
     placeholder="Insira a satura  o" class="sendBox big">
168                     <br>
169                     <button id="wait-button" class="view" onmousedown="
    printRequest()">Print</button>
170                 </div>
171
172                 <script>
173                     document.getElementById('wait-button').
    addEventListener('click', function () {
174                         const container = document.getElementById('
    webcam-stream');
175
176                         // Fun  o para verificar a exist ncia da
    imagem
177                         function checkImageExists() {
178                             fetch('?check_image=1') // Requisi  o para
     o mesmo arquivo
179                                 .then(response => response.json())
180                                 .then(data => {
181                                     if (data.exists) {
182                                         container.src = 'img/img.png';
183                                     } else {
```

```
184                                     setTimeout(checkImageExists,
    1000);
185                                 }
186                             })
187                             .catch(err => console.error('Erro ao
    verificar imagem:', err));
188                     }
189 
190                     // Inicia a verifica  o
191                     checkImageExists();
192                 });
193             </script>
194 
195             <script>
196             function printRequest() {
197                 var exposure = document.getElementById('exposure').
    value;
198                 var brightness = document.getElementById('brightness
    ').value;
199                 var contrast = document.getElementById('contrast').
    value;
200                 var gain = document.getElementById('gain').value;
201                 var saturation = document.getElementById('saturation
    ').value;
202 
203                 // Criar a requisi  o para enviar os dados via GET
204                 var xhttp = new XMLHttpRequest();
205                 var url = "print.php?exposure=" + encodeURIComponent
    (exposure) +
206                         "&brightness=" + encodeURIComponent(
    brightness) +
207                         "&contrast=" + encodeURIComponent(contrast)
    +
208                         "&gain=" + encodeURIComponent(gain) +
209                         "&saturation=" + encodeURIComponent(
    saturation);
210 
211                 xhttp.open("GET", url, true);
212                 xhttp.send();
```

```
213
214                 }
215                 </script>
216
217             </div>
218
219         </div>
220     </div>
221
222 </body>
```

.2 Estilização da interface: menu.css

```
1  *{padding: 0; margin: 0; box-sizing: border-box;}
2
3  body {
4      margin: 0;
5      font-family: Arial, sans-serif;
6  }
7
8  .menu {
9      background-color: black;
10     color: white;
11     padding: 10px 20px;
12     display: flex;
13     justify-content: space-between;
14     align-items: center;
15 }
16
17 .menu .logo {
18     font-size: 20px;
19     font-weight: bold;
20 }
21
22 .menu nav a {
23     color: white;
24     text-decoration: none;
25     margin-left: 20px;
26 }
27
```

```
.menu nav a:hover {
    text-decoration: none;
}
```

.3 Estilização da interface: btn.css

```
.btnBox{
  margin: 0px;
  position: relative;
    width: 100%;
    display: flex;
    flex-direction: column;
}

.btnRow{
  display: flex;
  justify-content: center;
  flex-direction: row;
  width: 100%;
  position: relative;
  margin:0;
  padding:0;
}

.btn{
  padding:10px 35px;
  border:none;
  background: black;
  color:white;
  font-size: 20px;
  position: relative;
    transition-duration: 0.25s;
  clip-path: polygon(25% 6.7%, 75% 6.7%, 100% 50%, 75% 93.3%, 25% 93.3%,
    0% 50%);
  clip-path: polygon(25% 6.7%, 75% 6.7%, 100% 40%, 100% 60%, 75% 93.3%,
   25% 93.3%, 0% 60%, 0% 40%);
}

.btn:hover{
  background: indianred;
```

```
}

.btnSpace{
  margin:10px 50px;
}

.view{
  background: rgb(223,110,110);
    background: linear-gradient(70deg, rgba(223,110,110,1) 0%, rgba
   (163,58,58,1) 70%, rgba(167,49,88,1) 85%, rgba(170,22,22,1) 100%);
  border-radius: 25%/50%;
  padding:10px 35px;
  transition-duration: 0.25s;
  border: none;
  color: white;
}

.view:hover{
  animation: pulse 1s;
}

@keyframes pulse {
  0% {
    box-shadow: 0 0 0 0px rgba(223,110,110, 0.5);
  }
  100% {
    box-shadow: 0 0 0 20px rgba(223,110,110, 0);
  }
}

.sendBox{
  border: none;
  width: 70%;
  padding: 5px;
  margin: 10px 0 20px 0;
  border-radius: 5% / 50%;
  transition-duration: 0.1s;
}

```

```
.mini{
  width: 23%;
}

.big:hover{
  width: 100%;
  border-radius: 10% / 30%
}

#right-up{
  transform: rotate(-30deg);
  bottom :-12px;
  left:-14px;
}

#left-up{
  transform: rotate(-150deg);
  bottom :-12px;
  right:-14px;
}

#right-down{
  transform: rotate(30deg);
  top :-12px;
  left: -14px;
}

#left-down{
  transform: rotate(150deg);
  top :-14px;
  right:-14px;
}

.clip1{

  clip-path: polygon(5% 50%, 50% 10%, 95% 50%, 50% 90%);

}

```

```
.clip2{
  clip-path: polygon();
  top :-13px;
  margin: 0 3px;
}
```

.4 Estilização da interface: container.css

```
.container{
    width: 100%;
    position: relative;
    display: flex;
    flex-direction: row;
}

.edge-column{
    display: block;
    align-items: center;
    width: 25%;
}

.middle-column{
    width: 50%;
}

.video-container {
    margin:50px auto;
    width: 640px; /* Largura do v deo */
    height: 480px; /* Altura do v deo */
    background-color: #000; /* Fundo preto */
    border: 5px solid #333;
    position: relative;
}
.video-container img {
    width: 100%;
    height: 100%;
    object-fit: cover;
}

.config-box{
```

```
    box-shadow: 1px 1px 10px rgb(80,80,80);
    background: rgb(240, 240, 240);
    width: 80%;
    margin: auto;
    display: block;
}

.title-box{
    width: 100%;
    background: black;
    margin-top: 50px;
    color: white;
    padding: 20px !important;
    font-size: 20px;
    text-align: center;
    font-weight: bold;
}

.line{
    background: rgb(223,110,110);
    background: linear-gradient(45deg, rgba(223,110,110,1) 0%, rgba
   (163,58,58,1) 51%, rgba(167,49,88,1) 100%);
    width: 100%;
    height: 7px;
}

.sub-title{
    color: firebrick;
    font-size: 20px;
    margin: 20px 0 10px 50px;
    font-weight: bold;
}

.sub-line{
    width: 80%;
    margin: auto;
    height: 2px;
}

```

```
.form{
    width: 100%;
    padding: 30px 20px;
    display: inline-block;
    /*display: flex;*/
    /*flex-direction: column;*/
}

.form label{
    display: block;
}
```

APÊNDICE B - CÓDIGOS: Algoritmo PHP para comunicação com C

.1 Escrita no PIPE: automatic.php

```
<?php

if (isset($_GET['cmd'])) {
    $pidFile = '/fifo/teste/PID.txt';
    $file = '/fifo/teste/fifo';
    $cmd = $_GET['cmd'];

    if($cmd=="04"){
        echo "Que legal.\n";
        $signal = $_GET['signal'];
        $deg = $_GET['deg'];
        $min = $_GET['min'];
        $sec = $_GET['sec'];
        $T_hour = $_GET['T_hour'];
        $T_min = $_GET['T_min'];

        $message = $cmd . $signal . $deg . $min . $sec . $T_hour .
   $T_min; // Montando a mensagem
    } elseif ($cmd=="01") {
        $message = $cmd;
    }


    // Verificar se o FIFO existe, sen o criar
```

```
24      if (!file_exists($file)) {
25          if (!posix_mkfifo($file, 0777)) {
26              die("Erro ao criar o Named Pipe.\n");
27          }
28      }
29
30      // Abrir o FIFO para escrita e enviar o comando do motor
31      $pipe = fopen($file, 'w');
32      if ($pipe) {
33          fwrite($pipe, $message);
34          fclose($pipe);  // Fechar o FIFO ap s a escrita
35      } else {
36          echo "Erro ao abrir o pipe para escrita.\n";
37      }
38
39
40      // Verificar se o arquivo de PID existe
41      if (file_exists($pidFile)) {
42          // Ler o PID do arquivo
43          $pid = file_get_contents($pidFile);
44
45          // Verificar se o PID    v lido
46          if (is_numeric($pid)) {
47              // Monta o comando para enviar o sinal SIGHUP ao processo
48              $command = "/usr/bin/sudo kill -HUP " . escapeshellarg($pid)
    ;
49
50              // Executa o comando
51              exec($command, $output, $return_var);
52
53              // Verifica o resultado
54              if ($return_var === 0) {
55                  echo "Sinal HUP enviado com sucesso para o processo com
    PID $pid.\n";
56              } else {
57                  echo "Falha ao enviar o sinal para o processo com PID
    $pid. C digo de retorno: $return_var\n";
58              }
59          } else {
```

```
60              echo "PID inv lido no arquivo $pidFile.\n";
61          }
62      } else {
63          echo "Arquivo $pidFile n o encontrado.\n";
64      }
65
66  }
67  ?>
```

.2 Escrita no PIPE: direction.php

```
1   <?php
2
3   if (isset($_GET['cmd'])) {
4       $pidFile = '/fifo/teste/PID.txt';
5       $file = '/fifo/teste/fifo';
6       $cmd = $_GET['cmd'];
7
8       if($cmd=="2"){
9           $motor = $_GET['motor'];
10          $dir = $_GET['dir'];
11          $message = $cmd . $motor . $dir;
12      } elseif ($cmd=="1") {
13          $message = $cmd;
14      }
15
16
17      // Verificar se o FIFO existe, sen o criar
18      if (!file_exists($file)) {
19          if (!posix_mkfifo($file, 0777)) {
20              die("Erro ao criar o Named Pipe.\n");
21          }
22      }
23
24      // Abrir o FIFO para escrita e enviar o comando do motor
25      $pipe = fopen($file, 'w');
26      if ($pipe) {
27          fwrite($pipe, $message);
28          fclose($pipe);  // Fechar o FIFO ap s a escrita
29          echo "Comando '$message' enviado para o FIFO com sucesso.\n";
```

```
30      } else {
31          echo "Erro ao abrir o pipe para escrita.\n";
32      }
33
34
35      // Verificar se o arquivo de PID existe
36      if (file_exists($pidFile)) {
37          // Ler o PID do arquivo
38          $pid = file_get_contents($pidFile);
39
40          // Verificar se o PID    v lido
41          if (is_numeric($pid)) {
42              // Monta o comando para enviar o sinal SIGHUP ao processo
43              $command = "/usr/bin/sudo kill -HUP " . escapeshellarg($pid)
    ;
44
45              // Executa o comando
46              exec($command, $output, $return_var);
47
48              // Verifica o resultado
49              if ($return_var === 0) {
50                  echo "Sinal HUP enviado com sucesso para o processo com
    PID $pid.\n";
51              } else {
52                  echo "Falha ao enviar o sinal para o processo com PID
    $pid. C digo de retorno: $return_var\n";
53              }
54          } else {
55              echo "PID inv lido no arquivo $pidFile.\n";
56          }
57      } else {
58          echo "Arquivo $pidFile n o encontrado.\n";
59      }
60  }
61  ?>
```

.3 Escrita no PIPE: print.php

```
1  <?php
2
```

```php
// Verificar se todos os par metros esperados est o presentes
if (isset($_GET['exposure'], $_GET['brightness'], $_GET['contrast'],
    $_GET['gain'], $_GET['saturation'])) {
    // Obter os valores dos par metros
    $exposure = escapeshellarg($_GET['exposure']);
    $brightness = escapeshellarg($_GET['brightness']);
    $contrast = escapeshellarg($_GET['contrast']);
    $gain = escapeshellarg($_GET['gain']);
    $saturation = escapeshellarg($_GET['saturation']);

    // Montar o comando para executar o programa com os par metros
    $command = "./photo $exposure $brightness $contrast $gain
    $saturation";

    // Executa o comando
    exec($command, $output, $return_var);

    // Para depura  o, voc  pode exibir a sa da e o status do
    retorno
    echo "Sa da: " . implode("\n", $output) . "<br>";
    echo "Status do retorno: " . $return_var;
} else {
    echo "Erro: Par metros insuficientes fornecidos.";
}
?>
```

APÊNDICE B - CÓDIGOS: Algoritmo C de aplicação

.1 Aplicação para controle do motor: motorCtr.c

```c
#include <wiringPi.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <fcntl.h>
#include <unistd.h>
#include <string.h>
#include <signal.h>
#include <semaphore.h>
#include <pthread.h>
```

```
11
12 #define PIN_STEP_X 0  // GPIO 17
13 #define PIN_DIR_X 1   // GPIO 18
14 #define PIN_STEP_Y 3  // GPIO 17
15 #define PIN_DIR_Y 4   // GPIO 18
16 #define PIPE_FILE "/fifo/teste/fifo"
17
18 #define MOTOR_STEP 1.8/8
19 #define Ix 19.2*120./17
20 #define Iy 19.2*206./18
21 #define DAY ((23.*60.+56.)*60.+4.)
22 #define ROT 360.
23
24 // Vari veis globais
25 int fd;                    // FILE descriptor do FIFO
26 char CMD[10000] = "";      // Comando
27 char command[2] = "";
28 int com = 0;
29 unsigned int motor, dir;
30 unsigned int deg_singnal, deg, deg_min, deg_sec, hour, min;
31 float timeX, timeY;
32 unsigned long long int duration = 0;
33 int process;
34 sem_t S1;
35
36 // Struct para argumentos das threads
37 typedef struct {
38     int dlyTime;
39     int stepPin;
40 } MotorArgs;
41
42 // Fun   es de inicializa  o
43 void setupTMC2208() {
44     pinMode(PIN_STEP_X, OUTPUT);
45     pinMode(PIN_DIR_X, OUTPUT);
46     pinMode(PIN_STEP_Y, OUTPUT);
47     pinMode(PIN_DIR_Y, OUTPUT);
48 }
49
```

```
50 void initFIFO() {
51     fd = open(PIPE_FILE, O_RDONLY);
52     if (fd == -1) {
53         perror("Erro ao abrir o Named Pipe");
54         exit(EXIT_FAILURE);
55     }
56 }
57 
58 void saveProcessID() {
59     process = getpid();
60     FILE *pidFile = fopen("/fifo/teste/PID.txt", "w");
61     if (pidFile == NULL) {
62         perror("N o foi poss vel criar o arquivo PID.txt");
63         exit(EXIT_FAILURE);
64     }
65     fprintf(pidFile, "%d", process);
66     fclose(pidFile);
67 }
68 
69 // Manipulador de sinal
70 void handler(int signum) {
71     ssize_t bytesRead;
72     memset(CMD, 0, sizeof(CMD));
73 
74     while ((bytesRead = read(fd, CMD, sizeof(CMD) - 1)) > 0) {
75         CMD[bytesRead] = '\0';
76         printf("Dados recebidos: %s\n", CMD);
77     }
78 
79     if (bytesRead < 0) {
80         perror("Erro ao ler o FIFO");
81     }
82 
83     sscanf(CMD, "%2s", command);
84     com = strtol(command, NULL, 16);
85     if (com & 0x02) {
86         sscanf(CMD + 2, "%1x%1x", &motor, &dir);
87     } else if (com & 0x04) {
88         sscanf(CMD + 2, "%1x%2d%2d%2d%2d%2d", &deg_singnal, &deg, &
```

```
    deg_min, &deg_sec, &hour, &min);
89          duration = (hour * 60 + min) * 60 * 10000;
90          printf("%llu\n", duration);
91      }
92
93      sem_post(&S1);
94  }
95
96  // Fun   o para controle dos motores
97  void* direction(void* arg) {
98      MotorArgs* motorArgs = (MotorArgs*) arg;
99      int dlyTime = motorArgs->dlyTime;
100     int stepPin = motorArgs->stepPin;
101
102     while (1) {
103         if (com & 0x01) {
104             break;
105         }
106         digitalWrite(stepPin, HIGH);
107         delayMicroseconds(dlyTime / 2.0);
108         digitalWrite(stepPin, LOW);
109         delayMicroseconds(dlyTime / 2.0);
110     }
111     return NULL;
112 }
113
114 // Fun   es auxiliares
115 void calcDlyy() {
116     float lat = deg + (deg_min + deg_sec / 60.0) / 60.0;
117     float rad = lat * (M_PI / 180.0);
118     timeX = 1000000 * (DAY * MOTOR_STEP / (ROT * Ix)) / cos(rad);
119     timeY = 1000000 * (DAY * MOTOR_STEP / (ROT * Iy)) / sin(rad);
120 }
121
122 void waitOneMinute() {
123     struct timespec req = { .tv_sec = 60, .tv_nsec = 0 };
124     nanosleep(&req, NULL);
125 }
126
```

```
127 void controlMotors() {
128     //thread1 corresponde ao acionamento do motorX e thread2 para o
    motor y
129     pthread_t thread1, thread2;
130 
131     if (com & 0x02) {
132         if (motor & 0x01) {
133             digitalWrite(PIN_DIR_X, dir & 0x01 ? HIGH : LOW);
134             MotorArgs motorArgs1 = {200, PIN_STEP_X};
135             pthread_create(&thread1, NULL, direction, &motorArgs1);
136             pthread_join(thread1, NULL);
137         } else if (motor & 0x02) {
138             digitalWrite(PIN_DIR_Y, dir & 0x01 ? HIGH : LOW);
139             MotorArgs motorArgs2 = {200, PIN_STEP_Y};
140             pthread_create(&thread2, NULL, direction, &motorArgs2);
141             pthread_join(thread2, NULL);
142         } else if (motor & 0x04) {
143             digitalWrite(PIN_DIR_X, dir & 0x01 ? HIGH : LOW);
144             digitalWrite(PIN_DIR_Y, dir & 0x04 ? HIGH : LOW);
145             MotorArgs motorArgs1 = {200, PIN_STEP_X};
146             MotorArgs motorArgs2 = {200, PIN_STEP_Y};
147             pthread_create(&thread1, NULL, direction, &motorArgs1);
148             pthread_create(&thread2, NULL, direction, &motorArgs2);
149             pthread_join(thread1, NULL);
150             pthread_join(thread2, NULL);
151         }
152     } else if (com & 0x04) {
153         calcDlyy();
154         digitalWrite(PIN_DIR_X, LOW);
155         digitalWrite(PIN_DIR_Y, deg_singnal & 0x01 ? LOW : HIGH);
156         MotorArgs motorArgs1 = {timeX, PIN_STEP_X};
157         MotorArgs motorArgs2 = {timeY, PIN_STEP_Y};
158         pthread_create(&thread1, NULL, direction, &motorArgs1);
159         pthread_create(&thread2, NULL, direction, &motorArgs2);
160 
161         unsigned long startTime = micros();
162         while (1) {
163             unsigned long elapsedTime = micros() - startTime;
164             if (elapsedTime >= duration * 100) {
```

```
165                 pthread_cancel(thread1);
166                 pthread_cancel(thread2);
167                 break;
168             }
169             waitOneMinute();
170         }
171 
172         pthread_join(thread1, NULL);
173         pthread_join(thread2, NULL);
174     }
175 }
176 
177 // Fun   o principal
178 int main() {
179     sem_init(&S1, 0, 0);
180 
181     if (wiringPiSetup() == -1) {
182         printf("Erro ao inicializar o wiringPi.\n");
183         return 1;
184     }
185 
186     setupTMC2208();
187     saveProcessID();
188     initFIFO();
189 
190     if (signal(SIGHUP, handler) == SIG_ERR) {
191         perror("N  o foi poss vel registrar o manipulador de sinal");
192         exit(EXIT_FAILURE);
193     }
194 
195     while (1) {
196         sem_wait(&S1);
197         controlMotors();
198     }
199 
200     return 0;
201 }
```

.2 Aplicação para configuração da camera e a captura de imagem: photo.c

```c
#include <stdio.h>
#include <stdlib.h>
#include <pthread.h>
#include <unistd.h> // Para usar sleep()

void *record_video(void *arg) {

    const char *command = "ffmpeg -y -f v4l2 -i /dev/video0 -ss 3 -frames:v 1 -q:v 2 img/img.png";

    // Executa o comando do sistema para iniciar a grava  o
    int ret = system(command);
    if (ret == -1) {
        perror("Erro ao executar o comando de grava  o");
        pthread_exit(NULL);
    }

    pthread_exit(NULL);
}

void *configure_camera(void *arg) {
    sleep(1);

    // Recebe os valores de exposi  o e ganho do argumento
    int *config = (int *)arg;
    int exposure_time = config[0];
    int brightness = config[1];
    int contrast = config[2];
    int gain = config[3];
    int saturation = config[4];

    // Monta os comandos para alterar os controles da c mera
    char command[256];
    int ret;

    snprintf(command, sizeof(command), "v4l2-ctl -d /dev/video0 --set-ctrl=exposure_time_absolute=%d", exposure_time);
    ret = system(command);
    if (ret == -1) {
```

```
38          perror("Erro ao configurar tempo de exposi  o");
39          pthread_exit(NULL);
40      }
41
42      snprintf(command, sizeof(command), "v4l2-ctl -d /dev/video0 --set-
    ctrl=brightness=%d", brightness);
43      ret = system(command);
44      if (ret == -1) {
45          perror("Erro ao configurar brilho");
46          pthread_exit(NULL);
47      }
48
49      snprintf(command, sizeof(command), "v4l2-ctl -d /dev/video0 --set-
    ctrl=contrast=%d", contrast);
50      ret = system(command);
51      if (ret == -1) {
52          perror("Erro ao configurar contraste");
53          pthread_exit(NULL);
54      }
55
56      snprintf(command, sizeof(command), "v4l2-ctl -d /dev/video0 --set-
    ctrl=gain=%d", gain);
57      ret = system(command);
58      if (ret == -1) {
59          perror("Erro ao configurar ganho");
60          pthread_exit(NULL);
61      }
62
63      snprintf(command, sizeof(command), "v4l2-ctl -d /dev/video0 --set-
    ctrl=saturation=%d", saturation);
64      ret = system(command);
65      if (ret == -1) {
66          perror("Erro ao configurar satura  o");
67          pthread_exit(NULL);
68      }
69
70      pthread_exit(NULL);
71  }
72
```

```
73 int main(int argc, char *argv[]) {
74     if (argc != 6) {
75         fprintf(stderr, "Uso: %s <tempo_de_exposicao> <brilho> <
    contraste> <ganho> <saturacao>\n", argv[0]);
76         return 1;
77     }
78 
79     int exposure_time = atoi(argv[1]);
80     int brightness = atoi(argv[2]);
81     int contrast = atoi(argv[3]);
82     int gain = atoi(argv[4]);
83     int saturation = atoi(argv[5]);
84 
85     // Valida  o dos valores de entrada
86     if (exposure_time <= 0 || brightness < 0 || contrast < 0 || gain < 0
     || saturation < 0) {
87         fprintf(stderr, "Erro: Os valores devem ser positivos e o tempo
    de exposi  o maior que 0.\n");
88         return 1;
89     }
90 
91     int config[5] = {exposure_time, brightness, contrast, gain,
    saturation};
92 
93     pthread_t thread1, thread2;
94 
95     // Cria a primeira thread para gravar o v deo
96     if (pthread_create(&thread1, NULL, record_video, NULL) != 0) {
97         perror("Erro ao criar a thread de grava  o");
98         return 1;
99     }
100 
101     // Cria a segunda thread para configurar a c mera
102     if (pthread_create(&thread2, NULL, configure_camera, config) != 0) {
103         perror("Erro ao criar a thread de configura  o");
104         return 1;
105     }
106 
107     // Aguarda as threads terminarem
```

```
108     pthread_join(thread1, NULL);
109     pthread_join(thread2, NULL);
110 
111     return 0;
112 }
```

.3 Inicialização do script do motor ao iniciar o sistema operacional: boot.service

```
1  [Unit]
2  Description=Meu script para testes
3  Wants=network-online.target
4  After=network-online.target
5  
6  [Service]
7  ExecStart=/var/www/html/telescope/sucesso
8  User=root
9  Type=simple
10 IOSchedulingClass=realtime
11 IOSchedulingPriority=0
12 CPUSchedulingPolicy=rr
13 CPUSchedulingPriority=99
14 
15 [Install]
16 WantedBy=multi-user.target
```

BIBLIOGRAFIA UTILIZADA

[1] G. B. LIMA NETO, "Astronomia de posição," *Notas de Aula do Curso AGA*, vol. 106, 2018.

[2] KStars. (2018) Stellarium web. Acessado em 17/04/2024. [Online]. Available: https://kstars.kde.org/

[3] Stellarium. Stellarium. Acessado em 17/04/2024. [Online]. Available: https://stellarium.org/

[4] R. A. Serway and J. W. J. Jr., *Física para Cientistas e Engenheiros - Volume 4 - Luz, Óptica e Física Moderna*, 9th ed. Cengage Learning Brasil, 2019, tradução da 9ª edição norte-americana.

[5] J. TEXEREAU, "How to make a telescope." *Garden City*, 1963.

[6] K. d. S. Oliveira Filho and M. d. F. O. Saraiva, "Astronomia e astrofísica," *São Paulo: Editora Livraria da Física*, vol. 780, no. 2004, p. 183, 2004.

[7] J. P. WILSON, Charles E.; SADLER, *Kinematics and Dynamics of Machinery*, 3rd ed. Pearson Education Limited, 2014.

[8] J. C. d. Almeida, K. F. d. Lima, and R. Barbieri, *Elementos de máquinas: projeto de sistemas mecânicos*. São Paulo: Editora Blucher, 2022.

[9] F. Provenza, "Pro-tec-desenhista de máquinas," *São Paulo: F. Provenza*, 1991.

[10] Apache Software Foundation, *Apache HTTP Server Documentation*, 2024, acessado em: 15 de setembro de 2024. [Online]. Available: https://httpd.apache.org/

[11] S. Andrew, H. Bos *et al.*, *Sistemas operacionais modernos*. Bookman Editora, 2024.