Universidade de Brasília
Faculdade de Tecnologia

Desenvolvimento de um Braço Robótico de 6 Graus de Liberdade para Fins Cinematográficos

Alexander Paschoaletto

TRABALHO DE GRADUAÇÃO

ENGENHARIA DE CONTROLE E AUTOMAÇÃO

Brasilía

2022

Universidade de Brasília
Faculdade de Tecnologia

Desenvolvimento de um Braço Robótico de 6 Graus de Liberdade para Fins Cinematográficos

Alexander Paschoaletto

Trabalho de Graduação submetido como requisito parcial para obtenção do grau de Engenheiro de Controle e Automação.

Orientador: Profa. Dra. Carla Koike
Coorientador: Prof. Dr. Jones Yudi

Brasilía
2022

Paschoaletto, Alexander.
P279d Desenvolvimento de um Braço Robótico de 6 Graus de Liberdade para Fins Cinematográficos / Alexander Paschoaletto; orientador Carla Koike; coorientador Jones Yudi. -- Brasilía, 2022.
116 p.

Trabalho de Graduação em Engenharia de Controle e Automação -- Universidade de Brasília, 2022.

1. Robótica. 2. Desenvolvimento. 3. Filmagem automatizada. 4. Maker. I. Koike, Carla, orient. II. Yudi, Jones, coorient. III. Título

Universidade de Brasília
Faculdade de Tecnologia

Desenvolvimento de um Braço Robótico de 6 Graus de Liberdade para Fins Cinematográficos

Alexander Paschoaletto

Trabalho de Graduação submetido como requisito parcial para obtenção do grau de Engenheiro de Controle e Automação.

Trabalho aprovado. Brasilía, 5 de maio de 2022:

Profa. Dra. Carla Maria Chagas E Cavalcante Koike, CIC/UnB
Orientadora

Prof. Dr. Guilherme Caribé de Carvalho
Examinador interno

Profa. Dra. Dianne Magalhães Viana
Examinador externo

Eng. Cassio Urias Furlan
Examinador externo

Brasilía
2022

Este trabalho é dedicado àqueles que não se acomodam,
que não se conformam, que querem inspirar e melhorar o mundo.

“It had long since come to my attention
that people of accomplishment rarely
sat back and let things happen to them.
They went out and happened to things.”
(Leonardo Da Vinci)

Resumo

O inegável crescimento do movimento Maker ao redor do mundo, aliado à popularização de tecnologias de fabricação como o corte a laser e a impressão 3D, têm sido de vital importância para democratizar a inovação na sociedade. A iniciativa e capacidade de desenvolver novos produtos não mais é restrita a empresas nem precisa custar uma fortuna, podendo pessoas comuns também desenvolverem e prototiparem suas próprias soluções em um espaço curto de tempo e a um custo plenamente viável. Este trabalho descreve o projeto e a implementação de um manipulador robótico de baixo custo, aos moldes cinemáticos dos principais modelos presentes na indústria e com o objetivo de ser usado como suporte para filmagens automatizadas. Para atingir este feito parte-se de um robô inicial de propósito geral batizado de Pégaso, analisa-se suas características mecânicas, eletrônicas e computacionais, observa-se o desempenho relacionado à aplicação cinematográfica e propõe-se mudanças nas três áreas mencionadas que refinem sua performance. O robô final é um modelo amplamente modificado, com maior volume de trabalho e capacidades técnicas teóricas superiores às do modelo original.

Palavras-chave: Robótica. Desenvolvimento. Filmagem automatizada. Maker.

Abstract

The undeniable expansion of the Maker initiative worldwide, along with the dissemination of manufacturing techniques such as laser cutting and 3D printing, have been of vital importance on yielding access to innovation on society. The action and capacity of developing new products is no longer restricted to companies or has to cost a fortune, being regular people also able to develop and prototype their own solutions in a short amount of time and at a perfectly accessible fashion. The present paper describes the project and implementation of a low-cost robotic manipulator, following the kinematic model of them main options available on industry and focusing on the use as a support on making automated movie takes. In order to accomplish this goal a general purpose robotic arm named Pégaso is used as a starting point, with its mechanical, electronic and computational characteristics being analyzed, as well as its capacity of making movie takes. It is then proposed and implemented changes on the analyzed issues to further improve the robot performance. The final product is a heavily modified model, with greater workspace and enhanced theoretical capabilities when compared to the original model.

Keywords: Robotics. Development. Automated film-making. Maker.

Lista de ilustrações

Lista de símbolos

Símbolos Latinos

- Z_A número de dentes da engrenagem sol
- Z_B número de dentes da engrenagem planeta
- Z_C número de dentes da engrenagem anel
- X eixo do sistema de cordenadas cartesiano ortogonal a Y e Z
- Y eixo do sistema de cordenadas cartesiano ortogonal a X e Z
- Z eixo do sistema de coordenadas cartesiano ortogonal a X e Y
- U eixo do sistema de coordenadas cilíndrico ortogonal a U
- V eixo do sistema de coordenadas cilíndrico ortogonal a V
- Ex eixo do sistema de coordenadas do efetuador ortogonal a Ey e Ez
- Ey eixo do sistema de coordenadas do efetuador ortogonal a Ex e Ez
- Ez eixo do sistema de coordenadas do efetuador ortogonal a Ex e Ey
- L distância entre a junta D e a junta E (Link D) [mm]
- Lx Projeção de L no eixo X do referencial de origem [mm]
- Ly Projeção de L no eixo Y do referencial de origem [mm]
- Lz Projeção de L no eixo Z do referencial de origem [mm]
- a comprimento do link entre a junta atual e a próxima
- b distância entre a próxima junta da cadeia e o efetuador terminal
- c distância entre a junta atual e o alvo
- $\vec{a}$ Vetor A
- $\vec{c}$ Vetor C
- i contador
- N limite do contador

Símbolos Gregos

- α ângulo [°]
- β ângulo [°]
- δ ângulo [°]
- θ ângulo [°]

Siglas

- FDM *Fused Deposition Modeling*
- PLA *Polylactic Acid*
- PETG *Polyethylene terephthalate glycol*
- ARM *Advanced RISC Machines*
- AVR *Alf (Egil Bogen) and Vegard (Wollan) 's RISC Processor*
- SRAM *Static Random Access Memory*
- EEPROM *Electronically Erasable Programmable Read-Only Memory*
- UART *Universal Assynchronous Receiver/Transmitter*
- I2C *Inter-Integrated Circuit*
- SPI *Serial Peripheral Interface*
- RS232 *Recommended Standard 232*
- OTG *On-The-Go*
- IDE *Integrated Development Environment*
- PIO *Programmable Input/Output*
- NEMA *National Electrical Manufacturers Association*
- HTTP *Hypertext Transfer Protocol*

Sumário

1 Introdução

1.1 Contexto

A robótica é uma área da ciência orientada à construção de máquinas capazes de realizar tarefas repetitivas de maneira automatizada, precisa e coordenada por humanos. Originalmente nascidos para operar em ambientes fabris em meio ao século XX, os robôs expandiram suas aplicações e atualmente têm participação em várias áreas da sociedade, sendo as principais industrial, médica, audiovisual, educativa e pessoal (PAGLIARINI; LUND, 2017). Com isso, hoje em dia é possível contar com robôs para fabricar carros, performar cirurgias, realizar filmagens de comerciais e até aspirar o chão.

Especialmente na última década, com a popularização massiva das impressoras 3D e o fortalecimento da cultura *Maker*, a pesquisa e o desenvolvimento de novas aplicações para a robótica têm tido custos consistentemente mais acessíveis, além de poderem ser feitos por pessoas comuns no conforto de suas casas. Por consequência o leque de aplicações da robótica também aumenta, e mostra-se capaz de atender demandas tradicionalmente consideradas muito específicas para justificarem os custos de uma produção industrial com economia de escala.

1.2 Filmagem Robotizada

Uma das aplicações que se encaixa na categoria é a filmagem robotizada. A adaptação de manipuladores industriais para realizar gravações de filmes já mostrou potencial em *blockbusters* como o filme Gravidade (2013)(VENTUREBEAT, 2013), conseguindo realizar movimentações em cena até então incompatíveis com recursos tradicionais. Mais recentemente (2018), canais de YouTube como o de tecnologia Marques Brownlee (BROWNLEE, 2018) passaram a adotar tais robôs para filmar objetos como celulares e fones de ouvido em vídeos de avaliação destes produtos. A limitação óbvia da adaptação industrial, no entanto, é o conjunto de espaço, orçamento e capacitação necessários para utilizar o sistema. Frente a estes fatores e considerando um possível aumento na abrangência do público-alvo desta aplicação, o movimento *Maker* demonstra claro potencial em gerar soluções alternativas que permitam aliar produção em baixa escala com custo de implementação acessível (HOLM, 2015), possivelmente até mantendo a curva de aprendizado rápida para atrair o uso por artistas independentes e criadores de conteúdo do próprio YouTube.

1.3 Escolha do Projeto

Dado o cenário descrito acima, foi identificado um potencial ainda não explorado de adaptar o conceito de filmagem robotizada para um novo formato que democratize o acesso a este tipo de utilidade. A escolha do projeto, portanto, foi de desenvolver a aplicação cinematográfica em um robô de baixo custo, implementando nele as adaptações e melhorias necessárias para cumprir adequadamente a função.

Neste contexto havia um robô perfeito para servir de ponto de partida no desenvolvimento deste trabalho: um protótipo de manipulador robótico batizado de Pégaso. Ele havia sido confeccionado sem uma funcionalidade ou objetivo definidos até o início deste trabalho de graduação, quando passou a levar a finalidade cinematográfica em conta para toda modificação que se sucedeu. A partir deste ponto, passou a se chamar PegasoV3.

1.4 Objetivos

O objetivo principal deste trabalho é desenvolver a aplicação de filmagem robotizada para o Projeto Pégaso, permitindo que um usuário seja capaz de configurá-lo sem dificuldades para performar uma variedade de tomadas de vídeo simples. Mais especificamente, pode-se citar:

- A revisão e melhoria do projeto mecânico, elétrico e eletrônico do robô;
- O desenvolvimento e a implementação de um modelo cinemático adequado; e
- Implementação de um método de controle de fácil operação para um usuário comum, permitindo que ele possa configurar o robô para realizar movimentos planejados.

1.5 Estrutura do Documento

O presente documento é organizado da seguinte maneira: O Capítulo 2 aborda os trabalhos anteriores utilizados como base para a elaboração do mostrado nestas páginas. O Capítulo 3 descreve o robô original, usado como ponto de partida, e resume os principais aspectos a serem trabalhados em uma nova versão. O Capítulo 4 discorre sobre a escolha de novos componentes eletrônicos e sobre as mudanças que estes refletiram no código-fonte e interface. O Capítulo 5 disserta sobre a evolução da estrutura física do robô para atender aos pontos de melhoria citados no Capítulo 3. O Capítulo 6 aborda a análise de trabalhos anteriores sobre cinemática inversa e a implementação no robô de uma adaptação do Método de Triangulação. Por fim, o Capítulo 7 menciona as mudanças adicionais feitas ao robô com o intuito de proporcionar uma melhoraria quanto à suavidade da movimentação, e o Capítulo 8 conclui o trabalho e menciona implementações futuras.

2 Trabalhos Correlatos

O desenvolvimento dos chamados robôs articulados de mesa - inclusive para uso como suporte de filmagem (WLKATA, 2019) - não é inédito, mas as informações disponibilizadas são escassas e ocasionalmente advindas de produtos comerciais, com apenas parte de seu conteúdo disponível no formato *open-source* (WLKATA, s.d.). Trabalhos científicos que exploram o tema da robótica aliada a câmeras variam de assistência em procedimentos cirúrgicos (DAS et al., 2010) ao cálculo de caminhos e desvio de obstáculos por plataformas móveis (.R; HONNARAJU; MURALI, 2015), mas somente em (SIEMASZ; TOMCZUK; MALECHA, 2020) é mostrado um conteúdo com foco próximo ao deste trabalho: o desenvolvimento integral de um robô nas tipologias comuns ao meio industrial - cartesiano, articulado, SCARA, etc - com o emprego de técnicas acessíveis de fabricação. No entanto, o artigo não aborda detalhes quanto às escolhas na construção do robô, que teve seu projeto aproveitado de um modelo *open-source* existente.

Frente a esse cenário, o trecho do desenvolvimento do robô descrito neste trabalho que de fato contou com trabalhos anteriores para ser desenvolvido foi o da implementação do controle cartesianso, onde foram analisados os artigos (MULLER-CAJAR; MUKUNDAN, 2007), (ARISTIDOU; LASENBY, 2011) e (WANG; CHEN, 1991) e suas diferentes propostas para solução da cinemática inversa até finalmente escolher aquele que julgou mais adequado para servir de base para o PegasoV3. A análise em maiores detalhes das propostas avaliadas estão contidas no capítulo 6, que discute esta parte do desenvolvimento.

3 O Robô Inicial

Conforme dito na introdução, um robô compatível com a aplicação proposta já existia quando este trabalho teve início, estando confome mostrado na figura 1. Seu desenvolvimento foi idealizado em caráter de hobby e sem enfoque em uma aplicação específica. Ao longo das próximas seções serão explorados os principais detalhes que o compunham antes de começar a receber os aprimoramentos relacionados à adaptação para o propósito cinematográfico.

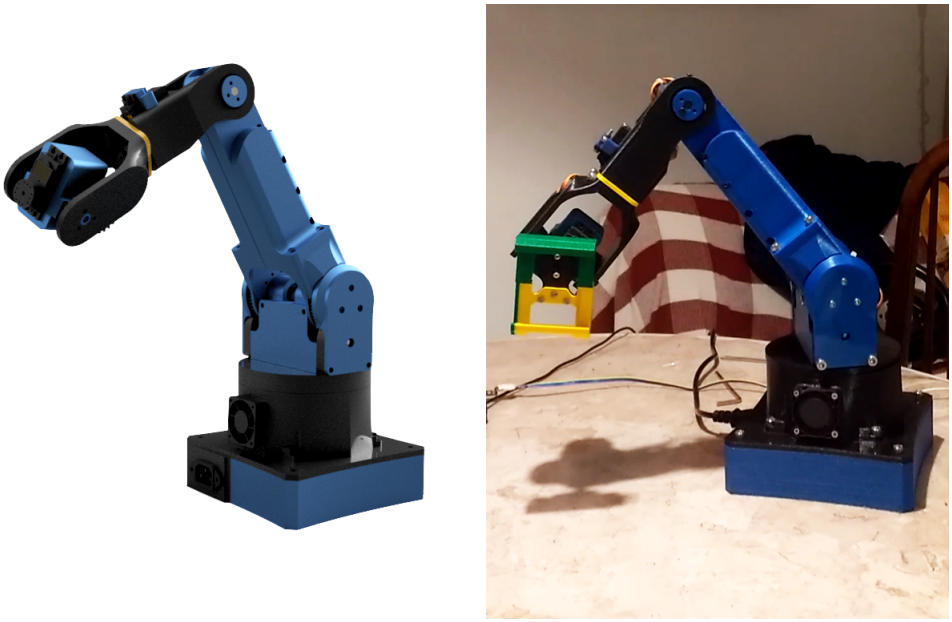


Figura 1 – Desenho 3D e Robô físico no início deste Trabalho.

3.1 Estrutura e Motorização

Todas as peças que compõem a estrutura do robô, com exceção do sistema de transmissão, são projetadas no software Autodesk Fusion 360 e fabricadas em uma impressora 3D de tecnologia FDM. Algumas seções do robô são feitas em PETG (partes azuis da figura 1) e outras em PLA (pretas, verdes e amarelas). Este último, apesar de propiciar peças com qualidade de acabamento em geral superior às de PETG, não é uma solução ideal por se tratar de um plástico que perde resistência mecânica em temperaturas relativamente baixas (cerca de 60ºC) (SUDER et al., 2021), o que pode ser um problema relevante dado o aquecimento potencial dos motores em longos períodos de funcionamento. Logo, é conveniente traçar o primeiro ponto de mudança como a substituição de todas as peças de PLA por alternativas de PETG conforme novas partes são confeccionadas para atender as melhorias.

Quanto à motorização, sendo um manipulador robótico com proposta cinemática similar aos principais modelos industriais - tais como o FANUC LRMate(FANUC, s.d.) e

o ABB IRB 1200(ABB, s.d.[a]) -, sua configuração segue o padrão de 6 juntas (graus de liberdade), sendo três para o corpo e três para o punho. O primeiro conjunto, posicionado próximo à base, conta com três motores de passo NEMA 17. O segundo, mais próximo do efetuador terminal, é movido por meio de três servomotores TowerPro MG-995. A escolha de cada arranjo se deu levando em conta o preço, peso e controlabilidade. Os MG-995, Apesar de terem um deslocamento limitado a 180° e um passo mínimo de 1°, também têm, em relação aos motores de passo, 1/4 do peso, torque considerável de 110N.cm (TOWERPRO, s.d.), custo similar e maior facilidade de controle. Afim de facilitar a referência a cada junta, adotou-se a nomenclatura ilustrada na figura 2: Juntas Base, A, B, C, D e E. Os motores de cada uma recebem os mesmos nomes para facilitar a associação: Motor Base, A, B, C, D e E.

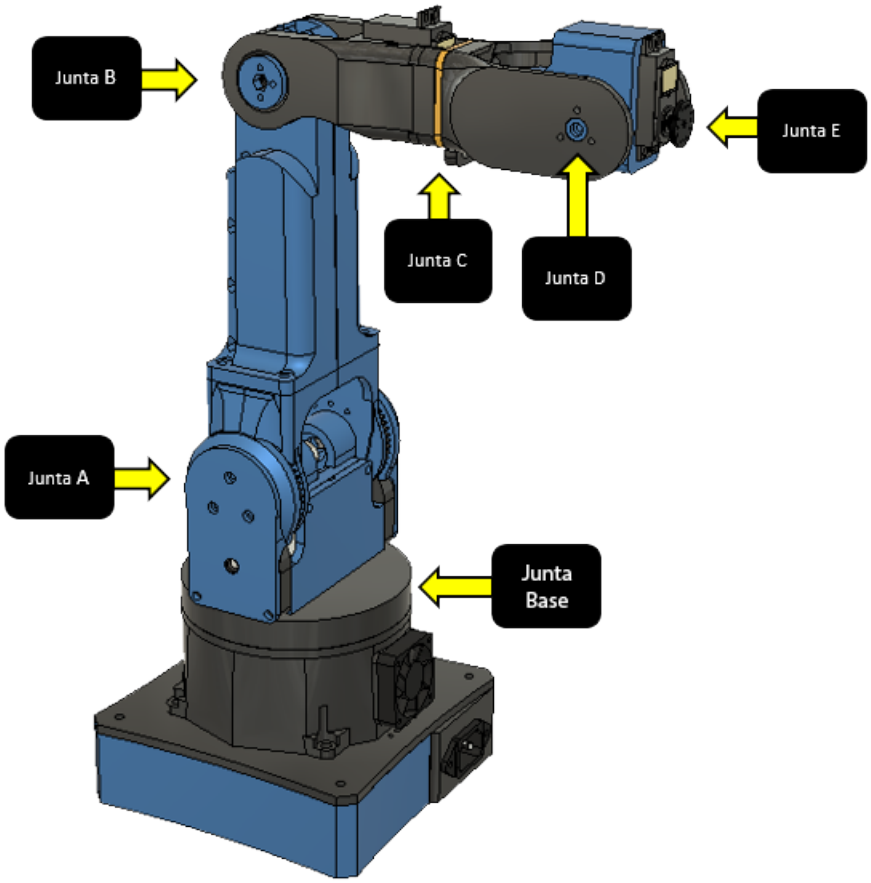


Figura 2 – Nomenclatura convencionada para os eixos do robô.

Os motores de passo, apesar de proverem uma excelente controlabilidade, não possuem torque nativamente expressivo. Os NEMA 17 usados, de modelo 17HS3401, têm 28N.cm de torque estático mínimo segundo o datasheet (2..., s.d.) (correspondente a 25% do valor do MG995). Assim, sua anexação às juntas mais demandadas do robô é feita por intermédio de caixas de redução 25:1 próprias para o sistema, conforme ilustrado em mais detalhes na figura 3.

O movimento do motor B para sua junta ocorre por intermédio de um sistema de

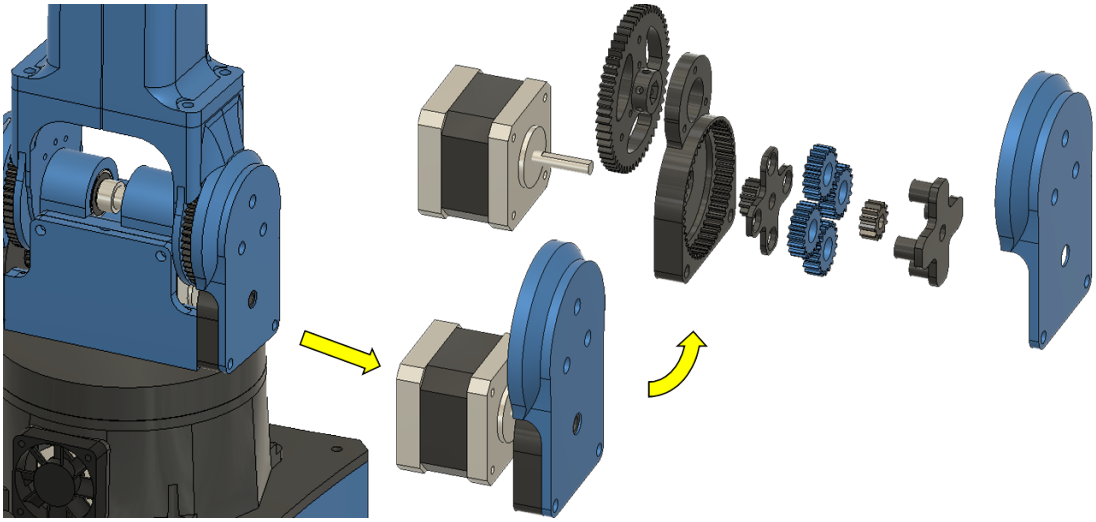


Figura 3 – Redução 25:1 originalmente adotada para as juntas A e B.

transmissão, pois a preocupação com a distribuição de peso do robô levou à decisão de mover a unidade motriz para mais próxima da base. Com isto, para compor o sistema são usadas peças disponíveis comercialmente: uma correia síncrona, duas polias de alumínio passo GT2 e dois eixos de aço carbono com perfil hexagonal - cortados sob medida a partir de uma chave Allen 8mm. Além disso, rolamentos 6082RS - de diâmetro interno perfeito para passagem dos eixos em questão - asseguram um apoio sólido das partes na estrutura do robô sem riscos de danificar as peças impressas vizinhas. O arranjo descrito está ilustrado na figura 4.

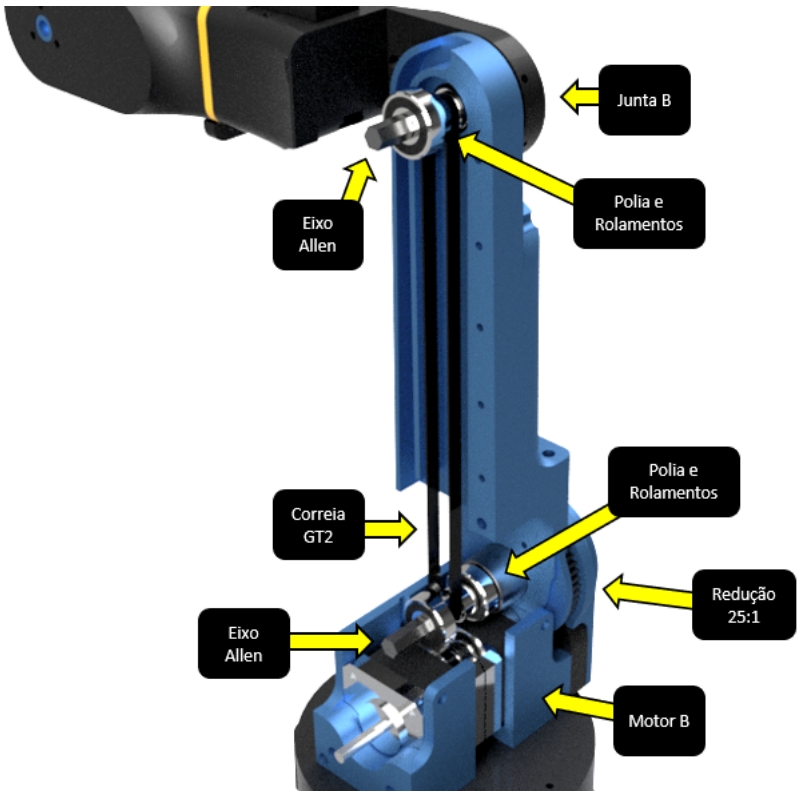


Figura 4 – Transmisssão de movimento do motor B para a respectiva junta.

Um problema reportado para este layout de transmissão, no entanto, é que, devido à falta de polias de apoio para tensionar adequadamente a correia, há a ocorrência de uma ampla zona morta na junta B (cerca de 30°). Assim traça-se outro ponto de mudança, mitigar este inconveniente através da inclusão de polias de apoio ou da reformulação completa do sistema de transmissão.

Por fim, o desenho geral do conjunto adota a simetria axial (observável na figura 5) por esta ser capaz de prover uma melhor distribuição de cargas na estrutura do que um variante assimétrica (a exemplo do ABB IRB 6700(ABB, s.d.[b])). A contrapartida óbvia desta opção é a limitação de deslocamento da junta B, semelhante à que ocorre em um braço humano quando tenta-se flexioná-lo. Considerando o tamanho reduzido do robô, tal situação gera uma perda considerável de seu volume de trabalho - visualizável na figura 6 - e portanto limita as possibilidades de filmagem que ele poderia fazer quando implementada a aplicação cinematográfica. Assim define-se um terceiro ponto de mudança: modificar o desenho do manipulador de modo a aumentar a amplitude da junta B.

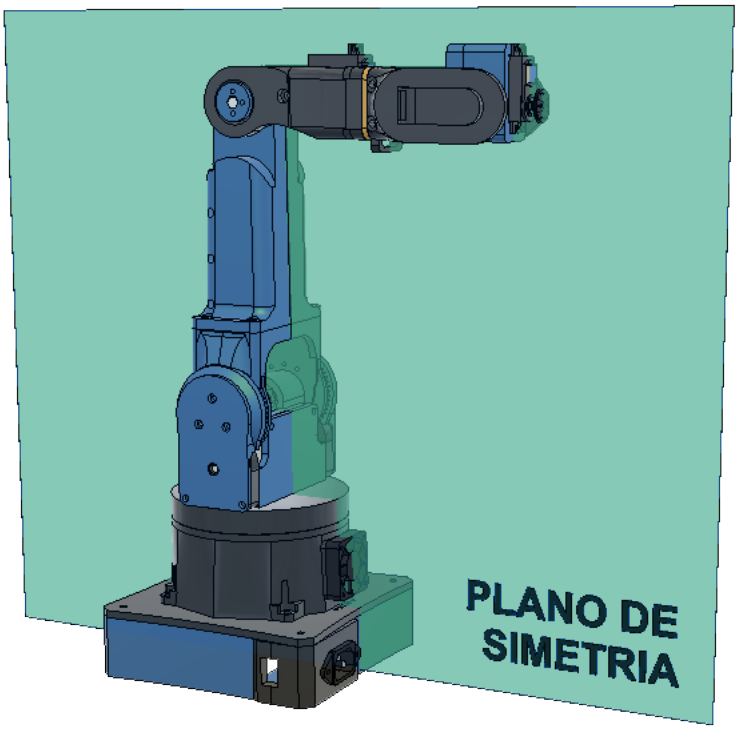


Figura 5 – Simetria axial adotada para o robô.

3.2 Elétrica e Eletrônica

A central eletrônica do robô é composta basicamente de um microcontrolador, reguladores de tensão, conectores de entrada e saída e drivers para os motores de passo, todos montados em uma placa genérica perfurada de fenolite e conectados por fios cortados manualmente, conforme mostrado na figura 7. O microcontrolador utilizado é um Arduino Pro

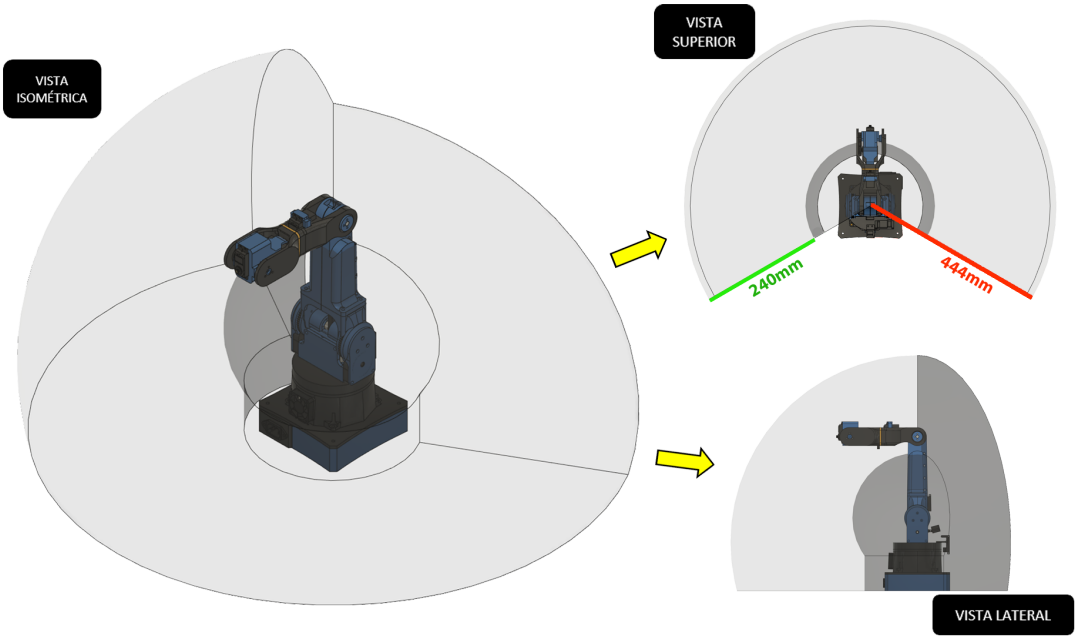


Figura 6 – Volume de trabalho do robô original.

Mini com um processador ATmega328 de arquitetura AVR 8 bits e clock 16MHz. A memória da placa também é centralizada nele: um conjunto Flash de 32KB para armazenamento do programa, SRAM de 1KB para uso durante a execução e uma EEPROM de 1KB para guardar dados não voláteis do usuário - por exemplo, sequências de posições gravadas do robô. Não há suporte nativo a armazenamento externo (ex.: cartões de memória), o que de início configura mais um ponto de mudança a ser implementada - já que a EEPROM do Arduino não favorece o armazenamento de dados no formato de arquivos, e somente 1KB de memória não-volátil facilmente pode ficar insuficiente para sequências de posições muito complexas ou longas.

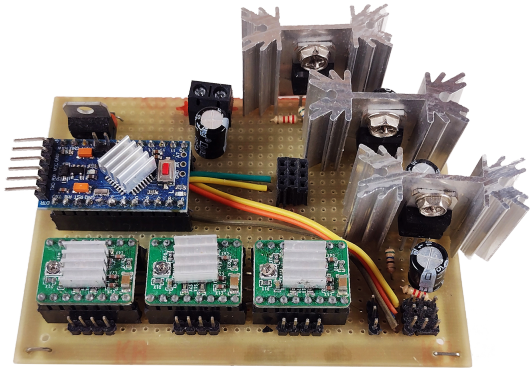


Figura 7 – Placa de circuito original do robô.

Para cada motor de passo do robô há um drivers que o controla de acordo com o comando do Arduino. O uso de drivers é necessário porque uma ligação direta entre o Arduino e estes motores é inviável por variados motivos, desde a disponibilidade de portas digitais do Arduino - cada motor de passo bipolar implica em 4 fios - até o nível de corrente

que o microcontrolador pode suprir - segundo o datasheet(2..., s.d.), cada motor de passo exige até 1.3A por bobina, contra o máximo fornecível de 40mA por pino do ATmega328 (8-BIT..., s.d.). Com o uso destes drivers, toda a interação do Arduino com cada motor fica reduzida a somente dois fios: um para definir o sentido de giro e outro para efetivamente controlar a rotação. Os drivers são do modelo Allegro A4988, que além de propiciarem a interface mencionada ainda conseguem dividir cada passo cheio do motor - de 1.8° - em até 16 micropassos - diminuindo o ângulo para 0.1125° e permitindo a geração de movimentos mais suaves e precisos (A4988..., s.d.). A figura 8 ilustra este modelo de driver e suas conexões.

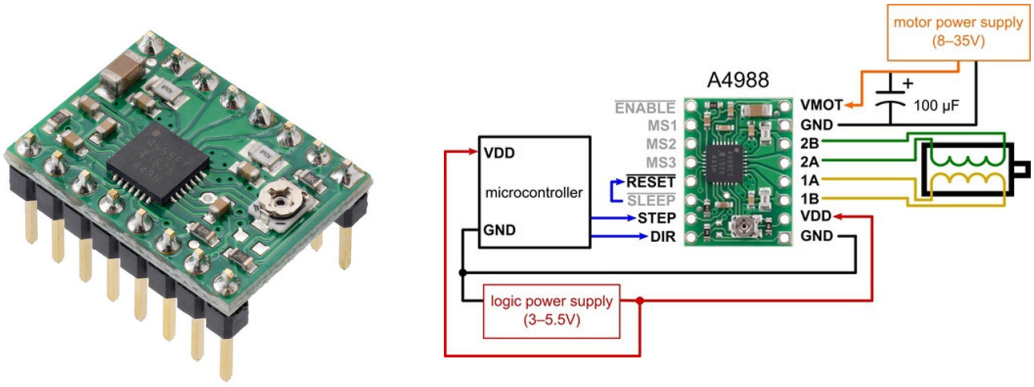


Figura 8 – Driver Allegro A4988.(POLOLU, s.d.[a])

.

Quanto à parte elétrica, Há os seguintes requisitos de corrente e tensão para cada componente: Os drivers Allegro A4988 exigem alimentação superior a 8V para energizarem a si e aos motores de passo (A4988..., s.d.), enquanto os servomotores exigem alimentação entre 4.8V e 7.2V (TOWERPRO, s.d.) e o Arduino Pro Mini exige 5V (8-BIT..., s.d.). O elemento mais demandante em termos de corrente é o motor de passo - com valor de até 1.3A no pior caso (2..., s.d.) - seguidos de 1.2A para cada servomotor (TOWERPRO, s.d.). O Arduino não fornece valores oficiais, mas testes realizados com o regulador L7805 mostraram que ele nunca chegou a exigir mais de 800mA. Assim, no absoluto pior caso teórico do funcionamento do robô a fonte precisa fornecer a soma de 8.3A. Assumindo no entanto que dificilmente este pior caso ocorreria, considerou-se que a metade deste valor - cerca de 4.15A - poderia ser suficiente para o cenário rotineiro.

Com isto, a placa é energizada com uma fonte chaveada 12V 5A e adota reguladores de tensão específicos para cada componente. Os drivers Allegro podem receber os 12V diretamente e assim está feito, mas para o Arduino, por exemplo, um regulador LM7805 - fixo para 5V - intermedia a energização seguindo o esquemático ilustrado na figura 9. Para cada um dos três servomotores são adotados LM317 com resistências ajustadas para suprir 6V e 1.5A, conforme figura 10 (em ambos os esquemáticos há capacitores presentes para estabilizar oscilações da fonte). O robô foi testado em diversas situações por 6 meses e não

foram encontrados problemas na alimentação da parte elétrica.

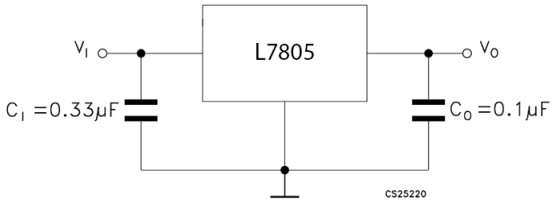


Figura 9 – Circuito esquemático do regulador L7805.

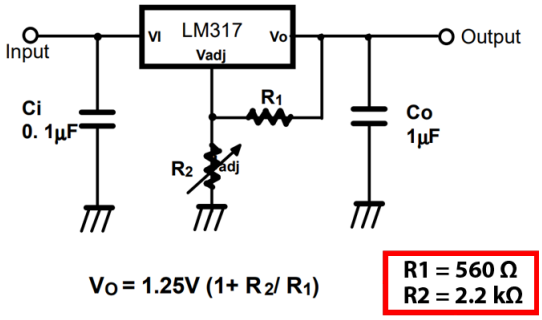


Figura 10 – Circuito esquemático do regulador LM317 para uma saída de 6V.

3.3 Interface com Usuário

O robô recebe comandos do usuário por meio dos pinos RS232 TTL do Arduino Pro Mini. Para lidar com este padrão de comunicação usa-se um módulo externo CP2102, que converte a codificação RS232 para USB UART e assim permite a troca de informações diretamente com computadores e celulares munidos de adaptadores OTG. Uma vez conectado este módulo ao Arduino por meio de um cabo de jumpers genérico - arranjo ilustrado na figura 11 -, basta ligar o módulo CP2102 ao computador ou celular, abrir qualquer aplicação que permita a troca de mensagens UART - tais como o monitor Serial do Arduino IDE - e iniciar o envio de comandos de texto ao robô no formato definido nos parágrafos que seguem. A figura 13 ilustra a interface.

O principal comando do robô consiste em mandá-lo mover uma das juntas individualmente, por ou para um determinado ângulo, a depender do referencial adotado, e em determinada velocidade. O formato do comando é similar a "Ba +030 500", onde:

- o primeiro caractere é o eixo a mover (base (b), A, B, C, D, E);
- o segundo é o referencial, se coordenadas absolutas (a) ou incrementais (i);

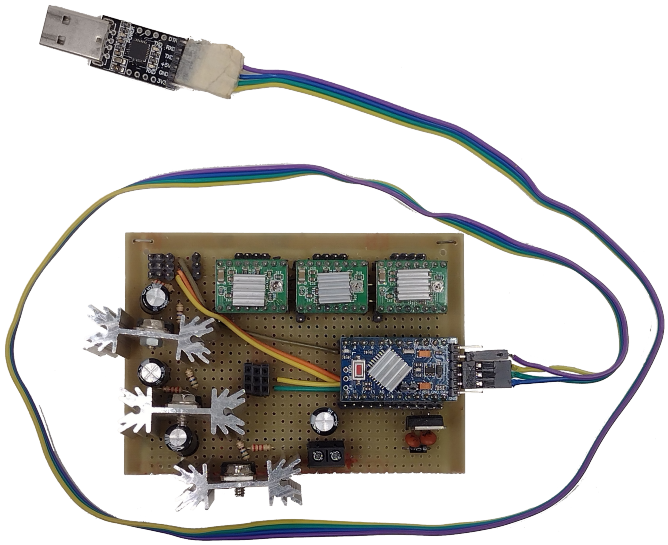


Figura 11 – Módulo CP2102 conectado ao Arduino com cabo de jumpers.

- o terceiro é a direção do movimento, se horário (+) ou anti-horário (-);
- o quarto é o módulo do deslocamento incremental ou a posição absoluta, a depeder do modo selecionado, em graus; e
- o quinto é a velocidade de movimentação (parâmetro adimensional).

Por exemplo, "Ai +020 100" move a junta A em coordenadas incrementais, 20 graus horários na velocidade 100. "Da -120 050" por sua vez move a junta D para a posição absoluta -120° na velocidade 50. Uma demonstração do funcionamento pode ser conferido na primeira metade do vídeo neste **link**.

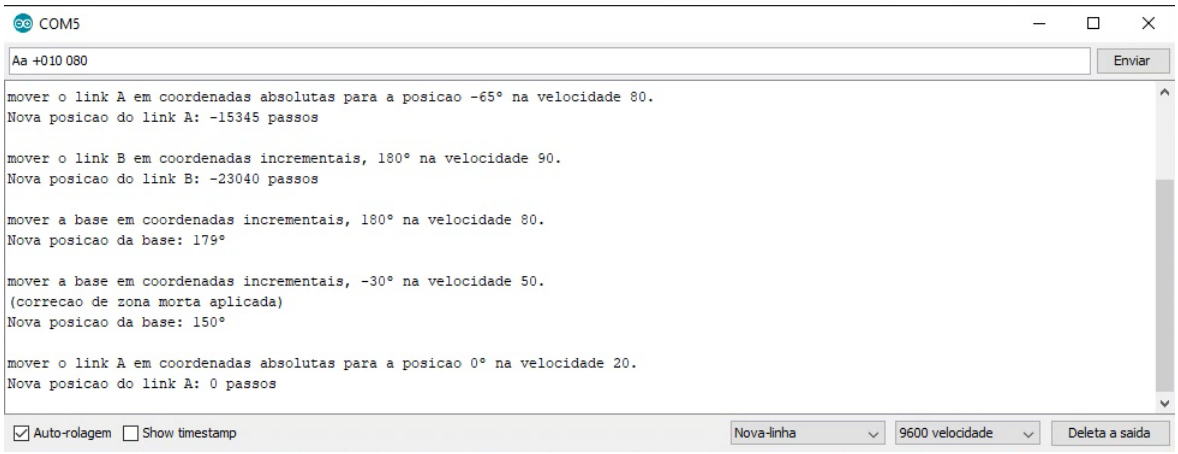


Figura 12 – Interface original de comunicação com o robô.

Além deste comando há outros voltados para o gerenciamento das funções do robô. Uma vez atingida alguma posição que o usuário queira guardar, o comando ’pos’ inclui esta em uma fila com espaço para até 20 posições. A partir de duas gravadas, o comando ’auto’ faz o robô mover-se sequencialmente por elas, na ordem que foram inseridas, em

um movimento de "ligar os pontos" onde todas as juntas têm intervalos calculados para começarem e terminarem o movimento no mesmo instante de tempo. Se o usuário desejar adicionar uma posição no meio da fila já existente, pode usar o comando 'pXX', onde XX é um número de 00 a 19 (se já houver uma previamente gravada ali, o usuário é antes de mais nada questionado se deseja substituir ou simplesmente incrementar todas as seguintes para abrir espaço).

O comando 'fila' informa as posições até então registradas na sequência e o comando 'remover pXX' retira o elemento XX específico dela. Por fim, se o usuário quiser guardar a fila em caráter permanente, pode entrar com 'gravar' que toda a sequência é armazenada na EEPROM de 1kB do Arduino, fazendo assim com que a fila permaneça resgatável mesmo que o robô seja desligado. De forma complementar, o comando 'carregar' traz a sequência armazenada nesta memória caso ela exista. É exatamente pela limitação Devido ao tamanho da EEPROM a fila é restrita a 20 posições: cada posição gravada gasta cerca de 27 bytes, totalizando 540 bytes para 20 espaços, e o restante ficou reservado para implementações futuras. Para cada comando que o usuário insere o robô fornece uma resposta de acordo.

Notavelmente, a interface apresentada não se mostra muito interessante para usuários com pouco conhecimento em eletrônica e programação, mesmo que perfeitamente funcional. A estética pobre da comunicação por linha de comando, a necessidade de depender de um software terceiro pouco utilizado pelo público geral com monitor serial para envio de comandos de texto e o uso obrigatório de uma conexão cabeada genérica entre o Arduino e o dispositivo do usuário (notebook, celular, etc) podem tornar a experiência de uso trabalhosa, maçante e, inclusive, desencadear desinteresse no uso da aplicação. Sendo assim, define-se mais um ponto de mudança - o desenvolvimento de uma nova interface, de acesso mais simplificado e mais agradável ao uso.

Um detalhe importante de se observar é que não há comandos relacionados ao movimento por coordenadas cartesianas - toda implementação segue a lógica de movimentação por ângulos das juntas. Isto ocorre porque a cinemática inversa - fundamento matemático que correlaciona a posição angular do robô ao sistema XYZ - ainda não está implementada. Não é possível fazê-lo no Arduino Pro Mini devido às suas limitações de memória e processamento. Com isto define-se mais um ponto de melhoria, este condicionado à troca do microcontrolador por um modelo de configuração superior.

3.4 Código-fonte e Algoritmo

Como o elemento de controle da placa é um Arduino, todo o código-fonte responsável por processar os comandos do usuário, gerenciar o movimento dos motores e ler/escrever dados na EEPROM foi desenvolvido na linguagem Arduino - amplamente baseada em C/C++ - e implementado no ambiente de desenvolvimento integrado do mesmo grupo - o

Arduino IDE. Na maioria dos casos de programação com esta ferramenta, o código-fonte é o próprio firmware e não há suporte a sistema de arquivos ou uma função main executada por padrão. Toda a implementação em C/C++ é convertida em instruções de máquina AVR e gravada diretamente na memória flash do Pro Mini.

Na função cíclica principal, *void loop()*, o microcontrolador mantém o robô parado, aguardando comandos da interface com o usuário, e os executa no caso de receber algo válido. Para o movimento dos motores, o algoritmo implementado converte o deslocamento pedido para cada junta - de graus para passos de cada motor - e procura realizar uma distribuição homogênea, de modo que todos os eixos envolvidos comecem e parem de mover no mesmo instante de tempo. Eis um exemplo em pseudocódigo: juntas [A, B] devem ir de [15°, 30°] para [45°, 90°]. Se as constantes das juntas A e B forem respectivamente 368.06 e 393.06 passos/grau, teremos:

```
1      A.passos <- 368.06 * (45-15)
2      B.passos <- 393.06 * (90-30)
3
4      passoTotais <- A.passos + B.passos
5
6      A.intervalo <- passosTotais / A.passos
7      B.intervalo <- passosTotais / B.passos;
```

Este cálculo tem a vantagem de tomar como referencial apenas a própria quantidade de passos que os eixos somados vão dar, em vez de depender de uma constante arbitrária que possa se provar grande demais em deslocamentos curtos ou vice-versa. A partir daí inicia-se o processo de escolha de qual eixo dar o próximo passo com base em valores *acumulados* de movimentação de cada um, seguindo lógica similar ao seguinte pseudocódigo:

```
1
2      A.acumulado <- A.intervalo
3      B.acumulado <- B.intervalo
4
5      enquanto passosTotais > 0:
6       escolhido <- menor(A.acumulado, B.acumulado)
7       escolhido.darUmPasso()
8       escolhido.acumulado <- escolhido.acumulado+escolhido.intervalo
9       passosTotais <- passosTotais - 1
```

Este algoritmo mostra-se bastante eficaz em promover um movimento simultâneo e preciso dos eixos, fazendo com que não só eles comecem e terminem de mover ao mesmo tempo como também apresentem erro pequeno de deslocamento em relação ao esperado - neste robô, menos de 1° em todos os eixos. Em nenhum momento no algoritmo a velocidade das juntas foi levada em consideração, entretanto (as variáveis "intervalo" na verdade eram

usadas apenas para escolher o próximo eixo a dar um passo - ou seja, aquele com o menor "intervalo acumulado"). Para o cenário de um deslocamento simultâneo de várias juntas, incluir intervalos reais entre os passos para conferir diferentes velocidades ao movimento levaria à modificação do pseudocódigo acima para o seguinte formato:

```
1
2       A.acumulado = A.intervalo
3       B.acumulado = B.intervalo
4       ultimoInter = 0
5
6       enquanto passosTotais > 0:
7           escolhido <- menor(A.acumulado, B.acumulado)
8           escolhido.darUmPasso()
9           escolhido.acumulado <- escolhido.intervalo
10          passosTotais <- passosTotais - 1
11
12          inter <- escolhido.acumulado - ultimoInter
13          ultimoInter <- inter
14          intervaloDeFato <- int(inter)
15          delay(intervaloDeFato)
```

O problema intrínseco desta abordagem é que o número de instruções dentro do loop dobra, pulando de 4 para 8 linhas, e com apenas duas delas realmente orientadas à execução do movimento. É o equivalente na teoria da manufatura enxuta aos tempos de ciclo, produtivo de fato, e setup, improdutivo porém necessário. Na computação, o setup é também chamado de *overhead*. Considerando que um montante de 50000 passos totais não seria uma estimativa irrealista em um movimento rotineiro, o *overhead* computacional fica não-negligenciável e, com a velocidade do robô provida pelo Arduino Pro Mini no primeiro algoritmo (observável na demonstração da seção 3.5 a seguir), incluir o parâmetro de velocidade tornaria o movimento lento. Caso um microcontrolador de maior clock de fato viesse a substituir o Pro Mini em implementações futuras, o problema do *overhead* poderia se tornar menos relevante e o controle de velocidade poderia vir a ser testado.

Para movimentos de eixos individuais, no entanto, a preocupação exibida acima não é relevante porque não há necessidade de calcular dinamicamente o próximo intervalo ou eixo a ser movido. Assim, no código do robô foi incluída uma função específica para movimentos de apenas um eixo, com algortimo dado a seguir:

```
1       escolhido.intervalo = constanteDeTempo / passosTotais
2
3       enquanto passosTotais > 0 :
4           escolhido.darUmPasso()
5           delay(escolhido.intervalo)
6           passosTotais <- passos totais - 1
```

3.5 Demonstração de Filmagem

O robô exatamente como descrito nas seções acima foi testado para o uso cinematográfico em abril de 2020, onde apesar de conseguir executar a aplicação, não era adaptado para tal e possuía limitações em sua execução. O teste, que foi filmado e encontra-se disponível neste **link**, permite observar bem como funcionava a velocidade e a movimentação do robô por entre os vários pontos inclusos na fila dele. Um "making of" do processo de gravação destas posições também está disponível neste **link**. Neste último a imagem está acelerada em uma proporção de 20:1, o que indica que o ato de preparar o movimento (*tempo de setup*) levou cerca de 18 minutos quando feito pelo operador, este já familiarizado com os comandos da interface.

Figura 13 – Primeira gravação de vídeo com o robô.

Observa-se também no primeiro vídeo que a qualidade da filmagem robotizada não está satisfatória, apresentando muita vibração e descentralização do objeto sendo filmado. Enquanto este segundo item pode ser facilmente relacionado à falta de uma cinemática inversa para um melhor planejamento de trajetória, a causa do primeiro é principalmente, mas não apenas, relacionada com a dinâmica de atuação dos servomotores, que possuem resolução considerada muito grosseira para filmagens: o deslocamento mínimo de 1 grau é insuficiente para proporcionar a suavidade exigida na movimentação do robô. Simplesmente incluir reduções que melhorem este ponto irá necessariamente restringir o deslocamento dos eixos. Em face ao problema, inclui-se mais um ponto de mudança: a substituição dos servomotores por alternativas capazes de atenderem melhor a aplicação cinematográfica.

3.6 Resumo dos Pontos de Melhoria

Ao longo das seções anteriores foram explorados detalhes do robô escolhido como ponto de partida para este Trabalho de Graduação, bem como melhorias a serem realizadas no mesmo para atingir satisfatoriamente a aplicação cinematográfica. Eis um resumo dos pontos abordados:

M1. Substituir todas as peças em PLA por PETG;

M2. Diminuir zona morta na transmissão da junta B;

M3. Aumentar o deslocamento possível da junta B;

M4. Incluir uma unidade externa de armazenamento não-volátil;

M5. Desenvolver uma nova interface, mais amigável ao usuário inexperiente;

M6. Desenvolver a cinemática inversa para controle cartesiano; e

M7. Substituir todos os servomotores por alternativas mais adequadas para a filmagem robotizada.

4 Melhorias na Eletrônica e no Código-Fonte

Com os pontos de melhoria já definidos a partir das características prévias do robô escolhido como base, chega o momento de planejar as ações para resolvê-los. A abordagem escolhida em relação a este processo é aproveitar a implementação das mudanças para também melhorar o nível geral do projeto.

Um bom exemplo para ilustrar esta postura está na solução aos itens M4, M5, M6 e M7 da seção 3.6. Todos estes envolvem de alguma forma a modificação da placa de circuito do robô - vista na figura 7, mas nenhum deles poderia ser resolvido com uma simples solda de novos terminais ou fios cortados na mesma placa. Por exemplo, a inclusão de uma unidade de armazenamento (item M4) demandaria um remapeamento das conexões existentes para liberar os pinos SPI do microcontrolador (utilizados em comunicação com cartões SD). Já a troca dos servomotores (item M7) possivelmente necessitaria de mais pinos para o comando de cada novo motor, o que somado a uma nova interface simplificada através de cabos USB comuns (item M5) , à necessidade de maior poder de processamento implícito em uma cinemática inversa embarcada (item M6) e a um controle de velocidade para interpolações indica a necessidade de troca do microcontrolador por um modelo mais potente.

A troca do microcontrolador implica em trocar a placa de circuito perfurada por uma nova, ligeiramente maior e capaz de abarcar tanto a nova unidade de controle como os demais componentes: por exemplo, as conexões para os novos motores e para a unidade de armazenamento externo. Neste contexto, optou-se por desenvolver e encomendar com um fabricante especializado uma placa de circuito própria para atender aos novos requisitos, e assim, todo o processo de montagem dos componentes na nova placa fica bastante simplificado.

4.1 Escolha dos Novos Componentes

Para o desenvolvimento da nova placa faz-se necessário primeiro proceder com a escolha dos componentes eletrônicos a serem adicionados e alterados em relação à original.

O primeiro e mais importante deles é o microcontrolador. Feitas as devidas consideraçoes de necessidade de pinos de entrada e saída, recursos, custo da placa e tamanho em relação ao Arduino, optou-se por substituir o Pro Mini existente pelo Raspberry Pi Pico (RASPBERRY..., s.d.), observável na figura 14.

Os detalhes mais relevantes são 5 pinos a mais de entrada e saída (26 ante 21),

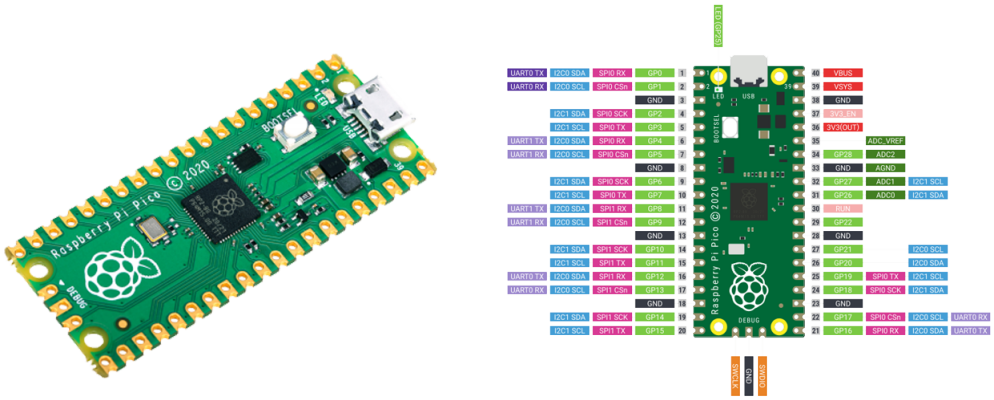


Figura 14 – Raspberry Pi Pico.(RASPBERRY…, s.d.)

arquitetura de 32 bits em vez de 8, memórias Flash e SRAM amplamente maiores, uma porta microUSB nativa para o uso de cabos USB comerciais e um recurso exclusivo de acionamento das saídas digitais chamado *Programmable Input/Output* ou *PIO*. Com ele é possível associar máquinas de estado configuráveis em hardware aos pinos de saída, podendo a partir daí acioná-los de maneira periódica sem envolvimento direto do processador (embora esta funcionalidade ainda careça de documentação no momento de escrita deste relatório, pode ser extremamente útil para diminuir o *overhead* do algoritmo de movimentação). Como mostrado na figura 15, em basicamente tudo menos no tamanho a unidade Raspberry se mostra uma opção melhor frente ao Pro Mini.

	Arduino Pro Mini	Raspberry Pi Pico
Processador	ATmega328P Single-core AVR, 8 bits	RP2040 Dual-core ARM M0+, 32 bits
Vel. Clock	16MHz	125MHz
Nível Lógico	5V	3.3V
Protocolos suportados	UART (x1), SPI (x1), I2C (x1), PWM (x6)	UART (x2), SPI (x2) I2C (x2), PWM (x16)
GPIO	20 pinos	26 pinos
Memória Flash	32kB	2MB
SRAM	2kB	264kB
EEPROM	1kB	Não
Saída USB	Não	Sim
Tamanho	19mm x 33mm	21mm x 51mm

Figura 15 – Comparativo dos principais aspectos técnicos dos microcontroladores.

Com a escolha do microcontrolador feita, é a vez dos motores. Considerando a necessidade explicitada na seção 3.5 de trocar os servomotores por alternativas capazes de proporcionar movimentos mais suaves e melhor atender ao propósito cinematográfico, foi decidido abandonar a configuração mista presente no Projeto Pégaso original e usar apenas motores de passo em todas as juntas. Há um número de fatores que justificam esta decisão, sendo os principais:

- Os motores de passo podem contar com os drivers abordados na seção 3.3, que exigem somente um pino a mais no controle que os servomotores;
- A unificação do tipo de motor para o projeto facilita o processo de obtenção dos materiais para a construção do mesmo, bem como simplifica a programação das rotinas de movimentação ao fazerem todos os eixos seguirem o mesmo algoritmo; e
- A unificação também resulta em peças similares ao longo das juntas do robô, o que simplifica o processo de design e diminui a variedade de peças necessárias para a construção do manipulador.

A decisão de manter apenas motores de passo no robô implica no acréscimo de três drivers Allegro A4988 ou similares na lista de componentes da futura placa, ao mesmo tempo que desobriga a necessidade dos circuitos de energização dos servomotores baseados no regulador LM317 (descritos na seção 3.3). Apesar da remoção destes circuitos indicar uma simplificação bem-vinda na placa, optou-se por mantê-los como forma de poder reverter para os servomotores em caso de problemas na implementação ou até usá-los no efetuador terminal em aplicações futuras. Com esta alteração a nova placa conta com suporte a seis motores de passo e até três servomotores, totalizando até 9 graus de liberdade possíveis.

Como os MG-995 passaram a ser opcionais, os circuitos com reguladores LM317 têm uma chave geral acrescida à entrada de tensão na placa para permitir desligá-los caso não estejam efetivamente em uso. Além disso, os servomotores passam a ser obrigatoriamente controláveis apenas através de um módulo externo PCA9685(16-CHANNEL..., s.d.), específico para gerar sinais PWM compatíveis com estes motores, e que por sua vez recebe comandos do Raspberry Pico via protocolo I2C. Como o I2C permite o compartilhamento de sinal com diversos outros módulos, não há reserva permanente de portas digitais do microcontrolador para atender ao uso eventual destes motores.

Quanto ao armazenamento externo, foi decidido usar um cartão SD e assim reservar os pinos digitais 12 a 15 do Raspberry Pi Pico para a comunicação, já que eles têm a funcionalidade do protocolo SPI integrada e assim permitem uma comunicação dedicada com este dispositivo (RASPBERRY..., s.d.).

Por fim, resta observar o ganho obtido pelo conjunto de pinos a mais que o novo microcontrolador apresenta sobre o antigo. Conforme ilustrado na figura 8, os drivers A4988 exigem 6 pinos de controle para uma funcionalidade completa:

- um pino DIR para definir o sentido do giro do motor;
- um pino STEP para dar um passo no sentido definido;
- um pino ENABLE para habilitar o motor; e

- três pinos MS1, MS2 e MS3 para definir o deslocamento por passo do motor, que pode variar de 0.1125° a 1.8° por passo a depender da combinação.

Porém, apenas dois destes exigem exclusividade para cada motor empregado (DIR e STEP). Os bits de resolução (MS1, MS2 e MS3) e o de habilitar motor (ENABLE) podem ser compartilhados entre todos os eixos ao custo de não ser possível configurar deslocamentos específicos para cada um ou ainda habilitar somente uma parte dos motores durante um movimento. No entanto, esta restrição não constitui um problema significativo face à economia de pinos digitais exigidos para o controle dos motores, que passam de 36 em um sistema completamente configurável (todos os 6 pinos para cada um dos eixos) a meros 16 (2 por eixo + 4 para o controle de resolução e habilitação dos motores). Sendo assim, considerando que:

- os motores ocupam 16 pinos digitais;
- a inclusão do armazenamento externo ocupa mais 4 pinos para o protocolo SPI; e
- o controle opcional dos servomotores se dá pelo padrão I2C - portanto ocupando mais dois pinos do microcontrolador;

É perceptível que este novo arranjo de elementos exige 22 conexões. Com o Arduino Pro Mini e seu conjunto de 21 pinos de propósito geral, claramente é impossível implementar esta mudança. Com o Raspberry Pi Pico, no entanto, há 26 pinos e a consequente sobra de 4 deles para recursos adicionais.

4.1.1 Acréscimo da Capacidade Wi-Fi

Com nenhum dos elementos periféricos citados acima exigindo o uso da comunicação UART, a disponibilidade de quatro pinos extras para a inclusão de possíveis novos elementos e o novo microcontrolador funcionando em nível lógico 3.3V (comum a periféricos para comunicação sem fio) viu-se a oportunidade de incluir a funcionalidade Wi-Fi ao projeto através da anexação de um módulo ESP-01s aos componentes da placa. Com ela o robô pode passar a receber comandos sem a necessidade de cabos físicos e diretamente da rede, o que favorece o desenvolvimento de uma interface mais amigável conforme prevista no item M4 da seção 3.6. O ESP-01s - ilustrado na figura 16 - possui suporte à banda de frequência de 2.4GHz, um chip ESP8266 programável também de 32 bits e os exatos pinos RX/TX que compõem a comunicação UART. Exige portanto somente dois pinos do Raspberry Pico para a troca de dados, restando dois para recursos adicionais.

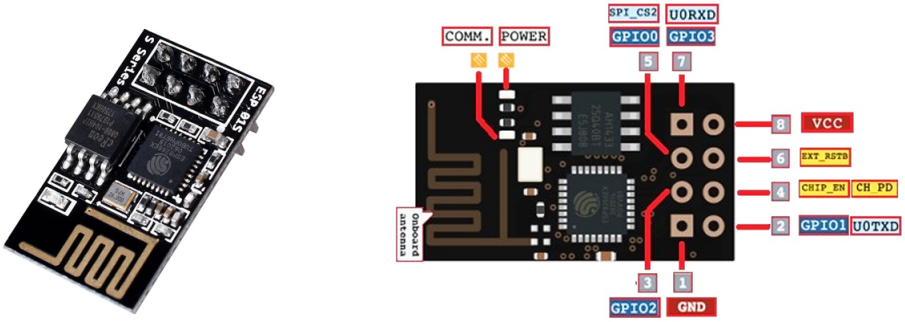


Figura 16 – Módulo Wi-Fi ESP-01s.(PROJECTS, s.d.)

4.1.2 Substituição dos Drivers A4988 por DRV8825

Uma outra melhoria realizada é a substituição dos drivers Allegro A4988 originais dos motores de passo por modelos TI DRV8825 (POLOLU, s.d.[b]). A razão da troca consiste em expandir as possibilidades de controle dos motores, com o DRV8825 permitindo dividir um passo cheio do motor em até 32 micropassos ante o máximo de 16 conferidos pela opção da Allegro. Esta diferença significa uma diminuição do mínimo deslocamento por passo de 0.1125° para 0.05625° no caso dos motores NEMA 17 escolhidos para o robô, o que contribui para movimentos ainda mais precisos em caso de necessidade. A figura 17 ilustra o novo componente. Apesar de esta ser uma mudança de um dos elementos principais discutidos anteriormente, nenhum impacto ocorreu em termos do projeto da placa justamente porque este novo dispositivo compartilha das exatas mesmas conexões externas que o A4988. Assim, não apenas toda a explicação relacionada ao antigo driver é válida para o novo como também é possível substituir um pelo outro em um mesmo lugar do circuito sem necessidade de modificar nenhum outro componente periférico.

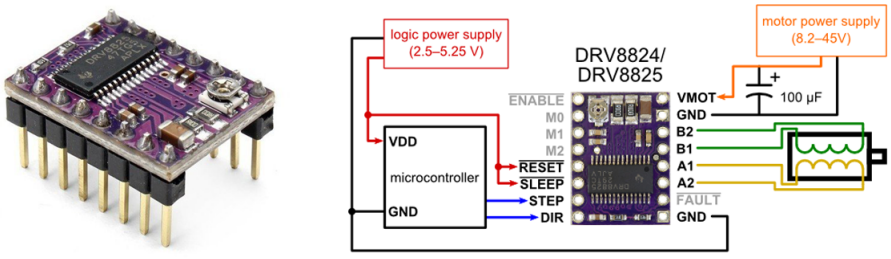


Figura 17 – Driver TI DRV8825.(POLOLU, s.d.[b])

4.1.3 Resumo de Conexões da Nova Placa

As conexões estabelecidas entre o novo microcontrolador e os periféricos definidos nas seções anteriores - drivers dos motores de passo, o cartão de memória, a conexão I2C e o módulo Wi-Fi ESP-01s - estão resumidas na figura 18 para fácil referência.

Pino	Conectado a
GP0	DIR driver base
GP1	STEP driver base
GP2	DIR driver A
GP3	STEP driver A
GP4	DIR driver B
GP5	STEP driver B
GP6	DIR driver C
GP7	STEP driver C
GP8	DIR driver D
GP9	STEP driver D
GP10	STEP driver E
GP11	DIR driver E
GP12	Cartão SD - SPI MISO
GP13	Cartão SD - SPI CS
GP14	Cartão SD - SPI SCK
GP15	Cartão SD - SPI MOSI
GP16 (TX)	ESP-01 - UART RX
GP17 (RX)	ESP01 - UART TX
GP18	M2 todos os drivers
GP19	ENABLE todos os drivers
GP20	Livre
GP21	Livre
GP22	M1 todos os drivers
GP26	I2C SDA
GP27	I2C SCL
GP28	M0 todos os drivers

Figura 18 – Conexões do Raspberry Pi Pico com os elementos periféricos da placa.

4.2 A PegasoBoard v1.0

A nova central eletrônica do robô - batizada de PegasoBoard v1.0 - foi projetada no software Autodesk Eagle PCB e produzida na China pela empresa JLCPCB. Uma imagem do projeto no Eagle pode ser conferida na figura 19, o resultado da fabricação pode ser visto na figura 20, a placa já com os componentes soldados pode ser observada na figura 21, e uma comparação visual direta entre o antes e o depois pode ser vista na figura 22.

4.3 Reformulação do código e interface

A troca do microcontrolador não apenas representa uma melhoria nas capacidades técnicas do robô como também permite que o código-fonte deste seja desenvolvido em Micropython ao invés de Arduino C/C++. Esta mudança não só confere a possibilidade de rescrever as instruções em uma linguagem mais amigável e popular como também permite contar com um ambiente REPL (*Read-Eval-Print-Loop*) e um sistema de arquivos para facilitar a organização, execução, *debugging* e armazenamento do código. Apesar de haver um custo computacional maior em rodar programas em Micropython por esta ser uma linguagem interpretada, dadas as aparentes vantagens foi tomada a decisão de reescrever

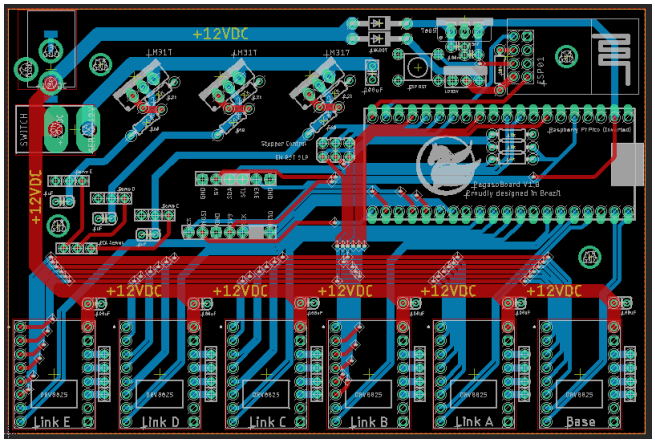


Figura 19 – Nova placa projetada no software Autodesk Eagle PCB.

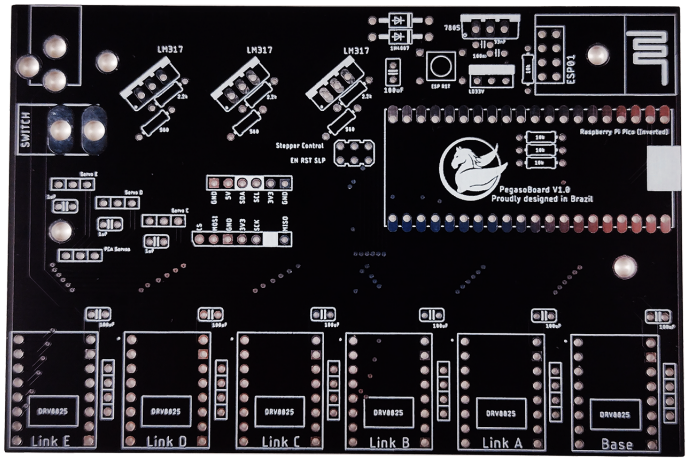


Figura 20 – Nova placa após a fabricação.

todo o código na nova linguagem. Quanto à interface, foi decidido utilizar a funcionalidade Wi-Fi recém-implementada para desenvolver um novo meio de interação com o robô, mais gráfico e dinâmico que o ambiente de linha de comando mas sem subtrair as funcionalidades presentes até então.

Após o planejamento do layout e a idealização do código foi concluída uma versão inicial da página, formatada para o controle do robô por eixo (junta). Ela pode ser conferida na figura 23. Uma vantagem imediata que esta solução apresenta sobre a anterior é que, enquanto a linha de comando depende de um software serial externo (como o Arduino IDE) e de uma conexão USB, o que acaba por favorecer o controle do robô por computador, com esta nova versão qualquer dispositivo conectado à mesma rede e com um navegador web integrado pode enviar comandos ao robô sem grandes dificuldades. De fato, após a instauração da página HTTP passou a ser possível controlá-lo de maneira potátil e remota, similar à mostrada na figura 24.

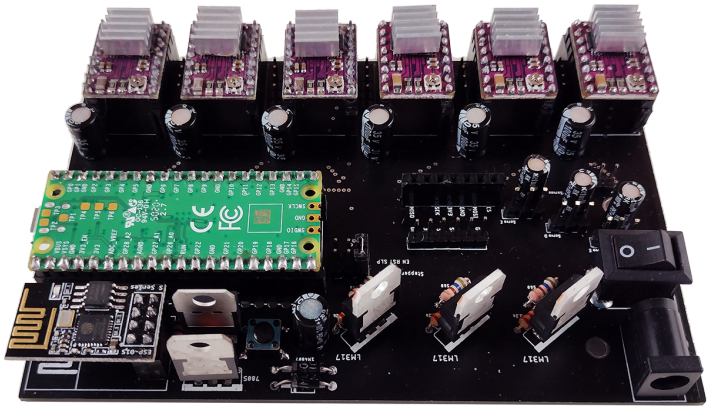


Figura 21 – Nova placa com componentes soldados e montados.

O usuário já consegue em poucos passos configurar um movimento para o robô:

1. Escolher no lado esquerdo da tela qual eixo deseja mover;
2. Configurar no canto superior direito a largura do deslocamento (em graus) e a velocidade (em graus/segundo);
3. Mover o eixo em questão no sentido desejado através do botões maiores "+" e "−";
4. Quando o robô chegar na posição que se deseje guardar na fila, pressionar o botão "ADD" ao centro. Novas posições vão sendo adicionadas sequencialmente;
5. Uma vez que se deseje observar a interpolação entre as posições guardadas na fila, pressionar o botão "play" no canto inferior esquerdo. Se for desejável que o movimento se repita três vezes consecutivas, acionar antes do "play" o botão "loop" ao lado;
6. Para gravar a sequência de posições no cartão de memória, apertar o botão "STORE". Para, por outro lado, carregar uma sequência já existente nele, apertar o botão "LOAD".

Há mais algumas funcionalidades embutidas, mas com este conjunto simples de etapas já é possível conferir ao braço robótico uma interpolação relativamente complexa. De modo a manter um nível de interação com o robô mais interessante, para cada comando enviado ao Pegaso uma resposta é gerada e fica aparente no display branco centralizado. Se o comando for válido ele confirma e se for inválido ele nega, também justificando o porquê da negativa: por exemplo, se o usuário tentar mover além dos limites do eixo o robô informa que é impossivel mover porque a posição final desejada está fora do alcance.

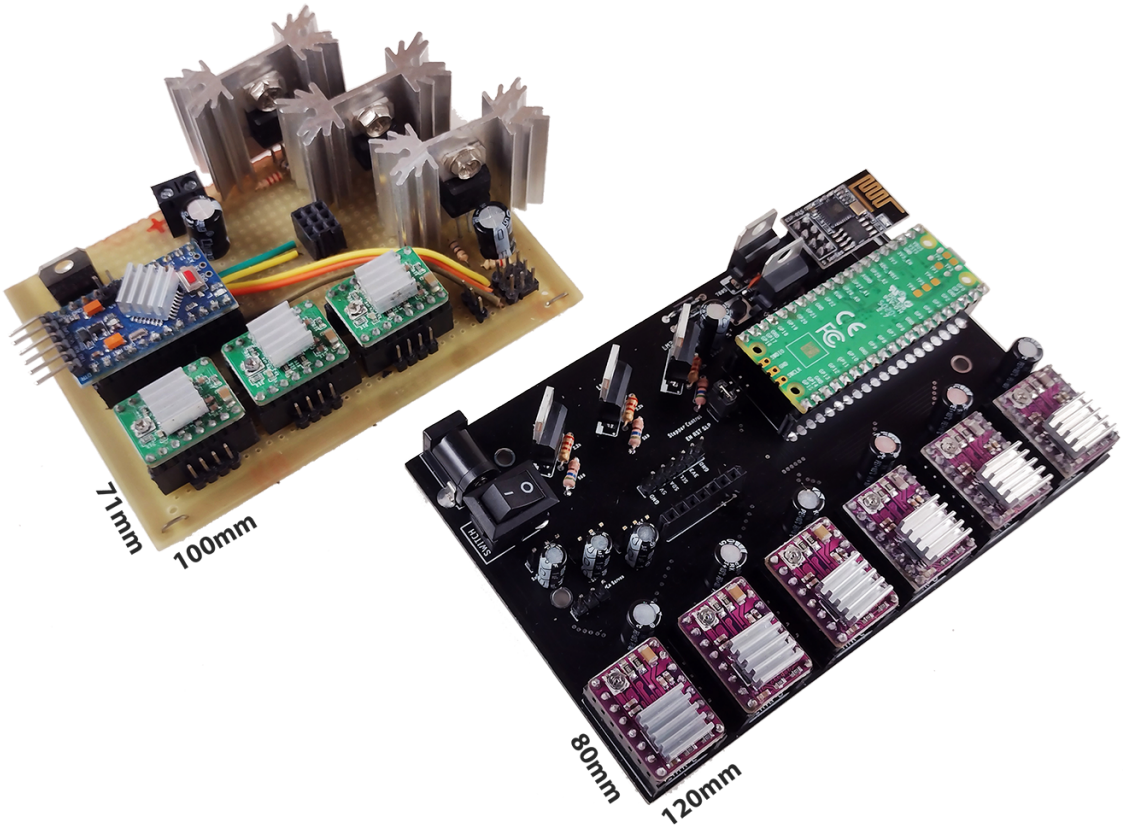


Figura 22 – comparativo visual entre as placas antiga e nova.

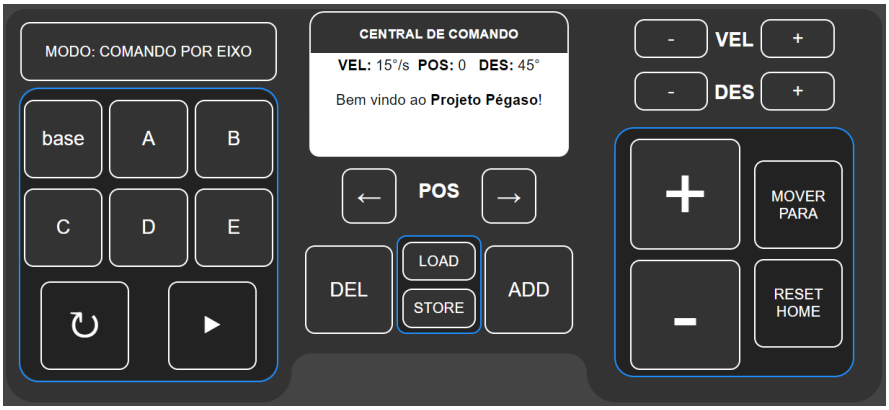


Figura 23 – Primeira interface gráfica do robô.

Figura 24 – Interface gráfica acessível via celular.

5 Melhorias Estruturais

Com a escolha dos novos componentes e o desenvolvimento da nova placa de circuito, considera-se a alteração da parte eletrônica concluída e migra-se o foco para a estrutura física do robô. Nesta seção serão abordadas as mudanças feitas no projeto das peças e encaixes com o intuito de atender aos pontos de melhoria traçados na seção 3.6, em particular os itens M2 e M3.

Em complemento a estes, três pontos extras relacionados à estrutura foram delineados para melhorar a confiabilidade das peças de impressão 3D em se tratando do desgaste por movimentação das juntas:

1. Todos os parafusos responsáveis por conexões estruturais deveriam ser acompanhados de porcas correspondentes (em vez de roscas criadas nas próprias peças impressas) para se manterem firmes;
2. Todos os pontos de movimentação das juntas devem contar com rolamentos para assegurar o contato com mínimo de atrito; e
3. Pontos críticos da sustentação das juntas deveriam ter arruelas associados aos parafusos como forma de distribuir melhor a pressão de aperto nas peças impressas e assim evitar deformações.

A demanda do terceiro item levou ao desenvolvimento de uma arruela exclusiva para o projeto (visível na figura 25), concebida no software AutoCAD e cortada a laser sob encomenda em uma chapa de aço inox 0.8mm. Sua idealização foi pensada para contemplar parafusos M3, os mais abundantes do projeto.

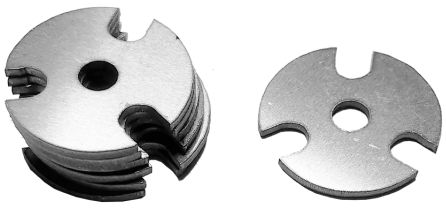


Figura 25 – Arruela de três pontos.

A ideia principal é que no robô há algumas conexões, especialmente as das juntas, que exigem mais de um parafuso para prover um encaixe livre de escorregamentos e do risco de afrouxamento do aperto com o tempo (comum quando se usa apenas um parafuso). Com

este modelo autoral até quatro parafusos podem estar associados a uma única arruela sem deixar de manter a distribuição de pressão adequada para o contexto do projeto.

Tendo os pontos citados acima como orientação para o desenvolvimento das novas peças, a primeira iteração do desenho poderia ter início.

5.1 Primeira Iteração

Com a decisão de trocar os servomotores MG-995 por motores de passo NEMA 17, é necessário redesenhar na totalidade as peças que compunham a estrutura da junta B em diante, já que elas eram adaptadas inteiramente para o conjunto de motores anterior. Além disso, devido ao fato das novas opções precisarem de uma caixa de redução para multiplicar o baixo torque dos eixos, conforme descrito na seção 3.1, fica claro que o desenho das novas peças deve contemplar a inclusão destas caixas para uma integração bem-sucedida dos NEMA 17. Optou-se então adotar uma redução do tipo planetária (ilustrada na figura 26), pois além de esta oferecer uma boa distribuição de cargas entre seus componentes e ser relativamente compacta frente a opções mais simples de mesmo nível redutivo, também se destaca por prover um alinhamento natural entre os eixos de entrada e saída(KHK, s.d.).

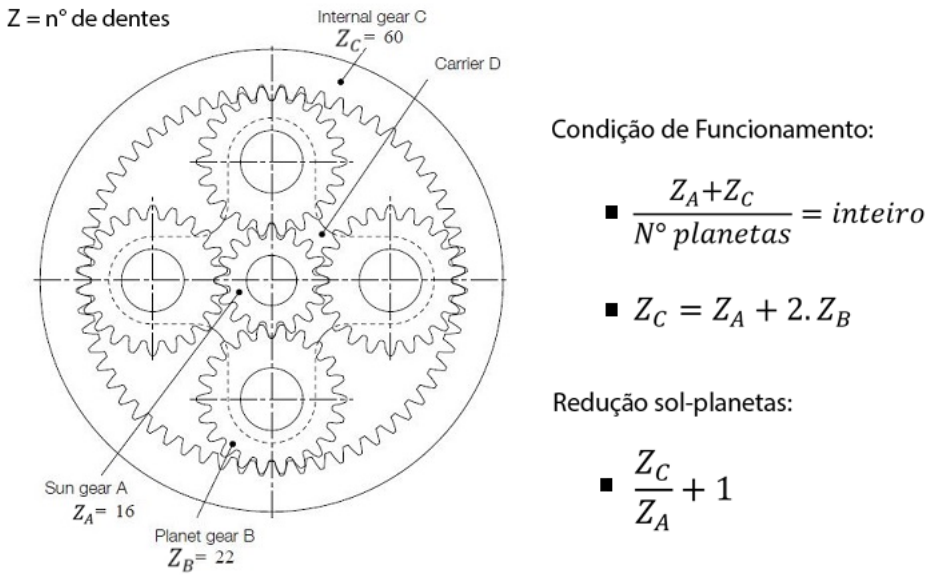


Figura 26 – Redução Planetária conforme modelagem teórica.(KHK, s.d.)

Esta última característica também permite que reduções planetárias possam ser acopladas umas nas outras, incrementando portanto o nível total da redução ao mesmo tempo que expande o tamanho físico do conjunto em apenas uma direção. Isto faz com que este tipo de associação de engrenagens seja amplamente modular, sendo facilmente aumentada ou diminuída conforme necessidade através da inserção ou remoção dos ditos níveis que a compõe. A modelagem da caixa de redução na tipologia planetária pode ser visualizada na figura 27. No caso, também acrescenta-se um rolamento 6004 2RS à última

etapa para prover adequadamente um contato entre o eixo de saída e a estrutura externa (anel) do conjunto.

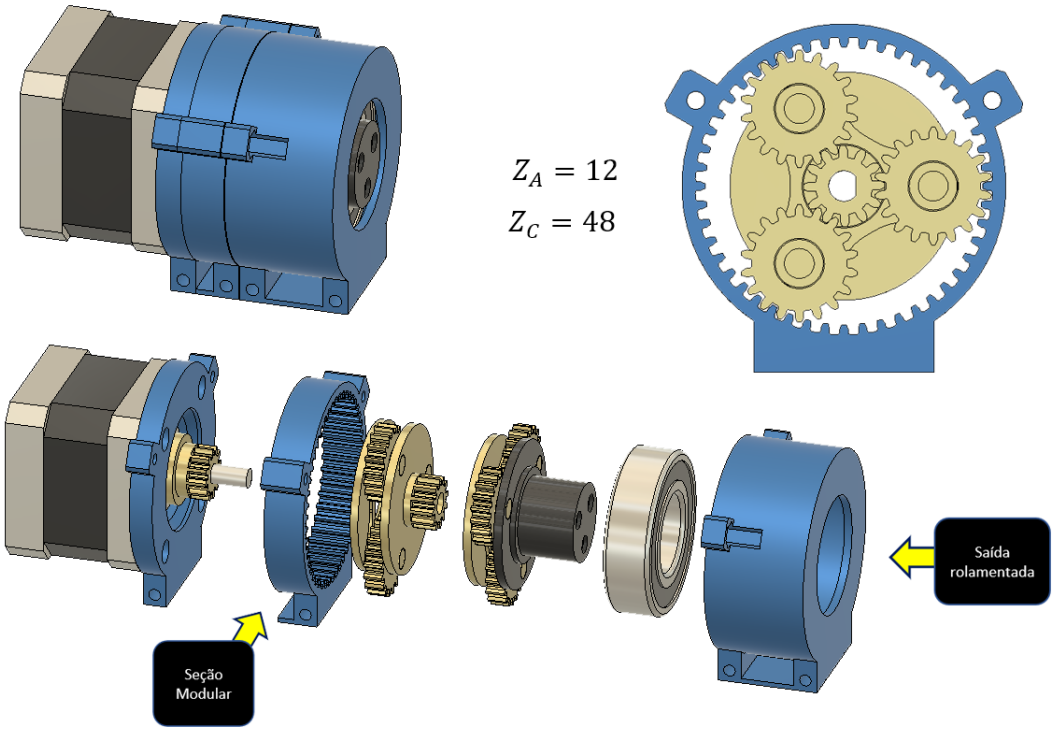


Figura 27 – Redução Planetária desenvolvida.

Um teste deste conjunto na velocidade máxima obtida na época pode ser visto neste **link**.

Uma vez que o conjunto motor de passo e redução esteja definido, o desenvolvimento das demais peças para substituir as partes que antes continham os servomotores foi realizado. Cada nível planetário do sistema desenvolvido possui número de dentes da engrenagem sol Za = 12 e número de dentes do anel Zc = 48, o que multiplica por um fator de aproximadamente 5:1 o torque de entrada. Para as três juntas remodeladas, C, D e E, foram inclusos dois níveis - totalizando aproximadamente 25:1 por junta. O desenho do novo conjunto pode ser visto na figura 28, e a integração dele ao Projeto Pégaso original pode ser observada na figura 29. É possível perceber que, conforme explicado quanto às vantagens da unificação do tipo de motor na sessão 4.1, os links C e D podem passar a contar com peças iguais em suas estruturas - e o link E somente difere no formato externo do anel (peça azul na fig. 28) por ser o eixo do efetuador. O emprego de peças semelhantes simplifica o processo de design e tende a diminuir a complexidade na montagem do projeto.

Quando as novas peças foram impressas e os testes com esta nova configuração executados, os eixos com os novos motores funcionaram de maneira mais suave, conforme esperado, mas devido ao fato de terem cerca de quatro vezes o peso dos originais dois problemas foram observados de imediato com a montagem completa:

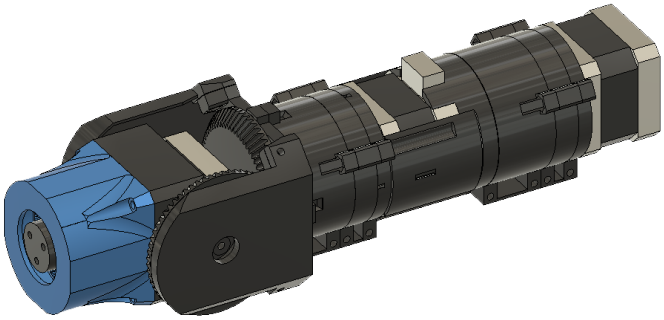


Figura 28 – Nova estrutura para os links C, D e E.

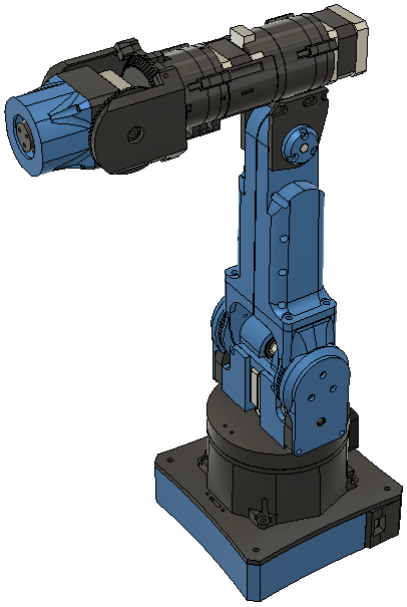


Figura 29 – Nova estrutura integrada ao corpo do robô original.

- Os eixos mais distantes da base, agora mais pesados, passaram a deslocar o centro de massa do robô com muito mais facilidade durante as movimentações do mesmo, levando ao tombamento da estrutura em casos mais graves; e
- O novo peso das juntas levou a uma solicitação maior de torque dos motores A e B, e no caso do primeiro chegaram a torná-lo insuficiente para manter o movimento a partir de certas inclinações desta junta. Com isto, o robô literalmente perdia a capacidade de se sustentar a partir de determinado momento e inevitavelmente a estrutura colapsava.

Com a ocorrência frequente destes graves defeitos de projeto, testes de movimentação ficaram muito restritos e impossibilitaram praticamente qualquer uso real do robô naquele estado - o que levou ao início da segunda iteração.

5.2 Segunda Iteração

Os problemas observados na primeira iteração demonstram que o motor A claramente precisa de uma redução maior que a 25:1 original da junta para garantir a sustentação do robô em todo o deslocamento de seus eixos. O desenho do conjunto de peças próximas à base, no entanto, não foi originalmente projetado para receber modificações desta natureza e tornaria pouco indicada qualquer tentativa de apenas alterar uma peça ou outra da região para enfim comportar uma redução maior. Optou-se então por redesenhar todo o arranjo de peças em torno dos motores A e B de modo a otimizar o processo de melhoria.

Não foi necessário projetar uma nova caixa de redução para o motor A, pois como mostrado na seção 5.1, um conjunto modular foi projetado e mostra-se útil neste contexto. Descarta-se então o conjunto redutivo original do motor A (ilustrado na figura 3) e em seu lugar inclui-se uma associação de três níveis da nova redução (figura 27), totalizando aproximadamente 125:1 ou cerca de 3500N.cm para o motor empregado. A nova estrutura que recebe este conjunto deve permitir novas expansões da caixa ou mesmo a troca do motor de passo por um NEMA 17 mais potente, caso constatada insuficiência mesmo após o aumento para 125:1. No entanto, conforme mostrado na figura 30, com os três níveis redutivos encaixados o motor A sozinho ocupa praticamente todo o espaço no topo da base, reduzindo o ambiente disponível para o motor B.

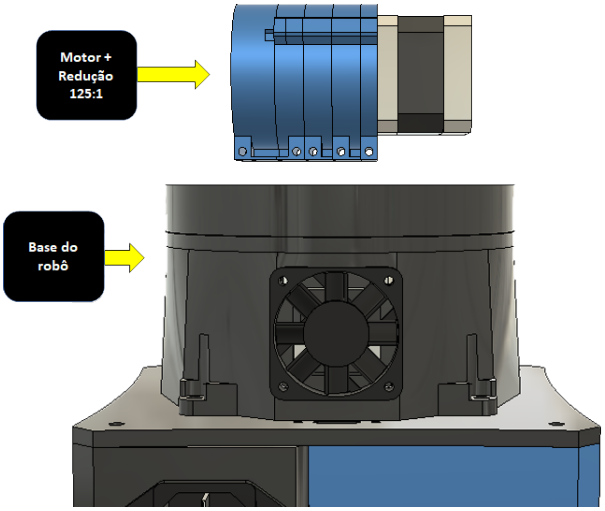


Figura 30 – Espaço ocupado pela nova redução da junta A.

Frente a estas condições, os motores A e B não poderiam mais serem dispostos de maneira simétrica como antes sem provocar um aumento significativo na área da base. Assim, procurando aproveitar a oportunidade das modificações para resolver também o item M3 da seção 3.6 (aumentar o deslocamento possível da junta B), optou-se por abandonar a simetria axial do robô e passar para um layout assimétrico, com o link A sendo movido para um dos lados da base em benefício de liberar espaço no meio do robô. Esta realocação se

Figura 31 – Robô industrial ABB IRB 6700.(ABB, s.d.[b])

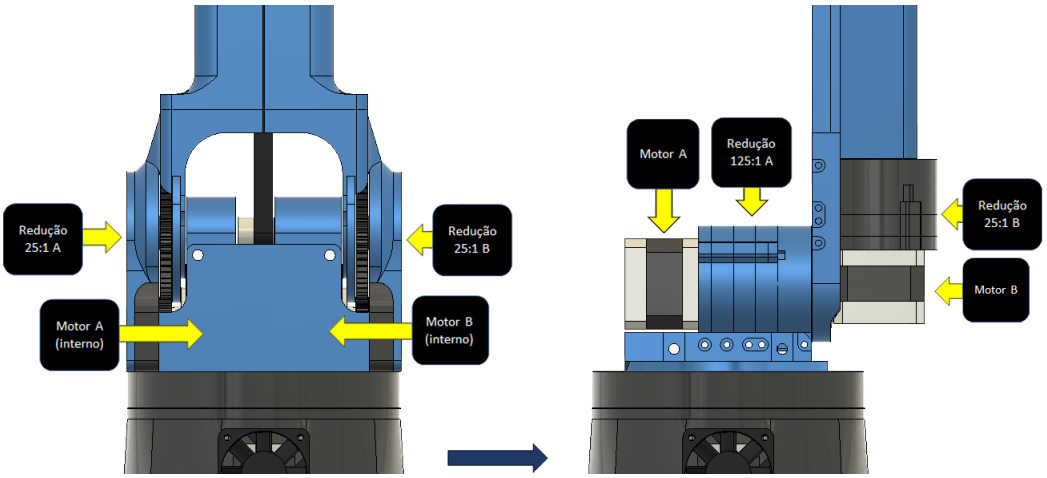


Figura 32 – Evolução no arranjo dos motores A e B.

assemelha à estrutura do modelo industrial ABB IRB 6700, ilustrado na figura 31.

Uma vez removido o motor B de sua posição original, entretanto, a redução 25:1 anexada a ele (figura 3) precisa ser readequada. Foi decidido estender o uso do novo conjunto planetário (figura 27) também para este motor, levando à unificação do tipo de redução para todos os eixos exceto a base. Essa decisão novamente economiza tempo na remodelagem das peças e promove a diminuição da variedade de componentes do projeto, vantagens já esclarecidas na seção 4.1. O novo arranjo desenhado para os motores A e B, com uma comparação visual entre o antes e o depois, pode ser visto na figura 32.

O reposicionamento do motor B manteve sua localização próxima à base para manter a distribuição de peso anterior, mas mudou a orientação de seu eixo, deixando-o paralelo ao próprio link A. Com isso, o sistema de transmissão também necessida de adaptações. Levando em conta este detalhe e sem esquecer do ponto de melhoria M2 da seção 3.6, que

envolve diretamente a transmissão da junta B e a diminuição da zona morta que nela havia, foram identificadas basicamente duas opções:

- adaptar o sistema de correia e polias vigente para o novo local, incluindo mecanismos de tensionamento para reduzir a zona morta do eixo; ou
- desenvolver um novo formato de transmissão, feito sob medida para a nova posição do motor B e conferindo potenciais vantagens sobre a solução original.

Enquanto a primeira opção mostrava-se razoável por demandar menos tempo de remodelagem, a segunda apresenta oportunidade de uma melhoria mais interessante para o sistema. Dentre as possibilidades analisadas, percebeu-se que escolher a alternativa mais trabalhosa e migrar do sistema de correias para um de eixo rígido poderia conferir nao só a diminuição da zona morta pela extinção do "efeito mola" da correia como também auxiliar na própria rigidez estrutural do robô, vantagem inegável ao se considerar que o link A passa a ser bem mais solicitado mecanicamente com o abandono do layout simétrico. Novamente fazendo uso de um eixo metálico cortado sob medida a partir de uma chave Allen 8mm, o sistema de transmissão foi remodelado para um novo formato, observável na figura 33. Uma outra vantagem não planejada que surgiu com o novo desenho foi a maior facilidade de montagem comparada à versão anterior.

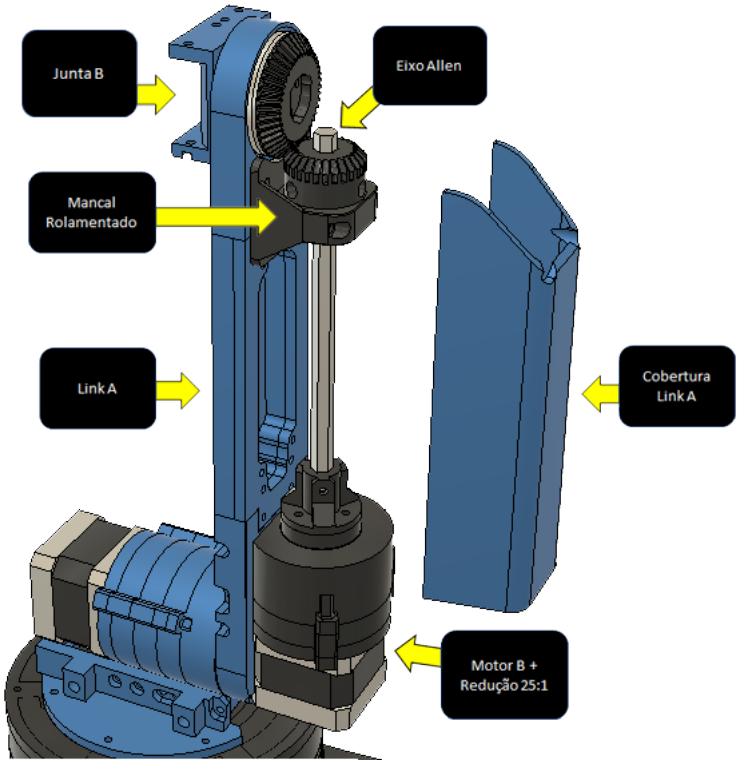


Figura 33 – Nova transmissão para a junta B.

Apesar da a nova forma de propagar o movimento do motor para sua respectiva junta prover o bônus de auxiliar na estrutura, julgou-se que seria interessante também incluir

peças próprias para esta função ao longo do link A, pois, com o layout assimétrico, o risco de empeno ou até colapso das partes mais solicitadas tende a ser bem maior do que quando se pode contar com apoio distribuído simetricamente. Estes transtornos são causados pelo momento da força que o peso dos links sustentados (no caso, C, D e E) gera quando o ponto de apoio (junta B) é posto em desalinho com o centro de massa. Essas peças podem ser impressas, mas o nível de amparo ideal escolhendo tal estratégia potencialmente ocorreria somente com componentes excessivamente grandes e trabalhosos de fabricar. Seguindo outro caminho, entendeu-se que contar novamente com elementos metálicos seria uma opção mais adequada de reforço estrutural - geralmente oferecendo rigidez maior que alternativas impressas para o mesmo espaço ocupado.

Assim, optou-se por anexar dentro das peças estruturais do link A uma dupla de hastes metálicas. A integração de tais hastes ao corpo da estrutura pode ser visualizada na visão explodida da figura 34.

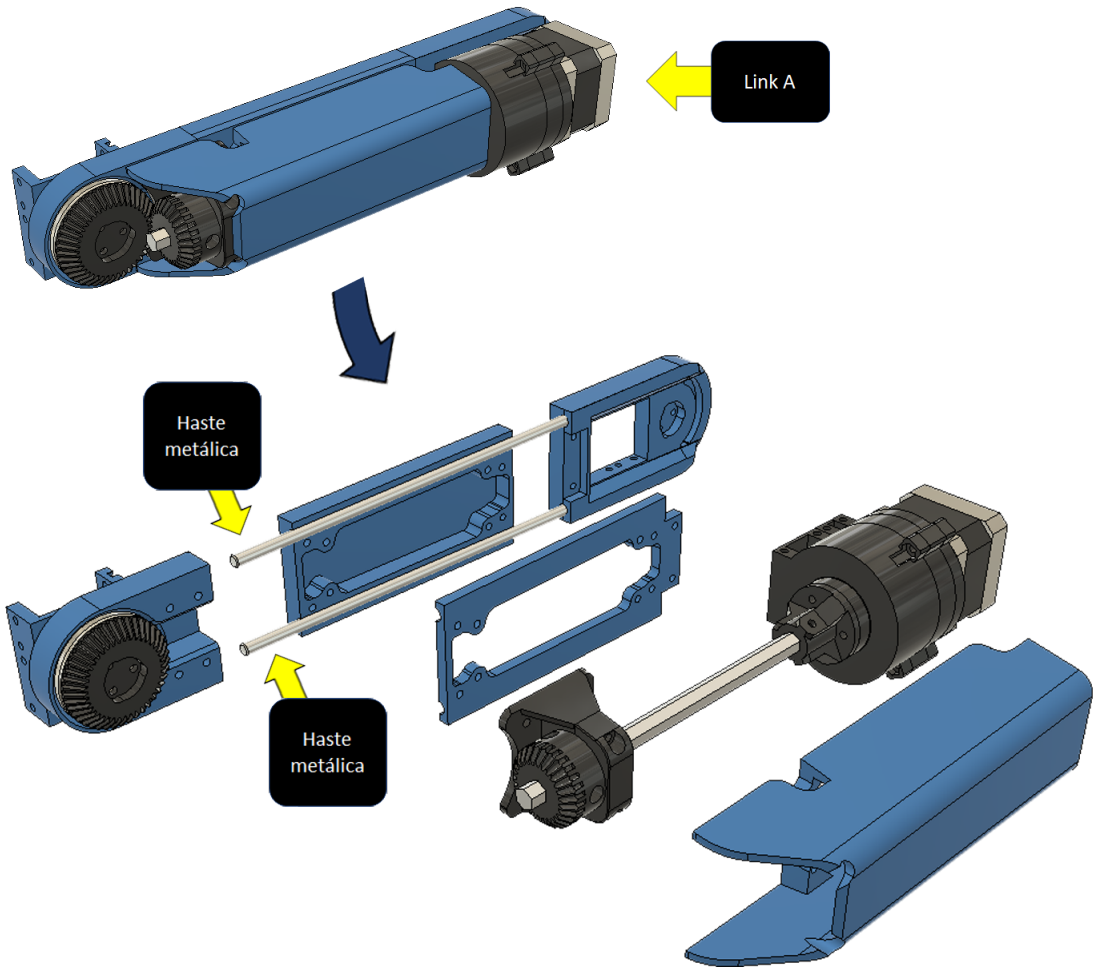


Figura 34 – Hastes metálicas integradas à estrutura do link A.

Há ainda uma última alteração, na estrutura do link D - que deixa de contar com peças de apoio nos dois lados do link E para sustentá-lo apenas pelo lado direito, levando a mais um trecho do robô onde uma solução assimétrica é empregada. A alteração, que pode

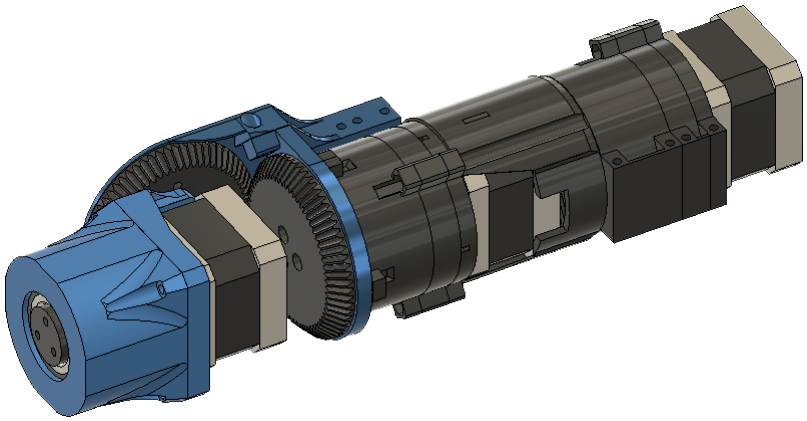


Figura 35 – Nova estrutura após modificação do link D.

ser vista na figura 35, se deu por três motivos principais:

- Durante a montagem física do robô foi observado que um lado apenas já era suficiente para sustentar o conjunto de peças da junta E;
- Em determindas posições atingidas pelo robô a presença do apoio no lado esquerdo restringe o movimento que o link C poderia propiciar ao robô sem causar colisões com o link A; e
- Na configuração anterior (ilustrada na figura 28) o motor era obrigado a ficar posicionado com a saída de fios posicionada verticalmente, o que fazia com que ela sofresse um amplo deslocamento durante a movimentação e, portanto, tivesse maior risco de desconexão espontânea, desgaste e rompimento eventual dos fios. Com o lado esquerdo livre de apoios o cabo do motor poderia sair nesta direção, e, alinhado com o eixo da junta D, sofrer um deslocamento menor. Este ponto pode ser exemplificado pela figura 36, onde observa-se que há uma diferença significativa na movimentação do cabo a depender de por qual lado ele se conecta ao motor.

Uma vez com as modificações dos links A, B, C, D e E (exploradas acima) projetadas e implementadas, resta observar oportunidades de melhoria estrutural na base do robô. Ela dispõe de uma estrutura simples, dividida em três seções principais conforme indicado na figura 37: fonte, tronco e link base.

- A *fonte* contém os elementos elétricos e eletrônicos do robô, como a entrada de energia (padrão AS02), a fonte chaveada 12V 5A e a placa de circuito impresso (PegasoBoard V1.0);
- O *tronco*, estrutura intermediária fixa em relação à fonte, contém uma ventoinha de refrigeração dos componentes eletrônicos e o motor de passo 17HS3401 que movimentava o link base.

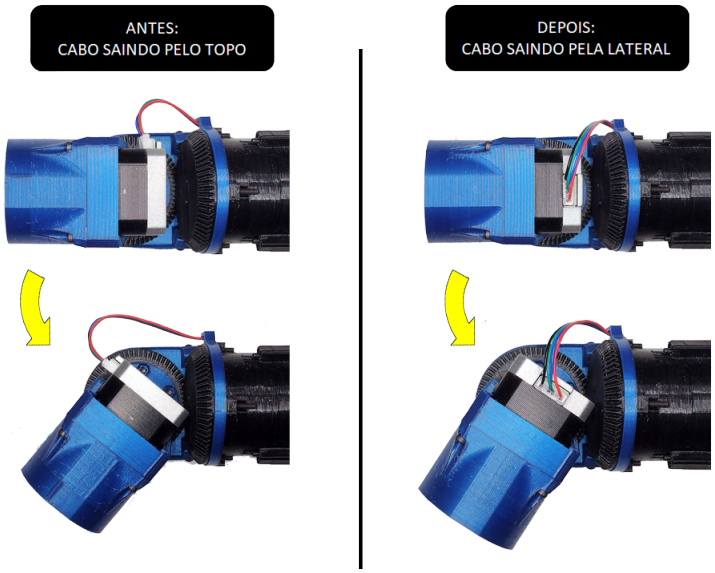


Figura 36 – Mudança no deslocamento do cabo.

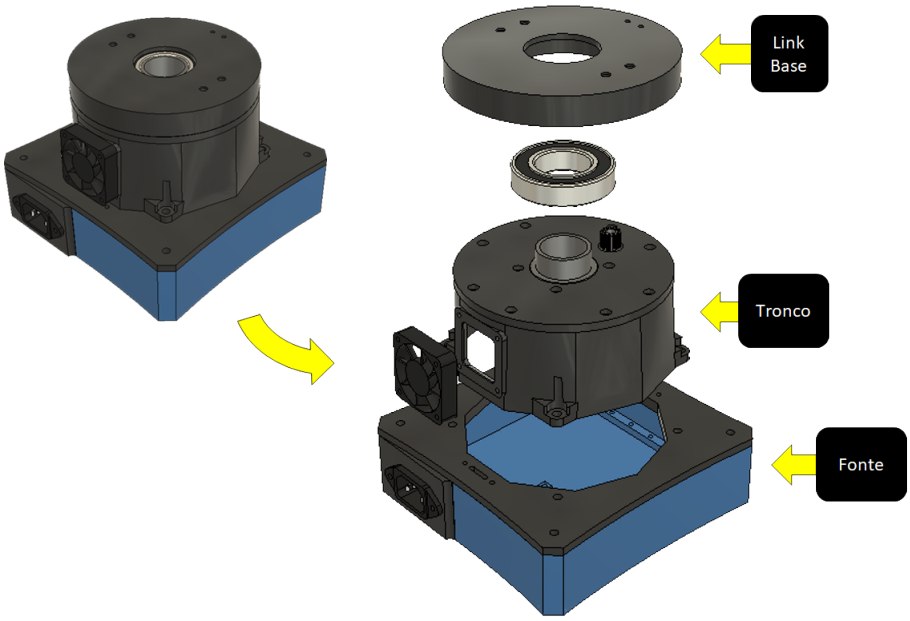


Figura 37 – Base original do robô.

- o *link base* é composto de um disco por onde todo o resto do robô se apoia, sendo a única estrutura dentre as três capaz de se mexer. Transmite para os demais links o movimento advindo do motor de passo por meio de uma redução 7:1. Esta redução, no caso, é composta por uma engrenagem de 10 dentes conectada ao motor no tronco e outra de 70 dentes integrada ao lado interno do link base.

A base do robô é capaz de mover o conjunto sem travamentos (quando respeitados os limites de velocidade da base) ou quebras de nenhum componente, mas por ter sido projetada levando em conta o conjunto antigo e mais leve de motores e peças, apenas um rolamento central 6006 2RS e um sistema de encaixe sem trava do disco no tronco foram

inclusos no robô original. Com a atualização do layout de motores, no entanto, estender suficientemente o robô no plano horizontal ("esticá-lo para frente") move o centro de massa do sistema para fora do amparo da fonte e inevitavelmente tomba o robô na direção do deslocamento.

Mesmo que a parte imóvel da base seja presa, a parte giratória desacopla-se por ser somente encaixada no tronco (sem o auxílio de travas) e o tombamento ocorre novamente. Já em situações de movimento dentro da "zona segura", sem ocorrência de tombamento, também é observado que o link base se inclina em relação ao tronco conforme o robô se move, denotando uma clara falta de apoio ao deslocamento do robô. Em face a estes problemas, decidiu-se:

- alterar o desenho das peças para incluir mais rolamentos na superfície do tronco, orientados radial e perpendicularmente em relação ao rolamento central para prover suporte extra ao disco móvel;
- modificar também o desenho da peça central do tronco (onde se encaixa o rolamento) para conter uma trava que impeça o desacoplamento espontâneo do link base; e
- incluir contrapesos para compensar o deslocamento do centro de massa do robô.

A principal peça a ser modificada com base nestas decisões é o tronco, que conforme visto na figura 37 foi concebida em um único volume. Apesar disso tornar o design e o encaixe no conjunto da base mais simples e direto, o fato de ser um componente inteiriço relativamente grande dificulta a fabricação, já que o tempo de impressão deste item pode facilmente atingir 9 horas de trabalho e os riscos de aparecerem defeitos durante o processo que prejudiquem as propriedades estéticas ou mecânicas da peça final são maiores. Um outro problema está relacionado à falta de compatibilidade do projeto com novas mudanças: qualquer alteração que deseje-se promover no tronco significa ter de reimprimir toda a peça de uma vez. Com a análise das decisões tomadas acima e destas desvantagens, O desenho do tronco foi refeito e está ilustrado na figura 38.

Com ele, o topo do tronco - ainda sujeito a alterações - fica separado da parede, permitindo que modificações possam ser feitas apenas na região. A parede está dividida em três segmentos para auxiliar no processo de impressão. As agora 4 partes que compõem o tronco unem-se por meio de parafusos M5 e não apresentam dificuldades de montagem adicionais frente à solução de peça única inicialmente adotada.

conforme ilustra a figura da nova base, o lado superior do tronco conta com quatro rolamentos de apoio igualmente espaçados e uma peça central com furos verticais. Estes furos alojam parafusos M3 que fixam a agora existente trava vertical do link base (peça esta fabricada da mesma maneira que as arruelas de três pontos). Assim, o problema do desa-

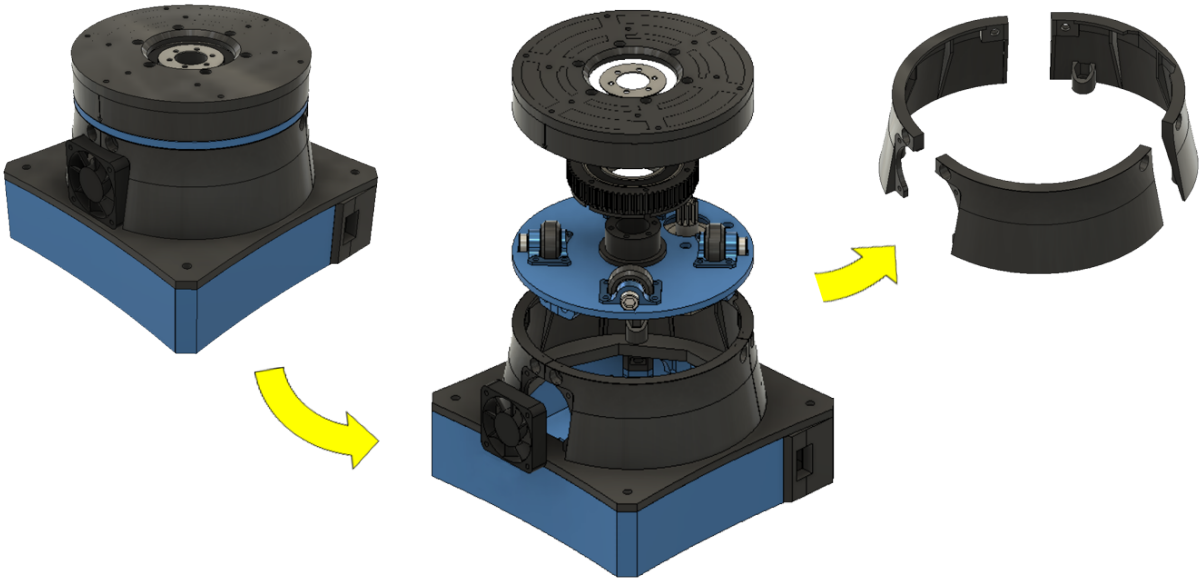


Figura 38 – Base nova do robô.

figs/contrapeso.png

Figura 39 – Contrapeso instalado atrás do link A.

coplamento está resolvido e os novos elementos de apoio ao movimento do robô conferem maior robustez ao sistema.

Para lidar com o problema relatado de tombamento inclui-se uma peça metálica de cerca de 700g no topo do link base, logo atrás do motor A (figura 39), mas somente como uma solução paliativa ao problema ao considera-se que, dadas a posição e massa do objeto, o único resultado a ser produzido com a adoção deste é um aumento da "zona segura", sem ainda cobrir todo o alcance horizotal do robô. Um contrapeso maior e justificado por cálculos será implementado na terceira iteração.

De posse das modificações decididas, o robô montado com todas as implementações relatadas (observável na figura 40) já não possui muitas das peças originais e mantém pouca semelhança com o robô usado como ponto de partida. De fato, somente a estrutura da fonte (indicada na figura 37) permaneceu intacta. Com o final da segunda iteração, o manipulador passa a ser chamado de PegasoV3 e conta com melhorias acumuladas correspondentes aos itens M1, M2, M3, M4, M5 e M7 dos pontos descritos na seção 3.6.

Especificamente com relação ao primeiro deles - substituir todas as peças de PLA por PETG - este ocorreu naturalmente conforme novas peças foram confeccionadas para atender às melhorias descritas, com as cores das peças mantidas por razões estéticas. Quanto ao segundo - diminuir a zona morta na transmissão da junta B - constatou-se que o novo sistema de eixo rígido (figura 33) a apresentava zona morta na transmissão sensivelmente menor, em torno de 2°, com o que restou sendo migrado para a redução planetária do motor

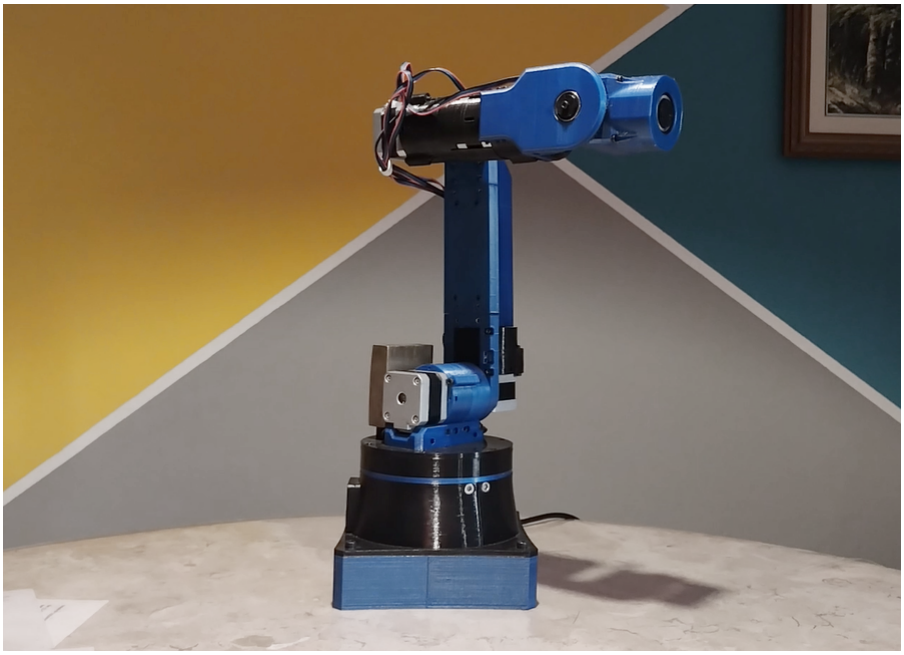


Figura 40 – Projeto Pégaso após as mudanças da segunda iteração.

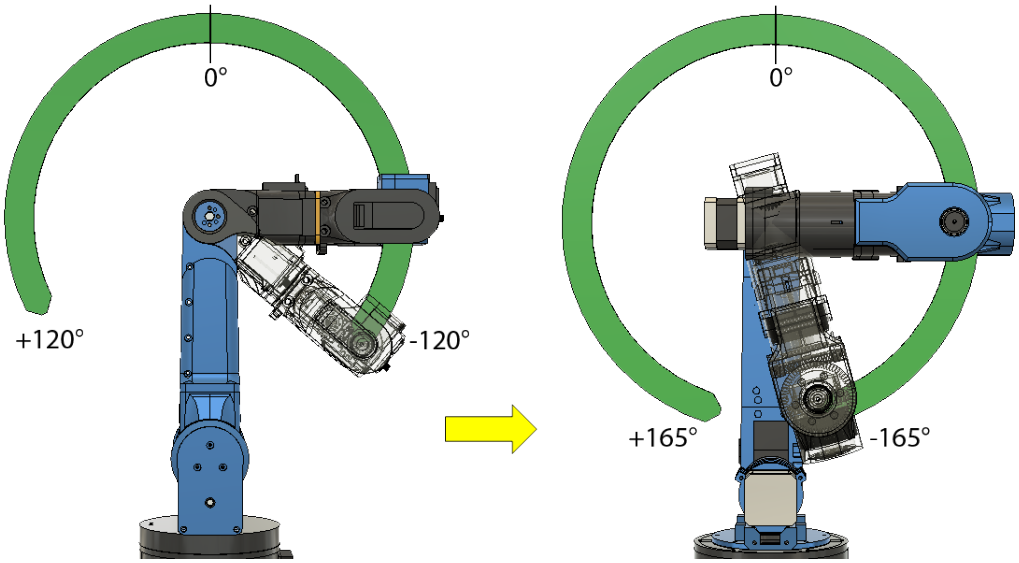


Figura 41 – Evolução da amplitude do link B após a segunda iteração.

B.

Em relação ao terceiro ponto de melhoria, aumentar o deslocamento possível da junta B, observa-se que foi o mais visivelmente atendido com as modificações discorridas nestas duas iterações. Conforme ilustrado na figura 41, a amplitude de cerca de 240° do robô inicial se transformou em um montante de aproximadamente 330°, em grande parte graças à assimetia do novo design.

O robô não chegou a ser testado para o propósito cinematográfico ao final desta iteração, mas movimentações genéricas foram devidamente registradas e podem ser conferidas nos links a seguir. É notável que, apesar de ainda imperfeita, a movimentação da máquina

está mais suave em relação ao ponto de partida.

- VÍDEO 1
- VÍDEO 2

5.3 Terceira Iteração

Enquanto as duas primeiras iterações do projeto mecânico foram responsáveis por transformar profundamente o desenho do robô, levando-o a uma versão modificada em relação à original batizada de PegasoV3, esta terceira fase tem como objetivo principal o refinamento de aspectos do projeto já existentes e a inclusão de características que melhoram a interação do usuário com o robô. Especificamente, esta iteração versa sobre:

1. a substituição do contrapeso genérico adotado por uma peça projetada sob medida para o PegasoV3;
2. a modificação do desenho da base para incluir acessos ao cartão microSD e à entrada microUSB da placa de circuito; e
3. a substituição dos motores de passo dos links D e E por alternativas mais leves.

Para o desenvolvimento do contrapeso julgou-se primeiramente que a melhor localização para a peça deve ser o mais baixo possível, assim contribuindo para diminuir a altura do centro de gravidade e conferindo maior estabilidade ao conjunto. Desse modo, a peça genérica inserida na segunda iteração (figura 39) dá lugar a uma placa inteiriça colocada abaixo da fonte, com formato seguindo a silhueta do local de instalação e espessura determinada a partir da densidade do material e do peso necessário para balancear o robô. O pior caso utilizado como referência para o cálculo do contrapeso foi aquele com o robô completamente esticado horizontalmente, conforme ilustrado pela figura 42.

Nesta figura considera-se a extremidade da fonte (ponto A) como o "mancal de apoio" do cálculo estático de peso, pois ela corresponde ao último ponto de suporte que a base pode prover. Se o contrapeso for capaz de manter o centro de massa dentro da região indicada pela letra B, é razoável considerar que o robô não irá tombar. Um grau de certeza maior somente é possível com a inclusão de cálculos de mecânica dinâmica, mas dado o fato de o PegasoV3 ser desprovido de atingir grandes acelerações e a natureza bem mais complexa deste assunto, entendeu-se que a princípio o estudo dinâmico não é necessário, e sua realização será feita somente caso a abordagem estática se mostre insuficiente. Como uma forma de compensar a ausência deste estudo, no entanto, o valor teórico encontrado terá um acréscimo de 15% como fator de segurança.

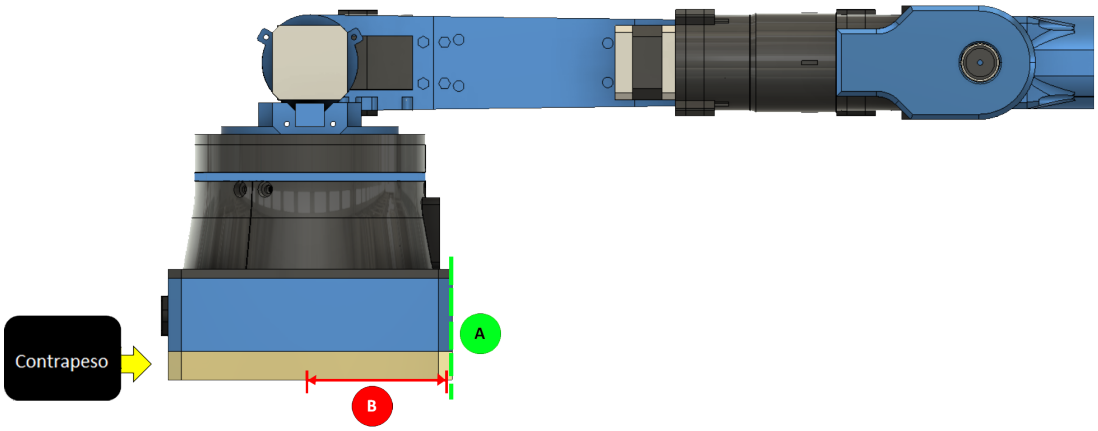


Figura 42 – Pior cenário de solicitação de esforços no robô.

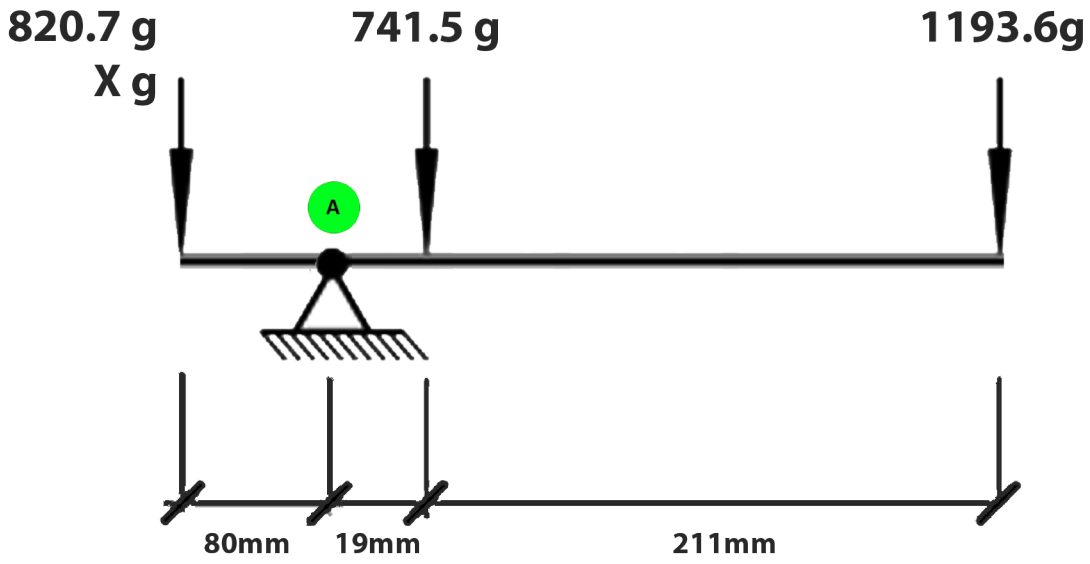


Figura 43 – Representação esquemática da distribuição de esforços.

A figura 43 representa a distribuição de esforços pontuais correspondentes ao cenário da figura 42. Para obter os valores e distâncias mostrados, desmontou-se o robô em três segmentos que constituem corpos rígidos entre si e mediu-se suas massas e centros de gravidade individualmente. Os respectivos segmentos e valores medidos foram:

- segmento 1: base (820.7g) e contrapeso (**X**g)
- segmento 2: link A (741.5g)
- segmento 3: links B, C, D e E (1193.6g)

Com estes dados em mãos, foi possível estabelecer a equação de equilíbrio dada a seguir, que encontrou um valor de 2,787kg no contrapeso (assumindo uma distribuição homogênea da densidade). Acrescendo-se o fator de segurança de 15%, o peso esperado para o contrapeso seria de cerca de 3,205kg.

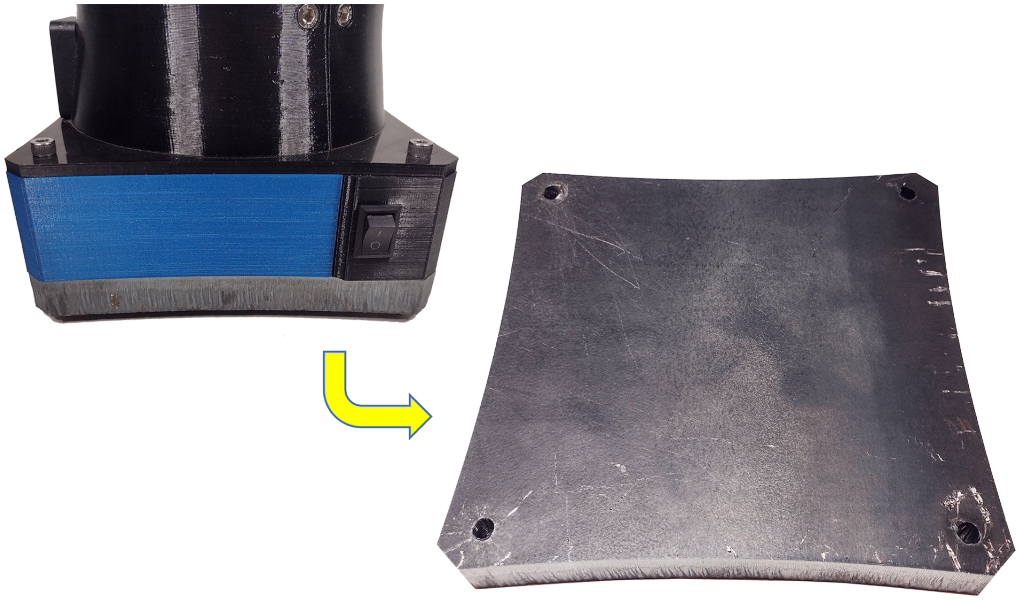


Figura 44 – Novo contrapeso encomendado sob medida.

```
(80)(820,7) + (80)X = (19)(741,5) + (19 + 211)(1193,6)
65656 + (80)X = 14088,5 + (230)(1193,6)
65656 + (80)X = 14088,5 + 274528
65656 + (80)X = 288616,5
(80)X = 288616,5 - 65656
(80)X = 222960,5
X = 2787,006 g
X = 2,787 kg
```

considerando a área da base de 256cm2 e o aço SAE 1008 (d=7,872g/cm3) como referência para a confecção do contrapeso, a espessura da chapa a ser cortada deve ser de 16mm para gerar uma peça de 3.224kg (acima dos 3.205kg, conforme determinado). A peça resultante, bem como sua localização na base do robô, pode ser visualizada na figura 44.

Prosseguindo para as modificações da base que permitissem o acesso ao cartão de memória e à entrada USB da placa de circuito - item 2 da lista - optou-se por seguir uma estratégia similar ao do novo tronco (figura 38) e refazer a estrutura da fonte - originalmente também constituída em uma peça única - para separar o forro das paredes. Esta iniciativa se deu pelos mesmos exatos motivos expostos na segunda iteração: uma peça inteiriça não só significa mais horas de fabricação de uma vez como também é menos adaptável a mudanças futuras.

O novo forro conta com pontos de aparafusamento específicos da PegasoBoard v1.0 em vez de deixá-la "flutuando" no espaço da fonte destinado a ela, e levando em conta a inclusão do novo contrapeso (figura 44), não mais precisa ser um elemento estrutural - afinal, toda a rigidez necessária para o componente passa a ser garantida pela peça metálica recém-instalada. Com estas mudanças, o forro da fonte pode ser uma mais fino e ter espaços vazados

para economizar tempo e material para fabricação. As novas paredes, por sua vez, podem ser adaptadas e modificadas individualmente conforme a necessidade de desenvolvimento do projeto. Como a peça inteiriça original não possuía nenhum tipo de acesso em suas paredes, qualquer alteração no código-fonte ou nos arquivos do cartão de memória requeriam que antes fosse desmontada a base do robô. Na nova versão, ilustrada na figura 45, estão inclusos um rebaixo para acesso ao cartão de memória em uma das paredes e um acesso à porta microUSB da PegasoBoard em outra. Além disso, nas quinas internas são inclusas peças com porcas M5 para garantir uma maior confiabilidade do acoplamento do contrapeso na fonte (a solução anterior contava com roscas na própria peça impressa). A impressão das peças diminuiu de cerca de 10 horas da unidade inteiriça original para entre 2 a 3 horas por cada parte do conjunto, o que representa uma maior chance de sucesso na fabricação e um menor custo de reimpressão caso haja algum problema.

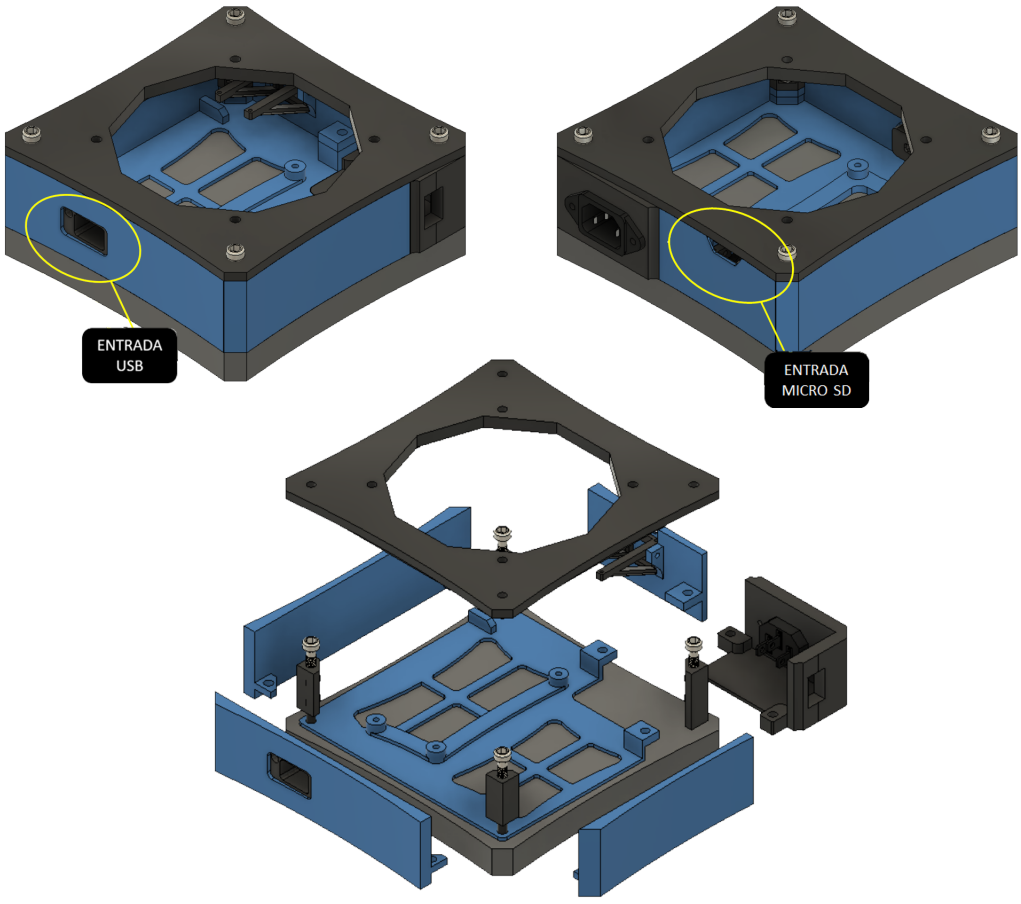


Figura 45 – novo desenho da fonte.

Avança-se agora para o terceiro item da lista. O robô utilizado como ponto de partida conta com os dois conjuntos de motores mencionados na primeira seção do capítulo 3: três motores de passo (para os links base, A e B, respectivamente) e três servomotores (links C, D e E). Esta configuração foi escolhida originalmente pelas seguintes razões:

- motores de passo: acessíveis, fáceis de controlar e com movimentos suaves e precisos.

Possuem torque nativo baixo e peso considerável, mas podem contar com caixas de redução e ser dispostos bem próximos à base para evitar interferências negativas na carga paga do robô.

- servomotores: igualmente acessíveis e fáceis de controlar, mas com 1/4 do peso dos motores de passo e torque sensivelmente maior, o que os torna adequados para o posicionamento próximo ao efetuador terminal. Possuem, no entanto, funcionamento com ruídos no giro e resolução mínima de apenas 1° por passo, valor considerado grosseiro demais para a aplicação cinematográfica.

Na primeira iteração esta configuração foi modificada, com a substituição dos servomotores por também motores de passo. Devido ao princípio de unificar os componentes do projeto para facilitar a obtenção de peças e a montagem do conjunto, os motores substitutos foram do mesmo modelo que os já adotados para os links base, A e B - 17HS3401. Trata-se de um motor padrão NEMA 17 com 28N.cm de torque estático segundo o datasheet (2…, s.d.).

No entanto, optou-se por realizar uma nova troca após haver a constatação que os motores dos links D e E podem estar superdimensionados. Afinal, o link B, responsável por sustentar um peso maior que os outros dois, conta com a mesma redução 25:1 e portanto gera o mesmo torque que os demais. Se eventuais problemas relacionados ao torque insuficiente de um motor aparecerem, necessariamente o farão primeiro através da junta B. Assim, as juntas D e E são novamente modificadas, desta vez para receber alternativas mais leves de motor de passo. Com isso pode haver o benefício do alívio do peso nas juntas mais distantes da base sem compromissos reais de perda de capacidade da carga paga.

Mantendo-se os novos motores na tipologia NEMA, há a vantagem de não ser necessário remodelar nenhuma peça das juntas alteradas. Isto ocorre porque o padrão NEMA determina características das dimensões, tolerâncias, cabeamentos e aspectos funcionais dos motores que devem ser respeitadas por todos os modelos, tornando a intercambiabilidade de peças uma vantagem na escolha de tais componentes. Os NEMA 17, conforme ilustrado na figura 46, têm diâmetro do eixo de 5mm (podendo ser cilíndricos ou do tipo ’D’) e base de 42mm x 42mm com chanfros a 45 graus. A ampla maioria também possui 1.8° por passo, e o que pode mudar de um modelo para outro é basicamente o comprimento do estator - quanto mais longo, mais pesado será o motor e maior tenderá a ser o torque que ele gera.

Os componentes escolhidos para a substituição são do modelo 17HS4023 - um motor NEMA 17 de apelido "pancake" originalmente criado para uso em impressoras 3D. Segundo seu datasheet (2…, s.d.), ele apresenta 14N.cm de torque estático - metade do proporcionado pelos motores substituídos - mas também apresenta um comprimento do estator menor (28mm ante 40mm) e, conforme ilustra a figura 47, 82.4g a menos de massa. Com a substituição ocorrendo nos links D e E, são 164.8g poupados ao todo.

Devido a serem mais curtos, os novos motores também podem contar com espaçado-

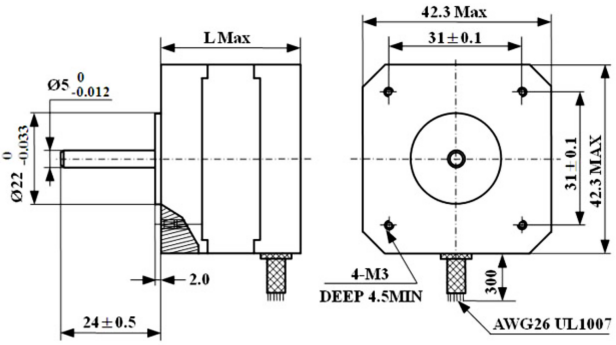


Figura 46 – Padrão NEMA 17.(PROTOTYPING, s.d.)

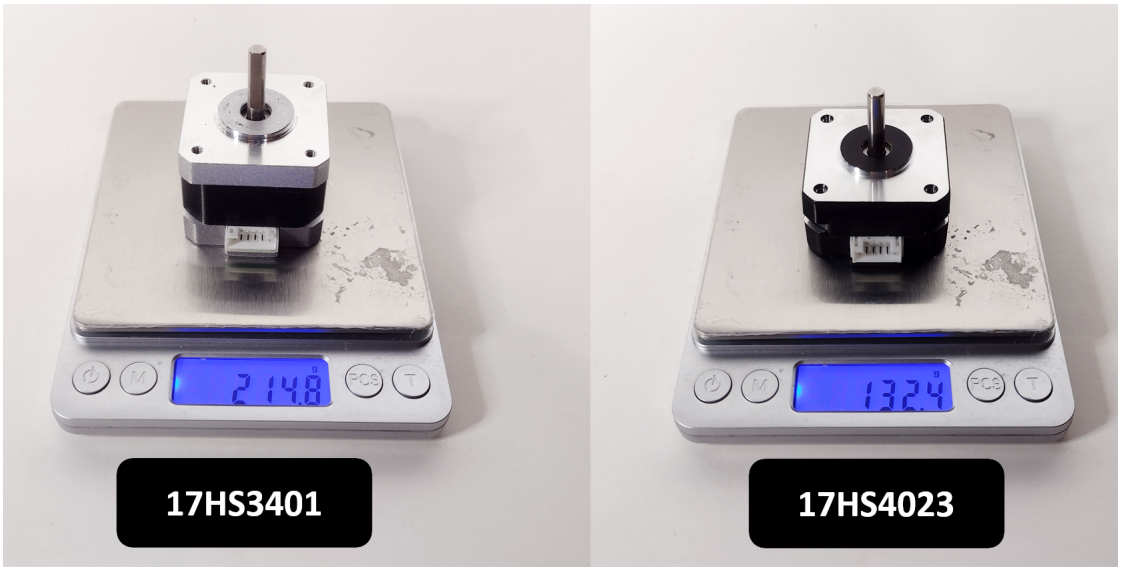


Figura 47 – Comparação de peso entre os motores 17HS3401 e 17HS4023.

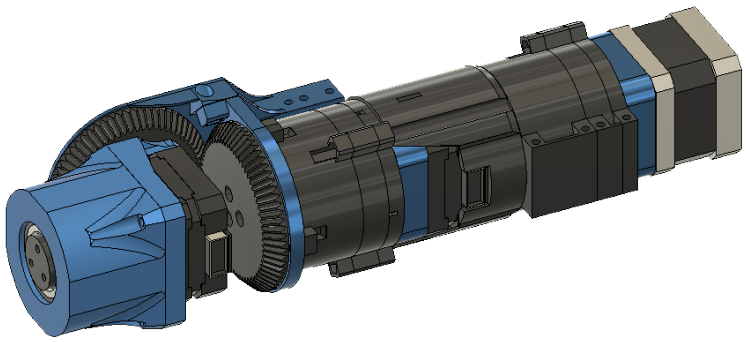


Figura 48 – Nova estrutura após a troca dos motores D e E.

res que preenchem o espaço vazio deixado e serem recuados um pouco. Assim, deslocam o centro de massa das juntas D e E para mais próximo da base. Enquanto a figura 48 ilustra o resultado, a imagem 49 mostra a mudança obtida: 165.5g a menos de peso e um recuo aproximado de 22mm no centro de massa do conjunto.

A figura 49 mostra que o link C recebeu o mesmo espaçador, apesar de não ter tido envolvimento com a troca relatada no parágrafo anterior. Isto ocorreu em caráter adicional após ser constatado que o espaçador utilizado para afastar o motor D também pode afastar

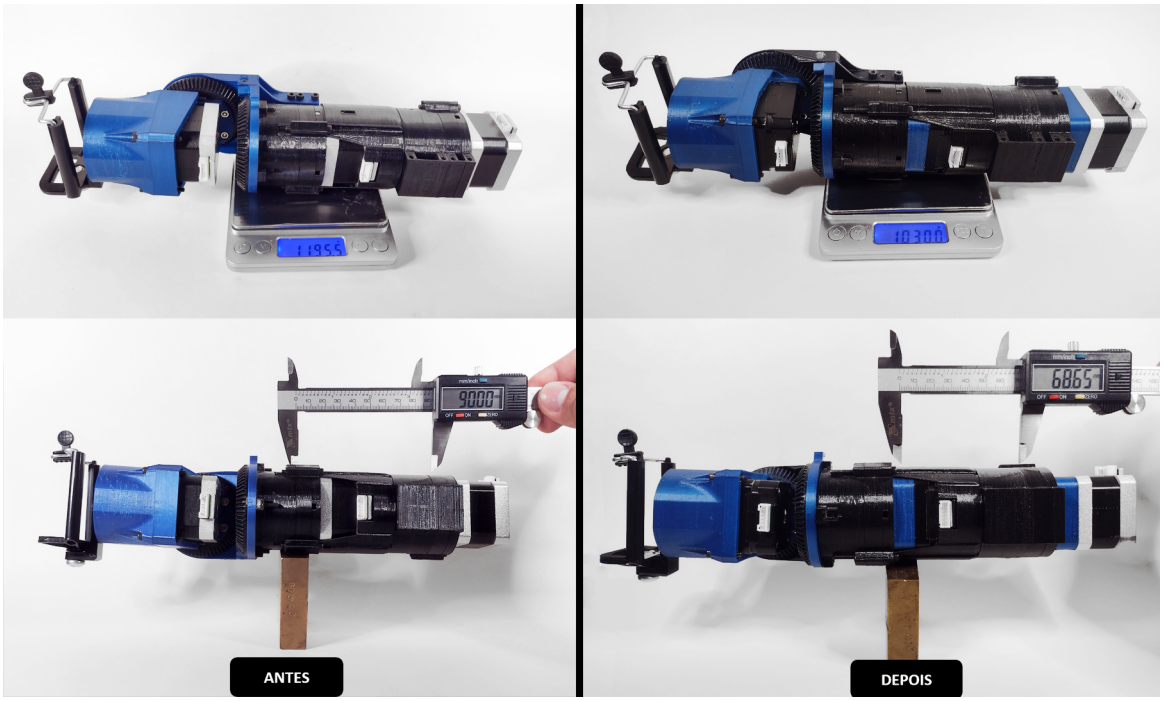


Figura 49 – Melhoria de peso e recuo do centro de massa da nova estrutura.

o motor C, e assim deslocar ainda mais o centro de massa do conjunto CDE para perto da base. Como pode ser visto, as modificações surtiram o efeito desejado de redução de peso e conseguiram mover o centro gravitacional na direção desejada em cerca de 22mm, favorecendo a estabilidade e a carga útil do robô. Também é conveniente observar que a com as mudanças descritas a margem de segurança do contrapeso encomendado aumenta.

Para finalizar as modificações da terceira iteração, são incluídas em caráter estético três peças para cobrir o motor e a redução 125:1 da junta A. Também foram incluídos organizadores nos cabos dos motores B, C, D e E para diminuir as chances de enganchamento durante as movimentações das juntas. O resultado após todas as modificações pode ser conferido na figura 50, e o novo volume de trabalho do robô pode ser visto na figura 51.

Figura 50 – Robô após modificações da terceira iteração.

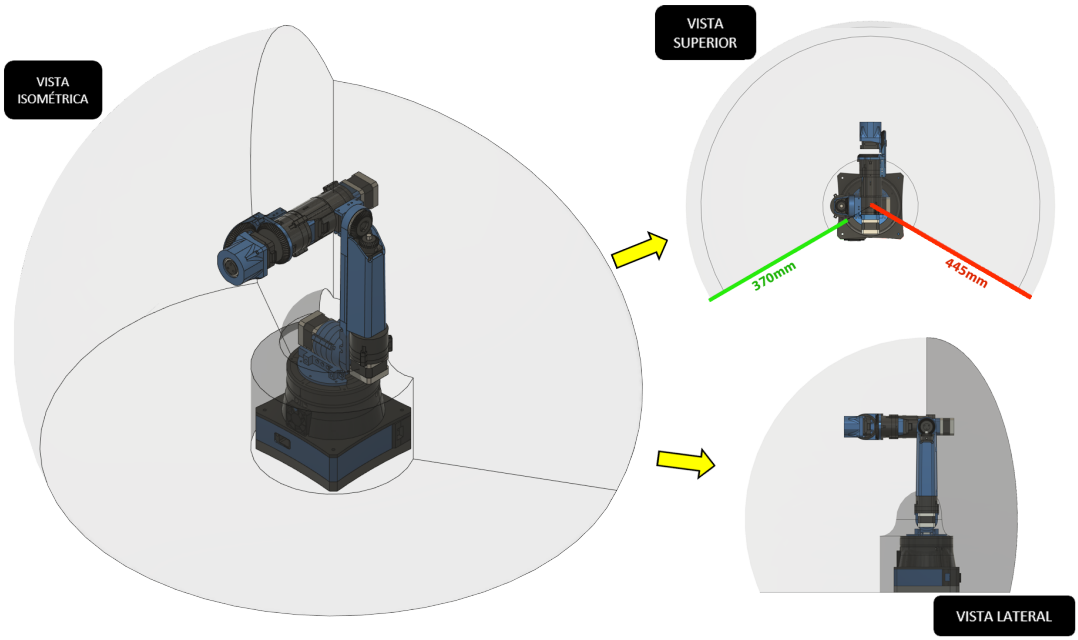


Figura 51 – Volume de trabalho do PegasoV3.

6 Implementação do Controle Cartesiano

Com o robô tendo passado pelas três rodadas de mudanças estruturais abordadas na seção anterior e obtido relativo sucesso por entre elas, conseguindo atender ao objetivo da troca de motores, aumento do deslocamento da junta B (cotovelo) e a adoção de um contrapeso adequado (entre outros feitos), entendeu-se que a questão mecânica encontra-se bem encaminhada e que o desenvolvimento do controle cartesiano do PegasoV3 pode ser iniciado. Ao longo deste capítulo serão abordadas a teoria, as análises e a implementação relativas ao assunto.

6.1 Breve introdução à cinemática

A cinemática (aplicada à robótica) é um recurso que estabelece uma correlação entre os ângulos das juntas do robô e a posição ocupada por elas no espaço tridimensional. Assim, é de grande utilidade porque permite que se controle o movimento da máquina focando na trajetória a ser feita ao invés de no ângulo específico que cada uma de suas juntas está se posicionando, e isto configura uma maneira de operar o robô muito mais intuitiva por ser próxima à forma como humanos controlam os seus próprios membros. Com o uso da cinemática, tarefas como mover o efetuador terminal em linha reta ficam transparentes ao usuário, evitando que este tenha de realizar os movimentos uma junta de cada vez.

Além disso, como é possível calcular com ela a localização espacial de todas as juntas do robô a partir de seus ângulos, fica possível também estabelecer restrições adicionais entre as juntas que expandam o controle para não apenas *em que ponto* no espaço deseja-se chegar, mas também *com que orientação*, e isto é especialmente relevante para o contexto cinematográfico porque a inclinação da câmera em relação ao objeto filmado constitui um dos pontos chave da composição de uma cena, seja ela foto ou vídeo. Conforme ilustra a figura 52, se dispõe-se da posição desejada para o efetuador e o comprimento L da estrutura que o conecta com a junta precedente, pode-se definir um ângulo α com que queremos o efetuador alcançando o alvo e, usando trigonometria simples, descobrir o ponto no espaço que tal junta precedente deve assumir.

Existem duas maneiras de utilizar a cinemática em favor dos cálculos de movimentação do robô, sendo a decisão de qual delas usar baseada na informação que se deseja obter. Se há o conjunto de ângulos das juntas e com elas deseja-se a posição no espaço em que o robô se encontra, o método é chamado de Cinemática Direta. Se por outro lado há a posição cartesiana e quer-se encontrar os ângulos, então temos a Cinemática Inversa. Como o foco do

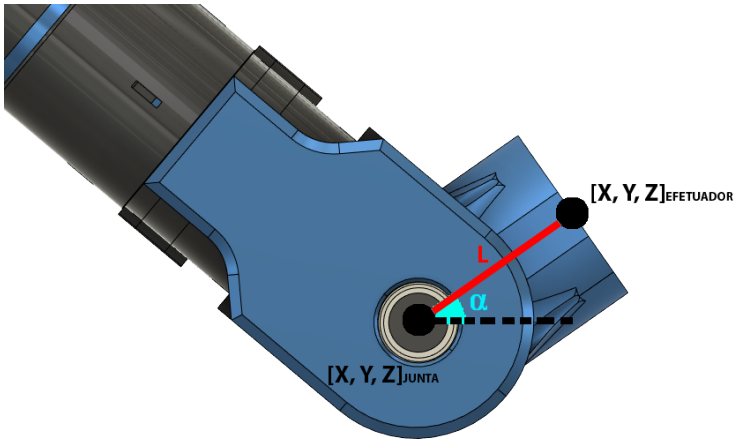


Figura 52 – Posição da junta precedente a partir da inclinação.

controle cartesiano envolve definir trajetórias e fazer o robô percorrê-las, o método inverso tende a ser mais utilizado para calcular os ângulos desejados - e o direto fica reservado para confirmar se com os ângulos calculados realmente chega-se ao ponto pertencente à trajetória.

Entretanto, apesar de ser geralmente usada em caráter complementar, a cinemática direta tende a ser mais fácil de lidar por ter somente uma solução possível para o conjunto de ângulos fornecidos para a conta. Em contraste, como ilustra a figura 53, o método inverso pode apresentar mais de uma solução válida e, a depender de como é calculado, também não chegar a nenhuma. Por esta razão, ao longo do tempo foram surgindo novas propostas de algortimos para a cinemática inversa que procuravam resolver os problemas descobertos nos anteriores.

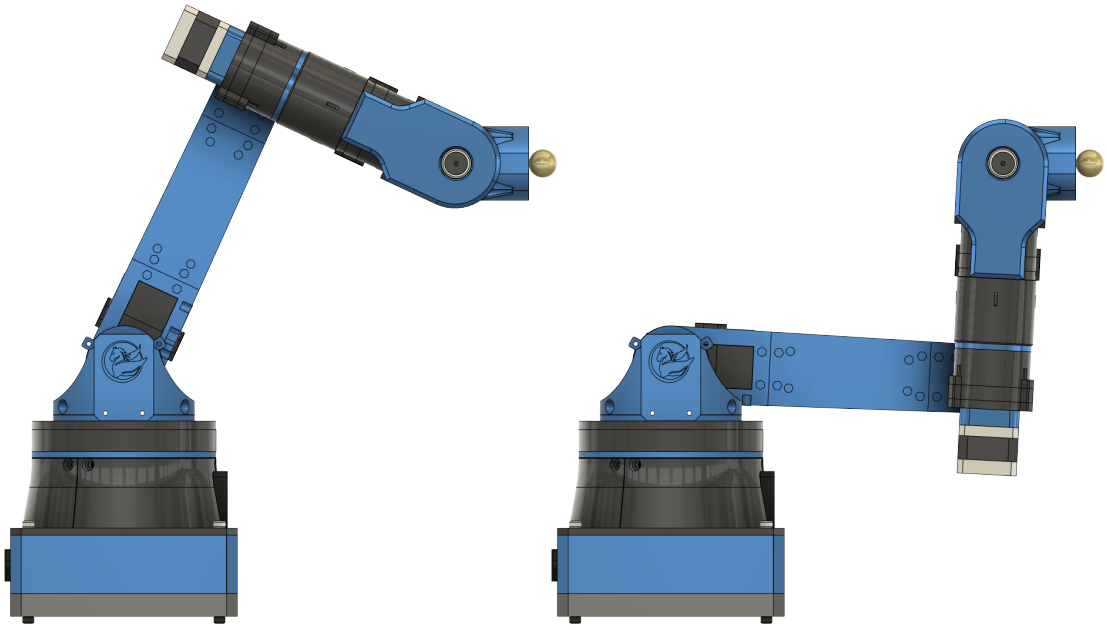


Figura 53 – Duplicidade de soluções possíveis para um mesmo ponto no espaço.

Antes de qualquer implementação, porém, é necessário primeiramente derivar, a partir da estrutura do robô, o seu *modelo cinemático* (também chamado de *cadeia cinemática*). Este procedimento corresponde a reduzir o corpo da máquina a um conjunto de formas simples que represente acuradamente as articulações e trechos rígidos do equipamento, conforme ilustrado na figura 54. A partir daí, o próximo passo é inserir o modelo do robô em um espaço cartesiano, com origem e eixos ortogonais bem definidos, pois é com relação a estes eixos que a máquina será localizada espacialmente. Essa inserção pode ser observada na figura 55.

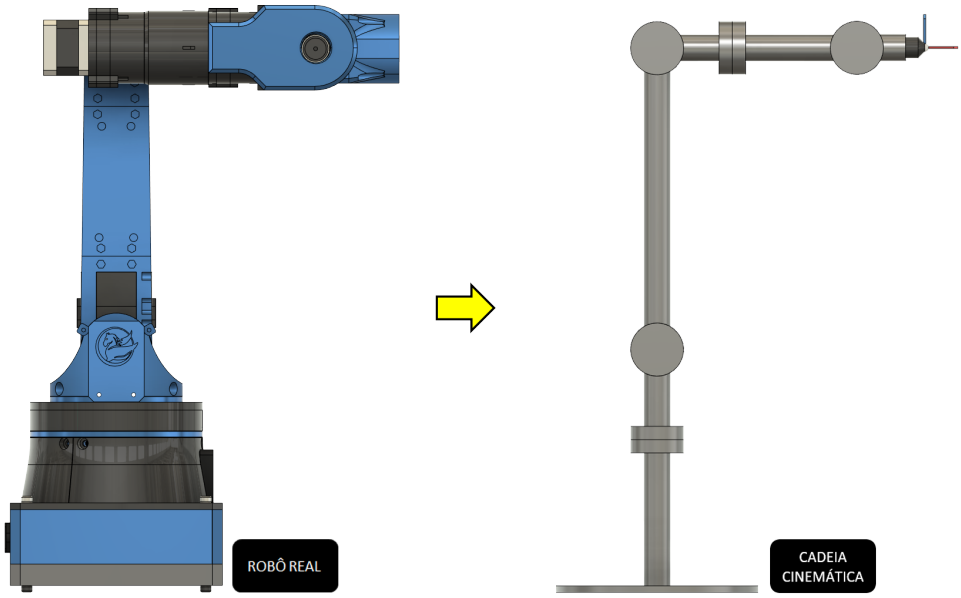


Figura 54 – Derivação da cadeia cinemática a partir do robô real.

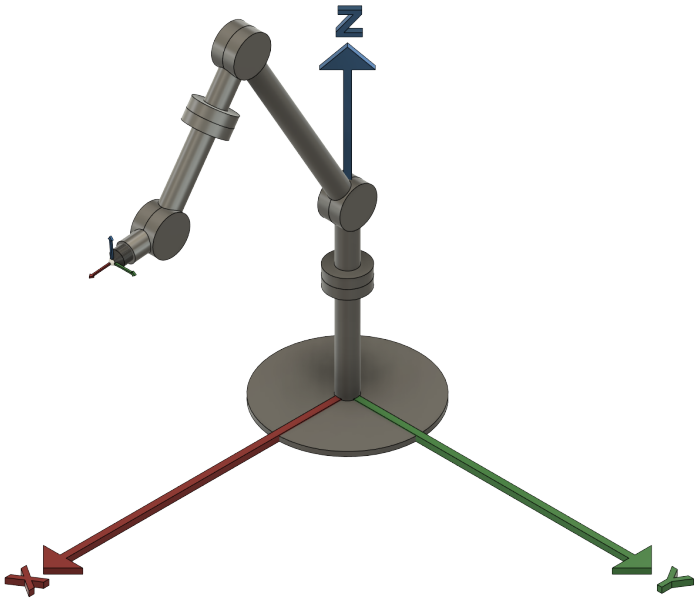


Figura 55 – Cadeia cinemática inclusa nos eixos ortogonais.

Através da figura 55 também é possível observar a escolha feita para os sentidos de

cada eixo. O eixo X está alinhado com a posição inicial do robô e representa portanto o que seria o movimento "para frente e para trás". O eixo Y mensura o deslocamento "para a esquerda e para a direita", e por fim, o eixo Z responde pela movimentação "para cima e para baixo". Nestes quesitos, o sentido +X vai para frente, o +Y para a esquerda e o +Z para cima. Já a origem do sistema, onde os três eixos ortogonais se encontram, está localizada no centro geométrico da base, ao nível do chão.

6.2 Trabalhos Correlatos e Escolha do Algoritmo-Base

Conforme dito anteriormente, não existe uma única forma de se calcular a cinemática inversa de um robô, pois ao longo dos anos diversas soluções foram sendo apresentadas por diferentes autores em busca de corrigir inconsistências de modelagens precedentes. Um dos métodos mais difundidos atualmente, bastante usado na animação de personagens em jogos e proposto por Wang e Cheng em 1991 (WANG; CHEN, 1991), é o chamado Descendente de Coordenadas Cíclico (*Cyclic Coordinate Descent*, ou CCD). Indicado por ser simples em termos de cálculos computacionais, tal método tem por desvantagem a exigência de um número variável de iterações (por isso o nome "cíclico") para chegar a um resultado satisfatório - o que torna seus tempos de execução inconsistentes - e não é incomum que leve a um posicionamento pouco natural das juntas intermediárias da cadeia cinemática (MULLER-CAJAR; MUKUNDAN, 2007)(ARISTIDOU; LASENBY, 2011) - o que sobre a ótica de alcançar o ponto desejado não necessariamente configura um problema, mas é associado a um 'custo' maior de deslocamento das juntas envolvidas, e isso torna o movimento mais longo. Uma outra alternativa popular é o chamado FABRIK (*Forward And Backward Reaching Inverse Kinematics*), proposto em 2011 por Aristidou e Lasenby (ARISTIDOU; LASENBY, 2011), que oferece proposta similar de movimentação a baixo custo computacional ainda exigindo menos iterações e chegando a resultados mais naturais de posição que o CCD.

No entanto, mesmo o FABRIK exige algumas iterações (segundo Aristidou e Lasenby, em média quinze) para ser calculado, e considerando o poder computacional limitado do Raspberry Pi Pico (microcontrolador do PegasoV3) frente a robôs industrias reais ou mesmo a computadores pessoais modernos, entende-se que uma alternativa que una a simplicidade de cálculo a uma contagem baixa, ou de preferência única, de iterações pode ser mais adequada para a restrição computacional. Neste contexto há um candidato promissor para a tarefa chamado Método da Triangulação, proposto por Cajar e Mukundan em 2007 (MULLER-CAJAR; MUKUNDAN, 2007).

Neste método a ideia é tecer a análise partindo da base da cadeia cinemática em direção ao efetuador, percorrendo cada junta intermediária e formando um triângulo que a conecta com o efetuador terminal e o alvo desejado (ver figura 56):

- O lado **a** do triângulo é o comprimento do link entre a junta atual e a próxima;

- o lado **b** é a distância em linha reta entre a próxima junta da cadeia e o efetuador terminal;
- o lado **c** é a distância entre a junta atual e o alvo.

Como a distância **b** é sempre menor ou igual ao tamanho da cadeia cinemática restante, infere-se que se o triângulo puder ser formado, o alvo poderá ser atingido. Com uma formação bem-sucedida, o método aplica a lei dos cossenos em Δ**abc** e encontra o ângulo δ da figura 56. Adicionalmente, formam-se também dois vetores:

- o vetor $\vec{a}$ parte da junta atual e aponta para a próxima junta da cadeia; e
- o vetor $\vec{c}$ parte da junta atual e aponta para o alvo que deseja-se atingir.

O produto escalar destes vetores resulta no ângulo que a junta originalmente se encontrava em relação ao alvo. Se deste ângulo subtrairmos o δ recém-descoberto, teremos θ - que corresponde ao valor que a junta em análise deve girar para o ponto desejado ser atingido.

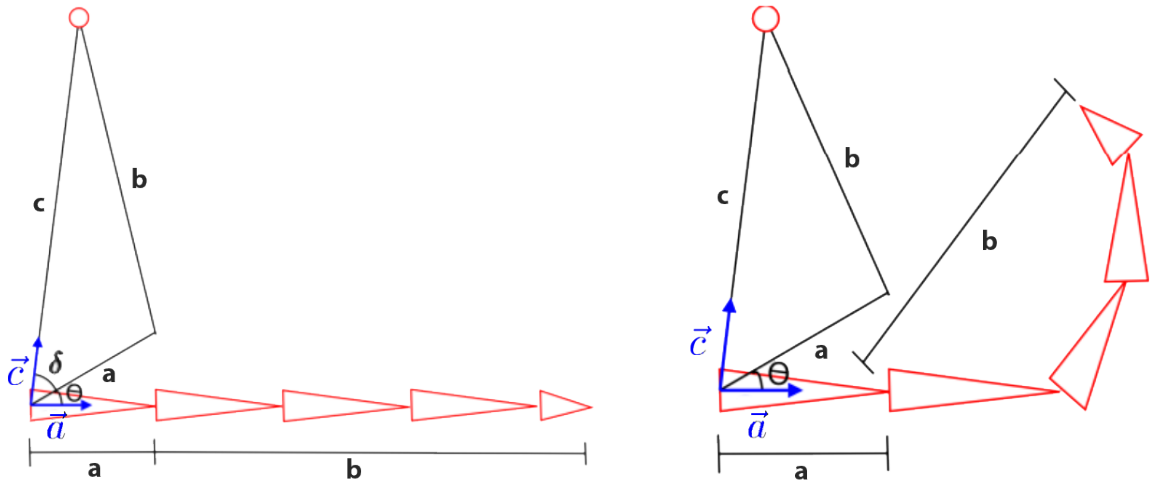


Figura 56 – Algoritmo de triangulação.(MULLER-CAJAR; MUKUNDAN, 2007)

Analisando os dois exemplos ilustrados na figura 56 e a natureza do algoritmo descrita acima, é perceptível que como a cadeia cinemática restante já possui comprimento equivalente ao lado **b**, o próximo triângulo teria $\mathbf{a} + \mathbf{b} = \mathbf{c}$, e portanto, $\delta = 0$. Na prática, isso indica que o cálculo do ângulo da próxima junta seria o último necessário e poderia já começar no produto escalar entre os novos vetores $\vec{a}$ e $\vec{c}$ formados. Ou seja, o método da Triangulação sequer precisa percorrer toda a cadeia cinemática, podendo interromper o processo se observar que já antes do ciclo completo de cálculos o alvo pode ser atingido.

Mais importante ainda, mesmo que uma passagem por todas as juntas se faça necessária, ela só precisará ocorrer uma vez - visto que o ângulo θ descoberto para cada uma

já é o que se objetivava para atingir o alvo. Assim, tendo cumprido com os requisitos de simplicidade de cálculo e exigindo somente uma iteração para obter o valor das juntas, o método da Triangulação de Cajar e Mukundan mostra-se muito interessante para o contexto e será usado como base para a elaboração da cinemática inversa do robô.

6.3 Adaptação do Método da Triangulação para o PegasoV3

Apesar de promissor, o algortimo proposto por Cajar e Mukundan é descrito sem envolver limitações no movimento das juntas, admitindo que todas elas podem rotacionar livremente para qualquer direção e reservando a adoção destas restrições a trabalhos futuros. Devido ao fato de o PegasoV3 possuir tais limites de giro em suas juntas, característica esta comum aos robôs articulados, é proposto o desenvolvimento de uma adaptação do método para contemplar o robô deste Trabalho. Se ainda assim for possível preservar ao máximo a liberdade de atuação das juntas, o algoritmo tenderá a manter sua simplicidade e essência.

6.3.1 Versão 1

Inicia-se a adaptação do método simplificando o modelo cinemático do robô para um onde não existam as juntas base, C e E. Se esta restrição for imposta, então as três restantes - A, B e D, em um esquema similar ao mostrado na figura 57 - ficarão com eixos de rotação paralelos, o que as leva a compartilharem o mesmo plano de atuação e consequentemente limita a análise a um problema 2D. No entanto, é possível tridimensionalizar a atuação do robô simplesmente devolvendo a junta base à cadeia cinemática, o que efetivamente permite que o referido plano possa girar em torno dela e assim contemplar qualquer ponto do volume de trabalho.

O arranjo descrito acima, que especifica a localização de um ponto cartesiano em termos de um plano 2D e um ângulo, é conhecido na matemática como sistema de coordenadas cilíndicas e pode ser ilustrado pela figura 58. Este sistema utiliza também três parâmetros, [u, v, θ], que se relacionam com as coordenadas [x,y,z] da representação cartesiana atrávés das seguintes equações:

- $u = \sqrt{x^2 + y^2}$
- $v = z$
- $\theta = tan^{-1}(y/x)$

Com estas informações em mãos pode-se criar uma primeira versão do algoritmo, mostrado a seguir, que utiliza o princípio da Triangulação após feita a transposição entre sistemas de coordenadas do ponto desejado.

Figura 57 – Cadeia cinemática restrita às juntas A, B e D (eixos de rotação paralelos).

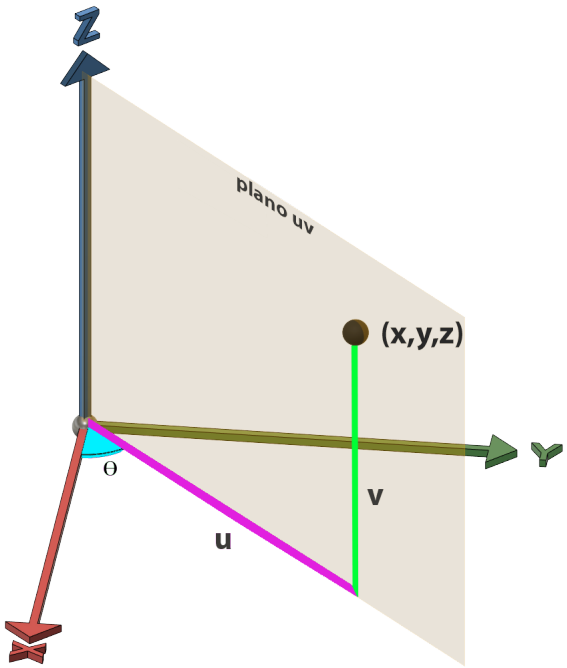


Figura 58 – Sistema de coordenadas cilíndrico.

```
    [VERSAO 1] - ALGORITMO ADAPTADO DE CINEMATICA INVERSA

    1. calcular [u,v, theta] a partir de [x,y,z]
        angbase <- theta

    2. calcular vetores vet_a e vet_c, partindo da junta A

    3. calcular o angulo da junta A pelo metodo da Triangulacao:
```

```
        triangulo abc:
            lado a: distancia [u,v] entre juntas A e B
            lado b: distancia [u,v] entre juntas B e Efetuador
            lado c: distancia [u,v] entre juntas A e Efetuador

        angA <- leiDosCossenos(a,b,c) - produtoEscalar(vet_a,
           vet_c)

    4. utilizar triangulo formado, trocando os lados b e c de
       lugar,
    para descobrir o angulo no vertice da junta B:
        angB <- leiDosCossenos(a,c,b) - 180

    5. retornar [angbase, angA, angB]
```

Percebe-se que não há necessidade de calcular o ângulo da junta D porque, conforme descrito anteriormente, o triângulo **Δabc** formado no passo 3 já possui dois de seus lados com comprimentos iguais a links da cadeia cinemática (o lado **a** corresponde ao comprimento do Link A, enquanto o lado **b** corresponde ao comprimento dos Links B + C + D). Ou seja, os ângulos dele já estão relacionados aos ângulos finais que se desejam para as juntas. O passo 4 fica sendo, portanto, calcular outro ângulo interno do triângulo - o que fica no vértice correspondente ao Link B. Como a referência da junta B é o link A, deve-se subtrair 180° do segundo ângulo para que o deslocamento de B seja obtido.

A seguir serão dados dois exemplos práticos de aplicação do algortimo. Deve-se assumir para fins de cálculos que, uma vez atingindo o plano *uv*, o robô sempre estará inicialmente com suas juntas nas coordenadas [u,v] descritas na figura 59, que correspondem à posição HOME do robô desconsiderado o ângulo da base.

```
    EXEMPLO 1 - Coordenada [x,y,z] = [100, 200, 300]

    1. Passando para o sistema de coordenadas cilindricas:
        u <- 223.6
        v <- 300
        theta <- 63.43

    2.  vet_a <- [0, 409.41 - 182.71] --> vet_a = [0, 226.7]
        vet_c <- [223.6 - 0, 300 - 182.71] --> vet_c = [223.6,
           117.29]

    3. triangulo abc:
        a <- distancia [juntas A -> B] = 226.7
        b <- distancia [juntas B -> Efetuador] = 202.13
        c <- distancia [junta A -> alvo] = 252.49

        leiDosCossenos(a,b,c) <- 49.54
        produtoEscalar(vet_A, vet_C) <- 62.32
```

```
18
19          angA <- 49.54 - 62.32 = -12.78
20
21      4.  leiDosCossenos(a,c,b) <- 71.88
22          angB <- 71.88 - 180 = -108.12
23
24      5.  [angbase, angA, angB] <- [63.43, -12.78, -108.12]
25
26
27
28      EXEMPLO 2 - Coordenada [x,y,z] = [0, 120, 250]
29
30      1.  u <- 120
31          v <- 250
32          theta <- 90
33
34      2.  vet_a <- [0, 409.41 - 182.71] --> vet_a = [0, 226.7]
35          vet_c <- [120 - 0, 250 - 182.71] --> vet_c = [120, 67.29]
36
37      3. triangulo abc:
38          a <- distancia [juntas A -> B] = 226.7
39          b <- distancia [juntas B -> Efetuador] = 202.13
40          c <- distancia [junta A -> alvo] = 137.58
41
42          leiDosCossenos(a,b,c) <- 61.81
43          produtoEscalar(vet_A, vet_C) <- 60.72
44
45          angA <- 1.09
46
47      4.  angB <- -143.13
48
49      5.  [angbase, angA, angB] <- [90, 1.09, -143.13]
```

A figura 60 ilustra a cadeia cinemática posicionada conforme os dois exemplos demonstrados. O alvo aparece como uma esfera colorida em azul, e como é possível observar, está em contato com o efetuador terminal nos dois casos; ele foi portanto capaz de atingir a posição. No entanto, conforme explicado no início desta seção, além de atingir a posição desejada também é de interesse que possa-se decidir com qual orientação fazer isto. Assim, é necessária a progressão do algoritmo para uma nova versão capaz de lidar com os requisitos adicionais.

6.3.2 Versão 2

Para poder trabalhar com o controle de orientação do efetuador terminal, deve-se primeiramente inserir nele um sistema de coordenadas cartesianas próprio - que aqui será chamado [Ex,Ey,Ez], conforme ilustra a figura 61. Ao contrário da origem, tal sistema é móvel - e é justamente na comparação entre os dois que será determinada sua orientação. Os

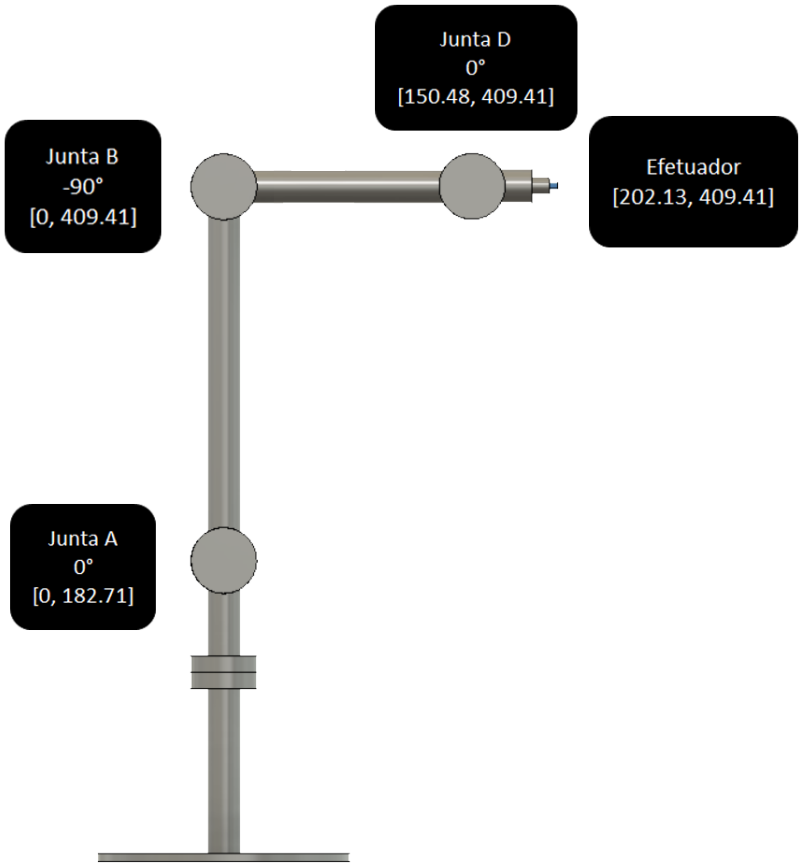


Figura 59 – Coordenadas iniciais das juntas no plano *uv*.

parâmetos responsáveis por promover alterações na inclinação do efetuador são as rotações em torno dos eixos ortogonais do sistema recém-criado, cuja nomenclatura adotada na literatura é a de *Roll*, *Pitch* e *Yaw*.

- Roll: rotação do efetuador terminal em torno de Ex.
- Pitch: rotação do efetuador terminal em torno de Ey.
- Yaw: rotação do efetuador terminal em torno de Ez.

Conforme ilustra novamente a figura 61. Agora, além de especificar o [x,y,z] que o robô deve atingir, o usuário também poderá definir uma orientação [roll,pitch,yaw] com que quer que a aproximação aconteça. O método de triangulação original proposto por Cajar e Mukundan não envolve este tipo de especificação, mas é possível aproveitar a essência do raciocínio apresentado por eles no desenvolvimento de uma versão que inclua a orientação do efetuador.

Com a inclusão dos ângulos de inclinação do efetuador é perceptível que a definição do plano *uv* deverá ser reformulada, pois desta vez as juntas C e E terão de ser inclusas na conta - e não mais será possível inserir o corpo do robô inteiramente em um plano

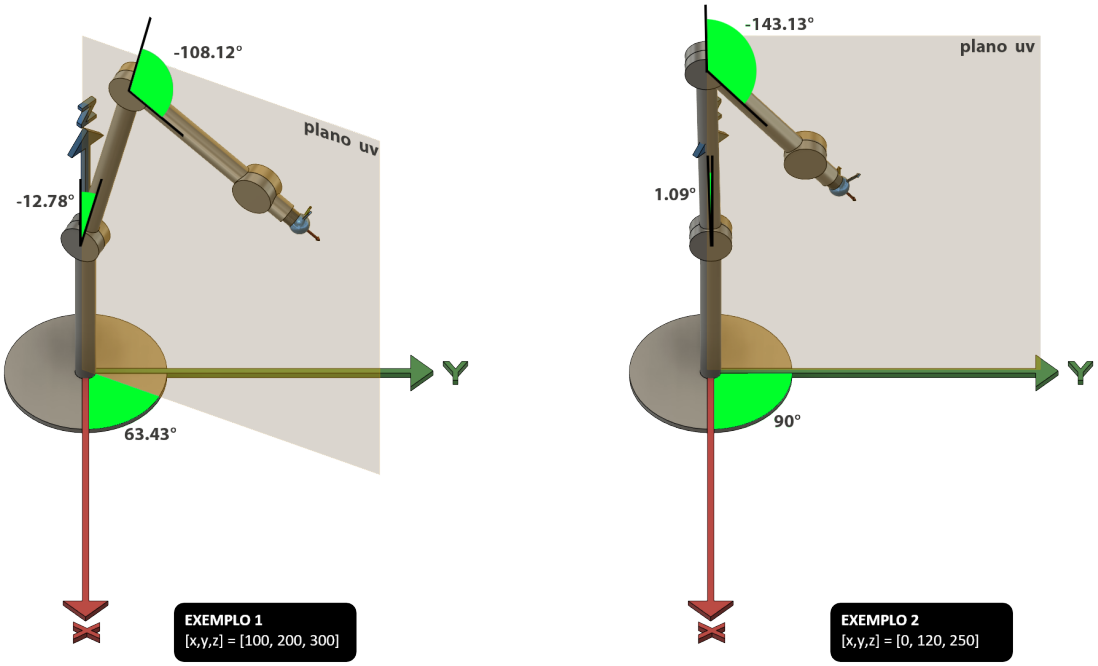


Figura 60 – Posicionamento da cadeia cinemática para a versão 1 do algoritmo.

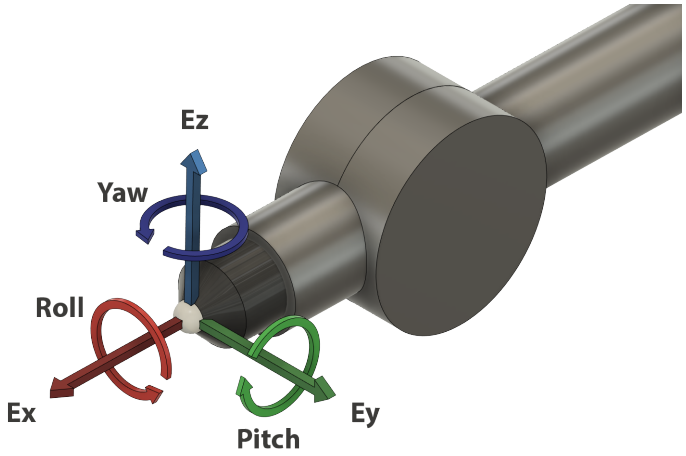


Figura 61 – Sistema cartesiano do efetuador.

2D, conforme feito na versão 1. Em vez disso, a combinação de ângulos das juntas C e D demonstra capacidade de deslocar o efetuador terminal e invalidar a análise bidimensional como feita até então. A figura 62 ilustra o caso: em um cenário onde o robô deve se posicionar no alvo com algum nível de Yaw, o plano definido pelas juntas A, B e D (usado na versão 1 como plano *uv*) difere do que passa pela junta E.

É nítido que a junta D permanece dentro do plano ABD em todos os momentos, pois além de ser um dos componentes óbvios deste plano, o único impacto que a junta C tem sobre ela é girá-la no lugar. Sendo isto verdade, se a análise da cinemática inversa puder ser movida de alguma forma para a localização [x,y,z] da junta D ao invés do efetuador terminal, então todas as juntas remanescentes da cadeia - base, A, B e C - continuam pertencendo ao mesmo plano, e isto mantém viva boa parte da filosofia do primeiro algoritmo.

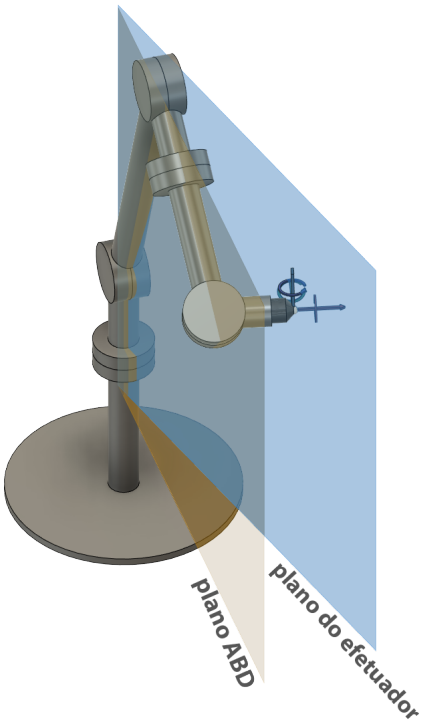


Figura 62 – Diferença de planos causada por Yaw.

É possível obter a localização cartesiana da junta D a partir do conjunto [x,y,z] e [Roll, Pitch, Yaw] dado para o efetuador, pois conhece-se a distância entre os dois pontos (sempre igual ao comprimento L da figura 52) e podem ser usados os próprios parâmetros de orientação que o usuário forneceu para descobrir onde uma junta vai estar em relação a outra. É interessante observar que destes três parâmetros é preciso se preocupar apenas com dois - o Pitch e o Yaw - visto que como o eixo Ex é concêntrico com o link D, o parâmetro Roll não tem poder de mudar suas coordenadas cartesianas - e além disso, movimentações realizadas por ele já são plenamente atendidas pela junta E, discutida mais adiante.

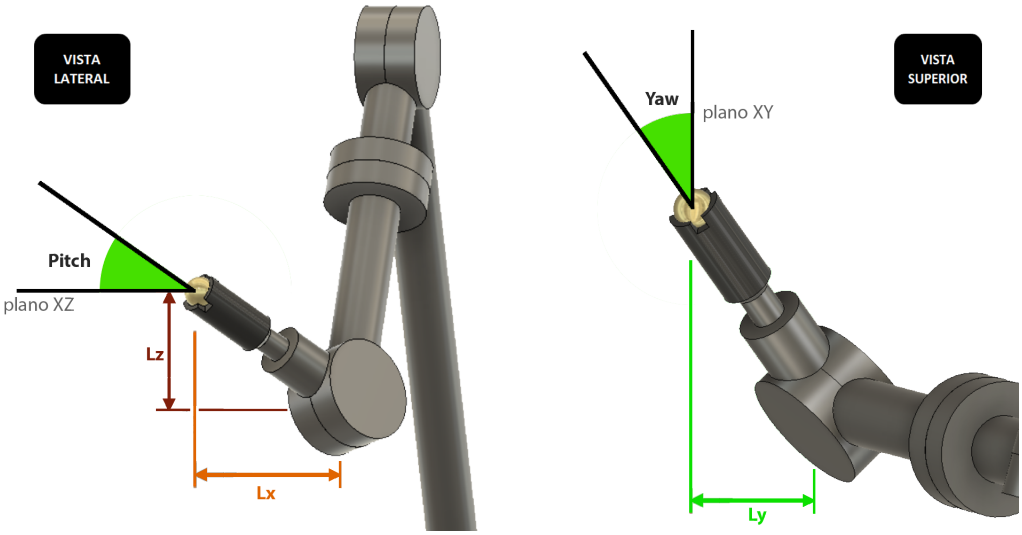


Figura 63 – Lx, Ly e Lz.

Feitas estas considerações, a estratégia formulada para obter as coordenadas da junta D a partir do efetuador foi decompor o comprimento L que separa os dois elementos da cadeia (figura 52) em um conjunto [Lx,Ly,Lz], tal qual observável na figura 63. Este conjunto equivale às coordenadas vetoriais de L, se relacionando com os ângulos de Pitch e Yaw fornecidos pelo usuário da seguinte forma:

- $Lx = L * cos(Pitch)cos(Yaw)$
- $Ly = L * cos(Pitch)sen(Yaw)$
- $Lz = L * sen(Pitch)$

Lx, Ly, e Lz são, portanto, *offsets* que a associação de L com os ângulos de orientação causam na posição original do efetuador, coincidindo com o ponto em que a junta D deve permanecer para atender à especificação pedida. Sendo assim, as coordenadas da junta D podem finalmente ser calculadas:

- $D[x] = Efetuador[x] - Lx$
- $D[y] = Efetuador[y] - Ly$
- $D[z] = Efetuador[z] - Lz$

Tendo esta informação, pode-se usar o Método da Triangulação assumindo que o alvo está em D[x,y,z] e que a cadeia cinemática termina na junta D. Fazer desta maneira leva a uma solução para os ângulos das juntas base, A e B aos moldes da versão 1, mas o valor da própria D não poderá ser descoberto com esse procedimento justamente porque nesta abstração ela é a ponta da cadeia, não um elemento intermediário.

O ângulo D poderá ser calculado em seguida, onde um novo uso do método proposto por Cajar e Mukundan permite formar um triângulo com o próximo conjunto de juntas da cadeia cinemática, tal qual ilustrado pelo triângulo direito na figura 64. Para que isto possa ser feito, no entanto, precisa-se saber a nova posição ocupada pela junta B, que teve sua localização mudada com a triangulação anterior. Isso implica que no momento de transição entre a primeira etapa do cálculo (triângulo esquerdo) e a segunda (triângulo direito) deverá haver um *update* nas coordenadas cartesianas desta junta, tarefa resolvível com trigonometria simples. É importante observar também que a junta C **não** é relevante para descobrir o ângulo D porque ela não é capaz de alterar o comprimento de nenhuma das arestas do triângulo direito na figura 64, que serão os elementos usados para a aplicação da lei dos cossenos.

Se com as duas etapas de triangulação descritas e ilustradas descobre-se o ângulo da junta D, resta saber como fazê-la apontar para o alvo desejado. Para isso conta-se com a junta C, que por sua vez pode ter seu deslocamento calculado com auxílio de dois novos vetores. Eles entram nesta que pode ser vista como a terceira etapa do novo algoritmo:

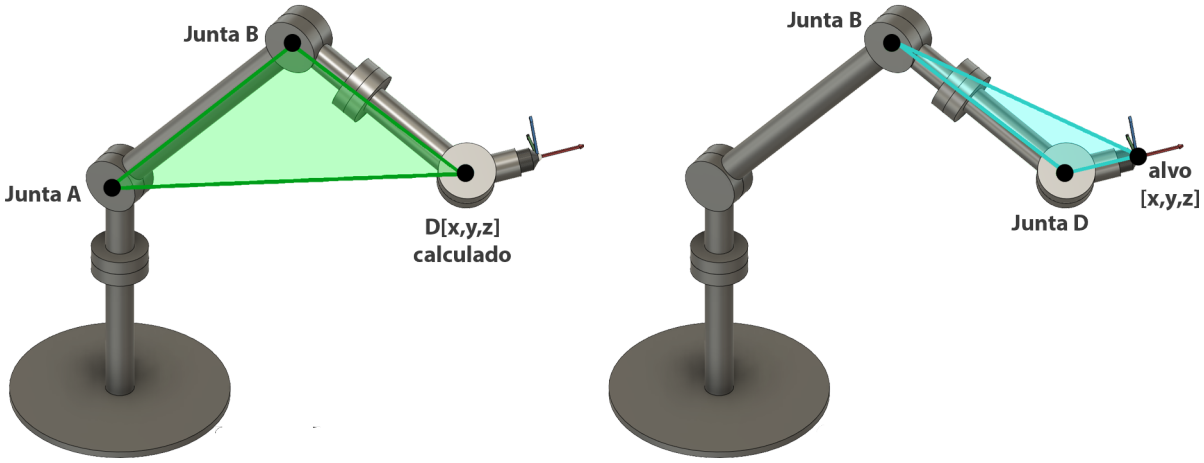


Figura 64 – Etapas de triangulação.

- Bp: vetor fixo de referência, perpendicular a ambas as juntas B e C; e
- Dp: vetor móvel auxiliar, que sai perpendicular do prolongamento do link C e aponta em direção ao alvo.

Uma visualização de Bp e Dp pode ser conferida na figura 65. É possível observar que enquanto o produto *escalar* dos dois vetores fornece o ângulo em módulo que a junta C deve girar, o produto *vetorial* complementa a informação fornecendo o sentido. Explica-se: o resultado da segunda operação será um novo vetor, ortogonal a Bp e Dp e colinear com o link C. A depender da posição de Dp em relação a Bp, tal vetor resultante irá apontar ou em direção à junta B ou no sentido contrário a ela.

Logo, basta verificar o sentido do vetor resultante que se obtém para que lado a junta C deve girar. O módulo, como já dito, é obtido com o produto escalar dos mesmos vetores. Uma vez compiladas estas e as demais informações fornecidas ao longo dos parágrafos anteriores, a versão 2 do algoritmo foi enfim elaborada e pode ser acompanhada a seguir.

```
1
2     [VERSAO 2] ALGORITMO ADAPTADO DE CINEMATICA INVERSA
3
4     1. calcular D[x,y,z] a partir de [x,y,z][pitch,yaw]:
5         L <- distancia entre juntas D e E = link D
6         D[x] <- [x] - L*cos(pitch)cos(yaw)
7         D[y] <- [y] - L*cos(pitch)sen(yaw)
8         D[z] <- [z] - L*sen(pitch)
9
10    2. calcular [u,v,theta] a partir de D[x,y,z]
11        D[u,v] <- [u,v]
12        angbase <- theta
13
14    3. calcular vetores vet_a e vet_c, partindo da junta A
15
16    4. calcular o angulo da junta A pelo metodo da Triangulacao:
17        triangulo abc:
18            lado a: distancia entre juntas A e B
19            lado b: distancia entre juntas B e D
20            lado c: distancia [u,v] entre juntas A e D[u,v]
21
22        angA <- leiDosCossenos(a,b,c) - produtoEscalar(vet_a,
              vet_c)
23
24    5. utilizar triangulo formado, trocando os lados b e c de
          lugar,
25    para descobrir o angulo no vertice da junta B:
26        angB <- leiDosCossenos(a,c,b) - 180
27
28    6. atualizar posicao [x,y,z] da junta B
29
30    7. calcular o angulo da junta D pelo metodo da Triangulacao:
31        triangulo abc:
32            lado a: distancia entre juntas B e D
33            lado b: distancia entre juntas B e [x,y,z] original
34            lado c: comprimento L
35
36        angD <- 180 - leiDosCossenos(a,b,c)
37
38    8. calcular o angulo da junta C:
39        calcular vetor Bp
40        calcular vetor Dp
41
42        vtr <- produtoVetorial(Dp,Bp)
43        se vtr aponta para junta B:
44            angC <- produtoEscalar(Dp,Bp)
45        senao:
46            angC <- -produtoEscalar(Dp,Bp)
47
48    9. retornar [angbase, angA, angB, angC, angD]
```

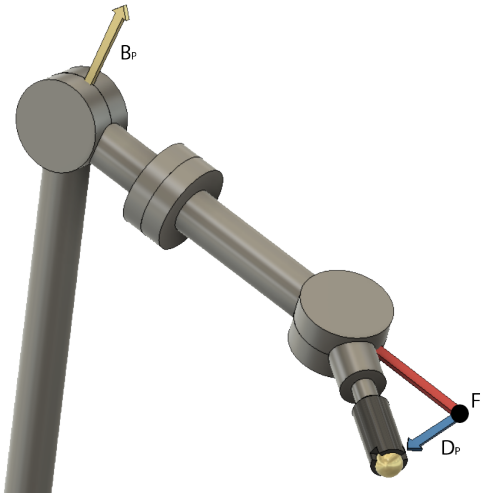


Figura 65 – Vetores Bp e Dp.

Para ilustrar o novo algoritmo são fornecidos a seguir mais dois exemplos, onde o foco é chegar aos mesmos pontos de antes porém adicionando os parâmetros de orientação. É interessante ressaltar que mesmo que eles apareçam nulos, os ângulos fornecidos pela versão 2 serão diferentes da versão 1. Isto ocorre porque zerar o trio [roll,pitch,yaw] não significa, aos olhos do algoritmo, ignorar a existência deles e considerar como se o objetivo na verdade fosse somente chegar ao ponto [x,y,z]. Em vez disso, o valor zero indica apenas que não há diferença entre a orientação do efetuador e a da origem do sistema. O resultado esperado para [roll, pitch,yaw] = [0,0,0] portanto é que o efetuador se mantenha com seu sistema [Ex,Ey,Ez] alinhado integralmente com os eixos ortogonais X, Y e Z.

```
1
2     EXEMPLO 3
3     Coordenada [x,y,z] [roll,pitch,yaw] = [100,200,300] [0,0,0]
4
5     1.  L <- 51.65 (distancia entre D e E)
6           Lx <- 51.65 * cos(0)sen(0) = 51.65    --> D[x] = 48.35
7           Ly <- 51.65 * cos(0)sen(0) = 0        --> D[y] = 200
8           Lz <- 51.65 * sen(0) = 0              --> D[z] = 300
9
10    2.  D[x,y,z] <- [48.35, 200, 300]
11        D[u] <- 205.76
12        D[v] <- 300
13        theta <- 76.50
14
15    3.  vet_a <- [0, 409.41 - 182.71] -> vet_a = [0, 226.7]
16        vet_c <- [205.76 - 0, 300 - 182.71] -> vet_c = [205.76,
              117.29]
17
18    4.  triangulo abc:
19        a <- distancia [juntas A -> B] = 226.7
20        b <- distancia [juntas B -> Efetuador] = 202.13
21        c <- distancia [junta A -> alvo] = 236.84
```

```
22
23          leiDosCossenos(a,b,c) <- 37.81
24          produtoEscalar(vet_A, vet_C) <- 60.32
25
26          angA <- 37.81 - 60.32 = -22.51
27
28      5.  leiDosCossenos(a,c,b) <- 74.75
29          angB <- 74.75 - 180 <- -105.25
30
31      6.  novo B[u,v] apos deslocamento da junta A <- [86.79, 392.14]
32          convertendo para sistema [x,y,z]:
33          B[x] <- B[u]*cos(theta) = 20.39
34          B[y] <- B[u]*cos(theta) = 84.36
35          B[z] <- B[v] = 392.14
36
37      7.  novo triangulo abc:
38          a <- distancia [juntas B -> D] = 150.48
39          b <- distancia [junta B -> alvo original] = 167.93
40          c <- L = 51.65
41
42          angD <- 180 - leiDosCossenos(a,b,c) = 180 - 100.71 = 79.29
43
44
45      8.  angA + angB + 90 <- -37.76
46          Bp[u,v] <- [-sen(-37.76), cos(-37.76)] = [0.6123, 0.7906]
47          logo --> Bp[x,y,z] <- [0.14, 0.59, 0.79]
48
49          Descobrindo origem do vetor Dp (ver fig. 6.14):
50          F[u,v] <- D[u,v] + [-L*sen(-37.76), L*cos(-37.76)]
51          F[x,y,z] <- [50.13, 207.37, 294.12]
52
53          Dp[x,y,z] <- alvo - F[x,y,z] = [49.87, -7.37, -5.88]
54
55          vtr <- produtoVetorial(Dp, Bp) = [-9.33, -38.58, 30.74]
56
57          saindo de D[x,y,z], vtr aponta para [39.02, 161.42, 269.26]
58              este ponto fica a 146.24mm da junta B.
59              D[x,y,z] fica a 150.48mm da junta B.
60              logo, vtr aponta para a junta B.
61
62          angC <- produtoEscalar(Dp, Bp) = 81.58
63
64      9.  versao 1:
65          [angbase, angA, angB]
66          [63.43, -12.78, -108.12]
67
68          versao 2:
69          [angbase, angA, angB, angC, angD]
70          [76.50, -22.51, -105.25, 81.58, 79.29]
71
72
73
```

```
    EXEMPLO 4
    Coordenada [x,y,z] [roll,pitch,yaw] = [0,120,250] [0,0,0]

    1.  [Lx, Ly, Lz] <- [51.65, 0, 0]
        D[x,y,z] <- [-51.65, 120, 250]

    2.  D[u,v] <- [130.64, 250]
        theta <- 113.28

    3.  vet_a <- [0, 409.41 - 182.71] = [0, 226.7]
        vet_c <- [130.64 - 0, 250 - 182.71] = [130.64, 67.29]

    4.  triangulo abc:
        a <- distancia [juntas A -> B] = 226.7
        b <- distancia [juntas B -> Efetuador] = 202.13
        c <- distancia [junta A -> alvo] = 146.95

        angA <- -21.83

    5.  angB <- leiDosCossenos(a,c,b) - 180 = -140.23

    6.  novo B[u,v] apos deslocamento da junta A <- [84.28, 393.16]
        convertendo para sistema [x,y,z]:
        B[x,y,z] <- [-33.32, 77.42, 393.16]

    7.  novo triangulo abc:
        a <- distancia [juntas B -> D] = 150.48
        b <- distancia [junta B -> alvo original] = 153.03
        c <- L = 51.65

        angD <- 180 - leiDosCossenos(a,b,c) = 97.00

    8.  Bp[x,y,z] <- [-0.37, 0.87, 0.31]
        Dp[x,y,z] <- [50.88, 1.78, -5.99]

        vtr <- produtoVetorial(Dp, Bp) = [5.78, -13.43, 45.13]

        vtr aponta para [-45.87, 106.57, 295.13]
            este ponto fica a 103.04mm da junta B.
            D[x,y,z] fica a 150.48mm da junta B.
            logo, vtr aponta para a junta B.

        angC <- produtoEscalar(Dp, Bp) = 112.27

    9.  versao 1:
        [angbase, angA, angB]
        [90, 1.09, -143.13]

        versao 2:
        [angbase, angA, angB, angC, angD]
        [113.29, -21.83, -140.23, 67.73, 97.00]
```

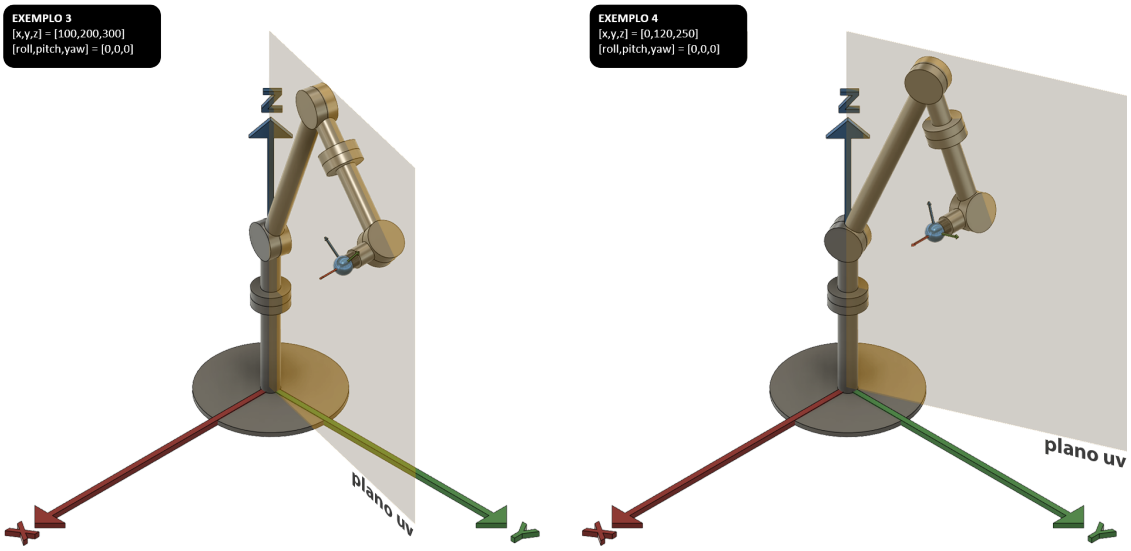


Figura 66 – Posicionamento da cadeia cinemática para a versão 2 do algoritmo.

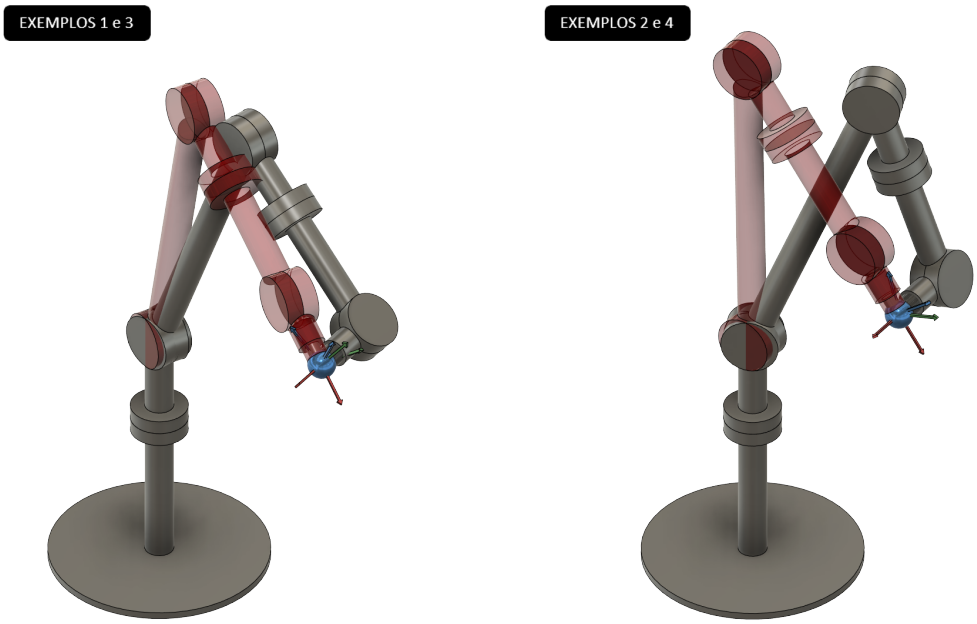


Figura 67 – Comparativo visual entre as versões 1 e 2 do algoritmo.

Para complementar os exemplos são mostradas duas figuras. A primeira delas, 66, ilustra o posicionamento da cadeia cinemática de acordo com os ângulos encontrados nos exemplos 3 e 4, ou seja, mostra o resultado da versão 2 do algoritmo para dois conjuntos [x,y,z] [roll,pitch,yaw] dados. Da mesma maneira que a figura dos exemplos 1 e 2 (60), o alvo aparece como uma esfera azul - e como pode ser observado, o robô novamente foi capaz de chegar a ele. Já a segunda figura, 67, sobrepõe os resultados dos exemplos 1, 2, 3 e 4 em seus respectivos pares para mostrar a diferença na forma como a cadeia cinemática atinge os pontos desejados. As cadeias dos exemplos 1 e 2 (versão 1 do algoritmo) são ilustradas em vermelho translúcido para fácil diferenciação.

6.3.3 Versão 3

Apesar de trazer consigo uma melhoria significativa na controlabilidade do robô, permitindo que além de determinar a posição [x,y,z] o usuário possa também controlar os parâmetros pitch e yaw, a segunda versão do algoritmo ainda não está completa por não levar em consideração o parâmetro de rolagem (roll). Na verdade, ela não envolve em nenhum momento a junta E - que gira o efetuador terminal - o que faz com que a orientação deste possivelmente fique prejudicada mesmo com o correto posicionamento dos outros 5 graus de liberdade do robô.

Isto ocorre porque, apesar de o posicionamento da cadeia cinemática ser calculado em relação a um ponto fixo (a origem do sistema cartesiano), o referencial da junta E é a junta D - o que significa que, em relação à origem, a orientação de E pode mudar mesmo que nenhuma especificação no parâmetro de rolagem tenha sido feita. A figura 68, que resgata o exemplo 3 dando enfoque no posicionamento do efetuador, ilustra esta característica: Mesmo que roll tenha sido determinado como valendo 0, o sistema [Ex,Ey,Ez] do efetuador está inclinado em relação à origem.

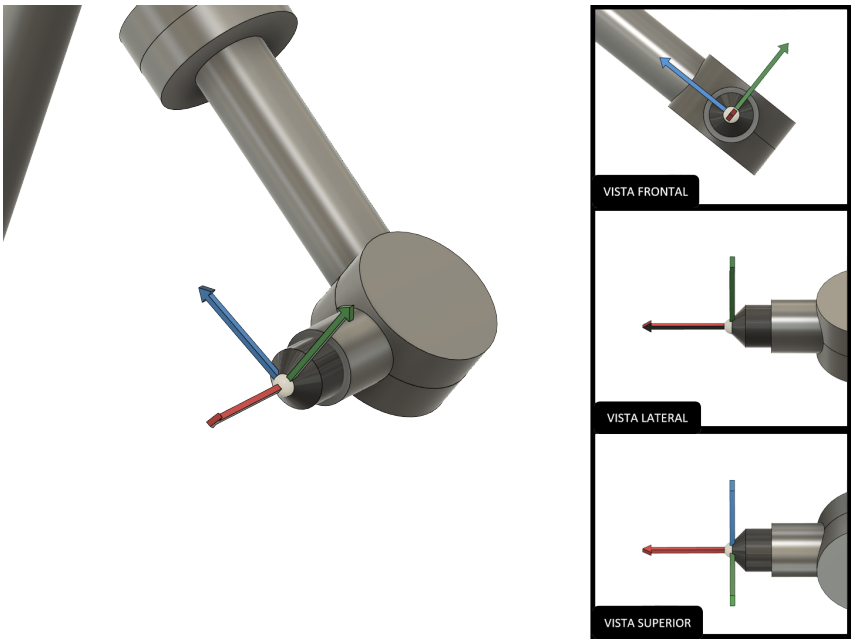


Figura 68 – Rolagem colateral do efetuador.

É razoável concluir, portanto, que para uma implementação bem-sucedida da cinemática inversa o referencial móvel do efetuador precisará receber algum tipo de compensação para manter-se fixo em relação à origem. Isto significa que ao final da versão 2 do algoritmo deve ser anexado um cálculo para a junta E semelhante ao seguinte formato:

- E = roll + Eoffset()

Onde *Eoffset()* analisa e corrige o ângulo colateral formado entre o referencial da

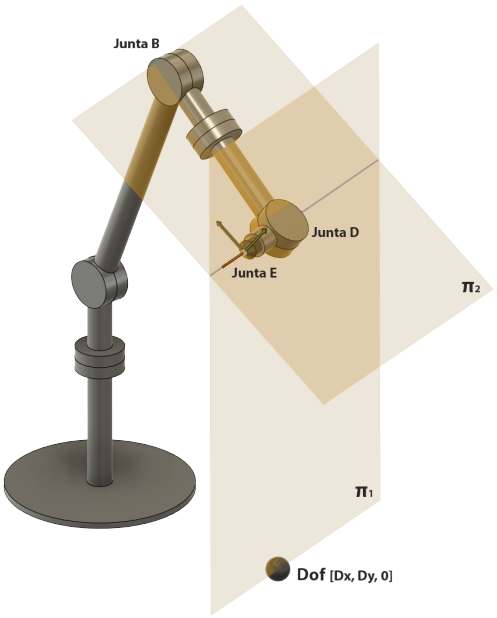


Figura 69 – planos π_1 e π_2.

junta E e a origem do sistema cartesiano. O cálculo deste pode ser feito criando dois novos planos e dois novos pontos específicos para a situação:

- Plano π_1: formado pelos pontos {junta D, junta E, Dof}
- Plano π_2: formado pelos pontos {junta D, junta E, junta B}

Conforme ilustrado na figura 69. Aqui, *Dof* equivale ao ponto D[x,y,z] com coordenada z = 0. Se adotada esta convenção, então π_1 representa um plano sempre verticalizado e π_2 representa um plano com inclinação proporcional ao tombamento do conjunto [Ex,Ey,Ez] que o efetuador sofre em relação à origem do sistema cartesiano. Além disso, vale ressaltar que devido ao fato das juntas D e E serem os pontos em comum entre os planos, o trecho da estrutura que as conecta (correspondente ao link D) incluirá Ex e será o eixo de rotação de ambos π_1 e π_2. O ângulo formado, portanto, será correspondente ao offset desejado, e pode ser obtido descobrindo-se os vetores normais aos planos e aplicando entre eles o produto escalar. Assim, o algoritmo para *Eoffset()* fica da seguinte forma:

```
1
2     *Lembrando que, ao final da versao 2 do algoritmo,
3     E[x,y,z] coincide com o alvo (ver fig. 2.64)*
4
5     Eoffset():
6     1.  calcular vetor normal ao plano 1:
7             vet_DE <- vetor ligando D[x,y,z] a E[x,y,z]
8             vet_DDof <- vetor ligando D[x,y,z] a Dof[x,y,z]
9
```

```
            vet_normal_p1 <- produtoVetorial(vet_DE, vet_DDof)

    2.  calcular vetor normal ao plano 2:
            vet_DE <- calculado acima
            vet_DB <- vetor ligando D[x,y,z] a B[x,y,z]

            vet_normal_p2 <- produtoVetorial(vet_DB, vet_DE)

    3.  descobrir se offset eh positivo ou negativo:
            vtr <- produtoVetorial(vet_normal_p1, vet_normal_p2)

            se vtr, saindo da junta E, aponta para a junta D:
                ang <- produtoEscalar(vet_normal_p1, vet_normal_p2)
            senao:
                ang <- -produtoEscalar(vet_normal_p1,
                   vet_normal_p2)

    4. retornar [ang]
```

É interessante observar que a mesma estratégia usada para descobrir o ângulo da junta C aparece no passo 3 para descobrir o offset da junta E: faz-se o produto vetorial dos vetores que compõem o ângulo e descobre-se para onde o resultado aponta, se para a junta antecessora na cadeia cinemática - no caso acima, a junta D - ou para fora dela. Feitas estas considerações, a versão 3 do algoritmo é a versão 2 com o acréscimo, ao final, do cálculo da rolagem da junta E compensada pelo *Eoffset()*. Isto faz com que agora seja possível determinar corretamente o giro do efetuador terminal, além de também viabilizar o controle do robô em todos os seus seis parâmetros de posicionamento - [x,y,z] [roll, pitch, yaw].

Partindo dos exemplos 3 e 4 iustrados na figura 66, tem-se que o acréscimo do cálculo para a junta E descrito acima fica da seguinte maneira:

```

    EXEMPLO 5
    ao final do exemplo 3 temos que:
        B[x,y,z] <- [20.39, 84.36, 392.14]
        D[x,y,z] <- [48.35, 200, 300]
        E[x,y,z] <- alvo = [100, 200, 300]

    1.  calcular vetor normal ao plano 1:
            vet_DE <- [100,200,300] - [48.35,200,300] = [51.65,0,0]
            vet_DDof <- [48.35,200,0] - [48.35,200,300] =
               [0,0,-300]

            produtoVetorial(vet_DE, vet_DDof) <- [0,15495,0]
            vet_normal_p1 <- [0, 15495, 0]
```

```
14
15      2.  calcular vetor normal ao plano 2:
16              vet_DE <- [51.65,0,0]
17              vet_DB <- [20.39,84.36,392.14] - [48.35,200,300]
18                     = [-27.96,-115.64,92.14]
19
20              produtoVetorial(vet_DB, vet_DE) <- [0,4759.03,5972.81]
21              vet_normal_p2 <- [0, 4759.03, 5972.81]
22
23      3.  descobrir se offset eh positivo ou negativo:
24
25              normalizando os vetores:
26              vet_normal_p1 <- [0, 1, 0]
27              vet_normal_p2 <- [0, 0.623, 0.782]
28
29              vtr <- produtoVetorial(vet_normal_p1, vet_normal_p2)
30              vtr <- [0.782, 0, 0]
31
32              Saindo de E[x,y,z], vtr aponta para [100.782, 200, 300]
33                  este ponto fica a 52.43mm da junta D.
34                  E[x,y,z] fia a 51.65mm da junta D.
35                  logo, vtr nao aponta para a junta D.
36
37              ang <- -produtoEscalar(vet_normal_p1, vet_normal_p2)
38                  = -51.457
39
40      4.  [ang] <- [-51.457]
41          angE <- roll - 51.547, mas roll = 0 --> angE <- -51.547
42
43          [angbase, angA, angB, angC, angD, angE]
44          [76.50, -22.51, -105.25, 81.58, 79.29, -51.547]
45
46
47
48      EXEMPLO 6
49      ao final do exemplo 4 temos que:
50          B[x,y,z] <- [-33.32, 77.42, 393.16]
51          D[x,y,z] <- [-51.65, 120, 250]
52          E[x,y,z] <- alvo = [0, 120, 250]
53
54      1.  calcular vetor normal ao plano 1:
55              vet_DE <- [51.65, 0, 0]
56              vet_DDof <- [0, 0, -250]
57
58              vet_normal_p1 <- [0, 12912.5, 0]
59
60      2.  calcular vetor normal ao plano 2:
61              vet_DE <- [51.65, 0, 0]
62              vet_DB <- [18.33, -42.58, 143.16]
63
64              vet_normal_p2 <- [0, 7394.21, 2199.26]
65
```

```
3.  descobrir se offset eh positivo ou negativo:

        normalizando os vetores:
        vet_normal_p1 <- [0, 1, 0]
        vet_normal_p2 <- [0, 0.96, 0.28]

        vtr <- produtoVetorial(vet_normal_p1, vet_normal_p2)
        vtr <- [0.28, 0, 0]

        Saindo de E[x,y,z], vtr aponta para [0.28, 120, 250]
            este ponto fica a 51.93mm da junta D.
            E[x,y,z] fia a 51.65mm da junta D.
            logo, vtr nao aponta para a junta D.

        ang <- -produtoEscalar(vet_normal_p1, vet_normal_p2)
            = -16.26

4.  [ang] <- [-16.26]
    angE <- roll - 16.26, mas roll = 0 --> angE <- -16.26

    [angbase, angA, angB, angC, angD, angE]
    [113.29, -21.83, -140.23, 67.73, 97.00, -16.26]
```

A figura 70 ilustra o posicionamento da cadeia cinemática para os exemplos 5 e 6. Como a versão 3 do algortimo aproveita o desenvolvimento e os resultados da versão 2, os ângulos das demais juntas da cadeia cinemática permanecem os mesmos de antes. Assim, o comparativo visual entre os resultados da versão 2 e 3, disponível na imagem 71, tem foco apenas no efetuador terminal, pois a junta E é a única que realmente muda. Conforme esperado, a correção feita para o referencial do efetuador levou-o a manter-se alinhado à origem do sistema.

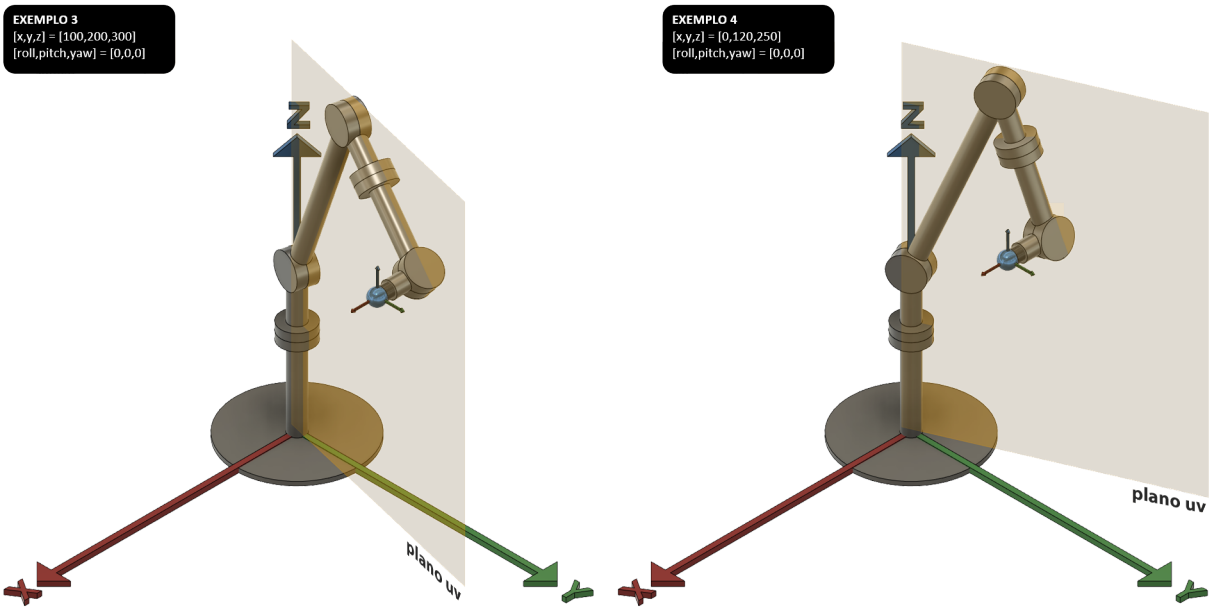


Figura 70 – Posicionamento da cadeia cinemática para a versão 3 do algoritmo.

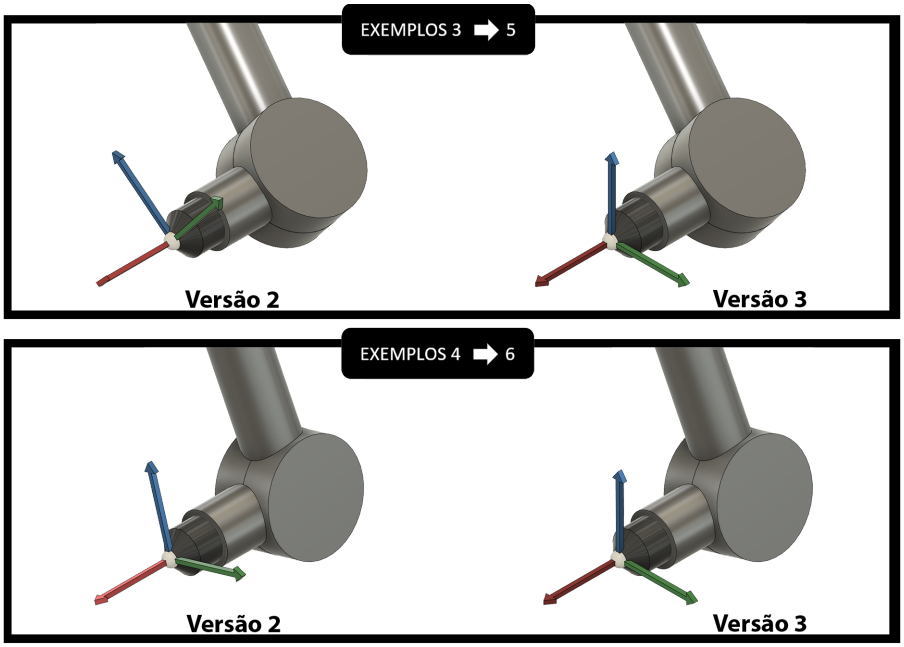


Figura 71 – comparativo visual entre as versões 2 e 3 do algoritmo.

6.4 Singularidades, Descontinuidades e Indefinições

Segundo Donelan (DONELAN, 2007), a singularidade de um manipulador robótico ocorre quando o número de graus de liberdade disponíveis de imediato ao efetuador terminal do robô difere do valor que se espera tendo como base o número de graus de liberdade conferidos pela cadeia cinemática. Sob uma perspectiva matemática, ocorre quando a matriz Jacobiana que descreve o robô passa a ter posto menor que o número de juntas disponíveis (no caso de um manipulador similar ao PegasoV3, 6). Na prática, é uma situação em que duas ou mais juntas do robô passam a conferir o mesmo tipo de movimento à cadeia cinemática, e que pode desencadear comportamento indesejado na movimentação do robô - seja através de uma mudança repentina na velocidade de suas juntas para manter um movimento programado, seja uma impossibilidade de o robô conseguir calcular sua trajetória apesar de parecer ser claramente possível que ele a faça.

Este tipo de situação é uma inconveniente realidade de algoritmos calculados com matrizes jacobianas, conforme apontado por Cajar e Mukundan (MULLER-CAJAR; MUKUNDAN, 2007) e Aristidou e Lasenby (ARISTIDOU; LASENBY, 2011), mas não representa uma preocupação real no caso do Método de Triangulação usado como base para a construção da cinemática inversa descrita nas páginas anteriores. Isto ocorre porque ele não utiliza matrizes Jacobianas em seus cálculos, e portanto não está sucetível às desvantagens que aparecem quando elas são empregadas. Além disto, conforme demonstrado por Cajar e Mukundan, se existe uma combinação de juntas que leva a cadeia cinemática a alcançar o alvo desejado, o método dá a certeza de encontrá-la.

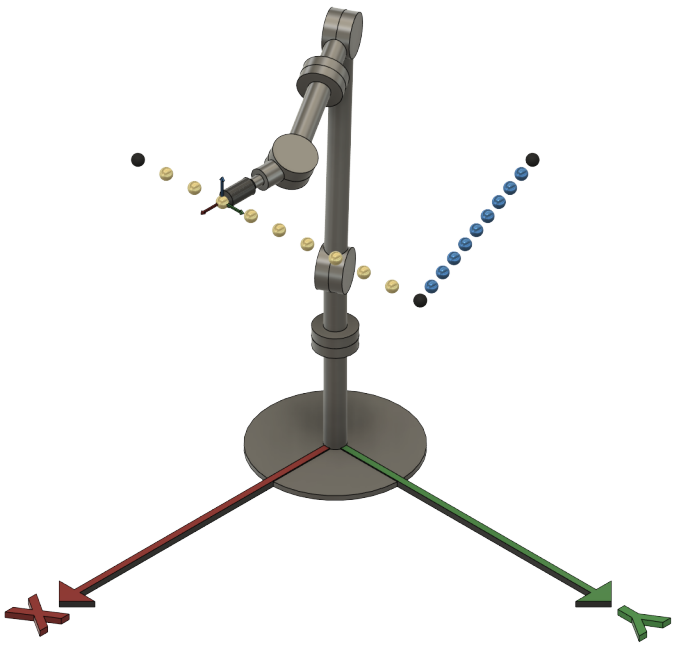


Figura 72 – Simulação de trajetória do robô.

A garantia de uma solução encontrada pelo algortimo, no entanto, não é certeza de que o movimento calculado pelo robô será contínuo. É possível que ele encontre uma solução para o alvo determinado que, apesar de atingível, o leve a executar uma trajetória com mudanças abruptas de ângulo das juntas. É importante ressaltar que apesar de o efeito ser similar ao possivelmente causado por uma singularidade, as causas são distintas - na singularidade, a matriz caminha para uma divisão por zero que indetermina o resultado do cálculo, enquanto na não-continuidade o resultado é sempre encontrado, só não necessariamente é o esperado.

Durante as simulações realizadas foram feitos vários testes similares ao ilustrado na figura 72, onde foram demarcados diversos pontos ao longo de retas e, para cada um deles, foi rodada a versão 3 do algoritmo de cinemática inversa (descrito em detalhes na seção anterior) desenvolvido para o PegasoV3 com base no método da Triangulação. O resultado obtido foi que, para trajetórias com pontos ao longo de um mesmo plano XY, a cinemática inversa foi de fato capaz de entregar valores próximo de ângulos das juntas para pontos vizinhos - o que faz sentido dada a maneira sistemática como o código funciona, recorrendo a funções contínuas em seu corpo para encontrar o conjunto de ângulos para cada ponto no espaço dado. Um vídeo específico da trajetória de duas retas ilustrada na figura 72 pode ser visto neste **link**.

Um problema de não-continuidade foi encontrado, no entanto, quando o movimento comandado envolvia transitar entre os semiplanos de coordenada Y positiva para negativa, ou vice-versa, sob certos conjuntos de ângulo *pitch* definidos pelo usuário. Quando este parâmetro era configurado para um valor negativo, diga-se -45°, e a posição [x,y,z] escolhida tinha altura o suficiente para deixar a junta D "apontando para baixo" em relação à junta C (ver figura 73), o resultado observado era que o algoritmo tendia a aumentar o *módulo* do ângulo de C conforme o robô se aproximava do plano XZ (y=0), mas realizando este aumento em sentidos contrários: com a trajetória composta por pontos com $y > 0$ a tendência observada era crescer rumo à marca de +180° (conforme ilustra a figura 74), e quando envolvia pontos com $y < 0$, a direção contrária era tomada, beirando os -180°.

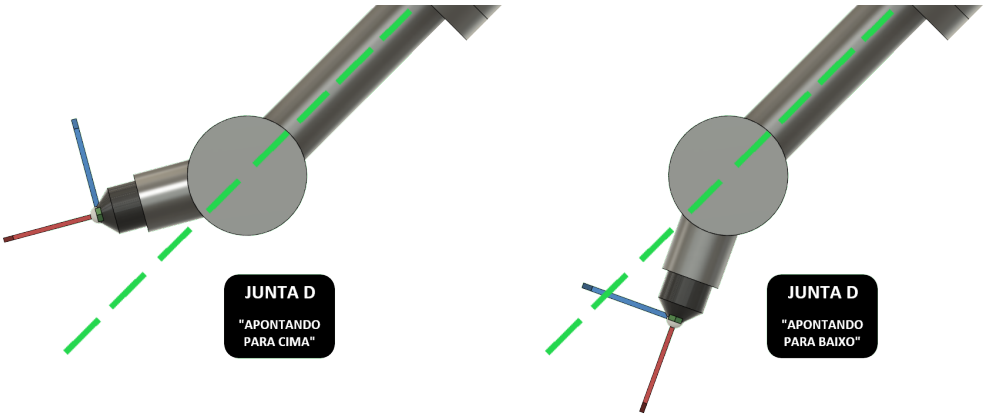


Figura 73 – Orientação da junta D em relação à junta C.

Isto levava a uma mudança brusca na orientação do robô porque, ao cruzar o plano XZ, imediatamente o valor da junta C passava de +180° para -180° ou vice-versa, causando a necessidade do robô girar cerca de 360° para adequar-se ao novo ângulo calculado e assim um desvio momentâneo da trajetória tornava-se perceptível. Procurou-se contornar este inconveniente com acréscimos ao algoritmo que espelhassem o valor das três juntas ao detectar a ocorrência destas condições, mas as medidas não obtiveram sucesso pleno na estratégia porque a resolução de uma não-continuidade em determinado local do volume de trabalho do robô levava ao surgimento de novas descontinuidades em outros locais, ou seja, o problema era apenas realocado. Consideou-se que, mesmo se a presença destes fenômenos não pudesse ser mitigada, o robô ainda manteria sua movimentação contínua em boa parte do seu volume de trabalho - bastando não precisar cruzar o plano XZ. Neste cenário, entendeu-se que por ora a movimentação do robô orientada a filmagens poderia ficar restrita aos semiespaços, seja y positivo ou negativo, e que novas tentativas de correção das descontinuidades seriam reservadas para trabalhos futuros.

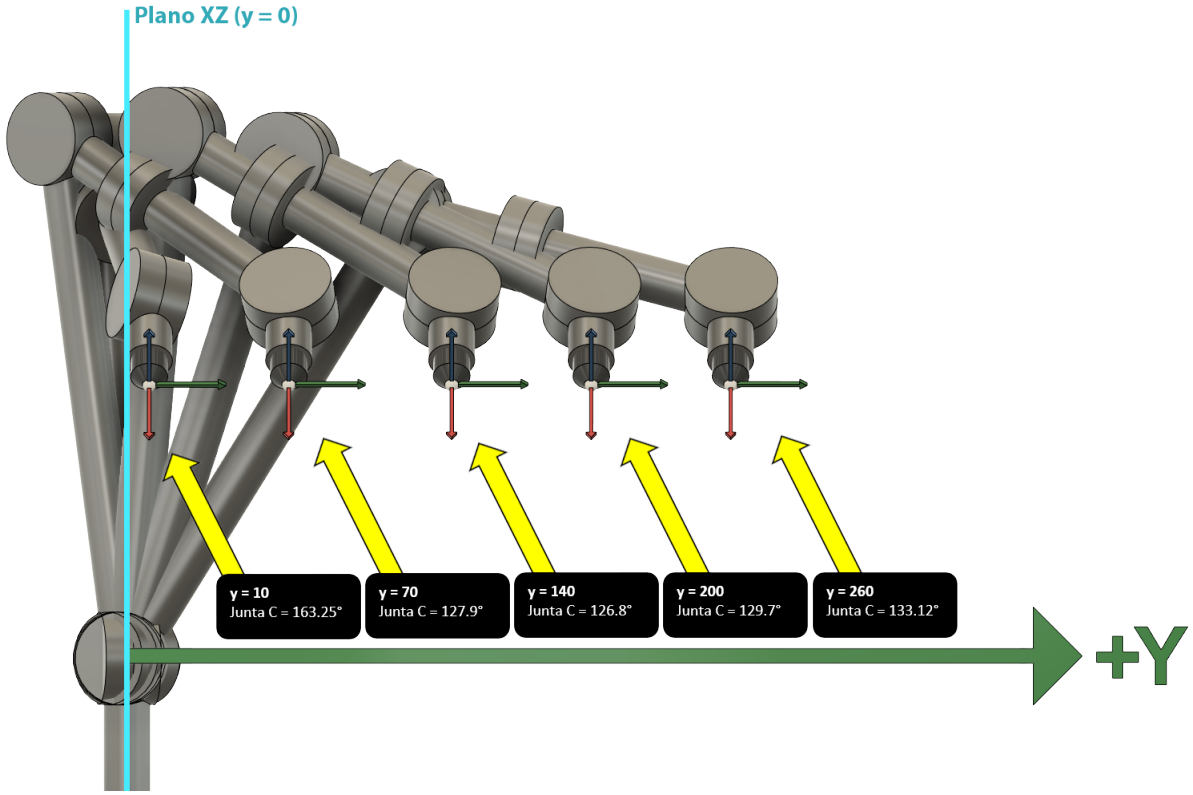


Figura 74 – Variação da junta C conforme o alvo se aproxima de y=0.

Por outro lado, um outro tipo de problema também intrínseco ao cálculo da cinemática inversa tal qual desenvolvida para o robô deste Trabalho pôde ser facilmente detectado e resolvido: o das indefinições de cálculo. A indefinição ocorre quando o método matemático utilizado para obter o resultado da conta não é mais capaz de retornar um valor válido para a mesma, podendo levar a infinitas soluções ou a nenhuma delas. Muito embora as singularidades descritas no início desta seção compartilhem da ocorrência de indeterminações,

elas têm a característica adicional de levar o robô a perder um ou mais graus de liberdade pelo alinhamento do eixo de atuação de suas juntas, sendo consideradas portanto um caso específico.

As duas indefinições que podem aparecer no cálculo da versão 3 do algoritmo são ligadas à obtenção dos ângulos das juntas C e E, especificamente porque ambas contam com a formação de vetores ou planos que eventualmente podem não ser formados. No caso da junta C, como é possivel relembrar através da figura 65, o vetor Dp depende que a junta D esteja com ângulo diferente de 0° para existir, já que seu módulo equivale a *LinkD*sen(angD)*. Se tal ângulo for zero, portanto, o vetor Bp será submetido a um produto vetorial com Dp = [0,0,0] e o resultado do cálculo será o próprio vetor nulo, que não possui atributos definidos de ângulo. Esta é uma indefinição.

Para o caso da junta E o cálculo envolve a formação de dois planos π_1 e π_2, conforme já explicado anteriormente e ilustrado na figura 69. Estes planos são definidos com três pontos, mas uma regra essencial que deve ser observada é que eles não podem os três fazerem parte de uma mesma reta. Se isto acontecer, a determinação do vetor normal ao plano será prejudicada porque terá como operandos dois vetores linearmente dependentes entre si. Uma rápida inspeção sobre os componentes de cada plano mostra que os pontos que formam π_1 ficarão colineares no caso de pitch em $\pm 90°$, e os que compõem π_2 o farão quando o ângulo da junta D for 0° (ou seja, é a mesma condição que compromete o cálculo da junta C). Nestas condições ocorre o mesmo problema de indefinição causada por multiplicação com vetores nulos, que não têm atributo de ângulo.

As estratégias definidas para contornar as indefinições são duas. A primeira, adotada para o caso de quando a junta D fica em 0°, é a de guardar em variáveis de histórico a última posição válida ocupada pelas juntas C e E. Toda vez que o robô chega a um novo lugar cujo posicionamento das juntas leva a resultados indefinidos de cálculo, ele ignora o procedimento padrão e utiliza o valor do histórico no lugar. Esta estratégia se baseia na ideia de que, como os ângulos do robô quando inserido nos semiespaços +Y e -Y são contínuos ao longo das trajetórias, se no meio de um movimento a indefinição acontecer haverá menos prejuízo computacional, e possivelmente até no percurso das juntas, se elas simplesmente repetirem o último resultado válido em vez de mudarem a estratégia para criar um novo ponto descentralizado e manter possível o uso dos planos π_1 ou π_2.

A segunda estratégia foi adotada para o cenário do ângulo de pitch estar configurado para $\pm 90°$. Especificamente neste contexto os eixos ortogonais do efetuador podem ser considerados alinhados (ou seja, junta E = 0°), inclusive sob uma noção intuitiva, se:

- o eixo Ex estiver paralelo ao eixo Z da origem;
- o eixo Ey estiver paralelo ao eixo Y da origem; e

- o eixo Ez estiver paralelo ao eixo X da origem.

Conforme ilustra a figura 75. Esta configuração tende a ser entendida como uma situação de alinhamento natural até do ponto de vista intuitivo porque os três eixos ortogonais de cada sistema estão devidamente paralelizados. Desta forma, entendeu-se que o algoritmo de correção mais adequado para a situação disposta pode ser simplesmente descobrir o ângulo que o ponto de coordenadas [x,y] do alvo forma com a origem do sistema (não confundir com o ângulo da base, que a partir da versão 2 deixou de ter o mesmo valor) e usá-lo como valor de Eoffset() ao invés do cálculo original. Assim, a função pode ser alterada para o formato:

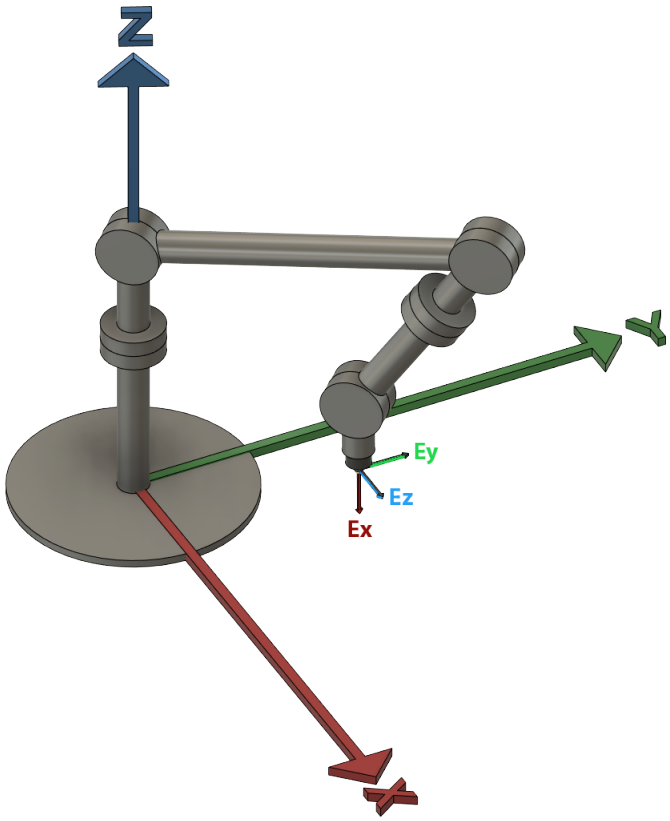


Figura 75 – Alinhamento entre os sistemas do efetuador e da origem.

```
1    Eoffset():
2        se pitch = 90 ou pitch = -90:
3            ang <- arctan(y/x)
4        senao:
5            ...
6            *executar calculo original*
7            ...
8        retornar [ang]
```

A implementação desta estratégia mostrou-se bem-sucedida, com uma boa transição entre ângulos da junta E calculados com a fórmula original de Eoffset() e com a alternativa acima, acionada no contexto específico de um ângulo de ±90° configurado para pitch. A versão completa do algoritmo de cinemática, já contemplando o trabalho feito para corrigir as indefinições encontradas ao longo do cálculo, pode ser vista na seção 6.5.

6.5 Algoritmo Final de Cinemática Inversa

```
1.  calcular D[x,y,z] a partir de [x,y,z][roll,pitch,yaw]:
    L <- distancia entre juntas D e E = link D
    D[x] <- [x] - L*cos(pitch)cos(yaw)
    D[y] <- [y] - L*cos(pitch)sen(yaw)
    D[z] <- [z] - L*sen(pitch)

2.  calcular [u,v,theta] a partir de D[x,y,z]
    D[u,v] <- [u,v]
    angbase <- theta

3.  calcular vetores vet_a e vet_c, partindo da junta A

4.  calcular o angulo da junta A pelo metodo da Triangulacao:
    triangulo abc:
        lado a: distancia entre juntas A e B
        lado b: distancia entre juntas B e D
        lado c: distancia [u,v] entre juntas A e D[u,v]

    angA <- leiDosCossenos(a,b,c) - produtoEscalar(vet_a,
       vet_c)

5.  utilizar triangulo formado, trocando os lados b e c
    de lugar, para descobrir o angulo no vertice da junta B:
    angB <- leiDosCossenos(a,c,b) - 180

6.  atualizar posicao [x,y,z] da junta B

7.  calcular o angulo da junta D pelo metodo da Triangulacao:
    triangulo abc:
        lado a: distancia entre juntas B e D
        lado b: distancia entre juntas B e [x,y,z] original
        lado c: comprimento L

    angD <- 180 - leiDosCossenos(a,b,c)

8.  calcular o angulo da junta C:
    se angD = 0:
        angC <- ultimoC
    senao:
        calcular vetor Bp
        calcular vetor Dp

        vtr <- produtoVetorial(Dp,Bp)
        se vtr aponta para junta B:
            angC <- produtoEscalar(Dp,Bp)
        senao:
            angC <- -produtoEscalar(Dp,Bp)

```

```
9.  calcular Eoffset():
    se angD = 0:
        angE <- ultimoE
    senao, se pitch = 90 ou pitch = -90:
        ang <- arctan(y/x)
    senao:
    9.1.calcular vetor normal ao plano 1:
        vet_DE <- vetor ligando D[x,y,z] a E[x,y,z]
        vet_DDof <- vetor ligando D[x,y,z] a Dof[x,y,z]

        vet_normal_p1 <- produtoVetorial(vet_DE, vet_DDof)

    9.2.calcular vetor normal ao plano 2:
        vet_DE <- calculado acima
        vet_DB <- vetor ligando D[x,y,z] a B[x,y,z]

        vet_normal_p2 <- produtoVetorial(vet_DB, vet_DE)

    9.3.descobrir se offset eh positivo ou negativo:
        vtr <- produtoVetorial(vet_normal_p1, vet_normal_p2)

        se vtr, saindo da junta E, aponta para a junta D:
            ang <- produtoEscalar(vet_normal_p1, vet_normal_p2)
        senao:
            ang <- -produtoEscalar(vet_normal_p1,
               vet_normal_p2)

10. angE <- roll - ang

11. salvar angC e angE para tratar indefinicoes futuras:
    ultimoC = angC
    ultimoE = angE

12. retornar [angbase, angA, angB, angC, angD, angE]
```

6.6 Integração ao Código-Fonte

Com a cinemática inversa tendo tomado forma, iniciou-se a implementação do controle cartesiano no código-fonte do robô. Desenvolveu-se a função *IK()*, que toma como entrada os parâmetros [x,y,z] [roll,pitch,yaw], executa o algoritmo da seção **??** e retorna o conjunto de ângulos que o robô deve adotar para chegar à posição dada. Houve relativo sucesso na adaptação para a linguagem Micropython, tendo sido possível obter com a função *IK()* respostas similares às obtidas com as simulações. Seguindo as recomendações dadas pela documentação oficial da linguagem (AU, 2018)(DOCS, s.d.), foram utilizadas variáveis de nomes curtos para poupar uso de memória RAM e organizou-se *IK()* como uma coleção de diversas funções menores ao invés da adoção de uma estratégia monolítica (todas as instruções escritas diretamente), o que contribui para diminuir o tempo de execução.

A cinemática inversa, no entanto, é apenas o recurso que conecta uma posição no espaço a um conjunto de ângulos das juntas. Para viabilizar o controle cartesiano, é necessário também implementar uma nova função que use tal recurso para o planejamento e execução de trajetórias. Neste contexto, um bom ponto de partida pode ser adotar a mesma estratégia utilizada durante a fase de simulação, em que entre os pontos final e inicial do movimento estabelecem-se diversos pontos intermediários compondo um caminho reto entra a origem e o destino - arranjo similar ao já mostrado com a figura 72.

Para cada um destes pontos é executada a função *IK()*, que entrega o conjunto de ângulos necessário, bastando então que o robô assuma cada conjunto sequencialmente para performar a trajetória linear. Para a interpolação entre os pontos é possível aproveitar o procedimento já existente no robô utilizado como ponto de partida, que concatena os deslocamentos individuais de cada junta de modo a fazer todas as envolvidas começarem e terminarem de mover-se no mesmo instante de tempo, conforme já descrito na seção 3.4.

Vale ressaltar que este procedimento não controla a natureza do percurso que o efetuador faz, somente a sincronia de cada eixo para assegurar um início e um final simultâneos da movimentação. Isto significa que durante o trânsito entre dois pontos calculados da trajetória não há como garantir que o robô irá se manter no traçado, caso a estratégia descrita seja adotada. A solução então pode ser incluir um número N de pontos intermediários tal que o espaçamento entre eles não permita um desvio significativo, caso ele ocorra. Quanto mais pontos, portanto, maior a *resolução* do movimento calculado - mas mais vezes será necessário chamar a função IK(), e portanto mais custoso será o cálculo da interpolação. Assim, é importante encontrar um equilíbrio entre a precisão e a performance.

Se for convencionado *movEixo(jun, vel)* como a função mencionada acima que coordena o movimento do robô de modo que suas juntas comecem e terminem de se mover no mesmo instante de tempo - tomando como argumento *jun* o conjunto de ângulos que o robô deve chegar e *vel* a velocidade -, pode-se escrever a rotina de cálculo da trajetória linear:

```
movXYZ(alvo, N, Vmax):

1. descobrir o deslocamento total para cada eixo:
    des[x] <- alvo[x] - pos_atual[x]
    des[y] <- alvo[y] - pos_atual[y]
    des[z] <- alvo[z] - pos_atual[z]

    des[ro] = alvo[ro] - pos_atual[ro]
    des[pi] = alvo[pi] - pos_atual[pi]
    des[ya] = alvo[ya] - pos_atual[ya]

2. descobrir o deslocamento entre os N pontos dados:
    d[x] <- des[x] / N
    d[y] <- des[y] / N
    d[z] <- des[z] / N

    d[ro] <- des[ro] / N
    d[pi] <- des[pi] / N
    d[ya] <- des[ya] / N

3. calcular o conjunto de angulos para cada ponto,
    comecando pelo primeiro ponto intermediario:
    ponto = pos_atual

    i=0
    enquanto i < N:
        ponto[x] <- ponto[x] + d[x]
        ponto[y] <- ponto[y] + d[y]
        ponto[z] <- ponto[z] + d[z]

        ponto[ro] += d[ro]
        ponto[pi] += d[pi]
        ponto[ya] += d[ya]

        angulos[i] <- IK(ponto)
        vels[i] <- calcula_vel(i, N, Vmax)
        i <- i + 1

4. mover o robo por entre os angulos calculados:
    i <- 0
    enquanto i < N:
        movEixo(angulos[i], vels[i])
```

Onde *calcula_vel(i, N, Vmax)* é uma função que determina a velocidade do robô com base em que fase da trajetória requisitada ele se encontra. Os primeiros 20% dos pontos são sujeitos a uma aceleração, e os últimos 20% a uma desaceleração, seguindo uma curva de formato sigmoide conforme mostrado na figura 76. Nas equações da figura, i é o i-ésimo ponto intermediário da trajetória (que vai de 0 a N) e α e β são constantes ajustadas em

código para manter o cálculo da velocidade sempre positivo.

Nos 60% intermediários, o robô se mantém na velocidade Vmax inserida. As porcentagens foram escolhidas apenas para fins demonstrativos, podendo ser alteradas na função *calcula_vel* caso outra combinação se mostre mais interessante. Com esta estratégia a saída da inércia e a volta à condição estacionária se dão de maneira suave, o que contribui para um movimento resultante com menos oscilações residuais indesejadas (típicas de saídas abruptas).

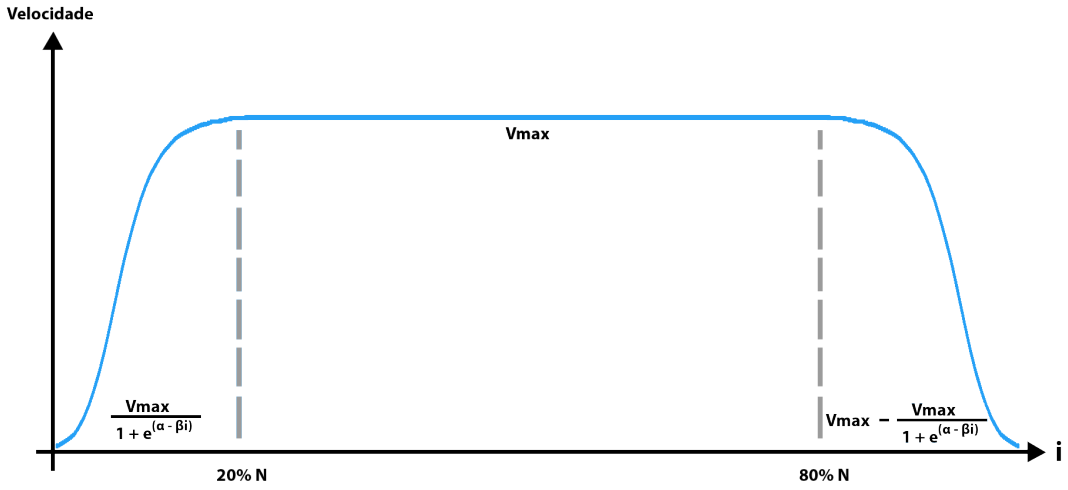


Figura 76 – Curva teórica de velocidade do robô.

6.7 Mudanças na Interface

Naturalmente não faria sentido implementar em código a cinemática inversa e o controle cartesiano se o usuário não pudesse fazer uso destes novos recursos; era preciso implementar mudanças também na interface para contemplar as adições feitas. Aproveitou-se que o número de juntas de controle individual - base, A, B, C, D e E - é o mesmo que o de eixos de controle cartesiano - X, Y, Z, Roll, Pitch e Yaw - para manter o mesmo layout da página, modificando apenas o esquema de cores para fácil diferenciação e a legenda dos botões para referenciar adequadamente a nova modalidade de controle. Desta forma, portanto, conseguiu-se manter praticamente toda a página html original.

A figura 77 mostra as agora três telas que podem ser encontradas pelo usuário do robô quando acessa sua interface.

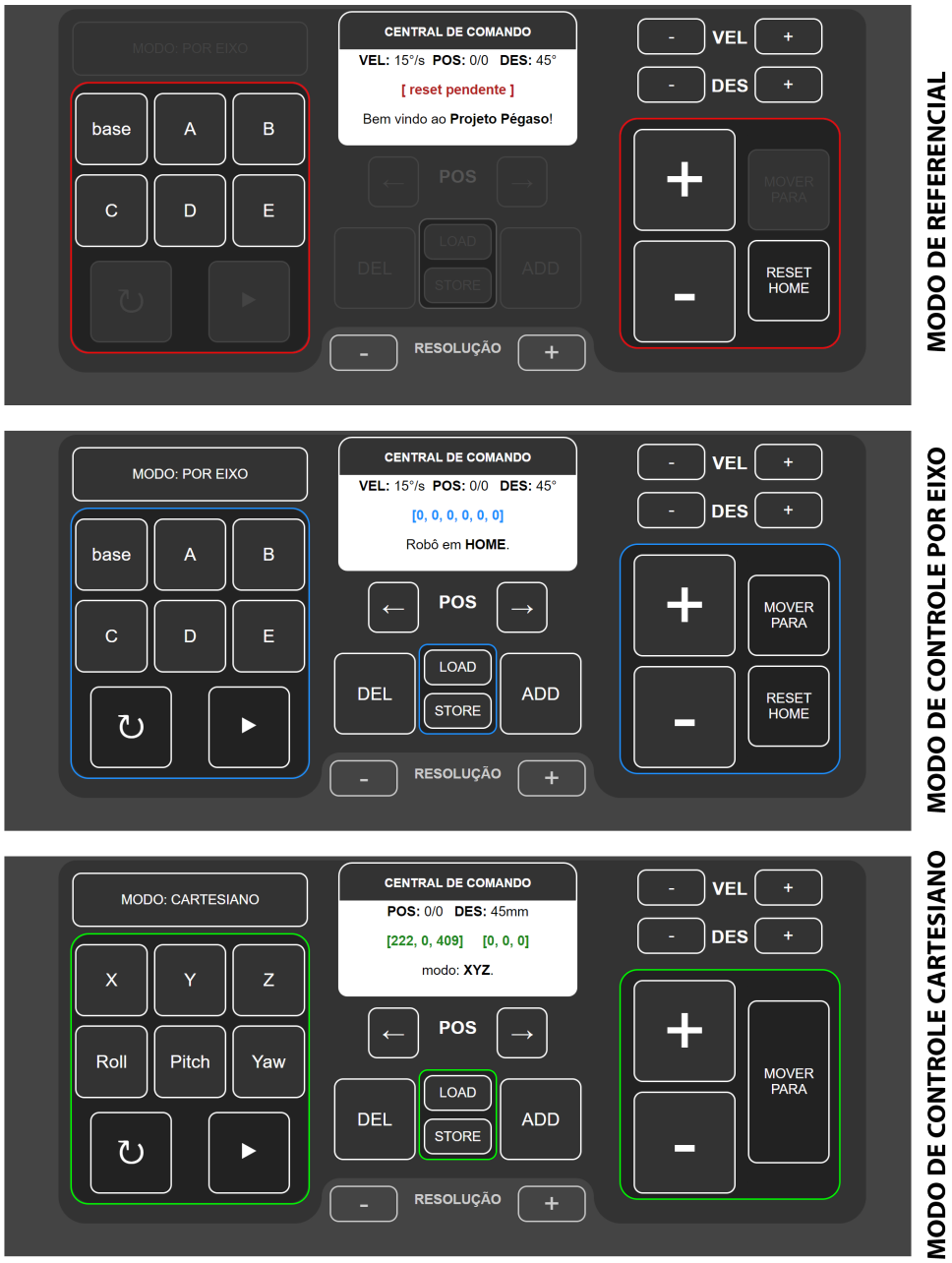


Figura 77 – As três telas da interface.

- **Modo de Referencial:** É a primeira tela que aparece quando se carrega a página do robô, logo após tê-lo ligado. Possui boa parte de seus botões bloqueados porque serve ao único propósito de permitir que o usuário referencie manualmente os eixos do robô. Assim, ele seleciona um dos seis eixos individualmente e vai movendo-os até que o PegasoV3 atinja sua posição HOME. Uma vez tendo feito isto, basta pressionar a tecla "RESET HOME" que a interface migra para a modo de controle por eixo.
- **Modo de Controle por Eixo:** É a tela original do PegasoV3. Através dela é possível mover os eixos do robô individualmente, criar uma fila de posições e comandar que o

robô as percorra sequencialmente, gravar a a mesma no cartão de memória ou carregar alguma já disponível anteriormente nele, e alterar a velocidade e a magnitude de cada incremento posicional através dos botões "VEL" e "DES", entre outras funcionalidades. Possui no canto superior esquerdo um botão de modo que altera o nome de acordo com a tela em que o usuário se encontra. Pressionar este botão irá mudar a interface para o modo de controle cartesiano.

- **Modo de Controle Cartesiano:** Possuindo detalhes em cor verde como forma de permitir fácil associação visual quanto ao modo de controle do robô, esta tela permite que o usuário opere a máquina em termos das coordenadas cartesianas de seu efetuador terminal, tal qual desenvolvido e explicado em detalhes nas seções anteriores. Possui a maioria dos recursos do controle por eixo, com algumas diferenças: A fila de posições neste modo é administrada paralelamente à fila do modo por eixo, podendo o usuário criar rotinas diferentes para cada um se assim o desejar; e não é possível resetar a posição HOME neste modo devido a problemas potencialmente muito maiores que poderiam decorrer de uma referência reconfigurada incorretamente nesta página comparada à outra.

Os três modos contam com duas adições em relação à interface gráfica original, visualizável na figura 23. A primeira delas é o novo recurso de alterar a resolução dos motores, disponível no canto central inferior. Com o pressionar dos botões '+' e '-' desta seção é possível aterar a relação de micropassos dos drivers DRV8825, indo do ajuste fino e lento de 32 micropassos/passo até a opção mais abrupta de 4 micropassos/passo. A inclusão desta funcionalidade se deu com a constatação de que, para o propósito cinematográfico, a performance do robô tenderá a ser melhor com a resolução mais fina, porém durante a fase do ajuste das posições da fila a velocidade baixa que ela confere torna o processo muito demorado. Assim, a presença deste item permite que o "setup" do movimento seja feito de maneira mais dinâmica e reserva a opção mais precisa e estável para o momento em que ela se mostra mais interessante.

A segunda inclusão foi a de informações na central de comando, onde as mensagens do robô são mostradas ao usuário. Agora há o aviso de que o referenciamento dos eixos ainda precisa ser feito no caso da primeira tela, e para as outras duas fica aparente a posição atual em que o robô se encontra - seja no conjunto de graus das juntas no caso do modo por eixo, ou nas coordenadas xyz e ângulos de orientação do efetuador no caso do modo cartesiano. Também é possível verificar no menu superior em qual posição da fila o usuário se encontra e a posição 0 por definição é sempre a HOME, podendo então o usuário dirigir o robô a ela com mais facilidade.

7 Mudanças Adicionais de Estrutura e Motorização

Neste último capítulo de desenvolvimento serão abordadas as melhorias de caráter geral implementadas no robô com o intuito de deixá-lo mais robusto e adequado para a tarefa cinematográfica. Neste contexto foram realizadas as seguintes modificações:

- Troca dos motores das juntas A e B por alternativas mais potentes;
- Alteração na tipologia das engrenagens usadas no sistema planetário para dentes helicoidais ao invés de retos; e
- Melhoria do sistema de transmissão da junta D.

7.1 Troca de Motores das Juntas A e B

O arranjo original de motorização do robô era uma combinação de três motores de passo e três servomotores, conforme já explicado nas primeiras seções deste Trabalho. Esta opção mostrava-se interessante porque, enquanto os motores de passo poderiam conferir precisão e ao mesmo tempo se alojarem próximos à base - causando pouca interferência na carga paga - os servomotores assumiriam as juntas de orientação do efetuador com torque expressivo e peso sensivelmente menor que suas contrapartidas de passo. Mas apesar de fazer sentido teórico, esta combinação foi logo descartada com as primeiras alterações realizadas (capítulo 4) porque o movimento que o robô era capaz de produzir nesta configuração possuía desvantagens relevantes para o propósito cinematográfico (mencionadas na seção 3.5).

O novo arranjo de motorização adotado já com a primeira iteração de trabalhos estruturais (seção 5.1) contemplava somente motores de passo em todos os eixos do robô. Eles eram de apenas um modelo, 17HS3401, para diminuir a variabilidade de peças do projeto, mas a implementação nas juntas mais afastadas da base trouxe um aumento de peso excessivo. Assim, Na terceira iteração (seção 5.3) uma versão mais leve de NEMA 17 foi inclusa no lugar, levando a um alívio de peso de cerca de 165g à modificação inicial. O robô manteve-se capaz de movimentar-se com e sem um celular (carga paga desejada) acoplado ao efetuador quando comparado ao emprego dos motores mais fortes, e portanto a modificação foi considerada bem-sucedida.

Estendendo a lógica que levou aos motores mais leves nas juntas próximas ao efetuador, foi decidido ampliar a margem de segurança do robô substituindo os motores das juntas A e B - as mais solicitadas mecanicamente - por opções mais pesadas e potentes.

Devido ao fato de ambos ficarem sobre a base, atrás do ponto de equilíbrio no diagrama de esforços traçado (figura 43), é perceptível que uma aumento no peso deles pode trazer contribuições positivas para o balanceamento no pior cenário de cargas estáticas. No caso do motor B, também promover um recuo no centro de massa do link A. Para a substituição foram escolhidos motores 17HS4401, que dispõem de 12N.cm a mais de torque estático que os 17HS3401 (levando a um total de 40N.cm). Com a redução de 125:1 anexa ao motor da junta A, são 1500N.cm teóricos a mais de torque estático disponível, um aumento de 42% sobre o valor original.

7.2 Alteração da Tipologia das Engrenagens

O PegasoV3 dispõe de apenas dois tipos de engrenagem em sua constituição: O tipo cilíndrico de dentes retos, que equipa a redução simples da base e os conjuntos planetários dos demais eixos, e o tipo cônico de dentes retos, que equipa as transmissões de movimento perpendicular para as juntas B e D. Independente do tipo, todas as engrenagens empregadas no projeto são fabricadas da mesma maneira que a quase totalidade das peças do robô - utilizando filamento PETG e uma impressora 3D. E esta, apesar de ser uma opção muito bem vinda para fabricar protótipos e validar ideias, não é a opção indicada para a confecção de peças que exijam tolerâncias apertadas. Não é difícil que uma máquina desregulada faça com que as engrenagens impressas apresentem, por exemplo, alguma imperfeição nos dentes ou excentricidade invisíveis aos olhos, porém perceptíveis com ruídos e trepidações na movimentação do mecanismo em que ela é inserida.

Apesar de ao longo deste Trabalho terem sido experimentados diversos ajustes na impressora e variações nas tolerâncias das engrenagens, não foi possível chegar a uma configuração que dirimisse o efeito das imperfeições nas peças. Considerando a aplicação cinematográfica proposta, é perceptível que as vibrações na movimentação das juntas decorrentes destas imperfeições podem prejudicar a qualidade das filmagens realizadas. Neste contexto propõe-se uma forma alternativa de abordar o problema: alterar a tipologia das engrenagens, trocando-as por modelos cilíndricos de dentes helicoidais - conforme ilustrado na figura 78.

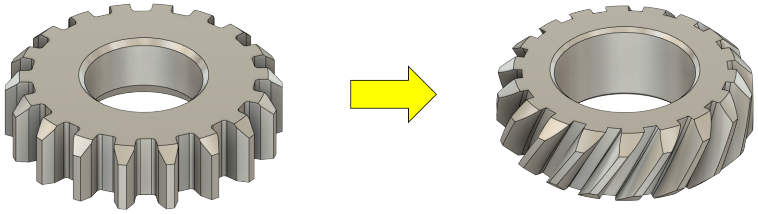


Figura 78 – Nova tipologia das engrenagens: de dentes retos para helicoidais.

Engrenagens do tipo ilustrado têm um ângulo aplicado no momento de confecção dos dentes que os deixam inclinados em relação às peças retas equivalentes. Em virtude disso, uma associação de engrenagens helicoidais promove um acoplamento gradual entre os dentes que ameniza o contato e resulta em uma transmissão mais suave, especialmente quando em alta velocidade. Além disso, há um menor ruído associado ao funcionamento e uma maior resistência mecânica por parte dos dentes da engrenagem, que fica com uma distribuição de esforços mais tridimensionalizada. Além disso, como a inclinação (formalmente chamada de *ângulo de hélice*) aumenta o volume de cada dente em relação à contrapartida reta, o sistema de redução hélica ainda conta com uma tolerância ao aquecimento maior (SINGH, 2019)(MEDIA, 2018).

A maior desvantagem relacionada à tipologia helicoidal consiste na geração de esforços de reação ao longo do eixo em que ela se insere, o que também a leva a perder um pouco de rendimento da transmissão. Em sistemas com grandes forças envolvidas, como por exemplo caixas de redução de carros e máquinas pesadas, esse esforço de reação precisa ser combatido com a adoção de um invólucro mais reforçado e um rolamento indicado para cargas axiais, o que acarreta em mais peso para a caixa de redução. O robô desenvolvido neste trabalho, no entanto, não possui forças envolvidas comparáveis às de um carro, e o rolamento 6004-2RS já presente em sua caixa de redução, sendo projetado para velocidades da ordem de 11000 RPM e carga dinâmica básica de 9.95 kN (ROLAMENTOS..., s.d.), foi considerado suficiente para conter o surgimento de tais esforços.

A nova caixa de redução planetária, desenvolvida para a tipologia helicoidal e com dimensões e pontos de encaixe semelhantes aos do modelo originalmente projetado (figura 27), pode ser visualizada na figura 79.

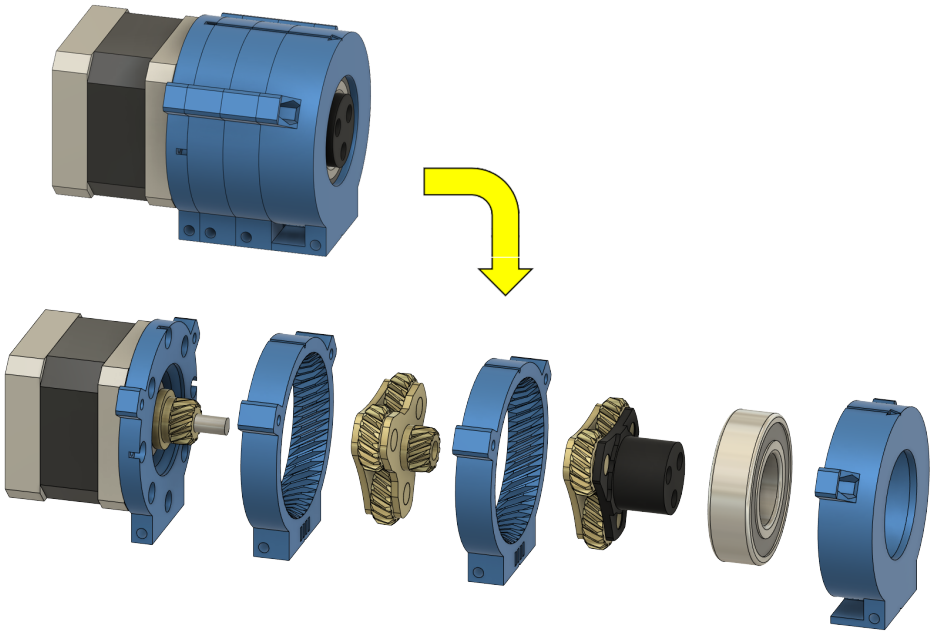


Figura 79 – Nova caixa de redução, adequada para a tipologia helicoidal.

7.3 Melhoria do sistema de transmissão da junta D

À medida que testes de movimentação foram sendo realizados, a junta D do robô conforme ilustrada na figura 48 passou a apresentar problemas eventuais de travamentos, perda de sincronia dos dentes e folgas além do tolerado, que comprometem severamente a capacidade do robô de movimentar-se de maneira estável e condizente com a proposta cinematográfica. Após uma rodada de investigações em torno da referida junta, entendeu-se que a possível causa podia estar relacionada:

- ao atrito excessivo entre a engrenagem cônica fixada à junta D e a peça lateral que segurava o conjunto; e
- ao desenho em si das engrenagens cônicas, que poderia estar com um projeto de acoplamento dos dentes ineficiente.

Enquanto o primeiro ponto pode ser resolvido diminuindo a largura da respectiva engrenagem, o segundo exige pelo menos o redesenho dos dentes das duas engrenagens cônicas de forma a melhorar o engate entre elas. Em face a esta constatação, julgou-se uma ideia melhor reformular o sistema por completo, aproveitando o desenho das peças que já compõem a junta B (ver topo da figura 33). A principal razão consiste em que, ao contrário da junta D em análise, o acoplamento do eixo B não apresentou qualquer problema de desempenho até o momento, e isto fornece maior certeza quanto ao provável sucesso que esta opção leva. Como as engrenagens advindas da junta B são nitidamente menores que as da junta D, não é possível simplesmente posicioná-las na região desejada - uma reestruturação da maioria das peças em volta deve acompanhar a mudança. Neste contexto, propõe-se o arranjo visto na figura 80.

O motor e a redução planetária são recuados para abrir espaço para as novas engrenagens, de desenho similar às da junta B. A engrenagem cônica guiada mescla-se a uma outra, cilíndrica de dentes retos, que por sua vez leva o movimento adiante para o último elemento da transmissão - este sim acoplado à junta D e mais fino que a engrenagem original para resolver a questão do atrito mencionada no primeiro item.

A proporção entre as duas engrenagens de transmissão perpendicular se mantém em 1:1, mas o nível seguinte segue uma relação 2:1 devido à diferença de tamanho entre os dois componentes finais de dente reto. Com esse incremento à redução planetária 25:1, o valor teórico total da junta D passa a ser de 50:1, ou 700N.cm considerando o torque original do motor de 14N.cm. Para concluir, com a remoção das engrenagens cônicas originais houve uma liberação de espaço vertical nas proximidades da junta que permitiu um aumento de amplitude da mesma de -90°/+90° para -115°/+115°, um aumento de 27%. A comparação entre o antes e o depois está disponível na figura 81. E por fim, o robô após este último

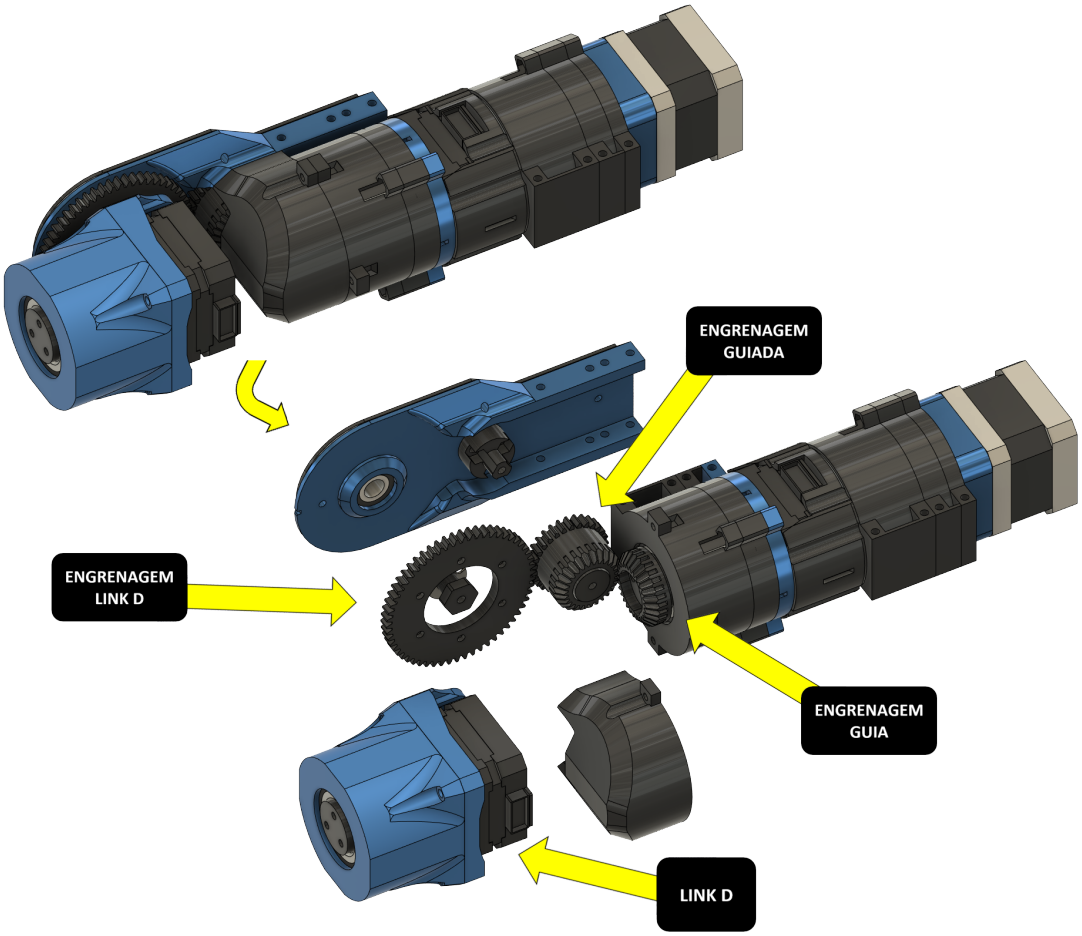


Figura 80 – Novo arranjo para a junta D.

conjunto de modificações pode ser visto na imagem 82, junto à interface criada para o controle cartesiano.

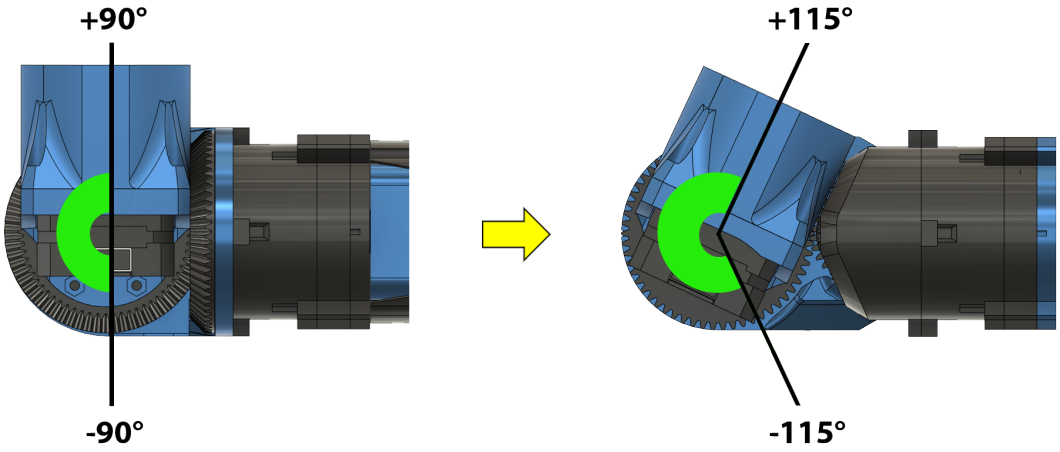


Figura 81 – Nova amplitude para a junta D.

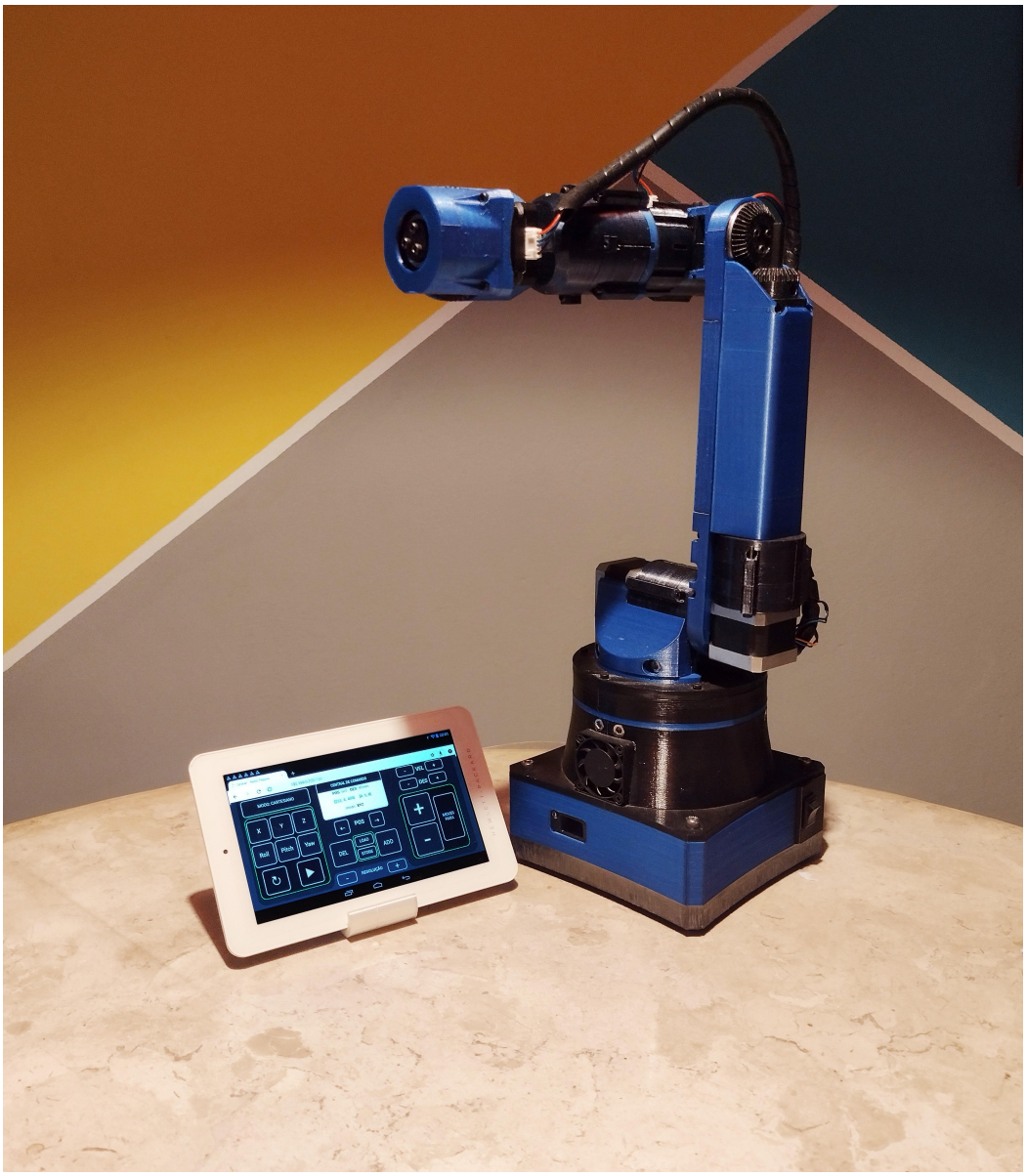


Figura 82 – PegasoV3 após as últimas modificações realizadas.

8 Conclusões

Ao longo deste Trabalho de Graduação foi possível explorar e desenvolver melhorias nos principais aspectos que compõem o robô. De posse da versão inicial dotada de uma placa de circuito artesanal, interface por linha de comando, opções de controle limitadas, dois tipos de motores e construção simétrica, foi proposto um conjunto de pontos de melhoria usadas como referência para executar todas as mudanças descritas neste documento. O robô resultante, batizado de PegasoV3, conta com alterações significativas em toda sua composição que o tornam mais preciso, robusto, organizado e controlável, tendo conseguido lidar com a maior parte dos referidos pontos. Somente uma peça da estrutura do robô usado como ponto de partida manteve-se inalterada após o curso de todas as mudanças.

O projeto também contou com a implementação bem-sucedida de um dos elementos mais importantes da robótica, a cinemática inversa, tendo sido comparados trabalhos anteriores do campo e ponderado que a opção mais adequada para o contexto do robô era o Método da Triangulação de Cajar e Mukundan. Houve a observação de descontinuidades e o tratamento de indefinições que o algoritmo adaptado para o PegasoV3 poderia vir a apresentar. A interface do robô foi adequada para o novo modo de controle - por coordenadas cartesianas ao invés de apenas pelo conjunto de ângulos das juntas - e foram feitas mudanças mecânicas adicionais com foco em melhorar a suavidade da movimentação do robô.

8.1 Trabalhos Futuros

Conforme o desenvolvimento deste Trabalho de Graduação tomou forma, foram abordados diversos pontos e muitos deles apresentaram boas respostas às modificações feitas. Ainda assim, há muitos aspectos não abordados nestas páginas que irão receber atenção em desenvolvimentos futuros. Por exemplo, apesar de o controle cartesiano ter se tornado uma possibilidade com a implementação bem-sucedida da cinemática inversa ao código do robô, faltou incluir também o cálculo direto para a confirmação do resultado. O robô ainda não dispõe de sensores que acusem a inclinação das juntas e o método de referenciamento dos eixos é manual, características certamente indesejadas com relação à confiabilidade da operação do robô.

Devido à falta de tempo hábil para este trabalho, a zona morta das juntas não foi resolvida. Também não houve testes extensivos das novas reduções planetárias de engrenagens helicoidais nem a confirmação da carga paga com que o robô consegue lidar. É importante também realizar testes de movimentação para obter as partes do volume de trabalho que irão render os melhores resultados cinematográficos. Por fim, apesar de o desenvolvimento do PegasoV3 ter sido orientado pelo conceito de prestar assistência na confecção de tomadas

de vídeo, poucos foram os momentos no trabalho em que o robô foi mostrado performando tarefas neste sentido. Assim, é necessário testá-lo melhor para esta finalidade.

Referências

.R, D.; HONNARAJU, B.; MURALI, S. Path Generation for Robot Navigation using a Single Camera. **Procedia Computer Science**, v. 46, p. 1425–1432, dez. 2015. DOI: 10.1016/j.procs.2015.02.061. Citado na p. 17.

ABB. **IRB 1200 | ABB Robotics - Industrial Robots Portfolio**. Website. Disponível em: <https://new.abb.com/products/robotics/industrial-robots/irb-1200>. Citado na p. 19.

ABB. **IRB 6700 | ABB Robotics - Industrial Robots Portfolio**. Website. Disponível em: <https://new.abb.com/products/robotics/industrial-robots/irb-6700>. Citado nas pp. 21, 46.

ALLEGRO. **A4988 DMOS Microstepping Driver with Translator and Overcurrent Protection**. Rev. 8. Citado na p. 23.

ARISTIDOU, A.; LASENBY, J. FABRIK: A fast, iterative solver for the Inverse Kinematics problem. **Graphical Models**, v. 73, p. 243–260, set. 2011. DOI: 10.1016/j.gmod.2011.05.003. Citado nas pp. 17, 65, 87.

ATMEL. **8-bit AVR Microcontroller with 4/8/16/32K Bytes In-System Programmable Flash**. Rev. 8271BS-AVR-04/10. Citado na p. 23.

AU, P. **Writing Fast and Efficient Micropython**. Youtube. 2018. Disponível em: <https://youtu.be/hHec4qL00x0>. Citado na p. 94.

BROWNLEE, M. **Dope Tech: Camera Robots!** Youtube. 2018. Disponível em: <https://youtu.be/UIwdCN4dV6w>. Citado na p. 15.

DAS, A.; SHIRWALKAR, V.; KAR, D.; PANDA, S. Robotic Camera Assistant for Laparoscopic Surgery. In: p. 3. Citado na p. 17.

DOCS, M. **Maximising MicroPython speed**. Website. Disponível em: <https://docs.micropython.org/en/latest/reference/speed_python.html>. Citado na p. 94.

DONELAN, P. Singularity-theoretic methods in robot kinematics. **Robotica**, v. 25, p. 641–659, nov. 2007. DOI: 10.1017/S0263574707003748. Citado na p. 87.

FANUC. **Robô Industrial FANUC LRMate 200iD**. Website. Disponível em: <https://www.fanuc.eu/pt/pt/rob%5C%C3%5C%B4s/p%5C%C3%5C%A1gina-filtro-rob%5C%C3%5C%B4s/s%5C%C3%5C%A9rie-lrmate/lrmate-200-id>. Citado na p. 18.

HOLM, E. J. V. Makerspaces and Contributions to Entrepreneurship. **Procedia - Social and Behavioral Sciences**, v. 195, p. 24–31, 2015. World Conference on Technology, Innovation and Entrepreneurship. ISSN 1877-0428. DOI: https://doi.org/10.1016/j.sbspro.2015.06.167. Disponível em: <https://www.sciencedirect.com/science/article/pii/S1877042815036460>. Citado na p. 15.

KHK. **Gear Systems | KHK Gear Manufacturer**. Website. Disponível em: <https://khkgears.net/new/gear_knowledge/gear_technical_reference/gear_systems.html>. Citado na p. 42.

MEDIA, D. **HELICAL VS. STRAIGHT CUT GEARS | Donut Media**. Youtube. 2018. Disponível em: <https://youtu.be/Zy4kYFWZ1_g>. Citado na p. 102.

MULLER-CAJAR, R.; MUKUNDAN, R. Triangulation: A new algorithm for Inverse Kinematics. **Image and Vision Computing - IVC**, jan. 2007. Citado nas pp. 17, 65, 66, 87.

NXP SEMICONDUCTORS. **16-channel, 12-bit PWM Fm+ i2C bus LED controller**. Rev. 04. Citado na p. 33.

PAGLIARINI, L.; LUND, H. The future of Robotics Technology. **Journal of Robotics, Networking and Artificial Life**, v. 3, p. 270, fev. 2017. DOI: 10.2991/jrnal.2017.3.4.12. Citado na p. 15.

POLOLU. **A4988 Stepper Driver**. Website. Disponível em: <https://www.pololu.com/product/1182>. Citado na p. 23.

POLOLU. **DRV8825 Stepper Driver**. Website. Disponível em: <https://www.pololu.com/product/2133>. Citado na p. 35.

PROJECTS, T. E. **Introduction to ESP-01 - The Engineering Projects**. Website. Disponível em: <https://www.theengineeringprojects.com/2019/05/introduction-to-esp-01.html>. Citado na p. 35.

PROTOTYPING, S. **Stepper Motor NEMA 17 42x42mm**. Website. Disponível em: <https://www.smart-prototyping.com/Stepper-Motor-NEMA-17-42x42mm.html>. Citado na p. 59.

RASPBERRY FOUNDATION. **Raspberry Pi Pico Datasheet**. Rev. 150df05-clean. Citado nas pp. 31–33.

SIEMASZ, R.; TOMCZUK, K.; MALECHA, Z. ScienceDirect ScienceDirect 3D printed robotic arm with elements of artificial intelligence. **Procedia Computer Science**, v. 176, p. 3741–3750, out. 2020. DOI: 10.1016/j.procs.2020.09.013. Citado na p. 17.

SINGH, S. K. Helical Gear Design and Analysis: A Review. **IJRAR**, v. 6, jan. 2019. DOI: E-ISSN2348-1269,P-ISSN2349-5138. Citado na p. 102.

SKF. **Rolamentos rígidos de esferas**. Rev. 01. Citado na p. 102.

SUDER, J.; BOBOVSKÝ, Z.; MLOTEK, J.; VOCETKA, M.; ZEMAN, Z.; SAFAR, M. EXPERIMENTAL ANALYSIS OF TEMPERATURE RESISTANCE OF 3D PRINTED PLA COMPONENTS. **MM Science Journal**, v. 2021, p. 4322–4327, mar. 2021. DOI: 10.17973/MMSJ.2021_03_2021004. Citado na p. 18.

TOWERPRO. **Metal Gear Servo Motor MG-995**. Website. Disponível em: <https://www.towerpro.com.tw/product/mg995/>. Citado nas pp. 19, 23.

USONGSHINE. **2 Phase Hybrid Pancake Stepper Motor**. Rev. 01. Citado na p. 58.

USONGSHINE. **2 Phase Hybrid Stepper Motor**. Rev. 04. Citado nas pp. 19, 23, 58.

VENTUREBEAT. **How Robots Filmed Hollywood's Latest Blockbuster, 'Gravity'**. Website. Out. 2013. Disponível em: <https://venturebeat.com/2013/10/07/robots-filmed-hollywoods-next-blockbuster/>. Citado na p. 15.

WANG, L.-C.; CHEN, C. A combined optimization method for solving the inverse kinematics problem of mechanical manipulators. **Robotics and Automation, IEEE Transactions on**, v. 7, p. 489–499, set. 1991. DOI: 10.1109/70.86079. Citado nas pp. 17, 65.

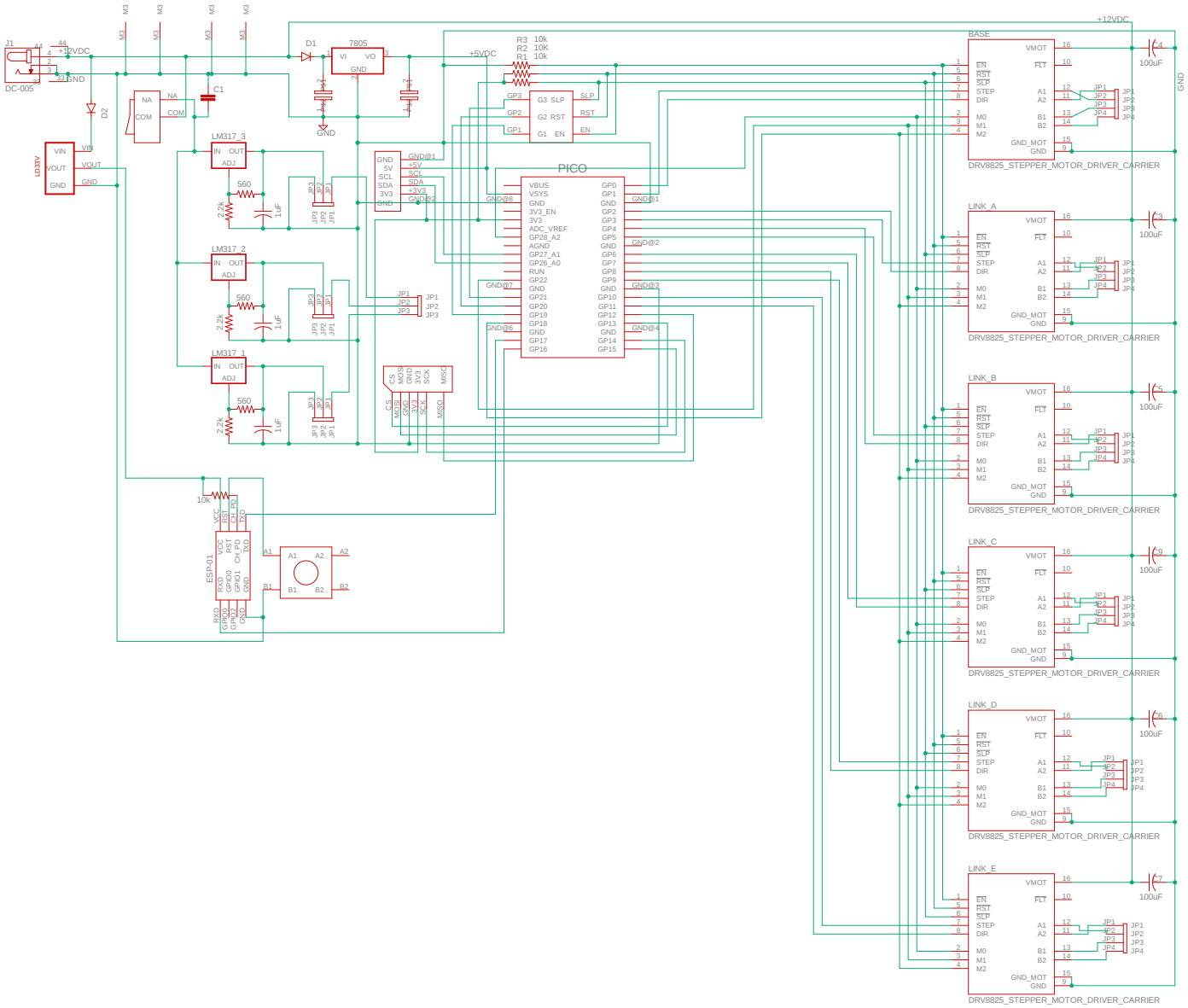
WLKATA. **Download Center - Software and Support Documents**. Website. Disponível em: <https://www.wlkata.com/support/download-center>. Citado na p. 17.

WLKATA. **WLKATA Mirobot - The Smallest 6 axis Robotic Arm**. Youtube. 2019. Disponível em: <https://youtu.be/75v19xe99Vk?t=194>. Citado na p. 17.

Apêndices

APÊNDICE A – Esquemático PegasoBoard v1.0

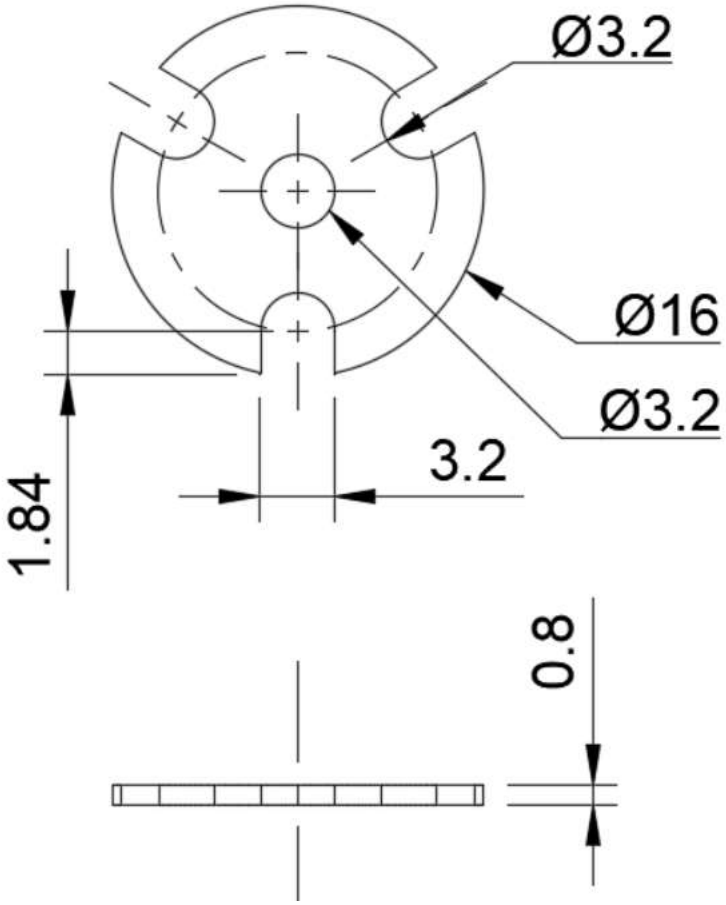
A seguir encontra-se o esquemático das conexões elétricas da PegasoBoard v1.0.

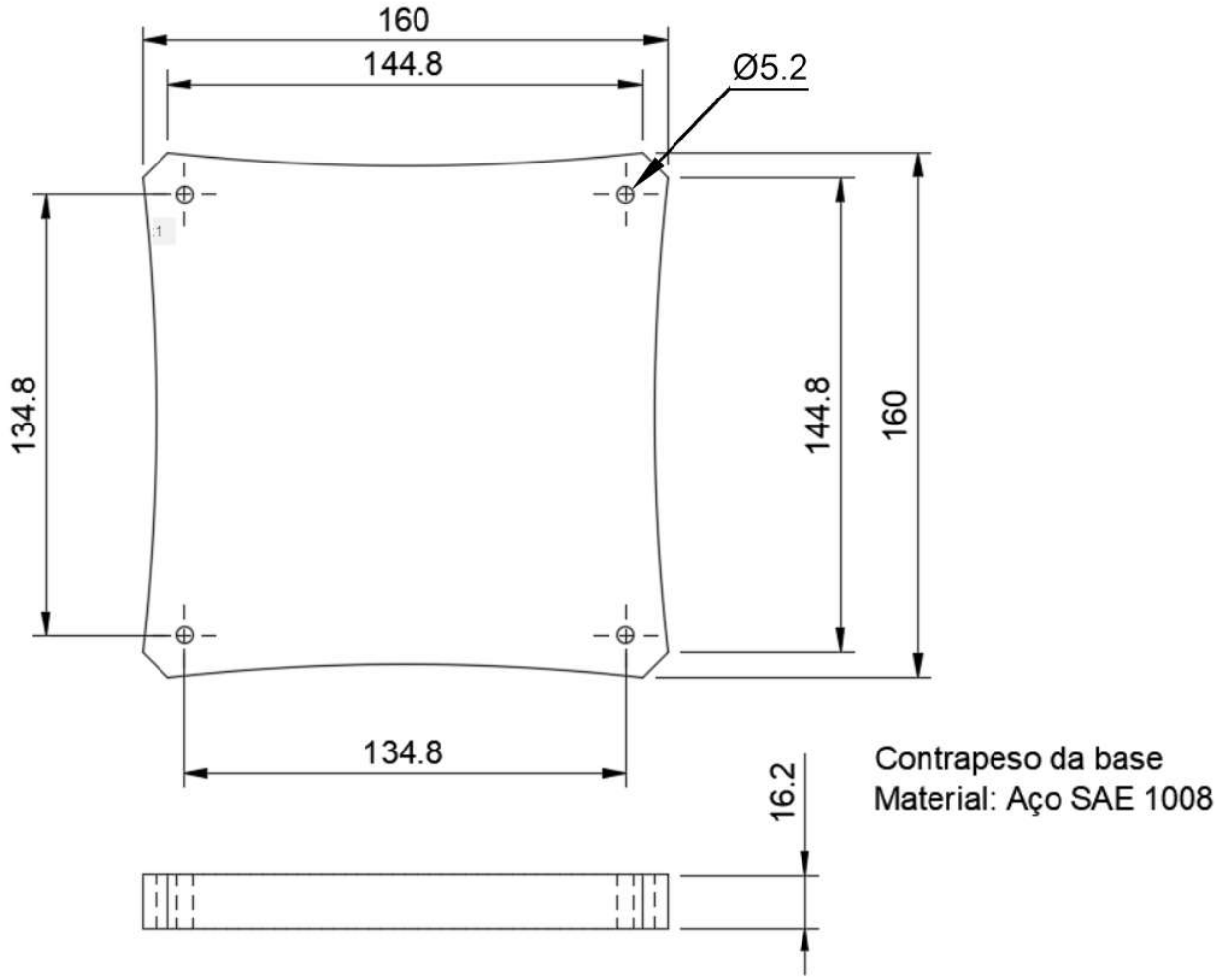


APÊNDICE B – Desenhos Técnicos

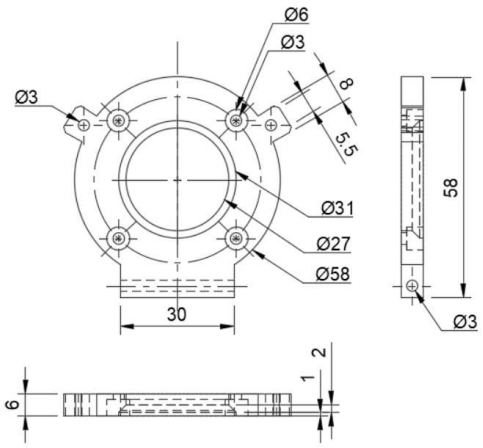
Os desenhos técnicos ilustrados a seguir contemplam as principais peças e conjuntos desenvolvidos ao longo do trabalho. As medidas são dadas em milímetros.

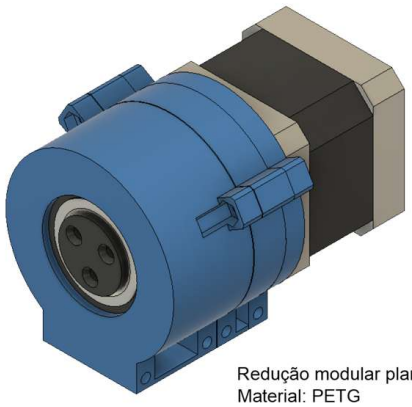
Arruela de três pontos
Material: chapa aço 0.8mm




160
144.8
Ø5.2
134.8
144.8
160
134.8
16.2
Contrapeso da base
Material: Aço SAE 1008

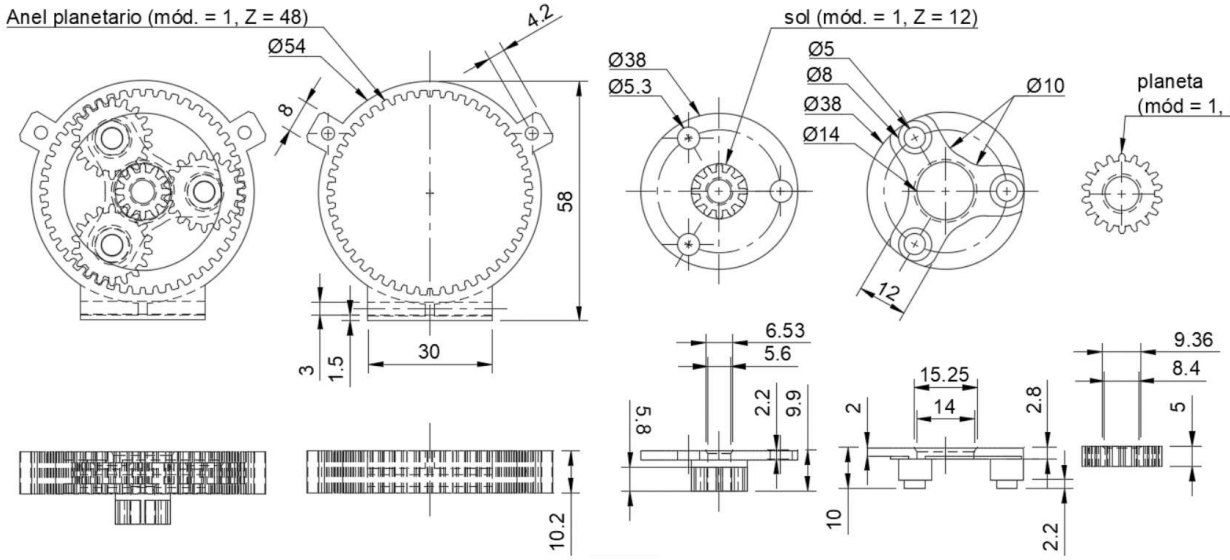
base



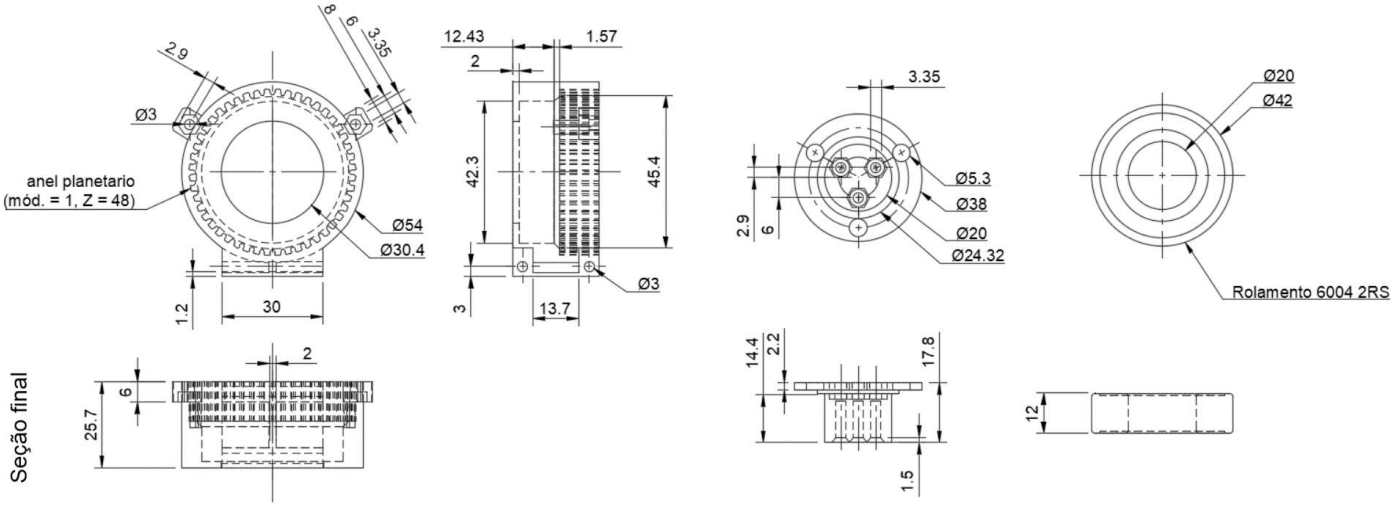


Redução modular planetária de dentes retos
Material: PETG
Motor: NEMA 17

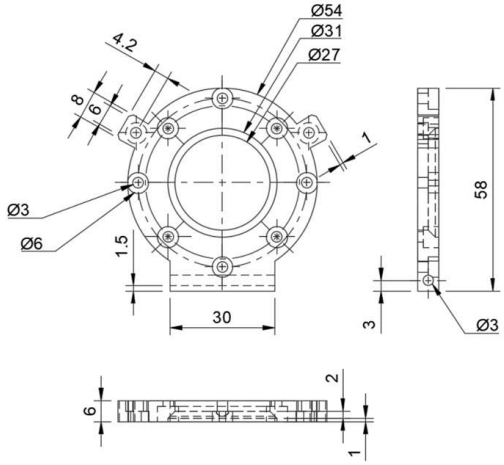
Seção modular

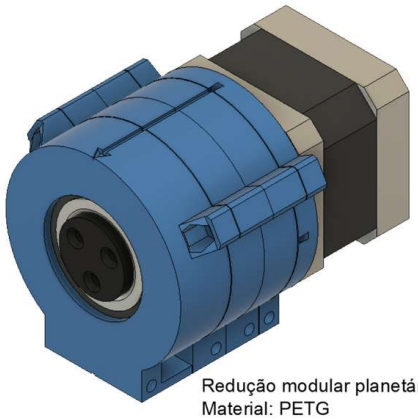


Seção final



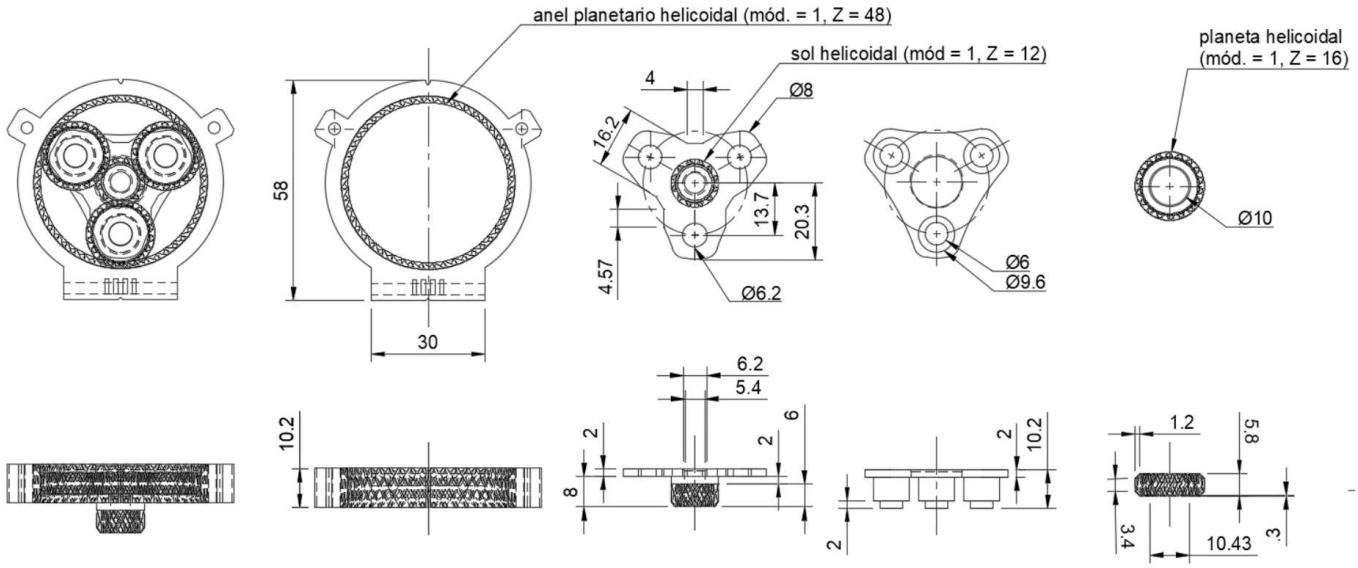
base





Redução modular planetária de dentes helicoidais
Material: PETG
Motor: NEMA 17

Seção modular



Seção final