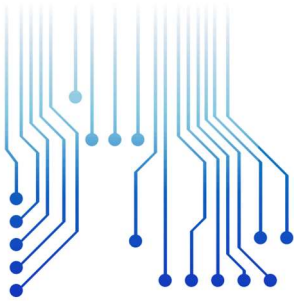
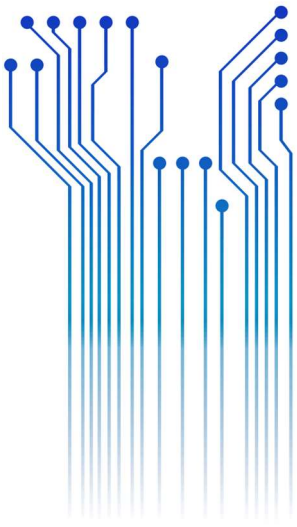


CURSO DE GRADUAÇÃO EM ENGENHARIA ELÉTRICA

FELIPE HENRIQUE NEIVA DO NASCIMENTO

RELATÓRIO DE ESTÁGIO

PROJETO DE MANIPULADOR DELTA



Campina Grande
2016

FELIPE HENRIQUE NEIVA DO NASCIMENTO

PROJETO DE MANIPULADOR DELTA

Relatório de Estágio Supervisionado submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Controle e Automação

Orientador:
Professor Antônio Marcus Nogueira Lima, D. Sc.

Campina Grande
2016

FELIPE HENRIQUE NEIVA DO NASCIMENTO

PROJETO DE MANIPULADOR DELTA

Relatório de Estágio Supervisionado submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Controle e Automação

Aprovado em / /

Professor Avaliador
Universidade Federal de Campina Grande
Avaliador

Professor Antônio Marcus Nogueira Lima
Universidade Federal de Campina Grande
Orientador, UFCG

AGRADECIMENTOS

Agradeço a meu pai, Fernando, e meu irmão, Pedro, pela força e companheirismo familiar ao longo dos anos de curso e da vida.

Agradeço aos professores Antônio Marcus, Marcos Morais e Alexandre Cunha pela dedicação ao ensino e paciência.

Agradeço por fim a Vinicius e ao Senai Stenio Lopes, pela confiança depositada.

‘‘I do not fear computers. I fear the lack of them’’
- Isaac Asimov.

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

A área de engenharia robótica é foco de crescente estudo e investimento nos últimos anos, tanto nas frontes de pesquisa da área pelo mundo, quanto nas aplicações desses conhecimentos em produtos de países de primeira ponta e países em desenvolvimento.

Introduzido inicialmente na década de 1980 por Clavel, o robô delta rotacional foi alvo de diversos estudos ao longo das últimas décadas e suas aplicações variam desde um simples manipulador robótico até braços cirúrgicos. Na última década, robôs delta tem sido usados no ambiente de impressoras 3D, onde sua precisão, resolução e distribuição de carga tem chamado atenção para produção de produtos mais baratos e com maior eficiência.

O estágio foi realizado no Senai Stenio Lopes, localizado em Campina Grande. Tendo início em 15 de maio de 2015 e com término em 14 de outubro de 2015 com carga horária de 20 horas semanais. A unidade é um centro de ensino técnico e presta serviços para diversas empresas no setor de tecnologia. Um dos laboratórios encontrados na unidade é o laboratório de automação, centro de desenvolvimento e serviços da unidade. O estágio foi realizado nesse laboratório.

No estágio foi iniciado o projeto de uma impressora 3D Delta Rotacional, além de ser criada uma base para criação de outros robôs que utilizam a estrutura Delta. Esse relatório tem como objetivo sintetizar os conhecimentos teóricos e práticos adquiridos no estágio e servir de base para seguimento do projeto na empresa.

2 PROJETO DE IMPRESSORA DELTA

2.1 MOTIVAÇÃO

O laboratório de automação do Senai Stenio Lopes possuía, na época, uma das poucas impressoras 3D de médio a grande porte na cidade de Campina Grande, e de reuniões entre o estagiário e a equipe do laboratório, surgiu a ideia de desenvolver o protótipo de uma impressora 3D de baixo custo para atender a necessidade do mercado local. Sendo um projeto de grande porte e devido ao tempo limitado do estagiário no projeto, foi decidido que este seria dividido em partes, divididas da seguinte forma:

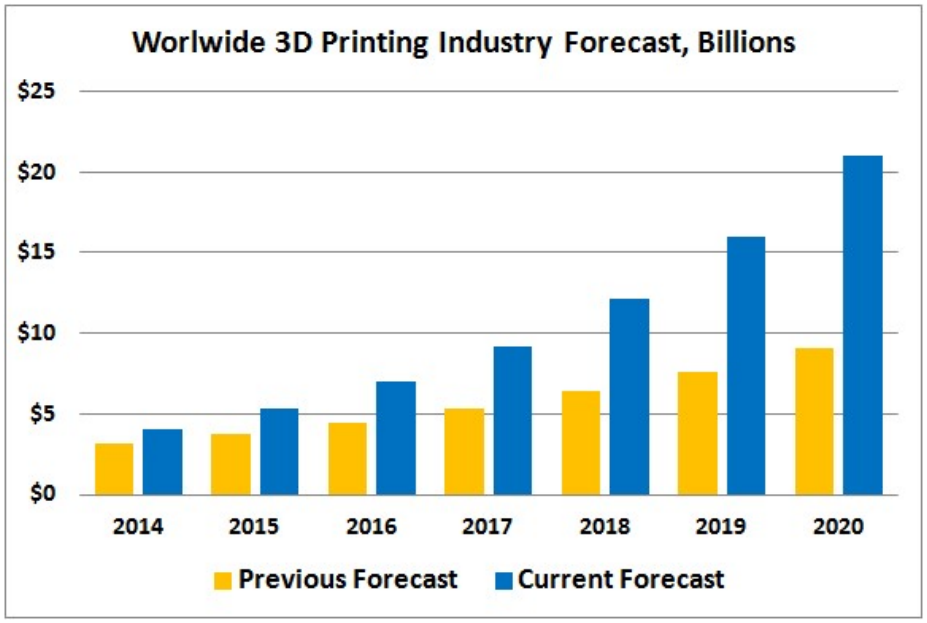
- **Parte 1 – Planejamento e Pesquisa de Mercado**: Aqui é realizada uma pesquisa de mercado e planejamento geral do projeto e seus objetivos.
- **Parte 2 – Estrutura do robô**: A estrutura básica é decidida e estudada, com embasamento teórico, design e protótipo da mesma.
- **Parte 3 – Cabeçote de impressão**: Aqui é projetado o cabeçote de impressão do material, sendo decidido o tipo de material a ser impresso, o projeto do sistema a imprimir esse material e incluindo os controles relacionados ao sistema, como controle de temperatura e extrusão do material.
- **Parte 4 – Construção do protótipo final**: O cabeçote de impressão e a estrutura da impressora seriam unificados nessa etapa, com ajustes sendo realizados para o sistema e um programa de controle geral sendo implementado.

Com isso decidido, esse estágio ficou responsável com as duas primeiras partes do projeto.

2.2 PESQUISA DE MERCADO

Uma rápida pesquisa de mercado foi feita com relação a impressoras 3D e sua disponibilidade no mercado local. É um mercado em crescimento continuo, como mostra a figura 1 retirada do artigo de março de 2015 da Forbes (Colombus, 2015).

Figura 01: Mercado de Impressão 3D (Colombus, 2015)



Atualmente, não existe no Brasil uma empresa local de fabricação ou mesmo distribuição de impressoras 3D. Isso demonstra uma demanda de mercado cada vez maior no país, e os custos de importação e imposto aplicados ao produto tem um impacto negativo na economia do país.

Na pesquisa de mercado de impressoras 3D, podemos classificar as impressoras pela tecnologia de seu método de impressão e pela sua estrutura física.

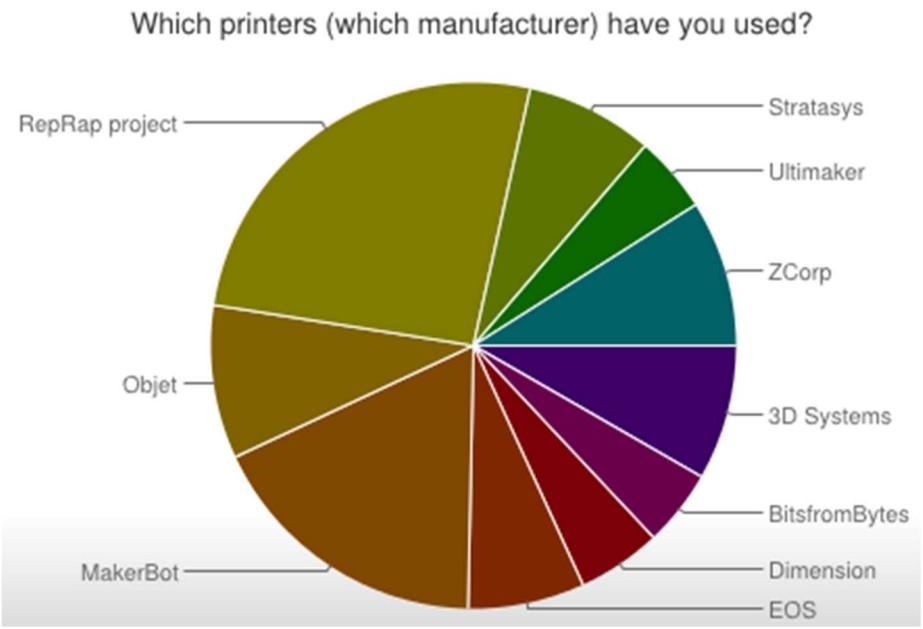
Os métodos de impressão incluem *Estereolitografia* (SLA), *Processamento Digital de Luz* (DLP), *Modelagem por Fusão e Deposição* (FDM), *Sinterização Seletiva de Laser* (SLS), *Fusão Seletiva a Laser* (SLM), *Fusão por Feixe de Elétrons* (EBM) e *Manufatura por Objetos Laminados* (LOM). Destes, o mais difundido, barato e de tecnologia mais simples é o FDM.

A estrutura por diversas vezes depende da tecnologia aplicada na impressão do material, mas a grande maioria das impressoras 3D do mercado atual caem nas categorias de Impressoras Cartesianas ou Impressoras Delta.

Das impressoras no mercado, destaca-se as impressoras da série RepRap^{TM}, por serem impressoras 3D de baixo custo, auto replicadora e de projeto aberto. A tecnologia aberta da série RepRap^{TM} serviu de base para a maioria das pesquisas desse projeto.

Na figura 2 observa-se uma distribuição do mercado de impressoras 3D datado em 2012.

Figura 02: Distribuição de mercado de impressoras 3D em 2012 (RepRap)



2.3 PLANEJAMENTO

Após o estudo de mercado local e mundial a respeito de Impressoras 3D, a próxima decisão de projeto foi qual o tipo de máquina a ser projetada e suas vantagens. Um estudo foi realizado tendo como base os modelos atuais do mercado, a utilização do robô em outras aplicações e o custo do produto final. A utilização de um robô em configuração paralela se mostrou a melhor escolha.

Nenhuma decisão final foi feita com relação ao tipo de material a ser impresso, mas as decisões do projeto de estrutura não levaram em consideração as consequências da estrutura para impressão por outro método além de FDM.

Nessa secção será abordado a diferença entre robôs em configuração serie e paralelo e em seguida uma comparação entre robôs do tipo delta é realizada para chegar a decisão final.

2.3.1 CONFIGURAÇÃO SÉRIE E PARALELO

Braços robóticos utilizados na indústria são classificados em duas categorias que incluem robôs seriais e robôs paralelos, ilustrado nas figuras 3 e 4. Quando comparados, robôs paralelos tem as vantagens de maior precisão, maior rigidez, uma capacidade de carga maior e uma baixa inercia (Young & Lin, 2016).

Figura 3: Robô de configuração série (Young & Lin, 2016).

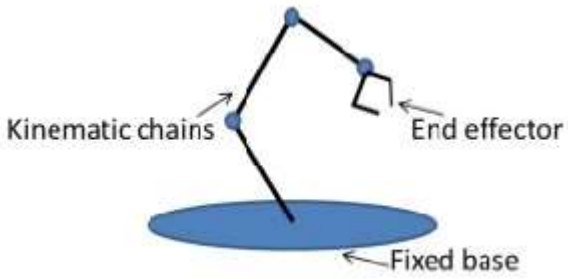
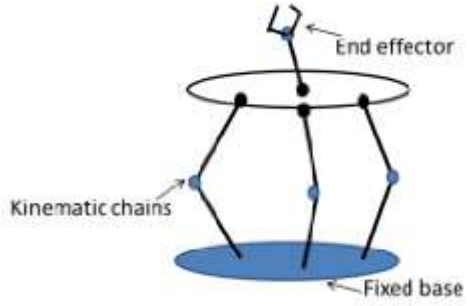


Figura 4: Robô em configuração paralelo (Young & Lin, 2016).

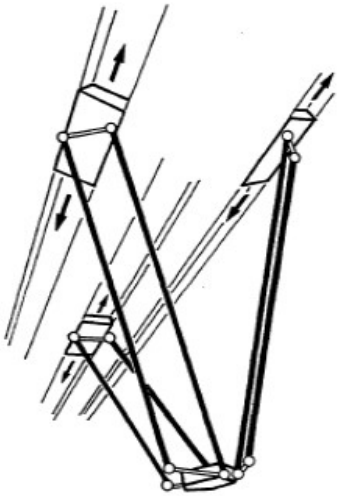


No campo de impressão 3D, os dois tipos de máquinas mais utilizadas são a Cartesiana e as de modelo Delta. Uma impressora cartesiana utiliza uma área quadrada para imprimir, e o movimento do seu cabeçote é decomposto em direções X, Y e Z separadas que são controladas individualmente por 3 motores. Já no modelo Delta os três braços são organizados em uma configuração triangular e a carga é distribuída entre os três motores, diminuindo a carga individual de cada um. Essa diferença influencia diretamente na velocidade e precisão da máquina (Young & Lin, 2016).

2.3.2 ROBÔ DELTA: LINEAR E ROTACIONAL

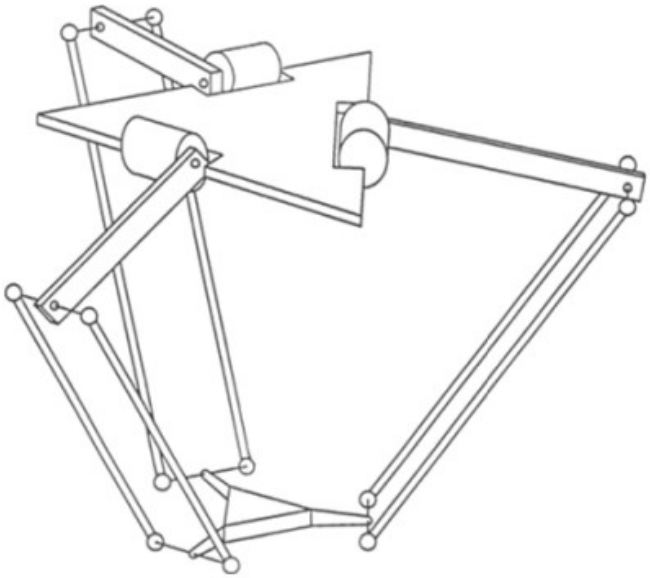
Robôs do tipo delta ainda podem ser classificados com relação as suas articulações, sendo separados em articulações lineares ou rotacionais, ilustrados nas figuras 5 e 6. No levantamento de mercado verificou-se a existência de impressoras 3D de ambos os tipos.

Figura 05: Ilustração de um robô delta linear horizontal (Bouri & Clavel, 2010)



Com base nos resultados verificados por (Bouri & Clavel, 2010), em uma comparação entre uma articulação do Delta Linear e uma do Delta Rotacional, seus comportamentos são praticamente iguais e sua sensitividade é praticamente idêntica, com uma pequena vantagem para o tipo Delta Rotacional e uma dinâmica mais rápida. Apesar disso, a não linearidade do modelo rotacional torna seu controle mais difícil devido a não linearidades na matriz jacobiana.

Figura 06: Ilustração de um robô delta rotacional (Bouri & Clavel, 2010)



Nesse ponto do projeto as possíveis aplicações da máquina fora do escopo de impressão 3D foram consideradas e, uma vez que a estrutura da máquina rotacional se mostrou mais adaptável a uma utilização no mercado como um manipulador robótico, esta foi escolhida quando comparada com a linear vertical.

3 EMBASAMENTO TEÓRICO

3.1 ROBÔ DELTA ROTACIONAL

A ideia de um braço robótico paralelo surgiu no início dos anos 80, proposta pelo Professor Reymond Clavel e sua equipe (Clavel & Rey, 1999). A principal função desse novo robô era a de manusear objetos leves a altas velocidades. Ele foi um marco na era de desenvolvimento de robôs industriais, podendo funcionar com acelerações de até 15G para operações de *pick and place* (Young & Lin, 2016). Sua robustez e alta velocidade o introduziu não apenas na indústria de empacotamento, mas também na indústria farmacêutica e de cirurgias médicas. Recentemente, como abordado nesse relatório, robôs delta rotacionais tem sido utilizados como impressoras 3D.

Nesse capitulo será abordado toda a fundamentação teórica estudada por trás do Robô Delta Rotacional, incluindo sua cinemática direta e inversa, sua cinemática de velocidade, sua dinâmica, sua área de trabalho teórica, e para o caso de funcionamento como manipulador, seu planejamento e controle de trajetória.

3.1.1 ESTRUTURA

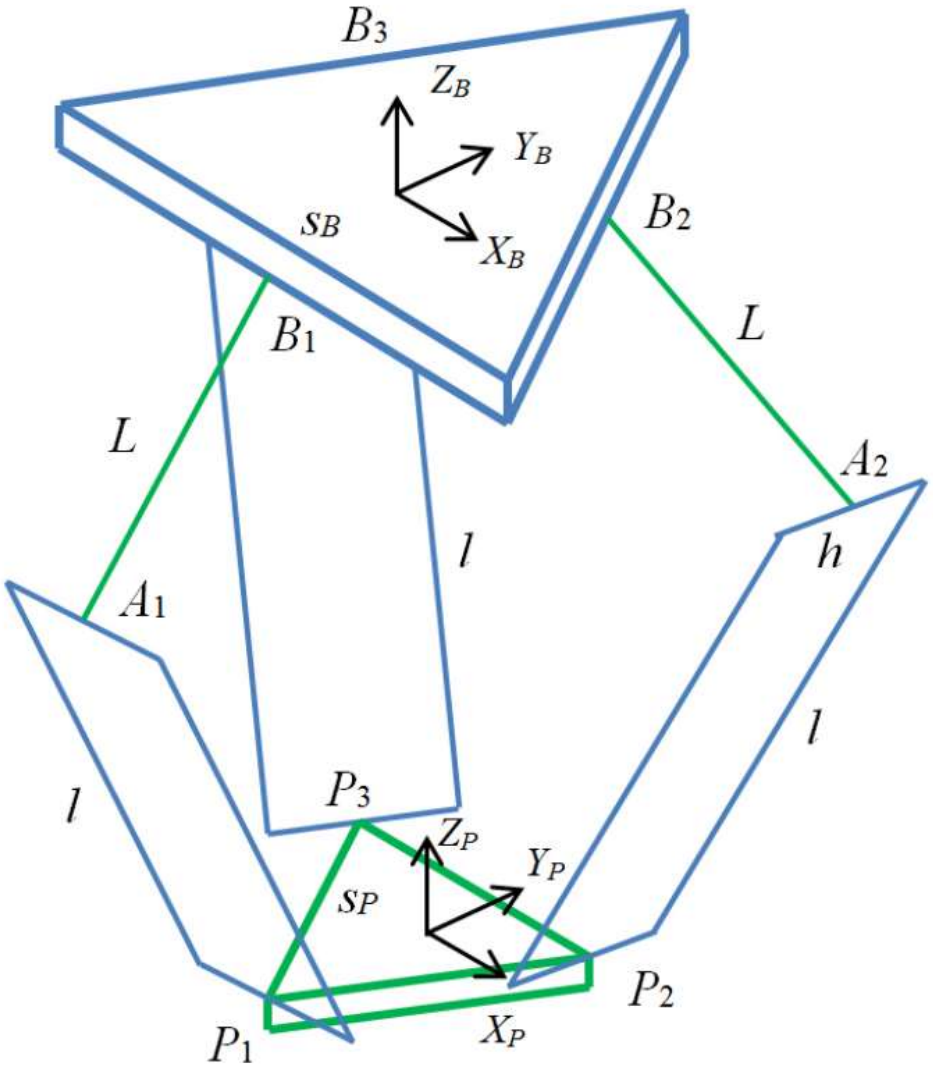
A estrutura geral de um robô delta rotacional é apresentada na figura 7. O conceito chave por trás do design do robô delta rotacional é o uso de paralelogramas na estrutura. A estrutura do paralelograma mantém a plataforma móvel sempre paralela a base fixa e provê apenas movimento translacional a plataforma nas direções x, y e z (Shareef, 2015).

No delta paralelo, os atuadores são montados na base fixa. As ligações dos três paralelogramas são conectados aos atuadores por articulações de revolução. A outra ponta dos paralelogramas é conectada a uma pequena plataforma triangular móvel que se move com três graus de liberdade. Normalmente, robôs Delta são robôs de direção direta e requerem mecanismos simples de controle se comparados com robôs de direção indireta. A potência de um motor à plataforma móvel é transmitida através de hastes rígidas em vez de por caixas de transmissão ou polias, o que faz o mecanismo mais preciso e menos ruidoso (Shareef, 2015).

Com todos os atuadores conectados a base fixa, os braços podem ser feitos de materiais leves para que os braços não tenham que carregar pesos extras aos atuadores. Esse mecanismo de movimentação rápida dá a vantagem de alta velocidade a robôs Delta e eles podem atingir altas acelerações (Shareef, 2015).

O estudo da estrutura foi realizado com base em (Williams II, 2016). No que se refere, a figura 7 abaixo apresenta o modelo do robô delta rotacional utilizado, onde pode-se associar os pontos $\boldsymbol{B_i}$, $i=1,2,3$ como o quadril, pontos $\boldsymbol{A_i}$, $i=1,2,3$ como joelhos, e pontos $\boldsymbol{P_i}$, $i=1,2,3$ como calcanhares da estrutura. O tamanho do lado do triângulo equilateral da base é S_B e o tamanho do lado da plataforma móvel é S_P. O triângulo da plataforma móvel é invertido com relação a base em orientação constante, como mostra a imagem.

Figura 07: Estrutura do robô delta rotacional (Williams II, 2016)



A referência cartesiana da base é **{B}**, cuja origem é localizada no centro do triângulo equilateral da base. A referência cartesiana da plataforma é **{P}**, cuja origem é localizada no centro do triângulo equilateral da plataforma. A orientação de **{P}** é sempre idêntica a orientação de **{B}** então a matriz de rotação ${}^{B}_{P}\boldsymbol{R} = \boldsymbol{I_3}$é constante. As variáveis das articulações são $\boldsymbol{\theta} = \{\theta_1\ \theta_2\ \theta_3\}^T$e as variáveis cartesianas são ${}^{B}\boldsymbol{P}_P = \{x\ y\ z\}^T$. A estrutura apresenta alta simetria com as três pernas superiores de tamanho L e as três inferiores com tamanho l.

Os pontos **B** da base são constantes com relação a base e os pontos **P** da plataforma são constantes com relação a plataforma:

$$ {}^{B}\boldsymbol{B}_1 = \begin{Bmatrix} 0 \\ -w_B \\ 0 \end{Bmatrix} \qquad {}^{B}\boldsymbol{B}_2 = \begin{Bmatrix} \frac{\sqrt{3}}{2}w_B \\ \frac{1}{2}w_B \\ 0 \end{Bmatrix} \qquad {}^{B}\boldsymbol{B}_3 = \begin{Bmatrix} -\frac{\sqrt{3}}{2}w_B \\ \frac{1}{2}w_B \\ 0 \end{Bmatrix} \qquad \textbf{(1)} $$

$$ {}^{P}\boldsymbol{P}_1 = \begin{Bmatrix} 0 \\ -u_P \\ 0 \end{Bmatrix} \qquad {}^{P}\boldsymbol{P}_2 = \begin{Bmatrix} \frac{S_P}{2} \\ w_P \\ 0 \end{Bmatrix} \qquad {}^{P}\boldsymbol{P}_3 = \begin{Bmatrix} -\frac{S_P}{2} \\ w_P \\ 0 \end{Bmatrix} \qquad \textbf{(2)} $$

Os vértices do triângulo equilátero da base fixa são:

$$ {}^{B}\boldsymbol{b}_1 = \begin{Bmatrix} \frac{S_B}{2} \\ -w_B \\ 0 \end{Bmatrix} \qquad {}^{B}\boldsymbol{b}_2 = \begin{Bmatrix} 0 \\ u_B \\ 0 \end{Bmatrix} \qquad {}^{B}\boldsymbol{b}_3 = \begin{Bmatrix} -\frac{S_B}{2} \\ -w_B \\ 0 \end{Bmatrix} \qquad \textbf{(3)} $$

Onde:

$$ w_B = \frac{\sqrt{3}}{6}S_B \qquad u_B = \frac{\sqrt{3}}{3}S_B \qquad w_P = \frac{\sqrt{3}}{6}S_P \qquad u_P = \frac{\sqrt{3}}{3}S_P \qquad \textbf{(4)} $$

A tabela resume as constantes utilizadas até o momento.

Tabela: Constantes do modelo Delta Rotacional utilizados

CONSTANTE	REPRESENTAÇÃO
$\boldsymbol{S_B}$	Lado do triângulo equilateral de base
$\boldsymbol{S_P}$	Lado do triângulo equilateral da plataforma
$\boldsymbol{L}$	Comprimento das pernas superiores
$\boldsymbol{l}$	Comprimento das pernas do paralelograma inferior
$\boldsymbol{h}$	Largura das pernas do paralelograma inferior
$\boldsymbol{w_B}$	Distância planar do {0} até a lateral da base
$\boldsymbol{u_B}$	Distância planar do {0} até o vértice da base
$\boldsymbol{w_P}$	Distância planar do {P} até a lateral da plataforma
$\boldsymbol{u_P}$	Distância planar do {P} até o vértice da plataforma

Com o diagrama cinemático apresentado, as seguintes equações são escritas para o robô Delta:

$$\{{}^{B}\boldsymbol{B}_i\} + \{{}^{B}\boldsymbol{L}_i\} + \{{}^{B}\boldsymbol{l}_i\} = \{{}^{B}\boldsymbol{P}_P\} + [{}^{B}_{P}\boldsymbol{R}]\{{}^{P}\boldsymbol{P}_P\} = \{{}^{B}\boldsymbol{P}_P\} + \{{}^{P}\boldsymbol{P}_i\} \qquad i=1,2,3 \qquad \textbf{(5)}$$

Onde $[{}^{B}_{P}\boldsymbol{R}] = [\boldsymbol{I}_3]$ uma vez que rotações não são permitidas no robô Delta.

Essas equações impõem que as pernas inferiores devem possuir o tamanho constante *l*:

$$l_i = \|\{{}^{B}\boldsymbol{l}_i\}\| = \|\{{}^{B}\boldsymbol{P}_P\} + \{{}^{P}\boldsymbol{P}_i\} - \{{}^{B}\boldsymbol{B}_i\} - \{{}^{B}\boldsymbol{L}_i\}\| \qquad i=1,2,3 \qquad \textbf{(6)}$$

$$l_i = \|\{{}^{B}\boldsymbol{l}_i\}\|^2 = l_{ix}^2 + l_{iy}^2 + l_{iz}^2 \qquad i=1,2,3 \qquad \textbf{(7)}$$

Os vetores de $\{{}^{B}\boldsymbol{L}_i\}$ são dependentes das variáveis de articulação $\boldsymbol{\theta}$:

$${}^{B}\boldsymbol{L}_1 = \begin{Bmatrix} 0 \\ -L\cos\theta_1 \\ -L\sin\theta_1 \end{Bmatrix} \qquad {}^{B}\boldsymbol{L}_2 = \begin{Bmatrix} \frac{\sqrt{3}}{2}L\cos\theta_2 \\ \frac{1}{2}L\cos\theta_2 \\ -L\sin\theta_2 \end{Bmatrix} \qquad {}^{B}\boldsymbol{L}_3 = \begin{Bmatrix} -\frac{\sqrt{3}}{2}L\cos\theta_3 \\ \frac{1}{2}L\cos\theta_3 \\ -L\sin\theta_3 \end{Bmatrix} \qquad \textbf{(8)}$$

Substituindo as equações acima nas equações escritas para o delta nos dá:

$${}^{B}\boldsymbol{l}_1 = \begin{Bmatrix} x \\ y + L\cos\theta_1 + a \\ z + L\sin\theta_1 \end{Bmatrix} \; {}^{B}\boldsymbol{l}_2 = \begin{Bmatrix} x - \frac{\sqrt{3}}{2}L\cos\theta_2 + b \\ y - \frac{1}{2}L\cos\theta_2 + c \\ z + L\sin\theta_2 \end{Bmatrix} \; {}^{B}\boldsymbol{l}_3 = \begin{Bmatrix} x + \frac{\sqrt{3}}{2}L\cos\theta_3 - b \\ y - \frac{1}{2}L\cos\theta_3 + c \\ z + L\sin\theta_3 \end{Bmatrix} \qquad \textbf{(9)}$$

Onde:

$$a = w_B - u_P$$

$$b = \frac{S_P}{2} - \frac{\sqrt{3}}{2} w_B \qquad \textbf{(10)}$$

$$c = w_P - \frac{1}{2} w_B$$

Por fim temos as três equações que definem o modelo cinemático do robô delta:

$$2L(y + a)\cos\theta_1 + 2zL\sin\theta_1 + x^2 + y^2 + z^2 + a^2 + L^2 + 2ya - l^2 = 0 \textbf{ (11)}$$

$$-L\left(\sqrt{3}(x + b) + y + c\right)\cos\theta_2 + 2zL\sin\theta_2 + x^2 + y^2 + z^2 + b^2 + c^2 + L^2 + 2xb + 2yc - l^2 = 0$$

$$-L\left(\sqrt{3}(x - b) - y - c\right)\cos\theta_3 + 2zL\sin\theta_3 + x^2 + y^2 + z^2 + b^2 + c^2 + L^2 - 2xb + 2yc - l^2 = 0$$

3.1.2 CINEMÁTICA INVERSA

A cinemática inversa do robô delta rotacional é aquela que vai transformar as coordenadas x, y e z do centro da plataforma móvel nas três variáveis de controle θ_1, θ_2 e θ_3 dos três atuadores. Ou seja, dado a posição cartesiana do ponto de referência da plataforma móvel ${}^{B}\boldsymbol{P}_P = \{x\ y\ z\}^T$, calculamos os três ângulos de revolução θ que nos dá essa posição. A solução de cinemática inversa do robô delta pode ser resolvida independentemente para cada uma das pernas do robô. Geometricamente cada perna da solução de cinemática inversa é uma intersecção entre um círculo de raio conhecido *L* com centro no ponto B e uma esfera de raio *l* centrada no ponto P.

A solução apresentada aqui, usada em (Williams II, 2016), é uma de análise das equações cinemáticas do robô Delta. As três equações escalares apresentadas são da forma:

$$E_i\cos\theta_i + F_i\sin\theta_i + G_i = 0 \qquad \text{i =1,2,3} \qquad \textbf{(12)}$$

Onde:

$$E_1 = 2L(y + a)$$

$$F_1 = 2zL \qquad \textbf{(13)}$$

$$G_1 = x^2 + y^2 + z^2 + a^2 + L^2 + 2ya - l^2$$

$$E_2 = -L\left(\sqrt{3}(x+b) + y + c\right) \qquad E_3 = -L\left(\sqrt{3}(x-b) - y - c\right)$$

$$F_2 = 2zL \quad \textbf{(14)} \qquad F_3 = 2zL \quad \textbf{(15)}$$

$$G_2 = x^2 + y^2 + z^2 + b^2 + c^2 + L^2 + 2xb + 2yc - l^2$$

$$G_3 = x^2 + y^2 + z^2 + b^2 + c^2 + L^2 - 2xb + 2yc - l^2$$

Essa equação é comumente conhecida em mecânica e é resolvida utilizando **Substituição da Tangente do Meio Ângulo**.

Se definirmos

$$t_i = \tan\frac{\theta_i}{2} \qquad \textbf{(16)}$$

Então:

$$\cos\theta_i = \frac{1-t_i^2}{1+t_i^2} \qquad \textbf{(17)}$$

$$\sin\theta_i = \frac{2t_i}{1+t_i^2} \qquad \textbf{(18)}$$

Substituindo na equação *EFG*, temos:

$$E_i\left(\frac{1-t_i^2}{1+t_i^2}\right) + F_i\left(\frac{2t_i}{1+t_i^2}\right) + G_i = 0 \qquad E_i(1-t_i^2) + F_i(2t_i) + G_i(1+t_i^2) = 0$$

$$(G_i - E_i)t_i^2 + (2F_i)t_i + (G_i + E_i) = 0 \qquad \textbf{(19)}$$

O que nos leva:

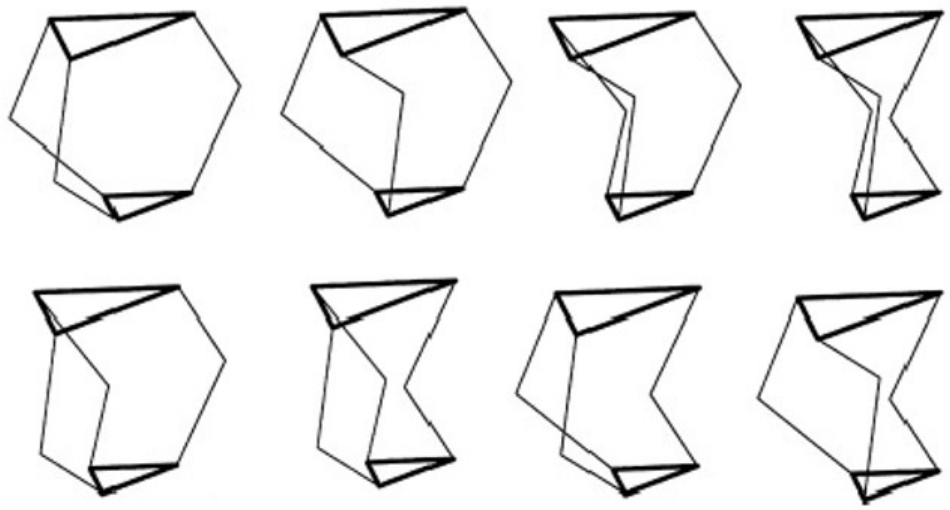
$$t_{i_{1,2}} = \frac{-F_i \pm \sqrt{E_i^2 + F_i^2 - G_i^2}}{G_i - E_i} \qquad \textbf{(20)}$$

Resolvemos para θ_i invertendo a fórmula original da tangente de meio ângulo.

$$\theta_i = 2\tan^{-1}(t_i) \qquad \textbf{(21)}$$

Duas soluções provem da fórmula quadrática. As duas soluções estão corretas uma vez que o joelho da estrutura pode se localizar para dentro ou para fora. Isso nos dá 8 possíveis soluções, como na figura 8 abaixo (Olsson, 2009). No entanto, a solução comumente utilizada é a com todos os joelhos para fora.

Figura 08: Possíveis soluções para a cinemática inversa (Olsson, 2009)



3.1.3 CINEMÁTICA DIRETA

A cinemática direta é aquela que determina a posição do (*x*, *y*, *z*) do centro da plataforma do manipulador com base nas entradas $\boldsymbol{\theta} = \{\theta_1\ \theta_2\ \theta_3\}^T$.

A análise da cinemática direta do Delta Rotacional foi realizada seguindo (Olsson, 2009) e adaptada as variáveis já especificadas na análise da estrutura.

Considere três esferas com o centro nos joelhos (**A)** de cada perna da estrutura, e tendo como raio o comprimento das pernas inferiores *l*. A cinemática direta do robô delta pode ser calculada pela intersecção das três esferas. Essas três esferas se interceptaram em dois locais. Em um deles a coordenada *z* é positiva e na outra negativa, apenas o ponto que se encontra dentro da área de trabalho representa a interseção correta.

A cinemática direta do robô delta rotacional é complicada, pois envolve resolver o sistema de equações não lineares definido pelas três esferas. A utilização de cinemática direta não é comum em aplicações com Delta Rotacional, sendo utilizada em sistemas avançados, como um braço cirúrgico Delta, acompanhado de um encoder de alta precisão como uma maneira de conseguir dados com relação a posição atual do mesmo.

De acordo com o estudo realizado, as três equações esféricas do tipo $(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 = r^2$ são definidas como[6]:

(22)

$$(x - [\cos(\alpha_1)(L\cos(\theta_1) + (w_B - u_P))])^2 + \left(y - \left[-\left((w_B - u_P) + L\cos(\theta_1)\right)\sin(\alpha_1)\right]\right)^2 + (z - [-L\sin(\theta_1)])^2 = l^2$$

$$(x - [\cos(\alpha_2)(L\cos(\theta_2) + (w_B - u_P))])^2 + \left(y - \left[-\left((w_B - u_P) + L\cos(\theta_2)\right)\sin(\alpha_2)\right]\right)^2 + (z - [-L\sin(\theta_2)])^2 = l^2$$

$$(x - [\cos(\alpha_3)(L\cos(\theta_3) + (w_B - u_P))])^2 + \left(y - \left[-\left((w_B - u_P) + L\cos(\theta_3)\right)\sin(\alpha_3)\right]\right)^2 + (z - [-L\sin(\theta_3)])^2 = l^2$$

Onde $\alpha = \{0, 120^o, 240^o\}$ são as orientações dos três braços.

3.1.4 CINEMÁTICA DE VELOCIDADE

A cinemática de velocidade é definida de acordo com (Williams II, 2016), onde se toma a primeira derivada das três equações de posição apresentadas anteriormente:

(23)

$$2L\dot{y}\cos\theta_1 - 2L(y+a)\dot{\theta}_1\sin\theta_1 + 2L\dot{z}\sin\theta_1 + 2Lz\dot{\theta}_1\cos\theta_1 + 2x\dot{x} + 2(y+a)\dot{y} + 2z\dot{z} = 0$$

$$-L(\sqrt{3}\dot{x} + \dot{y})\cos\theta_2 + L(\sqrt{3}(x+b) + y + c)\dot{\theta}_2\sin\theta_2 + 2L\dot{z}\sin\theta_2 + 2Lz\dot{\theta}_2\cos\theta_2 + 2(x+b)\dot{x} + 2(y+c)\dot{y} + 2z\dot{z} = 0$$

$$L\left(\sqrt{3}\dot{x} - \dot{y}\right)\cos\theta_3 - L(\sqrt{3}(x-b) - y - c)\dot{\theta}_3\sin\theta_3 + 2L\dot{z}\sin\theta_3 + 2Lz\dot{\theta}_3\cos\theta_3 + 2(x-b)\dot{x} + 2(y+c)\dot{y} + 2z\dot{z} = 0$$

Que podem ser reescritas como:

(24)

$$x\dot{x} + (y+a)\dot{y} + L\dot{y}\cos\theta_1 + z\dot{z} + L\dot{z}\sin\theta_1 = L(y+a)\dot{\theta}_1\sin\theta_1 - Lz\dot{\theta}_1\cos\theta_1$$

$$2(x+b)\dot{x} + 2(y+c)\dot{y} - L\left(\sqrt{3}\dot{x} + \dot{y}\right)\cos\theta_2 + 2z\dot{z} + 2L\dot{z}\sin\theta_2 = -L\left(\sqrt{3}(x+b) + y + c\right)\dot{\theta}_2\sin\theta_2 - 2Lz\dot{\theta}_2\cos\theta_2$$

$$2(x-b)\dot{x} + 2(y+c)\dot{y} + L\left(\sqrt{3}\dot{x} - \dot{y}\right)\cos\theta_3 + 2z\dot{z} + 2L\dot{z}\sin\theta_3 = L\left(\sqrt{3}(x-b) - y - c\right)\dot{\theta}_3\sin\theta_3 - 2Lz\dot{\theta}_3\cos\theta_3$$

Finalmente, essas equações são escritas em forma matricial:

$$[\mathbf{A}]\{\dot{\mathbf{X}}\} = [\mathbf{B}]\{\dot{\boldsymbol{\theta}}\}$$

$$\begin{bmatrix} x & x + a + L\cos\theta_1 & z + L\sin\theta_1 \\ 2(x+b) - \sqrt{3}L\cos\theta_2 & 2(y+c) - L\cos\theta_2 & 2(z + L\sin\theta_2) \\ 2(x-b) + \sqrt{3}L\cos\theta_3 & 2(y+c) - L\cos\theta_3 & 2(z + L\sin\theta_3) \end{bmatrix} \begin{Bmatrix} \dot{x} \\ \dot{y} \\ \dot{z} \end{Bmatrix} = \begin{bmatrix} b_{11} & 0 & 0 \\ 0 & b_{22} & 0 \\ 0 & 0 & b_{33} \end{bmatrix} \begin{Bmatrix} \dot{\theta}_1 \\ \dot{\theta}_2 \\ \dot{\theta}_3 \end{Bmatrix} \quad \textbf{(25)}$$

Onde:

$$b_{11} = L[(y+a)\sin\theta_1 - z\cos\theta_1]$$

$$b_{22} = -L[\left(\sqrt{3}(x+b) + y + c\right)\sin\theta_2 + z\cos\theta_2]$$

$$b_{33} = L[\left(\sqrt{3}(x-b) - y - c\right)\sin\theta_3 - 2z\cos\theta_3]$$

3.2 ÁREA DE TRABALHO

Ao dimensionar um manipulador ou impressora Delta, é essencial saber a área que o robô poderá acessar e, devido a geometria Delta, a precisão e resolução do mesmo em todas as camadas da área de trabalho. Durante as pesquisas e simulações realizadas, encontrou-se que para motores reais a resolução e precisão do robô varia com sua distância do centro da área de trabalho (ponto ótimo). Isso pode ser visto ao analisarmos a cinemática de velocidade e aceleração do sistema, onde verifica-se nas equações que as mesmas são dependentes da posição ${}^{B}\boldsymbol{P}_{P} = \{x\ y\ z\}^{T}$ do mesmo.

Simulações foram realizadas no Matlab para obter dados sobre essa variação de resolução quanto ao delta rotacional e a influência de cada braço em todas as direções. No entanto, não é de interesse do protótipo desse projeto encontrar uma solução ótima para o modelo, apenas aplicá-lo com sucesso. Simulações também foram feitas para calcular a resolução com base nos parâmetros do robô. A resolução no plano cartesiano vai depender diretamente do raio da base, do raio da plataforma e do tamanho do braço e antebraço do robô.

Além disso, simuladores encontrados na Internet dispõem de calcular a cinemática, resolução e dimensões de um robô Delta (Delta Robot Forward/Inverse Kinematics Calculations, s.d.) (Parallel Delta robot, s.d.). Mais sobre a área de trabalho e otimização podem ser encontrados em (Courteille, Deblaise, & Maurine, 2009), (Stan, Manic, Szep, & Balan, 2011), (Angel, Bermúdez, & Muñoz, 2013) e (Prempraneerach , 2014).

3.3 MOTOR DE PASSO

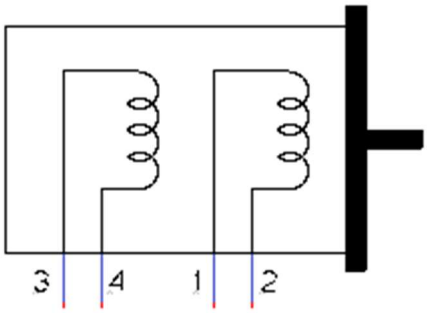
No levantamento do estudo sobre robôs Delta e impressoras 3D, encontrou-se que boa parte das impressoras 3D no mercado utilizam motores de passo em suas aplicações, por naturalmente possuírem um alto poder de controle de posição ao custo de um baixo torque quando comparado com motores DC.

Motores de passo são motores elétricos DC sem escova que dividem sua rotação completa em um número igual de passos. Um comando pode ser enviado ao motor para que ele se mova e mantenha sua posição sem que qualquer retorno de sensores seja

necessário, um controle em malha aberta, desde que a relação de torque e velocidade do motor seja devidamente calculada para a aplicação (Liptak, 2005).

Para tanto, os motores de passo possuem um número fixo de polos magnéticos que determinam o número de passos por revolução. Motores de passo utilizados na indústria de impressoras 3D normalmente são bipolares e possuem passos de 1.8 graus, totalizando 200 passos por revolução. Esses motores são compostos por um rotor, estator, duas bobinas, rolamentos e carcaça. Normalmente possuem enrolamentos em paralelo, que requerem mais corrente mas possuem uma menor indutância. Uma representação de um motor de passo bipolar de 4 fios se encontra na figura 9.

Figura 9: Representação de um motor de passo bipolar (Stepper motor, s.d.)



3.3.1 Circuito de Acionamento

A performance de um motor de passo está fortemente ligada a seu circuito de acionamento. As curvas de torque do motor podem ser estendidas para serem utilizadas em altas velocidades se a polaridade do estator for invertida rapidamente. Para superar a indutância dos enrolamentos mais rapidamente, a tensão de acionamento deve ser maior e é necessário limitar a alta corrente imposta pelo aumento dessa tensão.

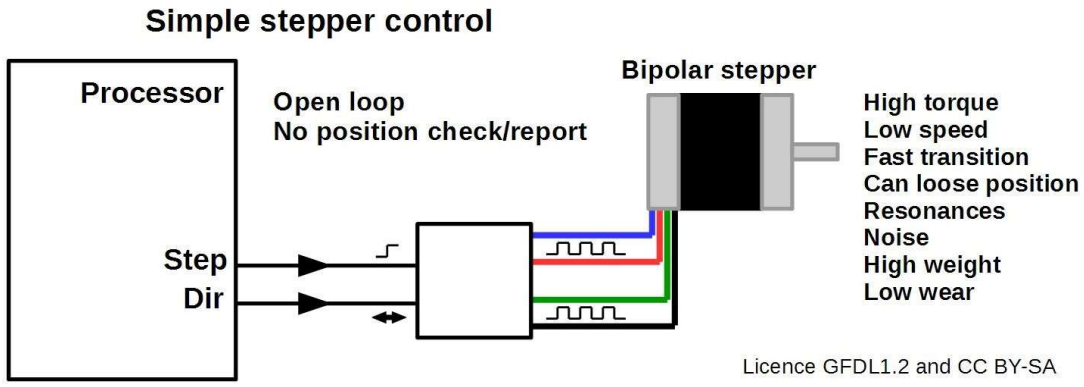
O estudo do circuito de acionamento de um motor de passos estava fora do escopo do projeto, uma vez que o objetivo final era um produto e não uma pesquisa de performance. Com a pesquisa de mercado realizada, os drivers **Polulu A4988** da figura 10, foram escolhidos para o projeto pela performance que apresentaram em outras plataformas de impressão 3D.

Figura 10: Drive de motor de passo Polulu A4988 (A4988)

3.3.2 CONTROLE DO MOTOR DE PASSOS

O controle mais utilizado para um motor de passos é o controle de malha aberta apresentado na figura 11, onde a direção da corrente nas bobinas é responsável pelo movimento de um passo do motor. Controlando a corrente de uma bobina com relação a outra permite um movimento mais refinado e preciso, dividindo os passos fixos em passos menores chamados micro passos. Esse controle, desde que a relação torque e velocidade seja respeitada, permite um sistema de posição precisa sem a necessidade de medir essa precisão (Motor Control Loop, s.d.).

Figura 11: Controle em malha aberta de motor de passo (Motor Control Loop, s.d.)



3.3.3 TEORIA DE MICRO PASSOS

Técnicas de micro passos são aquelas que utilizam de modulação PWM para controlar as correntes das duas bobinas e ter um controle ainda maior da posição do motor.

A variação das correntes nas duas bobinas do motor é o que faz o motor de passo girar. Uma das técnicas para visualizar isso é por um diagrama fasorial. Ao plotar a corrente de um enrolamento, I_a, em função do segundo enrolamento, I_b, podemos observar os fasores resultantes e sua relação com a rotação do motor. Em controles simples de motores de passo, por exemplo, as correntes em cada enrolamento variam entre I_{max}, 0 e $-I_{max}$, o que permite que o motor funcione apenas em operação de passo completo, meio passo ou passo de onda.

Nas figuras 12 e 13, podemos observar um motor funcionando em operação de passo completo. Em operação de passo completo, a corrente em cada enrolamento é I_{max} ou $-I_{max}$, e cada passo no diagrama representa um passo completo do motor, com uma sequência completa de quatro passos, uma vez que cada 90^o no diagrama de fase representa um passo completo do motor.

Assumindo o torque adequado, qualquer trajeto continuo que passe pelos quatro quadrantes do diagrama de fases com pelo menos um ponto por quadrante é suficiente para rotacionar o motor.

Figura 12: Diagrama de fases de motor de passo funcionando em passo completo (Microstepping Tutorial, s.d.)

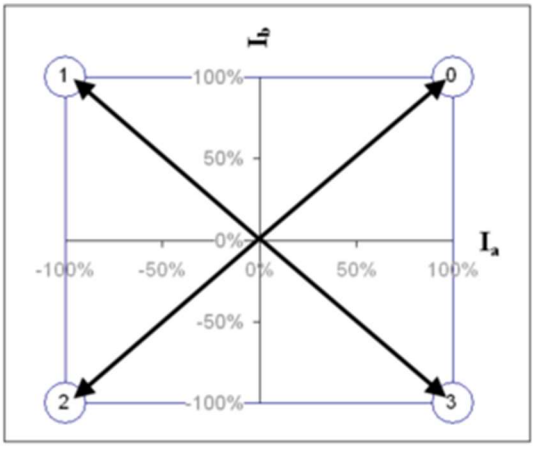
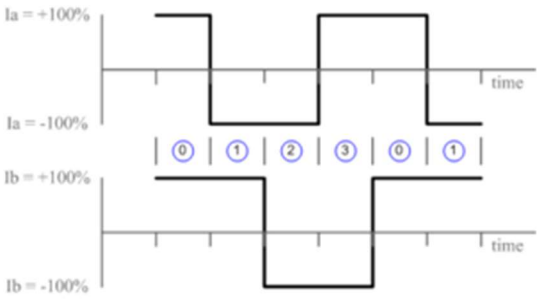


Figura 13: Corrente dos enrolamentos em função do tempo para operação a passo completo (Microstepping Tutorial, s.d.)



Se o controle for projetado com a capacidade de controlar a magnitude da corrente em cada enrolamento, pode-se implementar uma sequência de micro passos, como no exemplo das figuras 14 e 15.

Figura 14: Diagrama de fases de motor funcionando em micro passos seno cosseno (Microstepping Tutorial, s.d.)

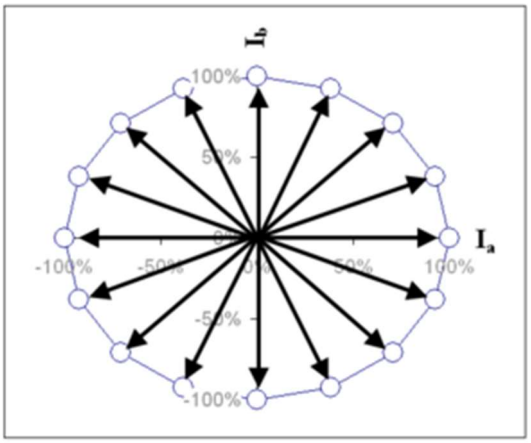
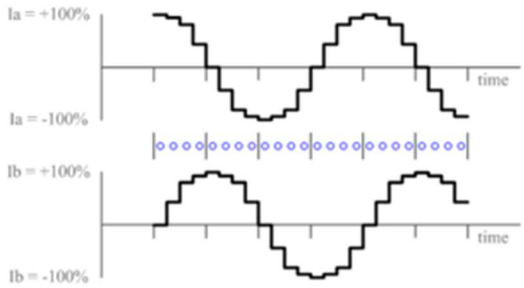


Figura 15: Corrente dos enrolamentos em função do tempo para operação em micro passos seno cosseno (Microstepping Tutorial, s.d.)



Como dito, a cada 90^o entre fasores no diagrama de fases, um passo completo é dado pelo motor. Qualquer fasor em um ângulo entre 0^o e 90^o representa uma posição

atingida por micro passo. Se, por exemplo, tomarmos como o início de um passo no eixo de I_a e o próximo passo iniciando em I_b, podemos representar o ângulo procurado para o micro passo como:

$$\theta = \tan^{-1}\left(\frac{I_b}{I_a}\right) \qquad \textbf{(26)}$$

A escolha da magnitude do fasor é decidida com base na potência necessária e no requisito de torque da aplicação, uma vez que a magnitude é igual a raiz da potência sobre a resistência do motor e o torque é diretamente proporcional a potência.

Apesar de ser chamativa a ideia de, por exemplo, utilizar 256 micro passos por passo completo de um motor de 1.8^o graus por passo para ter acesso a uma resolução de 51.200 micro passos por rotação completa do motor, é importante notar as limitações do sistema. Uma delas é a que o torque incremental cairá drasticamente quanto maior a quantidade de micro passos (Microstepping: Myths and Realities, s.d.). Outro problema é que o torque vai diminuir com o aumento de velocidade, já que a altas velocidades e um alto valor de micro passos, o motor vai girar mais rápido do que os comandos do PWM serão enviados aos enrolamentos (Parente, 2012).

3.4 PLANEJAMENTO DE TRAJETÓRIA

Robôs industriais são usados em automação contemporânea como o meio principal de aumentar a produtividade do sistema e a qualidade do produto. Eles têm sido amplamente utilizados em demandas de alta precisão e velocidade. Exemplos incluem atividades de soldagem, manuseio e colagem. A produtividade de tais robôs aumenta ainda mais com otimização de trajetória. Movimentação ótima aumenta a utilização geral do sistema e aumenta a sobrevida da máquina. Na otimização é determinada os sinais de controle que moverão o manipulador do ponto inicial ao final enquanto minimizam algum objetivo, como consumo de energia, tempo, etc. Normalmente a otimização de manipuladores robóticos é complexa graças as não linearidades e dinâmica complexa do sistema (Shareef, 2015).

Especificamente para a impressora 3D, planejamento e otimização de trajetória é calculada diretamente pelo CURA e transformado em código G a ser transformado em sinais de controle para o robô. No entanto, pensando na possibilidade futura de utilizar o

Delta Rotacional em outras aplicações, como um manipulador robótico ou dispositivo de soldagem, uma rápida pesquisa foi feita sobre o tema de forma a se deixar uma base para os próximos passos do projeto. Todas as técnicas estudadas aqui e algumas outras são melhor definidas em (Shareef, 2015).

Planejamento de trajetória é o primeiro passo da otimização de manipuladores robóticos. A maioria das técnicas de planejamento de rotas considera apenas a cinemática do robô e não considera sua dinâmica. No planejamento de movimento, as posições de parada do robô e o caminho entre essas posições devem ser bem definidas. O principal objetivo de planejamento de trajetória é evitar a colisão com obstáculos. O planejamento em si pode ser dividido então em planejamento de trajetória com ou sem obstáculos.

A rota do planejamento de trajetórias sem obstáculos é uma linha reta conectando o ponto inicial e o final. O principal desafio aqui é definir o perfil de velocidades ao longo dessa linha, que, no caso do manipulador delta, são as velocidades de cada um dos três motores no espaço das articulações para que o movimento no espaço cartesiano siga a linha definida.

No planejamento de trajetórias com obstáculos estáticos, diversas variáveis são envolvidas na hora do planejamento além de evitar colisões, como encontrar a mínima distância ao obstáculo, encontrar uma rota suave até o objetivo, etc. Já o planejamento de trajetória com objetos dinâmicos é mais complexo com relação ao planejamento com objetos estáticos, pois requerem mais trabalho em áreas como a modelagem do objeto em movimento, predição do movimento do objeto, etc. Usualmente ambientes e obstáculos são bem definidos no ramo da indústria e dificilmente há a interferência de objetos dinâmicos.

Técnicas de planejamento de trajetória usam conhecimento prévio, possuindo uma descrição completa e precisa do ambiente e obstáculos e normalmente essas técnicas são utilizadas off-line.

Entre as muitas técnicas utilizadas para planejamento de trajetória temos, por exemplo, *Probabilistic RoadMap* (PRM) e *Algoritmo Genético* (GA). É importante salientar que a literatura estudada trata apenas de planejamento de trajetória em duas dimensões, nos eixos cartesianos x e y. Exemplos numéricos e comparações de performance entre PRM e GA para um manipulador delta são encontradas em (Shareef, 2015).

3.4.1 PROBABILISTIC ROADMAP

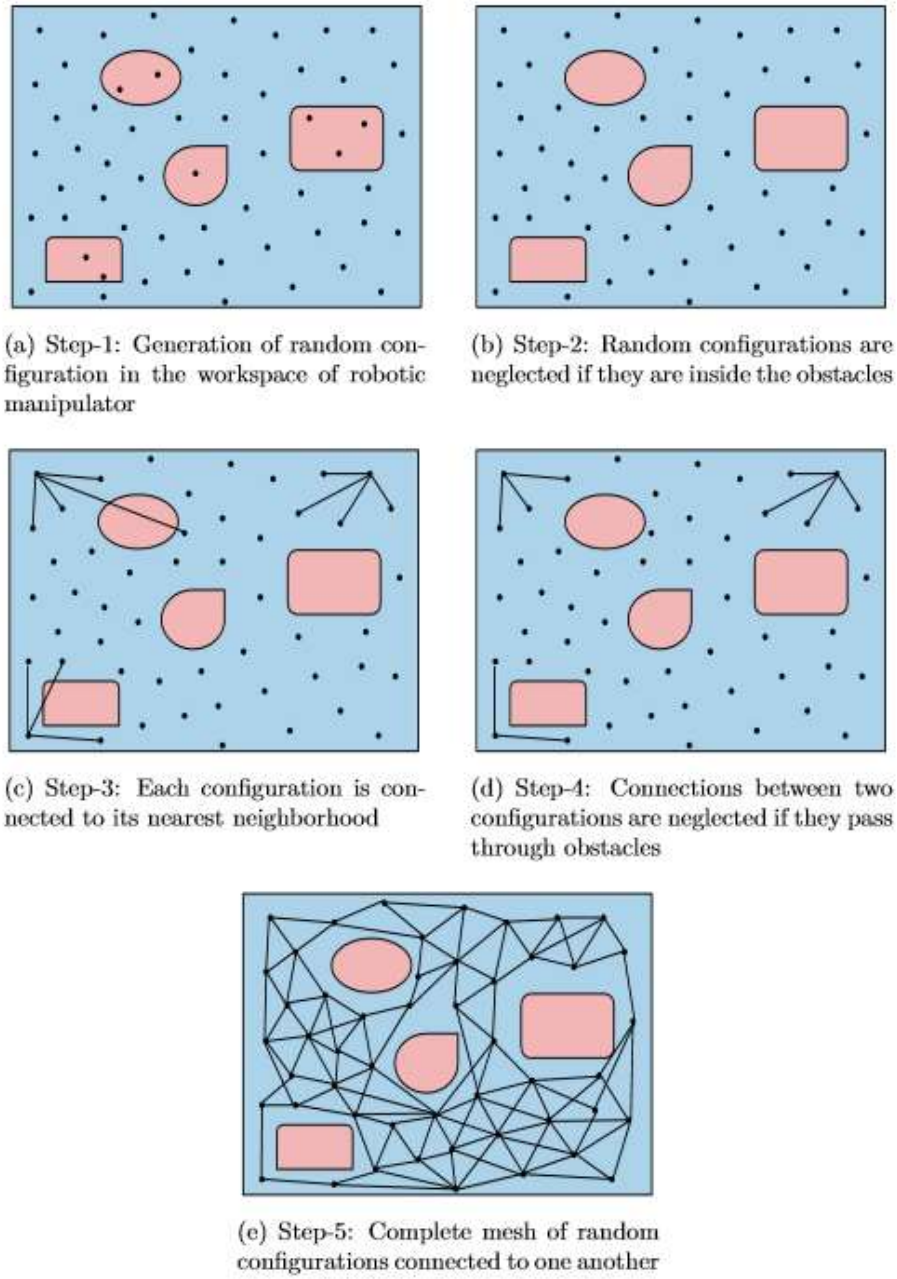
A medida que as dimensões e graus de liberdade do robô aumentam, PRM se destaca por ser um método eficiente para planejamento de rota em alta dimensionalidade. O método, proposto por (Kavraki, Švestka, Latombe, & Overmars, 1996), (Kavrakiand & Latombe, 1994), consiste de duas fases, uma de aprendizagem e outra de questionamento. Ela é genérica, fácil de ser implementada e pode ser aplicada a robôs com *n* graus de liberdade. PRM é um planejador completo, ou seja, sempre vai determinar uma solução ou determinar que nenhuma solução existe, e é computacionalmente veloz pois utiliza um planejador local relativamente fraco.

Na fase de aprendizagem, toda a área de trabalho permitida ao robô é explorada e configurações randômicas são inseridas dentro dessa área. Essa fase pode ser dividida nos seguintes passos:

- **Passo 1**: Uma configuração randômica é criada dentro da área de trabalho do manipulador delta. Um maior número de configurações vai aumentar a suavidade da trajetória ao custo de um incremento no tempo de computação.
- **Passo 2:** É verificado se alguma das configurações randômicas se encontra fora da área de trabalho permitida ou dentro da posição de objetos. As configurações assim verificadas são removidas.
- **Passo 3:** As configurações randômicas são conectadas as configurações vizinhas, normalmente as *k* configurações vizinhas que possuem distância menor que uma distância pré-determinada.
- **Passo 4:** Nessa etapa, as conexões de todas as configurações são verificadas para saber se estas estão passando por algum obstáculo, e estas são removidas.
- **Passo 5:** Na última etapa é obtida a rede completa de configurações randômicas permitidas na área de trabalho ligadas umas às outras.

A figura 16 mostra as diferentes etapas de aprendizagem e construção no PRM.

Figura 16: Fase de aprendizagem do PRM (Shareef, 2015)



(a) Step-1: Generation of random configuration in the workspace of robotic manipulator

(b) Step-2: Random configurations are neglected if they are inside the obstacles

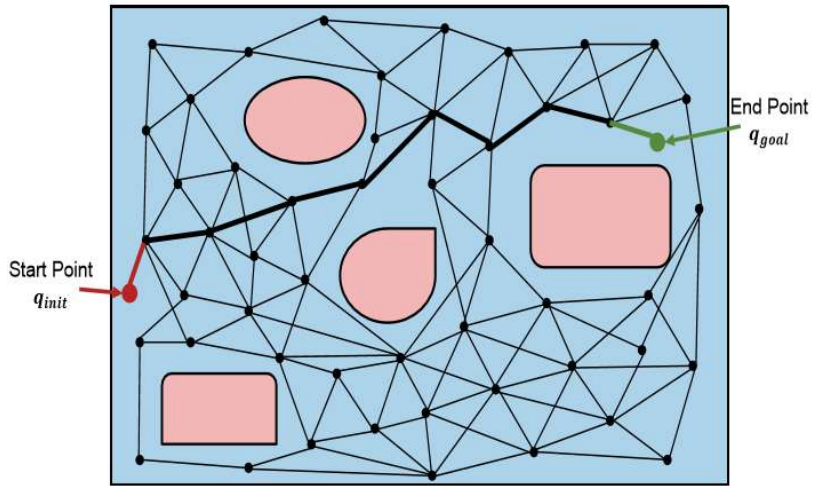
(c) Step-3: Each configuration is connected to its nearest neighborhood

(d) Step-4: Connections between two configurations are neglected if they pass through obstacles

(e) Step-5: Complete mesh of random configurations connected to one another

Na fase de questionamento, uma trajetória da posição inicial até a final na área de trabalho permitida é encontrada usando o mapa criado na fase de aprendizagem. Na primeira etapa da fase de questionamento, posições iniciais e finais são conectadas as configurações mais próximas. Na etapa seguinte, o algoritmo de Dijkstra (Dijkstra, 1959) é utilizado para encontrar o caminho mais curto da posição inicial a posição final. Uma representação da fase de questionamento é representada na figura 17.

Figura 17: Fase de questionamento do PRM (Shareef, 2015)



Como o próprio nome diz, PRM é um método probabilístico e é provável encontrar diferentes resultados a cada utilização. Para encontrar a melhor solução, PRM é utilizado múltiplas vezes e o melhor resultado é selecionado.

3.4.2 Algoritmo Genético

Do outro lado, como um dos representantes de controle inteligente aplicado a planejamento de rotas, tem-se planejamento de rotas por Algoritmo genético. Algoritmo genético é o mais popular tipo de algoritmo evolucionário e é amplamente utilizado em otimização. As limitações de algoritmo genético com relação a outros algoritmos de otimização são a sua tendência de convergir para ótimos locais e a dificuldade de sua utilização.

Algoritmo genético é uma técnica de procura global e paralela onde populações de candidatos a soluções, chamadas de indivíduos, evoluem em direção a uma solução melhor.

Uma população inicial deve ser definida. Geralmente em algoritmos genéticos essas soluções são determinadas randomicamente, mas nesse caso é possível que essas soluções incluam obstáculos em suas trajetórias. Uma forma de evitar isso é limitar a população inicial a uma faixa especifica para ter certeza que nenhum cromossomo gerado se encontra fora de soluções práticas. O número de iterações até uma solução ótima em algoritmo genético geralmente é reduzido quando se possui uma população inicial otimizada. No caso de algoritmo genético para planejamento de trajetória, a população inicial são as rotas aleatórias entre o ponto inicial e o final que não passam por obstáculos.

Cada trajetória é considerada um cromossomo e cada nó discreto na trajetória é considerada um gene.

Uma função de custo deve ser definida. O objetivo principal é achar a trajetória ótima entre a posição inicial e a final na presença de obstáculos com base em um critério. Diferentes critérios de otimização podem ser definidos, como tempo mínimo, energia mínima, menor distância, etc. Pela literatura (Shareef, 2015), a função de aptidão definida para algoritmo genético de planejamento de trajetória é definida como:

$$f = \sum_{i=1}^{n-1} d(P_0[i], P_0[i+1]) \qquad \textbf{(27)}$$

Onde:

$$d(P_0[i], P_0[i+1]) = \sqrt{(P_{0x}[i+1] - (P_{0x}[i])^2 + (P_{0y}[i+1] - (P_{0y}[i])^2}$$

Onde p_i é o gene de índice *i* do cromossomo, *n* é o tamanho do cromossomo, *d* é a distância entre dois nós consecutivos, e x_i e y_i são as coordenadas x e y do robô.

Em seguida, um método de seleção deve ser selecionado. A ideia em algoritmos genéticos para encontrar a solução ótima é a de que os melhores genes no cromossomo devem sobreviver e novas gerações de cromossomos devem evoluir dela. No procedimento de seleção, os melhores genes são selecionados da população. O processo de seleção constitui de três passos. Primeiro os cromossomos são avaliados com relação a função objetivo e valores são atribuídos a eles baseado na função de aptidão. Diferentes métodos existem para selecionar os melhores cromossomos, como seleção de Boltzman, seleção por torneio, seleção por classificação, etc.

No próximo passo, *crossover* ocorre entre dois cromossomos selecionados para formar dois filhos. Se não existe *crossover*, os filhos são cópias dos pais. A porcentagem de crossover entre dois cromossomos é escolhida, estando normalmente entre 80% e 90%. No caso de planejamento de trajetória utilizando algoritmo genético, o *crossover* mistura duas trajetórias diferentes e gera duas novas trajetórias. Condições especiais tem que ser aplicadas às trajetórias filhas para que estas também evitem obstáculos.

Em algoritmo genético, elitismo é o termo dado onde cópias dos cromossomos pais, normalmente os melhores na iteração, são mantidos após a nova geração de cromossomos ser gerada. Elitismo aumenta a performance do algoritmo genético uma vez que as melhores soluções são sempre mantidas de uma iteração para outra.

Por fim, após o crossover, mutações ocorrem na nova geração de cromossomos. Na mutação, um *bit* muda aleatoriamente no cromossomo filho ou uma pequena mudança ocorre no gene, a depender do código do cromossomo. Mutações expandem o espaço de pesquisa para regiões maiores para que todas as soluções não caiam em ótimos locais do problema, garantindo uma procura global. Normalmente a taxa de mutação deve ser baixa e a faixa recomendada é entre 0.5% a 1%.

3.4.3 OTIMIZAÇÃO DE TRAJETÓRIA

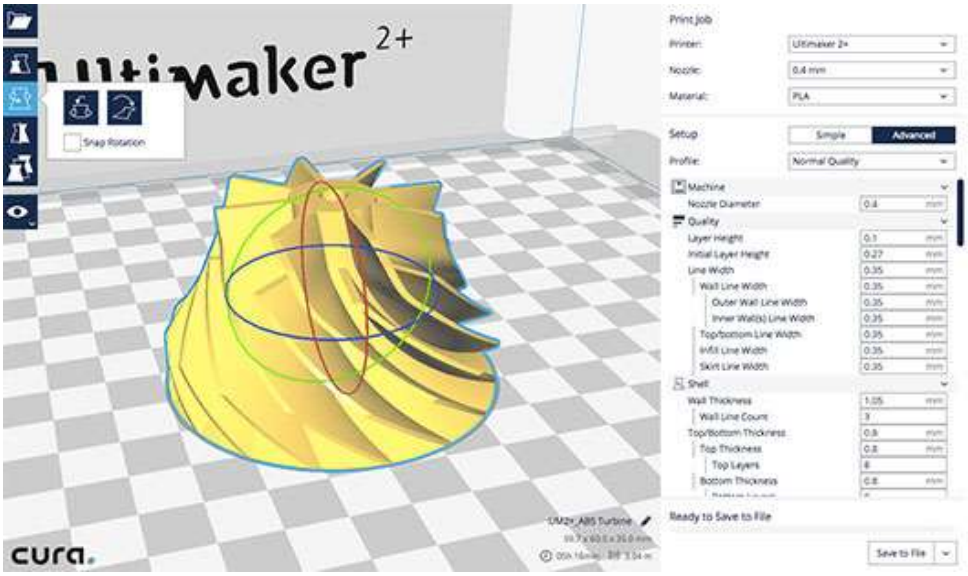
Otimização de trajetória é a segunda parte de otimização de rota onde um caminho geométrico pré-definido, obtido no planejamento de trajetória, é otimizado para um determinado objetivo. A importância de otimização de trajetória é óbvia do ponto de vista da relação entre execução da tarefa, produtividade e consumo de energia. A literatura estudada (Shareef, 2015) passa por um extensivo estudo histórico sobre a pesquisa na área desde os anos 1960 até sua utilização e pesquisa no mercado atual. Ela chega, por fim, aos métodos do *Plano de Fases* (PF), de *Programação Dinâmica* (PD) e *Controle Ótimo e Mecânica Discreta* (COMD) e faz uma comparação entre os três métodos em um manipulador delta.

3.5 CURA™

Cura™ é um software para impressoras 3D por extrusão de plástico de código aberto produzido e liberado gratuitamente pela Ultimaker que carrega arquivos STL, 3MF ou OBJ para serem imprimidos e automaticamente realizam o corte (*slicing*) do modelo e todo o planejamento de trajetória com base no modelo da impressora e transforma tudo isso em um arquivo em código G para ser interpretado pela impressora em questão.

O Cura™ possui um acervo de modelos de configuração de Impressoras 3D disponíveis no mercado e possui disponibilidade para introdução de modelos de terceiros. O software ainda conta com mais de 200 parâmetros ajustáveis para impressão da peça 3D, que estão fora do escopo de estudo dessa fase do projeto.

Figura 18: Interface do Cura (Cura 2, s.d.)



3.5.1 POSICIONAMENTO E TRAJETÓRIA DE UMA IMPRESSORA 3D

Como dito no estudo da área de trabalho de um manipulador delta, é possível chegar a resoluções bem pequenas com um bom controle de micro passos, principalmente próximo do centro de operação do manipulador. No entanto, por menor que seja essa resolução, esse espaço de trabalho ainda é discreto, e aproximações tem que ser feitas com relação a posição final da plataforma móvel ao fim de uma trajetória.

Se a trajetória de impressão de um modelo 3D fosse definida ponto a ponto, essas aproximações, apesar de pequenas, iam atribuir um erro incremental a cada ponto que deformaria o modelo 3D a ser imprimido. Graças a isso, um estudo no software do Cura™ demonstrou que o planejamento de trajetória da impressão 3D é feito com uma série de caminhos definidos apenas pelo ponto inicial e final, sem a necessidade de o robô parar ou calcular corretamente a posição de todos os pontos do caminho. Em outras palavras, após os pontos da trajetória serem definidos, a cinemática de velocidade a aceleração da impressora em questão é utilizada pela otimização de trajetória para garantir que a plataforma móvel passe exatamente por aquele caminho em vez de passar ponto a ponto. Isso permite uma quantidade de aproximações muito menor e uma deformação do modelo original praticamente inexistente para estruturas com poucos ou nenhum trajeto circular.

3.5.2 IMPRESSORAS CARTESIANAS E ROTACIONAIS

Durante a duração desse projeto de estágio, a utilização de impressoras rotacionais na versão atual do Cura (2.1.3) ainda não havia sido implementada, então foi realizado um estudo dos parâmetros do Cura™ e impressão 3D, como taxa de extrusão de material, sensores de temperatura, rigidez da estrutura, formas da estrutura. Essa parte especifica do estudo realizado não interfere com o planejamento do projeto, e pode ser conferida diretamente no manual do Cura.

A diferença nos cálculos de velocidade e posições entre uma impressora cartesiana e uma rotacional torna inviável a utilização do Cura sem o *plugin* de acesso a impressoras rotacionais. Uma solução viável, ao termino da bateria de testes do protótipo do projeto, é a criação de um perfil da impressora para versões antigas do Cura.

3.5.3 MODELO DE IMPRESSORA

Uma vez que a versão atual do Cura™ permite apenas a implementação de impressoras cartesianas, grande parte dos parâmetros de implementação de um novo modelo de impressora dizem respeito ao cabeçote de impressão, como temperatura, taxa de extrusão, velocidade de extrusão, diâmetro da abertura da ponta do cabeçote, etc.

Os únicos parâmetros com relação a estrutura da impressora são os torques dos motores, suas resoluções e a área de trabalho quadrada da impressora.

3.5.4 CÓDIGO G

Código G é um nome comum para a linguagem de programação que comanda máquinas operatrizes de Controle Numérico Computadorizado (CNC). Usado principalmente na indústria de automação, ela faz parte do ramo da engenharia assistida por computador. Foi desenvolvido no início da década de 1960 pela Aliança das Indústrias Eletrônicas, e uma revisão final foi aprovada em fevereiro de 1980 como RS274D (Silva, 2002).

CNC é um sistema que permite o controle de máquinas, sendo utilizado principalmente em tornos e centros de usinagem. Ele permite o controle simultâneo de vários eixos, através de uma lista de movimentos escrita num código específico.

Usualmente um programa CNC consiste de algumas poucas linhas para um trabalho simples até milhares de linhas, no caso de uma impressão 3D, para um trabalho maior. Programas CNC normalmente iniciam com palavras N no começo de cada linha ou a cada conjunto de linhas. O propósito da palavra N é a de identificação do número da linha ou do bloco de código. Além de comandos N, comandos G e M formam a grande maioria dos comandos de um código CNC. Fora isso, a interpretação de código G segue um padrão, mas muitos comandos não definidos pelo padrão são definidos especificamente para uma máquina (Silva, 2002). Para o caso do Cura, a fatiamento e o planejamento de trajetória da impressão vai ser transformada para um arquivo de código G, normalmente com dezenas de milhares de linhas, que utiliza código G para determinar a posição inicial da trajetória, a posição final da trajetória, a velocidade com a qual a impressora deve percorrer aquela trajetória, a velocidade final naquele ponto, em quanto tempo ela deve percorrer aquela trajetória, e dados para a extrusão de material do cabeçote, como velocidade de extrusão, tempo de extrusão, aquecimento do material, etc.

3.5.5 CONTROLE DO SISTEMA

Com o código G pronto, portanto, a impressora 3D interpreta o código e transforma as informações sobre velocidade e posição usando as equações de cinemática inversa, cinemática de velocidade e cinemática de aceleração para descobrir a variação em cada ângulo dos motores de passo e com qual velocidade esses ângulos devem mudar, para em seguida transformar isso nos códigos PWM enviados para o circuito de acionamento dos atuadores.

4 SIMULAÇÕES E TESTES

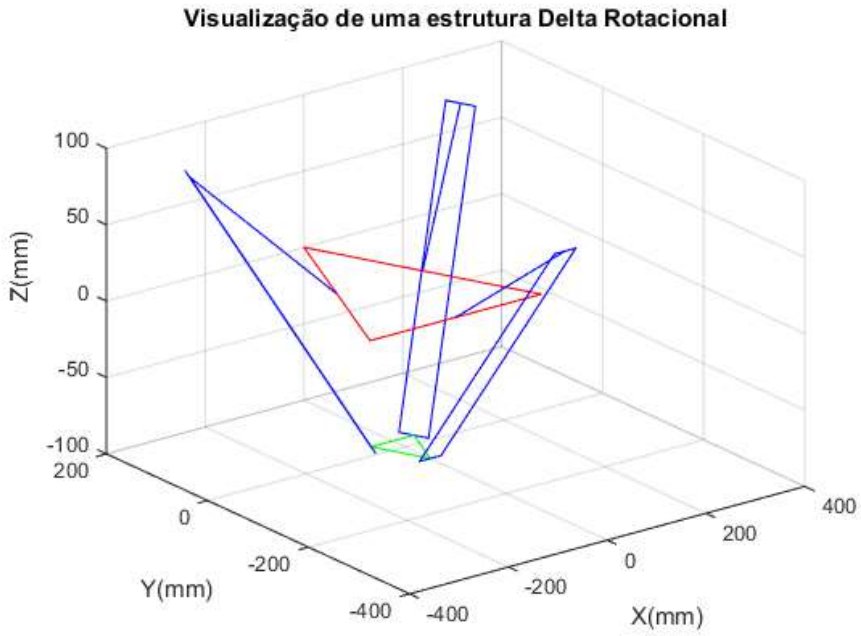
Simulações e testes foram desenvolvidos ao longo dessa etapa do projeto para validar o conhecimento adquirido com o material disponível. As simulações foram realizadas no Matlab e o teste do controle do motor de passo foi feito utilizando um Arduino Uno, um driver Polulu A4898 e um motor de passo Nema 17. Os códigos utilizados para todas as simulações e testes se encontram nos apêndices A e B.

4.1 MATLAB

Foram implementadas no Matlab as soluções para a cinemática inversa e direta estudadas no embasamento teórico. Para tanto, os parâmetros do robô delta foram determinados através de ferramentas encontradas na web (Delta Robot Forward/Inverse Kinematics Calculations, s.d.) (Parallel Delta robot, s.d.), onde procurou-se por meio ADHOC achar a melhor resolução para a área de trabalho procurada, um cubo de 30 cm de lado. Os valores encontrados por esse meio não condizem com os valores otimizados sugeridos por Clavel em (Bouri & Clavel, 2010) e não representam os valores finais do protótipo da estrutura.

Para uma visualização geral do sistema, uma ferramenta gráfica simples e básica foi criada para visualizar a posição atual do manipulador Delta bom base nos ângulos dos três atuadores e na posição cartesiana da plataforma. Por fim, diversos testes foram feitos com a área de trabalho com base nos parâmetros adotados. Uma imagem dessa visualização pode ser vista na figura 19.

Figura 19: Visualização da Estrutura Delta (Autor)



4.1.1 Cinemática Inversa

O cálculo da cinemática inversa no Matlab é direto. As únicas observações aqui são a manipulação de números complexos realizada pelo Matlab e o código utilizado para sempre obter o ângulo onde o joelho do manipulador Delta está para fora. Nas figuras 20 e 21 é possível observar o gráfico de como os três ângulos dos atuadores variam quando as funções dos valores x, y e z são senos deslocados no tempo.

Figura 20: Translocação teste no Espaço Cartesiano (Autor)

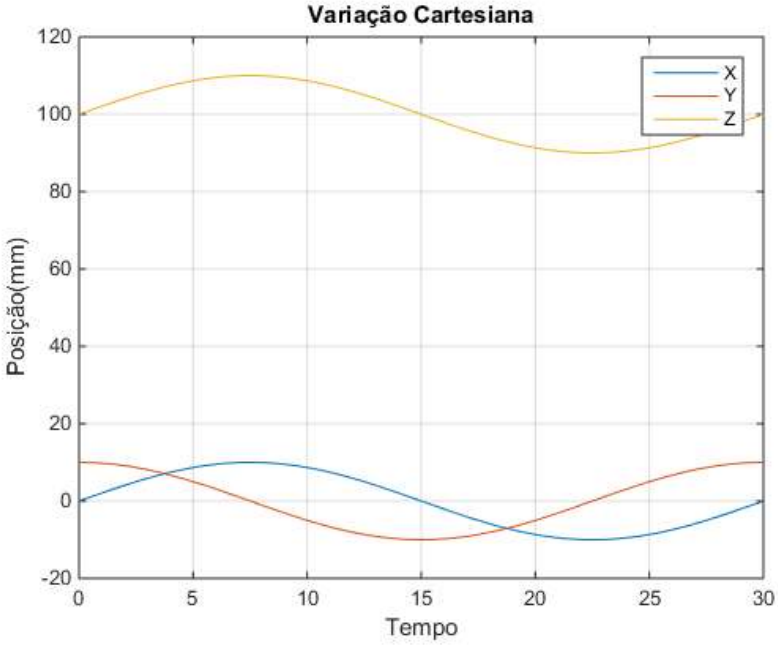
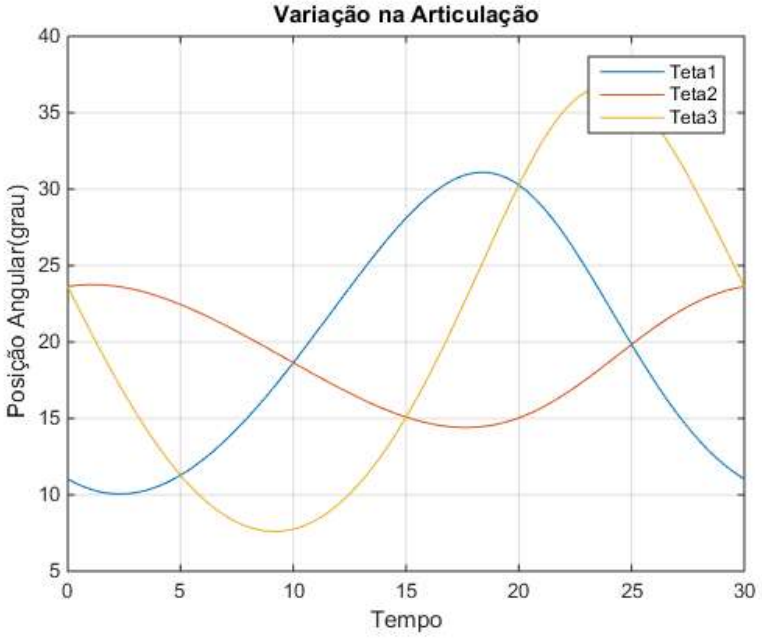


Figura 21: Translocação teste no Espaço Angular (Autor)



4.1.2 Cinemática Direta

Como dito na literatura (Williams II, 2016), o cálculo da cinemática direta é computacionalmente mais complexo que o da inversa. Nas aplicações como impressora 3D a cinemática direta raramente é utilizada devido a sua carga computacional. A cinemática direta é normalmente utilizada para se medir a posição do Delta rotacional quando não se está sendo utilizado motores de passo. Este cálculo é feito no Matlab através de da função *fsolve* para resolver o sistema de três equações não lineares.

4.1.3 Área de Trabalho

O estudo da área de trabalho não é o foco desse projeto, mas importantes informações podem ser retiradas com uma avaliação rápida do mesmo. Entre os dados procurados foram o ponto ótimo de trabalho, onde a resolução é máxima, e como essa resolução varia no eixo z. Para fazer uma análise precisa, uma simulação longa e computacionalmente pesada teria que ser feita com a cinemática direta, e, por falta de recursos computacionais, apenas um código simples utilizando a cinemática inversa foi utilizado como observação da área de trabalho.

Outro ponto importante a observar com uma simulação de dados da área de trabalho é a influência de cada braço na movimentação do plano (x,y). Por exemplo, no movimento em uma linha tangente a um dos braços, a influência daquele atuador na posição final daquela linha é virtualmente nula. Em contraparte, a influência na linha normal ao braço do atuador é máxima.

É possível observar isso analiticamente ao analisarmos a cinemática de velocidade do robô delta rotacional.

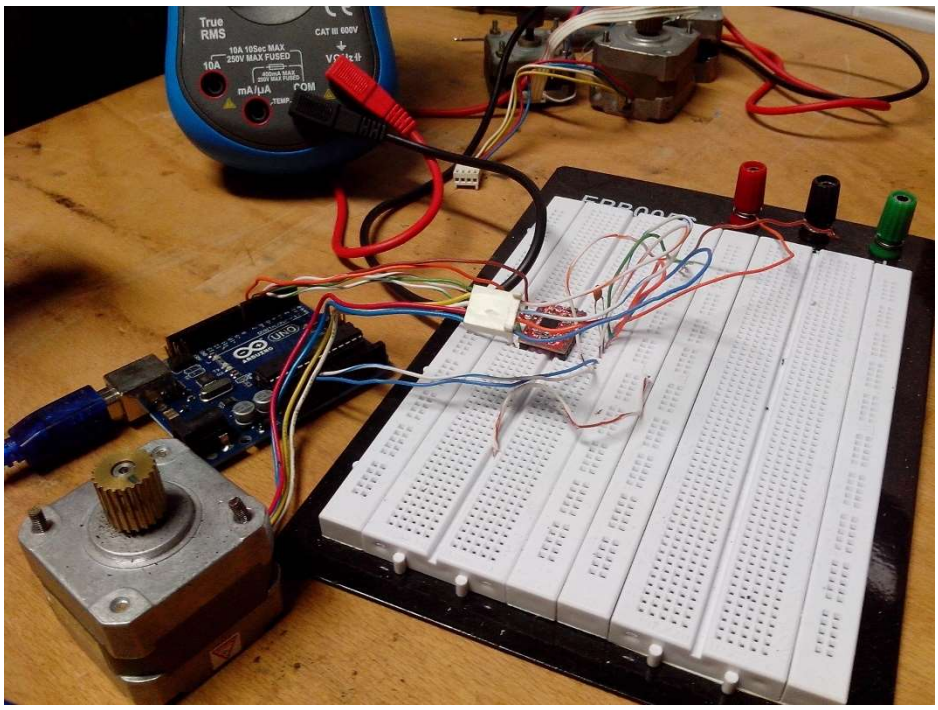
4.2 Arduino

Antes da chegada do material selecionado para construção do protótipo, um teste foi realizado utilizando o material já disponível no laboratório, um Arduino Uno, um motor Nema 17 e um Polulu A4988. O teste físico foi realizado para observação prática da aplicação da cinemática inversa e o controle de posição do motor de passo.

4.2.1 CIRCUITO DE CONTROLE DO MOTOR DE PASSO

O circuito do motor de passo com o Arduino Uno foi esquematizado com o software Fritzing. Um modelo para o motor Nema 17 foi criado para o software. Mais informações do funcionamento do Polulo A4988 podem ser encontradas no site da Polulo e no datasheet do A4988. O Polulo A4988 foi configurado para operar em 16 micro passos por passo. Uma imagem do circuito montado pode ser vista na figura 22.

Figura 22: Circuito para teste da cinemática inversa do motor de passo (Autor)



4.2.2 CÓDIGO DO ARDUINO

Simples em sua utilização, o Arduino não inclui bibliotecas para fazer manipulações matemáticas complexas. Uma comprovação disso foi a inexistência de uma biblioteca para manipulação de números complexos no mesmo, sendo necessária a busca de uma biblioteca em terceiros.

Apesar disso, a utilização da cinemática inversa no Arduino foi realizada com sucesso, e, uma vez que o motor não possuía carga nenhuma, sua relação de velocidade e torque foi respeitada e ele se posicionou com sucesso nos pontos de teste.

5 PRÓXIMOS PASSOS

O material escolhido para construção do protótipo foi o mesmo que acompanha uma RepRap, uma vez que o levantamento do material escolhido peça por peça chegava a até 75% do produto de uma RepRap™ pronta. Esse material inclui cinco motores Nema 23, Polulu A4898 para cada motor e um Arduino Mega 2560, além da lista de peças que acompanha o RepRap.

Infelizmente, devido a um atraso no projeto, o material chegou no laboratório no dia 17 de outubro, três dias após o término do estágio.

No entanto, o andamento do projeto até o fim do estágio permitiu levantar os seguintes passos para o término da segunda parte do projeto:

- Uma vez que o controle do motor de passo em carga livre com micro passos de 16 micro passos por passo foi realizado com sucesso, o próximo passo, com o modelo CAD pronto da impressora, é calcular a carga total do sistema e verificar se essa se encontra na faixa de torque do motor. No entanto, como já foi estudado que a carga é distribuída ao longo dos três motores, dificilmente essa carga total excederá o limite e uma faixa de segurança, por exemplo.
- O próximo passo para a utilização do robô como impressora 3D é a criação do modelo para Cura, seja a versão atual ou uma anterior.
- Após isso, a próxima grande etapa é a criação de um código que interprete o código G gerado e consiga transformar isso em movimento dos motores utilizando as dinâmicas inversa e de velocidade.
- Testes e ajustes finais.

Para a utilização como manipulador, no entanto, em vez da geração de um código que interprete o código G é necessária a criação de um código que crie esse código G ao utilizar um planejador de trajetória e as cinemáticas inversa, de velocidade e de aceleração.

6 CONCLUSÃO

O principal benefício do estágio realizado no laboratório de automação do Senai Stenio Lopes foi a oportunidade de ser responsável direto por um projeto de engenharia tão cedo na carreira. As decisões e análises realizada, e a necessidade de aprender e estudar algo completamente novo por conta própria me engrandeceram como profissional. Os estudos realizados sobre a estrutura, motores de passo e planejamento de trajetórias se mostraram fundamentos que serão utilizados futuramente em outras aplicações.

Como país em desenvolvimento, o brasil apresenta ampla oportunidade de aplicações de automação e a versatilidade de um manipulador Delta rotacional o torna um forte candidato a aplicações na indústria local.

O projeto em si se mostrou mais desafiador do que esperado no que se refere a sua utilização como manipulador Delta. Quanto a impressora 3D, o desafio de transformação do código G não era esperado e terá que ser levado em conta nos próximos passos do projeto.

O cenário de impressão 3D cresce em ritmo acelerado no resto do mundo, e esse projeto, apesar de ainda rústico, se torna um bom passo para a criação de um mercado local que se aproveite desse cenário.

REFERÊNCIAS

A4988. (s.d.). Fonte: Polulo: https://www.pololu.com/file/0J450/A4988.pdf

ABNT. (2003). NBR 6028 - Informação e documentação - Resumo - Apresentação. (p. 2). Associação Brasileira de Normas Técnicas.

ABNT. (30 de 12 de 2011). NBR 14724 - Informação e documentação — Trabalhos acadêmicos — Apresentação. *Associação Brasileira de Normas Técnicas* (p. 11). ABNT.

Angel, L., Bermúdez, J., & Muñoz, O. (2013). Dynamic Optimization and Building of a Parallel Delta-Type Robot. *IEEE International Conference on Robotics and Biomimetics (ROBIO).* Shenzhen.

Bouri, M., & Clavel, R. (2010). The Linear Delta: Developments and Applications . *ISR / ROBOTIK* .

Clavel, R., & Rey, L. (1999). The Delta Parallel Robot. Em C. R. Boër, L. Molinari-Tosatti, & K. Smith, *Parallel Kinematic Machines* (pp. 401- 417). Londres.

Colombus, L. (31 de Março de 2015). *2015 Roundup Of 3D Printing Market Forecasts And Estimates*. Fonte: Forbes: http://www.forbes.com/sites/louiscolumbus/2015/03/31/2015-roundup-of-3d-printing-market-forecasts-and-estimates/#427f9cb91dc6

Courteille, E., Deblaise, D., & Maurine, P. (2009). Design Optimization of a Delta-Like Parallel Robot through Global Stiffness Performance Evaluation. *The 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems.* St. Louis.

Cura 2. (s.d.). Fonte: Ultimaker: https://ultimaker.com/en/resources/20406-installation-cura-2-1

Delta Robot Forward/Inverse Kinematics Calculations. (s.d.). Fonte: Marginally Clever Software 2012: https://www.marginallyclever.com/other/samples/fk-ik-test.html

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik*, 269-271.

Kavraki, L., Švestka, P., Latombe, J. C., & Overmars, M. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *IEEE Transactions on Robotics and Automation.*

Kavrakiand, L., & Latombe, J. C. (1994). Randomized preprocessing of configuration for fast path planning. *IEEE International Conference on Robotics and Automation.*

Liptak, B. G. (2005). *Instrument Engineers' Handbook, Fourth Edition, Volume Two: Process Control and Optimization.* CRC Press.

Microstepping Tutorial. (s.d.). Fonte: Zaber Wiki: http://www.zaber.com/wiki/Tutorials/Microstepping

Microstepping: Myths and Realities. (s.d.). Fonte: Micromo: http://www.micromo.com/microstepping-myths-and-realities

Motor Control Loop. (s.d.). Fonte: RepRap: http://reprap.org/wiki/Motor_control_loop

Olsson, A. (Fevereiro de 2009). Modeling and control of a Delta-3 robot . *Dissertação*.

Parallel Delta robot. (s.d.). Fonte: Automation, Robotics and Computer Vision lab (ARCV): http://arvc.umh.es/label/delta.html

Parente, B. (25 de Julho de 2012). *The dynamic performance of stepper motors*. Fonte: Schneider Electric Motion USA: http://motion.schneider-electric.com/technology-blog/the-dynamic-performance-of-stepper-motors/

Prempraneerach , P. (2014). Workspace and Dynamic Trajectory Tracking of Delta Parallel Robot. *International Computer Science and Engineering Conference (ICSEC).*

RepRap. (s.d.). Fonte: RepRap: http://reprap.org/

Shareef, Z. (7 de Maio de 2015). Path Planning and Trajectory Optimization of Delta Parallel Robot. Karachi, Paquistão.

Silva, S. D. (2002). *Cnc Programaçao De Comandos Numericos Computador.* Érica Editora.

Stan, S.-D., Manic, M., Szep, C., & Balan, R. (2011). Performance analysis of 3 DOF Delta parallel. *HSL.*

Stepper motor. (s.d.). Fonte: RepRap: http://reprap.org/wiki/Stepper_motor

Types of 3D printers or 3D printing technologies overview. (s.d.). Fonte: 3D Printing from scratch: http://3dprintingfromscratch.com/common/types-of-3d-printers-or-3d-printing-technologies-overview/

Williams II, R. L. (2016). *The Delta Parallel Robot: Kinematics Solutions.* Fonte: www.ohio.edu/people/williar4/html/pdf/DeltaKin.pdf

Young, C.-P., & Lin, Y.-B. (2016). The Spherical Motion Based on the Inverse Kinematics for A Delta Robot. *The 2nd International Conference on Control, Automation and Robotics.*

APÊNDICE A – CÓDIGOS DO MATLAB

Código Principal – Transformação de Coordenadas

```
%% 3D Printer Senai-SL - Transformação de coordenadas Delta Rotacional
% Author: Felipe Henrique Neiva do Nascimento

%% Controle de Posição
% transforma a posição do bico da estrutura delta para os três valores
de
% angulação

%% Inicialização da Simulação
clear all
close all
clc

% Variáveis, dados em milimetros
global Sb % Lado do triângulo equilateral da Base
global Sp % Lado do triângulo equilateral da plataforma
global L  % Tamanho das pernas superiores
global l  % Altura das pernas do paralelograma
global h  % Largura das pernas do paralelograma
global wb % Distância planar do 0 até um dos lados da base, wb =
sqrt(3)*Sb/6
global ub % Distância planar do 0 até o vertix da base, ub =
sqrt(3)*Sb/3
global wp % Distância planar de P até a lateral da plataforma, wp =
sqrt(3)*Sp/6
global up % Distância planar de P até o vertix da plataforma, up =
sqrt(3)*Sp/3

global B1
global B2
global B3

global b1
global b2
global b3

global P1
global P2
global P3

global x
global y
global z

global Motorstep; % Passo em cada motor de passo, em graus
global Steps; % Numero total de passos numa volta de 360 graus
global microsteps; % Numero de micro passos por volta
global totalmicrosteps; % Total de micropassos por volta
```

```
L = 230;
l = 320;
h = 131;
wb = 100;
ub = 2*wb;
up = 50;
wp = up/2;
Sb = 6*wb/sqrt(3);
Sp = 3*up/sqrt(3);

Motorstep = 1.8;
Steps = 360/Motorstep;
microsteps = 16;
totalmicrosteps = Steps*microsteps;
%% Definição de x, y e z

x = 00;
y = 00;
z = 700;

% Matrix de valores X, Y e Z para visualização da cinemática inversa

MaxTime = 30;
Passo = 0.1;
t = (0:Passo:MaxTime); %Faixa de valores
R = 10; %raio do movimento da circunferencia
X = R*sin(2*pi*t/MaxTime); % X e Y em uma circunferencia de raio R
variando como uma senoide
Y = R*cos(2*pi*t/MaxTime);
Z = 100 + 10*sin(2*pi*t/MaxTime); % Z variando como uma senoide de
amplitude 10 centrada em 100

TetaM1 = [];
TetaM2 = [];
TetaM3 = [];
%% Cinemática Inversa

[teta1, teta2, teta3] = IKDeltaRot(x,y,z);

%% Cinemática inversa no tempo

for i = 1:1:((MaxTime/Passo)+1)
   [TetaM1(i), TetaM2(i), TetaM3(i)] = IKDeltaRot(X(i),Y(i),Z(i));
end

%% Workspace

% Encontrando a altura z onde a diferenca de passo de uma altura pra
outra
% é a menor possivel. escolhe-se um passo minimo, nesse caso de
resolucao
% 0.01, e a partir do ponto 0,0,0 vai-se incrementando esse passo e
vendo a
% diferenca de angulo total dos motores desse passo para o ultimo. A
% tendencia é essa diferenca ir diminuindo, pois os motores estao se
% aproximando do ponto otimo. Uma vez nesse ponto otimo, a diferenca
vai
```

```
% aumentar no proximo passo. Quando ele deixar de diminuir e aumentar,
é
% onde encontraremos o ponto otimo.

% Encontrando o Centro do workspace
x = 0;
y = 0;
z = 0;

PassoZ = 0.01;

Diff_Ant = 1000;
Angulo_Anterior = 0;
Angulo_Atual = teta1;
Diff_Angulo = Diff_Ant/2;
Centro_Workspace = 0;
while Centro_Workspace == 0;

    [teta1, teta2, teta3] = IKDeltaRot(x,y,z);
    Angulo_Atual = teta1;
    Diff_Angulo = abs(Angulo_Atual - Angulo_Anterior);

if Diff_Angulo > Diff_Ant
    Centro_Workspace = z
else
    Angulo_Anterior = Angulo_Atual;
    Diff_Ant = Diff_Angulo;
    z = z + PassoZ;
end

end

% Resolução na direção de um braço
% Para a resolução na direção de um braço e usando cinemática inversa,
um
% valor minimo de mudanca naquela direção é proposto a partir do
centro, e
% verifica-se se a mudança de ângulo resultante é menor ou maior do
que a
% mudança mínima de cada micropasso. Se for menor, temos que aumentar
o passo
% e refazer até termos o valor mais próximo da mudança minima. Se for
maior,
% temos que diminuir o passo. Essa vai ser a resolução máxima no
centro. A
% diferenca de um passo pra outro tem q ser maior que o micropasso.
% Pergunta: Tem que ser multiplo?

PassoY = 0.1;
PassoZ = 0.1;
AngMin = 360/totalmicrosteps;
faixaAng = AngMin/5;

[teta1, teta2, teta3] = IKDeltaRot(x,y,z);
resolucao_minima_centro = 0;
teta1a = teta1;
teta2a = teta2;
teta3a = teta3;
```

```
while resolucao_minima_centro == 0

    zcentro = z + PassoZ;
    [teta1, teta2, teta3] = IKDeltaRot(x,y,zcentro);
    diffteta1 = teta1 - teta1a;
    diffteta2 = teta2 - teta2a;
    diffteta3 = teta3 - teta3a;

    if abs(diffteta1) > (AngMin+faixaAng)
        PassoZ = PassoZ - 0.5*PassoZ;
    elseif abs(diffteta1) < AngMin
            PassoZ = PassoZ + 0.5*PassoZ;
    else
        resolucao_minima_centro = PassoZ;
    end
end


bottomZ = 400;
WorkspaceHeight = 300;
resolutionHeight = 0.2;
resolutionRadius = 1;
x = 0;
y = 0;
z = bottomZ;
WorkspacePoints = [];
WorkspaceRadius = 100;
valido = 0;
r = resolutionRadius;
[teta1a, teta2a, teta3a] = IKDeltaRot(x,y,z);
for z = (bottomZ-resolutionHeight):-resolutionHeight:(bottomZ -
WorkspaceHeight)
   [teta1, teta2, teta3] = IKDeltaRot(x,y,z);
    diffteta1 = teta1 - teta1a;
    diffteta2 = teta2 - teta2a;
    diffteta3 = teta3 - teta3a;
    if abs(diffteta1) >= AngMin
        if abs(diffteta2) >= AngMin
            if abs(diffteta3) >= AngMin
                valido = 1;
            else
                valido = 0;
            end
        else
            valido = 0;
        end
    else
        valido = 0;
    end
    somapassos = (abs(diffteta1) + abs(diffteta2) +
abs(diffteta3))/AngMin;
    WorkspacePoints = [WorkspacePoints;
x,y,z,diffteta1,diffteta2,diffteta3,valido,0,somapassos];
    while r < WorkspaceRadius
        for tetar = 0:1:359
            xa = (r-resolutionRadius)*cosd(tetar);
            ya = (r-resolutionRadius)*sind(tetar);
            [teta1a, teta2a, teta3a] = IKDeltaRot(xa,ya,z);
            x = r*cosd(tetar);
            y = r*sind(tetar);
            [teta1, teta2, teta3] = IKDeltaRot(x,y,z);
```

```
            diffteta1 = teta1 - teta1a;
            diffteta2 = teta2 - teta2a;
            diffteta3 = teta3 - teta3a;
            if abs(diffteta1) >= AngMin
                if abs(diffteta2) >= AngMin
                    if abs(diffteta3) >= AngMin
                        valido = 1;
                    else
                        valido = 0;
                    end
                else
                    valido = 0;
                end
            else
                valido = 0;
            end
            somapassos = (abs(diffteta1) + abs(diffteta2) +
abs(diffteta3))/AngMin;
            WorkspacePoints = [WorkspacePoints;
x,y,z,diffteta1,diffteta2,diffteta3,valido,tetar, somapassos];
        end
        r = r+0.1*r;
    end
    x = 0;
    y = 0;
    %r = 0;
    [teta1a, teta2a, teta3a] = IKDeltaRot(x,y,z);
end

%% Teste simples de movimento
% Com base na diferenca de teta de um ponto e de outro, podemos
calcular a
% quantidade de micropassos necessarios para sair de um ponto a outro.
%{
[teta1a, teta2a, teta3a] = IKDeltaRot(x,y,z);

msg = 'valores de x, y e z'
x
y
z

prompt = 'Insira novo valor de x';
x = input(prompt);
prompt = 'Insira novo valor de y';
y = input(prompt);

[teta1, teta2, teta3] = IKDeltaRot(x,y,z);

diffteta1 = teta1 - teta1a
diffteta2 = teta2 - teta2a
diffteta3 = teta3 - teta3a

msg = 'número de micropassos necessários'

motor1 = diffteta1/AngMin
motor1 = diffteta2/AngMin
motor1 = diffteta3/AngMin
%}
%% NOTA: Apenas válido se a resolução permanecesse constante. Como não
é o caso, precisa-se definir a função de resolução.
```

```
%% Gráfico

%Variáveis

Pb1 = P1 + [x,y,-z];
Pb2 = P2 + [x,y,-z];
Pb3 = P3 + [x,y,-z];

L1 = [0, -L*cosd(teta1),L*sind(teta1)];
L2 = [sqrt(3)*L*cosd(teta2)/2, L*cosd(teta2)/2,L*sind(teta2)];
L3 = [-sqrt(3)*L*cosd(teta3)/2, L*cosd(teta3)/2,L*sind(teta3)];

A1 = B1 + L1;
A2 = B2 + L2;
A3 = B3 + L3;

fator = (1/16);

A11 = A1 + [fator*(b1(1) - b3(1)),fator*(b1(2) - b3(2)), 0];
A12 = A1 - [fator*(b1(1) - b3(1)),fator*(b1(2) - b3(2)), 0];
Pb11 = Pb1 + [fator*(b1(1) - b3(1)),fator*(b1(2) - b3(2)), 0];
Pb12 = Pb1 - [fator*(b1(1) - b3(1)),fator*(b1(2) - b3(2)), 0];

A21 = A2 + [fator*(b2(1) - b1(1)),fator*(b2(2) - b1(2)), 0];
A22 = A2 - [fator*(b2(1) - b1(1)),fator*(b2(2) - b1(2)), 0];
Pb21 = Pb2 + [fator*(b2(1) - b1(1)),fator*(b2(2) - b1(2)), 0];
Pb22 = Pb2 - [fator*(b2(1) - b1(1)),fator*(b2(2) - b1(2)), 0];

A31 = A3 + [fator*(b2(1) - b3(1)),fator*(b2(2) - b3(2)), 0];
A32 = A3 - [fator*(b2(1) - b3(1)),fator*(b2(2) - b3(2)), 0];
Pb31 = Pb3 + [fator*(b2(1) - b3(1)),fator*(b2(2) - b3(2)), 0];
Pb32 = Pb3 - [fator*(b2(1) - b3(1)),fator*(b2(2) - b3(2)), 0];
%Vetores

B1A1x = [B1(1),A1(1)];
B1A1y = [B1(2),A1(2)];
B1A1z = [B1(3),A1(3)];

B2A2x = [B2(1),A2(1)];
B2A2y = [B2(2),A2(2)];
B2A2z = [B2(3),A2(3)];

B3A3x = [B3(1),A3(1)];
B3A3y = [B3(2),A3(2)];
B3A3z = [B3(3),A3(3)];

A1A11x = [A1(1), A11(1)];
A1A11y = [A1(2), A11(2)];
A1A11z = [A1(3), A11(3)];

A1A12x = [A1(1), A12(1)];
A1A12y = [A1(2), A12(2)];
A1A12z = [A1(3), A12(3)];

A2A21x = [A2(1), A21(1)];
A2A21y = [A2(2), A21(2)];
A2A21z = [A2(3), A21(3)];
```

```
A2A22x = [A2(1), A22(1)];
A2A22y = [A2(2), A22(2)];
A2A22z = [A2(3), A22(3)];

A3A31x = [A3(1), A31(1)];
A3A31y = [A3(2), A31(2)];
A3A31z = [A3(3), A31(3)];

A3A32x = [A3(1), A32(1)];
A3A32y = [A3(2), A32(2)];
A3A32z = [A3(3), A32(3)];

Pb1p11x = [Pb1(1), Pb11(1)];
Pb1p11y = [Pb1(2), Pb11(2)];
Pb1p11z = [Pb1(3), Pb11(3)];

Pb1p12x = [Pb1(1), Pb12(1)];
Pb1p12y = [Pb1(2), Pb12(2)];
Pb1p12z = [Pb1(3), Pb12(3)];

Pb2p21x = [Pb2(1), Pb21(1)];
Pb2p21y = [Pb2(2), Pb21(2)];
Pb2p21z = [Pb2(3), Pb21(3)];

Pb2p22x = [Pb2(1), Pb22(1)];
Pb2p22y = [Pb2(2), Pb22(2)];
Pb2p22z = [Pb2(3), Pb22(3)];

Pb3p31x = [Pb3(1), Pb31(1)];
Pb3p31y = [Pb3(2), Pb31(2)];
Pb3p31z = [Pb3(3), Pb31(3)];

Pb3p32x = [Pb3(1), Pb32(1)];
Pb3p32y = [Pb3(2), Pb32(2)];
Pb3p32z = [Pb3(3), Pb32(3)];

b12x = [b1(1),b2(1)];
b12y = [b1(2),b2(2)];
b12z = [b1(3),b2(3)];

b23x = [b2(1),b3(1)];
b23y = [b2(2),b3(2)];
b23z = [b2(3),b3(3)];

b31x = [b3(1),b1(1)];
b31y = [b3(2),b1(2)];
b31z = [b3(3),b1(3)];

P12x = [Pb1(1),Pb2(1)];
P12y = [Pb1(2),Pb2(2)];
P12z = [Pb1(3),Pb2(3)];

P23x = [Pb2(1),Pb3(1)];
P23y = [Pb2(2),Pb3(2)];
P23z = [Pb2(3),Pb3(3)];

P31x = [Pb3(1),Pb1(1)];
P31y = [Pb3(2),Pb1(2)];
P31z = [Pb3(3),Pb1(3)];
```

```
l1p11x = [A11(1),Pb11(1)];
l1p11y = [A11(2),Pb11(2)];
l1p11z = [A11(3),Pb11(3)];

l1p12x = [A12(1),Pb12(1)];
l1p12y = [A12(2),Pb12(2)];
l1p12z = [A12(3),Pb12(3)];

l2p21x = [A21(1),Pb21(1)];
l2p21y = [A21(2),Pb21(2)];
l2p21z = [A21(3),Pb21(3)];

l2p22x = [A22(1),Pb22(1)];
l2p22y = [A22(2),Pb22(2)];
l2p22z = [A22(3),Pb22(3)];

l3p31x = [A31(1),Pb31(1)];
l3p31y = [A31(2),Pb31(2)];
l3p31z = [A31(3),Pb31(3)];

l3p32x = [A32(1),Pb32(1)];
l3p32y = [A32(2),Pb32(2)];
l3p32z = [A32(3),Pb32(3)];

l2p2x = [A2(1),Pb2(1)];
l2p2y = [A2(2),Pb2(2)];
l2p2z = [A2(3),Pb2(3)];

l3p3x = [A3(1),Pb3(1)];
l3p3y = [A3(2),Pb3(2)];
l3p3z = [A3(3),Pb3(3)];

figure(1)
plot3(b12x,b12y,b12z, 'Color', [1,0,0])
hold
plot3(b23x,b23y,b23z, 'Color', [1,0,0])
plot3(b31x,b31y,b31z, 'Color', [1,0,0])

plot3(P12x,P12y,P12z, 'Color', [0,1,0])
plot3(P23x,P23y,P23z, 'Color', [0,1,0])
plot3(P31x,P31y,P31z, 'Color', [0,1,0])

plot3(B1A1x,B1A1y,B1A1z, 'Color', [0,0,1])
plot3(B2A2x,B2A2y,B2A2z, 'Color', [0,0,1])
plot3(B3A3x,B3A3y,B3A3z, 'Color', [0,0,1])

plot3(l1p11x,l1p11y,l1p11z, 'Color', [0,0,1])
plot3(l1p12x,l1p12y,l1p12z, 'Color', [0,0,1])
plot3(A1A11x,A1A11y,A1A11z, 'Color', [0,0,1])
plot3(A1A12x,A1A12y,A1A12z, 'Color', [0,0,1])
plot3(Pb1p11x,Pb1p11y,Pb1p11z, 'Color', [0,0,1])
plot3(Pb1p12x,Pb1p12y,Pb1p12z, 'Color', [0,0,1])

plot3(l2p21x,l2p21y,l2p21z, 'Color', [0,0,1])
plot3(l2p22x,l2p22y,l2p22z, 'Color', [0,0,1])
plot3(A2A21x,A2A21y,A2A21z, 'Color', [0,0,1])
plot3(A2A22x,A2A22y,A2A22z, 'Color', [0,0,1])
plot3(Pb2p21x,Pb2p21y,Pb2p21z, 'Color', [0,0,1])
```

```
plot3(Pb2p22x,Pb2p22y,Pb2p22z, 'Color', [0,0,1])

plot3(l3p31x,l3p31y,l3p31z, 'Color', [0,0,1])
plot3(l3p32x,l3p32y,l3p32z, 'Color', [0,0,1])
plot3(A3A31x,A3A31y,A3A31z, 'Color', [0,0,1])
plot3(A3A32x,A3A32y,A3A32z, 'Color', [0,0,1])
plot3(Pb3p31x,Pb3p31y,Pb3p31z, 'Color', [0,0,1])
plot3(Pb3p32x,Pb3p32y,Pb3p32z, 'Color', [0,0,1])

title('Visualização de uma estrutura Delta Rotacional')
xlabel('X(mm)')
ylabel('Y(mm)')
zlabel('Z(mm)')

figure(2)
grid on
plot(t,X,t,Y,t,Z)
title('Variação Cartesiana')
xlabel('Tempo')
ylabel('Posição(mm)')
legend('X','Y','Z')
figure(3)
grid on
plot(t,TetaM1,t,TetaM2,t,TetaM3)
title('Variação na Articulação')
xlabel('Tempo')
ylabel('Posição Angular(grau)')
legend('Teta1','Teta2','Teta3')
```

Função de Cinemática Inversa

```
%% Função de cinemática inversa
% Variáveis, dados em milimetros
function [teta1,teta2,teta3] = IKDeltaRot(x,y,z)
global Sb; % Lado do triângulo equilateral da Base
global Sp; % Lado do triângulo equilateral da plataforma
global L; % Tamanho das pernas superiores
global l; % Altura das pernas do paralelograma
global h; % Largura das pernas do paralelograma
global wb; % Distância planar do 0 até um dos lados da base, wb =
sqrt(3)*Sb/6
global ub; % Distância planar do 0 até o vertix da base, ub =
sqrt(3)*Sb/3
global wp; % Distância planar de P até a lateral da plataforma, wp =
sqrt(3)*Sp/6
global up; % Distância planar de P até o vertix da plataforma, up =
sqrt(3)*Sp/6

global B1
global B2
global B3

global b1
global b2
global b3
```

```
global P1
global P2
global P3


% Articulações da Base com relação a Base

B1 = [0,-wb,0];
B2 = [sqrt(3)*wb/2,wb/2,0];
B3 = [(-1)*sqrt(3)*wb/2,wb/2,0];

% Vertices do triângulo da Base com relação a Base

b1 = [Sb/2,-wb,0];
b2 = [0,ub,0];
b3 = [(-1)*Sb/2,-wb,0];

% Articulações da plataforma com relacao a plataforma

P1 = [0,-up,0];
P2 = [Sp/2,wp,0];
P3 = [(-1)*Sp/2,wp,0];

%% Solução de cinemática Inversa

a = wb - up;
b = (Sp/2) - (sqrt(3)/2)*wb;
c = wp - (wb/2);

% Motor 1

E1 = 2*L*(y+a);
F1 = 2*z*L;
G1 = (x^2) + (y^2) + (z^2) + (a^2) + (L^2) + 2*y*a - (l^2);

% Motor 2

E2 = -L*(sqrt(3)*(x + b) + y + c);
F2 = 2*z*L;
G2 = (x^2) + (y^2) + (z^2) + (b^2) + (c^2) + (L^2) + 2*((x*b) + (y*c))
- (l^2);

% Motor 3

E3 = L*(sqrt(3)*(x - b) - y - c);
F3 = 2*z*L;
G3 = (x^2) + (y^2) + (z^2) + (b^2) + (c^2) + (L^2) + 2*(((-1)*x*b) +
(y*c)) - (l^2);

% Substituição de meio ângulo da tangente

t11 = (-F1 + sqrt((E1^2) + (F1^2) - (G1^2)))/(G1 - E1);
t12 = (-F1 - sqrt((E1^2) + (F1^2) - (G1^2)))/(G1 - E1);

t21 = (-F2 + sqrt((E2^2) + (F2^2) - (G2^2)))/(G2 - E2);
t22 = (-F2 - sqrt((E2^2) + (F2^2) - (G2^2)))/(G2 - E2);

t31 = (-F3 + sqrt((E3^2) + (F3^2) - (G3^2)))/(G3 - E3);
```

```
t32 = (-F3 - sqrt((E3^2) + (F3^2) - (G3^2)))/(G3 - E3);

% Angulos

if imag(t11) ~= 0
    teta11 = angle(2*atan(t11));
else
    teta11 = 2*atan(t11);
end

if imag(t12) ~= 0
    teta12 = angle(2*atan(t12));
else
    teta12 = 2*atan(t12);
end

if (abs(teta11) < abs(teta12))
    teta1 = teta11*180/pi;
else
    teta1 = teta12*180/pi;
end

if imag(t21) ~= 0
    teta21 = angle(2*atan(t21));
else
    teta21 = 2*atan(t21);
end

if imag(t22) ~= 0
    teta22 = angle(2*atan(t22));
else
    teta22 = 2*atan(t22);
end


if (abs(teta21) < abs(teta22))
    teta2 = teta21*180/pi;
else
    teta2 = teta22*180/pi;
end

if imag(t31) ~= 0
    teta31 = angle(2*atan(t31));
else
    teta31 = 2*atan(t31);
end

if imag(t32) ~= 0
    teta32 = angle(2*atan(t32));
else
    teta32 = 2*atan(t32);
end


if (abs(teta31) < abs(teta32))
    teta3 = teta31*180/pi;
else
    teta3 = teta32*180/pi;
end
```

```
end
```

Função de Cinemática Direta

```
%% Foward Kinematics Delta Paralelo Rotacional.
function [x,y,z] = FKDeltaRot(teta1,teta2,teta3)

global L; % Tamanho das pernas superiores
global l; % Altura das pernas do paralelograma
global wb; % Distancia planar do centro da base ao atuador
global up; % Distancia planar do centro da plataforma ao bra?o

Ra = wb;
Rb = up;
R = Ra - Rb;

alpha = [120*pi/180; 240*pi/180; 0];

eqn_1 = @(x) ((x(1) - (cos(alpha(1))*((L*cos(teta1*pi/180))+R)))^2) +
((x(2) - ((-1)*(R+L*cos(teta1*pi/180))*sin(alpha(1))))^2) + ((x(3) -
((-1)*L*sin(teta1*pi/180)))^2) - l^2;
eqn_2 = @(x) ((x(1) - (cos(alpha(2))*((L*cos(teta2*pi/180))+R)))^2) +
((x(2) - ((-1)*(R+L*cos(teta2*pi/180))*sin(alpha(2))))^2) + ((x(3) -
((-1)*L*sin(teta2*pi/180)))^2) - l^2;
eqn_3 = @(x) ((x(1) - (cos(alpha(3))*((L*cos(teta3*pi/180))+R)))^2) +
((x(2) - ((-1)*(R+L*cos(teta3*pi/180))*sin(alpha(3))))^2) + ((x(3) -
((-1)*L*sin(teta3*pi/180)))^2) - l^2;

eqn = @(x) [eqn_1(x); eqn_2(x); eqn_3(x)];

s0 = [0; 0; 0];

options = optimset('TolFun', 1e-8);
X = fsolve(eqn ,s0, options)

x = X(1);
y = X(2);
z = X(3);

end
```

APÊNDICE B - CÓDIGO NO ARDUINO

```
#include <math.h>
#include <complex.h>

//Global Variables
const int proporcao = 3;
const float Sb = 1.5*567/proporcao; // Lado do triângulo equilateral da Base
const float Sp = 76; // Lado do triângulo equilateral da plataforma
const float L = 524/proporcao;  // Tamanho das pernas superiores
const float l = 1244/proporcao;   // Altura das pernas do paralelograma
const float h = 131/proporcao; // Largura das pernas do paralelograma
const float wb = 1.5*164/proporcao; // Distância planar do 0 até um dos lados da base, wb = sqrt(3)*Sb/6
const float ub = 1.5*327/proporcao; // Distância planar do 0 até o vertix da base, ub = sqrt(3)*Sb/3
const float wp = 22; // Distância planar de P até a lateral da plataforma, wp = sqrt(3)*Sp/6
const float up = 44; // Distância planar de P até o vertix da plataforma, up = sqrt(3)*Sp/6
const float pi = 3.141592;

double x;
double y;
double z;
double diffteta1;
double diffteta2;
double diffteta3;
double Step = 1.8;
double microStep = 16;
double microStepAngle = Step/microStep;
double auxI;

double a = wb - up;
double b = ((Sp/2) - ((sqrt(3)/2)*wb));
double c = wp - (wb/2);
```

```
const int stepPin = 3;
const int dirPin = 4;
const int MS1 = 5;
const int MS2 = 6;
const int MS3 = 7;

struct ANG{
  double teta1;
  double teta2;
  double teta3;
  };

ANG posicao;
ANG posicaoAnterior;

ANG IKDeltaRot(double,double,double);

void setup() {
  // put your setup code here, to run once:
  Serial.begin(9600);

  pinMode(stepPin,OUTPUT);
  pinMode(dirPin,OUTPUT);
  pinMode(MS1,OUTPUT);
  pinMode(MS2,OUTPUT);
  pinMode(MS3,OUTPUT);

  digitalWrite(MS1,HIGH);
  digitalWrite(MS2,HIGH);
  digitalWrite(MS3,HIGH);

  x = 0;
  y = 0;
  z = 0;

  posicaoAnterior = IKDeltaRot(x,y,z);
}

void loop() {
  // put your main code here, to run repeatedly:
```

```
      for (z = 0; z < 400; z += 10){
        posicao = IKDeltaRot(x,y,z);
        diffteta1 = posicao.teta1 - posicaoAnterior.teta1;
        Serial.print(diffteta1);
        delay(5000);
        if (diffteta1 <= 0)
            digitalWrite(dirPin,LOW);
        else digitalWrite(dirPin,HIGH);
        for(auxI = 0; auxI < fabs(diffteta1); auxI+= microStepAngle){
          digitalWrite(stepPin,HIGH);
          delayMicroseconds(500);
          digitalWrite(stepPin,LOW);
          delayMicroseconds(500);
        }
        posicaoAnterior = posicao;
        }
        delay(3000);
      }

      ANG IKDeltaRot(double x, double y, double z){
        double E1, F1, G1, E2, F2, G2, E3, F3, G3;
        double teta11, teta12, teta21, teta22, teta31, teta32;
        Complex t11, t12, t21, t22, t31, t32;
        Complex aux, aux2;
        ANG Angulos;

          // Motor 1

          E1 = 2*L*(y+a);
          F1 = 2*z*L;
          G1 = (sq(x)) + (sq(y)) + (sq(z)) + (sq(a))  + (sq(L)) + 2*y*a
- (sq(l));

          // Motor 2

          E2 = -L*(sqrt(3)*(x + b) + y + c);
          F2 = F1;
          G2 = ((sq(x)) + (sq(y)) + (sq(z)) + (sq(b)) + (sq(c)) + (sq(L))
+ 2*((x*b) + (y*c)) - (sq(l)));

          // Motor 3
```

```
            E3 = L*(sqrt(3)*(x - b) - y - c);
            F3 = F1;
            G3 = (sq(x)) + (sq(y)) + (sq(z)) + (sq(b)) + (sq(c)) + (sq(L))
+ 2*(((-1)*x*b) + (y*c)) - (sq(l));

            // Substituição de meio ângulo da tangente

            aux.set((sq(E1) + sq(F1) - sq(G1)),0);
            t11 = (aux.c_sqrt() - F1)/(G1 - E1);
            t12 = (aux.c_sqrt() + F1)/(-G1 + E1);

            aux.set((sq(E2) + sq(F2) - sq(G2)),0);
            t21 = (aux.c_sqrt() - F2)/(G2 - E2);
            t22 = (aux.c_sqrt() + F2)/(-G2 + E2);

            aux.set((sq(E3) + sq(F3) - sq(G3)),0);
            t31 = (aux.c_sqrt() - F3)/(G3 - E3);
            t32 = (aux.c_sqrt() + F3)/(-G3 + E3);

            // Angulos

            t11 = t11.c_atan();
            t12 = t12.c_atan();
            t21 = t21.c_atan();
            t22 = t22.c_atan();
            t31 = t31.c_atan();
            t32 = t32.c_atan();

            teta11 = atan2(2*t11.imag(),2*t11.real());
            teta12 = atan2(2*t12.imag(),2*t12.real());
            teta21 = atan2(2*t21.imag(),2*t21.real());
            teta22 = atan2(2*t22.imag(),2*t22.real());
            teta31 = atan2(2*t31.imag(),2*t31.real());
            teta32 = atan2(2*t32.imag(),2*t32.real());

            Angulos.teta1 = fmax(fabs(teta11),fabs(teta12))*180/pi;
            Angulos.teta2 = fmax(fabs(teta21),fabs(teta22))*180/pi;
            Angulos.teta3 = fmax(fabs(teta31),fabs(teta32))*180/pi;

            return Angulos;
```

```
}
```