Este trabalho consiste na construção de um robô paralelo guiado por cabos para análise do espaço de trabalho e comparação dos cálculos da cinemática direta, inversa e estática usando a teoria de helicóides. A localização da plataforma é calculada pela aplicação de método iterativo para demonstrar que a partir das forças dadas em cada cabo é possível obter a localização (posição e orientação) da plataforma móvel. A estrutura do robô é constituída por uma plataforma móvel e sensores para medir as forças nos cabos.

Orientador: Dr. Eng. Aníbal Alexandre Campos Bonilla

Joinville, 2017

ANO 2017

MAICOL PETERSON GANDOLPHI DE ALMEIDA | PROJETO, CONSTRUÇÃO E ANÁLISE DA LOCALIZAÇÃO E DAS FORÇAS EM UM ROBÔ PARALELO GUIADO POR CABOS

UDESC

UNIVERSIDADE DO ESTADO DE SANTA CATARINA – UDESC
CENTRO DE CIÊNCIAS TECNOLÓGICAS – CCT
CURSO DE MESTRADO EM ENGENHARIA ELÉTRICA

DISSERTAÇÃO DE MESTRADO

PROJETO, CONSTRUÇÃO E ANÁLISE DA LOCALIZAÇÃO E DAS FORÇAS EM UM ROBÔ PARALELO GUIADO POR CABOS

MAICOL PETERSON GANDOLPHI DE ALMEIDA

JOINVILLE, 2017

MAICOL PETERSON GANDOLPHI DE ALMEIDA

PROJETO, CONSTRUÇÃO E ANÁLISE DA LOCALIZAÇÃO E DAS FORÇAS EM UM ROBÔ PARALELO GUIADO POR CABOS

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica - PPGEEL, Centro de Ciências Tecnológicas - CCT, Universidade do Estado de Santa Catarina - UDESC, como requisito parcial para a obtenção do título de Mestre em Engenharia Elétrica.

Orientador: Dr. Aníbal Alexandre Campos Bonilla

JOINVILLE - SC

2017

Ficha catalográfica elaborada pelo(a) autor(a), com
auxílio do programa de geração automática da
Biblioteca Setorial do CCT/UDESC

Peterson Gandolphi de Almeida, Maicol
PROJETO, CONSTRUÇÃO E ANÁLISE DA LOCALIZAÇÃO E DAS FORÇAS EM UM ROBÔ PARALELO GUIADO POR CABOS / Maicol Peterson Gandolphi de Almeida. - Joinville , 2017.
126 p.

Orientador: Aníbal Alexandre Campos Bonilla
Dissertação (Mestrado) - Universidade do Estado de Santa Catarina, Centro de Ciências Tecnológicas, Programa de Pós-Graduação Profissional em Engenharia Elétrica, Joinville, 2017.

1. robô paralelo guiado por cabos. 2. teoria de helicóides. I. Alexandre Campos Bonilla, Aníbal. II. Universidade do Estado de Santa Catarina. Programa de Pós-Graduação. III. Título.

Projeto, Construção e Análise da Localização e Forças de um Robô Paralelo Guiado por Cabos

por

Maicol Peterson Gandolphi de Almeida

Esta dissertação foi julgada adequada para obtenção do título de

MESTRE PROFISSIONAL EM ENGENHARIA ELÉTRICA

Área de concentração em "Automação de Sistemas"
e aprovada em sua forma final pelo

CURSO DE MESTRADO PROFISSIONAL EM ENGENHARIA ELÉTRICA
CENTRO DE CIÊNCIAS TECNOLÓGICAS DA
UNIVERSIDADE DO ESTADO DE SANTA CATARINA.

Banca Examinadora:

Prof. Dr. Aníbal Alexandre Campos Bonilla
CCT/UDESC (Orientador/Presidente)

Prof. Dr. Antônio Otaviano Dourado
UFSC

Prof. Dr. Douglas Wildgrube Bertol
CCT/UDESC

Joinville,SC, 26 de setembro de 2017.

Dedico este trabalho aos meus filhos Bernardo Bordin de Almeida e Vinícius Chepli de Almeida.

.

AGRADECIMENTOS

Agradeço a Universidade do Estado de Santa Catarina – UDESC pelo apoio institucional e ao professor Dr. Aníbal Alexandre Campos Bonilla pela paciência e dedicação nas correções e orientações neste período de aprendizado.

Agradeço a empresa Aeroville de Joinville pelo apoio financeiro na construção da estrutura.

Aos meus pais Naor de Almeida e Arlete Gandolphi por manter minha atenção nos estudos durante toda a infância, adolescência e juventude.

“Uma vez que você tenha experimentado voar, você andará pela terra com seus olhos voltados para o céu, pois lá você esteve e para lá você desejará voltar.”

Leonardo da Vinci.

RESUMO

Este trabalho consiste na construção de um robô paralelo guiado por cabos para análise do espaço de trabalho e comparação dos cálculos da cinemática direta, inversa e estática usando a teoria de helicóides. A localização da plataforma é calculada pela aplicação de método iterativo para demonstrar que a partir das forças dadas em cada cabo é possível obter a localização (posição e orientação) da plataforma móvel. A estrutura do robô é constituída por uma plataforma móvel e sensores para medir as forças nos cabos.

Palavras chaves: robô paralelo guiado por cabos, teoria de helicóides.

ABSTRACT

This work consists in the construction of a cable-driven parallel robot for analysis of the work space and comparison of the calculations of the direct, inverse and static kinematics using the theory of helicoids. The location of the platform is calculated by applying an iterative method to demonstrate that from the forces given in each cable it is possible to obtain the location (position and orientation) of the mobile platform. The structure of the robot consists of a mobile platform and sensors to measure the forces in the cables.

Key words: cable-driven parallel robot, theory of helicoids.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SÍMBOLOS

Θ	Ângulo Theta em torno do eixo X do sistema móvel {P}
φ	Ângulo Phi em torno do eixo Y do sistema móvel {P}
ψ	Ângulo Psi em torno do eixo Z do sistema móvel {P}
$^{A}_{B}R_{X}$	Matriz de rotação em X que leva {B} para o sistema {A}
$^{A}_{B}R_{Y}$	Matriz de rotação em Y que leva {B} para o sistema {A}
$^{A}_{B}R_{Z}$	Matriz de rotação em Z que leva {B} para o sistema {A}
$^{A}_{B}R_{XYZ}$	Matriz de rotação por ângulos de Euler que leva {B} para {A}
$^{A}\vec{P}$	Vetor do ponto P visto na referência {A}
$	Heligiro
$'	Heliforça
$\overrightarrow{F}$	Vetor da força
$\overrightarrow{M}$	Vetor do momento
$\hat{\$}$	Heligiro unitário normalizado
$\hat{S}$	Direção unitária normalizada da helicóide
$\vec{S}_0$	Vetor que parte da origem fixa {O} até o eixo da helicóide
J	Jacobiano
$\dot{q}$	Matriz contendo as variáveis do espaço das juntas
W	Peso
$^{o}\vec{A}_1$	Vetor do ponto (A) visto na origem fixa {O}
$\|\overrightarrow{B_1A_1}\|$	Norma ou comprimento do ponto B_1 até A_1
π	Valor de Pi na circunferência
c(Θ)	Cosseno de Theta
s(Θ)	Seno de Theta
B_{1x}	Distância em X do ponto B_1 até a origem móvel {P}
L_i	Comprimento real do cabo do ponto A_i até a fixação na mola.

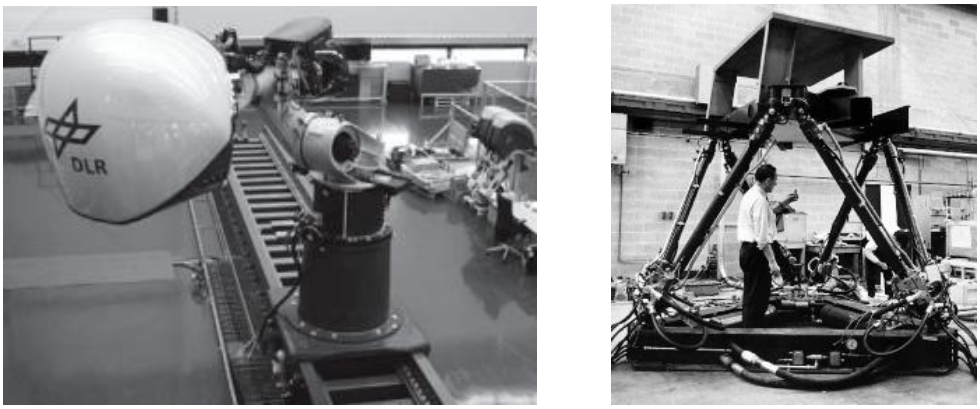
SUMÁRIO

1 INTRODUÇÃO

Este trabalho apresenta os cálculos teóricos e os resultados práticos da cinemática de posição e da estática de uma plataforma guiada por cabos montada no Centro de Ciências Tecnológicas (CCT) da Universidade do Estado de Santa Catarina (UDESC). O desenvolvimento consiste na construção das equações da cinemática direta, inversa e na estática do protótipo proposto, bem como da simulação de posições, orientações e forças em ambiente computacional e real. A teoria de helicóides (BALL 1900, HUNT 1978, CAMPOS 2004, SIMAS 2002, CARBONI 2012, POTT 2014) também é apresentada e utilizada como ferramenta principal no desenvolvimento das equações do robô guiado por cabos. Um estudo sobre robôs atuados por cabos é apresentado de forma a fundamentar a dissertação.

As plataformas para simulação de movimentos são construídas através de mecanismos seriais ou paralelos. Os simuladores de vôo são construídos usando estes tipos de plataformas para realizar os movimentos de uma cabine. Na comparação entre os simuladores com mecanismos seriais e paralelos, os paralelos possuem vantagens como baixo erro de posicionamento devido aos motores não estarem conectados de forma serial e movimentação de cargas maiores por manter todos os motores fixos na base (MERLET, 1989). A Figura 1 amostra exemplos de simuladores, o primeiro (SHARMA, 2013) montado num manipulador serial e o segundo (BONEV, 2003) sobre o primeiro manipulador paralelo patenteado para Klaus Cappel em 1964 nos Estados Unidos.

Figura 1 – Simuladores serial e paralelo.



Fonte: Sharma, 2013 e Bonev, 2003.

1.1 MOTIVAÇÃO

Com o surgimento da robótica, os simuladores ficaram mais sofisticados, realizando movimentos através de uma plataforma e proporcionando uma melhor imersão do usuário no simulador (BALADEZ, 2009).

A evolução da tecnologia trouxe novas maneiras de construir simuladores, além daqueles formados por braços mecânicos que movimentam uma plataforma e suportam todo o peso da cabine, também pode-se realizar o mesmo movimento com manipuladores paralelos rígidos ou guiados por cabos que reduzem drasticamente os esforços suportados pela estrutura do robô (TANG, 2014).

O projeto apresentado a seguir utiliza um robô atuado por quatro cabos que movimenta uma plataforma e informa através de sensores os valores das forças em cada cabo e a posição da plataforma dentro do espaço de trabalho.

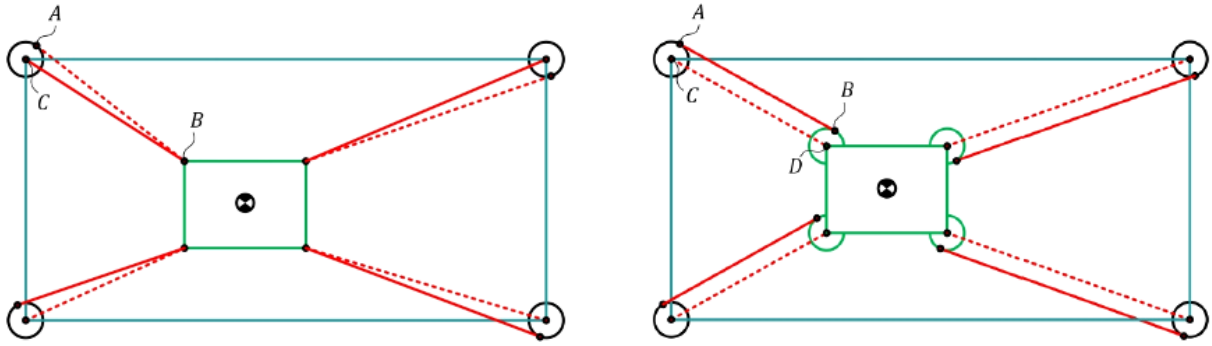
1.2 APRESENTAÇÃO DO PROBLEMA

Inicialmente é necessário realizar um estudo sobre robôs atuados por cabos e resolver as equações da cinemática do mecanismo de quatro cabos proposto para poder aplicá-las no protótipo e realizar uma análise entre teoria e prática. Para isso é usado teorias que promovam esta comparação, como a teoria de helicóides para o desenvolvimentos das equações lineares e não lineares do robô.

Simas et al (2002) demonstra que fazer o mapeamento analítico de um robô que possui mais juntas do que o exigido para seu movimento é impossível devido ao espaço das juntas possuir muitas incógnitas e por isso alguns trabalhos como os de Simas et al (2002), Valdiero et al (2001) e Gallardo et al (2003) tratam do uso da teoria das helicóides e aplicação da pseudo-inversa da matriz jacobiana como uma solução para este tipo de problema.

Gonzales-Rodrígues (2015) apresenta um problema na medição dos comprimentos dos cabos de um robô paralelo planar guiado por cabos. Conforme a montagem dos cabos é feita em torno das polias pode ocorrer uma diferença entre os comprimentos dos cabos calculados na teoria e os comprimentos medidos do robô real, a Figura 2 apresenta o problema e uma alternativa de rearranjo dada por Gonzales-Rodrígues.

Figura 2 – Erro de medição no comprimento dos cabos e uma solução proposta.

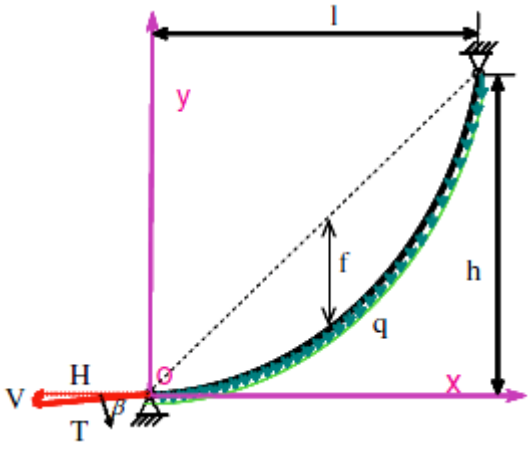


Fonte: GONZALES-RODRÍGUEZ, 2015.

Na Figura 2 a montagem da esquerda apresenta uma defasagem entre a fixação dos cabos na plataforma móvel e o ponto de tangência dos cabos nas roldanas, gerando uma imprecisão no posicionamento da plataforma. Na imagem à direita foi realizado um rearranjo inserindo-se roldanas nos pontos de fixação da plataforma para manter o alinhamento entre as roldanas da estrutura e da plataforma móvel, garantindo uma precisão melhor de posicionamento conforme apresentado no trabalho de Gonzales-Rodrígues (2015).

Outro problema apresentado em vários trabalhos aceitos na Segunda Conferência Internacional de Robôs Paralelos Guiados por Cabos (POTT, 2014) é o fato do cabo apresentar uma curva característica do seu peso próprio a medida que seu comprimento aumenta, chamada de curva catenária, veja na Figura 3 um exemplo.

Figura 3 – Curva catenária em robôs guiados por cabos



Fonte: POTT, 2014.

Além dos problemas apresentados é importante destacar o problema do espaço de trabalho restrito que um robô paralelo guiado por cabos possui devido a necessidade de tensionamento constante dos cabos e restrição de tensões próximas da ruptura dos mesmos (POTT, 2014).

1.3 OBJETIVOS

O objetivo principal deste trabalho é construir um protótipo de robô guiado por cabos, desenvolver as equações da cinemática e estática do robô através das dimensões do modelo tridimensional (3D), medir o espaço de trabalho do robô e apresentar testes e comparações dos resultados encontrados na teoria.

A partir das equações da cinemática do robô é demonstrado na prática que: dadas as forças atuantes em cada cabo é possível encontrar a localização da plataforma.

1.3.1 Objetivos Específicos

Para atingir o objetivo proposto, os seguintes objetivos específicos são definidos:

- Propor um modelo 3D de robô paralelo guiado por cabos, cuja plataforma móvel realize movimentos lineares sobre os eixos X, Y e Z e movimentos angulares de rolagem (*roll*) ou arfagem (*pitch*) em torno dos eixos X e Y. Os movimentos serão independentes em quatro graus de liberdade pelo fato de serem utilizados quatro motores, ou seja, quatro variáveis de entrada para a movimentação da plataforma.
- Realizar a medição do modelo em 3D para montar as equações seguindo a teoria das helicóides.
- Construir um protótipo real e comparar o estudo cinemático e estático com os movimentos reais do robô (posição, orientação e tensões nos cabos).
- Construir um sistema para sensoriamento de forças nos cabos.

- Utilizar um sistema com giroscópio digital e microcontrolador para medir a orientação da plataforma.

2 REVISÃO DE LITERATURA

Desde a época de Leonardo Da Vinci já se pensava em construir, através de peças mecânicas, algum tipo de estrutura na forma humana para atrair o público em festas oferecidas pelo rei. Leonardo Da Vinci imaginou construir um robô utilizando vigas de madeira, polias e cabos para simular os músculos e movimentar a estrutura humanóide (MORAN, 2006).

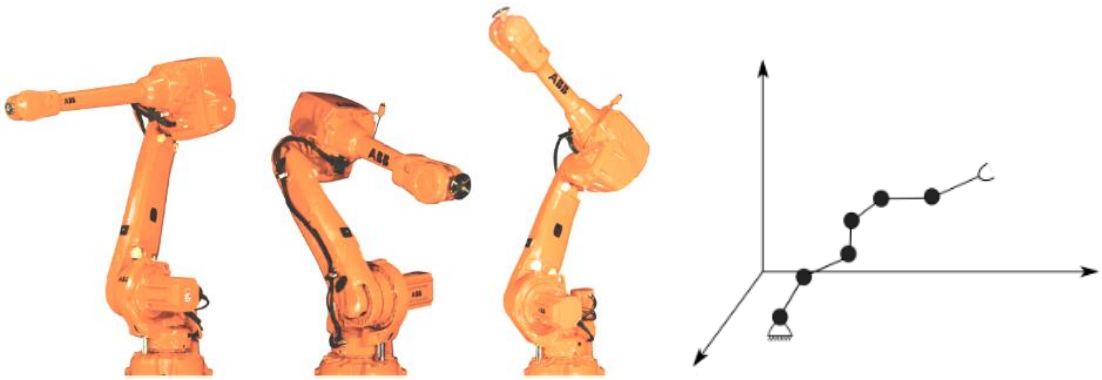
Necessariamente, a construção de uma máquina capaz de realizar movimentos programados exige o estudo prévio de todos os seus componentes e como estes se relacionam. Muitos trabalhos foram desenvolvidos nessa área, desde o desenvolvimento de métodos para calcular as equações que regem o movimento das partes de um robô (BALL, 1900) até a programação de sistemas eletrônicos de baixo custo (AUGUSTO, 2017).

Uma das referências encontradas na literatura é do autor Tsai (1999) que apresenta as classes de manipuladores. As principais características são descritas a seguir.

2.1 MANIPULADORES SERIAIS

Quando a estrutura topológica de um robô apresenta uma cadeia cinemática onde cada elo é conectado aos próximos elos por apenas um único caminho então chamamos este robô de serial. Esta cadeia cinemática é chamada de aberta (TSAI, 1999). A Figura 4 é um exemplo de robô serial construído pela empresa ABB.

Figura 4 – Robô serial ABB IRB 4600 e desenho esquemático de cadeia aberta.



Fonte: Mach, 2017 e Muraro, 2015.

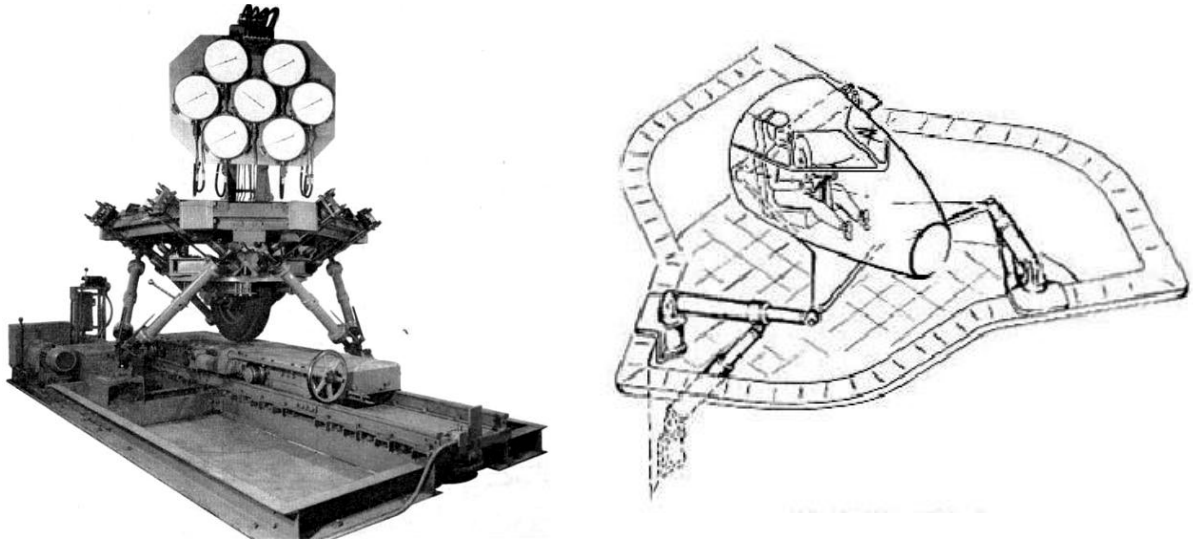
A posição de cada elo de um robô serial pode ser calculada através das equações formadas pelos cossenos diretores e transformada homogênea (CRAIG, 2005) que serão explicados ao decorrer deste trabalho.

2.2 MANIPULADORES PARALELOS RÍGIDOS

Um robô paralelo possui uma cadeia cinemática onde cada elo conecta-se com pelo menos dois caminhos, sendo chamada de cadeia cinemática fechada (TSAI 1999).

As estruturas cinemáticas paralelas surgiram associadas aos simuladores de voo por volta da década de 60 com o trabalho de Stewart na adaptação de um simulador de voo numa plataforma com 6 graus de liberdade chamada plataforma de Gough. Essa adaptação ficou conhecida como plataforma de Stewart ou plataforma de Gough-Stewart (STEWART, 1965). A Figura 5 apresenta o projeto original idealizado por Gough e o desenho esquemático do projeto de Doug Stewart.

Figura 5 – Plataforma de Gough (esquerda) e desenho esquemático de Stewart

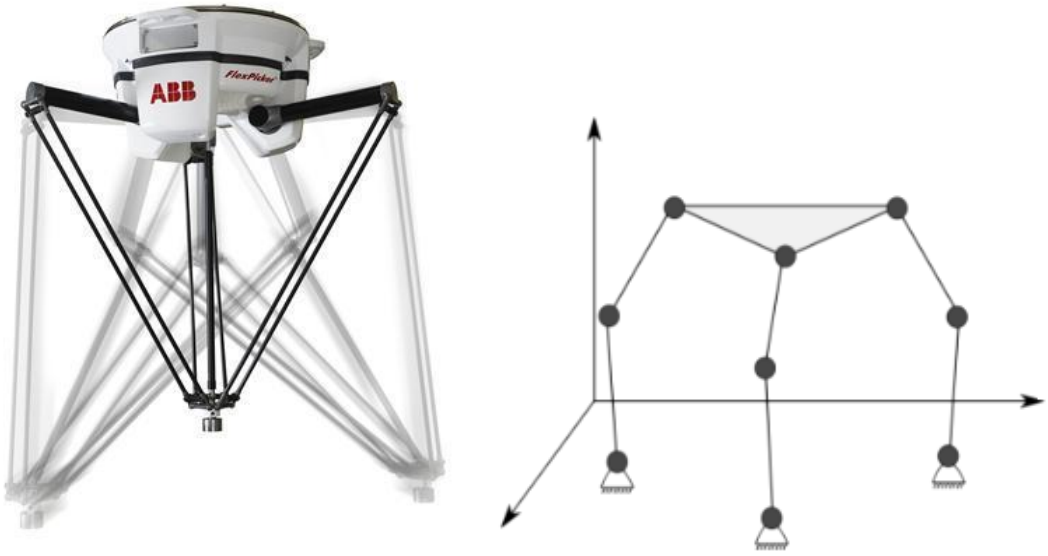


Fonte: Stewart,1965.

Na Figura 5 nota-se que o manipulador paralelo, também chamado de robô paralelo, possui uma base fixa e uma plataforma conectada por meio de atuadores que sofrem tração ou compressão.

A partir do trabalho de Stewart o interesse nessa arquitetura paralela vem crescendo, tanto na indústria quanto na medicina, setores de entretenimento, orientação de antenas, micromáquinas entre outras. Algumas fabricantes de robôs vem apostando na arquitetura paralela para aplicações de alta velocidade. A Figura 6 mostra o robô paralelo FlexPicker, um robô usado para tarefas de pegar e soltar objetos rapidamente, operação conhecida como *pick-and-place* (TARTARI, 2006).

Figura 6 - Robô paralelo FlexPicker e representação de sua cadeia cinemática.



Fonte: Tartari, 2006 e Muraro, 2015.

.

Alguns cálculos para determinar as posições dos elos e da plataforma de um manipulador paralelo são mais complexos devido ao espaço de trabalho ser mais restrito e por terem mais articulações passivas em relação a um robô serial (ANGELES et al, 1993 e MOURAIN, 1993).

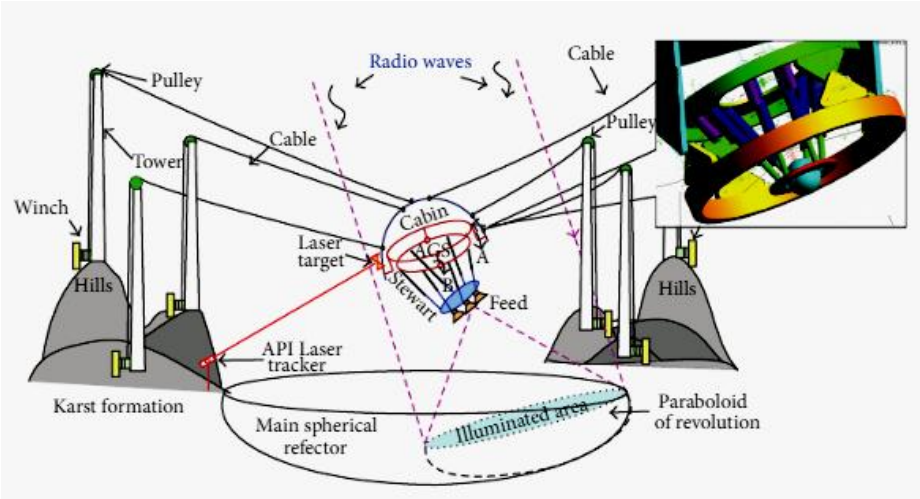
2.3 MANIPULADORES PARALELOS GUIADOS POR CABOS.

Os manipuladores paralelos guiados por cabos podem ser utilizados na área de reabilitação humana (PRIOR et al, 1990), visto na Figura 7, simuladores de voo (DUAN et al, 2014) como o apresentado na Figura 8, controle de posicionamento (MIERMEISTER et al, 2016) visto na Figura 9 entre outros (TANG, 2014).

Figura 7 - Manipulador paralelo guiado por cabos para a reabilitação humana

Fonte: Prior et al, 1990

Figura 8 – Simulador de voo guiado por cabos



Fonte: Duan, 2014.

Figura 9 – Manipulador guiado por cabos para o controle de posicionamento

Fonte: Miermeister et al, 2016.

Atualmente os manipuladores guiados por cabos também estão sendo usados em jogos de futebol (figura 10) para controlar o posicionamento das câmeras de televisão.

Figura 10 – Manipulador guiado por cabos Spidercam

Fonte: SPIDERCAM, disponível em: <www.spidercam.tv> Acesso em: 17 jul. 2017.

2.4 CINEMÁTICA DE MANIPULADORES

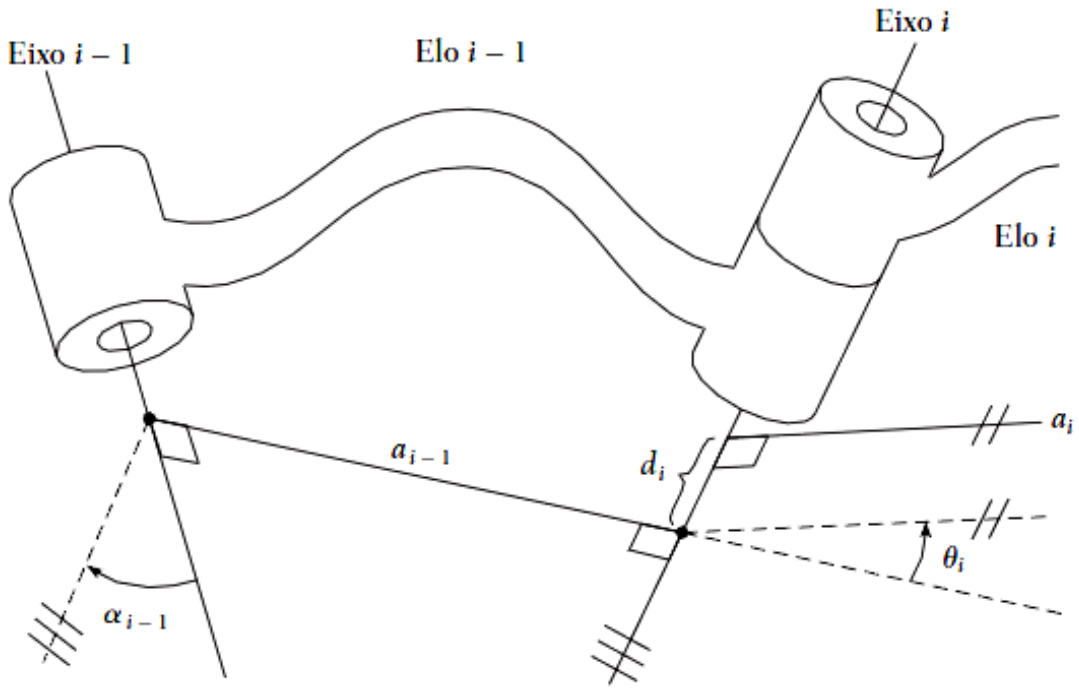
No estudo da cinemática dos robôs, tanto direta como inversa (GROOVER 1986) é preciso conhecer a teoria de geometria e trigonometria, pois a cinemática é responsável pelo equacionamento dos movimentos de cada parte de um robô (CRAIG, 2005).

Na sequência são apresentados alguns conceitos fundamentais da robótica, como a definição de manipulador, elo, junta e tipos de juntas.

2.4.1 Definição de manipulador, elo e junta

Segundo Craig (2005) um manipulador é composto por partes móveis unidas através de juntas, estas partes móveis são chamadas de elos. Um elo mantém uma relação entre os eixos de suas juntas. A relação que um elo mantém entre os seus eixos é constante e formada pela distância e angulação entre seus eixos, conforme visto na Figura 11.

Figura 11 – Representação de um elo



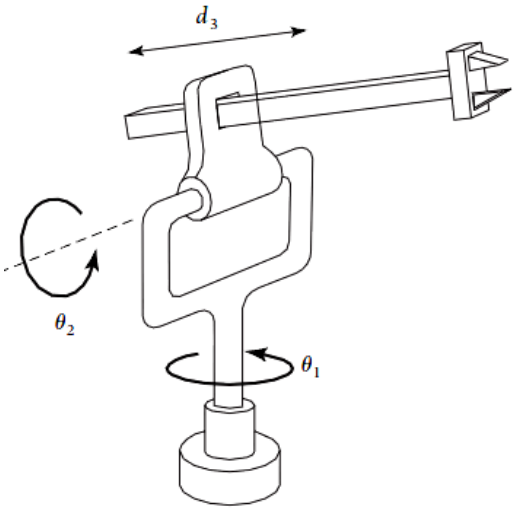
Fonte: CRAIG, 2005. A distância a_{i-1} e a angulação α_{i-1} formam a relação fixa do elo com seus dois eixos. Já a distância d_i e a angulação θ_i representam a conexão entre o elo i-1 e o elo i.

No exemplo acima os dois elos estão conectados através de uma junta rotativa, mas existem outros tipos de juntas, como a prismática e a helicoidal.

2.4.2 Juntas rotativa, prismática e parafuso

Na robótica serial a maioria dos elos é conectada a juntas que apresentam apenas um grau de liberdade, neste caso as juntas rotativas e prismáticas. Nas juntas rotativas um ângulo define a posição do elo seguinte e nas juntas prismáticas a posição do elo seguinte é definida através de uma distância. A Figura 12 representa um robô com duas juntas rotativas (abrevia-se como RR) e uma junta prismática (abrevia-se como P), também chamado de RRP (TSAI, 1999). A garra do robô é também chamada de efetuador.

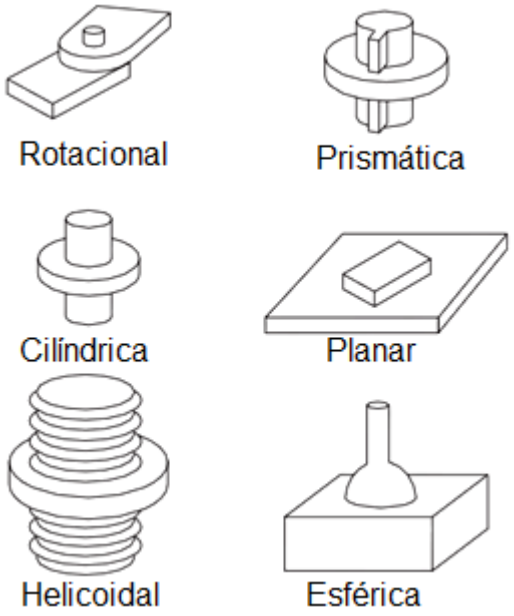
Figura 12 – Robô RRP



Fonte: CRAIG, 2005. Os ângulos θ_1 e θ_2 representam as juntas rotativas e a distância d_3 a prismática.

Nas juntas do tipo helicoidal estão presentes os movimentos de rotação e translação sobre um eixo e neste caso uma junta do tipo helicoidal pode ser decomposta em duas juntas: rotativa e prismática. Existem outros tipos de juntas, como as planares, esféricas e universais que também podem ser decompostas num conjunto de juntas (CRAIG, 2005). A Figura 13 mostra alguns tipos de juntas.

Figura 13 – Exemplo de juntas

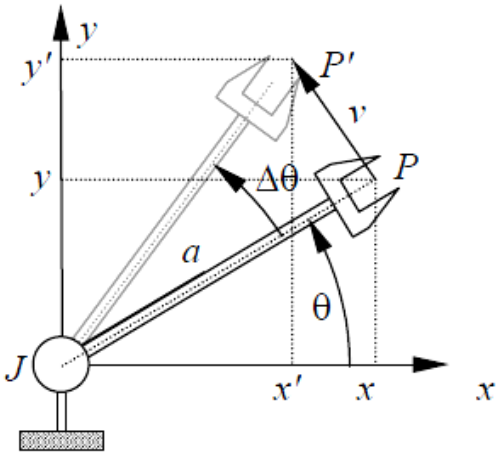


Fonte: CRAIG, 2005.

2.4.3 Cinemática Direta

A cinemática direta equaciona a posição do efetuador final em relação aos movimentos de cada junta do robô, podendo ser na maioria dos casos juntas rotativas que giram os elos em um ângulo específico ou prismáticas que movimentam os elos ao longo de um eixo escolhido (CRAIG, 2005). Como exemplo a Figura 14 contém uma junta rotativa e as equações para a localização do efetuador em relação aos eixos x e y.

Figura 14 – Junta rotativa e equações para cinemática direta.



Fonte: Craig, 2005. A posição P do efetuador pode ser determinada como x=a·cos(ɵ) e y= a·sen(ɵ).

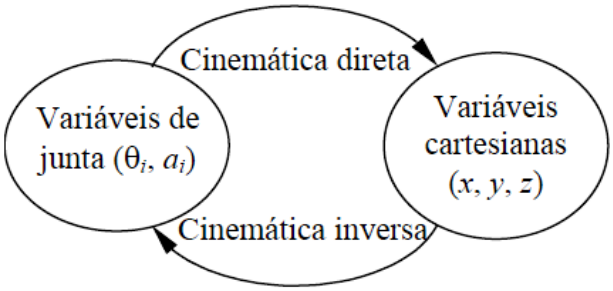
Na robótica quanto maior for o número de juntas rotativas, mais complexas são as equações devido ao movimento da junta posterior ser dependente do movimento da junta anterior. A literatura apresenta métodos para equacionar os movimentos de um robô e facilitar o trabalho para determinar a posição final do efetuador, entre estes métodos destaca-se o método de Denavit-Hartenberg (TSAI, 1999).

O objetivo da cinemática direta é encontrar a posição do efetuador (a garra do robô na Figura 14) través do deslocamento de cada junta, seja ela rotativa, prismática ou a combinação delas.

2.4.4 Cinemática Inversa

Se na cinemática direta determina-se a localização do efetuador a partir dos movimentos da junta do robô, na cinemática inversa determinam-se os movimentos das juntas para uma determinada posição do efetuador no espaço (TSAI, 1999). Nesse caso a posição final do efetuador é conhecida e os deslocamentos nas juntas são incógnitas. A figura 15 resume graficamente a cinemática direta e inversa.

Figura 15 – Representação gráfica do conceito de cinemática direta e inversa.



Fonte: Craig, 2005. Na cinemática direta é conhecido os deslocamentos de cada junta e com isso determina-se a posição do efetuador. Já na cinemática inversa é conhecida a posição do efetuador e determinam-se os deslocamentos de cada junta.

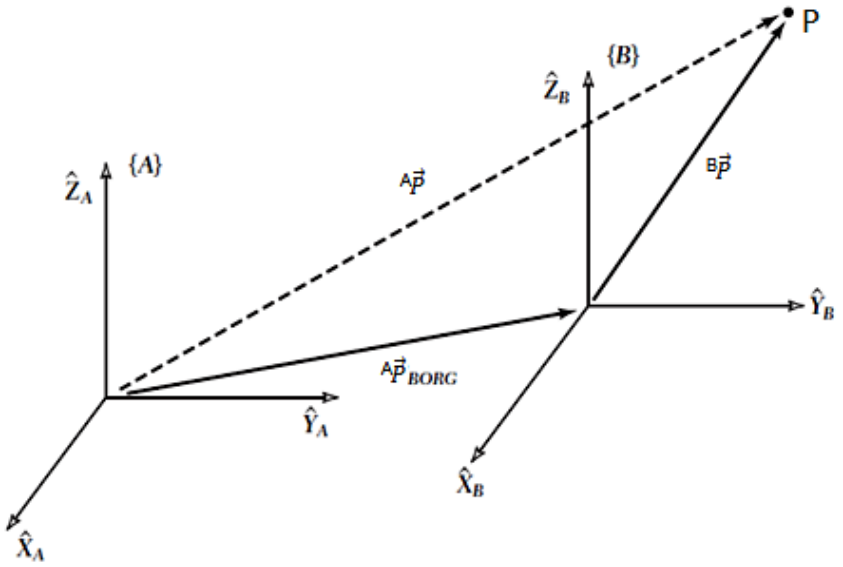
2.5 MATRIZ DE ROTAÇÃO E TRANSFORMAÇÃO HOMOGÊNEA

No estudo da cinemática usa-se como apoio os conceitos de cossenos diretores e transformação homogênea para representar pontos localizados em sistemas de coordenadas diferentes. Cada elo possui seu sistema de coordenada e todos os elos podem ser representados num sistema fixo que não se move com os movimentos do robô (TSAI, 1999). As Figuras 16 e 17 serão usadas como referência para as equações seguintes. Na Figura 16 temos o sistema móvel {B} sem rotacão e na Figura 17 o sistema {B} foi rotacionado sobre seus próprios eixos mantendo sua origem coincidente num sistema fixo {A}.

Na Figura 17 quando um sistema é rotacionado sucessivamente sobre seus próprios eixos temos a definição dos ângulos de Euler (CRAIG, 2005), no exemplo da Figura 17 temos a primeira rotação em torno do eixo $\hat{Z}'_B$ num ângulo ψ, a segunda em torno do eixo $\hat{Y}''_B$ num ângulo φ e a terceira em torno do eixo $\hat{X}'''_B$ num ângulo θ. Esta sequência de rotação pode ser adaptada de acordo com o projeto,

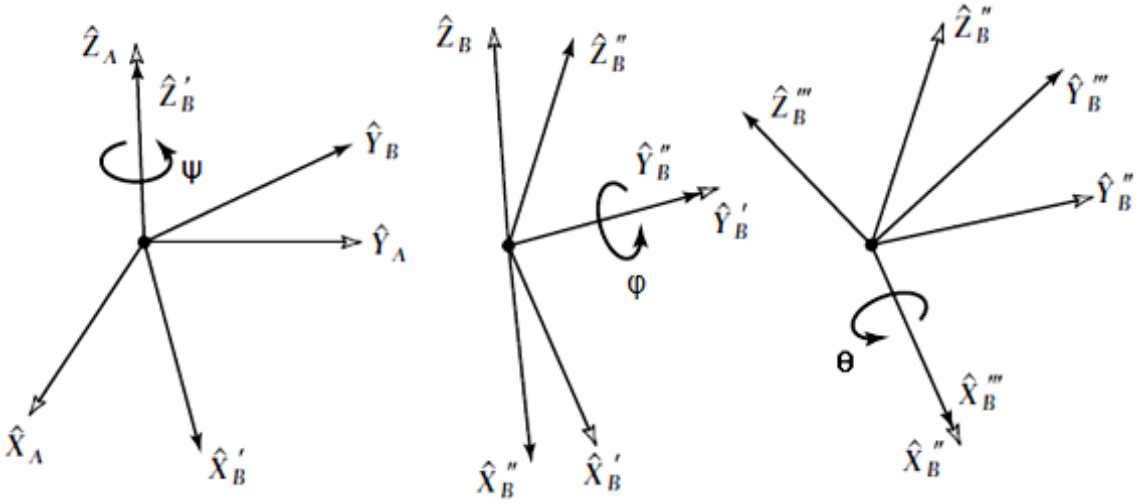
neste trabalho a sequência de rotação adotada será por ângulos de Euler XYZ, onde o eixo X indicará a rolagem (*roll*), o eixo Y a arfagem (*pitch*) e o eixo Z a guinada (*yaw*).

Figura 16 – Representação de um ponto P em um sistema {B} deslocado.



Fonte: Craig, 2005. Na figura acima o ponto P é a posição do efetuador no espaço e pode ser representado em relação ao sistema {B} através de um vetor $^B\vec{P}$ ou em relação ao sistema {A} através do vetor $^A\vec{P}$. Também pode-se representar a origem do sistema {B} no sistema {A} como $^A\vec{P}_{BORG}$.

Figura 17 – Rotação por ângulos de Euler ZYX no sistema {B}.



Fonte: Craig, 2005.

Considerando as rotações do sistema {B} realizadas sucessivamente em torno dos eixos $\hat{Z}'_B$, $\hat{Y}''_B$, e $\hat{X}'''_B$ através dos ângulos ψ, φ e θ pode-se representar as matrizes individuais de cada rotação vistas na referência {A} através dos cossenos diretores abaixo (CRAIG, 2005 e TSAI, 1999):

$$ {}^{A}_{B}R_X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -sen(\theta) \\ 0 & sen(\theta) & \cos(\theta) \end{bmatrix} \quad (1) $$

$$ {}^{A}_{B}R_Y = \begin{bmatrix} \cos(\varphi) & 0 & sen(\varphi) \\ 0 & 1 & 0 \\ -sen(\varphi) & 0 & \cos(\varphi) \end{bmatrix} \quad (2) $$

$$ {}^{A}_{B}R_Z = \begin{bmatrix} \cos(\psi) & -sen(\psi) & 0 \\ sen(\psi) & cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad (3) $$

Para que o sistema {B} da Figura 17 seja totalmente representado no sistema {A} após as três rotações pelos ângulos de Euler multiplicam-se as matrizes de rotação, logo:

$$ {}^{A}_{B}R_{ZYX} = {}^{A}_{B}R_Z \cdot {}^{A}_{B}R_Y \cdot {}^{A}_{B}R_X \quad (4) $$

A equação 4 será reformulada para este trabalho considerando as rotações pelos ângulos de Euler na sequência dos eixos X, Y e Z, resultando a equação 5:

$$ {}^{A}_{B}R_{XYZ} = {}^{A}_{B}R_X \cdot {}^{A}_{B}R_Y \cdot {}^{A}_{B}R_Z \quad (5) $$

Substituindo as equações (1), (2) e (3) em (5) tem-se a matriz de rotação que transporta o sistema {B} para o sistema {A}:

$$ {}^{A}_{B}R_{XYZ} = \begin{bmatrix} \mathrm{c}(\varphi)\mathrm{c}(\psi) & -\mathrm{c}(\varphi)\,s(\psi) & s(\varphi) \\ s(\theta)s(\varphi)c(\psi) + c(\theta)s(\psi) & -\mathrm{s}(\theta)\,s(\varphi)s(\psi) + c(\theta)\mathrm{c}(\psi) & -s(\theta)c(\varphi) \\ -\mathrm{c}(\theta)\,s(\varphi)c(\psi) + s(\theta)s(\psi) & \mathrm{c}(\theta)\,s(\varphi)s(\psi) + s(\theta)c(\psi) & c(\theta)c(\varphi) \end{bmatrix} \quad (6) $$

Para representar completamente um ponto P do sistema {B} em relação ao sistema {A} após sua translação e rotações usa-se a matriz de transformação homogênea (CRAIG, 2005 e TSAI, 1999):

$$\begin{bmatrix} {}^{A}\vec{P} \\ 1 \end{bmatrix} = \begin{bmatrix} {}^{A}_{B}R_{XYZ} & {}^{A}\vec{P}_{BORG} \\ 0\ 0\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} {}^{B}\vec{P} \\ 1 \end{bmatrix} \quad (7)$$

A equação (7) apresenta a matriz de transformação homogênea (TH) usada como operador matemático de transporte de um ponto localizado no sistema {B} para o sistema {A}.

$$\text{TH} = \begin{bmatrix} {}^{A}_{B}R_{XYZ} & {}^{A}\vec{P}_{BORG} \\ 0\ 0\ 0 & 1 \end{bmatrix} \quad (8)$$

No desenvolvimento das equações de movimento e forças de uma plataforma guiada por cabos, vários trabalhos usam a teoria das helicóides para determinar as incógnitas do problema (POTT, 2014 e SIMAS, 2002).

2.6 TEORIA DAS HELICÓIDES

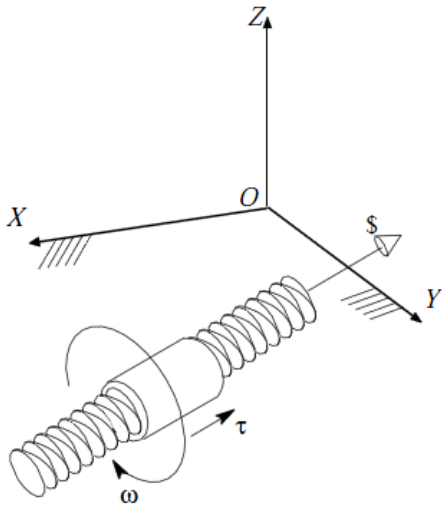
A teoria de helicóides já existe desde 1763 quando foi desenvolvida por Mozzi e pode ser utilizada na representação da cinemática e estática de um robô (MURARO, 2015). Alguns trabalhos como o de Campos (2004), Valdiero (2001), Simas (2002) e Carboni (2012) utilizaram a teoria de helicóides para montar as equações da cinemática e estática de um robô. Também foi desenvolvido um estudo para a geometria cinemática usando a teoria de helicóides (HUNT, 1978 e TSAI, 1999).

Uma helicóide é formada por uma reta que possui uma direção e por um valor escalar que é o passo da helicóide. Quando a reta que forma a helicóide é representada através de um vetor normalizado tem-se uma helicóide normalizada, representada pelo símbolo $\hat{\$}$ (BALL 1900, HUNT 1978, CAMPOS 2004).

Segundo Hunt (2000) o movimento de um objeto pode ser resumido numa rotação em torno de um eixo e numa translação sobre este mesmo eixo. Este

movimento combinado recebe o nome de heligiro e é representado através do símbolo de cifrão $, visto na Figura 18.

Figura 18 – Representação de um heligiro.



Fonte: CAMPOS, 2004. Nesta figura a rotação é representada por ω e a translação por τ.

A representação de um heligiro $ é composta por valores de rotações $[\omega_X, \omega_Y, \omega_Z]^T$ e velocidades lineares $[v_X, v_Y, v_Z]^T$ conforme equação 9:

$$\$ = \begin{bmatrix} \omega_X \\ \omega_Y \\ \omega_Z \\ v_X \\ v_Y \\ v_Z \end{bmatrix} = \begin{bmatrix} \vec{\omega} \\ \vec{v} \end{bmatrix} \quad (9)$$

Assim como um heligiro $ é representado pela dupla rotação e velocidade, uma heliforça $' é representada pela dupla força e momento (CAMPOS, 2004), conforme equação 10.

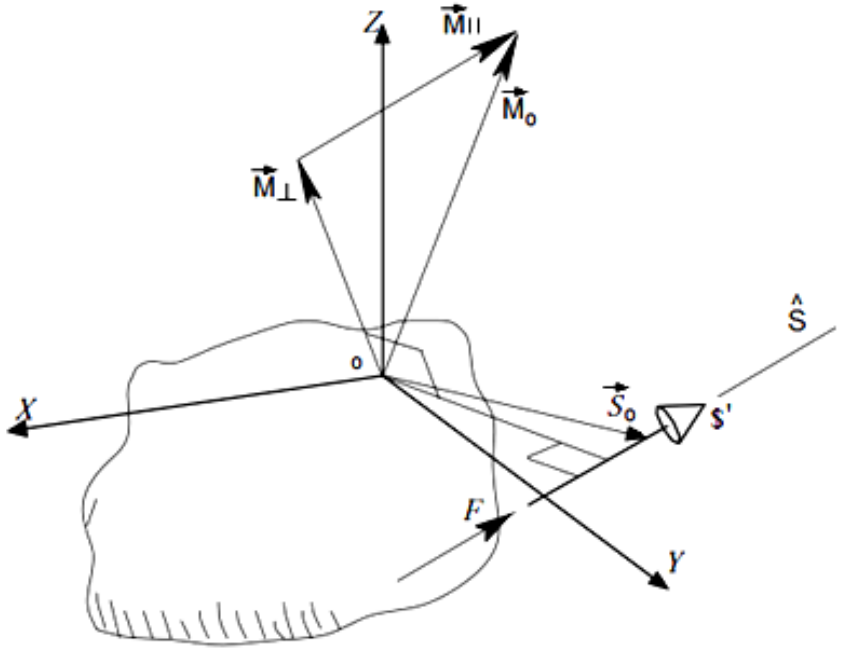
.

$$\$' = \begin{bmatrix} F_X \\ F_Y \\ F_Z \\ M_X \\ M_Y \\ M_Z \end{bmatrix} = \begin{bmatrix} \vec{F} \\ \vec{M} \end{bmatrix} \quad (10)$$

A heliforça também pode ser representada pela equação 11 em termos de uma helicóide normalizada e a magnitude da força. A figura 19 mostra os vetores dessa normalização.

$$\$' = \hat{\$}' \cdot F = \begin{bmatrix} \hat{S} \\ \vec{S}_0 \times \hat{S} \end{bmatrix} \cdot F \qquad (11)$$

Figura 19 – Representação de uma heliforça.



Fonte: CAMPOS, 2004.

Na figura 19 o vetor $\vec{S}_0$ é um vetor qualquer que parte da origem do sistema fixo {O} até o eixo de aplicação da força cuja direção unitária é representada por $\hat{S}$. Os vetores de momento perpendicular e paralelo ao eixo estão representados na figura para indicar o momento resultante $\vec{M}_0$ da força aplicada no corpo rígido.

2.6.1 Matriz Jacobiana

No estudo da cinemática diferencial a aplicação da matriz Jacobiana é utilizada para representar a velocidade do efetuador em relação as velocidade das juntas, da mesma forma que a matriz de transformação homogênea relaciona os

ângulos das juntas com a posição X, Y e Z do efetuador (SIMAS, 2002). Neste trabalho as juntas são os cabos e o efetuador é a plataforma móvel.

Assim como uma matriz de transformação homogênea é definida entre o espaço das juntas e o espaço cartesiano, a matriz Jacobiana é definida como uma relação entre a velocidade cartesiana e a velocidade angular (TSAI, 1999), a equação 12 define a matriz Jacobiana clássica:

$$\begin{bmatrix} \vec{v} \\ \vec{\omega} \end{bmatrix} = J \cdot \dot{q} \tag{12}$$

Onde $\vec{v}$ é o vetor velocidade no espaço cartesiano, $\vec{\omega}$ é o vetor de velocidade angular e $\dot{q}$ é o vetor de velocidades no espaço de coordenadas das juntas do robô.

A cinemática inversa de um robô pode ser determinada analiticamente e numericamente. No processo analítico determina-se uma função inversa que relacione o espaço das juntas (ângulos ψ, φ e θ) com o espaço cartesiano (valores de X, Y e Z). E no processo numérico a intenção é resolver a equação 12 por integração e prever o posicionamento do robô em torno de sua última posição (SIMAS, 2002).

Se um robô apresentar mais números de juntas do que precisa para realizar seu movimento então a metodologia analítica torna-se um sistema indeterminado e neste caso a saída é utilizar a equação 12 (SIMAS, 2002) pelo método de inversão da matriz pseudo-inversa (TSAI, 1999), neste caso a equação 12 é manipulada para resultar na equação 13.

$$\dot{q} = J^{-1} \cdot \begin{bmatrix} \vec{v} \\ \vec{\omega} \end{bmatrix} \tag{13}$$

2.6.2 Matriz Jacobiana Helicoidal

Pusey (2003) define que para relacionar o peso da plataforma com as forças em cada cabo é possível fazer uma comparação com a equação 12, resultando na equação 14.

$$\left[\begin{matrix}\vec{F}\\ \vec{M}\end{matrix}\right]_W = \sum_{i=1}^{4} \$'_i \tag{14}$$

Onde $\left[\begin{matrix}\vec{F}\\ \vec{M}\end{matrix}\right]_W$ é a heliforça referente ao peso da plataforma e $\$'_i$ é a heliforça de cada cabo (i=1 a 4). Substituindo a equação 11 em 14 chega-se na equação 15 para a heliforça do peso.

$$\$'_W = \sum_{i=1}^{4}\left(\hat{\$}'_i \cdot F_i\right) = \left[\hat{\$}'_1 \;\; \hat{\$}'_2 \;\; \hat{\$}'_3 \;\; \hat{\$}'_4\right] \cdot \begin{bmatrix}F_1\\F_2\\F_3\\F_4\end{bmatrix} = J'^{T} \cdot \begin{bmatrix}F_1\\F_2\\F_3\\F_4\end{bmatrix} \tag{15}$$

Onde J'^{T} é o jacobiano transposto das heliforças unitárias. Manipulando a equação 15 de forma análoga à equação 13 chega-se na equação 16.

$$\dot{F} = J'^{T-1} \cdot \$'_W \tag{16}$$

Onde $\dot{F}$ é o vetor que contém as magnitudes das forças.

O programa do apêndice 3 foi desenvolvido no software SciLab (ENTERPRISES, 2017) e utiliza a matriz jacobiana para o cálculo da localização e das forças nos cabos do robô. O trecho da tabela 1 abaixo foi retirado do programa.

Tabela 1 – Função em SciLab para cálculo do Jacobiano inverso

```
function [x, f, ea, iter] =
NewtonRaphson(f1, f2, f3, f4, f5, f6, x0, es, maxit, delta)
 interacoes = 0;   x = x0;
    while(1)
        [J,f] = jacobiano(f1, f2, f3, f4, f5, f6, x, delta);
        x_anterior = x;
        x = x - J\f;
  interacoes = iter+1;
  ea = 100*max(abs((x-x_anterior) ./x));
    if interacoes >= maxit || ea<=es then
          break
     end
 end
endfunction
```

2.7 MÉTODOS ITERATIVOS EM UM SISTEMA NÃO-LINEAR

Quando não é possível aproximar um sistema de equação não linear por um sistema linear a abordagem mais simples é o método iterativo, como os métodos clássicos de Newton, Broyden, Halley e também métodos recentes derivados dos clássicos, como o método de Potra-Pták, Chun e Ponto Médio (SOUZA, 2015). O método utilizado neste trabalho foi o de Newton devido ao conhecimento da posição inicial do robô.

2.7.1 Médo de Newton

Também chamado de método de Newton-Raphson, utilizado porque obtem-se uma convergência quadrática quando se arbitra um valor inicial próximo da solução e há a correção do erro de arredondamento ao longo das iterações (SOUZA, 2015).

O método de Newton pode convergir rapidamente quando uma aproximação inicial está próxima da solução exata, porém o método pode falhar se os valores iniciais de entrada estiverem longe da solução exata. (SOUZA, 2015).

A equação 17 define o método de Newton-Raphson:

$$x_{k+1} = x_k - J^{-1}(x_k)F(x_k) \tag{17}$$

Como visto na equação 17, um ponto fraco deste método é a necessidade de calcular a matriz jacobiana e sua inversa em cada iteração.

O algoritmo de Newton-Raphson está representado na tabela 2 e sempre que viável é utilizado como primeira opção em relação aos outros métodos (SOUZA, 2015).

Tabela 2 – Algoritmo de Newton-Raphson

Entrada:
Função F(x);
matriz Jacobiana J(x);
aproximação inicial x0;
número máximo de iterações nMax;
tolerância: tol

Saída: Solução aproximada $x = (x1, \ldots, xn)^T$

início
Para k = 0, 1, 2, . . . , (nMax − 1) faça
Calcular $F(x_k)$ e $J(x_k)$;
Resolver o sistema $J(x_k)s_k = -F(x_k)$ para determinar s_k;
$x_{k+1} \leftarrow x_k + s_k$;
se $|f_i(x_{k+1})| < tol \ \forall \ 1 \leq i \leq nMax$ então
Saída(x_{k+1});
Parar;
fim
fim
fim

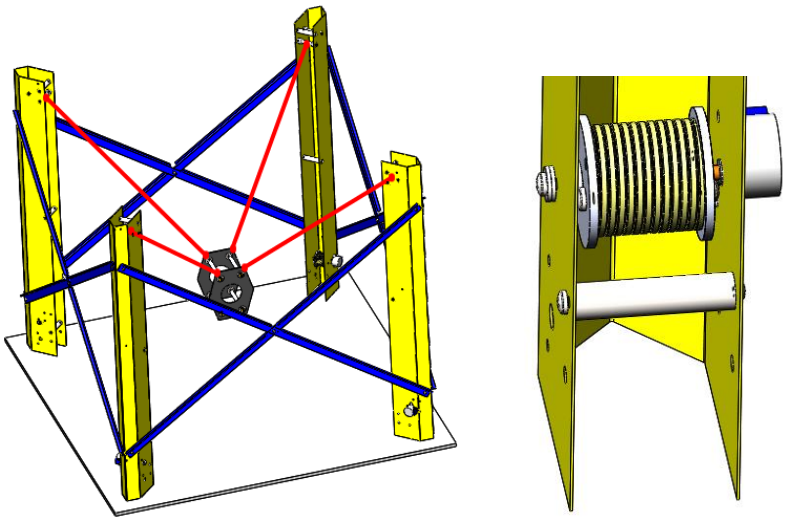
3 MATERIAIS E MÉTODOS

Este trabalho propõe a construção de um robô guiado por cabos para a análise da localização e forças nos cabos de uma plataforma móvel. O robô é constituído de estruturas metálicas feitas com chapas, sensores, cabos, microcontroladores e um computador para recebimento e envio de comandos ao robô. Antes das construção do robô real foi feita simulações em 3D para realizar algumas medições que pudessem ser utilizadas nos cálculos da cinemática em conjunto com a teoria de helicóides.

3.1 MODELAGEM 3D DO ROBO GUIADO POR CABOS

O mecanismo proposto na Figura 20 consiste de uma plataforma móvel (o objeto a ser simulado) ligada por quatro cabos atuados que podem ser enrolados ou desenrolados através de carretéis conectados a motores posicionados nas quatro vigas fixas que formam a base fixa do robô. As variáveis a serem controladas são os deslocamentos nos eixos X, Y e Z e o ângulo de rolagem (*roll*) ou arfagem (*pitch*).

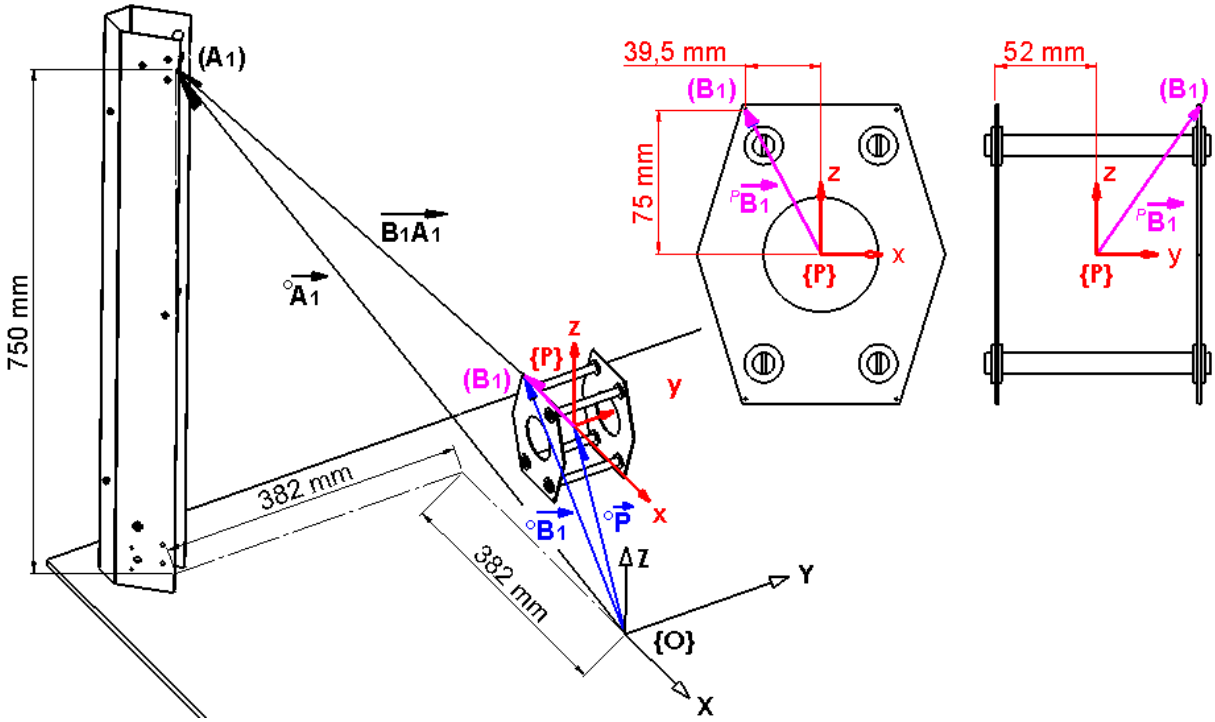
Figura 20 - Estrutura do robô (esquerda) e motor com carretel (direita).



Fonte: Produção do autor. Na imagem do carretel uma rosca é usada para o cabo não sobrepor-se quando o motor puxá-lo.

Os vetores e as medidas fixas representados na figura 21 foram utilizados para a formulação das equações de cinemática do robô.

Figura 21 – Vetores e medidas do robô

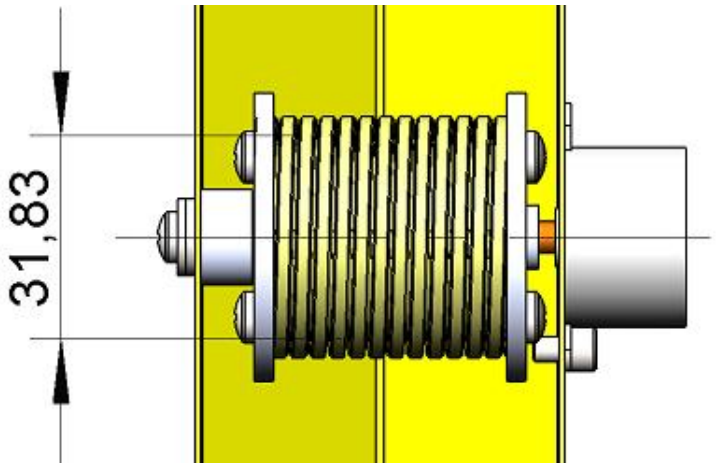


Fonte: Autor. O sistema fixo é representado por {O} através dos eixos X, Y e Z e o sistema móvel está representado por {P} através dos eixos x, y e z. A estrutura completa ste robô pode ser visualizada no link disponível junto ao apêndice dos programas.

Na figura 21 o vetor ${}^{o}\vec{A}_1$ é a representação do ponto (A_1) no sistema fixo {O}, assim como ${}^{o}\vec{B}_1$ e ${}^{o}\vec{P}$. O vetor ${}^{P}\vec{B}_1$ é a representação do ponto (B_1) no sistema {P}. A norma do vetor $\overrightarrow{B_1A_1}$ fornece o comprimento do cabo do ponto (B_1) até (A_1).

O posicionamento da plataforma móvel é realizado através dos cabos que são enrolados ou desenrolados por roldanas fixadas a motores de passo montados na estrutura das torres. As roldanas (figura 22) possuem uma rosca para acomodar os cabos, a cavidade foi calculada para que em uma volta do motor o cabo mova-se numa distância de 100 mm, facilitando os cálculos para a movimentação dos motores.

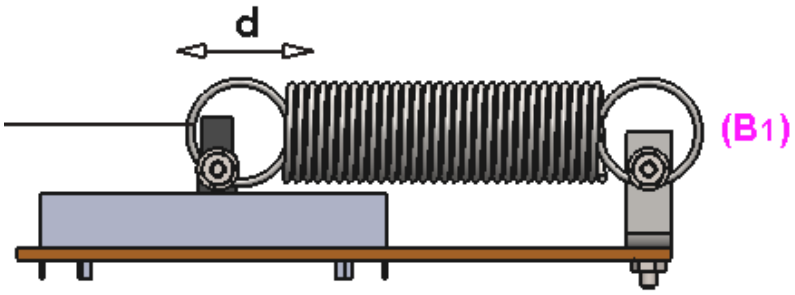
Figura 22 – Detalhe das roldanas para movimentação dos cabos



Fonte: Autor. O valor do diâmetro de 31,83 mm foi determinado para o motor enrolar ou desenrolar um comprimento de 100 mm a cada volta da circunferência, ou seja: 100 = 2πR. O detalhamento completo pode ser visualizado no link disponível com os programas do apêndice.

Além da modelagem da estrutura em 3D também foi modelada a montagem do sensor de força visto na figura 23.

Figura 23 – Modelagem 3D do sensor de força



Fonte: Autor. O deslocamento da mola causado pelo cabo movimenta o potenciômetro. A extremidade oposta da mola é conectada na plataforma móvel num furo localizado no ponto (B_1).

Este sensor é composto por um potenciômetro deslizante fixo a uma mola que ao ser deslocado envia um sinal analógico para um microcontrolador. Este sinal é usado para calcular a força que a mola está sujeita durante o tensionamento dos cabos. Para o cálculo da força foi realizada primeiramente o cálculo do fator K da mola a partir da medição do seu deslocamento a cada aplicação de carga, a tabela 3 apresenta os valores de deslocamento da mola para as cargas aplicadas.

A equação 18 representa a lei de Hook para deformações elásticas (WOLFGANG, 2012) e foi utilizada para o cálculo do fator K.

$$F - 1{,}471 = K \cdot d \tag{18}$$

Tabela 3 - Tabela para determinação do fator K da mola

Carga [N]	d [m]	K [N/m]	Desvio [N/m]	Desvio Padrão [N/m]
1,4715	0	Pré-carga	- - -	- - -
1,6670	0,0010	196,1325	0,0069	0,015
1,9610	0,0025	196,1328	0,0072	
2,0590	0,0030	196,1330	0,0074	
2,4520	0,0050	196,1330	0,0074	
2,6480	0,0060	196,1170	-0,0086	
3,0400	0,0080	196,1253	-0,0003	
3,4320	0,0100	196,1330	0,0074	
3,6280	0,0110	196,1330	0,0074	
3,8240	0,0120	196,0835	0,0421	
3,9230	0,0125	196,1330	0,0074	
	Média:	196,1256	- - -	- - -

A tabela 3 apresenta o fator K determinado para cada carga e a média de 196,1256 N/m. Considerando e desvio padrão, o valor de 196±0.015 N/m foi adotado nas equações para o cálculo das forças. Ainda na tabela 1 há uma pré-carga da mola de 1,4715 N/m, até esta carga a mola não apresentou deslocamento.

3.2 CINEMÁTICA INVERSA DO ROBÔ GUIADO POR CABOS

Para o cálculo da cinemática do robô é preciso determinar primeiramente o comprimento de cada cabo, que é a norma do vetor $\overrightarrow{B_1A_1}$ da figura 21, a relação entre os vetores é dada pela equação 19.

$$\overrightarrow{B_1A_1} = {}^{o}\vec{A}_1 - {}^{o}\vec{B}_1 \quad (19)$$

No entanto o vetor ${}^{o}\vec{B}_1$ sofre mudanças devido ao deslocamento e orientação do sistema móvel {P}. A equação 7 é aplicada para a determinação do vetor ${}^{o}\vec{B}_1$ resultando na equação 20 para o robô proposto.

$$\begin{bmatrix} {}^{o}\vec{B}_1 \\ 1 \end{bmatrix} = \begin{bmatrix} c(\varphi)c(\psi) & -c(\varphi)\,s(\psi) & s(\varphi) & PX \\ s(\theta)s(\varphi)c(\psi)+c(\theta)s(\psi) & -s(\theta)\,s(\varphi)s(\psi)+c(\theta)c(\psi) & -s(\theta)c(\varphi) & PY \\ -c(\theta)\,s(\varphi)c(\psi)+s(\theta)s(\psi) & c(\theta)\,s(\varphi)s(\psi)+s(\theta)c(\psi) & c(\theta)c(\varphi) & PZ \\ 0 & 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} B1x \\ B1y \\ B1z \end{bmatrix} \quad (20)$$

Se a orientação e a localização do sistema {P} forem dadas então através da equação 20 acha-se o vetor ${}^{o}\vec{B}_1$, pois os valores de B_{1x}, B_{1y} e B_{1z} são conhecidos da modelagem 3D da figura 21.

O vetor ${}^{o}\vec{A}_1$ na equação 19 também é conhecido e seus valores são vistos na figura 21. O comprimento $(B_1)(A_1)$ é a norma do vetor $\overrightarrow{B_1A_1}$ dada pela equação 21.

$$(B_1)(A_1) = \overrightarrow{\|B_1A_1\|} = \sqrt{\left({}^{o}A_{1X} - {}^{o}B_{1X}\right)^2 + \left({}^{o}A_{1Y} - {}^{o}B_{1Y}\right)^2 + \left({}^{o}A_{1Z} - {}^{o}B_{1Z}\right)^2} \quad (21)$$

O comprimento dos outros cabos é calculado da mesma forma e as equações 22, 23 e 24 resumem os passos anteriores feitos para $\overrightarrow{\|B_1A_1\|}$.

$$(B_2)(A_2) = \overrightarrow{\|B_2A_2\|} = \sqrt{\left({}^{o}A_{2X} - {}^{o}B_{2X}\right)^2 + \left({}^{o}A_{2Y} - {}^{o}B_{2Y}\right)^2 + \left({}^{o}A_{2Z} - {}^{o}B_{2Z}\right)^2} \quad (22)$$

$$(B_3)(A_3) = \overrightarrow{\|B_3A_3\|} = \sqrt{\left({}^{o}A_{3X} - {}^{o}B_{3X}\right)^2 + \left({}^{o}A_{3Y} - {}^{o}B_{3Y}\right)^2 + \left({}^{o}A_{3Z} - {}^{o}B_{3Z}\right)^2} \quad (23)$$

$$(B_4)(A_4) = \overrightarrow{\|B_4A_4\|} = \sqrt{\left({}^{o}A_{4X} - {}^{o}B_{4X}\right)^2 + \left({}^{o}A_{4Y} - {}^{o}B_{4Y}\right)^2 + \left({}^{o}A_{4Z} - {}^{o}B_{4Z}\right)^2} \quad (24)$$

Para o sistema {P} localizado na origem do sistema {O} e sem orientação (θ=0°, φ=0° e ψ=0°) acha-se o comprimento inicial de cada cabo pelas equações 21, 22, 23 e 24. A matriz de rotação da equação 6 dos pontos (B_1), (B_2), (B_3) e (B_4) para todos os ângulos iguais a zero é:

$${}^{O}_{B1}R_{XYZ} = {}^{O}_{B2}R_{XYZ} = {}^{O}_{B3}R_{XYZ} = {}^{O}_{B4}R_{XYZ} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Os valores dos vetores ${}^{P}\vec{B}_1$, ${}^{P}\vec{B}_2$, ${}^{P}\vec{B}_3$ e ${}^{P}\vec{B}_4$ são fixos no modelo 3D:

$${}^{P}\vec{B}_1 = \begin{bmatrix} -0.0395 \\ -0.0520 \\ 0.0750 \end{bmatrix} \quad {}^{P}\vec{B}_2 = \begin{bmatrix} -0.0395 \\ 0.0520 \\ 0.0750 \end{bmatrix} \quad {}^{P}\vec{B}_3 = \begin{bmatrix} 0.0395 \\ 0.0520 \\ 0.0750 \end{bmatrix} \quad {}^{P}\vec{B}_4 = \begin{bmatrix} 0.0395 \\ -0.0520 \\ 0.0750 \end{bmatrix}$$

A equação 19 resolvida para os vetores ${}^{o}\vec{B}_1$, ${}^{o}\vec{B}_2$, ${}^{o}\vec{B}_3$ e ${}^{o}\vec{B}_4$ também resulta nos valores dos vetores ${}^{P}\vec{B}_1$, ${}^{P}\vec{B}_2$, ${}^{P}\vec{B}_3$ e ${}^{P}\vec{B}_4$ uma vez que o sistema {O} coincide com {P} na posição inicial. O comprimento (B_i)(A_i) para o sistema {P} na posição inicial resulta na norma dos vetores $\overrightarrow{B_iA_i}$ (i=1 a 4):

$$\overrightarrow{\|B_1A_1\|} = \sqrt{(-0.3820 + 0.0395)^2 + (-0.3820 + 0.0520)^2 + (0.7500 - 0.0750)^2} = 0.8257 \text{ m}$$

$$\overrightarrow{\|B_2A_2\|} = \sqrt{(-0.3820 + 0.0395)^2 + (0.3820 - 0.0520)^2 + (0.7500 - 0.0750)^2} = 0.8257 \text{ m}$$

$$\overrightarrow{\|B_3A_3\|} = \sqrt{(-0.3820 + 0.0395)^2 + (-0.3820 + 0.0520)^2 + (0.7500 - 0.0750)^2} = 0.8257 \text{ m}$$

$$\overrightarrow{\|B_4A_4\|} = \sqrt{(-0.3820 + 0.0395)^2 + (-0.3820 + 0.0520)^2 + (0.7500 - 0.0750)^2} = 0.8257 \text{ m}$$

Os comprimentos de cada cabo podem ser calculados a partir da entrada de valores de orientação pelos ângulos θ, φ e ψ e de localização da plataforma pelos valores de P_X, P_Y e P_Z, caracterizando o conceito da cinemática inversa, onde a posição e orientação da plataforma são dadas.

Para o cálculo dos valores reais dos comprimentos dos cabos L_i e da medida M_i que cada motor deve enrolar ou desenrolar o cabo (i=1 a 4) deve-se considerar o comprimento inicial fixo de 0.057 m da mola e sua deformação d, conforme equação 18. Os valores dos comprimentos dos cabos L_i (i=1 a 4) são geometricamente representados como a distância dos pontos (A_i) localizados na estrutura fixa até o contato com a mola.

$L_i = [\|\overrightarrow{B_i A_i}\| - (0.057 + d)]$, i=1 a 4. (25)

O valor de L_i é atualizado para cada posição da plataforma e usado nos programas para o cálculo de movimentação dos motores. Por exemplo, se $P_X = 0$, $P_Y = 0$ e $P_Z = 0.350$ m e considerando os ângulos de orientações nulos então o comprimento $(B_i)(A_i)$ para i=1 a 4 é de 0.5760 m, calculado através do programa do apêndice 2 desenvolvido no software WxMaxima (VODOPIVEC, 2012).

Para um comprimento de 0.5760 m se o comprimento inicial dos cabos for de 0.8257 m então os motores devem puxar um comprimento de cabo de [0.8257 – (0.057 + 0)] – [0.5760 – (0.057 + d)]. O cálculo da deformação d da mola depende das forças atuantes nos cabos e são equacionadas através da estática do robô.

3.3 EQUAÇÕES DE ESTÁTICA DO ROBÔ

As equações para o cálculo das forças em cada cabo dependem do peso e da localização da plataforma, usando a equação 11 para uma somatória de heliforças chega-se a equação 26, onde o peso da plataforma é representado por W:

$\$'_1 + \$'_2 + \$'_3 + \$'_4 = \hat{\$}'_1 \cdot F_1 + \hat{\$}'_2 \cdot F_2 + \hat{\$}'_3 \cdot F_3 + \hat{\$}'_4 \cdot F_4 = \hat{\$}'_W \cdot W$ (26)

Os vetores normalizados $\hat{\$}'_1$, $\hat{\$}'_2$, $\hat{\$}'_3$ e $\hat{\$}'_4$ também foram definidos na equação 11 e para o robô modelado pode-se escolher de acordo com a figura 19 os valores dos vetores ${}^o\vec{B}_1$, ${}^o\vec{B}_2$, ${}^o\vec{B}_3$ e ${}^o\vec{B}_4$ para representar $\vec{S}_0$ na matriz. Já o vetor unitário $\hat{S}$ presente na matriz da equação 11 pode ser substituído pela direção de aplicação da força no cabo, assim a equação 27 representa de forma análoga a equação 11 aplicada no robô proposto neste trabalho.

$\$_i' = \hat{\$}_i \cdot F_i = \begin{bmatrix} \hat{S}i_{BA} \\ \overrightarrow{{}^oB_i} \times \hat{S}i_{BA} \end{bmatrix} \cdot F_i$, para i = 1 a 4 (27)

O vetor normalizado $\hat{S}i_{BA}$ é calculado através da equação 28.

$$\hat{S}i_{BA} = \left[\frac{\left({}^{o}\vec{A}_i - {}^{o}\vec{B}_i\right)}{\|\overrightarrow{B_iA_i}\|}\right] \text{ , para i = 1 a 4} \tag{28}$$

A equação 29 é a representação de um vetor unitário normalizado aplicado da direção do cabo 1. A representação para os demais cabos segue o mesmo raciocínio.

$$\hat{S}_{1BA} = \begin{bmatrix} \dfrac{\left({}^{\circ}A_{1X} - {}^{\circ}B_{1X}\right)}{\sqrt{\left({}^{\circ}A_{1X} - {}^{\circ}B_{1X}\right)^2 + \left({}^{\circ}A_{1Y} - {}^{\circ}B_{1Y}\right)^2 + \left({}^{\circ}A_{1Z} - {}^{\circ}B_{1Z}\right)^2}} \\ \dfrac{\left({}^{\circ}A_{1Y} - {}^{\circ}B_{1Y}\right)}{\sqrt{\left({}^{\circ}A_{1X} - {}^{\circ}B_{1X}\right)^2 + \left({}^{\circ}A_{1Y} - {}^{\circ}B_{1Y}\right)^2 + \left({}^{\circ}A_{1Z} - {}^{\circ}B_{1Z}\right)^2}} \\ \dfrac{\left({}^{\circ}A_{1Z} - {}^{\circ}B_{1Z}\right)}{\sqrt{\left({}^{\circ}A_{1X} - {}^{\circ}B_{1X}\right)^2 + \left({}^{\circ}A_{1Y} - {}^{\circ}B_{1Y}\right)^2 + \left({}^{\circ}A_{1Z} - {}^{\circ}B_{1Z}\right)^2}} \end{bmatrix} \tag{29}$$

O resultado do produto $\overrightarrow{{}^{o}B_i}$ x $\widehat{Si}_{BA}$ aplicado ao cabo 1 e a matriz da heliforça para o cabo 1, representada por $\$'_1$, podem ser encontrados no Apêndice 1.

A heliforça do peso $\widehat{\$}'_W \cdot$ W é calculada pela equação 11 da mesma forma e obtém-se a equação 30.

$$\$'_W = \hat{\$}_W \cdot \text{W} = \begin{bmatrix} 0 \\ 0 \\ -1 \\ \overrightarrow{P} \text{ x } \begin{bmatrix} 0 \\ 0 \\ -1 \end{bmatrix} \end{bmatrix} \cdot \text{W} \tag{30}$$

Onde $[0\ 0\ -1]^T$ é a direção unitária do peso na direção Z. O peso da plataforma é indicado pela letra W. O vetor $\vec{S}_0$ está sendo representado pelo vetor $\overrightarrow{P}$ na equação 30 porque também é um vetor que sai da origem fixa {O} e vai de encontro ao eixo de aplicação da força peso.

Resolvendo a equação 29 chega-se na equação 31 da heliforça que representa a força peso:

$$\$'_W = \begin{bmatrix} 0 \\ 0 \\ -W \\ -P_Y \cdot W \\ P_X \cdot W \\ 0 \end{bmatrix} \tag{31}$$

Realizando o equilíbrio de forças a partir da equação 25 é possível obter um sistema linear contendo seis equações e seis incógnitas, conforme as equações 31, 32, 33, 34 e 35.

$$\$'_1 = \begin{bmatrix} \frac{\left({}^{o}A_{1X} - {}^{o}B_{1X}\right)}{\overrightarrow{\|B_1A_1\|}} \\ \frac{\left({}^{o}A_{1Y} - {}^{o}B_{1Y}\right)}{\overrightarrow{\|B_1A_1\|}} \\ \frac{\left({}^{o}A_{1Z} - {}^{o}B_{1Z}\right)}{\overrightarrow{\|B_1A_1\|}} \\ \frac{{}^{o}B_{1Y} \cdot \left({}^{o}A_{1Z} - {}^{o}B_{1Z}\right)}{\overrightarrow{\|B_1A_1\|}} - \frac{{}^{o}B_{1Z} \cdot \left({}^{o}A_{1Y} - {}^{o}B_{1Y}\right)}{\overrightarrow{\|B_1A_1\|}} \\ \frac{{}^{o}B_{1Z} \cdot \left({}^{o}A_{1X} - {}^{o}B_{1X}\right)}{\overrightarrow{\|B_1A_1\|}} - \frac{{}^{o}B_{1X} \cdot \left({}^{o}A_{1Z} - {}^{o}B_{1Z}\right)}{\overrightarrow{\|B_1A_1\|}} \\ \frac{{}^{o}B_{1X} \cdot \left({}^{o}A_{1Y} - {}^{o}B_{1Y}\right)}{\overrightarrow{\|B_1A_1\|}} - \frac{{}^{o}B_{1Y} \cdot \left({}^{o}A_{1X} - {}^{o}B_{1X}\right)}{\overrightarrow{\|B_1A_1\|}} \end{bmatrix} \cdot F_1 \tag{32}$$

$$\$'_2 = \begin{bmatrix} \frac{\left({}^{o}A_{2X} - {}^{o}B_{2X}\right)}{\|\overrightarrow{B_2A_2}\|} \\ \frac{\left({}^{o}A_{2Y} - {}^{o}B_{2Y}\right)}{\|\overrightarrow{B_2A_2}\|} \\ \frac{\left({}^{o}A_{2Z} - {}^{o}B_{2Z}\right)}{\|\overrightarrow{B_2A_2}\|} \\ \frac{{}^{o}B_{2Y}\cdot\left({}^{o}A_{2Z} - {}^{o}B_{2Z}\right)}{\|\overrightarrow{B_2A_2}\|} - \frac{{}^{o}B_{2Z}\cdot\left({}^{o}A_{2Y} - {}^{o}B_{2Y}\right)}{\|\overrightarrow{B_2A_2}\|} \\ \frac{{}^{o}B_{2Z}\cdot\left({}^{o}A_{2X} - {}^{o}B_{2X}\right)}{\|\overrightarrow{B_2A_2}\|} - \frac{{}^{o}B_{2X}\cdot\left({}^{o}A_{2Z} - {}^{o}B_{2Z}\right)}{\|\overrightarrow{B_2A_2}\|} \\ \frac{{}^{o}B_{2X}\cdot\left({}^{o}A_{2Y} - {}^{o}B_{2Y}\right)}{\|\overrightarrow{B_2A_2}\|} - \frac{{}^{o}B_{2Y}\cdot\left({}^{o}A_{2X} - {}^{o}B_{2X}\right)}{\|\overrightarrow{B_2A_2}\|} \end{bmatrix} \cdot F_2 \quad (33)$$

$$\$'_3 = \begin{bmatrix} \frac{\left({}^{o}A_{3X} - {}^{o}B_{3X}\right)}{\|\overrightarrow{B_3A_3}\|} \\ \frac{\left({}^{o}A_{3Y} - {}^{o}B_{3Y}\right)}{\|\overrightarrow{B_3A_3}\|} \\ \frac{\left({}^{o}A_{3Z} - {}^{o}B_{3Z}\right)}{\|\overrightarrow{B_3A_3}\|} \\ \frac{{}^{o}B_{3Y}\cdot\left({}^{o}A_{3Z} - {}^{o}B_{3Z}\right)}{\|\overrightarrow{B_3A_3}\|} - \frac{{}^{o}B_{3Z}\cdot\left({}^{o}A_{3Y} - {}^{o}B_{3Y}\right)}{\|\overrightarrow{B_3A_3}\|} \\ \frac{{}^{o}B_{3Z}\cdot\left({}^{o}A_{3X} - {}^{o}B_{3X}\right)}{\|\overrightarrow{B_3A_3}\|} - \frac{{}^{o}B_{3X}\cdot\left({}^{o}A_{3Z} - {}^{o}B_{3Z}\right)}{\|\overrightarrow{B_3A_3}\|} \\ \frac{{}^{o}B_{3X}\cdot\left({}^{o}A_{3Y} - {}^{o}B_{3Y}\right)}{\|\overrightarrow{B_3A_3}\|} - \frac{{}^{o}B_{3Y}\cdot\left({}^{o}A_{3X} - {}^{o}B_{3X}\right)}{\|\overrightarrow{B_3A_3}\|} \end{bmatrix} \cdot F_3 \quad (34)$$

$$\$'_4 = \begin{bmatrix} \frac{\left({}^{o}A_{4X} - {}^{o}B_{4X}\right)}{\|\overrightarrow{B_4A_4}\|} \\ \frac{\left({}^{o}A_{4Y} - {}^{o}B_{4Y}\right)}{\|\overrightarrow{B_4A_4}\|} \\ \frac{\left({}^{o}A_{4Z} - {}^{o}B_{4Z}\right)}{\|\overrightarrow{B_4A_4}\|} \\ \frac{{}^{o}B_{4Y}\cdot\left({}^{o}A_{4Z} - {}^{o}B_{4Z}\right)}{\|\overrightarrow{B_4A_4}\|} - \frac{{}^{o}B_{4Z}\cdot\left({}^{o}A_{4Y} - {}^{o}B_{4Y}\right)}{\|\overrightarrow{B_4A_4}\|} \\ \frac{{}^{o}B_{4Z}\cdot\left({}^{o}A_{4X} - {}^{o}B_{4X}\right)}{\|\overrightarrow{B_4A_4}\|} - \frac{{}^{o}B_{4X}\cdot\left({}^{o}A_{4Z} - {}^{o}B_{4Z}\right)}{\|\overrightarrow{B_4A_4}\|} \\ \frac{{}^{o}B_{4X}\cdot\left({}^{o}A_{4Y} - {}^{o}B_{4Y}\right)}{\|\overrightarrow{B_4A_4}\|} - \frac{{}^{o}B_{4Y}\cdot\left({}^{o}A_{4X} - {}^{o}B_{4X}\right)}{\|\overrightarrow{B_4A_4}\|} \end{bmatrix} \cdot F_4 \quad (35)$$

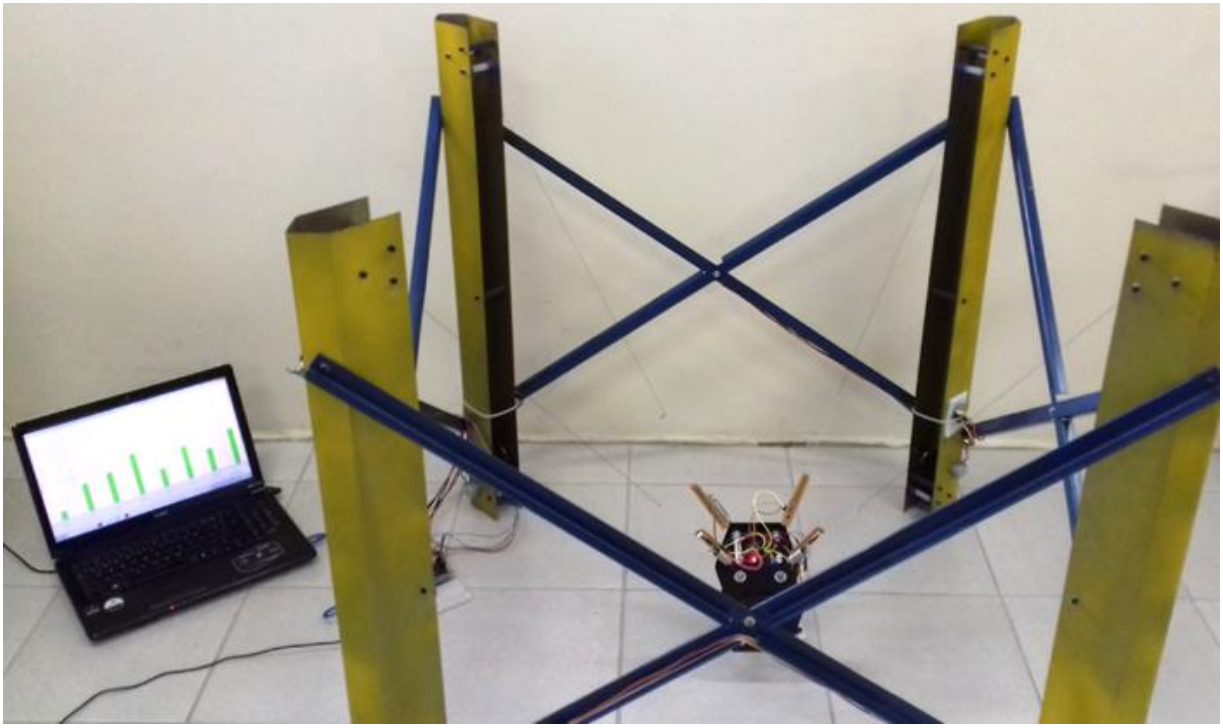
Como cada vetor ${}^{o}\vec{B}_i$ (i=1 a 4) possui os ângulos θ, φ e ψ em sua formulação e a matriz de heliforça também contém as coordenadas P_X, P_Y e P_Z em sua estrutura, o sistema linear $\hat{\$}'_1 \cdot F_1 + \hat{\$}'_2 \cdot F_2 + \hat{\$}'_3 \cdot F_3 + \hat{\$}'_4 \cdot F_4 = \hat{\$}_W \cdot W$ é resolvido se os valores de P_X, P_Y, P_Z, θ, φ e ψ forem conhecidos, resultando nos valores das magnitudes das forças nos cabos e nos seus comprimentos.

Por outro lado, se os valores das forças forem conhecidos então os valores de P_X, P_Y, P_Z, θ, φ e ψ podem ser encontrados através do método iterativo de Newton-Raphson usando condições de contorno próximas da última posição e orientação da plataforma móvel, neste caso as equações são não-lineares e determinadas através do programa desenvolvido no software SciLab disponibilizado no apêndice 3.

3.4 CONSTRUÇÃO DO ROBÔ

A estrutura fixa do robô foi feita de chapas metálicas suportadas por barras laterais parafusadas, conforme figura 24.

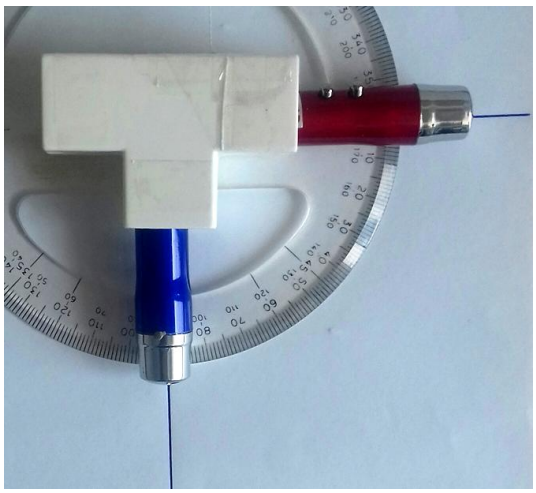
Figura 24 – Estrutura real do robô



Fonte: Autor.

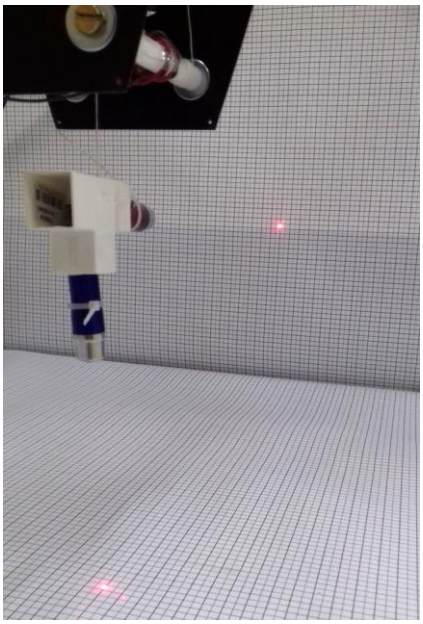
Para realizar a medição das posições X, Y e Z da origem da plataforma móvel foi utilizada uma montagem de apontadores *laser* conforme figuras 25 e 26.

Figura 25 – Montagem do dispositivo de medição das posições X, Y e Z



Fonte: Autor. Para melhor posicionamento das luzes foi utilizado um transferidor com marcação 90°

Figura 26 – Medição das posições X, Y e Z da plataforma móvel

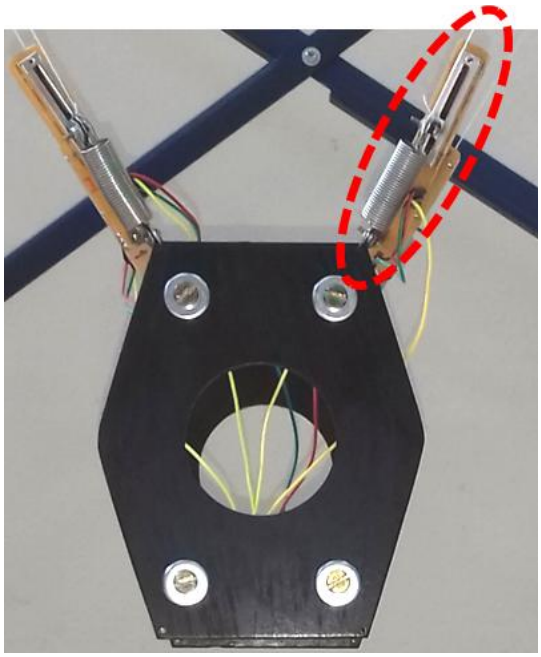


Fonte: Autor. A medição do eixo Z é realizada descontando a altura do feixe de luz até o centro da plataforma, que na montagem é de 0.075 m.

O sensor mola-potenciômetro usado para medir o deslocamento da mola foi conectado na plataforma através de um furo para passagem da extremidade da

mola. Os cabos suportam tensões maiores do que 9.81 N, mas o peso da plataforma é de aproximadamente 8.1226 N e não está aplicado apenas em um dos cabos. O material dos cabos é de 100% Poliéster formado por uma linha utilizada na costura de tecidos de couro, conhecida nas lojas como linha para pesponto. Esta linha é leve e não forma a curva catenária no espaço de trabalho do robô.

Figura 27 – Sensor de medição do deslocamento da mola



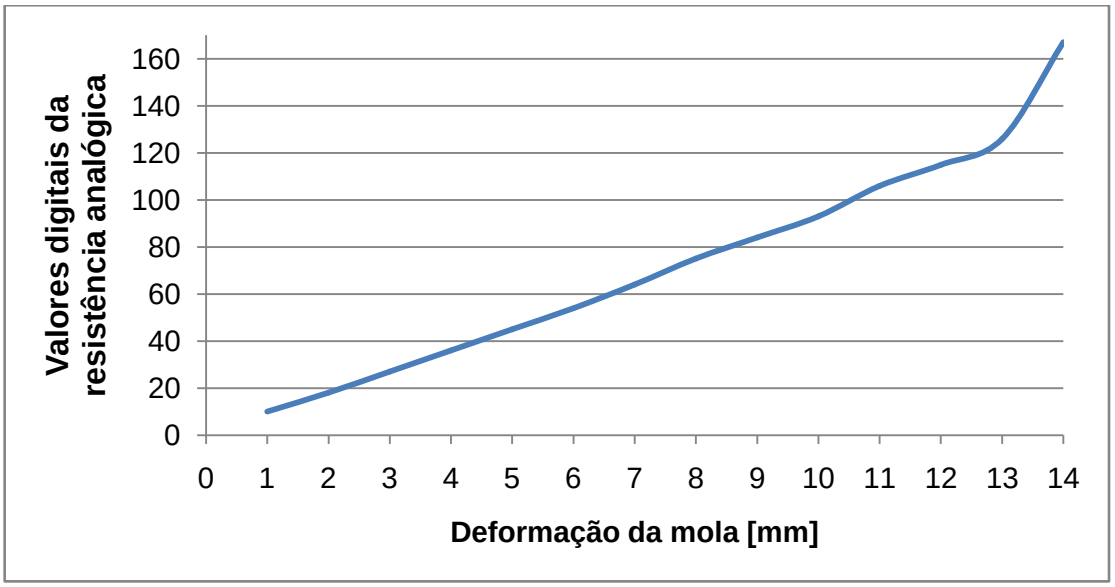
Fonte: Autor.

A conexão entre a mola e o potenciômetro deslizante foi realizada através de um parafuso e porca (figura 23) e a extremidade oposta da mola está presa pelo cabo. O deslocamento do potenciômetro foi medido em função do deslocamento da mola, a cada 1 mm de deformação da mola registrou-se o valor convertido para digital que o potenciômetro enviava na entrada analógica do microcontrolador. O sinal da resistência analógica foi convertido para valores digitais através da linha de código *map(analogRead(A_i),0,1023,0,255)*, onde A_i (i=0 a 3) é o pino referente a porta analógica do potenciômetro. O programa completo está disponível no apêndice 6. A partir de uma deformação da mola de 13 mm o valor da resistência analógica enviado pelo potenciômetro não é mais linear, conforme visto nas duas últimas linhas da tabela 4 e também na curva da figura 28.

Tabela 4 - Deformação da mola e valores digitais do potenciômetro

d [mm]	Valor Digital
1	10
2	18
3	27
4	36
5	45
6	54
7	64
8	75
9	84
10	93
11	106
12	115
13	126
14	167

Figura 28 – Gráfico dos valores do potenciômetro em função da mola



Fonte: Autor.

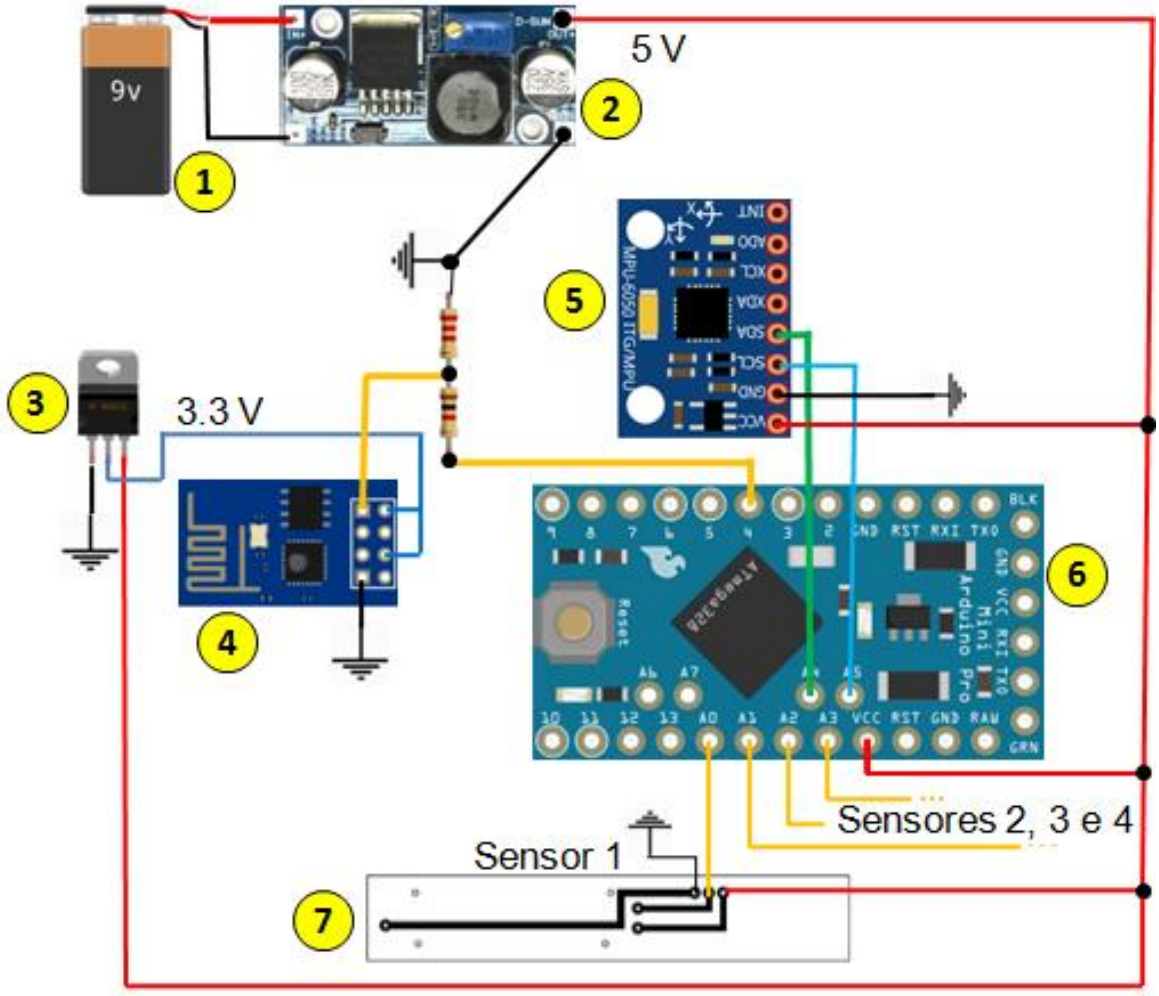
Pela análise da tabela 4 e figura 28, a deformação máxima da mola para o cálculo das forças será de 10 mm, ficando dentro da faixa linear do potenciômetro. Usando um paquímetro foi possível deformar a mola em 0,37 mm para obter uma diferença na resistência do potenciômetro. Abaixo de 0,37 mm o sinal permanece

no seu limiar. Usando a equação 18 o valor máximo da força que o sensor pode medir é de $343 \cdot 10^{-2}$ N.

3.5 MONTAGEM DOS COMPONENTES ELETRÔNICOS

O conjunto mola-potenciômetro foi conectado a um microcontrolador modelo Arduino Pro-Mini que, através de um programa desenvolvido em linguagem C, lê o sinal analógico do potenciômetro e converte para um valor digital. O valor desta conversão é transformado numa deformação da mola e enviado para um módulo WiFi ESP8266, configurado como cliente, conforme figura 29.

Figura 29 – Montagem dos componentes eletrônicos na plataforma móvel



Fonte: Autor. Para simplificar o esquema foi representado apenas a placa de circuito impresso do sensor mola-potenciômetro, sendo os demais substituídos pelo texto “Sensores 2, 3 e 4”.

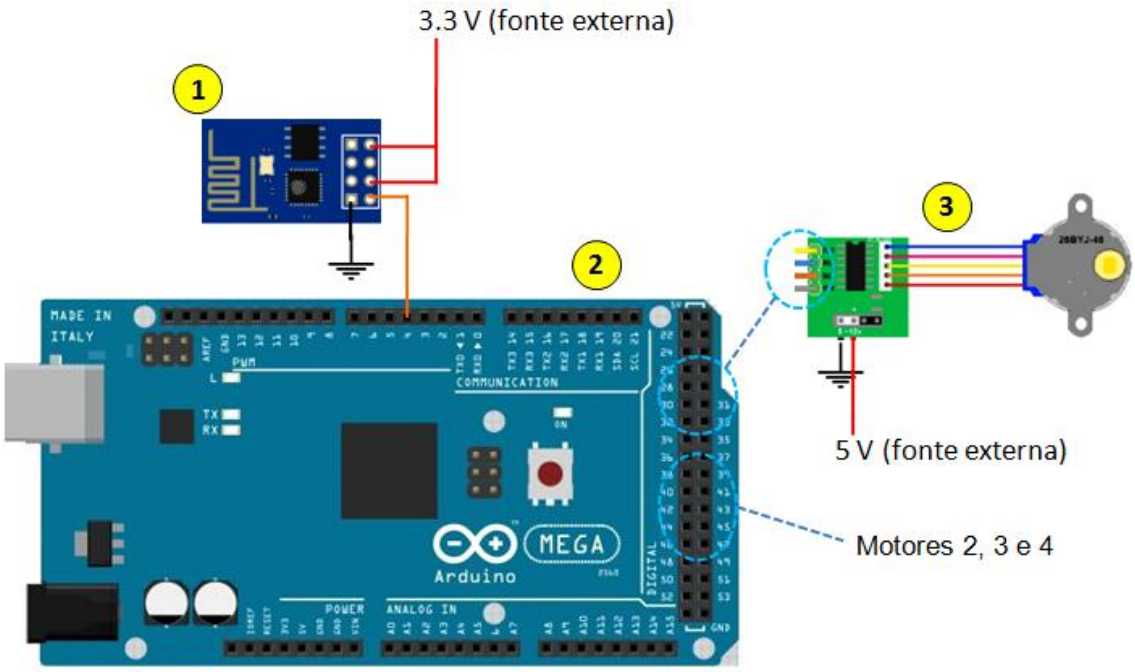
Seguindo a ordem da numeração na figura 29 os componentes são os seguintes:

1) Bateria de alimentação de 9V
2) Fonte chaveada para regular a tensão em 5V para alimentação do microcontrolador, módulo giroscópico, potenciômetros e regulador de tensão de 3.3V. Esta fonte é composta pelo CI LM2596 (INSTRUMENTS, 2017) e suporta uma corrente de até 3A. Segundo o fabricante, a velocidade de comutação é de 150KHz, ver anexo 1.
3) Regulador de tensão composto pelo CI LM1117 (INSTRUMENTS, 2017) para alimentação do módulo WiFi ESP8266 em 3.3V e com corrente máxima de operação de 800 mA. O anexo 2 contém mais detalhes sobre este componente.
4) Módulo WiFi ESP8266 (ESPRESSIF, 2017) responsável pelo envio dos dados pela rede sem fio. Opera em 3.3 V e consome até 30 mA de corrente.
5) Módulo giroscópico com CI MPU6050, este sensor contém num único CI um acelerômetro e um giroscópio, no total apresenta 6 graus de liberdade e consome uma corrente máxima de 3.9 mA. Este componente é usado na montagem para fornecer os ângulos de rotação em torno dos eixos da plataforma móvel, outras informações estão no anexo 4.
6) Microcontrolador Arduino modelo Pro Mini, usado para receber, processar e enviar os dados para o módulo ESP8266 através de um programa desenvolvido em linguagem C e disponível no apêndice 5. É baseado no processador ATMega328p AVR de 16Mhz e opera em 5v (ATMEL, 2016). Possui 14 saídas (ou entradas) digitais e 8 entradas analógicas.
7) Sensor formado por um potenciômetro deslizante de 10 KΩ e uma mola cujo fator K foi determinado em laboratório, ver figuras 23 e 27.

Para a comunicação entre o microcontrolador e o módulo ESP8266 foi preciso usar um divisor de tensão no pino de recepção do módulo devido a este operar com tensão de 3.3 V e o microcontrolador em 5 V. É possível conectar o pino digital 4 do microcontrolador ao pino de recepção 10 do microcontrolador que está localizado distante da plataforma, no caso da retirada do módulo ESP8266.

A montagem dos componentes ligados ao microcontrolador que movimenta os motores contém um módulo ESP8266 ligado a um microcontrolador Arduino e os motores de passo, a figura 30 ilustra esta montagem.

Figura 30 – Esquema eletrônico dos componentes de recepção dos dados



Fonte: Autor. Nesta figura o módulo ESP8266 pode ser retirado se necessário e o pino 10 pode ser conectado com o pino digital 4 do Arduino Pro Mini, responsável pelo envio dos dados dos sensores.

A figura 30 é composta pelos seguintes componentes:

1) Módulo ESP8266
2) Microcontrolador Arduino modelo MEGA 2560 (ATMEL, 2017) com 54 portas digitais, 16 portas analógicas, velocidade de clock de 16 MHz, tensão de operação de 5 V e corrente em cada pino de 40 mA. Este componente é responsável por movimentar os motores, receber os valores das forças nos cabos e enviar os dados para o computador via porta serial.
3) Motor de passo modelo 28BYJ-48 juntamente com o *driver* ULN2003 (INSTRUMENTS, 2015). É um motor unipolar que opera em 5 V e possui uma redução de 1/64, chegando a 4096 passos por volta. O *driver* ULN2006 opera com correntes de até 500 mA e tensões entre 5 e 12 V. O anexo 5 possui mais informações sobre este motor e o anexo 6 sobre o *driver*.

4 RESULTADOS

Através das equações lineares é possível obter em função da localização da plataforma os valores das forças nos cabos. As tabelas 5 e 6 possuem alguns valores fornecidos como entrada nos programas. Na tabela 5 as forças são determinadas pelo sistema linear da equação 25 e na tabela 6 as forças foram dadas como entrada e a posição da plataforma foi medida através do sensor giroscópico e do apontador *laser* montado na estrutura real.

Tabela 5 - Forças teóricas nos cabos dadas a posição da plataforma

Valores de Entrada						**Forças teóricas [x10^{-2} N]**			
P_X	**P_Y**	**P_Z**	**Θ°**	**φ°**	**Ψ°**	**F1**	**F2**	**F3**	**F4**
-0.008	0.005	0.225	3	3	0	317	264	284	315
0.000	0.000	0.300	8	5	0	249	249	249	249
-0.010	0.210	0.160	30	0	0	194	341	244	325

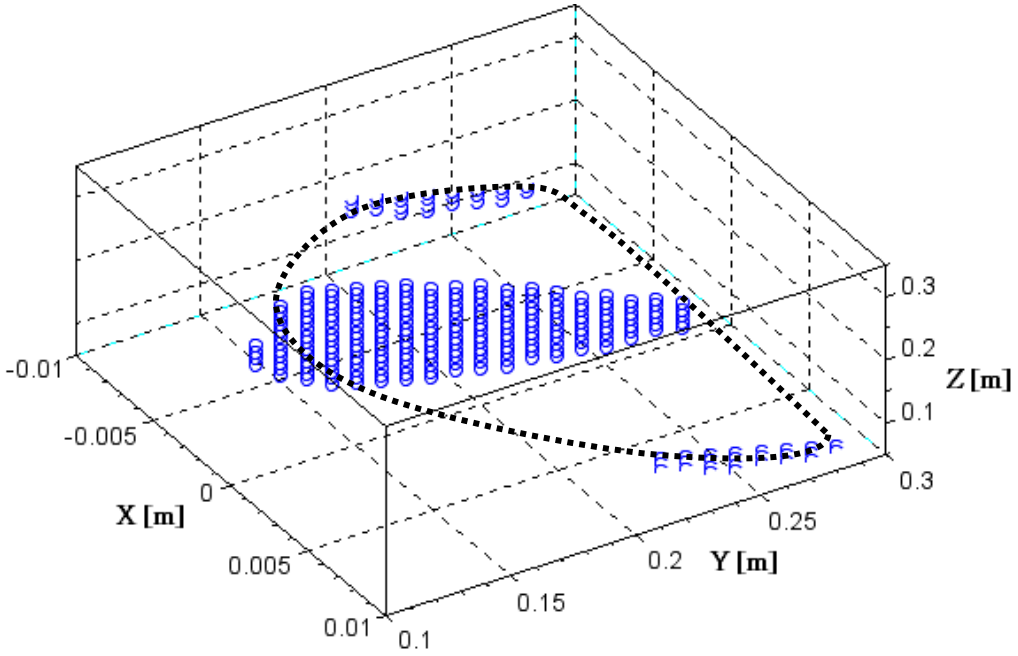
Tabela 6 - Posições medidas na prática em função das forças calculadas

Forças [x10^{-2} N]				**Localização da plataforma**					
F1	**F2**	**F3**	**F4**	**P_X (desvio)**	**P_Y (desvio)**	**P_Z (desvio)**	**Θ° (desvio)**	**φ° (desvio)**	**Ψ° (desvio)**
317	264	284	315	-0.006 (+0.002)	0.007 (+0.002)	0.229 (+0.004)	3 (0)	2 (+1)	0 (0)
249	249	249	249	0.001 (+0.001)	0.000 (0.000)	0.300 (0.000)	7 (-1)	4 (-1)	0 (0)
194	341	244	325	-0.008 (+0.002)	0.207 (-0.003)	0.153 (-0.004)	28 (-2)	0 (0)	0 (0)

Através do programa desenvolvido no software SciLab gera-se alguns espaços de trabalhos do robô, a figura 31 mostra o espaço de trabalho do robô para um ângulo de rolagem (*roll*) Θ = 30° e φ = Ψ = 0°. Foram dados incrementos de 0.010 m para os eixos X, Y e Z da estrutura. No eixo X a variação foi de -0.300 m até 0.300 m, no eixo Y de -0.280 m até 0.280 m e em Z de 0.000 até 0.280 m. Limitou-se

também o espaço de trabalho para que as forças nos cabos permanecessem positivas e abaixo de $343 \cdot 10^{-2}$ N.

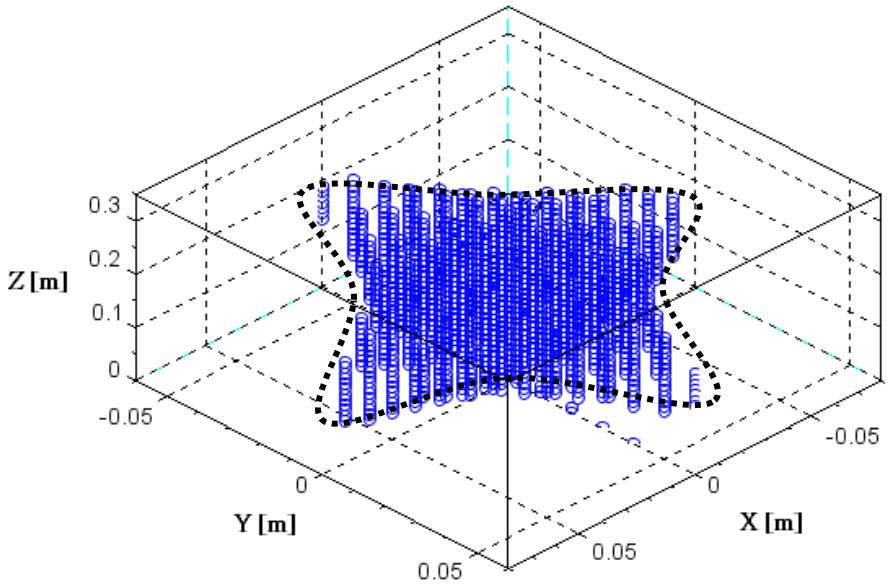
Figura 31 – Vista Isométrica do espaço de trabalho para Θ = 30°



Fonte: Autor.

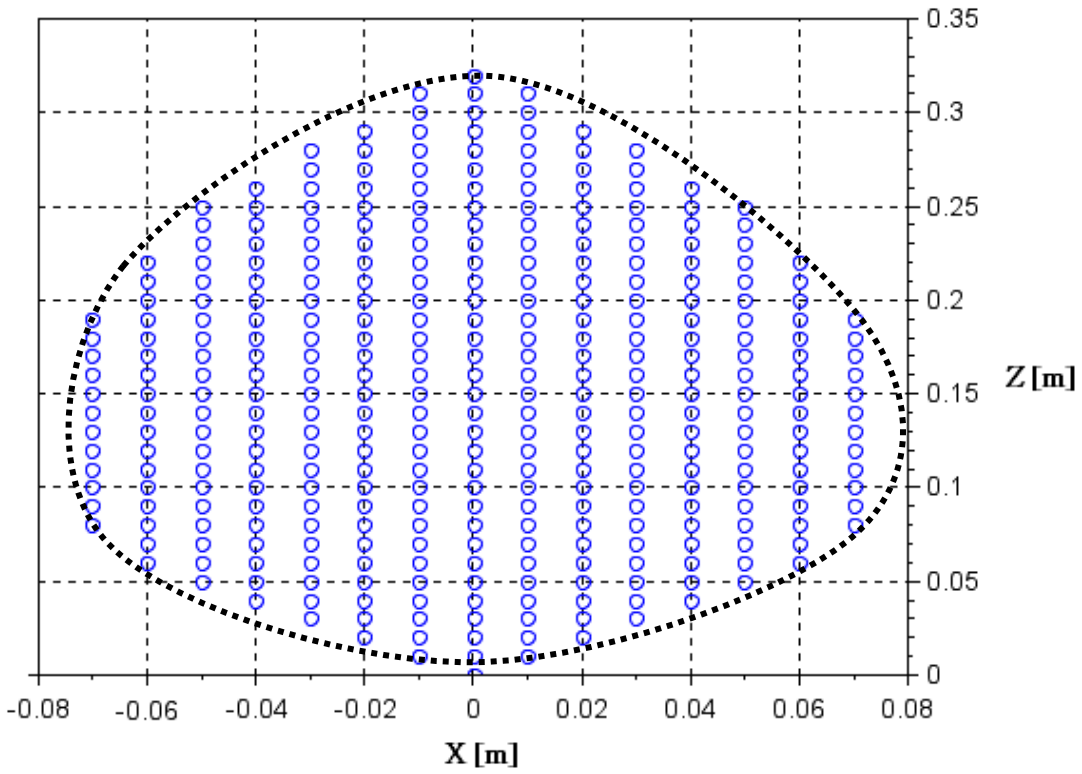
Também foi gerado o espaço de trabalho do robô considerando os ângulos iguais a zero, mantendo a plataforma sem rotação, conforme figuras 32, 33, 34 e 35.

Figura 32 – Vista isométrica do espaço de trabalho sem rotação da plataforma



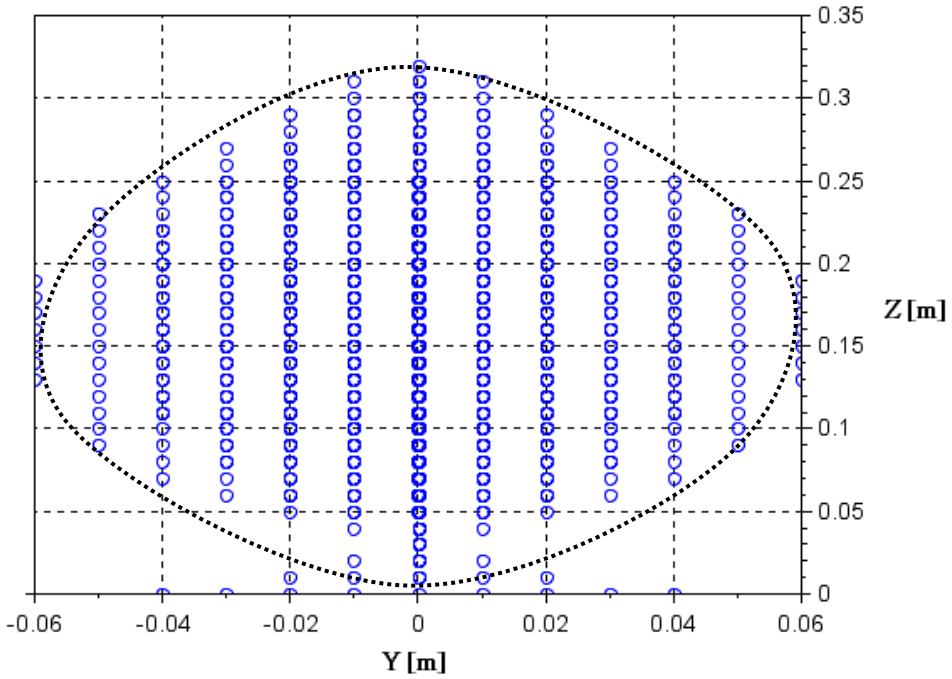
Fonte: Autor.

Figura 33 - Vista XZ do espaço de trabalho sem rotação da plataforma



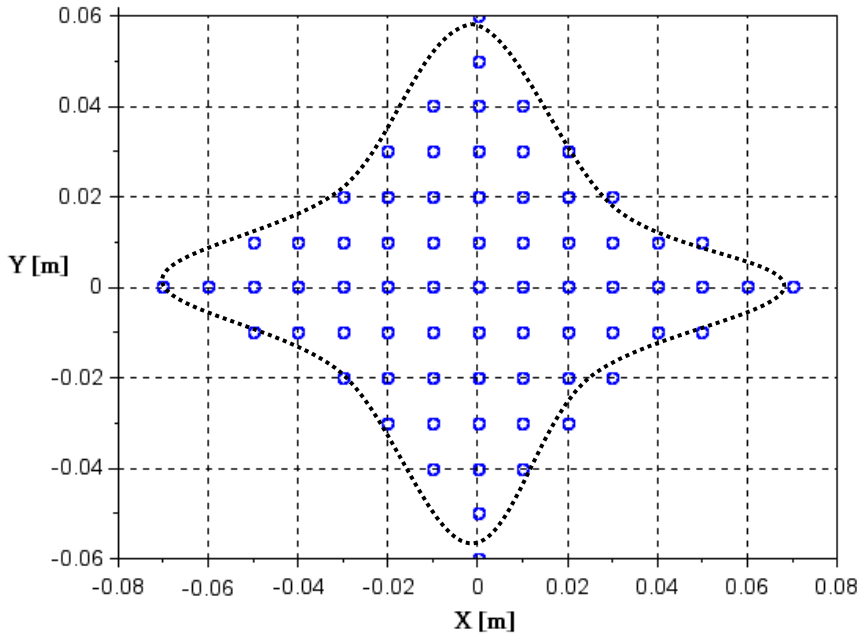
Fonte: Autor.

Figura 34 - Vista YZ do espaço de trabalho sem rotação da plataforma



Fonte: Autor.

Figura 35 - Vista XY do espaço de trabalho sem rotação da plataforma

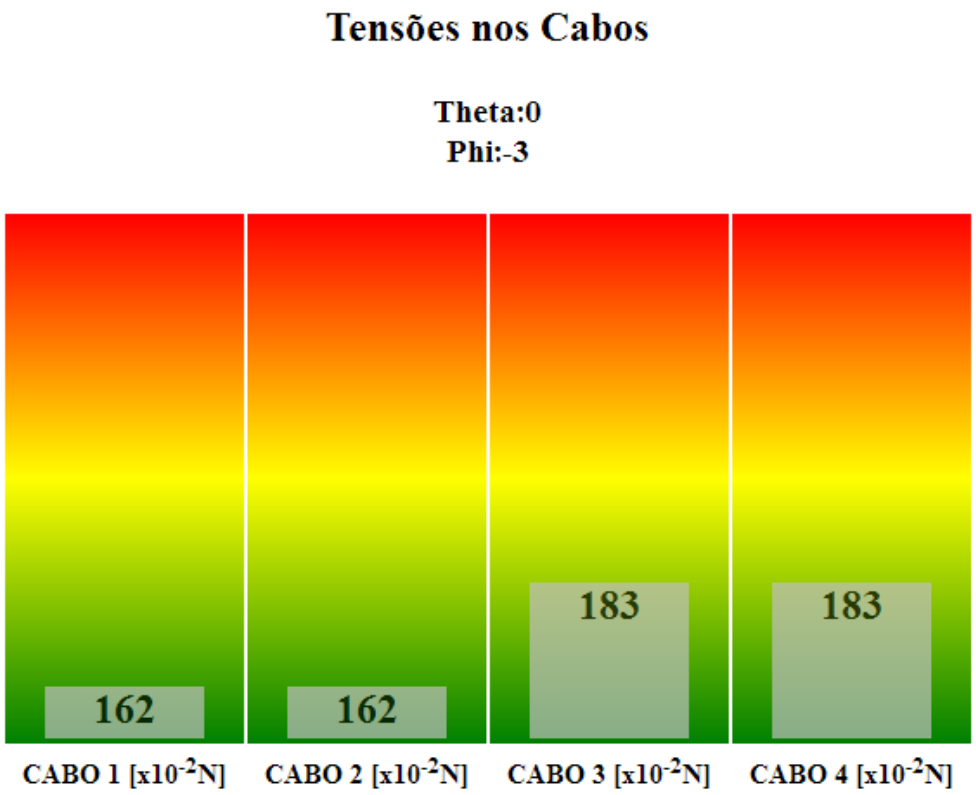


Fonte: Autor.

O valor analógico de cada potenciômetro é lido e enviado pelo microcontrolador Arduino Pro Mini através do módulo ESP8266, configurado como cliente, até outro módulo ESP 8266 que está configurado como servidor e conectado ao microcontrolador Arduino MEGA, enviando a informação para o software SciLab .

Através do programa criado no software SciLab é possível gerar o gráfico das forças em tempo real durante o movimento da plataforma, na figura 36 pode-se ver um exemplo do gráfico de forças gerado durante o posicionamento do robô. O gráfico pode ser visualizado num computador ou em um celular conectado diretamente a rede Wifi gerada pelo módulo ESP8266 configurado como cliente.

Figura 36 – Gráfico de forças gerado durante a movimentação do robô



Fonte: Autor.

5 DISCUSSÃO

A tabela 6 apresentou alguns desvios da posição da plataforma em relação aos cálculos teóricos, esses desvios aparecem por alguns motivos:

a) As forças na prática são calculadas pela deformação da mola e deslocamento do potenciômetro, ambos possuem resistência ao avanço: o potenciômetro por possuir peças deslizantes em contato e a mola por não estar totalmente alinhada na direção (B_i)(A_i) devido a atrito com o orifício em que está presa.
b) O sinal do potenciômetro sofre alteração somente a cada 0.37 mm de deformação da mola, fazendo com que o intervalo das forças calculadas não coincidam com as forças fornecidas, ou seja, os sensores medirão apenas forças com a mola deformada 0.00037 m, 0.00074 m, 0.00111 ... 0.00999 m.

Analisando os gráficos do espaço de trabalho percebe-se que o espaço alcançável pela plataforma diminui com o aumento do ângulo da plataforma. Já no gráfico em que não houve inclinação da plataforma obteve-se um espaço de trabalho maior e quase simétrico.

As equações lineares formadas a partir da teoria de helicóides serviram como base para o desenvolvimentos de programas que determinam as forças em cada cabo do robô, bem como os resultados da cinemática e estática.

Na parte estrutural alguns problemas como posicionamento dos parafusos na estrutura prejudicaram a manutenção do robô. Percebe-se na figura 22 que para montar o motor foi necessário algum tempo para segurar a porca na chapa, uma vez que o furo da porca ficou entre a roldana e a chapa, sendo impossível de segurar a porca somente com os dedos.

Na figura 26 a montagem dos apontadores *laser* mostra que ao mover a plataforma a linha (cabo) que prende as luzes irá balançar e girar, na prática foi preciso usar suportes para evitar que as luzes ficassem girando e realizar a medição corretamente.

Quando a plataforma não sofre inclinação nos seus eixos a tensão estática nos cabos é única para cada posição X, Y e Z, isso significa que é possível determinar a localização da plataforma apenas conhecendo as forças que atuam nos

cabos quando ela permanece estática, essa foma de trabalho resolveu o problema da medição da posição da plataforma com as luzes e foi confirmada pelas equações não lineares resolvidas com o software SciLab.

A construção do robô levou mais tempo do que o esperado porque os componentes eletrõnicos foram importados, levando entre 40 e 60 dias para chegarem.

Para a resolução das equações não lineares não utilizou-se o software WxMaxima porque o mesmo não resolvia o problema. Ao executar as mesmas funções no software SciLab o resultado foi calculado corretamente.

Este robô foi o primeiro robô paralelo guiado por cabos e construído no Centro de Ciências Tecnológicas da UDESC em Joinville – SC. Espera-se que outros trabalhos científicos sejam desenvolvidos a partir deste robô.

6 CONCLUSÕES

O projeto, análise e construção de um robô paralelo guiado por cabos exigem conhecimentos em geometria, programação, eletrônica e aplicações de teorias matemáticas que podem ser utilizadas nesta área.

Trabalhos relacionados foram desenvolvidos usando a teoria das helicóides como solução para a cinemática e estática de robôs paralelos. Conferências são realizadas anualmente para apresentar à comunidade científica os progressos nesta área, no entanto ainda são recentes porque somente no ano de 2014 foi realizada a segunda conferência de robôs paralelos guiados por cabos, sendo que o primeiro robô paralelo rígido foi idealizado em 1965 por Stewart.

A medição da força usando um sensor composto de mola e potenciômetro obteve boa precisão para o espaço de trabalho proposto.

Ao fornecer uma posição e uma orientação para o robô as equações lineares resultavam nas quatro forças aplicadas nos cabos. Ao fornecer as quatro forças as equações não lineares eram resolvidas pelo método de Newton-Raphson chegando nos valores de localização da plataforma.

Na prática de medição das forças nos cabos através de entradas de valores X, Y e Z do centro da plataforma móvel observou-se que as tensões em cada cabo não se repetem de uma localização para outra quando os ângulos θ, φ e ψ são nulos.

Robôs paralelos guiados por cabos podem ser uma solução para o desenvolvimento de simuladores em ambientes abertos, como campos de futebol e áreas ao ar livre.

REFERÊNCIAS

ANGELES, J.; HOMMEL, G.; KOVÁCS, P. **On the representation of rigid-body motion and its application to generalized platform manipulators**. Dordrecht: Springer, 1993.

ATMEL, Microchip. **ATmega328/P - Datasheets - Complete**. Disponível em: <www.atmel.com> Acesso em: 5 jun. 2016.

ATMEL, Microchip. **ATmega640/1280/1281/2560/2561 - Complete**. Disponível em: <www.atmel.com> Acesso em: 18 fev. 2017.

AUGUSTO, M.; DILDA, V.; LERMEN, R. **Desenvolvimento de um robô paralelo tipo delta controlado com arduino**. 2015. Monografia (Bacharelado em Engenharia Mecânica) – Faculdade Horizontina, Horizontina, 2015.

BALADEZ, F. O passado, o presente e o futuro dos simuladores. **Fasci-tech**, São Caetano do Sul, dez. 2009. Disponível em: <http://fatecsaocaetano.edu.br/fascitech/index.php/fascitech/article/view/4/4> Acesso em: 15 mar. 2017.

BALL, R. S. **A Treatise on the Theory of Screws**. [S.l.]: Archive. Disponível em: <https://archive.org/details/theoryscrews00ballrich> Acesso em: 10 maio 2016.

BONEV, I. **The true origins of parallel robots**. ParalleMIC, 2003. Disponível em: <www.parallemic.org/Reviews/Review007.html> Acesso em: 15 abr. 2016.

CAMPOS, A. **Cinemática diferencial de manipuladores empregando cadeias virtuais**. 2004. Tese (Doutorado em Engenharia Mecânica) – Universidade Federal de Santa Catarina, Florianópolis, 2004.

CARBONI, A. P.; SIMAS, H.; MARTINS, D. Modelagem por Helicóides de Restrições Redundantes. In: XXXI CONGRESSO DE MECÂNICA COMPUTACIONAL, 2012, Argentina. **Anais eletrônicos**. Argentina: Salta, 2012. Disponível em: < www.amcaonline.org.ar/ ojs/index.php/mc/article/viewFile/4222/4148 > Acesso em: 26 jun. 2016.

CONTE, S. D., DE-BOOR, C. **Elementary Numerical Analysis, an Algorithmic Approach**. São Paulo: McGraw-Hill Book Company, 1980.

CRAIG, John J. **Introduction to robotics: mechanics and control**. 3. ed. atual. Upper Saddle River: Pearson Prentice Hall, 2005.

DUAN, X., QIU, Y., DUAN, Q. and DU, J. Calibration and Motion Control of a Cable-driven Parallel Manipulator Based Triple-level Spatial Positioner. **Advances in Mechanical Engineering**, v. 2014, n. 10, 2015. Disponível em: < http://journals.sagepub.com/doi/10.1155/2014/368018 > Acesso em: 26 mai. 2017.

ENTERPRISES, S. **Scilab:** Free and Open Source Software for Numerical Computation, Version 6.0.0. Scilab Enterprises, Versailles, France, 2017. Disponível em: <www. scilab. org> Acesso em: 20 jul. 2017.

ESPRESSIF. **ESP8266EX Datasheet**. Disponível em: <espressif.com/en/ support/ download/documents> Acesso em: 15 set. 2017.

GALLARDO, J.; RICO, J.; FRISOLI, A.; CHECCACCI, D.; BERGAMASCO, M. Dynamics of parallel manipulators by means of screw theory. **Mechanism and Machine Theory**, v. 38, 1113–1131, 2003. Disponível em: < http://percro.sssup.it/ ~antony/papers/mech_mach_th_2003.pdf > Acesso em: 29 out. 2016.

GONZALEZ-RODRÍGUEZ, A. et al. A Novel Design to Improve Pose Accuracy for Cable Robots. In: PROCEEDINGS OF THE 14HT IFTOMM WORLD CONGRESS. **Anais eletrônicos**. Taiwan, 2015. Disponível em: < www.iftomm2015.tw/ IFToMM2015CD/PDF/OS2-018.pdf > Acesso em: 10 mai. 2017.

GROOVER, M. P.; WEISS, M.; NAGEL, R. N.; ODREY, N. G. **Industrial Robotics: Technology, Programming, and Applications**. New Delhi: TATA McGraw-Hill, 1986.

HUNT, K. H. **Kinematic Geometry of Mechanisms**. Oxford: Clarendon Press, 1978.

HUNT, K. H. Don’t cross-thread the screw. In: PROCEEDINGS OF BALL 2000 CONFERENCE, 2000, Indiana. **Anais eletrônicos**. Disponível em: < http://onlinelibrary.wiley.com/doi/10.1002/rob.10095/full > Acesso em: 12 jan. 2017.

INSTRUMENTS, Texas. **LM2596 SIMPLE SWITCHER® 4.5V to 40V, 3A Low Component Count Step-Down Regulator**. Disponível em: <www.ti.com/product/LM2596/datasheet> Acesso em: 15 set. 2017.

INSTRUMENTS, Texas. **LM1117 Space Saving 800mA Low-Dropout Linear Regulator with Internal Current Limit**. Disponível em: <www.ti.com/ product/LM1117> Acesso em: 15 set. 2017.

INSTRUMENTS, Texas. **ULN2003B High-voltage High-current Darlington Transistor Array**. Disponível em: <www.ti.com/product/LM1117>, Acesso em: 26 abr. 2015.

INVENSENSE. **MPU-6050 Six-Axis (Gyro + Accelerometer) MEMS MotionTracking™ Devices**. Disponível em: <www.invensense.com/ products/motion-tracking/6-axis/mpu-6050> Acesso em: 15 set. 2017.

KIATRONICS. **28BYJ-48 Stepper Motor 5VDC - Code: 70289 - SPECIAL!** Disponível em: <www.kiatronics.com/28byj-48-stepper-motor-5vdc-code-70289.html> Acesso em: 12 out. 2016.

LEAL, Rafael Della Giustina et al. **Impactos sociais e econômicos da robotização: Estudo de caso do projeto Roboturb**. Dissertação (Mestrado em Engenharia Elétrica) - Universidade Federal de Santa Catarina, Florianópolis, 2005.

MACH. **ABB IRB 4600**.Disponível em: <www.mach.ro/abb-irb-4600-en.html> Acesso em 27 ago. 2017.

MERLET, J. P. **Singular configurations of parallel manipulators and Grassman geometry**. France: Springer, 1989.

MIERMEISTER, P.; et al. The CableRobot Simulator Large Scale Motion Platform Based on Cable Robot Technology. In: INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS, 2016. **Anais eletrônicos**. Korea: Daejeon, 2016. Disponível em: < www.researchgate.net/publication/ 312288481_The_CableRobot_simulator_large_scale_motion_platform_based_on_cable_robot_technology > Acesso em: 14 mar. 2017.

MORAN, M. E. The da Vinci robot. **Journal of endourology**, New York, v. 20, n. 12, p. 986-990, 2006.

MOURAIN B. The 40 generic positions of a parallel robot. In: INTERNATIONAL SYMPOSIUM ON SYBOLIC AND ALGEBRAIC COMPUTATION, 1993. **Anais eletrônicos**. USA: New York, 1993. Disponível em: < https://dl.acm.org/ citation.cfm?id=164120> Acesso em: 25 fev. 2017.

MURARO, Thaís et al. **Análise cinemática e estática de um mecanismo espacial atuado por cabos aplicado à movimentação de pacientes**. 2015. Dissertação (Mestrado em Engenharia Mecânica) - Universidade Federal de Santa Catarina, 2015.

POTT, A.; BRUCKMANN, T. **Cable-Driven Parallel Robots: Proceedings of the Second International Conference on Cable-Driven Parallel Robots**. Switzerland: Springer, 2014.

PRIOR S. D.; WARNER P. R. A Review of World Rehabilitation Robotics Research. In: PROCEEDINGS OF THE IEE COLLOQUIUM ON HIGH-TECH HELP FOR THE HANDICAPPED, 1990. **Anais eletrônicos**. UK: London, 1990. Disponível em: < http://ieeexplore.ieee.org/document/189943/> Acesso em: 23 ago. 2017

PUSEY, J.; FATTAH, A.; AGRAWAL, S.. Design and workspace analysis of a 6–6 cable-suspended parallel robot. In: INTERNATIONAL CONFERENCE ON INTELLIGENT ROBOTS AND SYSTEMS, 2003. **Anais eletrônicos**. Nevada: Las Vegas, 2003. Disponível em: < http://ieeexplore.ieee.org/document/1249179/> Acesso em: 18 mar. 2017.

SHARMA, Karan et al. Evaluation of Human Safety in the DLR Robotic Motion Simulator using a Crash Test Dummy. In: INTERNATIONAL CONFERENCE ON ROBOTICS AND AUTOMATION, 2013. **Anais eletrônicos**. Germany: Karlsruhe,

2013. Disponível em: < http://ieeexplore.ieee.org/abstract/document/6630577/> Acesso em: 30 jun. 2016.

SIMAS, H.; MARTINS D.; GUENTHER R. Cinemática Inversa de Robôs via Helicóides. In: CONGRESSO NACIONAL DE ENGENHARIA MECÂNICA, 2002. **Anais eletrônicos**. Paraíba: João Pessoa, 2002.

SOUZA, E. A.; **Métodos Iterativos para Problemas Não Lineares**. 2015. Dissertação (Mestrado em Modelagem Computacional em Ciência e Tecnologia) - Universidade Federal Fluminense, Volta Redonda, 2015.

STEWART, Doug. A platform with six degrees of freedom. In: PROCEEDINGS OF THE INSTITUTION OF MECHANICAL ENGINEERS, 1965. **Anais eletrônicos**. Disponível em: < http://journals.sagepub.com/doi/abs/10.1243/PIME_PROC_1965_180_029_02> Acesso em: 10 out. 2016.

TANG, Xiaoqiang. **An overview of the development for cable-driven parallel manipulator**. Advances in Mechanical Engineering, v. 6, p. 823028, 2014.

TARTARI FILHO, Sylvio Celso. **Modelagem e otimização de um robô de arquitetura paralela para aplicações industriais**. 2006. Tese (Doutorado) - Universidade de São Paulo, São Paulo, 2006..

TSAI, L.-W. **Robot Analysis: the Mechanics of serial and parallel manipulators**. New York: John Wiley & Sons, 1999.

VALDIERO, A. C.; CAMPOS, A.; GUENTHER, R.; MATRINS, D. **Cálculo e Análise do Jacobiano de um manipulador paralelo de 3 graus de liberdade baseado na teoria dos helicóides**. In: IX CONGRESSO BRASILEIRO DE ENGENHARIA MECÂNICA, p. 10, 2001.

VODOPIVEC, A. **wxMaxima 17.05.1**. Disponível em: <andrejv.github.io/wxmaxima/download.html> Acesso em: 20 mar. 2017.

WOLFGANG, B.; GARY, D.; WESTFALL, H. D. **Física para Universitários - Mecânica**, Brasil: McGraw Hill, 2012.

APÊNDICE 1 - MATRIZ DA HELIFORÇA PARA O CABO 1

$$
\$'_1 = \begin{bmatrix}
\dfrac{(\,{}^{o}A_{1X} - {}^{o}B_{1X})}{\sqrt{(\,{}^{o}A_{1X} - {}^{o}B_{1X})^2 + (\,{}^{o}A_{1Y} - {}^{o}B_{1Y})^2 + (\,{}^{o}A_{1Z} - {}^{o}B_{1Z})^2}} & \dfrac{{}^{o}B_{1Z}\,(\,{}^{o}A_{1Y} - {}^{o}B_{1Y})}{\sqrt{(\,{}^{o}A_{1X} - {}^{o}B_{1X})^2 + (\,{}^{o}A_{1Y} - {}^{o}B_{1Y})^2 + (\,{}^{o}A_{1Z} - {}^{o}B_{1Z})^2}} - \dfrac{{}^{o}B_{1Y} * (\,{}^{o}A_{1Z} - {}^{o}B_{1Z})}{\sqrt{(\,{}^{o}A_{1X} - {}^{o}B_{1X})^2 + (\,{}^{o}A_{1Y} - {}^{o}B_{1Y})^2 + (\,{}^{o}A_{1Z} - {}^{o}B_{1Z})^2}} \\
\dfrac{(\,{}^{o}A_{1Y} - {}^{o}B_{1Y})}{\sqrt{(\,{}^{o}A_{1X} - {}^{o}B_{1X})^2 + (\,{}^{o}A_{1Y} - {}^{o}B_{1Y})^2 + (\,{}^{o}A_{1Z} - {}^{o}B_{1Z})^2}} & \dfrac{{}^{o}B_{1X}\,(\,{}^{o}A_{1Z} - {}^{o}B_{1Z})}{\sqrt{(\,{}^{o}A_{1X} - {}^{o}B_{1X})^2 + (\,{}^{o}A_{1Y} - {}^{o}B_{1Y})^2 + (\,{}^{o}A_{1Z} - {}^{o}B_{1Z})^2}} - \dfrac{{}^{o}B_{1Z} * (\,{}^{o}A_{1X} - {}^{o}B_{1X})}{\sqrt{(\,{}^{o}A_{1X} - {}^{o}B_{1X})^2 + (\,{}^{o}A_{1Y} - {}^{o}B_{1Y})^2 + (\,{}^{o}A_{1Z} - {}^{o}B_{1Z})^2}} \\
\dfrac{(\,{}^{o}A_{1Z} - {}^{o}B_{1Z})}{\sqrt{(\,{}^{o}A_{1X} - {}^{o}B_{1X})^2 + (\,{}^{o}A_{1Y} - {}^{o}B_{1Y})^2 + (\,{}^{o}A_{1Z} - {}^{o}B_{1Z})^2}} & \dfrac{{}^{o}B_{1Y}\,(\,{}^{o}A_{1X} - {}^{o}B_{1X})}{\sqrt{(\,{}^{o}A_{1X} - {}^{o}B_{1X})^2 + (\,{}^{o}A_{1Y} - {}^{o}B_{1Y})^2 + (\,{}^{o}A_{1Z} - {}^{o}B_{1Z})^2}} - \dfrac{{}^{o}B_{1X} * (\,{}^{o}A_{1Y} - {}^{o}B_{1Y})}{\sqrt{(\,{}^{o}A_{1X} - {}^{o}B_{1X})^2 + (\,{}^{o}A_{1Y} - {}^{o}B_{1Y})^2 + (\,{}^{o}A_{1Z} - {}^{o}B_{1Z})^2}}
\end{bmatrix} * F1
$$

APÊNDICE 2 - PROGRAMA DESENVOLVIDO NO WXMAXIMA PARA CÁLCULO DOS COMPRIMENTOS DOS CABOS DO ROBÔ

```
/*
ESTE PROGRAMA ESTÁ DISPONIBILIZADO NO LINK DO GOOGLE DRIVE ABAIXO:

https://drive.google.com/open?id=0B9jot9RnR3XiSkFBNFZEWlBWT2s

Projeto, Simulação e Análise da Localização, Orientação e Forças nos Cabos de um Robô
Paralelo Guiado por Cabos – Joinville, 2017.
Desenvolvimento de um sistema linear via teoria das helicóides para determinação das
quatro forças que atuam nos cabos do robô proposto.
-------------------------------------------------------------------------------
  ENTRADA DOS VALORES PELO USUÁRIO:
  O ponto P é a origem da plataforma localizado no centro do sistema móvel {P}
  Os ângulos são em torno dos eixos x, y e z do sistema {P}
-------------------------------------------------------------------------------
*/
P:matrix([0],[0],[0])$
theta: 0 * %pi/180;
phi:   0 * %pi/180;
psi:   0 * %pi/180;
peso:0.828*9.81$
 /* Peso da plataforma é de 828 gr.*/
/* Valores das coordenadas dos pontos "Ai" fixo no sistema {O} */
Ao1:matrix([-0.3820],[-0.3820],[0.7500])$
Ao2:matrix([-0.3820],[0.3820],[0.7500])$
Ao3:matrix([0.3820],[0.3820],[0.7500])$
Ao4:matrix([0.3820],[-0.3820],[0.7500])$
/* Coordenadas dos pontos "Bi" fixos na plataforma */
Bp1:matrix([-0.0395],[-0.0520],[0.0750])$
Bp2:matrix([-0.0395],[0.0520],[0.0750])$
Bp3:matrix([0.0395],[0.0520],[0.0750])$
Bp4:matrix([0.0395],[-0.0520],[0.0750])$
/*  Cossenos diretores em torno dos eixos do sistema móvel {P}  */;
Rxo:matrix([1,0,0],[0,cos(theta),-sin(theta)],[0,sin(theta),cos(theta)]);
```

```
Ryo:matrix([cos(phi),0,sin(phi)],[0,1,0],[-sin(phi),0,cos(phi)]);
Rzo:matrix([cos(psi),-sin(psi),0],[sin(psi),cos(psi),0],[0,0,1]);
/* Matriz de rotação de Euler considerando a sequência XYZ  */
EulerXYZ:Rxo.Ryo.Rzo;
/* Inclusão do valor 1 para o cálculo da matriz de transformação homogênea  */
Ao1_1:addrow(Ao1,[1])$
Ao2_1:addrow(Ao2,[1])$
Ao3_1:addrow(Ao3,[1])$
Ao4_1:addrow(Ao4,[1])$
Bp1_1:addrow(Bp1,[1])$
Bp2_1:addrow(Bp2,[1])$
Bp3_1:addrow(Bp3,[1])$
Bp4_1:addrow(Bp4,[1])$
P_1:addrow(P,[1])$
/* Inclusão do valor 0 para montar a matriz de transformação homogênea */;
EulerXYZ_0:addrow(EulerXYZ,[0,0,0])$
/* Montagem da matriz de transformação homogênea  */
TH:addcol(EulerXYZ_0,P_1);
/* CÁLCULO DOS QUATRO VETORES "BA" */
/*  Bo1_1 é o vetor "B" da referência móvel {P} transportado para a referência fixa {O}  */;
Bo1_1:TH.Bp1_1$
B1A1:Ao1_1-Bo1_1;
Bo2_1:TH.Bp2_1$
B2A2:Ao2_1-Bo2_1;
Bo3_1:TH.Bp3_1$
B3A3:Ao3_1-Bo3_1;
Bo4_1:TH.Bp4_1$
B4A4:Ao4_1-Bo4_1;
/*  O COMPRIMENTO L DE CADA CABO É A NORMA DO VETOR "BA"  */;
L1:sqrt(B1A1[1,1]^2+B1A1[2,1]^2+B1A1[3,1]^2);
L2:sqrt(B2A2[1,1]^2+B2A2[2,1]^2+B2A2[3,1]^2);
L3:sqrt(B3A3[1,1]^2+B3A3[2,1]^2+B3A3[3,1]^2);
L4:sqrt(B4A4[1,1]^2+B4A4[2,1]^2+B4A4[3,1]^2);
/*  Heligiro do peso  Sp */;
Sp:matrix([0],[0],[-peso],[-P[2,1]*peso],[P[1,1]*peso],[0]);
/* Retirando o valor 1 para usar o vetor 'Bo' como 'S0'  */
```

```
Bo1:matrix([Bo1_1[1,1]],[Bo1_1[2,1]],[Bo1_1[3,1]])$
Bo2:matrix([Bo2_1[1,1]],[Bo2_1[2,1]],[Bo2_1[3,1]])$
Bo3:matrix([Bo3_1[1,1]],[Bo3_1[2,1]],[Bo3_1[3,1]])$
Bo4:matrix([Bo4_1[1,1]],[Bo4_1[2,1]],[Bo4_1[3,1]])$
/*  Um vetor qualquer 'S0' que parte de {O} até o eixo da heliforça pode ser 'Bo'  */
S01:Bo1$
S02:Bo2$
S03:Bo3$
S04:Bo4$
/* Vetor 'S' unitário simplificado e expandido  */
Su1:matrix([S1X],[S1Y],[S1Z]);
Su1A1B1:(Ao1-Bo1)/L1;
Su2:matrix([S2X],[S2Y],[S2Z]);
Su2A2B2:(Ao2-Bo2)/L2;
Su3:matrix([S3X],[S3Y],[S3Z]);
Su3A3B3:(Ao3-Bo3)/L3;

Su4:matrix([S4X],[S4Y],[S4Z]);
Su4A4B4:(Ao4-Bo4)/L4;
/*  CÁLCULO DO PRODUTO S0xS  */
S01xSu1:matrix([S01[2,1]*Su1[3,1]-Su1[2,1]*S01[3,1]],[S01[3,1]*Su1[1,1]-
Su1[3,1]*S01[1,1]],[S01[1,1]*Su1[2,1]-Su1[1,1]*S01[2,1]])$
S02xSu2:matrix([S02[2,1]*Su2[3,1]-Su2[2,1]*S02[3,1]],[S02[3,1]*Su2[1,1]-
Su2[3,1]*S02[1,1]],[S02[1,1]*Su2[2,1]-Su2[1,1]*S02[2,1]])$
S03xSu3:matrix([S03[2,1]*Su3[3,1]-Su3[2,1]*S03[3,1]],[S03[3,1]*Su3[1,1]-
Su3[3,1]*S03[1,1]],[S03[1,1]*Su3[2,1]-Su3[1,1]*S03[2,1]])$
S04xSu4:matrix([S04[2,1]*Su4[3,1]-Su4[2,1]*S04[3,1]],[S04[3,1]*Su4[1,1]-
Su4[3,1]*S04[1,1]],[S04[1,1]*Su4[2,1]-Su4[1,1]*S04[2,1]])$
/* MONTAGEM DA MATRIZ DE HELIFORÇA PARA CADA CABO  */;
S1:Su1$
S2:Su2$
S3:Su3$
S4:Su4$
S1:addrow(S1,[S01xSu1[1,1]],[S01xSu1[2,1]],[S01xSu1[3,1]]);
S2:addrow(S2,[S02xSu2[1,1]],[S02xSu2[2,1]],[S02xSu2[3,1]]);
S3:addrow(S3,[S03xSu3[1,1]],[S03xSu3[2,1]],[S03xSu3[3,1]]);
```

```
S4:addrow(S4,[S04xSu4[1,1]],[S04xSu4[2,1]],[S04xSu4[3,1]]);
/* Equilíbrio de forças em X, Y e Z provenientes das heliforças  */
eq1:S1[1,1]*F1 + S2[1,1]*F2 + S3[1,1]*F3 + S4[1,1]*F4 - Sp[1,1];
eq2:S1[2,1]*F1 + S2[2,1]*F2 + S3[2,1]*F3 + S4[2,1]*F4 - Sp[2,1]$
eq3:S1[3,1]*F1 + S2[3,1]*F2 + S3[3,1]*F3 + S4[3,1]*F4 - Sp[3,1]$
eq4:S1[4,1]*F1 + S2[4,1]*F2 + S3[4,1]*F3 + S4[4,1]*F4 - Sp[4,1]$
eq5:S1[5,1]*F1 + S2[5,1]*F2 + S3[5,1]*F3 + S4[5,1]*F4 - Sp[5,1]$
eq6:S1[6,1]*F1 + S2[6,1]*F2 + S3[6,1]*F3 + S4[6,1]*F4 - Sp[6,1]$

/* Substituição de valores para as 6 equações  */
eq7:Su1[1,1]=Su1A1B1[1,1]$
eq8:Su1[2,1]=Su1A1B1[2,1]$
eq9:Su1[3,1]=Su1A1B1[3,1]$

eq10:Su2[1,1]=Su2A2B2[1,1]$
eq11:Su2[2,1]=Su2A2B2[2,1]$
eq12:Su2[3,1]=Su2A2B2[3,1]$

eq13:Su3[1,1]=Su3A3B3[1,1]$
eq14:Su3[2,1]=Su3A3B3[2,1]$
eq15:Su3[3,1]=Su3A3B3[3,1]$

eq16:Su4[1,1]=Su4A4B4[1,1]$
eq17:Su4[2,1]=Su4A4B4[2,1]$
eq18:Su4[3,1]=Su4A4B4[3,1]$
eq1:subst([eq7,eq10,eq13,eq16],eq1);
eq2:subst([eq8,eq11,eq14,eq17],eq2)$
eq3:subst([eq9,eq12,eq15,eq18],eq3)$
eq4:subst([eq8,eq9,eq11,eq12,eq14,eq15,eq17,eq18],eq4)$
eq5:subst([eq7,eq9,eq10,eq12,eq13,eq15,eq16,eq18],eq5)$
eq6:subst([eq7,eq8,eq10,eq11,eq13,eq14,eq16,eq17],eq6)$
E1:eq1$
E2:eq2$
E3:eq3$
E4:eq4$
E5:eq5$
```

```
E6:eq6$
/* Solução da equação linear Forças em [N] e negativas na direção de -Z. */
globalsolve: true$
F:float(algsys([E1,E2,E3,E4,E5,E6],[F1,F2,F3,F4]));
/*  Conferindo as seis equações substituindo as forças encontradas  */
FORCES:F[1]$
E1:subst([FORCES[1],FORCES[2],FORCES[3],FORCES[4]],E1);
E2:subst([FORCES[1],FORCES[2],FORCES[3],FORCES[4]],E2);
E3:subst([FORCES[1],FORCES[2],FORCES[3],FORCES[4]],E3);
E4:subst([FORCES[1],FORCES[2],FORCES[3],FORCES[4]],E4);
E5:subst([FORCES[1],FORCES[2],FORCES[3],FORCES[4]],E5);
E6:subst([FORCES[1],FORCES[2],FORCES[3],FORCES[4]],E6);
/* valores das forças em [x10^-2 N] para comparar com os medidos pelo sistema mola-potenciômetro */
SENSOR_1:-F1*100$
SENSOR_2:-F2*100$
SENSOR_3:-F3*100$
SENSOR_4:-F4*100$
SENSOR_1:subst(FORCES[1],SENSOR_1);
SENSOR_2:subst(FORCES[2],SENSOR_2);
SENSOR_3:subst(FORCES[3],SENSOR_3);
SENSOR_4:subst(FORCES[4],SENSOR_4);
/*  Deslocamento de cada mola
    1.4715 [N] é a pré-carga da mola.
*/
x:(SENSOR_1*0.01 - 1.4715)/196$
Xmola1:x*1000;
x:(SENSOR_2*0.01 - 1.4715)/196$
Xmola2:x*1000;
x:(SENSOR_3*0.01 - 1.4715)/196$
Xmola3:x*1000;
x:(SENSOR_4*0.01 - 1.4715)/196$
Xmola4:x*1000;
/* Comprimento dos cabos em [mm] do ponto (Ai) até o início da mola (comprimento inicial da mola é 57 mm)  */
C1:(L1*1000)-Xmola1-57;
```

C2:(L2*1000)-Xmola2-57;

C3:(L3*1000)-Xmola3-57;

C4:(L4*1000)-Xmola4-57;

/* O motor de passo executa 2048 passos para dar uma volta puxando 100 mm do cabo */

Lo1: 826-57$

Lo2: 826-57$

Lo3: 826-57$

Lo4: 826-57$

Lo5: 826-57$ /* comprimento dos cabos com a plataforma em 0,0,0 e retirando o comprimento da mola*/

/* Para a plataforma sair da origem 0,0,0 e ir até as coordenadas PX, PY e PZ (dadas no início do programa)

deve-se subtrair o comprimento anterior do atual e converter para o número de passos que o motor deve realizar.

Valores de passo negativos significa que o motor deve puxar o cabo.

*/

PASSOS_MOTOR1: (C1 - Lo1)*2048/100;

PASSOS_MOTOR2: (C2 - Lo2)*2048/100;

PASSOS_MOTOR3: (C3 - Lo3)*2048/100;

PASSOS_MOTOR4: (C4 - Lo4)*2048/100;

APÊNDICE 3 - PROGRAMA DESENVOLVIDO NO SCILAB PARA CÁLCULO DAS EQUAÇÕES LINEARES E NÃO LINEARES DO ROBÔ

```
// ESTE PROGRAMA ESTÁ DISPONIBILIZADO NO LINK DO GOOGLE DRIVE ABAIXO:

//          https://drive.google.com/open?id=0B9jot9RnR3XiSkFBNFZEWlBWT2s

clear;
clc;

tempo_inicial=getdate();
disp("INÍCIO:");
disp(tempo_inicial);

global peso;
global F1;
global F2;
global F3;
global F4;
global Lo;

Lo = 826 - 57; // Comprimento inicial do cabo até a fixação na mola

peso = (828/1000)*9.81;

B1 = [-0.0395; -0.052; 0.075;1];
B2 = [-0.0395; 0.052; 0.075;1];
B3 = [0.0395; 0.052; 0.075;1];
B4 = [0.0395; -0.052; 0.075;1];

A01 = [-0.382;-0.382;0.750;1];
A02 = [-0.382;0.382;0.750;1];
A03 = [0.382;0.382;0.750;1];
A04 = [0.382;-0.382;0.750;1];

PX=-0;  // Valores em metros
PY=0;
PZ=0;

theta=0*%pi/180;
phi=  0*%pi/180;
psi=  0*%pi/180;

          RX0=[1,       0,          0;
                0,  cos(theta),   -sin(theta);
                0,  sin(theta),    cos(theta)];

          RY0=[cos(phi),0,    sin(phi);
                        0,   1,       0;
                 -sin(phi),0,    cos(phi)];
```

```
RZ0=[cos(psi),-sin(psi),0;
     sin(psi),cos(psi), 0;
            0,     0,      1];

EulerXYZ=RX0*RY0*RZ0;

EulerXYZ = [EulerXYZ;[0,0,0]];


TH = [EulerXYZ,[PX;PY;PZ;1]];
B01 = TH * B1;
B02 = TH * B2;
B03 = TH * B3;
B04 = TH * B4;

B1A1=A01-B01;
B2A2=A02-B02;
B3A3=A03-B03;
B4A4=A04-B04;

NORMA_B1A1=sqrt(B1A1(1,1)^2+B1A1(2,1)^2+B1A1(3,1)^2);
NORMA_B2A2=sqrt(B2A2(1,1)^2+B2A2(2,1)^2+B2A2(3,1)^2);
NORMA_B3A3=sqrt(B3A3(1,1)^2+B3A3(2,1)^2+B3A3(3,1)^2);
NORMA_B4A4=sqrt(B4A4(1,1)^2+B4A4(2,1)^2+B4A4(3,1)^2);
//************************************************************

// EQ1:      EQ1F1 * F1 + EQ1F2 * F2 + EQ1F3 * F3 + EQ1F4 * F4 = 0

EQ1F1=((-0.052*cos(phi)*sin(psi)+0.0395*cos(phi)*cos(psi)-0.075*sin(phi)-PX-
0.382))/(sqrt((0.052*(cos(psi)*cos(theta)-
sin(phi)*sin(psi)*sin(theta))+0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))+0.075*c
os(phi)*sin(theta)-PY-0.382)^2+(0.0395*(sin(psi)*sin(theta)-
sin(phi)*cos(psi)*cos(theta))+0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))-
0.075*cos(phi)*cos(theta)-PZ+0.75)^2+(-0.052*cos(phi)*sin(psi)+0.0395*cos(phi)*cos(psi)-
0.075*sin(phi)-PX-0.382)^2));

EQ1F2=((0.052*cos(phi)*sin(psi)+0.0395*cos(phi)*cos(psi)-0.075*sin(phi)-PX-
0.382))/(sqrt((-0.052*(cos(psi)*cos(theta)-
sin(phi)*sin(psi)*sin(theta))+0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))+0.075*c
os(phi)*sin(theta)-PY+0.382)^2+(0.0395*(sin(psi)*sin(theta)-sin(phi)*cos(psi)*cos(theta))-
0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))-0.075*cos(phi)*cos(theta)-
PZ+0.75)^2+(0.052*cos(phi)*sin(psi)+0.0395*cos(phi)*cos(psi)-0.075*sin(phi)-PX-0.382)^2));

EQ1F3=((0.052*cos(phi)*sin(psi)-0.0395*cos(phi)*cos(psi)-0.075*sin(phi)-
PX+0.382))/(sqrt((-0.052*(cos(psi)*cos(theta)-sin(phi)*sin(psi)*sin(theta))-
0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))+0.075*cos(phi)*sin(theta)-
PY+0.382)^2+(-0.0395*(sin(psi)*sin(theta)-sin(phi)*cos(psi)*cos(theta))-
0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))-0.075*cos(phi)*cos(theta)-
PZ+0.75)^2+(0.052*cos(phi)*sin(psi)-0.0395*cos(phi)*cos(psi)-0.075*sin(phi)-PX+0.382)^2));

EQ1F4=((-0.052*cos(phi)*sin(psi)-0.0395*cos(phi)*cos(psi)-0.075*sin(phi)-
PX+0.382))/(sqrt((0.052*(cos(psi)*cos(theta)-sin(phi)*sin(psi)*sin(theta))-
0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))+0.075*cos(phi)*sin(theta)-PY-
```

```
0.382)^2+(-0.0395*(sin(psi)*sin(theta)-
sin(phi)*cos(psi)*cos(theta))+0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))-
0.075*cos(phi)*cos(theta)-PZ+0.75)^2+(-0.052*cos(phi)*sin(psi)-0.0395*cos(phi)*cos(psi)-
0.075*sin(phi)-PX+0.382)^2));

EQ1 = 0;

// EQ2:
        EQ2F1=((0.052*(cos(psi)*cos(theta)-
sin(phi)*sin(psi)*sin(theta))+0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))+0.075*c
os(phi)*sin(theta)-PY-0.382))/(sqrt((0.052*(cos(psi)*cos(theta)-
sin(phi)*sin(psi)*sin(theta))+0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))+0.075*c
os(phi)*sin(theta)-PY-0.382)^2+(0.0395*(sin(psi)*sin(theta)-
sin(phi)*cos(psi)*cos(theta))+0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))-
0.075*cos(phi)*cos(theta)-PZ+0.75)^2+(-0.052*cos(phi)*sin(psi)+0.0395*cos(phi)*cos(psi)-
0.075*sin(phi)-PX-0.382)^2));

        EQ2F2=((-0.052*(cos(psi)*cos(theta)-
sin(phi)*sin(psi)*sin(theta))+0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))+0.075*c
os(phi)*sin(theta)-PY+0.382))/(sqrt((-0.052*(cos(psi)*cos(theta)-
sin(phi)*sin(psi)*sin(theta))+0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))+0.075*c
os(phi)*sin(theta)-PY+0.382)^2+(0.0395*(sin(psi)*sin(theta)-sin(phi)*cos(psi)*cos(theta))-
0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))-0.075*cos(phi)*cos(theta)-
PZ+0.75)^2+(0.052*cos(phi)*sin(psi)+0.0395*cos(phi)*cos(psi)-0.075*sin(phi)-PX-0.382)^2));

        EQ2F3=((-0.052*(cos(psi)*cos(theta)-sin(phi)*sin(psi)*sin(theta))-
0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))+0.075*cos(phi)*sin(theta)-
PY+0.382))/(sqrt((-0.052*(cos(psi)*cos(theta)-sin(phi)*sin(psi)*sin(theta))-
0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))+0.075*cos(phi)*sin(theta)-
PY+0.382)^2+(-0.0395*(sin(psi)*sin(theta)-sin(phi)*cos(psi)*cos(theta))-
0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))-0.075*cos(phi)*cos(theta)-
PZ+0.75)^2+(0.052*cos(phi)*sin(psi)-0.0395*cos(phi)*cos(psi)-0.075*sin(phi)-PX+0.382)^2));

        EQ2F4=((0.052*(cos(psi)*cos(theta)-sin(phi)*sin(psi)*sin(theta))-
0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))+0.075*cos(phi)*sin(theta)-PY-
0.382))/(sqrt((0.052*(cos(psi)*cos(theta)-sin(phi)*sin(psi)*sin(theta))-
0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))+0.075*cos(phi)*sin(theta)-PY-
0.382)^2+(-0.0395*(sin(psi)*sin(theta)-
sin(phi)*cos(psi)*cos(theta))+0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))-
0.075*cos(phi)*cos(theta)-PZ+0.75)^2+(-0.052*cos(phi)*sin(psi)-0.0395*cos(phi)*cos(psi)-
0.075*sin(phi)-PX+0.382)^2));

EQ2 = 0;

// EQ3:
        EQ3F1=((0.0395*(sin(psi)*sin(theta)-
sin(phi)*cos(psi)*cos(theta))+0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))-
0.075*cos(phi)*cos(theta)-PZ+0.75))/(sqrt((0.052*(cos(psi)*cos(theta)-
sin(phi)*sin(psi)*sin(theta))+0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))+0.075*c
os(phi)*sin(theta)-PY-0.382)^2+(0.0395*(sin(psi)*sin(theta)-
sin(phi)*cos(psi)*cos(theta))+0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))-
0.075*cos(phi)*cos(theta)-PZ+0.75)^2+(-0.052*cos(phi)*sin(psi)+0.0395*cos(phi)*cos(psi)-
0.075*sin(phi)-PX-0.382)^2));
```

```
        EQ3F2=((0.0395*(sin(psi)*sin(theta)-sin(phi)*cos(psi)*cos(theta))-
0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))-0.075*cos(phi)*cos(theta)-
PZ+0.75))/(sqrt((-0.052*(cos(psi)*cos(theta)-
sin(phi)*sin(psi)*sin(theta))+0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))+0.075*c
os(phi)*sin(theta)-PY+0.382)^2+(0.0395*(sin(psi)*sin(theta)-sin(phi)*cos(psi)*cos(theta))-
0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))-0.075*cos(phi)*cos(theta)-
PZ+0.75)^2+(0.052*cos(phi)*sin(psi)+0.0395*cos(phi)*cos(psi)-0.075*sin(phi)-PX-0.382)^2));

        EQ3F3=((-0.0395*(sin(psi)*sin(theta)-sin(phi)*cos(psi)*cos(theta))-
0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))-0.075*cos(phi)*cos(theta)-
PZ+0.75))/(sqrt((-0.052*(cos(psi)*cos(theta)-sin(phi)*sin(psi)*sin(theta))-
0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))+0.075*cos(phi)*sin(theta)-
PY+0.382)^2+(-0.0395*(sin(psi)*sin(theta)-sin(phi)*cos(psi)*cos(theta))-
0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))-0.075*cos(phi)*cos(theta)-
PZ+0.75)^2+(0.052*cos(phi)*sin(psi)-0.0395*cos(phi)*cos(psi)-0.075*sin(phi)-PX+0.382)^2));
        EQ3F4=((-0.0395*(sin(psi)*sin(theta)-
sin(phi)*cos(psi)*cos(theta))+0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))-
0.075*cos(phi)*cos(theta)-PZ+0.75))/(sqrt((0.052*(cos(psi)*cos(theta)-
sin(phi)*sin(psi)*sin(theta))-
0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))+0.075*cos(phi)*sin(theta)-PY-
0.382)^2+(-0.0395*(sin(psi)*sin(theta)-
sin(phi)*cos(psi)*cos(theta))+0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))-
0.075*cos(phi)*cos(theta)-PZ+0.75)^2+(-0.052*cos(phi)*sin(psi)-0.0395*cos(phi)*cos(psi)-
0.075*sin(phi)-PX+0.382)^2));

EQ3 = -peso;

// EQ4:
        EQ4F1=(((0.0395*(sin(psi)*sin(theta)-
sin(phi)*cos(psi)*cos(theta))+0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))-
0.075*cos(phi)*cos(theta)-PZ+0.75)*(-0.052*(cos(psi)*cos(theta)-sin(phi)*sin(psi)*sin(theta))-
0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))-
0.075*cos(phi)*sin(theta)+PY))/(sqrt((0.052*(cos(psi)*cos(theta)-
sin(phi)*sin(psi)*sin(theta))+0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))+0.075*c
os(phi)*sin(theta)-PY-0.382)^2+(0.0395*(sin(psi)*sin(theta)-
sin(phi)*cos(psi)*cos(theta))+0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))-
0.075*cos(phi)*cos(theta)-PZ+0.75)^2+(-0.052*cos(phi)*sin(psi)+0.0395*cos(phi)*cos(psi)-
0.075*sin(phi)-PX-0.382)^2))-((-0.0395*(sin(psi)*sin(theta)-sin(phi)*cos(psi)*cos(theta))-
0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))+0.075*cos(phi)*cos(theta)+PZ)*(0.052
*(cos(psi)*cos(theta)-
sin(phi)*sin(psi)*sin(theta))+0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))+0.075*c
os(phi)*sin(theta)-PY-0.382))/(sqrt((0.052*(cos(psi)*cos(theta)-
sin(phi)*sin(psi)*sin(theta))+0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))+0.075*c
os(phi)*sin(theta)-PY-0.382)^2+(0.0395*(sin(psi)*sin(theta)-
sin(phi)*cos(psi)*cos(theta))+0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))-
0.075*cos(phi)*cos(theta)-PZ+0.75)^2+(-0.052*cos(phi)*sin(psi)+0.0395*cos(phi)*cos(psi)-
0.075*sin(phi)-PX-0.382)^2)));

        EQ4F2=(((0.0395*(sin(psi)*sin(theta)-sin(phi)*cos(psi)*cos(theta))-
0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))-0.075*cos(phi)*cos(theta)-
PZ+0.75)*(0.052*(cos(psi)*cos(theta)-sin(phi)*sin(psi)*sin(theta))-
0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))-
```

```
0.075*cos(phi)*sin(theta)+PY))/(sqrt((-0.052*(cos(psi)*cos(theta)-
sin(phi)*sin(psi)*sin(theta))+0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))+0.075*c
os(phi)*sin(theta)-PY+0.382)^2+(0.0395*(sin(psi)*sin(theta)-sin(phi)*cos(psi)*cos(theta))-
0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))-0.075*cos(phi)*cos(theta)-
PZ+0.75)^2+(0.052*cos(phi)*sin(psi)+0.0395*cos(phi)*cos(psi)-0.075*sin(phi)-PX-0.382)^2))-
((-0.0395*(sin(psi)*sin(theta)-
sin(phi)*cos(psi)*cos(theta))+0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))+0.075*c
os(phi)*cos(theta)+PZ)*(-0.052*(cos(psi)*cos(theta)-
sin(phi)*sin(psi)*sin(theta))+0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))+0.075*c
os(phi)*sin(theta)-PY+0.382))/(sqrt((-0.052*(cos(psi)*cos(theta)-
sin(phi)*sin(psi)*sin(theta))+0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))+0.075*c
os(phi)*sin(theta)-PY+0.382)^2+(0.0395*(sin(psi)*sin(theta)-sin(phi)*cos(psi)*cos(theta))-
0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))-0.075*cos(phi)*cos(theta)-
PZ+0.75)^2+(0.052*cos(phi)*sin(psi)+0.0395*cos(phi)*cos(psi)-0.075*sin(phi)-PX-
0.382)^2)));

        EQ4F3=(((-0.0395*(sin(psi)*sin(theta)-sin(phi)*cos(psi)*cos(theta))-
0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))-0.075*cos(phi)*cos(theta)-
PZ+0.75)*(0.052*(cos(psi)*cos(theta)-
sin(phi)*sin(psi)*sin(theta))+0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))-
0.075*cos(phi)*sin(theta)+PY))/(sqrt((-0.052*(cos(psi)*cos(theta)-sin(phi)*sin(psi)*sin(theta))-
0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))+0.075*cos(phi)*sin(theta)-
PY+0.382)^2+(-0.0395*(sin(psi)*sin(theta)-sin(phi)*cos(psi)*cos(theta))-
0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))-0.075*cos(phi)*cos(theta)-
PZ+0.75)^2+(0.052*cos(phi)*sin(psi)-0.0395*cos(phi)*cos(psi)-0.075*sin(phi)-PX+0.382)^2))-
((0.0395*(sin(psi)*sin(theta)-
sin(phi)*cos(psi)*cos(theta))+0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))+0.075*c
os(phi)*cos(theta)+PZ)*(-0.052*(cos(psi)*cos(theta)-sin(phi)*sin(psi)*sin(theta))-
0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))+0.075*cos(phi)*sin(theta)-
PY+0.382))/(sqrt((-0.052*(cos(psi)*cos(theta)-sin(phi)*sin(psi)*sin(theta))-
0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))+0.075*cos(phi)*sin(theta)-
PY+0.382)^2+(-0.0395*(sin(psi)*sin(theta)-sin(phi)*cos(psi)*cos(theta))-
0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))-0.075*cos(phi)*cos(theta)-
PZ+0.75)^2+(0.052*cos(phi)*sin(psi)-0.0395*cos(phi)*cos(psi)-0.075*sin(phi)-
PX+0.382)^2)));

        EQ4F4=(((-0.0395*(sin(psi)*sin(theta)-
sin(phi)*cos(psi)*cos(theta))+0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))-
0.075*cos(phi)*cos(theta)-PZ+0.75)*(-0.052*(cos(psi)*cos(theta)-
sin(phi)*sin(psi)*sin(theta))+0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))-
0.075*cos(phi)*sin(theta)+PY))/(sqrt((0.052*(cos(psi)*cos(theta)-sin(phi)*sin(psi)*sin(theta))-
0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))+0.075*cos(phi)*sin(theta)-PY-
0.382)^2+(-0.0395*(sin(psi)*sin(theta)-
sin(phi)*cos(psi)*cos(theta))+0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))-
0.075*cos(phi)*cos(theta)-PZ+0.75)^2+(-0.052*cos(phi)*sin(psi)-0.0395*cos(phi)*cos(psi)-
0.075*sin(phi)-PX+0.382)^2))-((0.0395*(sin(psi)*sin(theta)-sin(phi)*cos(psi)*cos(theta))-
0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))+0.075*cos(phi)*cos(theta)+PZ)*(0.052
*(cos(psi)*cos(theta)-sin(phi)*sin(psi)*sin(theta))-
0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))+0.075*cos(phi)*sin(theta)-PY-
0.382))/(sqrt((0.052*(cos(psi)*cos(theta)-sin(phi)*sin(psi)*sin(theta))-
0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))+0.075*cos(phi)*sin(theta)-PY-
0.382)^2+(-0.0395*(sin(psi)*sin(theta)-
sin(phi)*cos(psi)*cos(theta))+0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))-
```

```
0.075*cos(phi)*cos(theta)-PZ+0.75)^2+(-0.052*cos(phi)*sin(psi)-0.0395*cos(phi)*cos(psi)-
0.075*sin(phi)-PX+0.382)^2)));


EQ4 = -PY*peso;

// EQ5:
        EQ5F1=(((-0.052*cos(phi)*sin(psi)+0.0395*cos(phi)*cos(psi)-0.075*sin(phi)-PX-
0.382)*(-0.0395*(sin(psi)*sin(theta)-sin(phi)*cos(psi)*cos(theta))-
0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))+0.075*cos(phi)*cos(theta)+PZ))/(sqrt(
(0.052*(cos(psi)*cos(theta)-
sin(phi)*sin(psi)*sin(theta))+0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))+0.075*c
os(phi)*sin(theta)-PY-0.382)^2+(0.0395*(sin(psi)*sin(theta)-
sin(phi)*cos(psi)*cos(theta))+0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))-
0.075*cos(phi)*cos(theta)-PZ+0.75)^2+(-0.052*cos(phi)*sin(psi)+0.0395*cos(phi)*cos(psi)-
0.075*sin(phi)-PX-0.382)^2))-((0.052*cos(phi)*sin(psi)-
0.0395*cos(phi)*cos(psi)+0.075*sin(phi)+PX)*(0.0395*(sin(psi)*sin(theta)-
sin(phi)*cos(psi)*cos(theta))+0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))-
0.075*cos(phi)*cos(theta)-PZ+0.75))/(sqrt((0.052*(cos(psi)*cos(theta)-
sin(phi)*sin(psi)*sin(theta))+0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))+0.075*c
os(phi)*sin(theta)-PY-0.382)^2+(0.0395*(sin(psi)*sin(theta)-
sin(phi)*cos(psi)*cos(theta))+0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))-
0.075*cos(phi)*cos(theta)-PZ+0.75)^2+(-0.052*cos(phi)*sin(psi)+0.0395*cos(phi)*cos(psi)-
0.075*sin(phi)-PX-0.382)^2)));

        EQ5F2=(((0.052*cos(phi)*sin(psi)+0.0395*cos(phi)*cos(psi)-0.075*sin(phi)-PX-
0.382)*(-0.0395*(sin(psi)*sin(theta)-
sin(phi)*cos(psi)*cos(theta))+0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))+0.075*c
os(phi)*cos(theta)+PZ))/(sqrt((-0.052*(cos(psi)*cos(theta)-
sin(phi)*sin(psi)*sin(theta))+0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))+0.075*c
os(phi)*sin(theta)-PY+0.382)^2+(0.0395*(sin(psi)*sin(theta)-sin(phi)*cos(psi)*cos(theta))-
0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))-0.075*cos(phi)*cos(theta)-
PZ+0.75)^2+(0.052*cos(phi)*sin(psi)+0.0395*cos(phi)*cos(psi)-0.075*sin(phi)-PX-0.382)^2))-
((-0.052*cos(phi)*sin(psi)-
0.0395*cos(phi)*cos(psi)+0.075*sin(phi)+PX)*(0.0395*(sin(psi)*sin(theta)-
sin(phi)*cos(psi)*cos(theta))-0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))-
0.075*cos(phi)*cos(theta)-PZ+0.75))/(sqrt((-0.052*(cos(psi)*cos(theta)-
sin(phi)*sin(psi)*sin(theta))+0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))+0.075*c
os(phi)*sin(theta)-PY+0.382)^2+(0.0395*(sin(psi)*sin(theta)-sin(phi)*cos(psi)*cos(theta))-
0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))-0.075*cos(phi)*cos(theta)-
PZ+0.75)^2+(0.052*cos(phi)*sin(psi)+0.0395*cos(phi)*cos(psi)-0.075*sin(phi)-PX-
0.382)^2)));

        EQ5F3=(((0.052*cos(phi)*sin(psi)-0.0395*cos(phi)*cos(psi)-0.075*sin(phi)-
PX+0.382)*(0.0395*(sin(psi)*sin(theta)-
sin(phi)*cos(psi)*cos(theta))+0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))+0.075*c
os(phi)*cos(theta)+PZ))/(sqrt((-0.052*(cos(psi)*cos(theta)-sin(phi)*sin(psi)*sin(theta))-
0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))+0.075*cos(phi)*sin(theta)-
PY+0.382)^2+(-0.0395*(sin(psi)*sin(theta)-sin(phi)*cos(psi)*cos(theta))-
0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))-0.075*cos(phi)*cos(theta)-
PZ+0.75)^2+(0.052*cos(phi)*sin(psi)-0.0395*cos(phi)*cos(psi)-0.075*sin(phi)-PX+0.382)^2))-
((-0.052*cos(phi)*sin(psi)+0.0395*cos(phi)*cos(psi)+0.075*sin(phi)+PX)*(-
0.0395*(sin(psi)*sin(theta)-sin(phi)*cos(psi)*cos(theta))-
```

```
0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))-0.075*cos(phi)*cos(theta)-
PZ+0.75))/(sqrt((-0.052*(cos(psi)*cos(theta)-sin(phi)*sin(psi)*sin(theta))-
0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))+0.075*cos(phi)*sin(theta)-
PY+0.382)^2+(-0.0395*(sin(psi)*sin(theta)-sin(phi)*cos(psi)*cos(theta))-
0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))-0.075*cos(phi)*cos(theta)-
PZ+0.75)^2+(0.052*cos(phi)*sin(psi)-0.0395*cos(phi)*cos(psi)-0.075*sin(phi)-
PX+0.382)^2)));

        EQ5F4=(((-0.052*cos(phi)*sin(psi)-0.0395*cos(phi)*cos(psi)-0.075*sin(phi)-
PX+0.382)*(0.0395*(sin(psi)*sin(theta)-sin(phi)*cos(psi)*cos(theta))-
0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))+0.075*cos(phi)*cos(theta)+PZ))/(sqrt(
(0.052*(cos(psi)*cos(theta)-sin(phi)*sin(psi)*sin(theta))-
0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))+0.075*cos(phi)*sin(theta)-PY-
0.382)^2+(-0.0395*(sin(psi)*sin(theta)-
sin(phi)*cos(psi)*cos(theta))+0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))-
0.075*cos(phi)*cos(theta)-PZ+0.75)^2+(-0.052*cos(phi)*sin(psi)-0.0395*cos(phi)*cos(psi)-
0.075*sin(phi)-PX+0.382)^2))-
((0.052*cos(phi)*sin(psi)+0.0395*cos(phi)*cos(psi)+0.075*sin(phi)+PX)*(-
0.0395*(sin(psi)*sin(theta)-
sin(phi)*cos(psi)*cos(theta))+0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))-
0.075*cos(phi)*cos(theta)-PZ+0.75))/(sqrt((0.052*(cos(psi)*cos(theta)-
sin(phi)*sin(psi)*sin(theta))-
0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))+0.075*cos(phi)*sin(theta)-PY-
0.382)^2+(-0.0395*(sin(psi)*sin(theta)-
sin(phi)*cos(psi)*cos(theta))+0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))-
0.075*cos(phi)*cos(theta)-PZ+0.75)^2+(-0.052*cos(phi)*sin(psi)-0.0395*cos(phi)*cos(psi)-
0.075*sin(phi)-PX+0.382)^2)));

EQ5 = PX*peso;

// EQ6:
        EQ6F1=(((0.052*cos(phi)*sin(psi)-
0.0395*cos(phi)*cos(psi)+0.075*sin(phi)+PX)*(0.052*(cos(psi)*cos(theta)-
sin(phi)*sin(psi)*sin(theta))+0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))+0.075*c
os(phi)*sin(theta)-PY-0.382))/(sqrt((0.052*(cos(psi)*cos(theta)-
sin(phi)*sin(psi)*sin(theta))+0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))+0.075*c
os(phi)*sin(theta)-PY-0.382)^2+(0.0395*(sin(psi)*sin(theta)-
sin(phi)*cos(psi)*cos(theta))+0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))-
0.075*cos(phi)*cos(theta)-PZ+0.75)^2+(-0.052*cos(phi)*sin(psi)+0.0395*cos(phi)*cos(psi)-
0.075*sin(phi)-PX-0.382)^2))-((-0.052*cos(phi)*sin(psi)+0.0395*cos(phi)*cos(psi)-
0.075*sin(phi)-PX-0.382)*(-0.052*(cos(psi)*cos(theta)-sin(phi)*sin(psi)*sin(theta))-
0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))-
0.075*cos(phi)*sin(theta)+PY))/(sqrt((0.052*(cos(psi)*cos(theta)-
sin(phi)*sin(psi)*sin(theta))+0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))+0.075*c
os(phi)*sin(theta)-PY-0.382)^2+(0.0395*(sin(psi)*sin(theta)-
sin(phi)*cos(psi)*cos(theta))+0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))-
0.075*cos(phi)*cos(theta)-PZ+0.75)^2+(-0.052*cos(phi)*sin(psi)+0.0395*cos(phi)*cos(psi)-
0.075*sin(phi)-PX-0.382)^2)));


EQ6F4=(((0.052*cos(phi)*sin(psi)+0.0395*cos(phi)*cos(psi)+0.075*sin(phi)+PX)*(0.052*(cos(
psi)*cos(theta)-sin(phi)*sin(psi)*sin(theta))-
0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))+0.075*cos(phi)*sin(theta)-PY-
```

```
0.382))/(sqrt((0.052*(cos(psi)*cos(theta)-sin(phi)*sin(psi)*sin(theta))-
0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))+0.075*cos(phi)*sin(theta)-PY-
0.382)^2+(-0.0395*(sin(psi)*sin(theta)-
sin(phi)*cos(psi)*cos(theta))+0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))-
0.075*cos(phi)*cos(theta)-PZ+0.75)^2+(-0.052*cos(phi)*sin(psi)-0.0395*cos(phi)*cos(psi)-
0.075*sin(phi)-PX+0.382)^2))-((-0.052*cos(phi)*sin(psi)-0.0395*cos(phi)*cos(psi)-
0.075*sin(phi)-PX+0.382)*(-0.052*(cos(psi)*cos(theta)-
sin(phi)*sin(psi)*sin(theta))+0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))-
0.075*cos(phi)*sin(theta)+PY))/(sqrt((0.052*(cos(psi)*cos(theta)-sin(phi)*sin(psi)*sin(theta))-
0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))+0.075*cos(phi)*sin(theta)-PY-
0.382)^2+(-0.0395*(sin(psi)*sin(theta)-
sin(phi)*cos(psi)*cos(theta))+0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))-
0.075*cos(phi)*cos(theta)-PZ+0.75)^2+(-0.052*cos(phi)*sin(psi)-0.0395*cos(phi)*cos(psi)-
0.075*sin(phi)-PX+0.382)^2)));

        EQ6F2=(((-0.052*cos(phi)*sin(psi)-0.0395*cos(phi)*cos(psi)+0.075*sin(phi)+PX)*(-
0.052*(cos(psi)*cos(theta)-
sin(phi)*sin(psi)*sin(theta))+0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))+0.075*c
os(phi)*sin(theta)-PY+0.382))/(sqrt((-0.052*(cos(psi)*cos(theta)-
sin(phi)*sin(psi)*sin(theta))+0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))+0.075*c
os(phi)*sin(theta)-PY+0.382)^2+(0.0395*(sin(psi)*sin(theta)-sin(phi)*cos(psi)*cos(theta))-
0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))-0.075*cos(phi)*cos(theta)-
PZ+0.75)^2+(0.052*cos(phi)*sin(psi)+0.0395*cos(phi)*cos(psi)-0.075*sin(phi)-PX-0.382)^2))-
((0.052*cos(phi)*sin(psi)+0.0395*cos(phi)*cos(psi)-0.075*sin(phi)-PX-
0.382)*(0.052*(cos(psi)*cos(theta)-sin(phi)*sin(psi)*sin(theta))-
0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))-
0.075*cos(phi)*sin(theta)+PY))/(sqrt((-0.052*(cos(psi)*cos(theta)-
sin(phi)*sin(psi)*sin(theta))+0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))+0.075*c
os(phi)*sin(theta)-PY+0.382)^2+(0.0395*(sin(psi)*sin(theta)-sin(phi)*cos(psi)*cos(theta))-
0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))-0.075*cos(phi)*cos(theta)-
PZ+0.75)^2+(0.052*cos(phi)*sin(psi)+0.0395*cos(phi)*cos(psi)-0.075*sin(phi)-PX-
0.382)^2)));

        EQ6F3=(((-0.052*cos(phi)*sin(psi)+0.0395*cos(phi)*cos(psi)+0.075*sin(phi)+PX)*(-
0.052*(cos(psi)*cos(theta)-sin(phi)*sin(psi)*sin(theta))-
0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))+0.075*cos(phi)*sin(theta)-
PY+0.382))/(sqrt((-0.052*(cos(psi)*cos(theta)-sin(phi)*sin(psi)*sin(theta))-
0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))+0.075*cos(phi)*sin(theta)-
PY+0.382)^2+(-0.0395*(sin(psi)*sin(theta)-sin(phi)*cos(psi)*cos(theta))-
0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))-0.075*cos(phi)*cos(theta)-
PZ+0.75)^2+(0.052*cos(phi)*sin(psi)-0.0395*cos(phi)*cos(psi)-0.075*sin(phi)-PX+0.382)^2))-
((0.052*cos(phi)*sin(psi)-0.0395*cos(phi)*cos(psi)-0.075*sin(phi)-
PX+0.382)*(0.052*(cos(psi)*cos(theta)-
sin(phi)*sin(psi)*sin(theta))+0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))-
0.075*cos(phi)*sin(theta)+PY))/(sqrt((-0.052*(cos(psi)*cos(theta)-sin(phi)*sin(psi)*sin(theta))-
0.0395*(sin(phi)*cos(psi)*sin(theta)+sin(psi)*cos(theta))+0.075*cos(phi)*sin(theta)-
PY+0.382)^2+(-0.0395*(sin(psi)*sin(theta)-sin(phi)*cos(psi)*cos(theta))-
0.052*(cos(psi)*sin(theta)+sin(phi)*sin(psi)*cos(theta))-0.075*cos(phi)*cos(theta)-
PZ+0.75)^2+(0.052*cos(phi)*sin(psi)-0.0395*cos(phi)*cos(psi)-0.075*sin(phi)-
PX+0.382)^2)));

EQ6 = 0;
```

```
M1 = [EQ3F1,EQ3F2,EQ3F3,EQ3F4;
     EQ4F1,EQ4F2,EQ4F3,EQ4F4;
     EQ5F1,EQ5F2,EQ5F3,EQ5F4;
     EQ6F1,EQ6F2,EQ6F3,EQ6F4];

M2 =[EQ3;EQ4;EQ5;EQ6];

res = linsolve(M1,M2);

F1=res(1);
F2=res(2);
F3=res(3);
F4=res(4);

Xmola1 = ((F1-1.4715)/196)*1000;
Xmola2 = ((F2-1.4715)/196)*1000;
Xmola3 = ((F3-1.4715)/196)*1000;
Xmola4 = ((F4-1.4715)/196)*1000;


// Comprimento até a fixação na mola. O comprimento da mola sem força aplicada é de 57 mm.
C1 = (NORMA_B1A1*1000)-Xmola1-57;
C2 = (NORMA_B2A2*1000)-Xmola2-57;
C3 = (NORMA_B3A3*1000)-Xmola3-57;
C4 = (NORMA_B4A4*1000)-Xmola4-57;


disp("COMPRIMENTO DOS CABOS ATÉ A FIXAÇÃO NA MOLA:");
disp("C1 = "+string(round(C1))+" mm MOTOR 1: "+string(round((C1-Lo)*2048/100))+" passos com a plataforma saindo da origem.");
disp("C2 = "+string(round(C2))+" mm MOTOR 2: "+string(round((C2-Lo)*2048/100))+" passos com a plataforma saindo da origem.");
disp("C3 = "+string(round(C3))+" mm MOTOR 3: "+string(round((C3-Lo)*2048/100))+" passos com a plataforma saindo da origem.");
disp("C4 = "+string(round(C4))+" mm MOTOR 4: "+string(round((C4-Lo)*2048/100))+" passos com a plataforma saindo da origem.");

disp("CÁLCULO DAS FORÇAS PELAS EQUAÇÕES LINEARES: FZ = - PESO = "+string((EQ3F1*F1+EQ3F2*F2+EQ3F3*F3+EQ3F4*F4)*100)+" x10^-2N ou "+string(1000*(EQ3F1*F1+EQ3F2*F2+EQ3F3*F3+EQ3F4*F4)/9.81)+" Gramas");
disp("F1 = "+string(F1*100)+" [x10^-2N] F1x= "+string(EQ1F1*F1*100)+" F1y= "+string(EQ2F1*F1*100)+" F1z= "+string(EQ3F1*F1*100));
disp("F2 = "+string(F2*100)+" [x10^-2N] F2x= "+string(EQ1F2*F2*100)+" F2y= "+string(EQ2F2*F2*100)+" F2z= "+string(EQ3F2*F2*100));
disp("F3 = "+string(F3*100)+" [x10^-2N] F3x= "+string(EQ1F3*F3*100)+" F3y= "+string(EQ2F3*F3*100)+" F3z= "+string(EQ3F3*F3*100));
disp("F4 = "+string(F4*100)+" [x10^-2N] F4x= "+string(EQ1F4*F4*100)+" F4y= "+string(EQ2F4*F4*100)+" F4z= "+string(EQ3F4*F4*100));


// A PARTIR DESTA LINHA O OBJETIVO É ENCONTRAR PX, PY, PZ, Theta, Phi e Psi pelas forças F1, F2, F3 e F4 calculadas anteriormente.
```

```
function [J, f]=jacobiano(f1, f2, f3, f4, f5, f6, x, delta)

   // Derivada das 6 equações $' em relação a PX, PY, PZ, Theta, Phi e Psi

   delta2 = delta*%pi/180;
   df1dx1  =  (f1(x(1)+delta*x(1),x(2),x(3),x(4),x(5),x(6))  -  f1(x(1),x(2),x(3),x(4),x(5),x(6)))/
(delta*x(1));
   df1dx2  =  (f1(x(1),x(2)+delta*x(2),x(3),x(4),x(5),x(6))  -  f1(x(1),x(2),x(3),x(4),x(5),x(6)))/
(delta*x(2));
   df1dx3  =  (f1(x(1),x(2),x(3)+delta*x(3),x(4),x(5),x(6))  -  f1(x(1),x(2),x(3),x(4),x(5),x(6)))/
(delta*x(3));
   df1dx4  =  (f1(x(1),x(2),x(3),x(4)+delta2*x(4),x(5),x(6))  -  f1(x(1),x(2),x(3),x(4),x(5),x(6)))/
(delta2*x(4));
   df1dx5  =  (f1(x(1),x(2),x(3),x(4),x(5)+delta2*x(5),x(6))  -  f1(x(1),x(2),x(3),x(4),x(5),x(6)))/
(delta2*x(5));
   df1dx6  =  (f1(x(1),x(2),x(3),x(4),x(5),x(6)+delta2*x(6))  -  f1(x(1),x(2),x(3),x(4),x(5),x(6)))/
(delta2*x(6));

   df2dx1  =  (f2(x(1)+delta*x(1),x(2),x(3),x(4),x(5),x(6))  -  f2(x(1),x(2),x(3),x(4),x(5),x(6)))/
(delta*x(1));
   df2dx2  =  (f2(x(1),x(2)+delta*x(2),x(3),x(4),x(5),x(6))  -  f2(x(1),x(2),x(3),x(4),x(5),x(6)))/
(delta*x(2));
   df2dx3  =  (f2(x(1),x(2),x(3)+delta*x(3),x(4),x(5),x(6))  -  f2(x(1),x(2),x(3),x(4),x(5),x(6)))/
(delta*x(3));
   df2dx4  =  (f2(x(1),x(2),x(3),x(4)+delta2*x(4),x(5),x(6))  -  f2(x(1),x(2),x(3),x(4),x(5),x(6)))/
(delta2*x(4));
   df2dx5  =  (f2(x(1),x(2),x(3),x(4),x(5)+delta2*x(5),x(6))  -  f2(x(1),x(2),x(3),x(4),x(5),x(6)))/
(delta2*x(5));
   df2dx6  =  (f2(x(1),x(2),x(3),x(4),x(5),x(6)+delta2*x(6))  -  f2(x(1),x(2),x(3),x(4),x(5),x(6)))/
(delta2*x(6));

   df3dx1  =  (f3(x(1)+delta*x(1),x(2),x(3),x(4),x(5),x(6))  -  f3(x(1),x(2),x(3),x(4),x(5),x(6)))/
(delta*x(1));
   df3dx2  =  (f3(x(1),x(2)+delta*x(2),x(3),x(4),x(5),x(6))  -  f3(x(1),x(2),x(3),x(4),x(5),x(6)))/
(delta*x(2));
   df3dx3  =  (f3(x(1),x(2),x(3)+delta*x(3),x(4),x(5),x(6))  -  f3(x(1),x(2),x(3),x(4),x(5),x(6)))/
(delta*x(3));
   df3dx4  =  (f3(x(1),x(2),x(3),x(4)+delta2*x(4),x(5),x(6))  -  f3(x(1),x(2),x(3),x(4),x(5),x(6)))/
(delta2*x(4));
   df3dx5  =  (f3(x(1),x(2),x(3),x(4),x(5)+delta2*x(5),x(6))  -  f3(x(1),x(2),x(3),x(4),x(5),x(6)))/
(delta2*x(5));
   df3dx6  =  (f3(x(1),x(2),x(3),x(4),x(5),x(6)+delta2*x(6))  -  f3(x(1),x(2),x(3),x(4),x(5),x(6)))/
(delta2*x(6));

   df4dx1  =  (f4(x(1)+delta*x(1),x(2),x(3),x(4),x(5),x(6))  -  f4(x(1),x(2),x(3),x(4),x(5),x(6)))/
(delta*x(1));
   df4dx2  =  (f4(x(1),x(2)+delta*x(2),x(3),x(4),x(5),x(6))  -  f4(x(1),x(2),x(3),x(4),x(5),x(6)))/
(delta*x(2));
   df4dx3  =  (f4(x(1),x(2),x(3)+delta*x(3),x(4),x(5),x(6))  -  f4(x(1),x(2),x(3),x(4),x(5),x(6)))/
(delta*x(3));
   df4dx4  =  (f4(x(1),x(2),x(3),x(4)+delta2*x(4),x(5),x(6))  -  f4(x(1),x(2),x(3),x(4),x(5),x(6)))/
(delta2*x(4));
```

```
    df4dx5  =  (f4(x(1),x(2),x(3),x(4),x(5)+delta2*x(5),x(6))  -  f4(x(1),x(2),x(3),x(4),x(5),x(6)))/
(delta2*x(5));
    df4dx6  =  (f4(x(1),x(2),x(3),x(4),x(5),x(6)+delta2*x(6))  -  f4(x(1),x(2),x(3),x(4),x(5),x(6)))/
(delta2*x(6));

    df5dx1  =  (f5(x(1)+delta*x(1),x(2),x(3),x(4),x(5),x(6))  -  f5(x(1),x(2),x(3),x(4),x(5),x(6)))/
(delta*x(1));
    df5dx2  =  (f5(x(1),x(2)+delta*x(2),x(3),x(4),x(5),x(6))  -  f5(x(1),x(2),x(3),x(4),x(5),x(6)))/
(delta*x(2));
    df5dx3  =  (f5(x(1),x(2),x(3)+delta*x(3),x(4),x(5),x(6))  -  f5(x(1),x(2),x(3),x(4),x(5),x(6)))/
(delta*x(3));
    df5dx4  =  (f5(x(1),x(2),x(3),x(4)+delta2*x(4),x(5),x(6))  -  f5(x(1),x(2),x(3),x(4),x(5),x(6)))/
(delta2*x(4));
    df5dx5  =  (f5(x(1),x(2),x(3),x(4),x(5)+delta2*x(5),x(6))  -  f5(x(1),x(2),x(3),x(4),x(5),x(6)))/
(delta2*x(5));
    df5dx6  =  (f5(x(1),x(2),x(3),x(4),x(5),x(6)+delta2*x(6))  -  f5(x(1),x(2),x(3),x(4),x(5),x(6)))/
(delta2*x(6));

    df6dx1  =  (f6(x(1)+delta*x(1),x(2),x(3),x(4),x(5),x(6))  -  f6(x(1),x(2),x(3),x(4),x(5),x(6)))/
(delta*x(1));
    df6dx2  =  (f6(x(1),x(2)+delta*x(2),x(3),x(4),x(5),x(6))  -  f6(x(1),x(2),x(3),x(4),x(5),x(6)))/
(delta*x(2));
    df6dx3  =  (f6(x(1),x(2),x(3)+delta*x(3),x(4),x(5),x(6))  -  f6(x(1),x(2),x(3),x(4),x(5),x(6)))/
(delta*x(3));
    df6dx4  =  (f6(x(1),x(2),x(3),x(4)+delta2*x(4),x(5),x(6))  -  f6(x(1),x(2),x(3),x(4),x(5),x(6)))/
(delta2*x(4));
    df6dx5  =  (f6(x(1),x(2),x(3),x(4),x(5)+delta2*x(5),x(6))  -  f6(x(1),x(2),x(3),x(4),x(5),x(6)))/
(delta2*x(5));
    df6dx6  =  (f6(x(1),x(2),x(3),x(4),x(5),x(6)+delta2*x(6))  -  f6(x(1),x(2),x(3),x(4),x(5),x(6)))/
(delta2*x(6));

    J = [df1dx1, df1dx2, df1dx3, df1dx4, df1dx5, df1dx6;
         df2dx1, df2dx2, df2dx3, df2dx4, df2dx5 ,df2dx6;
         df3dx1, df3dx2, df3dx3, df3dx4, df3dx5, df3dx6;
         df4dx1 ,df4dx2, df4dx3, df4dx4, df4dx5, df4dx6;
         df5dx1, df5dx2, df5dx3, df5dx4, df5dx5, df5dx6;
         df6dx1, df6dx2, df6dx3, df6dx4, df6dx5, df6dx6];

    f = [f1(x(1),x(2),x(3),x(4),x(5),x(6));
         f2(x(1),x(2),x(3),x(4),x(5),x(6));
         f3(x(1),x(2),x(3),x(4),x(5),x(6));
         f4(x(1),x(2),x(3),x(4),x(5),x(6));
         f5(x(1),x(2),x(3),x(4),x(5),x(6));
         f6(x(1),x(2),x(3),x(4),x(5),x(6))];


endfunction

function [x, f, ea, iter]=NewtonRaphson(f1, f2, f3, f4, f5, f6, x0, es, maxit, delta)

    iter = 0;
    x=x0;
```

```
    while(1)
        [J,f] = jacobiano(f1, f2, f3, f4, f5, f6, x, delta);
        xold = x;
        x = x - J\f;
        iter = iter+1;
        ea = 100*max(abs((x-xold) ./x));

        if iter >= maxit || ea <= es then
            break
        end
    end
endfunction

// ABAIXO ESTÃO AS 6 EQUAÇÕES PROVENIENTES DE:  $^1*F1 + $^2*F2 + $^3*F3 + $^4*F4 = $^*Peso
// ESTAS EQUAÇÕES VIERAM DO SISTEMA ALGÉBRICO CRIADO NO WXMAXIMA.
function f=f1(x1, x2, x3, x4, x5, x6)
global peso;
global F1; // ESTAS SÃO AS FORÇAS CALCULADAS NO INÍCIO DO PROGRAMA
global F2;
global F3;
global F4;
    f=(F1*(-0.052*cos(x5)*sin(x6)+0.0395*cos(x5)*cos(x6)-0.075*sin(x5)-x1-
0.382))/(sqrt((0.052*(cos(x6)*cos(x4)-
sin(x5)*sin(x6)*sin(x4))+0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))+0.075*cos(x5)*sin
(x4)-x2-0.382)^2+(0.0395*(sin(x6)*sin(x4)-
sin(x5)*cos(x6)*cos(x4))+0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))-
0.075*cos(x5)*cos(x4)-x3+0.75)^2+(-0.052*cos(x5)*sin(x6)+0.0395*cos(x5)*cos(x6)-
0.075*sin(x5)-x1-0.382)^2))+(F4*(-0.052*cos(x5)*sin(x6)-0.0395*cos(x5)*cos(x6)-
0.075*sin(x5)-x1+0.382))/(sqrt((0.052*(cos(x6)*cos(x4)-sin(x5)*sin(x6)*sin(x4))-
0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))+0.075*cos(x5)*sin(x4)-x2-0.382)^2+(-
0.0395*(sin(x6)*sin(x4)-
sin(x5)*cos(x6)*cos(x4))+0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))-
0.075*cos(x5)*cos(x4)-x3+0.75)^2+(-0.052*cos(x5)*sin(x6)-0.0395*cos(x5)*cos(x6)-
0.075*sin(x5)-x1+0.382)^2))+(F2*(0.052*cos(x5)*sin(x6)+0.0395*cos(x5)*cos(x6)-
0.075*sin(x5)-x1-0.382))/(sqrt((-0.052*(cos(x6)*cos(x4)-
sin(x5)*sin(x6)*sin(x4))+0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))+0.075*cos(x5)*sin
(x4)-x2+0.382)^2+(0.0395*(sin(x6)*sin(x4)-sin(x5)*cos(x6)*cos(x4))-
0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))-0.075*cos(x5)*cos(x4)-
x3+0.75)^2+(0.052*cos(x5)*sin(x6)+0.0395*cos(x5)*cos(x6)-0.075*sin(x5)-x1-
0.382)^2))+(F3*(0.052*cos(x5)*sin(x6)-0.0395*cos(x5)*cos(x6)-0.075*sin(x5)-
x1+0.382))/(sqrt((-0.052*(cos(x6)*cos(x4)-sin(x5)*sin(x6)*sin(x4))-
0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))+0.075*cos(x5)*sin(x4)-x2+0.382)^2+(-
0.0395*(sin(x6)*sin(x4)-sin(x5)*cos(x6)*cos(x4))-
0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))-0.075*cos(x5)*cos(x4)-
x3+0.75)^2+(0.052*cos(x5)*sin(x6)-0.0395*cos(x5)*cos(x6)-0.075*sin(x5)-x1+0.382)^2));
endfunction

function f=f2(x1, x2, x3, x4, x5, x6)
global peso;
global F1;
global F2;
```

```
global F3;
global F4;
   f=(F1*(0.052*(cos(x6)*cos(x4)-
sin(x5)*sin(x6)*sin(x4))+0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))+0.075*cos(x5)*sin
(x4)-x2-0.382))/(sqrt((0.052*(cos(x6)*cos(x4)-
sin(x5)*sin(x6)*sin(x4))+0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))+0.075*cos(x5)*sin
(x4)-x2-0.382)^2+(0.0395*(sin(x6)*sin(x4)-
sin(x5)*cos(x6)*cos(x4))+0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))-
0.075*cos(x5)*cos(x4)-x3+0.75)^2+(-0.052*cos(x5)*sin(x6)+0.0395*cos(x5)*cos(x6)-
0.075*sin(x5)-x1-0.382)^2))+(F4*(0.052*(cos(x6)*cos(x4)-sin(x5)*sin(x6)*sin(x4))-
0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))+0.075*cos(x5)*sin(x4)-x2-
0.382))/(sqrt((0.052*(cos(x6)*cos(x4)-sin(x5)*sin(x6)*sin(x4))-
0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))+0.075*cos(x5)*sin(x4)-x2-0.382)^2+(-
0.0395*(sin(x6)*sin(x4)-
sin(x5)*cos(x6)*cos(x4))+0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))-
0.075*cos(x5)*cos(x4)-x3+0.75)^2+(-0.052*cos(x5)*sin(x6)-0.0395*cos(x5)*cos(x6)-
0.075*sin(x5)-x1+0.382)^2))+(F2*(-0.052*(cos(x6)*cos(x4)-
sin(x5)*sin(x6)*sin(x4))+0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))+0.075*cos(x5)*sin
(x4)-x2+0.382))/(sqrt((-0.052*(cos(x6)*cos(x4)-
sin(x5)*sin(x6)*sin(x4))+0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))+0.075*cos(x5)*sin
(x4)-x2+0.382)^2+(0.0395*(sin(x6)*sin(x4)-sin(x5)*cos(x6)*cos(x4))-
0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))-0.075*cos(x5)*cos(x4)-
x3+0.75)^2+(0.052*cos(x5)*sin(x6)+0.0395*cos(x5)*cos(x6)-0.075*sin(x5)-x1-
0.382)^2))+(F3*(-0.052*(cos(x6)*cos(x4)-sin(x5)*sin(x6)*sin(x4))-
0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))+0.075*cos(x5)*sin(x4)-x2+0.382))/(sqrt((-
0.052*(cos(x6)*cos(x4)-sin(x5)*sin(x6)*sin(x4))-
0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))+0.075*cos(x5)*sin(x4)-x2+0.382)^2+(-
0.0395*(sin(x6)*sin(x4)-sin(x5)*cos(x6)*cos(x4))-
0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))-0.075*cos(x5)*cos(x4)-
x3+0.75)^2+(0.052*cos(x5)*sin(x6)-0.0395*cos(x5)*cos(x6)-0.075*sin(x5)-x1+0.382)^2));
endfunction

function f=f3(x1, x2, x3, x4, x5, x6)
global peso;
global F1;
global F2;
global F3;
global F4;
   f=(F1*(0.0395*(sin(x6)*sin(x4)-
sin(x5)*cos(x6)*cos(x4))+0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))-
0.075*cos(x5)*cos(x4)-x3+0.75))/(sqrt((0.052*(cos(x6)*cos(x4)-
sin(x5)*sin(x6)*sin(x4))+0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))+0.075*cos(x5)*sin
(x4)-x2-0.382)^2+(0.0395*(sin(x6)*sin(x4)-
sin(x5)*cos(x6)*cos(x4))+0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))-
0.075*cos(x5)*cos(x4)-x3+0.75)^2+(-0.052*cos(x5)*sin(x6)+0.0395*cos(x5)*cos(x6)-
0.075*sin(x5)-x1-0.382)^2))+(F4*(-0.0395*(sin(x6)*sin(x4)-
sin(x5)*cos(x6)*cos(x4))+0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))-
0.075*cos(x5)*cos(x4)-x3+0.75))/(sqrt((0.052*(cos(x6)*cos(x4)-sin(x5)*sin(x6)*sin(x4))-
0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))+0.075*cos(x5)*sin(x4)-x2-0.382)^2+(-
0.0395*(sin(x6)*sin(x4)-
sin(x5)*cos(x6)*cos(x4))+0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))-
0.075*cos(x5)*cos(x4)-x3+0.75)^2+(-0.052*cos(x5)*sin(x6)-0.0395*cos(x5)*cos(x6)-
0.075*sin(x5)-x1+0.382)^2))+(F2*(0.0395*(sin(x6)*sin(x4)-sin(x5)*cos(x6)*cos(x4))-
```

```
0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))-0.075*cos(x5)*cos(x4)-x3+0.75))/(sqrt((-
0.052*(cos(x6)*cos(x4)-
sin(x5)*sin(x6)*sin(x4))+0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))+0.075*cos(x5)*sin
(x4)-x2+0.382)^2+(0.0395*(sin(x6)*sin(x4)-sin(x5)*cos(x6)*cos(x4))-
0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))-0.075*cos(x5)*cos(x4)-
x3+0.75)^2+(0.052*cos(x5)*sin(x6)+0.0395*cos(x5)*cos(x6)-0.075*sin(x5)-x1-
0.382)^2))+(F3*(-0.0395*(sin(x6)*sin(x4)-sin(x5)*cos(x6)*cos(x4))-
0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))-0.075*cos(x5)*cos(x4)-x3+0.75))/(sqrt((-
0.052*(cos(x6)*cos(x4)-sin(x5)*sin(x6)*sin(x4))-
0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))+0.075*cos(x5)*sin(x4)-x2+0.382)^2+(-
0.0395*(sin(x6)*sin(x4)-sin(x5)*cos(x6)*cos(x4))-
0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))-0.075*cos(x5)*cos(x4)-
x3+0.75)^2+(0.052*cos(x5)*sin(x6)-0.0395*cos(x5)*cos(x6)-0.075*sin(x5)-
x1+0.382)^2))+peso;
endfunction

function f=f4(x1, x2, x3, x4, x5, x6)
global peso;
global F1;
global F2;
global F3;
global F4;
   f=F1*(((0.0395*(sin(x6)*sin(x4)-
sin(x5)*cos(x6)*cos(x4))+0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))-
0.075*cos(x5)*cos(x4)-x3+0.75)*(-0.052*(cos(x6)*cos(x4)-sin(x5)*sin(x6)*sin(x4))-
0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))-
0.075*cos(x5)*sin(x4)+x2))/(sqrt((0.052*(cos(x6)*cos(x4)-
sin(x5)*sin(x6)*sin(x4))+0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))+0.075*cos(x5)*sin
(x4)-x2-0.382)^2+(0.0395*(sin(x6)*sin(x4)-
sin(x5)*cos(x6)*cos(x4))+0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))-
0.075*cos(x5)*cos(x4)-x3+0.75)^2+(-0.052*cos(x5)*sin(x6)+0.0395*cos(x5)*cos(x6)-
0.075*sin(x5)-x1-0.382)^2))-((-0.0395*(sin(x6)*sin(x4)-sin(x5)*cos(x6)*cos(x4))-
0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))+0.075*cos(x5)*cos(x4)+x3)*(0.052*(cos(x6)
*cos(x4)-
sin(x5)*sin(x6)*sin(x4))+0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))+0.075*cos(x5)*sin
(x4)-x2-0.382))/(sqrt((0.052*(cos(x6)*cos(x4)-
sin(x5)*sin(x6)*sin(x4))+0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))+0.075*cos(x5)*sin
(x4)-x2-0.382)^2+(0.0395*(sin(x6)*sin(x4)-
sin(x5)*cos(x6)*cos(x4))+0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))-
0.075*cos(x5)*cos(x4)-x3+0.75)^2+(-0.052*cos(x5)*sin(x6)+0.0395*cos(x5)*cos(x6)-
0.075*sin(x5)-x1-0.382)^2)))+F4*(((-0.0395*(sin(x6)*sin(x4)-
sin(x5)*cos(x6)*cos(x4))+0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))-
0.075*cos(x5)*cos(x4)-x3+0.75)*(-0.052*(cos(x6)*cos(x4)-
sin(x5)*sin(x6)*sin(x4))+0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))-
0.075*cos(x5)*sin(x4)+x2))/(sqrt((0.052*(cos(x6)*cos(x4)-sin(x5)*sin(x6)*sin(x4))-
0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))+0.075*cos(x5)*sin(x4)-x2-0.382)^2+(-
0.0395*(sin(x6)*sin(x4)-
sin(x5)*cos(x6)*cos(x4))+0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))-
0.075*cos(x5)*cos(x4)-x3+0.75)^2+(-0.052*cos(x5)*sin(x6)-0.0395*cos(x5)*cos(x6)-
0.075*sin(x5)-x1+0.382)^2))-((0.0395*(sin(x6)*sin(x4)-sin(x5)*cos(x6)*cos(x4))-
0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))+0.075*cos(x5)*cos(x4)+x3)*(0.052*(cos(x6)
*cos(x4)-sin(x5)*sin(x6)*sin(x4))-
0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))+0.075*cos(x5)*sin(x4)-x2-
```

```
0.382))/(sqrt((0.052*(cos(x6)*cos(x4)-sin(x5)*sin(x6)*sin(x4))-
0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))+0.075*cos(x5)*sin(x4)-x2-0.382)^2+(-
0.0395*(sin(x6)*sin(x4)-
sin(x5)*cos(x6)*cos(x4))+0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))-
0.075*cos(x5)*cos(x4)-x3+0.75)^2+(-0.052*cos(x5)*sin(x6)-0.0395*cos(x5)*cos(x6)-
0.075*sin(x5)-x1+0.382)^2)))+F2*(((0.0395*(sin(x6)*sin(x4)-sin(x5)*cos(x6)*cos(x4))-
0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))-0.075*cos(x5)*cos(x4)-
x3+0.75)*(0.052*(cos(x6)*cos(x4)-sin(x5)*sin(x6)*sin(x4))-
0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))-0.075*cos(x5)*sin(x4)+x2))/(sqrt((-
0.052*(cos(x6)*cos(x4)-
sin(x5)*sin(x6)*sin(x4))+0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))+0.075*cos(x5)*sin
(x4)-x2+0.382)^2+(0.0395*(sin(x6)*sin(x4)-sin(x5)*cos(x6)*cos(x4))-
0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))-0.075*cos(x5)*cos(x4)-
x3+0.75)^2+(0.052*cos(x5)*sin(x6)+0.0395*cos(x5)*cos(x6)-0.075*sin(x5)-x1-0.382)^2))-((-
0.0395*(sin(x6)*sin(x4)-
sin(x5)*cos(x6)*cos(x4))+0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))+0.075*cos(x5)*co
s(x4)+x3)*(-0.052*(cos(x6)*cos(x4)-
sin(x5)*sin(x6)*sin(x4))+0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))+0.075*cos(x5)*sin
(x4)-x2+0.382))/(sqrt((-0.052*(cos(x6)*cos(x4)-
sin(x5)*sin(x6)*sin(x4))+0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))+0.075*cos(x5)*sin
(x4)-x2+0.382)^2+(0.0395*(sin(x6)*sin(x4)-sin(x5)*cos(x6)*cos(x4))-
0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))-0.075*cos(x5)*cos(x4)-
x3+0.75)^2+(0.052*cos(x5)*sin(x6)+0.0395*cos(x5)*cos(x6)-0.075*sin(x5)-x1-
0.382)^2)))+F3*(((-0.0395*(sin(x6)*sin(x4)-sin(x5)*cos(x6)*cos(x4))-
0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))-0.075*cos(x5)*cos(x4)-
x3+0.75)*(0.052*(cos(x6)*cos(x4)-
sin(x5)*sin(x6)*sin(x4))+0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))-
0.075*cos(x5)*sin(x4)+x2))/(sqrt((-0.052*(cos(x6)*cos(x4)-sin(x5)*sin(x6)*sin(x4))-
0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))+0.075*cos(x5)*sin(x4)-x2+0.382)^2+(-
0.0395*(sin(x6)*sin(x4)-sin(x5)*cos(x6)*cos(x4))-
0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))-0.075*cos(x5)*cos(x4)-
x3+0.75)^2+(0.052*cos(x5)*sin(x6)-0.0395*cos(x5)*cos(x6)-0.075*sin(x5)-x1+0.382)^2))-
((0.0395*(sin(x6)*sin(x4)-
sin(x5)*cos(x6)*cos(x4))+0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))+0.075*cos(x5)*co
s(x4)+x3)*(-0.052*(cos(x6)*cos(x4)-sin(x5)*sin(x6)*sin(x4))-
0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))+0.075*cos(x5)*sin(x4)-x2+0.382))/(sqrt((-
0.052*(cos(x6)*cos(x4)-sin(x5)*sin(x6)*sin(x4))-
0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))+0.075*cos(x5)*sin(x4)-x2+0.382)^2+(-
0.0395*(sin(x6)*sin(x4)-sin(x5)*cos(x6)*cos(x4))-
0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))-0.075*cos(x5)*cos(x4)-
x3+0.75)^2+(0.052*cos(x5)*sin(x6)-0.0395*cos(x5)*cos(x6)-0.075*sin(x5)-
x1+0.382)^2)))+x2*peso;
endfunction

function f=f5(x1, x2, x3, x4, x5, x6)
global peso;
global F1;
global F2;
global F3;
global F4;
   f=F1*(((-0.052*cos(x5)*sin(x6)+0.0395*cos(x5)*cos(x6)-0.075*sin(x5)-x1-0.382)*(-
0.0395*(sin(x6)*sin(x4)-sin(x5)*cos(x6)*cos(x4))-
0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))+0.075*cos(x5)*cos(x4)+x3))/(sqrt((0.052*(c
```

```
os(x6)*cos(x4)-
sin(x5)*sin(x6)*sin(x4))+0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))+0.075*cos(x5)*sin
(x4)-x2-0.382)^2+(0.0395*(sin(x6)*sin(x4)-
sin(x5)*cos(x6)*cos(x4))+0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))-
0.075*cos(x5)*cos(x4)-x3+0.75)^2+(-0.052*cos(x5)*sin(x6)+0.0395*cos(x5)*cos(x6)-
0.075*sin(x5)-x1-0.382)^2))-((0.052*cos(x5)*sin(x6)-
0.0395*cos(x5)*cos(x6)+0.075*sin(x5)+x1)*(0.0395*(sin(x6)*sin(x4)-
sin(x5)*cos(x6)*cos(x4))+0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))-
0.075*cos(x5)*cos(x4)-x3+0.75))/(sqrt((0.052*(cos(x6)*cos(x4)-
sin(x5)*sin(x6)*sin(x4))+0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))+0.075*cos(x5)*sin
(x4)-x2-0.382)^2+(0.0395*(sin(x6)*sin(x4)-
sin(x5)*cos(x6)*cos(x4))+0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))-
0.075*cos(x5)*cos(x4)-x3+0.75)^2+(-0.052*cos(x5)*sin(x6)+0.0395*cos(x5)*cos(x6)-
0.075*sin(x5)-x1-0.382)^2)))+F4*(((-0.052*cos(x5)*sin(x6)-0.0395*cos(x5)*cos(x6)-
0.075*sin(x5)-x1+0.382)*(0.0395*(sin(x6)*sin(x4)-sin(x5)*cos(x6)*cos(x4))-
0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))+0.075*cos(x5)*cos(x4)+x3))/(sqrt((0.052*(c
os(x6)*cos(x4)-sin(x5)*sin(x6)*sin(x4))-
0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))+0.075*cos(x5)*sin(x4)-x2-0.382)^2+(-
0.0395*(sin(x6)*sin(x4)-
sin(x5)*cos(x6)*cos(x4))+0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))-
0.075*cos(x5)*cos(x4)-x3+0.75)^2+(-0.052*cos(x5)*sin(x6)-0.0395*cos(x5)*cos(x6)-
0.075*sin(x5)-x1+0.382)^2))-
((0.052*cos(x5)*sin(x6)+0.0395*cos(x5)*cos(x6)+0.075*sin(x5)+x1)*(-
0.0395*(sin(x6)*sin(x4)-
sin(x5)*cos(x6)*cos(x4))+0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))-
0.075*cos(x5)*cos(x4)-x3+0.75))/(sqrt((0.052*(cos(x6)*cos(x4)-sin(x5)*sin(x6)*sin(x4))-
0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))+0.075*cos(x5)*sin(x4)-x2-0.382)^2+(-
0.0395*(sin(x6)*sin(x4)-
sin(x5)*cos(x6)*cos(x4))+0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))-
0.075*cos(x5)*cos(x4)-x3+0.75)^2+(-0.052*cos(x5)*sin(x6)-0.0395*cos(x5)*cos(x6)-
0.075*sin(x5)-x1+0.382)^2)))+F2*(((0.052*cos(x5)*sin(x6)+0.0395*cos(x5)*cos(x6)-
0.075*sin(x5)-x1-0.382)*(-0.0395*(sin(x6)*sin(x4)-
sin(x5)*cos(x6)*cos(x4))+0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))+0.075*cos(x5)*co
s(x4)+x3))/(sqrt((-0.052*(cos(x6)*cos(x4)-
sin(x5)*sin(x6)*sin(x4))+0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))+0.075*cos(x5)*sin
(x4)-x2+0.382)^2+(0.0395*(sin(x6)*sin(x4)-sin(x5)*cos(x6)*cos(x4))-
0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))-0.075*cos(x5)*cos(x4)-
x3+0.75)^2+(0.052*cos(x5)*sin(x6)+0.0395*cos(x5)*cos(x6)-0.075*sin(x5)-x1-0.382)^2))-((-
0.052*cos(x5)*sin(x6)-0.0395*cos(x5)*cos(x6)+0.075*sin(x5)+x1)*(0.0395*(sin(x6)*sin(x4)-
sin(x5)*cos(x6)*cos(x4))-0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))-
0.075*cos(x5)*cos(x4)-x3+0.75))/(sqrt((-0.052*(cos(x6)*cos(x4)-
sin(x5)*sin(x6)*sin(x4))+0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))+0.075*cos(x5)*sin
(x4)-x2+0.382)^2+(0.0395*(sin(x6)*sin(x4)-sin(x5)*cos(x6)*cos(x4))-
0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))-0.075*cos(x5)*cos(x4)-
x3+0.75)^2+(0.052*cos(x5)*sin(x6)+0.0395*cos(x5)*cos(x6)-0.075*sin(x5)-x1-
0.382)^2)))+F3*(((0.052*cos(x5)*sin(x6)-0.0395*cos(x5)*cos(x6)-0.075*sin(x5)-
x1+0.382)*(0.0395*(sin(x6)*sin(x4)-
sin(x5)*cos(x6)*cos(x4))+0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))+0.075*cos(x5)*co
s(x4)+x3))/(sqrt((-0.052*(cos(x6)*cos(x4)-sin(x5)*sin(x6)*sin(x4))-
0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))+0.075*cos(x5)*sin(x4)-x2+0.382)^2+(-
0.0395*(sin(x6)*sin(x4)-sin(x5)*cos(x6)*cos(x4))-
0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))-0.075*cos(x5)*cos(x4)-
x3+0.75)^2+(0.052*cos(x5)*sin(x6)-0.0395*cos(x5)*cos(x6)-0.075*sin(x5)-x1+0.382)^2))-((-
```

```
0.052*cos(x5)*sin(x6)+0.0395*cos(x5)*cos(x6)+0.075*sin(x5)+x1)*(-0.0395*(sin(x6)*sin(x4)-
sin(x5)*cos(x6)*cos(x4))-0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))-
0.075*cos(x5)*cos(x4)-x3+0.75))/(sqrt((-0.052*(cos(x6)*cos(x4)-sin(x5)*sin(x6)*sin(x4))-
0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))+0.075*cos(x5)*sin(x4)-x2+0.382)^2+(-
0.0395*(sin(x6)*sin(x4)-sin(x5)*cos(x6)*cos(x4))-
0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))-0.075*cos(x5)*cos(x4)-
x3+0.75)^2+(0.052*cos(x5)*sin(x6)-0.0395*cos(x5)*cos(x6)-0.075*sin(x5)-x1+0.382)^2)))-
x1*peso;
endfunction

function f=f6(x1, x2, x3, x4, x5, x6)
global peso;
global F1;
global F2;
global F3;
global F4;
   f=F1*((((0.052*cos(x5)*sin(x6)-
0.0395*cos(x5)*cos(x6)+0.075*sin(x5)+x1)*(0.052*(cos(x6)*cos(x4)-
sin(x5)*sin(x6)*sin(x4))+0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))+0.075*cos(x5)*sin
(x4)-x2-0.382))/(sqrt((0.052*(cos(x6)*cos(x4)-
sin(x5)*sin(x6)*sin(x4))+0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))+0.075*cos(x5)*sin
(x4)-x2-0.382)^2+(0.0395*(sin(x6)*sin(x4)-
sin(x5)*cos(x6)*cos(x4))+0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))-
0.075*cos(x5)*cos(x4)-x3+0.75)^2+(-0.052*cos(x5)*sin(x6)+0.0395*cos(x5)*cos(x6)-
0.075*sin(x5)-x1-0.382)^2))-((-0.052*cos(x5)*sin(x6)+0.0395*cos(x5)*cos(x6)-0.075*sin(x5)-
x1-0.382)*(-0.052*(cos(x6)*cos(x4)-sin(x5)*sin(x6)*sin(x4))-
0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))-
0.075*cos(x5)*sin(x4)+x2))/(sqrt((0.052*(cos(x6)*cos(x4)-
sin(x5)*sin(x6)*sin(x4))+0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))+0.075*cos(x5)*sin
(x4)-x2-0.382)^2+(0.0395*(sin(x6)*sin(x4)-
sin(x5)*cos(x6)*cos(x4))+0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))-
0.075*cos(x5)*cos(x4)-x3+0.75)^2+(-0.052*cos(x5)*sin(x6)+0.0395*cos(x5)*cos(x6)-
0.075*sin(x5)-x1-
0.382)^2)))+F4*((((0.052*cos(x5)*sin(x6)+0.0395*cos(x5)*cos(x6)+0.075*sin(x5)+x1)*(0.052*
(cos(x6)*cos(x4)-sin(x5)*sin(x6)*sin(x4))-
0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))+0.075*cos(x5)*sin(x4)-x2-
0.382))/(sqrt((0.052*(cos(x6)*cos(x4)-sin(x5)*sin(x6)*sin(x4))-
0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))+0.075*cos(x5)*sin(x4)-x2-0.382)^2+(-
0.0395*(sin(x6)*sin(x4)-
sin(x5)*cos(x6)*cos(x4))+0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))-
0.075*cos(x5)*cos(x4)-x3+0.75)^2+(-0.052*cos(x5)*sin(x6)-0.0395*cos(x5)*cos(x6)-
0.075*sin(x5)-x1+0.382)^2))-((-0.052*cos(x5)*sin(x6)-0.0395*cos(x5)*cos(x6)-0.075*sin(x5)-
x1+0.382)*(-0.052*(cos(x6)*cos(x4)-
sin(x5)*sin(x6)*sin(x4))+0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))-
0.075*cos(x5)*sin(x4)+x2))/(sqrt((0.052*(cos(x6)*cos(x4)-sin(x5)*sin(x6)*sin(x4))-
0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))+0.075*cos(x5)*sin(x4)-x2-0.382)^2+(-
0.0395*(sin(x6)*sin(x4)-
sin(x5)*cos(x6)*cos(x4))+0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))-
0.075*cos(x5)*cos(x4)-x3+0.75)^2+(-0.052*cos(x5)*sin(x6)-0.0395*cos(x5)*cos(x6)-
0.075*sin(x5)-x1+0.382)^2)))+F2*(((-0.052*cos(x5)*sin(x6)-
0.0395*cos(x5)*cos(x6)+0.075*sin(x5)+x1)*(-0.052*(cos(x6)*cos(x4)-
sin(x5)*sin(x6)*sin(x4))+0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))+0.075*cos(x5)*sin
(x4)-x2+0.382))/(sqrt((-0.052*(cos(x6)*cos(x4)-
```

```
sin(x5)*sin(x6)*sin(x4))+0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))+0.075*cos(x5)*sin
(x4)-x2+0.382)^2+(0.0395*(sin(x6)*sin(x4)-sin(x5)*cos(x6)*cos(x4))-
0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))-0.075*cos(x5)*cos(x4)-
x3+0.75)^2+(0.052*cos(x5)*sin(x6)+0.0395*cos(x5)*cos(x6)-0.075*sin(x5)-x1-0.382)^2))-
((0.052*cos(x5)*sin(x6)+0.0395*cos(x5)*cos(x6)-0.075*sin(x5)-x1-
0.382)*(0.052*(cos(x6)*cos(x4)-sin(x5)*sin(x6)*sin(x4))-
0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))-0.075*cos(x5)*sin(x4)+x2))/(sqrt((-
0.052*(cos(x6)*cos(x4)-
sin(x5)*sin(x6)*sin(x4))+0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))+0.075*cos(x5)*sin
(x4)-x2+0.382)^2+(0.0395*(sin(x6)*sin(x4)-sin(x5)*cos(x6)*cos(x4))-
0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))-0.075*cos(x5)*cos(x4)-
x3+0.75)^2+(0.052*cos(x5)*sin(x6)+0.0395*cos(x5)*cos(x6)-0.075*sin(x5)-x1-
0.382)^2)))+F3*(((-0.052*cos(x5)*sin(x6)+0.0395*cos(x5)*cos(x6)+0.075*sin(x5)+x1)*(-
0.052*(cos(x6)*cos(x4)-sin(x5)*sin(x6)*sin(x4))-
0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))+0.075*cos(x5)*sin(x4)-x2+0.382))/(sqrt((-
0.052*(cos(x6)*cos(x4)-sin(x5)*sin(x6)*sin(x4))-
0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))+0.075*cos(x5)*sin(x4)-x2+0.382)^2+(-
0.0395*(sin(x6)*sin(x4)-sin(x5)*cos(x6)*cos(x4))-
0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))-0.075*cos(x5)*cos(x4)-
x3+0.75)^2+(0.052*cos(x5)*sin(x6)-0.0395*cos(x5)*cos(x6)-0.075*sin(x5)-x1+0.382)^2))-
((0.052*cos(x5)*sin(x6)-0.0395*cos(x5)*cos(x6)-0.075*sin(x5)-
x1+0.382)*(0.052*(cos(x6)*cos(x4)-
sin(x5)*sin(x6)*sin(x4))+0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))-
0.075*cos(x5)*sin(x4)+x2))/(sqrt((-0.052*(cos(x6)*cos(x4)-sin(x5)*sin(x6)*sin(x4))-
0.0395*(sin(x5)*cos(x6)*sin(x4)+sin(x6)*cos(x4))+0.075*cos(x5)*sin(x4)-x2+0.382)^2+(-
0.0395*(sin(x6)*sin(x4)-sin(x5)*cos(x6)*cos(x4))-
0.052*(cos(x6)*sin(x4)+sin(x5)*sin(x6)*cos(x4))-0.075*cos(x5)*cos(x4)-
x3+0.75)^2+(0.052*cos(x5)*sin(x6)-0.0395*cos(x5)*cos(x6)-0.075*sin(x5)-x1+0.382)^2)));
endfunction

// APENAS PARA PROVAR QUE DADAS AS FORÇAS O MÉTODO DE NEWTON-RAPHSON RETORNA OS VALORES DE PX, PY, PZ, Theta, Phi e Psi
// DIGITADOS NO INÍCIO DO PROGRAMA.
// O CHUTE INICIAL É A ÚLTIMA POSIÇÃO DA PLATAFORMA, CASO CONTRÁRIO O PROGRAMA ENCONTRA OUTRAS SOLUÇÕES PARA AS
// EQUAÇÕES NÃO LINEARES.
x0 = [PX+0.001;PY+0.001;PZ+0.001;theta+0.001;phi+0.001;psi+0.001];
es =    0.001;
maxit = 1000;
delta = 0.00000000000000001;

[x, f, ea, iter] = NewtonRaphson(f1, f2, f3, f4, f5, f6, x0, es, maxit, delta);

disp("--------------------------------------------------------------------------------");
disp("CÁLCULO DE PX, PY, PZ, Theta, Phi e Psi PELAS EQUAÇÕES NÃO LINEARES:");
disp("PX DADO = "+ string(PX*1000) + " mm    PX CALCULADO = "+string(x(1)*1000)+" mm");
disp("PY DADO = "+ string(PY*1000) + " mm    PY CALCULADO = "+string(x(2)*1000)+" mm");
disp("PZ DADO = "+ string(PZ*1000) + " mm    PZ CALCULADO = "+string(x(3)*1000)+" mm");
```

```
disp("Theta dado = "+ string(theta*180/%pi)+"°    Theta calculado = "+string(x(4)*180/%pi)+"
°");
disp("Phi dado = "+ string(phi*180/%pi)+"°   Phi calculado = "+string(x(5)*180/%pi)+" °");
disp("Psi dado = "+ string(psi*180/%pi)+"°   Psi calculado = "+string(x(6)*180/%pi)+" °");

disp("FIM");
tempo_final=getdate();
disp(tempo_final);
```

APÊNDICE 4 - PROGRAMA DO MICROCONTROLADOR QUE GERENCIA O RECEBIMENTO DOS DADOS E CONTROLA OS MOTORES

```
// ESTE PROGRAMA ESTÁ DISPONIBILIZADO NO LINK DO GOOGLE DRIVE ABAIXO:
//    https://drive.google.com/open?id=0B9jot9RnR3XiSkFBNFZEWlBWT2s

// Inclusão das bibliotecas:
#include <SoftwareSerial.h>
#include <AccelStepper.h>

// Modo de operação do motor
#define FULL4WIRE 4

int Pos1,Pos2,Pos3,Pos4=0; // Variáveis para cálculo do número de passos
int M1,M2,M3,M4;          // Variáveis para movimentar os motores 1, 2, 3 e 4.
int aX,aY=0;              // Variáveis para armazenamento dos potenciômetros do Joystick

// Variáveis que recebem as forças inseridas pelo usuário:
int F1_VALUE,F2_VALUE,F3_VALUE,F4_VALUE=0;

// Variáveis para ativar e desativar opções para leitura na tela e envio dos dados
// provenientes da janela Serial do computador:
short int ler,manual,UP_DOWN,FORCES;

// Variável loo é usada para ativar ou desativar o envio dos dados para a janela Serial.
// Variável YAW e YAW0 são usadas para tornar a primeira leitura de rotação no eixo Z do
// giroscópio igual a zero e as demais leituras serão baseadas na nova posição zero.
short int loo,YAW,YAW0;

// Variável para verificar se algum potenciômetro não está respondendo,
// caso o potenciômetro tenha acusado algum valor e depois seu sinal
// permaneceu em zero então o microcontrolador lerá novamente a porta
// serial para nova leitura.
short int verifica=0;

// DEFINIÇAO DOS PINOS DOS MOTORES
#define motor1Pin1  30    // IN1 do driver ULN2003 1
#define motor1Pin2  31    // IN2 do driver ULN2003 1
#define motor1Pin3  32    // IN3 do driver ULN2003 1
#define motor1Pin4  33    // IN4 do driver ULN2003 1

#define motor2Pin1  34    // IN1 do driver ULN2003 2
#define motor2Pin2  35    // IN2 do driver ULN2003 2
#define motor2Pin3  36    // IN3 do driver ULN2003 2
```

```
#define motor2Pin4  37     // IN4 do driver ULN2003 2

#define motor3Pin1  38     // IN1 do driver ULN2003 3
#define motor3Pin2  39     // IN2 do driver ULN2003 3
#define motor3Pin3  40     // IN3 do driver ULN2003 3
#define motor3Pin4  41     // IN4 do driver ULN2003 3

#define motor4Pin1  42     // IN1 do driver ULN2003 4
#define motor4Pin2  43     // IN2 do driver ULN2003 4
#define motor4Pin3  44     // IN3 do driver ULN2003 4
#define motor4Pin4  45     // IN4 do driver ULN2003 4

// A biblioteca AccelStepper deve inicializar as bobinas do motor na sequência
// IN1-IN3-IN2-IN4 quando usado o motor de passo modelo 28BYJ-48:
AccelStepper MOTOR_1(FULL4WIRE, motor1Pin1, motor1Pin3, motor1Pin2, motor1Pin4);
AccelStepper MOTOR_2(FULL4WIRE, motor2Pin1, motor2Pin3, motor2Pin2, motor2Pin4);
AccelStepper MOTOR_3(FULL4WIRE, motor3Pin1, motor3Pin3, motor3Pin2, motor3Pin4);
AccelStepper MOTOR_4(FULL4WIRE, motor4Pin1, motor4Pin3, motor4Pin2, motor4Pin4);

// pinos RX=10 e TX=11, onde RX é ligado ao TX do módulo ESP receptor ou no TX do
// microcontrolador que envia os dados dos sensores.
SoftwareSerial MEGA(10, 11);

// Variáveis usadas para recebimento dos dados quando usado o módulo ESP receptor.
uint8_t c1=0;
uint8_t c2=0;
uint8_t c3=0;
uint8_t c4=0;
uint8_t y_1=0;
uint8_t y2=0;
uint8_t p1=0;
uint8_t p2=0;
uint8_t r1=0;
uint8_t r2=0;
uint8_t sy=0;
uint8_t sp=0;
uint8_t sr=0;

// Variáveis que guardam o valor das orientações do giroscópio:
short int Y,P,R,Y_max,P_max,R_max=0;

// Variáveis para cálculo do deslocamento das molas:
float Xmola1,Xmola2,Xmola3,Xmola4=0;

// Variáveis para cálculo das forças:
int F1,F2,F3,F4=0;
```

```
// Função para leitura dos dados recebidos pela porta Serial RX (pino 10):
void SENSORS()
{
  //
  while (MEGA.available())
  {  // Primeiramente faz um leitura para descartar os dados acumulados na porta Serial.
    String B = MEGA.readStringUntil('|');
  }
  // Aguarda um breve tempo para que os dados atualizados cheguem.
  delay(150);

  // Lê os novos dados da porta Serial até encontrar o símbolo de |
  // e acende o LED do microcontrolador.
  while (MEGA.available())
   {
    digitalWrite(13, HIGH);
    String inData = MEGA.readStringUntil('|');

   // Se os caracteres "C1=" são recebidos então lê a partir do terceiro
   // caractere e multiplica por 0,37 mm que é o valor testado na prática,
   // ou seja, quando a mola extende 0,37 mm o potenciômetro acusa um movimento.
   // O mesmo para c2, c3 e c4:
   if (inData.indexOf("C1=")==0)
   {
     c1 = (uint8_t)inData.substring(3).toInt();
     Xmola1 = c1*0.37;

   }
   if (inData.indexOf("C2=")==0)
   {
     c2 = (uint8_t)inData.substring(3).toInt();
     Xmola2 = c2*0.37;
   }
   if (inData.indexOf("C3=")==0)
   {
     c3 = (uint8_t)inData.substring(3).toInt();
     Xmola3 = c3*0.37;
   }
   if (inData.indexOf("C4=")==0)
   {
     c4 = (uint8_t)inData.substring(3).toInt();
     Xmola4 = c4*0.37;
   }

   // Compara se os caracteres são referentes aos ângulos do sensor
```

```
// giroscópico:
if (inData.indexOf("Y1=")==0)
  {
    y_1 = (uint8_t)inData.substring(3).toInt();

  }
  if (inData.indexOf("Y2=")==0)
  {
    y2 = (uint8_t)inData.substring(3).toInt();

  }
  if (inData.indexOf("P1=")==0)
  {
    r1 = (uint8_t)inData.substring(3).toInt();

  }
  if (inData.indexOf("P2=")==0)
  {
    r2 = (uint8_t)inData.substring(3).toInt();

  }
  if (inData.indexOf("R1=")==0)
  {
    p1 = (uint8_t)inData.substring(3).toInt();

  }
  if (inData.indexOf("R2=")==0)
  {
    p2 = (uint8_t)inData.substring(3).toInt();

  }

  // As comparações a seguir verifica se os ângulos são positivos ou não:
  if (inData.indexOf("SY=")==0)
  {
    sy = (uint8_t)inData.substring(3).toInt();

  }
  if (inData.indexOf("SP=")==0)
  {
    sr = (uint8_t)inData.substring(3).toInt();

  }
  if (inData.indexOf("SR=")==0)
  {
    sp = (uint8_t)inData.substring(3).toInt();
```

```
    }
}
// Após receber os dados apaga o LED.
digitalWrite(13, LOW);

// Remonta os valores dos ângulos:
Y = (y2<<8)|y_1;
P = (p2<<8)|p1;
R = (r2<<8)|r1;

// Convenção de giro positivo ou negativo:
if (sy==0)
{
 Y=Y*(-1);
}
if (YAW0==0 && Y!=0)
{
 YAW = -Y;
 YAW0=1;
}
// Giro relativo em Z considerando o primeiro valor
// lido como o ponto zero de partida:
Y = Y + YAW;

if (sp==1)
{
 P=P*(-1);
}
if (sr==1)
{
 R=R*(-1);
}


if (Xmola1==0)
{
  F1=0;
}
else
{
  // Força no cabo 1 [x10^-2 N]
  // F - 1.4715 = K*X
  // K = 196   Pré-carga = 1.4715 N
  F1 = round(((196 * (Xmola1/1000))+1.4715)*100);
}
```

```
  if (Xmola2==0)
  {
    F2=0;
  }
  else
  {
    F2 = round(((196 * (Xmola2/1000))+1.4715)*100);
  }

  if (Xmola3==0)
  {
    F3=0;
  }
  else
  {
    F3 = round(((196 * (Xmola3/1000))+1.4715)*100);
  }

  if (Xmola4==0)
  {
    F4=0;
  }
  else
  {
    F4 = round(((196 * (Xmola4/1000))+1.4715)*100);
  }

}
// Esta função é chamada quando o usuário informar as forças desejadas nos cabos.
// A tolerância de 8 é usada devido a amplitude da mola (0.37, 0.74, 1.11 ...)
void VERIFICAR_FORCAS()
{
  while (F1<(F1_VALUE-8) || F2<(F2_VALUE-8) || F3<(F3_VALUE-8) || F4<(F4_VALUE-8) ||
F1>(F1_VALUE+8) || F2>(F2_VALUE+8) || F3>(F3_VALUE+8) || F4>(F4_VALUE+8))
  {
        // Se algum potenciômetro não está enviando o valor então repete a leitura:
        if (verifica==1)
        {
          while (F1==0 || F2==0 || F3==0 || F4==0)
          {
            SENSORS();
          }

        }
        else
```

```
{
  SENSORS();
}

// Mostra na janela Serial os valores das forças:
Serial.println("F [x10^-2 N]");
Serial.println("F1="+String(F1));
Serial.println("F2="+String(F2));
Serial.println("F3="+String(F3));
Serial.println("F4="+String(F4));
Serial.println(" ");
if (F1<F1_VALUE-8)
{
    // Para cada 2048 passos o cabo move-se 100 mm. Neste trecho do código o motor
    // irá girar para o cabo mover-se uma distância da mola equivalente entre a força
    // digitada e a força medida pelo sensor.

    Pos1=-200;
    MOTOR_1.move(Pos1);

}
if (F1>F1_VALUE+8)
{
  Pos1=round(((((((F1-F1_VALUE)*0.01)/196)*1000) * 2048) / 100));
  MOTOR_1.move(Pos1);
}

if (F2<F2_VALUE-8)
{
  Pos2=-200;
  MOTOR_2.move(Pos2);
}
if (F2>F2_VALUE+8)
{
  Pos2=round(((((((F2-F2_VALUE)*0.01)/196)*1000) * 2048) / 100));
  MOTOR_2.move(Pos2);
}

if (F3<F3_VALUE-8)
{
  Pos3=-200;
  MOTOR_3.move(Pos3);
}
if (F3>F3_VALUE+8)
{
  Pos3=round(((((((F3-F3_VALUE)*0.01)/196)*1000) * 2048) / 100));
```

```
          MOTOR_3.move(Pos3);
        }
        if (F4<F4_VALUE-8)
        {
          Pos4=-200;
          MOTOR_4.move(Pos4);
        }
        if (F4>F4_VALUE+8)
        {
          Pos4=round(((((((F4-F4_VALUE)*0.01)/196)*1000) * 2048) / 100));
          MOTOR_4.move(Pos4);
        }

        MOTOR_1.setSpeed(300);
        MOTOR_2.setSpeed(300);
        MOTOR_3.setSpeed(300);
        MOTOR_4.setSpeed(300);

        if (Pos1<0){MOTOR_1.setSpeed(-300);}
        if (Pos2<0){MOTOR_2.setSpeed(-300);}
        if (Pos3<0){MOTOR_3.setSpeed(-300);}
        if (Pos4<0){MOTOR_4.setSpeed(-300);}

        while (MOTOR_1.distanceToGo()!=0 || MOTOR_2.distanceToGo()!=0 ||
MOTOR_3.distanceToGo()!=0 || MOTOR_4.distanceToGo()!=0)
        {

          MOTOR_1.run();
          MOTOR_2.run();
          MOTOR_3.run();
          MOTOR_4.run();
        }
  }
}

void setup()
{
   // Pinagem do LED:
   pinMode(13, OUTPUT);

   // O valor zero é para não mostrar os resultados na janela Serial:
   ler=0;

   // Pinagem para o joystick:
   pinMode(0,INPUT);
   pinMode(1,INPUT);
```

```
  // Inicia as portas Seriais:
  MEGA.begin(9600);
  Serial.begin(9600);

  MOTOR_1.setMaxSpeed(300);       // passos por segundo
  MOTOR_1.setAcceleration(150);   // passos por segundo ao quadrado.
  Pos1=MOTOR_1.currentPosition();

  MOTOR_2.setMaxSpeed(300);
  MOTOR_2.setAcceleration(150);
  Pos2=MOTOR_2.currentPosition();

  MOTOR_3.setMaxSpeed(300);
  MOTOR_3.setAcceleration(150);
  Pos3=MOTOR_3.currentPosition();

  MOTOR_4.setMaxSpeed(300);
  MOTOR_4.setAcceleration(150);
  Pos4=MOTOR_4.currentPosition();

  // Inicia o microcontrolador com o Joystick ativo:
  manual=1;
  UP_DOWN=1;

}

void loop()
{
 // A estrutura While abaixo verifica os comandos que estão sendo enviados pelo
 // usuário via janela Serial:
 while(Serial.available())
 {
   String dados = Serial.readStringUntil('|');

   if (dados.indexOf("LER")==0)
   {
     // Usado para o microcontrolador enviar uma única vez os dados dos sensores:
     ler=1;
   }
   if (dados.indexOf("LOOP_ON")==0)
   {
     // Usado para os dados aparecerem na janela constantemente:
     loo=1;
   }
   if (dados.indexOf("LOOP_OFF")==0)
```

```
{
  loo=0;
}
if (dados.indexOf("FORCES")==0)
{
  // Usado para o microcontrolador receber o valor das forças
  // digitadas pelo usuário:
  FORCES=1;
  manual=0;
  UP_DOWN=0;
}

// As forças enviadas devem estar em [x10^-2 N]
// Exemplo: FORCES|F1=183|F2=249|F3=212|F4=292|
if (dados.indexOf("F1=")==0)
{
 F1_VALUE = dados.substring(3).toInt();
}
if (dados.indexOf("F2=")==0)
{
 F2_VALUE = dados.substring(3).toInt();
}
if (dados.indexOf("F3=")==0)
{
 F3_VALUE = dados.substring(3).toInt();
}
if (dados.indexOf("F4=")==0)
{
 F4_VALUE = dados.substring(3).toInt();
}
if (dados.indexOf("PARAR")==0)
{
  ler=0;

  MOTOR_1.setCurrentPosition(0);
  MOTOR_2.setCurrentPosition(0);
  MOTOR_3.setCurrentPosition(0);
  MOTOR_4.setCurrentPosition(0);

  Pos1=MOTOR_1.currentPosition();
  MOTOR_1.stop();

  Pos2=MOTOR_2.currentPosition();
  MOTOR_2.stop();

  Pos3=MOTOR_3.currentPosition();
```

```
  MOTOR_3.stop();

  Pos4=MOTOR_4.currentPosition();
  MOTOR_4.stop();
}
if (dados.indexOf("MANUAL")==0)
{
  manual=1;

  MOTOR_1.setCurrentPosition(0);
  MOTOR_2.setCurrentPosition(0);
  MOTOR_3.setCurrentPosition(0);
  MOTOR_4.setCurrentPosition(0);

  Pos1=MOTOR_1.currentPosition();
  MOTOR_1.stop();

  Pos2=MOTOR_2.currentPosition();
  MOTOR_2.stop();

  Pos3=MOTOR_3.currentPosition();
  MOTOR_3.stop();

  Pos4=MOTOR_4.currentPosition();
  MOTOR_4.stop();
}

if (dados.indexOf("AUTO")==0)
{

  MOTOR_1.setCurrentPosition(0);
  MOTOR_2.setCurrentPosition(0);
  MOTOR_3.setCurrentPosition(0);
  MOTOR_4.setCurrentPosition(0);

  Pos1=MOTOR_1.currentPosition();
  MOTOR_1.stop();

  Pos2=MOTOR_2.currentPosition();
  MOTOR_2.stop();

  Pos3=MOTOR_3.currentPosition();
  MOTOR_3.stop();

  Pos4=MOTOR_4.currentPosition();
  MOTOR_4.stop();
```

```
   manual=0;
   UP_DOWN=0;

}

// Usado para que o Joystick levante e abaixe a plataforma:
if (dados.indexOf("UP_ON")==0)
{
   MOTOR_1.setCurrentPosition(0);
   MOTOR_2.setCurrentPosition(0);
   MOTOR_3.setCurrentPosition(0);
   MOTOR_4.setCurrentPosition(0);

   Pos1=MOTOR_1.currentPosition();
   MOTOR_1.stop();

   Pos2=MOTOR_2.currentPosition();
   MOTOR_2.stop();

   Pos3=MOTOR_3.currentPosition();
   MOTOR_3.stop();

   Pos4=MOTOR_4.currentPosition();
   MOTOR_4.stop();
   UP_DOWN=1;
}
if (dados.indexOf("UP_OFF")==0)
{
   UP_DOWN=0;
}
if (dados.indexOf("M1=")==0)
{
  Pos1 = dados.substring(3).toInt();
  MOTOR_1.move(Pos1);
}
if (dados.indexOf("M2=")==0)
{
  Pos2 = dados.substring(3).toInt();
  MOTOR_2.move(Pos2);
}
if (dados.indexOf("M3=")==0)
{
  Pos3 = dados.substring(3).toInt();
  MOTOR_3.move(Pos3);
}
```

```
    if (dados.indexOf("M4=")==0)
    {
      Pos4 = dados.substring(3).toInt();
      MOTOR_4.move(Pos4);

    }
  }

  if (ler==1)
  {
    SENSORS();
    Serial.print("RECEBE");
    delay(150);
    String recebeu = Serial.readStringUntil('|');

    // Aguarda até que o usuário digite "PRONTO" na janela Serial:
    while(recebeu.indexOf("PRONTO")!=0)
    {
      recebeu = Serial.readStringUntil('|');
    }
    if (recebeu.indexOf("PRONTO")==0)
    {
       // Envia todos os dados uma única vez para a janela Serial:

Serial.print("X1="+String(Xmola1)+";F1="+String(F1)+";X2="+String(Xmola2)+";F2="+String(
F2)+";X3="+String(Xmola3)+";F3="+String(F3)+";X4="+String(Xmola4)+";F4="+String(F4)+";
Y="+String(Y)+";P="+String(P)+";R="+String(R));
    }
    ler=0;
  }
  if (loo==1)
  {
    // Lé os dados e envia para a janela Serial a cada 1 segundo, até que o usuário digite
"LOOP_OFF".
    SENSORS();

Serial.println("X1="+String(Xmola1)+";F1="+String(F1)+";X2="+String(Xmola2)+";F2="+String
(F2)+";X3="+String(Xmola3)+";F3="+String(F3)+";X4="+String(Xmola4)+";F4="+String(F4)+";
Y="+String(Y)+";P="+String(P)+";R="+String(R));
    delay(1000);
  }
  if (manual==0)
  {
     if (FORCES==1)
     {
       verifica=0;
```

```
      VERIFICAR_FORCAS();
      if (F1<(F1_VALUE-8) || F2<(F2_VALUE-8) || F3<(F3_VALUE-8) || F4<(F4_VALUE-8) ||
F1>(F1_VALUE+8) || F2>(F2_VALUE+8) || F3>(F3_VALUE+8) || F4>(F4_VALUE+8))
      {
        verifica=1;
        VERIFICAR_FORCAS();
      }

       MOTOR_3.stop();
       MOTOR_3.stop();
       MOTOR_3.stop();
       MOTOR_3.stop();

       MOTOR_1.setCurrentPosition(0);
       MOTOR_2.setCurrentPosition(0);
       MOTOR_3.setCurrentPosition(0);
       MOTOR_4.setCurrentPosition(0);

       Pos1=MOTOR_1.currentPosition();
       Pos2=MOTOR_2.currentPosition();
       Pos3=MOTOR_3.currentPosition();
       Pos4=MOTOR_4.currentPosition();

    }

    if (FORCES!=1)
    {

        MOTOR_1.setSpeed(300);
       MOTOR_2.setSpeed(300);
       MOTOR_3.setSpeed(300);
       MOTOR_4.setSpeed(300);

       if (Pos1<0){MOTOR_1.setSpeed(-300);}
       if (Pos2<0){MOTOR_2.setSpeed(-300);}
       if (Pos3<0){MOTOR_3.setSpeed(-300);}
       if (Pos4<0){MOTOR_4.setSpeed(-300);}

       while (MOTOR_1.distanceToGo()!=0 || MOTOR_2.distanceToGo()!=0 ||
MOTOR_3.distanceToGo()!=0 || MOTOR_4.distanceToGo()!=0)
       {
         MOTOR_1.run();
         MOTOR_2.run();
         MOTOR_3.run();
         MOTOR_4.run();
       }
```

```
    Pos1=0;
    Pos2=0;
    Pos3=0;
    Pos4=0;
   }

  FORCES=0;
}
if (manual==1)
{
 // Estrutura para movimentação da plataforma via Joystick:
 aX = analogRead(0);
 aY = analogRead(1);

 if (aX<200 && aY>200 && aY<800)
 {
  MOTOR_1.stop();
  MOTOR_1.setCurrentPosition(0);

  MOTOR_2.stop();
  MOTOR_2.setCurrentPosition(0);

  MOTOR_3.setSpeed(-300);
  if (MOTOR_3.distanceToGo()<101)
  {
   MOTOR_3.move(-512);

  }

  MOTOR_4.setSpeed(-300);
  if (MOTOR_4.distanceToGo()<101)
  {
   MOTOR_4.move(-512);

  }
  MOTOR_3.run();
  MOTOR_4.run();

 }
 if (aX>800 && aY>200 && aY<800)
 {
  MOTOR_1.setSpeed(-300);
  if (MOTOR_1.distanceToGo()<101)
  {
   MOTOR_1.move(-512);
  }
```

```
  MOTOR_2.setSpeed(-300);
  if (MOTOR_2.distanceToGo()<101)
  {
    MOTOR_2.move(-512);
  }

  MOTOR_3.stop();
  MOTOR_3.setCurrentPosition(0);

  MOTOR_4.stop();
  MOTOR_4.setCurrentPosition(0);
  MOTOR_1.run();
  MOTOR_2.run();
}
if (aX>200 && aX<800 && aY<200)
{
  if (UP_DOWN==0)
  {

  if (MOTOR_1.distanceToGo()<101)
  {
    MOTOR_1.move(-512);
  }

  if (MOTOR_4.distanceToGo()<101)
  {
    MOTOR_4.move(-512);
  }

  MOTOR_2.stop();
  MOTOR_2.setCurrentPosition(0);

  MOTOR_3.stop();
  MOTOR_3.setCurrentPosition(0);

  MOTOR_1.run();
  MOTOR_4.run();
 }
  else{
   if (MOTOR_1.distanceToGo()<101)
  {
    MOTOR_1.move(-512);
  }
   if (MOTOR_2.distanceToGo()<101)
  {
    MOTOR_2.move(-512);
```

```
  }
   if (MOTOR_3.distanceToGo()<101)
  {
    MOTOR_3.move(-512);
  }
   if (MOTOR_4.distanceToGo()<101)
   {
     MOTOR_4.move(-512);
   }
    MOTOR_1.setSpeed(-300);
    MOTOR_2.setSpeed(-300);
    MOTOR_3.setSpeed(-300);
    MOTOR_4.setSpeed(-300);

    MOTOR_1.run();
    MOTOR_2.run();
    MOTOR_3.run();
    MOTOR_4.run();
  }
}
 if (aX>200 && aX<800 && aY>800)
 {
   if (UP_DOWN==0)
   {
    MOTOR_1.stop();
    MOTOR_1.setCurrentPosition(0);

    if (MOTOR_2.distanceToGo()<101)
    {
      MOTOR_2.move(-512);
    }
    if (MOTOR_3.distanceToGo()<101)
    {
      MOTOR_3.move(-512);
    }

    MOTOR_4.stop();
    MOTOR_4.setCurrentPosition(0);

    MOTOR_2.setSpeed(-300);
    MOTOR_3.setSpeed(-300);

    MOTOR_2.run();
    MOTOR_3.run();

  }
```

```
    else{
      if (MOTOR_1.distanceToGo()<101)
      {
        MOTOR_1.move(512);
      }
      if (MOTOR_2.distanceToGo()<101)
      {
        MOTOR_2.move(512);
      }
      if (MOTOR_3.distanceToGo()<101)
      {
        MOTOR_3.move(512);
      }
      if (MOTOR_4.distanceToGo()<101)
      {
        MOTOR_4.move(512);
      }

      MOTOR_1.setSpeed(300);
      MOTOR_2.setSpeed(300);
      MOTOR_3.setSpeed(300);
      MOTOR_4.setSpeed(300);

      MOTOR_1.run();
      MOTOR_2.run();
      MOTOR_3.run();
      MOTOR_4.run();
    }
  }
  if (aX<200 && aY>800)
  {
    MOTOR_1.stop();
    MOTOR_1.setCurrentPosition(0);

    MOTOR_2.stop();
    MOTOR_2.setCurrentPosition(0);

    MOTOR_4.stop();
    MOTOR_4.setCurrentPosition(0);

    if (MOTOR_3.distanceToGo()<101)
      {
        MOTOR_3.move(-512);
      }

    MOTOR_3.setSpeed(-300);
```

```
  MOTOR_3.run();

}
if (aX<200 && aY<200)
{
  MOTOR_1.stop();
  MOTOR_1.setCurrentPosition(0);

  MOTOR_2.stop();
  MOTOR_2.setCurrentPosition(0);

  MOTOR_3.stop();
  MOTOR_3.setCurrentPosition(0);

  if (MOTOR_4.distanceToGo()<101)
    {
      MOTOR_4.move(-512);
    }
  MOTOR_4.setSpeed(-300);
  MOTOR_4.run();
}
if (aX>800 && aY<200)
{
  MOTOR_2.stop();
  MOTOR_2.setCurrentPosition(0);

  MOTOR_3.stop();
  MOTOR_3.setCurrentPosition(0);

  MOTOR_4.stop();
  MOTOR_4.setCurrentPosition(0);


  if (MOTOR_1.distanceToGo()<101)
    {
      MOTOR_1.move(-512);
    }
  MOTOR_1.setSpeed(-300);
  MOTOR_1.run();
}
if (aX>800 && aY>800)
{
  MOTOR_1.stop();
  MOTOR_1.setCurrentPosition(0);

  MOTOR_3.stop();
```

```
    MOTOR_3.setCurrentPosition(0);

    MOTOR_4.stop();
    MOTOR_4.setCurrentPosition(0);

    if (MOTOR_2.distanceToGo()<101)
     {
       MOTOR_1.move(-512);
     }
    MOTOR_2.setSpeed(-300);
    MOTOR_2.run();
   }
  }
}
```

APÊNDICE 5 - PROGRAMA DO MICROCONTROLADOR QUE GERENCIA OS SENSORES NA PLATAFORMA

```
#include <SoftwareSerial.h>
SoftwareSerial ESPserial(4, 3); // RX | TX
#include "I2Cdev.h"
#include "MPU6050_6Axis_MotionApps20.h"
#if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
   #include "Wire.h"
#endif
MPU6050 mpu;
#define OUTPUT_READABLE_YAWPITCHROLL
#define INTERRUPT_PIN 2  // use pin 2 on Arduino Uno & most boards
int sensor[8],MAX,i,x;
int sensorValue[8];
bool blinkState = false;
bool dmpReady = false;  // set true if DMP init was successful
uint8_t mpuIntStatus;   // holds actual interrupt status byte from MPU
uint8_t devStatus;      // return status after each device operation (0 = success, !0 = error)
uint16_t packetSize;    // expected DMP packet size (default is 42 bytes)
uint16_t fifoCount;     // count of all bytes currently in FIFO
uint8_t fifoBuffer[64]; // FIFO storage buffer
// orientation/motion vars
Quaternion q;           // [w, x, y, z]         quaternion container
VectorInt16 aa;         // [x, y, z]            accel sensor measurements
VectorInt16 aaReal;     // [x, y, z]            gravity-free accel sensor measurements
VectorInt16 aaWorld;    // [x, y, z]            world-frame accel sensor measurements
VectorFloat gravity;    // [x, y, z]            gravity vector
//float euler[3];         // [psi, theta, phi]    Euler angle container
float ypr[3];           // [yaw, pitch, roll]   yaw/pitch/roll container and gravity vector
uint8_t Y1,P1,R1,Y2,P2,R2,SINAL_Y,SINAL_P,SINAL_R=0;
short int Y,P,R=0;
volatile bool mpuInterrupt = false;     // indicates whether MPU interrupt pin has gone high
void dmpDataReady() {
   mpuInterrupt = true;
}
void setup() {
     #if I2CDEV_IMPLEMENTATION == I2CDEV_ARDUINO_WIRE
       Wire.begin();
       Wire.setClock(400000); // 400kHz I2C clock. Comment this line if having compilation
difficulties
   #elif I2CDEV_IMPLEMENTATION == I2CDEV_BUILTIN_FASTWIRE
      Fastwire::setup(400, true);
   #endif
     Serial.begin(9600);
```

```
  // Start the software serial for communication with the ESP8266
    ESPserial.begin(9600);
     while (!Serial);
  for (x=0;x<=3;x++)
  {
    sensorValue[x]=0;
    for (i=1;i<=10;i++)
    {
      switch(x)
      {
        case 0:sensor[x]=map(analogRead(A0),0,1023,0,255);break;
        case 1:sensor[x]=map(analogRead(A1),0,1023,0,255);break;
        case 2:sensor[x]=map(analogRead(A2),0,1023,0,255);break;
        case 3:sensor[x]=map(analogRead(A3),0,1023,0,255);break;
      }
          if (sensor[x]>MAX)
      {
        MAX=sensor[x];
      }
      delay(100);
    }
    sensor[x]=MAX;
    MAX=0;
  }
    mpu.initialize();
    pinMode(INTERRUPT_PIN, INPUT);
       devStatus = mpu.dmpInitialize();
    // supply your own gyro offsets here, scaled for min sensitivity
    mpu.setXGyroOffset(92);
    mpu.setYGyroOffset(-38);
    mpu.setZGyroOffset(2);
    mpu.setXAccelOffset(-1752);
    mpu.setYAccelOffset(1002);
    mpu.setZAccelOffset(769);
    if (devStatus == 0) {
         mpu.setDMPEnabled(true);
        // enable Arduino interrupt detection
        attachInterrupt(digitalPinToInterrupt(INTERRUPT_PIN), dmpDataReady, RISING);
        mpuIntStatus = mpu.getIntStatus();
        dmpReady = true;
        packetSize = mpu.dmpGetFIFOPacketSize();
    } else {
        }
}
void ACELEROMETRO()
{
```

```
// if programming failed, don't try to do anything
  if (!dmpReady) return;

  // wait for MPU interrupt or extra packet(s) available
  while (!mpuInterrupt && fifoCount < packetSize) {

  }
  // reset interrupt flag and get INT_STATUS byte
  mpuInterrupt = false;
  mpuIntStatus = mpu.getIntStatus();
  // get current FIFO count
  fifoCount = mpu.getFIFOCount();
  // check for overflow (this should never happen unless our code is too inefficient)
  if ((mpuIntStatus & 0x10) || fifoCount == 1024) {
     // reset so we can continue cleanly
     mpu.resetFIFO();
        // otherwise, check for DMP data ready interrupt (this should happen frequently)
  } else if (mpuIntStatus & 0x02) {
     // wait for correct available data length, should be a VERY short wait
     while (fifoCount < packetSize) fifoCount = mpu.getFIFOCount();
     // read a packet from FIFO
     mpu.getFIFOBytes(fifoBuffer, packetSize);
           // track FIFO count here in case there is > 1 packet available
     // (this lets us immediately read more without waiting for an interrupt)
     fifoCount -= packetSize;

     #ifdef OUTPUT_READABLE_YAWPITCHROLL
        // display Euler angles in degrees
        mpu.dmpGetQuaternion(&q, fifoBuffer);
        mpu.dmpGetGravity(&gravity, &q);
        mpu.dmpGetYawPitchRoll(ypr, &q, &gravity);
        Y=round(ypr[0] * 180/M_PI);
        if (Y<0)
        {
          Y=abs(Y);
          SINAL_Y=1;
        }
        else
        {
          SINAL_Y=0;
        }
        Y1 = Y & 0xFF;
        Y2 = Y>>8;
        P=round(ypr[1] * 180/M_PI);
        if (P<0)
        {
```

```
            P=abs(P);
            SINAL_P=1;
          }
          else
          {
            SINAL_P=0;
          }
          P1 = P & 0xFF;
          P2 = P>>8;
          R=round(ypr[2] * 180/M_PI);
          if (R<0)
          {
            R=abs(R);
            SINAL_R=1;
          }
          else
          {
            SINAL_R=0;
          }
          R1 = R & 0xFF;
          R2 = R>>8;
           #endif
    }
}
void loop() {

  for (x=0;x<=3;x++)
  {
    switch(x)
     {
       case 0:
            sensorValue[x]=map(analogRead(A0),0,1023,0,255)-sensor[x];
            if (sensorValue[x]<0)
            {
              sensorValue[x]=0;
            }
            break; // 0,37mm = 1 ... até 10mm permanece constante.
       case 1:
            sensorValue[x]=map(analogRead(A1),0,1023,0,255)- sensor[x];
            if (sensorValue[x]<0)
            {
              sensorValue[x]=0;
            }
            break;
       case 2:
            sensorValue[x]=map(analogRead(A2),0,1023,0,255)- sensor[x];
```

```
            if (sensorValue[x]<0)
            {
              sensorValue[x]=0;
            }

            break;
       case 3:
            sensorValue[x]=map(analogRead(A3),0,1023,0,255)- sensor[x];
            if (sensorValue[x]<0)
            {
              sensorValue[x]=0;
            }
             break;
     }
  }
  ACELEROMETRO();

ESPserial.print("C1="+String(sensorValue[0])+"|C2="+String(sensorValue[1])+"|C3="+String(
sensorValue[2])+"|C4="+String(sensorValue[3])+"|"+"Y1="+String(Y1)+"|"+"Y2="+String(Y2)+
"|"+"P1="+String(P1)+"|"+"P2="+String(P2)+"|"+"R1="+String(R1)+"|"+"R2="+String(R2)+"|"+"
SY="+String(SINAL_Y)+"|"+"SP="+String(SINAL_P)+"|"+"SR="+String(SINAL_R)+"|");
  delay(50);
}
```

APÊNDICE 6 - PROGRAMA DO MÓDULO WIFI ESP8266 PARA VISUALIZAÇÃO DO GRÁFICO DAS FORÇAS NOS CABOS

```
#include <ESP8266WiFi.h>
#include <WiFiClient.h>
#include <ESP8266WebServer.h>

uint8_t c1=0; uint8_t c2=0; uint8_t c3=0; uint8_t c4=0; uint8_t y_1=0;
uint8_t y2=0; uint8_t p1=0; uint8_t p2=0; uint8_t r1=0; uint8_t r2=0;
uint8_t sy=0; uint8_t sp=0; uint8_t sr=0;

short int YAW,YAW0=0;

short int Y,P,R=0;
float Xmola1,Xmola2,Xmola3,Xmola4=0;
int F1,F2,F3,F4=0;

const char* ssid = "Servidor_ESP";
const char* password = "12345678";

ESP8266WebServer server(80); //Server on port 80

void ENVIAR() {
  String TEXTO_HTML;
   LER_ARDUINO();

  int alturaF1 = round(F1*100/343);
  int alturaF2 = round(F2*100/343
  int alturaF3 = round(F3*100/343);
  int alturaF4 = round(F4*100/343);


  TEXTO_HTML="<html><head><meta http-equiv=\"refresh\" content=\"5\"><meta
charset=\"utf-8\" /><title>ROBÔ GUIADO POR CABOS</title></head><body>";
  TEXTO_HTML+="<h2 align=\"center\">Tensões nos Cabos</h2><table width=\"100%\"
height=\"90%\" border=\"0\" align=\"center\" cellpadding=\"3\" cellspacing=\"2\" cols=\"4\">";
  TEXTO_HTML+="<tr><td align=\"center\" colspan=\"4\"><h3 align=\"center\">Theta:"+
String(R)+"<br>Phi:"+ String(P)+"</h3></td></tr><tr><td align=\"center\" valign=\"bottom\"
style=\"height:100%; width:25%; background:linear-gradient(red,yellow,green);\">";
  TEXTO_HTML+="<div style=\"text-align:center; vertical-align:middle; width:70%; height:"+
String(alturaF1) +"%; background-color:silver; filter:opacity(alpha=70);";
  TEXTO_HTML+="-moz-opacity:0.7; opacity:0.7;\"><h2>"+ String(F1) +"</h2></div></td><td
align=\"center\" valign=\"bottom\" style=\"height:100%; width:25%;";
  TEXTO_HTML+="background:linear-gradient(red,yellow,green);\"><div style=\"text-
align:center; vertical-align:middle; width:70%; height:"+ String(alturaF2) +"%; background-
color:silver;";
  TEXTO_HTML+="filter:opacity(alpha=70);-moz-opacity:0.7; opacity:0.7;\"><h2>"+
String(F2) +"</h2></div></td><td align=\"center\" valign=\"bottom\" style=\"height:100%;
width:25%;";
  TEXTO_HTML+="background:linear-gradient(red,yellow,green);\"><div style=\"text-
align:center; vertical-align:middle; width:70%; height:"+ String(alturaF3) +"%; background-
color:silver;";
```

```
  TEXTO_HTML+="filter:opacity(alpha=70);-moz-opacity:0.7; opacity:0.7;\"><h2>"+
String(F3) +"</h2></div></td><td align=\"center\" valign=\"bottom\" style=\"height:100%;
width:25%;";
  TEXTO_HTML+="background:linear-gradient(red,yellow,green);\"><div style=\"text-
align:center; vertical-align:middle; width:70%; height:"+ String(alturaF4) +"%; background-
color:silver;";
  TEXTO_HTML+="filter:opacity(alpha=70);-moz-opacity:0.7; opacity:0.7;\"><h2>"+
String(F4) +"</h2></div></td></tr><tr><td align=\"center\"><b>CABO 1 [x10<sup>-
2</sup>N]</b>";
  TEXTO_HTML+="</td><td align=\"center\"><b>CABO 2 [x10<sup>-2</sup>N]</b></td><td
align=\"center\"><b>CABO 3 [x10<sup>-2</sup>N]</b></td><td align=\"center\">";
  TEXTO_HTML+="<b>CABO 4 [x10<sup>-
2</sup>N]</b></td></tr></table></body></html>";

  server.send(200, "text/html", TEXTO_HTML);
}

void setup(void){
  Serial.begin(9600);
  WiFi.mode(WIFI_AP);
  WiFi.softAP(ssid, password);

  IPAddress myIP = WiFi.softAPIP(); //Get IP address

  server.on("/", ENVIAR);

  server.begin();
}

void LER_ARDUINO(void) {

    while (Serial.available())
    {
       int limpar_serial = Serial.read();
    }

    delay(120);

    while (Serial.available())
    {
      String inData = Serial.readStringUntil('|');

       if (inData.indexOf("C1=")==0)
    {
      c1 = (uint8_t)inData.substring(3).toInt();
      Xmola1 = c1*0.37;

    }
    if (inData.indexOf("C2=")==0)
    {
      c2 = (uint8_t)inData.substring(3).toInt();
      Xmola2 = c2*0.37;
    }
```

```
if (inData.indexOf("C3=")==0)
{
  c3 = (uint8_t)inData.substring(3).toInt();
  Xmola3 = c3*0.37;
}
if (inData.indexOf("C4=")==0)
{
  c4 = (uint8_t)inData.substring(3).toInt();
  Xmola4 = c4*0.37;
}
   if (inData.indexOf("y_1=")==0)
   {
     y_1 = (uint8_t)inData.substring(3).toInt();

   }
   if (inData.indexOf("Y2=")==0)
   {
     y2 = (uint8_t)inData.substring(3).toInt();

   }
   if (inData.indexOf("P1=")==0)
   {
     p1 = (uint8_t)inData.substring(3).toInt();

   }
   if (inData.indexOf("P2=")==0)
   {
     p2 = (uint8_t)inData.substring(3).toInt();

   }
   if (inData.indexOf("R1=")==0)
   {
     r1 = (uint8_t)inData.substring(3).toInt();

   }
   if (inData.indexOf("R2=")==0)
   {
     r2 = (uint8_t)inData.substring(3).toInt();

   }
   if (inData.indexOf("SY=")==0)
   {
     sy = (uint8_t)inData.substring(3).toInt();

   }
   if (inData.indexOf("SP=")==0)
   {
     sp = (uint8_t)inData.substring(3).toInt();

   }
   if (inData.indexOf("SR=")==0)
   {
     sr = (uint8_t)inData.substring(3).toInt();
```

```
    }
  }


  // Remonta os valores dos ângulos:
Y = (y2<<8)|y_1;
P = (p2<<8)|p1;
R = (r2<<8)|r1;

// Convenção de giro positivo ou negativo:
if (sy==0)
{
 Y=Y*(-1);
}
if (YAW0==0 && Y!=0)
{
 YAW = -Y;
 YAW0=1;
}
// Giro relativo em Z considerando o primeiro valor
// lido como o ponto zero de partida:
Y = Y + YAW;

if (sp==1)
{
 P=P*(-1);
}
if (sr==1)
{
 R=R*(-1);
}


if (Xmola1==0)
{
  F1=0;
}
else
{
  // Força no cabo 1 [x10^-2 N]
  // F - 1.4715 = K*X
  // K = 196   Pré-carga = 1.4715 N
  F1 = round(((196 * (Xmola1/1000))+1.4715)*100);
}

if (Xmola2==0)
{
  F2=0;
}
else
{
  F2 = round(((196 * (Xmola2/1000))+1.4715)*100);
```

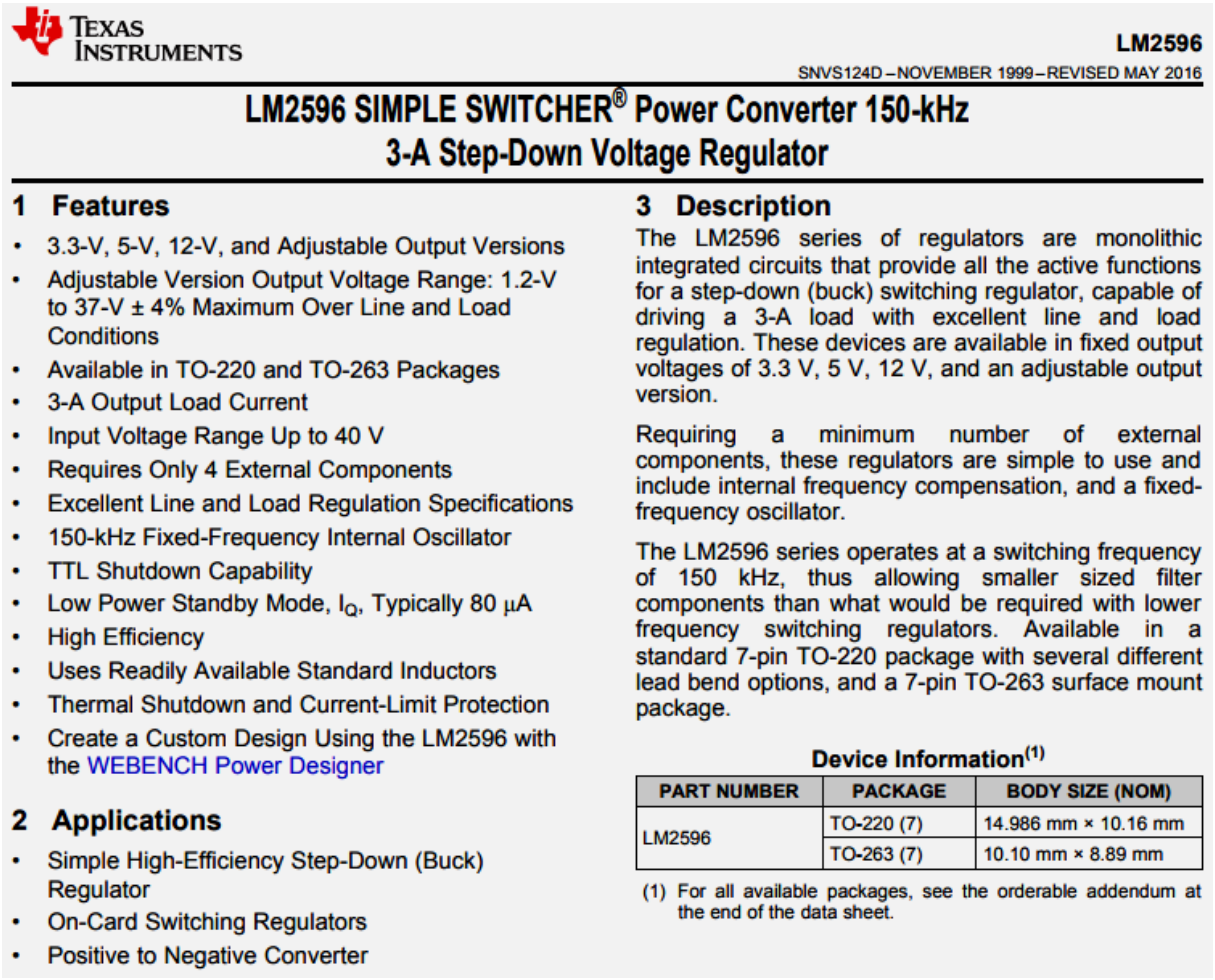
```
  }

  if (Xmola3==0)
  {
    F3=0;
  }
  else
  {
    F3 = round(((196 * (Xmola3/1000))+1.4715)*100);
  }

  if (Xmola4==0)
  {
    F4=0;
  }
  else
  {
    F4 = round(((196 * (Xmola4/1000))+1.4715)*100);
  }
}

void loop(void){
  server.handleClient();
}
```

ANEXO 1 - INFORMAÇÕES SOBRE O CI LM2596



TEXAS INSTRUMENTS

LM2596

SNVS124D – NOVEMBER 1999 – REVISED MAY 2016

LM2596 SIMPLE SWITCHER® Power Converter 150-kHz 3-A Step-Down Voltage Regulator

1 Features

- 3.3-V, 5-V, 12-V, and Adjustable Output Versions
- Adjustable Version Output Voltage Range: 1.2-V to 37-V ± 4% Maximum Over Line and Load Conditions
- Available in TO-220 and TO-263 Packages
- 3-A Output Load Current
- Input Voltage Range Up to 40 V
- Requires Only 4 External Components
- Excellent Line and Load Regulation Specifications
- 150-kHz Fixed-Frequency Internal Oscillator
- TTL Shutdown Capability
- Low Power Standby Mode, I_Q, Typically 80 μA
- High Efficiency
- Uses Readily Available Standard Inductors
- Thermal Shutdown and Current-Limit Protection
- Create a Custom Design Using the LM2596 with the WEBENCH Power Designer

2 Applications

- Simple High-Efficiency Step-Down (Buck) Regulator
- On-Card Switching Regulators
- Positive to Negative Converter

3 Description

The LM2596 series of regulators are monolithic integrated circuits that provide all the active functions for a step-down (buck) switching regulator, capable of driving a 3-A load with excellent line and load regulation. These devices are available in fixed output voltages of 3.3 V, 5 V, 12 V, and an adjustable output version.

Requiring a minimum number of external components, these regulators are simple to use and include internal frequency compensation, and a fixed-frequency oscillator.

The LM2596 series operates at a switching frequency of 150 kHz, thus allowing smaller sized filter components than what would be required with lower frequency switching regulators. Available in a standard 7-pin TO-220 package with several different lead bend options, and a 7-pin TO-263 surface mount package.

Device Information(1)

PART NUMBER	PACKAGE	BODY SIZE (NOM)
LM2596	TO-220 (7)	14.986 mm × 10.16 mm
	TO-263 (7)	10.10 mm × 8.89 mm

(1) For all available packages, see the orderable addendum at the end of the data sheet.

Fonte: INSTRUMENTS, 2017.

.

ANEXO 2 - INFORMAÇÕES SOBRE O CI LM1117

LM1117 800-mA Low-Dropout Linear Regulator

1 Features

- Available in 1.8 V, 2.5 V, 3.3 V, 5 V, and Adjustable Versions
- Space-Saving SOT-223 and WSON Packages
- Current Limiting and Thermal Protection
- Output Current 800 mA
- Line Regulation 0.2% (Maximum)
- Load Regulation 0.4% (Maximum)
- Temperature Range
  - LM1117: 0°C to 125°C
  - LM1117I: −40°C to 125°C

2 Applications

- Post Regulator for Switching DC–DC Converter
- High Efficiency Linear Regulators
- Battery Chargers
- Portable Instrumentation
- Active SCSI Termination Regulator

3 Description

The LM1117 is a low dropout voltage regulator with a dropout of 1.2 V at 800 mA of load current.

The LM1117 is available in an adjustable version, which can set the output voltage from 1.25 to 13.8 V with only two external resistors. In addition, it is available in five fixed voltages, 1.8 V, 2.5 V, 3.3 V, and 5 V.

The LM1117 offers current limiting and thermal shutdown. Its circuit includes a Zener trimmed bandgap reference to assure output voltage accuracy to within ±1%.

A minimum of 10-µF tantalum capacitor is required at the output to improve the transient response and stability.

Device Information(1)

PART NUMBER	PACKAGE	BODY SIZE (NOM)
LM1117, LM1117I	SOT-223 (4)	6.50 mm × 3.50 mm
	TO-220 (3)	14.986 mm × 10.16 mm
	TO-252 (3)	6.58 mm × 6.10 mm
	WSON (8)	4.00 mm × 4.00 mm
	TO-263 (3)	10.18 mm × 8.41 mm

(1) For all available packages, see the orderable addendum at the end of the data sheet.

Fonte: INSTRUMENTS, 2017.

ANEXO 3 - INFORMAÇÕES SOBRE O MÓDULO ESP8266

Table 1-1. Main Technical Specifications

Categories	Items	Parameters
Wi-Fi	Standards	FCC/CE/TELEC/SRRC
	Protocols	802.11 b/g/n/e/i
	Frequency Range	2.4G ~ 2.5G (2400M ~ 2483.5M)
	Tx Power	802.11 b: +20 dBm
		802.11 g: +17 dBm
		802.11 n: +14 dBm
	Rx Sensitivity	802.11 b: -91 dbm (11 Mbps)
		802.11 g: -75 dbm (54 Mbps)
		802.11 n: -72 dbm (MCS7)
	Antenna	PCB Trace, External, IPEX Connector, Ceramic Chip
Hardware	CPU	Tensilica L106 32-bit micro controller
	Peripheral Interface	UART/SDIO/SPI/I2C/I2S/IR Remote Control
		GPIO/ADC/PWM/LED Light & Button
	Operating Voltage	2.5V ~ 3.6V
	Operating Current	Average value: 80 mA
	Operating Temperature Range	-40°C ~ 125°C
	Storage Temperature Range	-40°C ~ 125°C
	Package Size	QFN32-pin (5 mm x 5 mm)
	External Interface	-
Software	Wi-Fi Mode	Station/SoftAP/SoftAP+Station
	Security	WPA/WPA2
	Encryption	WEP/TKIP/AES
	Firmware Upgrade	UART Download / OTA (via network)
	Software Development	Supports Cloud Server Development / Firmware and SDK for fast on-chip programming
	Network Protocols	IPv4, TCP/UDP/HTTP/FTP
	User Configuration	AT Instruction Set, Cloud Server, Android/iOS App

Fonte: ESPRESSIF, 2017.

ANEXO 4 - INFORMAÇÕES SOBRE O CI MPU6050

InvenSense	MPU-6000/MPU-6050 Product Specification	Document Number: PS-MPU-6000A-00 Revision: 3.2 Release Date: 11/16/2011

5 Features

5.1 Gyroscope Features

The triple-axis MEMS gyroscope in the MPU-60X0 includes a wide range of features:

- Digital-output X-, Y-, and Z-Axis angular rate sensors (gyroscopes) with a user-programmable full-scale range of ±250, ±500, ±1000, and ±2000°/sec
- External sync signal connected to the FSYNC pin supports image, video and GPS synchronization
- Integrated 16-bit ADCs enable simultaneous sampling of gyros
- Enhanced bias and sensitivity temperature stability reduces the need for user calibration
- Improved low-frequency noise performance
- Digitally-programmable low-pass filter
- Gyroscope operating current: 3.6mA
- Standby current: 5µA
- Factory calibrated sensitivity scale factor
- User self-test

5.2 Accelerometer Features

The triple-axis MEMS accelerometer in MPU-60X0 includes a wide range of features:

- Digital-output triple-axis accelerometer with a programmable full scale range of ±2*g*, ±4*g*, ±8*g* and ±16*g*
- Integrated 16-bit ADCs enable simultaneous sampling of accelerometers while requiring no external multiplexer
- Accelerometer normal operating current: 500µA
- Low power accelerometer mode current: 10µA at 1.25Hz, 20µA at 5Hz, 60µA at 20Hz, 110µA at 40Hz
- Orientation detection and signaling
- Tap detection
- User-programmable interrupts
- Free-fall interrupt
- High-G interrupt
- Zero Motion/Motion interrupt
- User self-test

5.3 Additional Features

The MPU-60X0 includes the following additional features:

- 9-Axis MotionFusion by the on-chip Digital Motion Processor (DMP)
- Auxiliary master I^2C bus for reading data from external sensors (e.g., magnetometer)
- 3.9mA operating current when all 6 motion sensing axes and the DMP are enabled

Fonte: INVENSENSE, 2017.

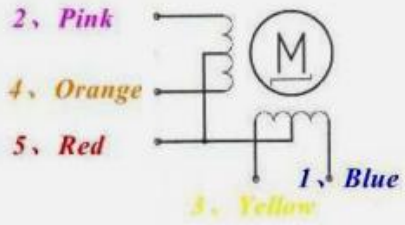
ANEXO 5 - INFORMAÇÕES SOBRE O MOTOR DE PASSO 28BYJ-48

Kiatronics
electronic design and manufacture

28BYJ-48 – 5V Stepper Motor

The 28BYJ-48 is a small stepper motor suitable for a large range of applications.

Rated voltage :	5VDC
Number of Phase	4
Speed Variation Ratio	1/64
Stride Angle	5.625°/64
Frequency	100Hz
DC resistance	50Ω±7%(25℃)
Idle In-traction Frequency	> 600Hz
Idle Out-traction Frequency	> 1000Hz
In-traction Torque	>34.3mN.m(120Hz)
Self-positioning Torque	>34.3mN.m
Friction torque	600-1200 gf.cm
Pull in torque	300 gf.cm
Insulated resistance	>10MΩ(500V)
Insulated electricity power	600VAC/1mA/1s
Insulation grade	A
Rise in Temperature	<40K(120Hz)
Noise	<35dB(120Hz,No load,10cm)
Model	28BYJ-48 – 5V



Fonte: KIATRONICS, 2015.

ANEXO 6 - INFORMAÇÕES SOBRE O DRIVER ULN2003

TEXAS INSTRUMENTS

ULN2002A, ULN2003A, ULN2003AI
ULQ2003A, ULN2004A, ULQ2004A

SLRS027O – DECEMBER 1976 – REVISED JANUARY 2016

ULN200x, ULQ200x High-Voltage, High-Current Darlington Transistor Arrays

1 Features

- 500-mA-Rated Collector Current (Single Output)
- High-Voltage Outputs: 50 V
- Output Clamp Diodes
- Inputs Compatible With Various Types of Logic
- Relay-Driver Applications

2 Applications

- Relay Drivers
- Stepper and DC Brushed Motor Drivers
- Lamp Drivers
- Display Drivers (LED and Gas Discharge)
- Line Drivers
- Logic Buffers

3 Description

The ULx200xA devices are high-voltage, high-current Darlington transistor arrays. Each consists of seven NPN Darlington pairs that feature high-voltage outputs with common-cathode clamp diodes for switching inductive loads.

The collector-current rating of a single Darlington pair is 500 mA. The Darlington pairs can be paralleled for higher current capability. Applications include relay drivers, hammer drivers, lamp drivers, display drivers (LED and gas discharge), line drivers, and logic buffers. For 100-V (otherwise interchangeable) versions of the ULx2003A devices, see the SLRS023 data sheet for the SN75468 and SN75469 devices.

The ULN2002A device is designed specifically for use with 14-V to 25-V PMOS devices. Each input of this device has a Zener diode and resistor in series to control the input current to a safe limit. The ULx2003A devices have a 2.7-kΩ series base resistor for each Darlington pair for operation directly with TTL or 5-V CMOS devices.

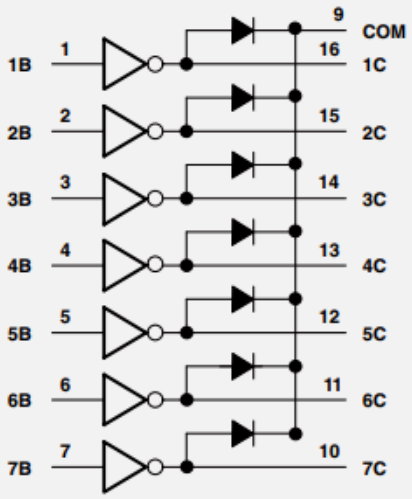
The ULx2004A devices have a 10.5-kΩ series base resistor to allow operation directly from CMOS devices that use supply voltages of 6 V to 15 V. The required input current of the ULx2004A device is below that of the ULx2003A devices, and the required voltage is less than that required by the ULN2002A device.

Device Information(1)

PART NUMBER	PACKAGE	BODY SIZE (NOM)
ULx200xD	SOIC (16)	9.90 mm × 3.91 mm
ULx200xN	PDIP (16)	19.30 mm × 6.35 mm
ULN200xNS	SOP (16)	10.30 mm × 5.30 mm
ULN200xPW	TSSOP (16)	5.00 mm × 4.40 mm

(1) For all available packages, see the orderable addendum at the end of the data sheet.

Simplified Block Diagram



Fonte: INSTRUMENTS, 2015.