UNIVERSIDADE FEDERAL DE SÃO CARLOS
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
DEPARTAMENTO DE ENGENHARIA MECÂNICA

ESTUDO E CONTROLE DE UM ROBÔ HEXÁPODE UTILIZANDO O *ROS (ROBOT OPERATING SYSTEM)*

VICTOR ABREU NUNES

SÃO CARLOS - SP
2020

ESTUDO E CONTROLE DE UM ROBÔ HEXÁPODE UTILIZANDO *ROS*

Trabalho de Conclusão de Curso apresentado ao Departamento de Engenharia Mecânica da Universidade Federal de São Carlos, para obtenção do título de bacharel em Engenharia Mecânica.

Orientador: Prof. Dr. Gustavo Franco Barbosa

São Carlos - SP
2020

UNIVERSIDADE FEDERAL DE SÃO CARLOS

Centro de Ciências Exatas e Tecnologia

Folha de aprovação

Trabalho de Conclusão de Curso apresentado como requisito parcial para a obtenção do título de bacharel em Engenharia Mecânica pela Universidade Federal de São Carlos.

Aprovado em: ____ / ____ / _____

_____

Orientador
Prof. Dr. Gustavo Franco Barbosa
Universidade Federal de São Carlos

_____

Membro da Banca (1)
Prof. Dr. Sidney Bruce Shiki
Universidade Federal de São Carlos

_____

Membro da Banca (2)
Eng. José Otávio Savazzi
Universidade Federal de São Carlos

AGRADECIMENTOS

Agradeço a meus pais e irmãos, por me darem condições de executar este trabalho, me apoiando e incentivando nos momentos dificeí;

Ao professor Gustavo, por ter sido meu orientador e executar esta função com dedicação, competência e amizade;

A todos os professores do curso de Engenharia Mecânica da UFSCar, por dedicarem seu tempo a ensinar, aconselhar e formar novos profissionais e me permitiram obter um pouco desse conhecimento;

A todos os colegas do curso, os quais convivi de forma próxima durante os últimos anos e me fizeram crescer como pessoa e me ajudaram durante a trajetória acadêmica;

E por fim a todos que participaram diretamente e indiretamente do desenvolvimento deste trabalho e de minha vida.

“Tenho a impressão de ter sido uma criança brincando à beira-mar, divertindo-me em descobrir uma pedrinha mais lisa ou uma concha mais bonita que as outras, enquanto o imenso oceano da verdade continua misterioso diante de meus olhos.”

(Isaac Newton)

RESUMO

Os avanços tecnológicos e a busca por melhoria de produtividade e eficiência, vem provocando muitas mudanças no ambiente industrial, incluindo a adoção de soluções na área da robótica. Neste contexto, o objetivo deste trabalho é desenvolver o modelo e controle de um robô hexápode, dentro da plataforma ROS (*Robot Operating System*). O robô é um projeto em andamento voltado para indústria aeroespacial, inserido no contexto de Indústria 4.0. Propõe-se desenvolver um modelo de cinemática direta e cinemática inversa, aplicá-lo ao programa de visualização do ROS, Rviz, de modo a simular um ciclo de passada. Será avaliado se o modelo de cinemática reproduz as trajetórias esperadas, e se é possível controlar o robô desenvolvido dentro da plataforma, dando espaço para futuros estudos com este robô.

Palavras-chave: Indústria 4.0; Robótica; Hexapóde, *Robot Operating System*.

ABSTRACT

The technological advancements and the search for more productivity and efficiency has been changing the industrial environment, including the adoption of solutions in robotic field. In this scenario the goal of this document is develop a model and control of an hexapod robot, inside ROS (Robot Operating System) environment. The robot is a working project developed for the aircraft industry, inside the concept of 4.0 Industry. The proposal is developing a forward and an inverse kinematic model of the robot and apply them inside the visualization software of ROS, Rviz, to simulate a gait cycle. The kinematic model will be evaluated with the expected trajectory and if is possible for control the robot inside the environment for development of upcoming projects.

Keyword: 4.0 Industry; Robotics; Hexapod; Robot Operating System.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

ROS – *Robot Operating System*

CPS – Cyber Physical System

GPS – Global Positioning System

DH – Denavit-Hartenberg

UFSCar – Universidade Federal de São Carlos

IoT – Internet of Things

CAD – Computer Aided Design

SUMÁRIO

1 INTRODUÇÃO

Indústria 4.0 (I4.0) é como é chamado o movimento de quarta revolução industrial, onde as máquinas, estoques e logística são conectados na forma de CPS (*Cyber-Physical System*), nas quais elementos físicos e computacionais trocam informações e alteram processos de forma automática, com o acompanhamento do produto em todas as etapas (KAGERMANN, 2013).

Um dos pilares utilizados na I4.0 é a IoT (*Internet of Things* ou Internet das Coisas), que permite que coisas como sensores, câmeras, atuadores se comuniquem entre si e com dados de uma fonte externa por exemplo, um computador (HERMAN; PENTEK; OTTO, 2015).

Com a I4.0 em ascensão, o uso da robótica para realização de tarefas repetitivas ou perigosas é incentivado, principalmente quando esta causa uma economia de tempo e de recursos para empresas.

A indústria aeronáutica também necessita de soluções para diversos procedimentos, incluindo a inspeção, que é onerosa e difícil por se tratar de ambientes muito amplos e com altitude elevada, mesmo assim um erro nesta operação pode causar acidentes graves.

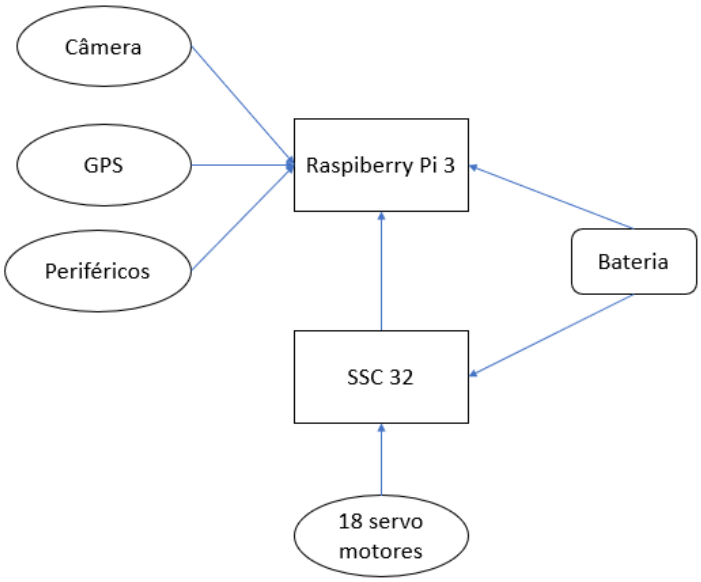
Estas inspeções são realizadas periodicamente e procuram falhas estruturais, superficiais de todos os componentes do avião incluindo asa, fuselagem e outros elementos estruturais.

Visando resolver este problema, foi proposto a utilização de um robô hexápode para inspeção de prendedores, como por exemplo rebites faltantes e defeituosos na asa de aeronaves. O robô foi escolhido pela adaptabilidade a diferentes tipos de terreno. O projeto do robô foi iniciado por um grupo de alunos da Universidade Federal de São Carlos que executou sua manufatura e montagem.

O robô hexápode proposto para realizar esta tarefa trata-se de um modelo com três servos motores por patas (18 no total), controlados por um controlador de servo motores, o SSC-32. Este controlador estará conectado a uma placa Raspberry Pi 3, microcontrolador que é ligado a câmera, que detectará a presença dos prendedores, sistema de localização GPS (G*lobal Positioning System)* e qualquer outro periférico que se seja necessário adicionar ao robô. O esquema de funcionamento deste robô hexápode é representado na Figura 1 e o robô utilizado é o representado na Figura

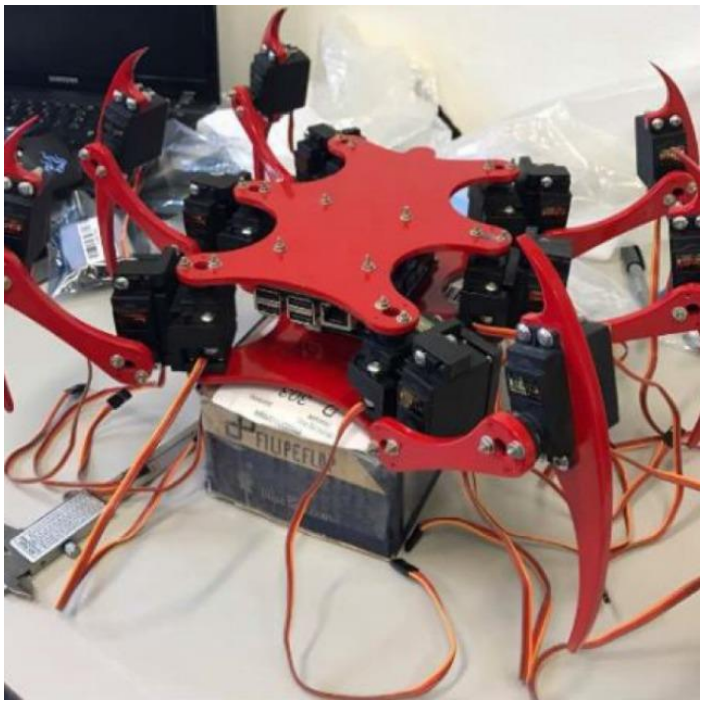
2. O projeto é semelhante ao apresentado por Ivan (2014).

Figura 1: Esquema de funcionamento do robô hexápode.



Fonte: Elaborado pelo autor.

Figura 2: Robô utilizado neste projeto.



Fonte: Elaborado pelo grupo de pesquisa da UFSCar.

Este trabalho visa desenvolver e implementar um sistema de controle e movimentação do robô hexápode para realizar passadas e diferentes poses, conseguindo realizar as diferentes tarefas de inspeção em uma asa de aeronave.

Para isto deve-se integrar o robô hexápode ao sistema operacional para robôs *ROS (Robot Operating System)* que contêm módulos de simulação, movimento e de visão computacional necessários para executar a tarefa.

O objetivo deste trabalho consiste em utilizando as equações de cinemática direta e inversa, controlar as patas do robô hexapóde para executar um movimento de passada, dentro da plataforma *ROS*.

O resultado será avaliado utilizando a ferramenta de visualização inclusa na plataforma, Rviz, após a execução dos algoritmos necessários, com ela será possível verificar se o robô atende aos comandos e excuta o movimento, também serão utilizados outras ferramentas de visualização como a biblioteca graphiz para observar a cadeia cinemática do robô.

Espera-se conseguir executar o movimento dentro da plataforma deixando o robô configurado para futuros projetos, onde poderão ser inclusas novas bibliotecas, como de localização ou visão computacional e melhorias na movimentação desenvolvida.

2 REVISÃO DA LITERATURA

Esta seção apresenta uma revisão da literatura a respeito de todos os tópicos abordados nesta pesquisa, observando o estado da arte e os pontos onde projeto de pesquisa é relevante para o cenário de pesquisa atual.

2.1 Indústria 4.0 (I4.0)

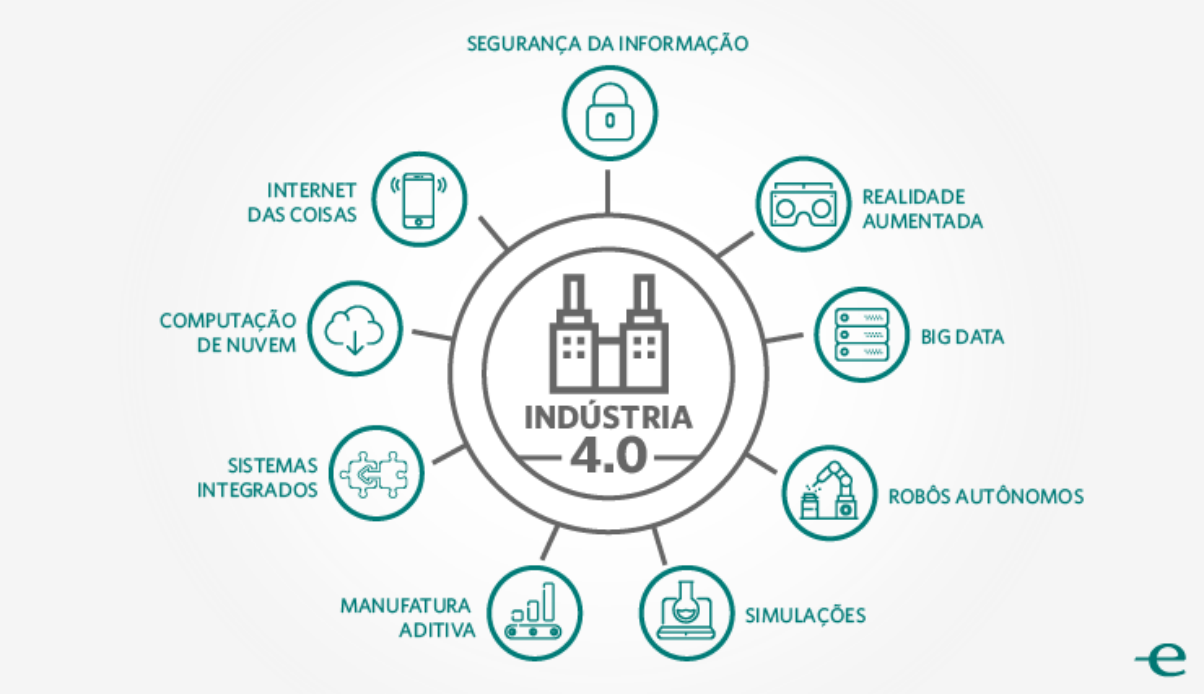
O processo de revoluções industriais surge com novas tecnologias que mudam a maneira que o homem realiza e organiza seu trabalho, na primeira revolução industrial ocorrida na Inglaterra, entre 1760 e 1860, houve a substituição do trabalho artesanal pela produção movida por máquinas a vapor.

Após este período ocorreu a segunda revolução industrial que ganhou força com o fordismo e o modelo de linha de montagem automatizada com esteiras e produção em massa.

A terceira revolução industrial foi a que consolidou-se em grande parte do século XX e XXI, com os avanços em automação, eletrônica, computação, biotecnologia entre outras áreas, é chamada por alguns de Revolução Técnico-Científica e Informacional (SAKURAI, ZUCHI, 2018).

O conceito de quarta revolução industrial ou Indústria 4.0 (I4.0), surgiu na Alemanha e consiste em não apenas automatizar processo mas, através de novas tecnologias como Internet das Coisas ou *Internet of Things (IoT)* e computação em nuvem organizar a produção em um *Cyber Physical System (CPS)* sistemas flexíveis e horizontais onde os dados gerados em cada etapa do processo possam ser acessados rapidamente e serem usados para mudanças na fábrica, as fábricas inteligentes, ou *Smart Factories* (KAGERMANN et al, 2013). A Figura 3 mostra as principais tecnologias da Indústria 4.0:

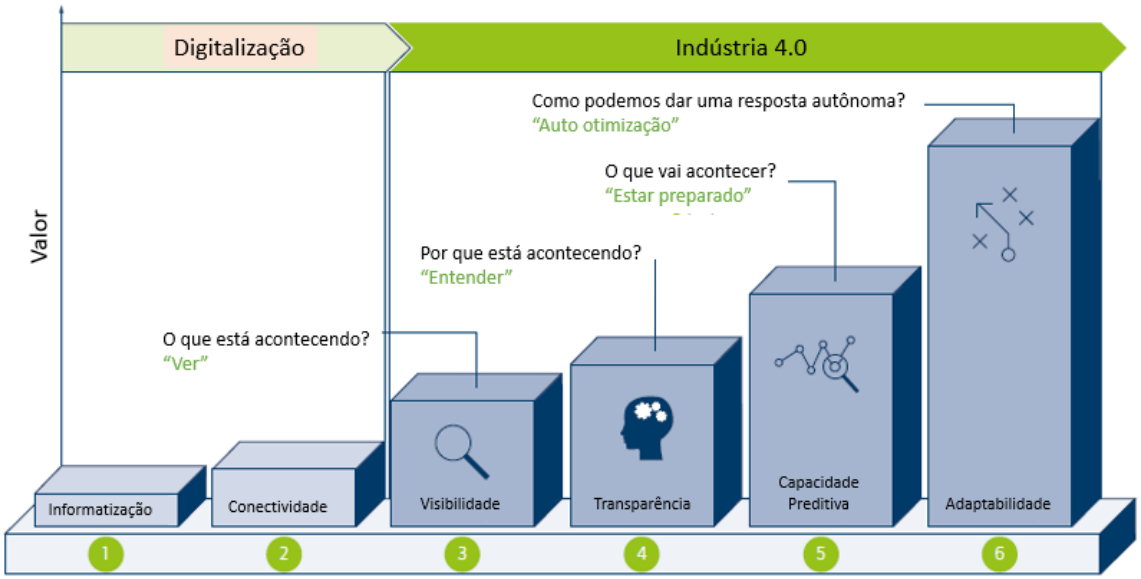
Figura 3: *Tecnologias da Indústria 4.0*



Fonte: https://www.lwtsistemas.com.br

De modo a passar por estas grandes mudanças as empresas necessitariam fazer mudanças para adaptar ou implementar estes modelos de negócio. Para cuidar deste tipo de abordagem foi feito uma avaliação das etapas necessárias para uma empresa adotar o modelo de I4.0 cada etapa promove o aumento da maturidade da instituição neste modelo de negócio. Todos os estágios estão representados na Figura 4.

Figura 4: Estágios para o desenvolvimento da Indústria 4.0.



Fonte: SCHUH et al, 2017 - Traduzido

No estágio de informatização as diferentes tecnologias são usadas separadamente, esta prática já é bem utilizada em certas empresas para atividades repetitivas. Um exemplo de informatização seria a adoção de um torno CNC.

No estágio seguinte, conectividade, temos as máquinas e processos conectados, em forma de dados, um exemplo seria a utilização de desenhos 3D ou 2D para o torno CNC ou corte de chapas. Para este fim servem os protocolos de comunicação como IP (*Internet Protocol*).

O terceiro estágio é visibilidade e consiste em utilizar informações de logística, processo e produto adquiridas por diversos meio para obter um modelo visual e acessível de toda a fábrica, modelo conhecido como fábrica digital. Usualmente nas empresas atuais apenas um grupo de pessoas tem acesso a informação de como anda um produto em determinado momento, dificultando a tomada de decisão. Este estágio se relaciona com o próximo, transparência, onde todas estas informações devem estar disponíveis para melhorias e manutenções, por exemplo.

Com todas estas medidas tomadas temos a capacidade preditiva, que permite com os dados atuais e situações anteriores, antever problemas e designar possíveis soluções.

O último estágio, adaptabilidade consiste em utilizar estes modelos e previsões para realizar decisões ativas na fábrica, como mudar sequência de processos por

esperar que uma máquina vá quebrar, por exemplo (Schuh, et al 2017).

O caso de estudo envolve I4.0 por utilizar a tecnologia de internet das coisas e robótica avançada no primeiro e segundo estágio de maturidade de implementação por se tratar de uma aplicação que não engloba toda uma planta e sim um processo específico, de inspeção de rebites em aeronaves.

2.2 Robô Hexápode

2.2.1 Conceito

No meio terrestre, os robôs usualmente são projetados de duas maneiras de locomoção, por rodas ou com pernas. Enquanto a primeira tem um custo menor e é melhor desenvolvida para terrenos planos, a locomoção por pernas torna possível a passagem por obstáculos como pedras e degraus por exemplo (RUBIO, VALERIO, ALPERT 2019).

Observa-se, portanto, que a locomoção com pernas é a mais indicada para terreno natural, tornando útil para acesso a áreas inacessíveis a outros veículos, outra vantagem é que se um robô movido por rodas sofre um problema com uma delas, sua locomoção é drasticamente prejudicada. Se, no entanto, o robô tiver um número de pernas redundante ele pode ajustar sua configuração para continuar se movendo. Também é possível utilizar as pernas como atuadores no caso de parada por exemplo.

As limitações que ocorrem no uso de pernas são as relativas baixas velocidades que atingem, dificuldade na montagem e complexidade no controle, também é necessário um número maior de atuadores para realizar um movimento (MACHADO, SILVA, 2006).

Dentre os robôs com pernas temos o que contém apenas uma, este robô se locomove pulando e deve se manter dinamicamente estável em todo o trajeto. Apesar de não parecer prático seu uso já foi sugerido para fins militares, pois um alvo que se move pulando seria difícil ser atingido e para exploração em lugares com baixa gravidade, que torna a locomoção com uma perna vantajosa. Um exemplo deste robô é mostrado na Figura 5 (MACHADO, SILVA, 2006).

Figura 5: Robô de uma perna ATRIAS.

Fonte: HURST, 2012

Outro tipo de robô que atrai atenção são os robôs bípedes, por lembrarem a movimentação humana. Porém o desenvolvimento deste tipo de movimento é dificultado pela complexidade de obter um balanço dinâmico. Porém os estudos estão evoluindo, a Figura 6 mostra o Robô Atlas, da Boston Dynamics, que é capaz de realizar diversos movimentos e se manter equilibrado (MACHADO, SILVA, 2006).

Figura 6: Robô Atlas caminhando em um terreno acidentado.

Fonte: KUINDERSMA et al, 2015

Dentre os robôs com mais pernas há destaque para dois grupos, os de 4 (quadrúpede) e os de 6 pernas (hexápode). Os robôs quadrúpedes podem ser montados em uma configuração semelhante aos hexápodes (que será explicado a

seguir) ou imitando a natureza. Como o robô Cheetah desenvolvido pelo MIT (*Massachusetts Institute of Technology*) mostrado na Figura 7, este robô é capaz de realizar movimentos variados, inclusive executar um mortal para trás (MACHADO, SILVA, 2006).

Figura 7: Robô Cheetah 3 do MIT.

Fonte: BLEDT et al, 2018

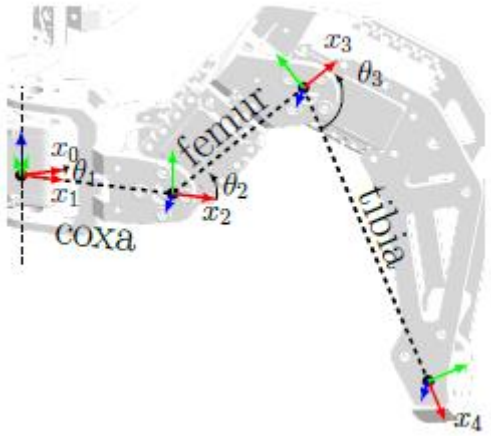
Já robôs hexápodes são fortemente inspirados na natureza, já que a maioria dos artrópodes contém 6 patas, este tipo de robô pode ser considerado robustos no caso de falha em uma das pernas, pois as outras podem continuar o movimento (WANG, DING, ROSETTA, 2010). Este robô foi o escolhido para este projeto de pesquisa pela sua versatilidade em vários tipos de terreno, enquanto mantêm a estabilidade por manter três pontos de contato com o solo. Na Figura 8 há um exemplo de robô hexápode.

Figura 8: *Exemplo de robô hexápode.*

Fonte: FAIGI, ČÍŽEK, 2019

As pernas dos robôs hexápodes usualmente são divididas em 3 partes com 3 juntas de revolução, a coxa, o fêmur e a tíbia como mostrado na Figura 9.

Figura 9: Perna de um robô hexápode.



Fonte: FAIGI, ČÍŽEK, 2019

Este é o modelo de robô estudado neste trabalho, e nas seções seguintes serão discutidos diferentes métodos de controle para o hexápode.

2.2.2 Cinemática Direta

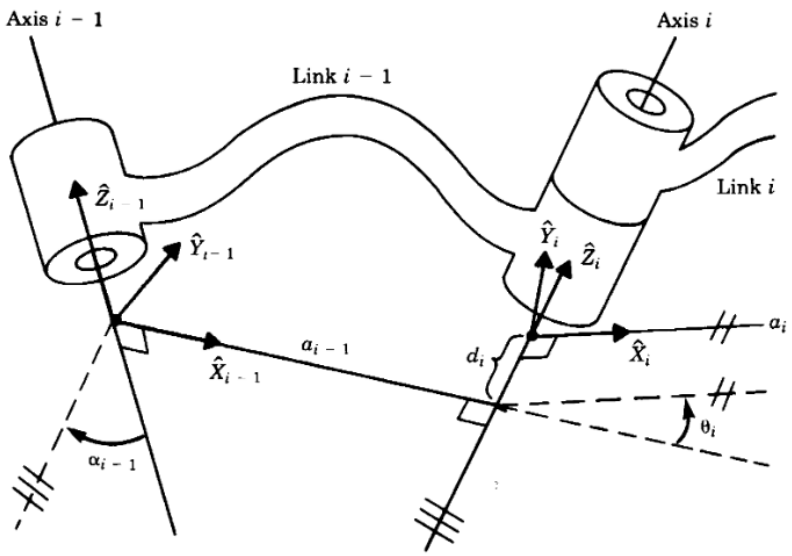
Para controlar e estudar um robô, usualmente é necessário um modelo cinemático do robô, algo que relacione a posição do elo final (garra, ferramenta ou no caso do hexápode a ponta da perna) com a posição das juntas, também chamada de configuração do robô.

Dois modelos são muito usados na robótica, o primeiro, de cinemática direta relaciona o posicionamento das juntas a posição final no plano cartesiano, já a cinemática inversa relaciona uma posição cartesiana com a configuração das juntas necessárias para chegar aquela posição.

Existe mais de um método para chegar no modelo de cinemática direta, mas iremos utilizar o método de Denavit-Hartenberg, como no trabalho de GUREL, 2017.

O método consiste em, dado um conjunto de juntas sendo **i** a junta analizada, **i+1** a junta seguinte e **i-1** a junta anterior, obter a matriz de transformação da junta **i** em relação a **i-1**, utilizando apenas 2 rotações e duas rotações, conforme a Figura 10:

Figura 10: Parâmetros de Denavit-Hartenberg



Fonte: LAGES, 2019

O posicionamento entre duas juntas é reduzido a quatro movimentos, desde que se cumpra as seguintes orientações:

- A origem do sistema de coordenada está alinhada a base do robô;

- Se alinhe o eixo Z_i ao eixo de movimento da junta (prismática ou rotativa);
- O eixo X_i pode ser escolhido livremente, mas deve ser perpendicular a Z_i, usualmente alinha-se X_i com X_{i-i} por conveniência.

Os quatro movimentos, em sequência, são:

- Translação em d_i ao longo de Z_i:

$$D_Z = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (1)$$

- Rotação em θ_{i-1} em torno de Z_i, alinhando X_i com X_{i-1}:

$$R_Z = \begin{bmatrix} \cos\theta_i & -\text{sen}\theta_i & 0 & 0 \\ \text{sen}\theta_i & \cos\theta_i & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2)$$

- Translação em a_{i-1} ao longo de X_i, tornando as origens coincidentes:

$$D_X = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (3)$$

- Rotação em α_{i-1} ao em torno de X_i, alinhando Z_i com Z_{i-1}:

$$R_X = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha_i & -\text{sen}\,\alpha_i & 0 \\ 0 & \text{sen}\,\alpha_i & \cos\alpha_i & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4)$$

A matriz de transformação T é dada pela multiplicação desses quatro movimentos, de modo que:

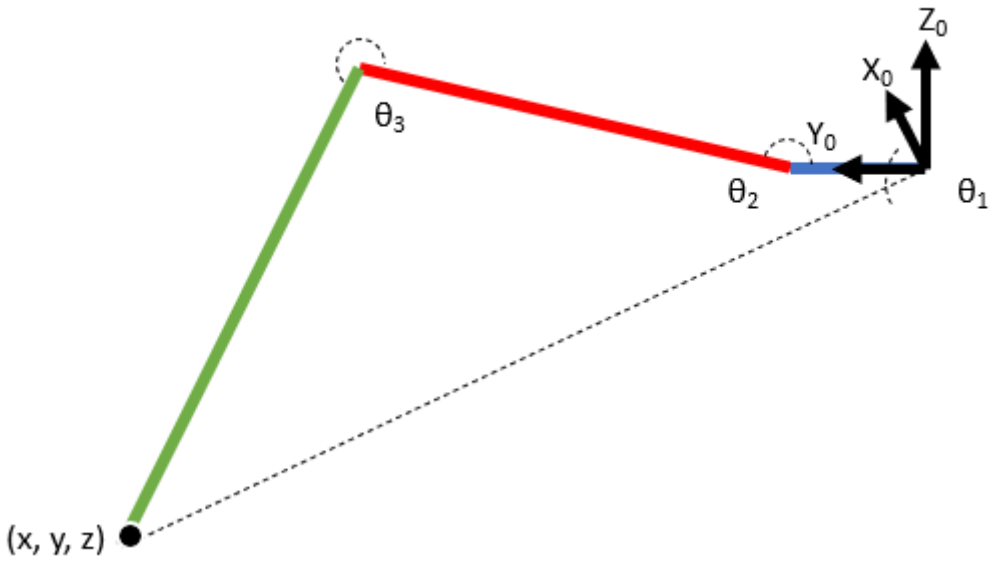
$$T=R_X.\ D_X.R_Z.D_Z \quad (5)$$

Este método ficará mais claro quando mostrado aplicado a perna do hexápode, uma explicação mais detalhada pode ser encontrada nas notas de aula de Lages (2000).

2.2.3 Cinemática Inversa

No problema de cinemática inversa devemos conseguir uma transformação que, dado uma posição no plano cartesiano x, y e z resulte em uma configuração de juntas θ_1, θ_2 e θ_3 que atinja aquela posição. Estes parâmetros podem ser observados na Figura 11.

Figura 11: Parâmetros para análise da cinemática inversa, com uma posição (x, y, z), obtemos os ângulos θ1, θ2 e θ3 para os servos motores.



Fonte: Elaborado pelo autor.

‘ Novamente há diferentes métodos de obter este resultado, porém neste caso utilizou-se o método analítico pela geometria do robô. Um exemplo de equações obtidas analiticamente foi realizado por Ghayour e Zareei (2012).

Além deste método, pode ser obtida a cinemática inversa de um robô utilizando meios numéricos, alguns destes meios podem ser feitos usando a matriz Jacobiana inversa, transposta, pseudo-inversa, método de Newton e Monte Carlo por exemplo. Como este não é o escopo deste trabalho, a referência e explicação para cada método pode ser observada em Aristidou e Lasenby (2009).

2.2.4 Geração de passadas

A geração de passadas é a maior parte do controle de um robô com pernas, em um hexápode codificar as passadas é difícil, pois é necessário coordenar o movimento de seis pernas simultaneamente. Muitas vezes robôs tem características diferentes, tornando difícil a passagem de um algoritmo para outro robô (LAROCHELLE, DASHNAW, PARKER, 2012).

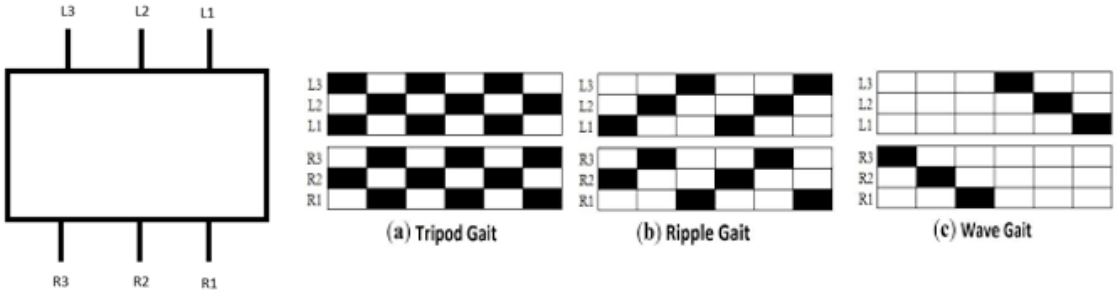
Existem diferentes tipos de controle de passada para robô, alguns propõe que o robô aprenda a passada através de algoritmos genéticos ou evolutivos, que utilizam princípios de biologia como aleatoriedade, mutação e evolução para buscar uma passada otimizada para o robô (BELTER, SKRZYPCZY´NSKI, 2010; LAROCHELLE, DASHNAW, PARKER, 2012).

Outros algoritmos focam no trânsito em terrenos acidentados, fazendo com que seja gerada uma trajetória para que a perna evite objetos no meio do caminho, com um certo custo de velocidade e de processamento que uma passada padronizada, há estudos que também focam na mudança do corpo do robô para o equilíbrio nestes movimentos, outros abordam diferentes tipos de passada natural, incluindo a adaptação quando há falha em alguma das pernas (FAIGI, ČÍŽEK, 2019; WANG, et al., 2010).

Dentre as marchas padronizadas para robôs, observa-se 3 tipos mais comuns *wave, ripple* e *tripod*. Na passada *wave* o robô movimenta primeiro as pernas do lado direito, uma por vez e depois do lado esquerdo, é a marcha mais estável, porém a mais lenta. Na marcha *ripple* movimenta-se primeira perna do lado esquerdo (L1) junto a última do lado direito (R3).e vice versa (R1 e L3), as pernas centrais se movem uma de cada vez entre estes ciclos, esta marcha é considerada de velocidade e estabilidade média (CAMPOS et al, 2010).

A última marcha, a *tripod* será a utilizada neste estudo, é a marcha mais rápida dentre as três, nela a primeira e a última perna de um lado (exemplo R1 e R3) e a central do outro (L2) se movimentam ao mesmo tempo, logo cada perna trabalha durante metade de um ciclo (CAMPOS et al, 2010)., as marchas citadas são mostradas na Figura 12.

Figura 12: Marchas do robô hexápode



Fonte: DARBHA, 2017 – Modificado

2.3 Matriz de palavras-chave

Para efeito de posicionamento desse trabalho perante o estado-da-arte, foram elencados trabalhos que relacionam as palavras-chaves utilizadas nessa pesquisa.

Tabela 1: Matriz de palavras-chave.

REFERÊNCIA	Robôs com pernas	Geração de passadas	Indústria 4.0	Robôs Móveis	Locomoção	Hexapóde	Cinemática
ARBHA, N. H., 2017	X					X	
ARISTIDOU, A.; LASENBY, J., 2009							X
BELTER, D.; SKRZYPCZYńSKI, P., 2010	X	X					
BLEDT et al, 2018	X						
CAMPOS, R.; MATOS, V.; SANTOS, C., 2010				X			
FAIGL, J.; ČÍŽEK , 2019					X		
GHAYOUR, M.; ZAREEI, A., 2011	X			X		X	X
GRIMES, J. A.; HURST, J. W., 2012	X						
GUREL, C. S., 2017						X	
HERMANN, M.; PENTEK, T.; OTTO, B, 2015			X				
KAGGERMANN, H.; WAHLSTER, W.; HELBIG, 2013			X				
KUINDERSMA S. et al, 2015	X			X			
LAGES W. F., 2019							X
LAROCHELLE K., DASHNAW S., PARKER G., 2012	X					X	
RUBIO F., VALERO F., LOPPIS-ALBERT C., 2019				X	X		
SAKURAI R., ZUCHI J. D., 2018			x				
NUNES V. A., 2020	**X**					**X**	**X**

Fonte: Elaborado pelo ator.

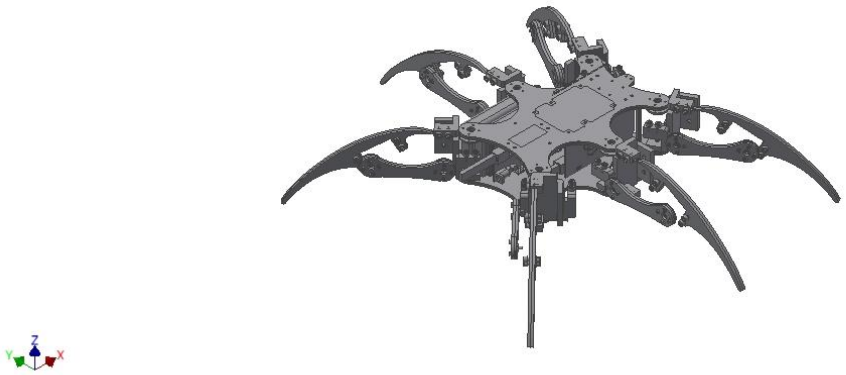
3 MATERIAIS E MÉTODOS

3.1 Dados utilizados

O robô hexápode não necessita necessariamente de um modelo CAD virtual para ser controlado, porém a sua utilização simplifica o controle e melhora a visualização, o robô construído teve sua modelagem e construção feita em trabalhos

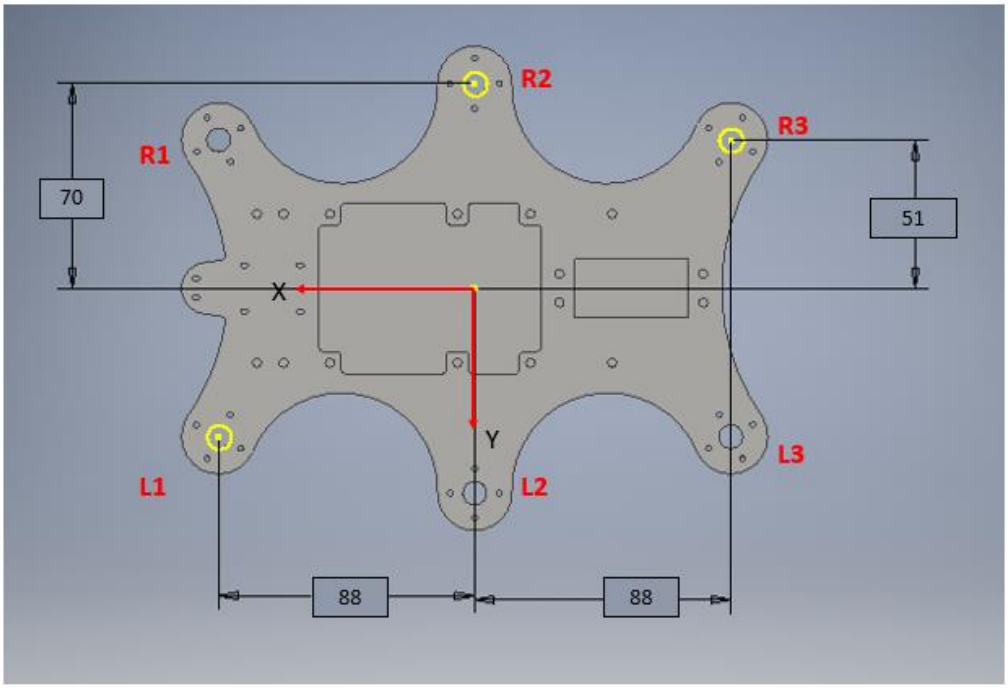
anteriores do grupo de pesquisa de robótica da Universidade Federal de São Carlos (UFSCar). A Figura 13 mostra o modelo CAD do robô e as Figuras 14 e 15 mostram algumas medidas tiradas de referência para a montagem do modelo cinemático.

Figura 13: Modelo CAD do robô hexápode, feito pelo grupo de estudo da UFSCar



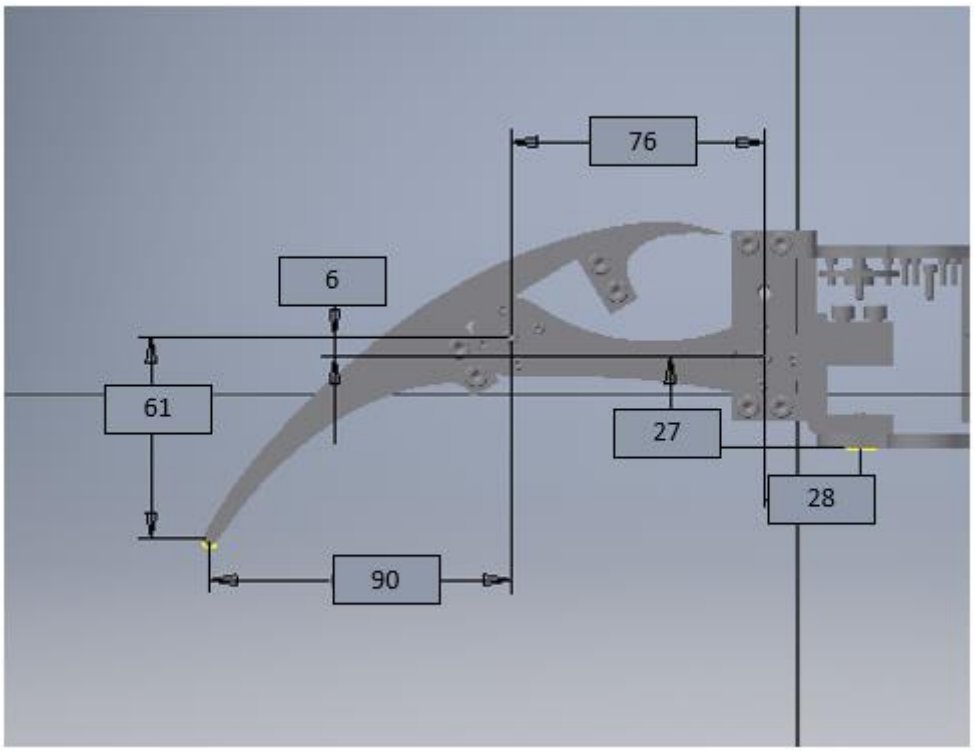
Fonte: Elaborado pelo ator.

Figura 14: Referência para a distância entre centro do robô e pernas.



Fonte: Elaborado pelo ator.

Figura 15: Referência para a medida da coxa, femur e tibia do robô.



Fonte: Elaborado pelo ator.

Utilizando as imagens de referência acima, pode-se utilizar as seguintes medidas para o modelo do robô, algumas medidas foram aproximadas para simplificação de cálculo.

Tabela 2: Tabela de posições da posição das pernas.

Perna	X (mm)	Y (mm)
R1	88	-51
R2	0	-70
R3	-88	-51
L1	88	51
L2	0	70
L3	-88	51

Fonte: Elaborado pelo ator.

Tabela 3: Comprimento da coxa, femur e tibia para modelo.

Segmento	Comprimento (mm) – aproximado
Coxa	30
Femur	80
Tíbia	110

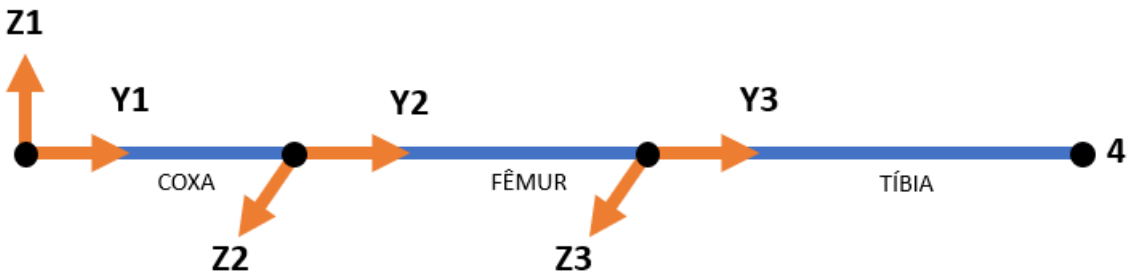
Fonte: Elaborado pelo ator.

3.2 Obtendo o modelo de cinemática direta do robô

Conforme citado na revisão bibliográfica, utilizou-se o método de Denavit-Hartenberg (DH) para obter a cinemática direta do robô, o método e a análise no Matlab apresentados são semelhantes ao trabalho de Gurel (2017).

Nele temos o eixo Z apontado para o eixo de movimento da junta e o eixo X apontado na direção da perna do robô, os segmentos da perna foram alinhados para simplificação do cálculo e o eixo X foi substituído por Y para coincidir com as coordenadas adotadas para o robô. A Figura 16 mostra a posição dos eixos.

Figura 16: Posição dos eixos para método DH.



Fonte: Elaborado pelo ator.

Novamente os quatro movimentos para cálculo da cinemática direta são:

- Translação em d_i ao longo de Z_i;
- Rotação em θ_{i-1} em torno de Z_i, alinhando X_i com X_{i-1};
- Translação em a_{i-1} ao longo de X_i, tornando as origens coincidentes;
- Rotação em α_{i-1} ao em torno de X_i, alinhando Z_i com Z_{i-1}.

A tabela abaixo mostra os parâmetros obtidos (β1, β2, β3 são as juntas do robô):

Tabela 4: Tabela de parâmetros DH.

i	**d_i**	**θ_i**	**a_i**	**α_i**
1	0	β1	30	90
2	0	β2	80	0
3	0	β3	110	0

Fonte: Elaborado pelo ator.

Com estes parâmetros é possível calcular a transformação entre juntas

$$T_{i,i+1}=R_X.\ D_X.R_Z.D_Z \quad (6)$$

A matriz de transformação entre o ponto 4 e o ponto 1 é dada pela transformação entre as juntas anteriores, logo:

$$T_{14}=T_{12}.T_{23}.T_{34} \qquad (7)$$

Executando todos esses cálculos, obteve-se a seguinte matriz de transformação:

$$T_{14}=\begin{bmatrix} T_{11} & T_{12} & T_{13} & T_{14} \\ T_{21} & T_{22} & T_{23} & T_{24} \\ T_{31} & T_{32} & T_{33} & T_{34} \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (8)$$

$$\mathbf{T_{11}} = \cos(\beta1)\,.\,\cos(\beta2)\,.\,\cos\left(\beta3-\frac{\pi}{2}\right) - \cos(\beta1)\,.\sen(\beta2).\sen\left(\beta3-\frac{\pi}{2}\right) \qquad (9)$$

$$\mathbf{T_{12}} = -\cos(\beta1)\,.\,\cos(\beta2)\,.\sen\left(\beta3-\frac{\pi}{2}\right) - \cos(\beta1)\,.\sen(\beta2).\cos\left(\beta3-\frac{\pi}{2}\right) \qquad (10)$$

$$\mathbf{T_{13}} = \sen(\beta1) \qquad (11)$$

$$\mathbf{T_{14}} = 110.\cos(\beta1).\cos(\beta2).\cos\left(\beta3-\frac{\pi}{2}\right)-110.\cos(\beta1).\operatorname{sen}(\beta2).\operatorname{sen}\left(\beta3-\frac{\pi}{2}\right)+80.\cos(\beta1).\cos(\beta2)+30.\operatorname{sen}(\beta1) \quad (12)$$

$$\mathbf{T_{21}} = \operatorname{sen}(\beta1).\cos(\beta2).\cos\left(\beta3-\frac{\pi}{2}\right)-\operatorname{sen}(\beta1).\operatorname{sen}(\beta2).\operatorname{sen}\left(\beta3-\frac{\pi}{2}\right) \quad (13)$$

$$\mathbf{T_{22}} = -\operatorname{sen}(\beta1).\cos(\beta2).\operatorname{sen}\left(\beta3-\frac{\pi}{2}\right)-\operatorname{sen}(\beta1).\operatorname{sen}(\beta2).\cos\left(\beta3-\frac{\pi}{2}\right) \quad (14)$$

$$\mathbf{T_{23}} = -\cos(\beta1) \quad (15)$$

$$\mathbf{T_{24}} = 110.\operatorname{sen}(\beta1).\cos(\beta2).\cos\left(\beta3-\frac{\pi}{2}\right)-110.\operatorname{sen}(\beta1).\operatorname{sen}(\beta2).\operatorname{sen}\left(\beta3-\frac{\pi}{2}\right)+80.\operatorname{sen}(\beta1).\cos(\beta2)+30.\cos(\beta1) \quad (16)$$

$$\mathbf{T_{31}} = \operatorname{sen}(\beta2).\cos\left(\beta3-\frac{\pi}{2}\right)+\cos(\beta2).\operatorname{sen}\left(\beta3-\frac{\pi}{2}\right) \quad (17)$$

$$\mathbf{T_{32}} = -\operatorname{sen}(\beta2).\operatorname{sen}\left(\beta3-\frac{\pi}{2}\right)+\cos(\beta2).\cos\left(\beta3-\frac{\pi}{2}\right) \quad (18)$$

$$\mathbf{T_{33}} = 0 \quad (19)$$

$$\mathbf{T_{34}} = 110.\operatorname{sen}(\beta2).\cos\left(\beta3-\frac{\pi}{2}\right)+110.\cos(\beta2).\operatorname{sen}\left(\beta3-\frac{\pi}{2}\right)+80.\operatorname{sen}(\beta2) \quad (20)$$

Neste trabalho nos interessaremos pelas posições (2,4), (1,4) e (3,4), estas equações indicam as posições X, Y e Z da ponta do robô dado os ângulos da junta β1, β2 e β3. Lembrando que o X e Y estão invertidos da posição usual por conta da convenção adotada no começo do projeto.

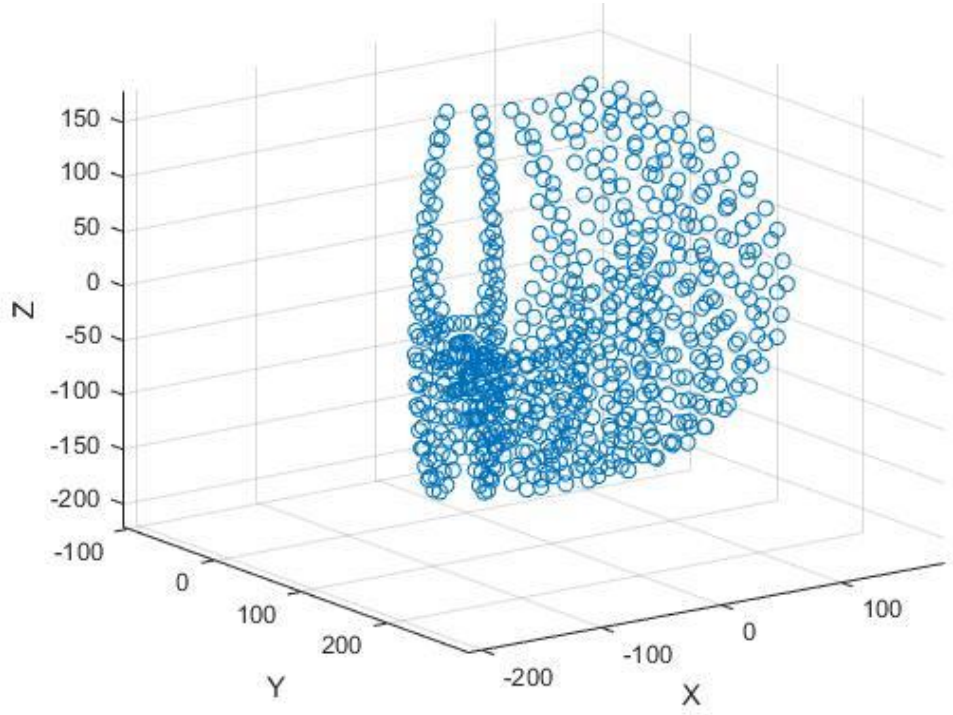
$$X = 110.\operatorname{sen}(\beta1).\cos(\beta2).\cos\left(\beta3-\frac{\pi}{2}\right)-110.\operatorname{sen}(\beta1).\operatorname{sen}(\beta2).\operatorname{sen}\left(\beta3-\frac{\pi}{2}\right)+80.\operatorname{sen}(\beta1).\cos(\beta2)+30.\cos(\beta1) \quad (21)$$

$$Y = 110.\cos(\beta1).\cos(\beta2).\cos\left(\beta3-\frac{\pi}{2}\right)-110.\cos(\beta1).\operatorname{sen}(\beta2).\operatorname{sen}\left(\beta3-\frac{\pi}{2}\right)+80.\cos(\beta1).\cos(\beta2)+30.\operatorname{sen}(\beta1) \quad (22)$$

$$Z = 110.\operatorname{sen}(\beta2).\cos\left(\beta3-\frac{\pi}{2}\right)+110.\cos(\beta2).\operatorname{sen}\left(\beta3-\frac{\pi}{2}\right)+80.\operatorname{sen}(\beta2) \quad (23)$$

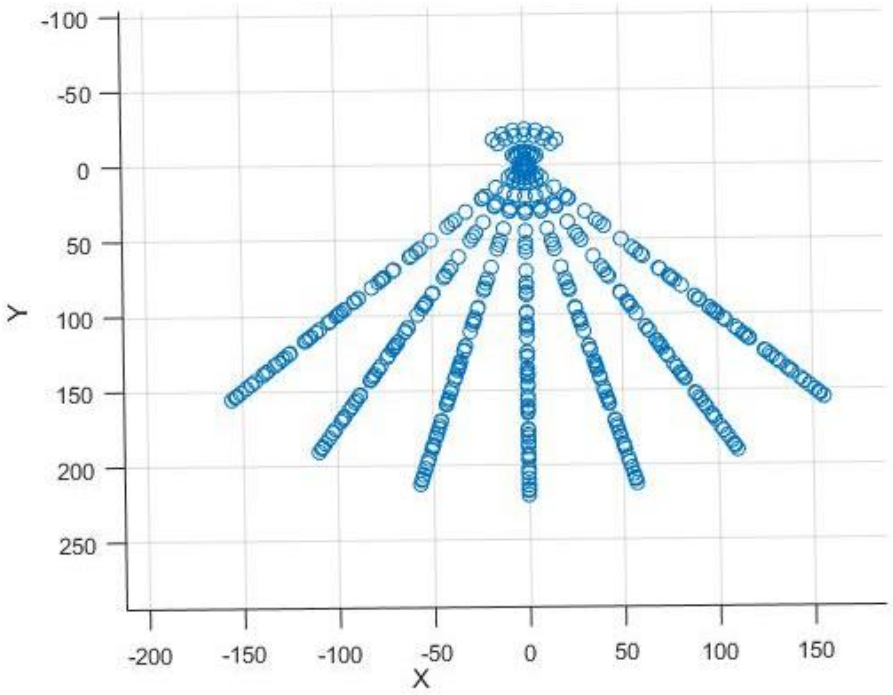
Com esta modelagem é possível observar o espaço de trabalho das pernas do robô, sua área alcançável, utilizando iterações com vários ângulos chegou-se no seguinte estudo de capacidade, mostrado nas Figuras 17, 18 e 19. Tal resultado é semelhante ao obtido por Gurel (2017). As unidades dos eixos estão em milímetros. O código para obtenção dos dados está contido no Apêndice A e foi feito utilizando o MATLAB. Os limites utilizado para as juntas inferior e superior foram de 45°, para representar melhor o movimento de uma pata no caso real.

Figura 17: Espaço de trabalho de uma perna, vista geral.



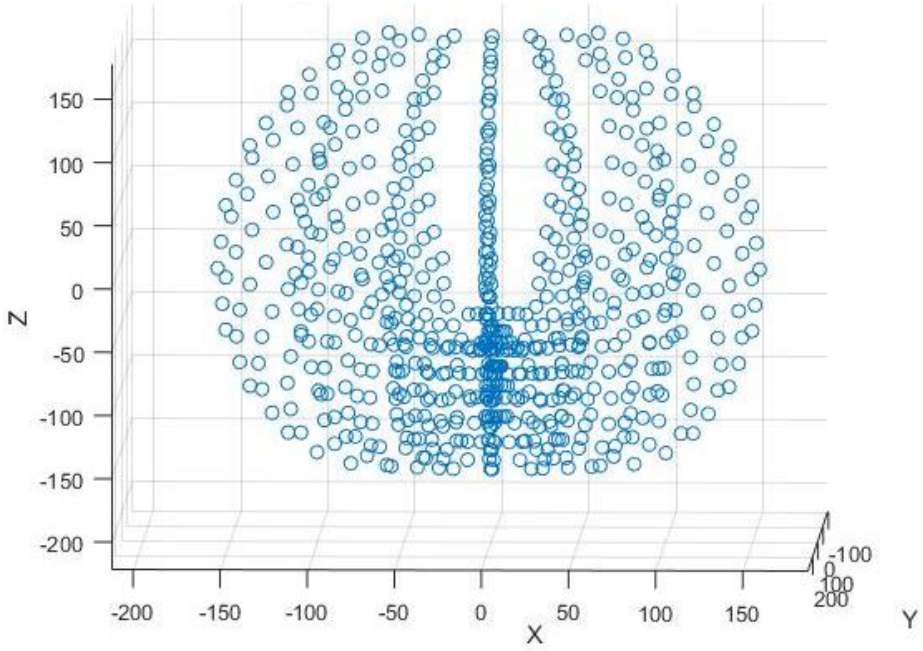
Fonte: Elaborado pelo ator.

Figura 18: Espaço de trabalho de uma perna, vista superior.



Fonte: Elaborado pelo ator.

Figura 19: Espaço de trabalho de uma perna, vista frontal.

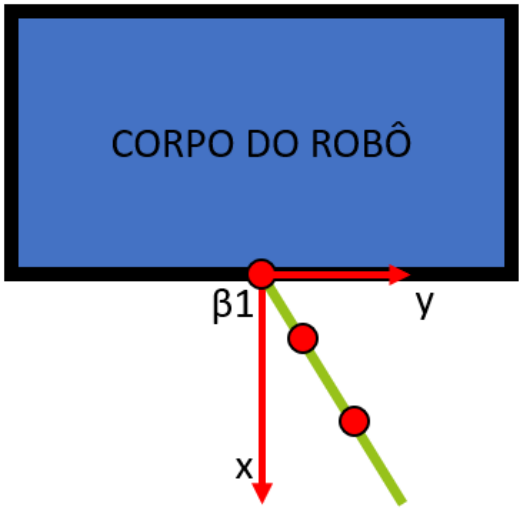


Fonte: Elaborado pelo ator.

3.3 Obtendo o modelo de cinemática inversa

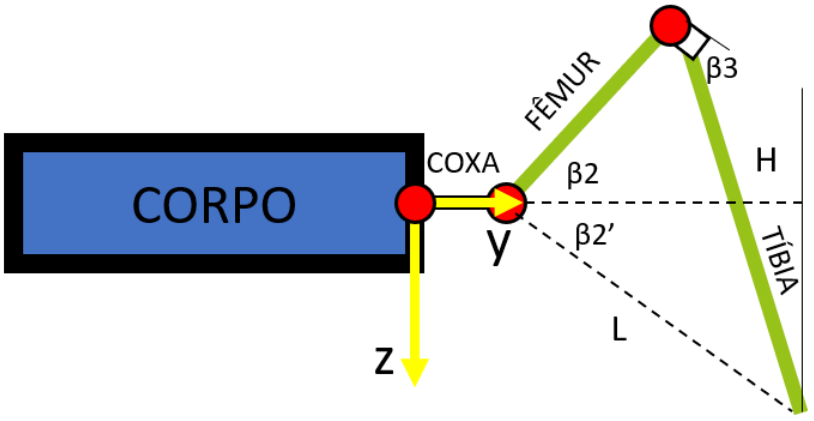
O modelo cinemático do robô foi obtido por método analítico, utilizando geometria, as imagens de referência para cálculo foram as Figuras 20 e 21.

Figura 20: Vista esquemática superior para cálculo da cinemática inversa.



Fonte: Elaborado pelo ator.

Figura 21: Vista esquemática lateral para cálculo da cinemática inversa.



Fonte: Elaborado pelo ator.

As equações de posição das juntas em relação as coordenadas X, Y e Z foram mostradas abaixo. *Coxa*, *Femur* e *Tibia* são as medidas de comprimento da pata, *x*,

y e *z* são as coordenadas e β1, β2, β3 são os ângulos das juntas. L é o comprimento entre a junta 2 e a ponta da para e H é a projeção do seguimento L no plano XY. As medidas estão em milímetros:

$$H = \sqrt{x^2+y^2} - Coxa \quad (24)$$

$$L = \sqrt{H^2+z^2} \quad (25)$$

$$\beta 1 = \tan^{-1}\left(\frac{x}{y}\right) \quad (26)$$

$$\beta 2 = \cos^{-1}\left(\frac{Tibia^2 - L^2 - Femur^2}{-2.L.Femur} - \tan^{-1}\left(\frac{z}{H}\right)\right) \quad (27)$$

$$\beta 3 = \frac{\pi}{2} - \cos^{-1}\left(\frac{L^2 - Femur^2 - Tibia^2}{-2.L.Femur}\right) \quad (28)$$

3.4 Criando o modelo do ROS

Com esses dados iniciais foi possível desenvolver o modelo do robô no *ROS*, todos os arquivos e códigos do desenvolvimento podem ser encontrados no repositório GIT em Nunes (2020).

A primeira etapa é construir o modelo cinemático do robô, com os arquivos 3D em STL e as dimensões obtidas do modelo CAD é possível criar um arquivo URDF (*Unified Robot Description Format*) do robô.

Arquivo URDF são especificações XML (*eXtensible Markup Language*) usadas, entre outras utilidades, para descrever robôs em estruturas de árvore. Ela cobre a cinemática, dinâmica, representação visual e modelo de colisão do robô

Neste arquivo estão inseridos as posições e orientações de todos os elos e juntas do robô, utilizou se um arquivo XACRO (*XML Macro*), para modelar uma perna do robô, este arquivo funciona como uma macro e esta perna modelada foi repassada em todas as posições, gerando um arquivo URDF completo. No Apêndice B foram incluídos os arquivos XACRO do robô.

A partir destes arquivos é possível criar um arquivo .launch no ROS, este arquivo executa ações como executar nós, aplicações e colocar parâmetros para a simulação visual, o arquivo *launch* pode ser observado nos Apêndices E e F.

Dentre os diversos pacotes para o ROS os mais relevantes para executar o experimento foram os seguintes:

- URDF: pacote com os parâmetros utilizados para especificar em XML sensores, modelos e cenários. É utilizado para descrição geral do robô.
- XACRO: permite construir arquivos mais curtos de descrição do robô e transformá-los em URDF com macros.
- robot_state_publisher: permite receber, por meio do joint_state_publisher o estado das juntas (joint_space) do robô e publicar para o pacote tf. Ele tem como entrada a posição das juntas e devolve a pose 3D do robô utilizando o modelo cinemático.
- joint _space_publisher: é o pacote responsável por verificar no modelo URDF as juntas não fixadas e publicar a posição enviada a elas ao estado das juntas (joint_states), por meio da interface visual. Foi o pacote utilizado para enviar as juntas no modelo de passada.

As Figuras 22 e 23 mostra o modelo de execução do robô para transmitir a passada, utilizando a ferramenta rqt_graph, que mostra todos os tópicos e nós utilizados durante a simulação visual. A primeira (Figura 22) mostra a simulação usando a interface gráfica, onde podemos controlar a junta manualmente. A segunda (Figura 23) mostra a criada neste experimento onde foi criado o nó “control”, que utiliza as equações de cinemática inversa e direta para publicar as juntas do robô. As elipses indicam os nós, que são os arquivos inseridos no pacote que podem publicar os tópicos, que estão representados como quadrados, n_rviz é o nó do programa visualizador do ROS, o já citado Rviz.

Figura 22: Esquema de comunicação utilizando a interface gráfica para as juntas.

Fonte: Elaborado pelo ator.

Figura 23: Esquema de comunicação utilizando o modelo cinemático criado.

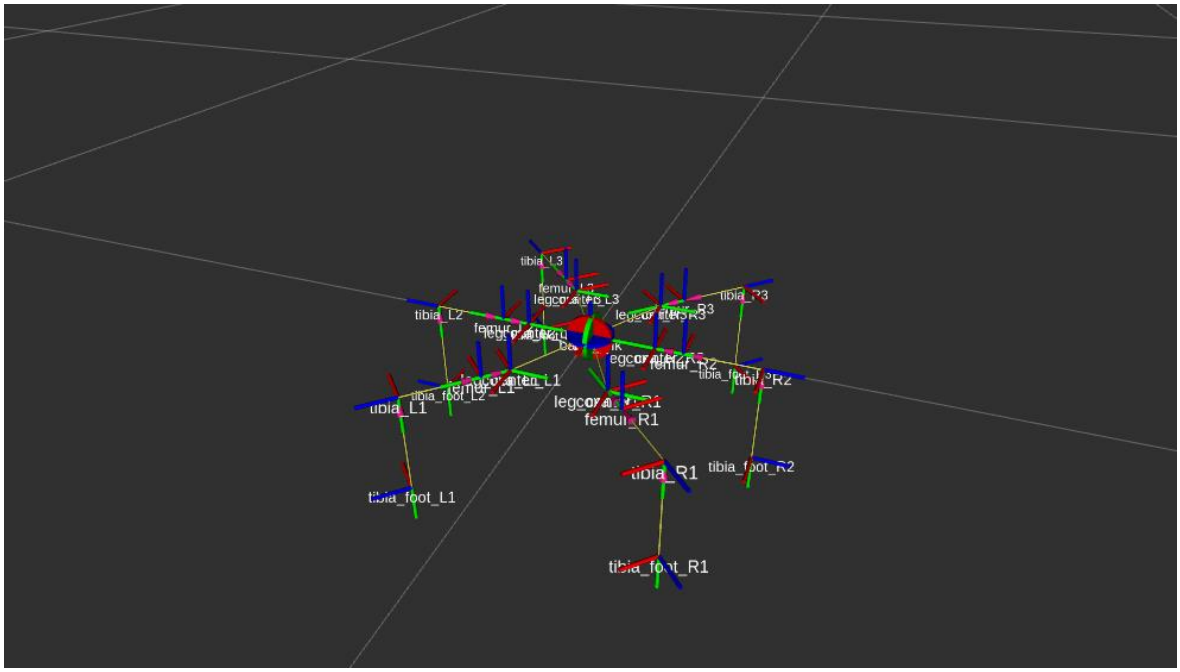
Fonte: Elaborado pelo ator.

O desenvolvimento detalhado de como o projeto foi feito dentro da plataforma, explicando a estrutura de arquivos e pacotes utilizados está presente no Apêndice G deste trabalho.

4 RESULTADOS E DISCUSSÕES

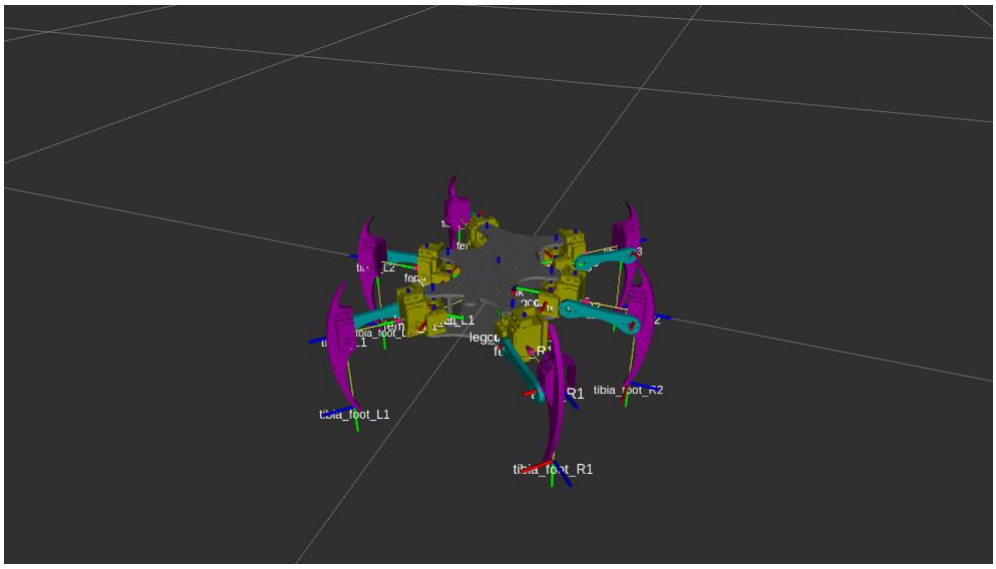
Com os dados geométricos do robô e o arquivo URDF criado, foi possível gerar a seguinte visualização, observada nas Figuras 25 e 26.

Figura 24: Representação do robô sem modelo 3D.



Fonte: Elaborado pelo ator.

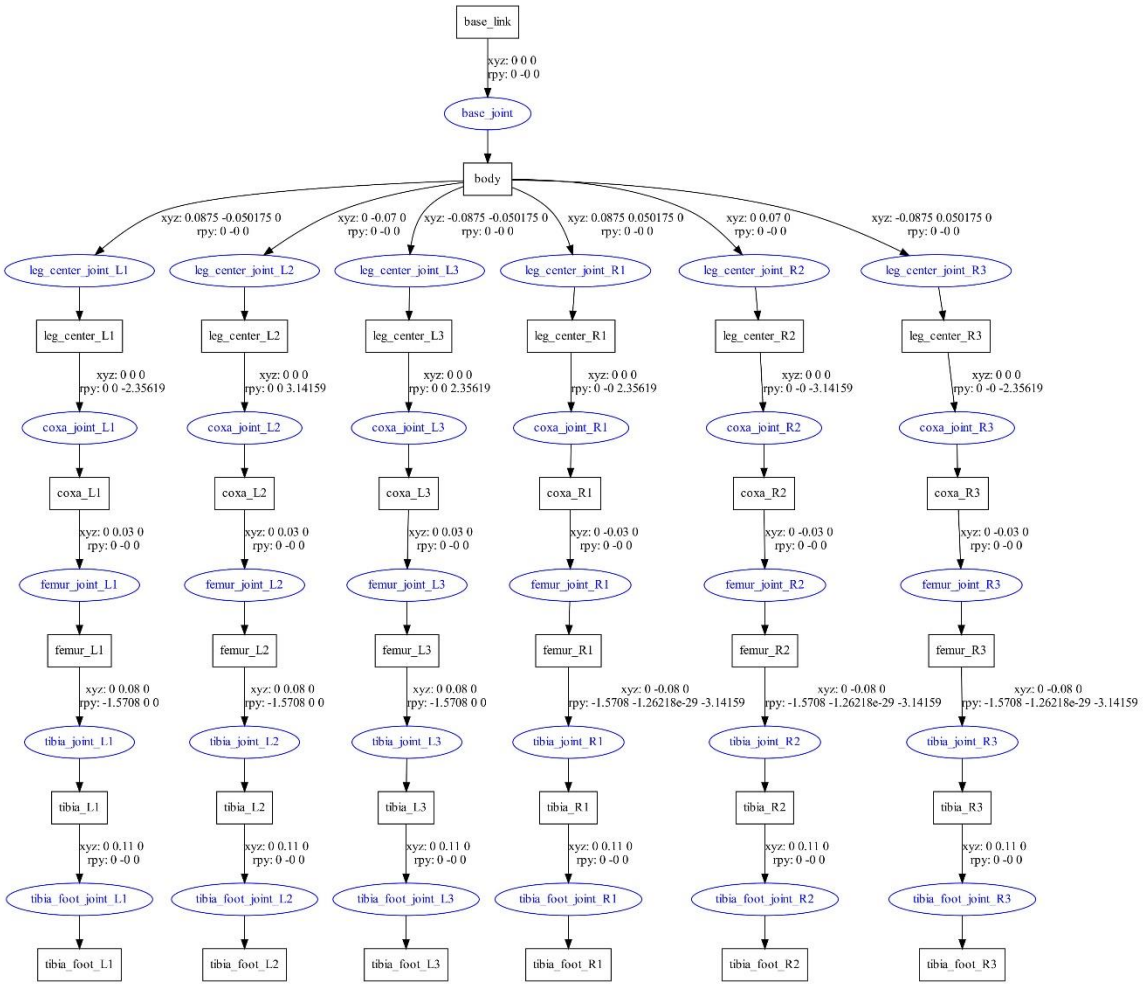
Figura 25: Representação do robô com modelo 3D.



Fonte: Elaborado pelo ator.

Pode-se também com a ferramenta Graphiz, obter a árvore cinemática do robô, que mostra a relação entre as juntas, suas respectivas posições e orientações (Figura 27).

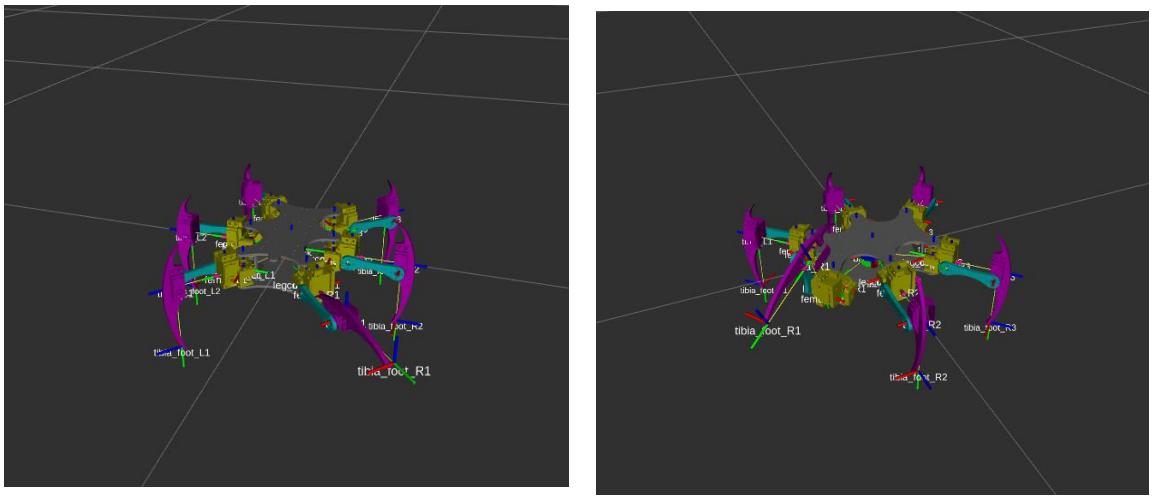
Figura 26: Representação do robô na árvore cinemática.



Fonte: Elaborado pelo ator.

Com a visualização feita pode-se tentar executar o controle do robô por meio das equações desenvolvidas, a Figura 28 mostra as pernas do robô se movimentando para algumas coordenadas arbitrárias utilizando o algoritmo de cinemática inversa.

Figura 27: Perna do robô utilizando os algoritmos de cinemática inversa.



Fonte: Elaborado pelo ator.

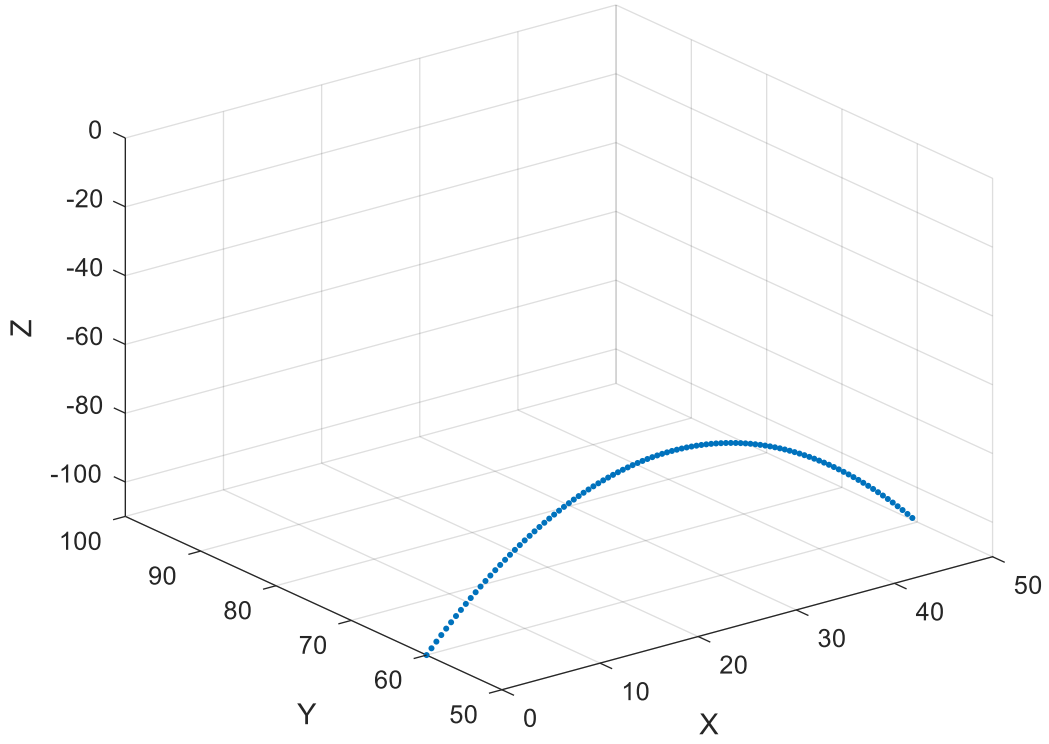
.

Foi utilizada uma função de segundo grau para realizar a passada mostrada abaixo, com *w* sendo comprimento da passada e *h* a altura (ambos em milímetro), há diferentes métodos para obter a passada porém para fins de demonstração esta função é adequada. A distância entre o base da perna e a ponta se manteve inalterada. Os códigos de controle do robô e os módulos criados em Python para ROS estão nos Apêndices C e D.

$$z = \left(\frac{4h}{w^2}\right)x^2 + \left(\frac{4h}{w}\right)x \quad (29)$$

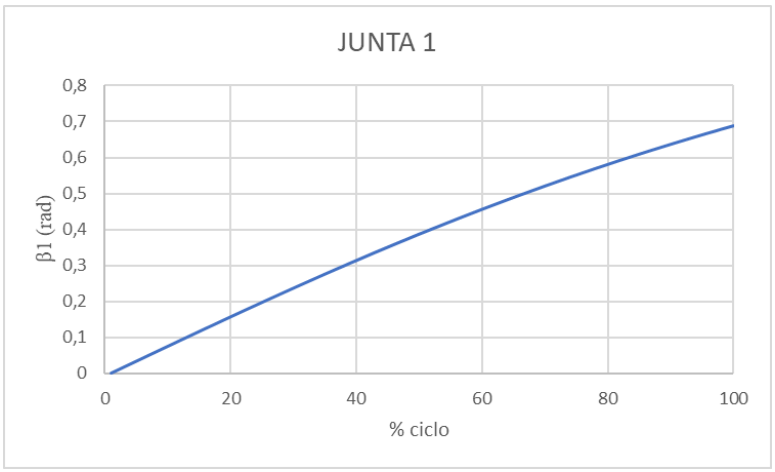
Obteve-se os seguintes estados de junta e posição da ponta da pata relacionada a base da perna exibidas para uma das pernas, a lateral direita (R1). Utilizou-se h = 40 mm, w = 50mm e distância em y de 60mm entre base da perna e ponta, estes resultados podem ser observados nas Figura 29 a 32. As posições foram relacionadas com a porcentagem do ciclo de passada, na falta da variável tempo presente em um modelo dinâmico.

Figura 28: Conjunto de posições X, Y e Z em uma passada (em milímetros).



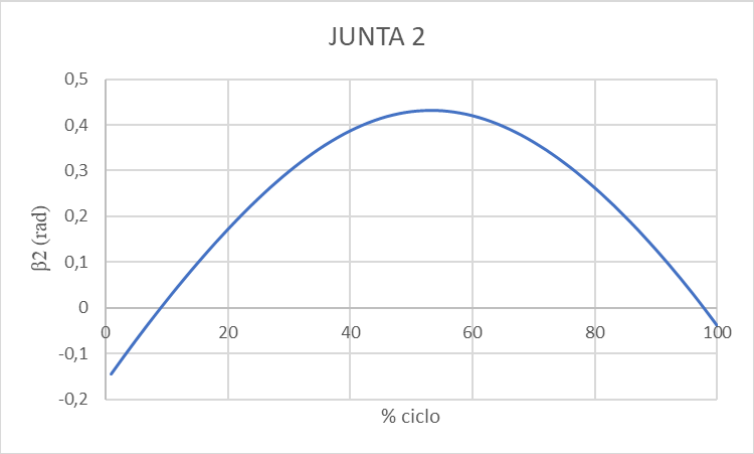
Fonte: Elaborado pelo ator.

Figura 29: Posição angular da primeira junta (coxa).



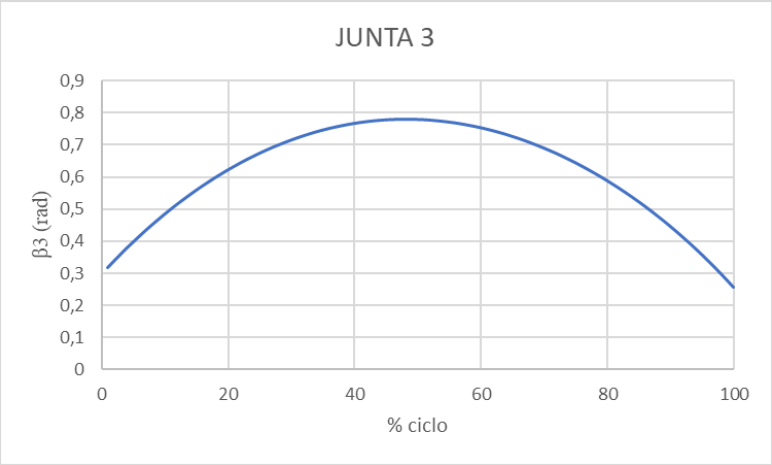
Fonte: Elaborado pelo ator.

Figura 30: Posição angular da segunda junta (femur).



Fonte: Elaborado pelo ator.

Figura 31: Posição angular da terceira junta (tibia).



Fonte: Elaborado pelo ator.

O *ROS – Robot Operating System* é uma ferramenta com grande potencial e seu uso na área acadêmica e profissional pode ser ampliada no campo de engenharia mecânica ou mecatrônica, principalmente no contexto de I4.0.

Buscou-se com o modelo de Denavit-Hartenberg um modelo de cinemática confiável para controlar e estudar o robô, junto a obtenção das equações de cinemática inversa de forma analítica.

Ambos mostraram-se eficazes para resolver o problema, conseguindo gerar posições para juntas e como observado no estudo de passada, movimentar o robô pelo trajeto programado. Por questões de incerteza e discrepâncias do modelo em

comparação ao mundo real, o comportamento indicado no gráfico não é o mesmo do robô físico, onde esses modelos serão aplicados em trabalhos futuros.

Os resultados também foram compatíveis com a literatura a Figura 35 mostra o espaço do trabalho de outro robô hexápode com outras dimensões e o resultado obtido é semelhante ao obtido no estudo de espaço de trabalho (vide Figura 17)

Para a execução da trajetória pedida, como esperado a posição de x é uma reta que vai avançando conforme passa o tempo de passada até atingir o comprimento selecionado, a posição y se mantém constante após a alteração da posição inicial para sua posição destino e a posição de z varia de forma parabólica, subindo a altura selecionada e voltando a zero após o final do ciclo.

A posição da junta também foi a esperada, com um crescimento do ângulo de forma linear na primeira junta, indicando uma revolução que leva a perna para frente, as outras duas juntas começam crescendo indicando uma retração da perna e depois vão retornando a posição próxima a original para se fixarem novamente no chão, dando espaço a outras pernas se movimentarem.

Algumas configurações se mostraram inadequadas para o robô simulado, mesmo não sendo o foco deste estudo, ao aumentar a altura de passada a um valor muito alto o robô acaba jogando a perna em uma posição muito acima da indicada, isso se deve a erros do sistema de controle do robô e a falta de um modelo mais preciso do mesmo, que necessitaria a configuração de motores e pesos, ficando indicada a trabalhos futuros.

Destaca-se como ponto importante a frequência na qual enviamos os dados ao controlador, uma frequência baixa faz com que o receptor do sinal não consiga atualizar seus dados para enviar às juntas, causando erro no programa ou falhas de movimentação, bem como frequências altas exigem equipamentos mais sofisticados e geram uma quantidade maior de dados a serem processados.

Neste estudo inicialmente utilizou-se o valor de 10Hz mas ele se mostrou baixo, não conseguindo gerar a visualização no programa. Aumentando a taxa de transmissão para 50Hz resolveu este problema.

Com este modelo é possível integrar mais funcionalidades do ROS como, módulos de percepção visual, biblioteca de transformações lineares e integração com *hardware.* Este é um bom ponto de melhoria de projeto, bem como um algoritmo de geraçãõ de passadas mais eficiente e adequado para uso real.

6 CONCLUSÕES/CONSIDERAÇÕES FINAIS

O trabalho realizou seu objetivo principal de modelar a cinemática do robô hexápode e aplicá-la dentro da plataforma ROS, podendo este documento servir de base para futuros desenvolvimentos com este tipo de robô.

Apesar destes fatos, há muitas áreas de melhoria neste trabalho, uma delas seria um uso extensivo da biblioteca pré-existente no ROS “tf”, para detectar posições dos elos e da ponta da perna, bem como realizar transformações de coordenadas entre diferentes referências, o que permitiria utilizar melhor os recursos instalados junto com a plataforma.

O desenvolvimento de equações para controle da posição do corpo também poderia ser desenvolvido, permitindo o robô se nivelar automaticamente com o ambiente, para melhorar sua estabilidade ou posicionar melhor seus sensores ou câmera.

REFERÊNCIAS

ARBHA, Naga Harika. **An Optimization Strategy For Hexapod Gait Transition**. 2017. 48 f. Dissertação (Mestrado) - Curso de Master Of Science In Electrical Engineering, Wright State University, Dayton, 2017.

ARISTIDOU, Andreas; LASENBY, Joan. **Inverse Kinematics: a review of existing techniques and introduction of a new fast iterative solver**. Cambridge: Cambridge University Engineering Department, 2009.

BELTER, Dominik; SKRZYPCZYńSKI, Piotr. A biologically inspired approach to feasible gait learning for a hexapod robot. **International Journal Of Applied Mathematics And Computer Science**, Poznań, v. 20, n. 1, p. 69-84, 1 mar. 2010. http://dx.doi.org/10.2478/v10006-010-0005-7.

BLEDT, G. et al., "MIT Cheetah 3: Design and Control of a Robust, Dynamic Quadruped Robot," ***2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)***, Madrid, 2018, pp. 2245-2252, doi: 10.1109/IROS.2018.8593885.

CAMPOS, Ricardo; MATOS, Vitor; SANTOS, Cristina**. Hexapod locomotion: a nonlinear dynamical systems approach**. In: Iecon 2010 - 36Th Annual Conference On Ieee Industrial Electronics Society, Glendale, nov. 2010. http://dx.doi.org/10.1109/iecon.2010.5675454.

FAIGL, Jan & ČÍŽEK, Petr. Adaptive locomotion control of hexapod walking robot for traversing rough terrains with position feedback only**. Robotics and Autonomous Systems**, 2019. 116. 10.1016/j.robot.2019.03.008.

GHAYOUR, Mostafa; ZAREEI, Amir. Direct Kinematic Analysis of a Hexapod Spider-Like Mobile Robot. **Advanced Materials Research**, [S.L.], v. 403-408, p. 5053-5060, nov. 2011. Trans Tech Publications, Ltd.. http://dx.doi.org/10.4028/www.scientific.net/amr.403-408.5053.

GRIMES, Jesse A.; HURST, Jonathan W. THE DESIGN OF ATRIAS 1.0 A UNIQUE MONOPOD, HOPPING ROBOT. **Adaptive Mobile Robotics**, p. 548-554, 18 jul. 2012. WORLD SCIENTIFIC. http://dx.doi.org/10.1142/9789814415958_0071

GUREL, Canberk Suat. **Hexapod Modelling, Path Planning, and Control.** University of Manchester, Maryland: [*s. n.*], 2017.

HERMANN, M.; PENTEK, T.; OTTO, B. **Design Principles for Industrie 4.0 Scenarios: A Literature Review.** [S.l.: s.n.], 2015.

IVAN, Biriuk. **Hexapod-робот под управлением ROS [Robô hexapod controlado por ROS]**. 2014. Disponível em: https://habr.com/ru/post/225845/. Acesso em: 01 set. 2019.

KAGGERMANN, H.; WAHLSTER, W.; HELBIG. **Recommendations for implementing the strategic initiative INDUSTRIE 4.0** (acatech STUDY), Munich, 2013.

KUINDERSMA, Scott *et al*. Optimization-based locomotion planning, estimation, and control design for the atlas humanoid robot. **Autonomous Robots**, [S.L.], v. 40, n. 3, p. 429-455, 31 jul. 2015. Springer Science and Business Media LLC. http://dx.doi.org/10.1007/s10514-015-9479-3.

LAGES, Walter Fetter. **Parâmetros de Denavit-Hartenberg**. 2019. Disponível em: http://www.ece.ufrgs.br/~fetter/eng10026/dh.pdf. Acesso em: 06 jul. 2020.

LAROCHELLE, Karen; DASHNAW, Sarah; PARKER, Gary. **Gait Evolution for a Hexapod Robot**, Connecticut College, New London, 2012.

MACHADO, José A. Tenreiro; SILVA, Manuel F. **An Overview of Legged Robots.** In: Mme 2006 – International Symposium On Mathematical Methods In Engineering, Ankara, v. 1, n. 1, p. 1-1, abr. 2016.

NUNES, Victor. **Hexapod Project**. 2020. Disponível em: https://github.com/victoran97/hexapod-project . Acesso em: 28 nov. 2020.

RUBIO, Francisco; VALERO, Francisco; LLOPIS-ALBERT, Carlos. A review of mobile robots: Concepts, methods, theoretical framework, and applications. **International Journal Of Advanced Robotic Systems**, Valencia, v. 1, n. 20, p. 70-84, 16 abr. 2019.

SAKURAI, Ruudi; ZUCHI, Jederson Donizete. A REVOLUÇÕES INDUSTRIAIS ATÉ A INDUSTRIA 4.0. **Revista Interface Tecnológica**, São Paulo, v. 15, n. 2, p. 480-491, 30 dez. 2018. http://dx.doi.org/10.31510/infa.v15i2.386.

SCHUH, G., ANDERL, R., GAUSEMEIER J., ten HOMPEL, M., WAHLSTER, W. (Eds.): **Industrie 4.0 Maturity Index. Managing the Digital Transformation of Companies** (acatech STUDY), Munich: Herbert Utz Verlag 2017.

WANG, Zhiying et al.. Locomotion Analysis of Hexapod Robot. **Climbing And Walking Robots**, Cambridge, v. 6, n. 28, p. 893-907, 1 mar. 2010. Http://dx.doi.org/10.5772/8822.

APÊNDICE A – Código para gerar o espaço de trabalho da perna (MATLAB)

```
X = [];
Y = [];
Z = [];

t1 = 0;
t2 = 0;
t3 = 0;
i = 1;
for t1 = -pi/4:pi/12:pi/4
    for t2 = -pi/4:pi/12:pi/2
        for t3 = -pi/4:pi/12:pi/2

        X(i) = 110*cos(t1)*cos(t2)*cos(t3-pi/2) - 110*cos(t1)*sin(t2)*sin(t3-pi/2) +
80*cos(t1)*cos(t2) + 30*cos(t1);
        Y(i) = 110*sin(t1)*cos(t2)*cos(t3-pi/2) - 110*sin(t1)*sin(t2)*sin(t3-pi/2) +
80*sin(t1)*cos(t2) + 30*sin(t1);
        Z(i) = 110*sin(t2)*cos(t3-pi/2) + 110*cos(t2)*sin(t3-pi/2) + 80*sin(t2);
        i=i+1;

        end
    end
end

scatter3(X,Y,Z);
% Create xlabel
xlabel({'X'},'FontSize',11);

% Create zlabel
zlabel({'Z'},'FontSize',11);

% Create title
title({'ESPAÇO DE TRABALHO DE UMA PERNA'},'FontSize',11);

% Create ylabel
ylabel({'Y'},'FontSize',11);

view(axes1,[-37.5 30]);
grid(axes1,'on');
```

APÊNDICE B – Modelo XACRO do robô (XACRO)

```
<?xml version="1.0"?>
<robot xmlns:xacro="http://ros.org/wiki/xacro" name="hexapod">

    <!-- Robot Frame -->

        <link name="base_link"/>

        <joint name="base_joint" type="fixed">
            <parent link="base_link"/>
            <child link="body"/>
            <origin xyz="0 0 0" rpy="0 0 0"/>
        </joint>

        <link name="body">

            <inertial>
                <mass value="0.8"/>
                <inertia ixx="0.007"  ixy="0"  ixz="0" iyy="0.012" iyz="0" izz="0.010"
/>
            </inertial>

            <visual>
                <origin xyz="0 0 -0.027" rpy="0 0 0"/>
                <geometry>
                        <mesh
filename="package://hexapod_description/meshes/BODY.stl"/>
                </geometry>
                <material name="grey">
                        <color rgba="0.5 0.5 0.5 1"/>
                </material>
            </visual>

            <collision>
                <origin xyz="0 0 -0.027" rpy="0 0 0"/>
                <geometry>
                    <mesh
filename="package://hexapod_description/meshes/BODY_SIMPLIFIED.stl"/>
                </geometry>
            </collision>

        </link>

    <!-- Pi (used for angle calculations)-->
    <xacro:property name="pi" value="3.1415926535897931"/>

    <!-- Propriedade das Juntas [Joint Properties]-->
    <xacro:property name="joint_lower_limit" value="-${1.5}"/>
    <xacro:property name="joint_upper_limit" value="${1.5}"/>
    <xacro:property name="joint_effort" value="10000"/>
    <xacro:property name="joint_velocity" value="100"/>

    <!--Macro for legs-->
    <xacro:macro name="leg" params="side num x y angle inv">

    <!-- Construindo a perna [Building a leg]-->
    <joint name="leg_center_joint_${side}${num}" type="fixed">
        <origin xyz="${x} ${y} 0" rpy="0 0 0"/>
        <parent link="body"/>
```

```
        <child link="leg_center_${side}${num}"/>
    </joint>

    <link name="leg_center_${side}${num}"/>

    <joint name="coxa_joint_${side}${num}" type="revolute">
        <origin xyz="0 0 0" rpy="0 0 ${angle}"/>
        <parent link="leg_center_${side}${num}"/>
        <child link="coxa_${side}${num}"/>
        <axis xyz="0 0 -1"/>
        <limit lower="${joint_lower_limit}" upper="${joint_upper_limit}"
effort="${joint_effort}" velocity="${joint_velocity}"/>
    </joint>

        <link name="coxa_${side}${num}">

            <inertial>
                <mass value="0.5"/>
                <inertia ixx="0.001"  ixy="0"  ixz="0" iyy="0" iyz="0" izz="0.001" />
            </inertial>

            <visual>
                <origin xyz="0 0 -0.025" rpy="0 0 0"/>
                <geometry>
                    <mesh
filename="package://hexapod_description/meshes/COXA_${side}.stl"/>
                </geometry>
                <material name="">
                    <color rgba="0.7 0.7 0 1"/>
                    </material>
            </visual>

            <collision>
                            <origin xyz="0 0 -0.025" rpy="0 0 0"/>
                <geometry>
                    <mesh
filename="package://hexapod_description/meshes/COXA_${side}_SIMPLIFIED.stl"/>
                </geometry>
            </collision>

        </link>

        <joint name="femur_joint_${side}${num}" type="revolute">
             <origin xyz="0 ${inv*0.030} 0" rpy="0 0 0" />
             <parent link="coxa_${side}${num}" />
             <child link="femur_${side}${num}" />
             <axis xyz="-1 0 0" />
             <limit lower="-0.52" upper="0.61" effort="${joint_effort}"
velocity="${joint_velocity}" />
          </joint>

        <link name="femur_${side}${num}">

            <inertial>
                <mass value="0.05"/>
                <inertia ixx="0"  ixy="0"  ixz="0" iyy="0" iyz="0" izz="0" />
            </inertial>

            <visual>
                    <origin xyz="${inv*0.022} 0 0" rpy="0 0 0" />
                    <geometry>
```

```
                        <mesh
filename="package://hexapod_description/meshes/FEMUR_${side}.stl" />
                    </geometry>
                    <material name="">
                        <color rgba="0 0.7 0.7 1" />
                    </material>
            </visual>

            <collision>
                        <origin xyz="${inv*0.022} 0 0" rpy="0 0 0" />
                <geometry>
                    <mesh
filename="package://hexapod_description/meshes/FEMUR_${side}_SIMPLIFIED.stl"/>
                </geometry>
            </collision>

        </link>

        <joint name="tibia_joint_${side}${num}" type="revolute">
            <origin xyz="0 ${inv*0.08} 0" rpy="${-pi/2} 0 ${(1-inv)*pi/2}" />
            <parent link="femur_${side}${num}" />
            <child link="tibia_${side}${num}" />
            <axis xyz="1 0 0" />
            <limit lower="-0.87" upper="1.22" effort="${joint_effort}"
velocity="${joint_velocity}" />
          </joint>

        <link name="tibia_${side}${num}">

            <inertial>
                <mass value="0.2"/>
                <inertia ixx="0"  ixy="0"  ixz="0" iyy="0" iyz="0" izz="0" />
            </inertial>

            <visual>
                    <origin xyz="${-inv*0.022} 0 0" rpy="${inv*pi/6+(1-inv)*pi/2}
${(1-inv)*pi/2} 0" />
                    <geometry>
                                <mesh
filename="package://hexapod_description/meshes/TIBIA_${side}.stl" />
                    </geometry>
                    <material name="">
                        <color rgba="0.7 0 0.7 1" />
                    </material>
            </visual>

            <collision>
                <origin rpy="0 0 0" xyz="0 0 0"/>
                <geometry>
                    <mesh
filename="package://hexapod_description/meshes/TIBIA_${side}_SIMPLIFIED.stl"/>
                </geometry>
            </collision>

        </link>

        <joint name="tibia_foot_joint_${side}${num}" type="fixed">
            <origin xyz="0 0.110 0" rpy="0 0 0" />
            <parent link="tibia_${side}${num}" />
            <child link="tibia_foot_${side}${num}" />
          </joint>
```

```
            <link name="tibia_foot_${side}${num}" />

        </xacro:macro>

        <xacro:leg side="R" num="1" x="0.0875"      y="0.050175"     angle="${pi*3/4}"
inv="-1"/>
            <xacro:leg side="R" num="2" x="0"               y="0.070"
angle="${pi}"         inv="-1"/>
            <xacro:leg side="R" num="3" x="-0.0875"     y="0.050175"  angle="${pi*5/4}"
inv="-1"/>
            <xacro:leg side="L" num="1" x="0.0875"          y="-0.050175"    angle="-
${pi*3/4}"    inv="1"/>
            <xacro:leg side="L" num="2" x="0"               y="-0.070"
angle="-${pi}"        inv="1"/>
            <xacro:leg side="L" num="3" x="-0.0875"     y="-0.050175"   angle="-
${pi*5/4}"    inv="1"/>

</robot>
```

APÊNDICE C – Conjunto de módulos com equações cinemáticas (Python)

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import numpy as np
#import tf.transformations as tf
from math import *
import cmath
#from geometry_msgs.msg import Pose, Quaternion

coxa = 30
femur = 80
tibia = 110

user_t1 = 0
user_t2 = 0
user_t3 = 0

def fwd_kin(theta1,theta2,theta3):

    X = tibia*sin(theta1)*cos(theta2)*cos(theta3-pi/2) -
tibia*sin(theta1)*sin(theta2)*sin(theta3-pi/2) + femur*sin(theta1)*cos(theta2) +
coxa*sin(theta1)

    Y = tibia*cos(theta1)*cos(theta2)*cos(theta3-pi/2) -
tibia*cos(theta1)*sin(theta2)*sin(theta3-pi/2) + femur*cos(theta1)*cos(theta2) +
coxa*cos(theta1)

    Z = tibia*sin(theta2)*cos(theta3-pi/2) + tibia*cos(theta2)*sin(theta3-pi/2) +
femur*sin(theta2)

    return X,Y,Z


def inv_kin(x,y,z):

    H = sqrt(x**2+y**2)-coxa
    L = sqrt(H**2+z**2)
    aux_theta2 = atan2(-z,H)

    theta1 = atan2(x,y)
    theta2 = acos((tibia**2-L**2-femur**2)/(-2*L*femur))-aux_theta2
    theta3 = pi/2 - acos((L**2-femur**2-tibia**2)/(-2*femur*tibia))

    return theta1,theta2, theta3
```

APÊNDICE D – Código do controle do robô hexapóde (Python)

```
#!/usr/bin/env python

import rospy
from sensor_msgs.msg import JointState
from std_msgs.msg import Header
from math import *
import cmath
import numpy as np
import kinematics_scripts
from kinematics_scripts import fwd_kin,inv_kin



def talker():
    pub = rospy.Publisher('joint_states', JointState, queue_size=50)
    rospy.init_node('joint_state_publisher')
    rate = rospy.Rate(50) # 50hz
    hello_str = JointState()
    hello_str.header = Header()
    hello_str.header.stamp = rospy.Time.now()
    hello_str.name = ['coxa_joint_R1', 'femur_joint_R1','tibia_joint_R1',
                      'coxa_joint_R2', 'femur_joint_R2','tibia_joint_R2',
                      'coxa_joint_R3', 'femur_joint_R3','tibia_joint_R3',
                      'coxa_joint_L1', 'femur_joint_L1','tibia_joint_L1',
                      'coxa_joint_L2', 'femur_joint_L2','tibia_joint_L2',
                      'coxa_joint_L3', 'femur_joint_L3','tibia_joint_L3']


    hello_str.position = [0,0,0,
                          0,0,0,
                          0,0,0,
                          0,0,0,
                          0,0,0,
                          0,0,0]
    hello_str.velocity = []
    hello_str.effort = []

    stepw = 0
    steph = 0
    height = 0.04*1000
    width = 0.05*1000
    y_dist = 0.06*1000
    res = 100
    ciclo = 1
    i=1

    a = 4*height/(width**2)
    b = 4*height/width


    R1ini = fwd_kin(0,0,0)
    R2ini = fwd_kin(0,0,0)
    R3ini = fwd_kin(0,0,0)
    L1ini = fwd_kin(0,0,0)
    L2ini = fwd_kin(0,0,0)
    L3ini = fwd_kin(0,0,0)

    R1 = (0,0,0)
```

```
    R2 = (0,0,0)
    R3 = (0,0,0)
    L1 = (0,0,0)
    L2 = (0,0,0)
    L3 = (0,0,0)

    while not rospy.is_shutdown():
      hello_str.header.stamp = rospy.Time.now()

      #Obtem a posicao da perna no momento atual
      posR1 = fwd_kin(hello_str.position[0],hello_str.position[1],-
hello_str.position[2])
      posR2 = fwd_kin(hello_str.position[3],hello_str.position[4],-
hello_str.position[5])
      posR3 = fwd_kin(hello_str.position[6],hello_str.position[7],-
hello_str.position[8])
      posL1 = fwd_kin(hello_str.position[9],hello_str.position[10],-
hello_str.position[11])
      posL2 = fwd_kin(hello_str.position[12],hello_str.position[13],-
hello_str.position[14])
      posL3 = fwd_kin(hello_str.position[15],hello_str.position[16],-
hello_str.position[17])

      print(posR1[1])

      if ciclo==1:
        R1x = R1ini[0]+(stepw/width)*width
        R1z = R1ini[2]-a*((stepw/width)*width)**2+b*(stepw/width)*width
        R1 = inv_kin(R1x, y_dist, R1z)
        R3x = R3ini[0]+(stepw/width)*width
        R3z = R3ini[2]-a*((stepw/width)*width)**2+b*(stepw/width)*width
        R3 = inv_kin(R3x, y_dist, R3z)
        L2x = L2ini[0]+(stepw/width)*width
        L2z = L2ini[2]-a*((stepw/width)*width)**2+b*(stepw/width)*width
        L2 = inv_kin(L2x, y_dist, L2z)

      else:
        L1x = L1ini[0]+(stepw/width)*width
        L1z = L1ini[2]-a*((stepw/width)*width)**2+b*(stepw/width)*width
        L1 = inv_kin(L1x, y_dist, L1z)
        L3x = L3ini[0]+(stepw/width)*width
        L3z = L3ini[2]-a*((stepw/width)*width)**2+b*(stepw/width)*width
        L3 = inv_kin(L3x, y_dist, L3z)
        R2x = R2ini[0]+(stepw/width)*width
        R2z = R2ini[2]-a*((stepw/width)*width)**2+b*(stepw/width)*width
        R2 = inv_kin(R2x, y_dist, R2z)

      if stepw/width == 1 and ciclo==1:
        stepw = 0
        ciclo = 2

      #  ciclo=2
      elif stepw/width == 1 and ciclo==2:
        stepw = 0
        ciclo = 1

      hello_str.position = [R1[0],R1[1],R1[2],
                            R2[0],R2[1],R2[2],
                            R3[0],R3[1],R3[2],
                            -L1[0],-L1[1],L1[2],
                            -L2[0],-L2[1],L2[2],
                            -L3[0],-L3[1],L3[2]]
```

```
        hello_str.velocity = []
        hello_str.effort = []

        stepw = stepw + width/res
        steph = steph + height/res

        pub.publish(hello_str)

        rate.sleep()


if __name__ == '__main__':
    try:
        talker()
    except rospy.ROSInterruptException:
        pass
```

APÊNDICE E – Arquivos de inicialização, com interface gráfica (Launch file)

```
<launch>

<arg name="model" default="xacro --inorder $(find
hexapod_description)/models/hexapod_model.xacro"/>
<arg name="gui" default="True"/>
<param name="robot_description" command="$(arg model)"/>
<param name="use_gui" value="$(arg gui)"/>
<node pkg="joint_state_publisher" type="joint_state_publisher"
name="joint_state_publisher"/>
<node pkg="robot_state_publisher" type="robot_state_publisher"
name="robot_state_publisher"/>
<node name="rviz" pkg="rviz" type="rviz"    args="-d $(find
hexapod_description)/launch/display_hexapod.rviz" />

</launch>
```

APÊNDICE F – Arquivos de inicialização, com cinemática (Launch file)

```
<launch>

<arg name="model" default="xacro --inorder $(find
hexapod_description)/models/hexapod_model.xacro"/>
<arg name="gui" default="True"/>
<param name="robot_description" command="$(arg model)"/>
<node pkg="robot_state_publisher" type="robot_state_publisher"
name="robot_state_publisher"/>
<node pkg="kinematics" name="control" type="control.py" output="screen"/>
<node name="rviz" pkg="rviz" type="rviz"    args="-d $(find
hexapod_description)/launch/display_hexapod.rviz" />

</launch>
```

APÊNDICE G – ESTRUTURA DE ARQUIVOS

O ROS utiliza pacotes para organizar partes de um projeto, e os diversos arquivos neste pacote são responsáveis pelas ações que aquele pacote podem realizar, podem ser modelo do robô, código para execução de tarefas e modelagem de parâmetros. Neste projeto temos dois pacote criados:

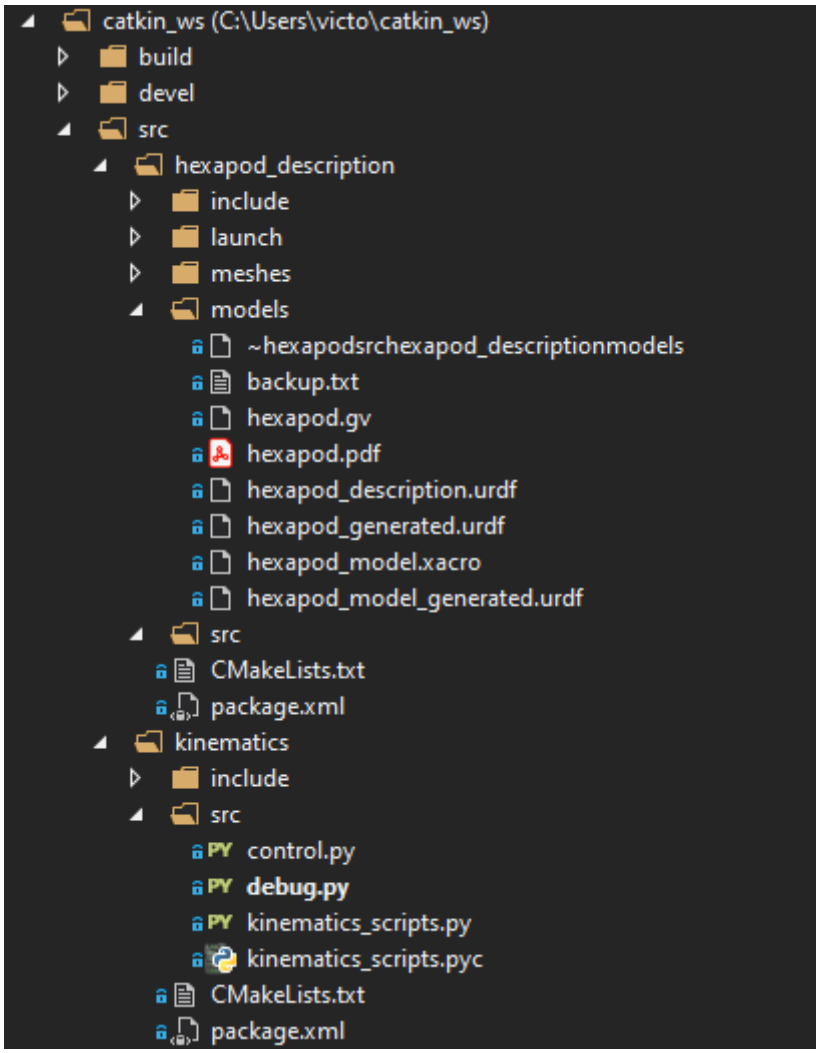
- **hexapod_description:** nele está o modelo cinemático do robô e os arquivos de inicialização da interface gráfica e simulação, está dividido em três partes. A pasta “models” contém os modelos hexapod_model.xacro e o hexapod_model_generated.urdf que são o modelo XACRO criado e URDF gerado para descrição do robô. A pasta “meshes” contém os arquivos STL que auxiliam a visualização do robô em 3D. A pasta “launch” contém os arquivos de inicialização do robô, o hexapod.launch gera a visualização do robô com a interface gráfica para controle das juntas e o hexapod_controled.launch utiliza os modelos de cinemática, para controlar o robô na visualização para executar uma passada.
- **Control:** Nesta pasta estão os códigos criados para controlar o robô, o arquivo “kinematics_scripts.py” contém o código em Python com a função de cinemática direta e cinemática inversa, o arquivo “control.py” contém a sequência de passada do robô hexapóde sendo publicada com o tópico “joint_state” no joint_state_publisher.

Outros arquivos e pastas importantes:

- **catkin_ws:** é o espaço de trabalho de um projeto no ROS, nele estão instalados os pacotes, dentro da pasta “src”.
- **CmakeList.txt:** é o arquivo utilizado para compilar um projeto Cmake (ferramenta utilizada para criar programa no ROS), nele estão contidos os caminhos dos arquivos, dependências e estrutura dos pacotes e espaço de trabalho.
- **package.xml:** arquivo que define propriedades como o nome do pacote, número da versão, autores, licensa e dependências.

A estrutura de arquivos descrita pode ser observada na Figura 24, os outros arquivos foram testes, arquivos de visualização ou referências.

Figura 32: Estrutura de arquivos do ROS.



Fonte: Elaborado pelo ator.