CONTROLE DE MANIPULADOR ROBÓTICO UTILIZANDO ROS

Leonardo Simião de Luna, PPGEM - Projeto, Materiais e Manufatura, leonardo.simiao.luna@usp.br
Glauco Augusto de Paula Caurin, SAE-EESC-USP, gcaurin@sc.usp.br
Mario Luiz Tronco, SEM-EESC-USP, mltronco@sc.usp.br

Resumo. *Este artigo descreve o processo de controle de um robô manipulador de 6 graus de liberdade através do framework ROS. Tal atividade é realizada visando integrar o manipulador a um sistema de controle distribuído que possa ser utilizado em aplicações de Indústria 4.0. A utilização do ROS vem ao encontro de permitir a modularidade e capacidade de integração do sistema. Partindo de um controle do robô através de seu espaço de juntas, utilizou-se uma biblioteca de robótica para se definir as cinemáticas direta e inversa do manipulador, possibilitando controlá-lo definindo trajetórias cartesianas. Descreve-se o experimento realizado para testar as movimentações e o comportamento do robô durante seu controle, atingindo os objetivos ao descrever trajetórias lineares durante a sua operação. A estratégia de controle descrita aqui será utilizada, futuramente, em uma estrutura de redes distribuída que utiliza um robô manipulador simulando uma máquina-ferramenta.*

Palavras chave: *ROS. Manipulação Robótica. Controle.*

1. INTRODUÇÃO

Enunciada por Kagermann et. al. (2013), a Indústria 4.0 representa um esforço de implementação de tecnologias que integrem sistemas cibernéticos aos sistemas físicos industriais. Entre as estratégias propostas da Indústria 4.0 está a utilização de sistemas de controle distribuídos, que permitem uma maior versatilidade do processo, facilitando gerenciamento, localização de falhas, identificação de gargalos, e criação de estratégias de manutenção preventiva e aumentando a eficiência do sistema (Hermann et. al., 2016). Visando a aplicação de elementos da Indústria 4.0, os autores deste trabalham buscam criar um estrutura de redes distribuída que simule, através da utilização de um robô manipulador KUKA KR16, uma máquina-ferramenta em operação. Tal máquina seria controlada remotamente através de sistemas em diferentes redes, utilizando de aplicações desejáveis para a Indústria 4.0, como o protocolo de comunicação MQTT (MQTT, 2021) e bases de dados armazenadas em nuvem.

Compondo essa estrutura, um dos sistemas necessários é a rede que controla o robô. Um dos requisitos desejados é que esta rede esteja suficientemente modularizada, o que facilita a sua integração no sistema distribuído. Neste sentido, a utilização do *Robot Operating System* (ROS) se adequa aos requisitos objetivados, visto que permite o encapsulamento do controle do manipulador robótico, e a independência do código utilizado neste ambiente.

Para cumprir o objetivo de simular uma máquina-ferramenta, torna-se necessário desenvolver um controle que possa permitir a movimentação da ferramenta na ponta do manipulador em trajetórias lineares. Para isso utiliza-se a biblioteca *Robotics Toolbox for Python* de Corke e Haviland (2021), que conta com ferramentas que permitem a definição da cinemática direta e inversa do robô. Isso permite que a criação de um controle que a partir de dada trajetória linear, consiga transmitir ao manipulador uma sequência de posições de junta que o possibilite descrever tal trajetória. Este artigo detalha a metodologia utilizada para estabelecer este controle, descreve o experimento utilizado para testá-la e demonstra as movimentações realizadas pelo robô durante o experimento.

2. MATERIAIS

2.1 Robot Operating Systeme e ROS-Industrial

O ROS teve sua origem em instituições de pesquisa e desenvolvimento em robótica, com o objetivo principal de prover desenvolvimento ágil de robôs e reutilização de algoritmos. Atua como um *framework*, encapsulando diferentes aplicações robóticas e encarregando-se da comunicação entre elas (ROS, 2021). Assim, demonstra-se uma solução viável para sistemas de controle distribuídos. Cada uma das aplicações a serem integradas pelo ROS podem ser projetadas individualmente, utilizando diferentes recursos e linguagens de programação.

Outra característica que o qualifica para a utilização em aplicações de Indústria 4.0 é sua estrutura de mensagens, que funciona em um sistema de organização produtor-consumidor (Rajkumar et. al., 1995). Tal organização permite uma redução da quantidade de mensagens a serem transmitidas, diminuindo a necessidade de utilização de *hardwares* com alta disponibilidade de recursos. Essa organização consiste de uma central de mensagens que controla a transmissão de informações entre os nós. Cada um dos nós que deseja fornecer mensagens ao sistema informa à central qual o

tópico ao qual estará publicando as mensagens. Cada nó que deseje ser informado por determinado tópico comunica a central e se inscreve naquele tópico. Diferentes centrais podem se conectar e transmitir informações entre si, facilitando a modularidade entre diferentes redes.

O ROS-Industrial formou-se através de um consórcio de pesquisadores e membros da indústria que têm por objetivo desenvolver aplicações para avançar o uso prático de ROS nos ambientes industriais (ROS-industrial, 2021). O repositório do consórcio tem pacotes para muitos robôs industriais, incluindo pacotes experimentais para manipuladores KUKA. Neste trabalho, utiliza-se os pacotes para a simulação do robô manipulador KR16 e os pacotes de comunicação com o *Robot Sensor Interface* (RSI).

Há três interfaces principais que permitem a comunicação entre o robô manipulador KUKA KR16 e o computador (Arbo et. al., 2020). Neste trabalho, foca-se na utilização do *Robot Sensor Interface* (RSI). Sendo uma interface que tem por objetivo a movimentação auxiliada por sensores, se adequa bem para etapas futuras do desenvolvimento da estrutura de rede, entre os quais a obtenção de dados de sensores para alimentar uma base de dados na nuvem para utilização posterior.

3. MÉTODOS

Os pacotes presentes no repositório do ROS-Industrial permitem o controle do robô KUKA KR16 através de seu espaço de juntas. Como se tem por objetivo utilizar o robô para simular uma máquina-ferramenta deve-se, através da utilização das cinemáticas direta e inversa do robô, criar um controle que permita a partir das movimentações lineares desejadas para a ferramenta na ponta do robô definir os ângulos da juntas. Esta seção descreve este processo, assim como descreve a configuração dos nós e dos tópicos utilizados no *framework* ROS para realizar a movimentação do manipulador.

3.1 Cinemática Direta

Para controlar o manipulador, é preciso compreender seu comportamento. Pode-se utilizar a cinemática direta do robô para determinar, através da atuação das juntas, a posição e a orientação final da ponta da ferramenta. Para isso, é utilizado o método de Denavit-Hartenberg (Siciliano et. al., 2009). No método, cada seção do robô é descrito por uma série de parâmetros relacionados à posição espacial de duas juntas vizinhas. Estes parâmetros descrevem a matriz de transformação da seção. A composição das matrizes de cada uma das seções do robô cria a matriz de transformação do mesmo. Esta matriz pode ser utilizada para encontrar a posição da ponta da ferramenta e sua orientação espacial em função dos ângulos das juntas. Para tal, utiliza-se os parâmetros como definidos em Corke (2017), os quais estão apresentados na Tab. (1). A representação visual na Fig. (1) auxilia a compreensão.

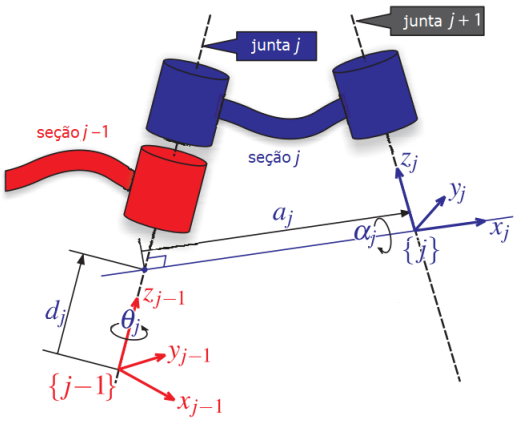


Figura 1: Demonstração dos parâmetros de Denavit-Hartenberg para cada seção entre duas juntas. Fonte: Corke (2017), adaptado)

O robô KR16 é um robô serial de 6 juntas, todas de revolução, com um punho esférico. A Tabela (2) descreve os parâmetros de cada seção. Precisa-se notar que os ângulos θ_3 e θ_6 são deslocados, para ter correspondência com o que o RSI define como origem com a origem definida em Tab. (1).

Tabela 1: Definição dos parâmetros de Denavit-Hartenberg de acordo com Corke (2017)

Ângulo da junta	θ_j	ângulo entre os eixos x_{j-1} e x_j com relação ao eixo z_{j-1}
Deslocamento da seção	d_j	distância entre a origem de $\{j-1\}$ até o eixo x_j ao longo do eixo z_{j-1}
Comprimento da seção	a_j	distância entre os eixos z_{j-1} e z_j ao longo do eixo x_{j-1}; caso os eixos se interseccionem, a distância no eixo paralelo a $\hat{z}_{j-1} \times \hat{z}_j$
Torção da seção	α_j	ângulo entre z_{j-1} e z_j com relação a x_j
Tipo de junta	σ_j	$\sigma = R$ para uma junta de revolução, $\sigma = P$ para uma junta prismática

Tabela 2: Tabela de Denavit-Hartenberg para o robô KUKA KR-16.

$\theta_\mathbf{j}$	$\mathbf{d_j}[m]$	$\mathbf{a_j}[m]$	$\alpha_\mathbf{j}$
q_1	0.675	0.26	$-90°$
q_2	0	0.68	$0°$
$q_3 - 90°$	0	-0.035	$90°$
q_4	-0.67	0	$-90°$
q_5	0	0	$90°$
$q_6 + 180°$	-0.158	0	$180°$

Assim, a matriz de transformação de cada seção do robô pode ser descrita,

$$^{(j-1)}T_j = \begin{bmatrix} R_j & t_j \\ 0_{3\times3} & 1 \end{bmatrix} = \begin{bmatrix} cos(\theta_j) & -sen(\theta_j)cos(\alpha_j) & sen(\theta_j)sen(\alpha_j) & a_j cos(\theta_j) \\ sen(\theta_j) & cos(\theta_j)cos(\alpha_j) & -cos(\theta_j)sen(\alpha_j) & \alpha_j sen(_j) \\ 0 & sen(\alpha_j) & cos(\alpha_j 1) & d_j \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{1}$$

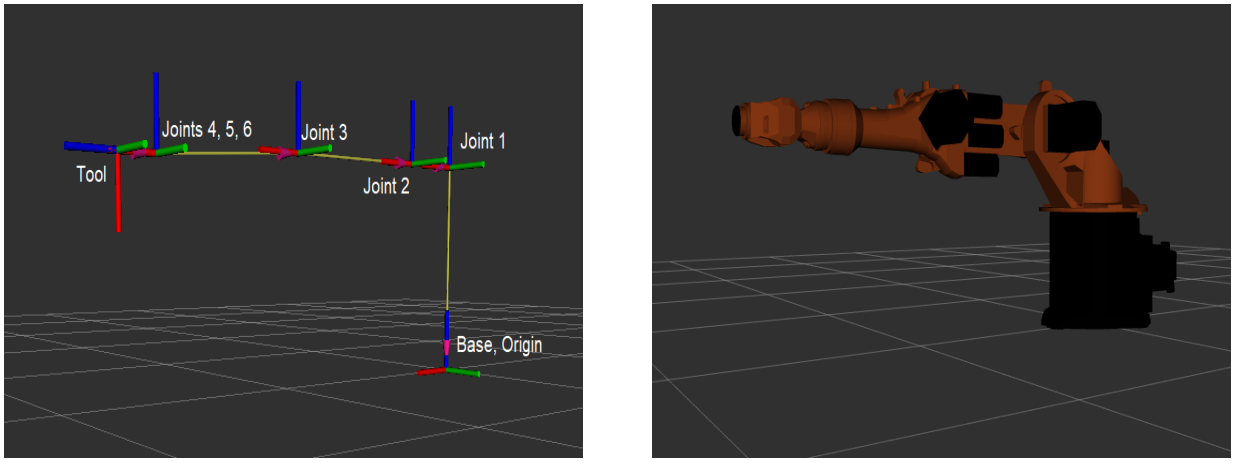
Onde R_j é a matriz de rotação para a seção do robô com relação à seção anterior, e t_j a posição cartesiana do fim da seção em relação à seção anterior. É possível compor a matriz de transformação do robô através da multiplicação de todas as matrizes das seções. Assim, pode-se chegar à matriz de transformação do robô na configuração de ângulo nulo ($q_j = 0$),

$$T_{robô} = \begin{bmatrix} R & t \\ 0_{3\times3} & 1 \end{bmatrix} = \prod_{j=1}^{n} {}^{(j-1)}T_j = \begin{bmatrix} 0 & 0 & 1 & 1.768 \\ 0 & 1 & 0 & 0 \\ -1 & 0 & 0 & 0.64 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2}$$

Onde R é a matriz de rotação para obter a orientação da ferramenta com relação à origem, e t é o vetor que indica sua posição cartesiana da ponta da ferramenta em relação à origem. Isto pode ser visto na Fig. (2). Os sistemas de orientação de todas as juntas estão representados na figura. As cores vermelho, verde e azul representam as direções positivas dos eixos x, y e z, respectivamente. Pode-se notar que a orientação positiva do eixo x nas coordenadas da ferramenta está na orientação negativa do eixo z, como demonstrado na matriz em Eq. (2).

3.2 Cinemática Inversa

Tendo acesso as matrizes que descrevem posição e orientação da ferramenta do robô, é possível determinar novas matrizes que representam novas posições e orientações após translações e rotações cartesianas da ferramenta. Para que, a partir dessas novas matrizes, encontre-se os ângulos que as geram, é necessário conhecer-se a cinemática inversa do robô. Usando a biblioteca *Robotics Toolbox for Python* de Corke e Haviland (2021), pode-se utilizar um método que deriva a cinemática inversa do robô numericamente, usando o método de Levenbert-Marquardt (Levenberg, 1944). Desta forma, podemos obter as posições de juntas q_i de uma matriz T_i que descreve posição e orientação da ponta da ferramenta do robô. Como se trata de um robô de seis graus de liberdade com um pulso esférico, há diferentes posições de junta que representam a mesma matriz T_i. Da forma como foi implementada a estratégia de controle, o movimentação ocorre em pequenos incrementos de posição, o que faz com que o robô permaneça sempre com a mesma configuração, que é definida em Corke (2017) como a configuração de ombro levantado, braço à direita do ombro e pulso torcido.



(a) Juntas, seções e sistemas de orientação do robô (b) Modelo do robô

Figura 2: Representações do robô KR16 na configuração de ângulo nulo

3.3 Nós e tópicos definidos no ROS

A Figura (3) demonstra os nós e tópicos ativos durante a simulação do experimento. Controla-se o robô através do nó */joint_client_py*. Ele se comunica com o nó */kuka_hardware_interface*, publicando um estado de juntas desejado e aguardando pela confirmação de que a ação foi realizada. O nó de interface então publica a junta desejada para o */robot_state_publisher*, que manda as novas posições de junta para o RSI, controlando a movimentação do robô. A interface com o *hardware* também envia os dados do robô para o nó */file_output_py* que é o nó que colecta todos os dados. Através desse nó, o ROS gerencia a transmissão dos dados para as redes externas.

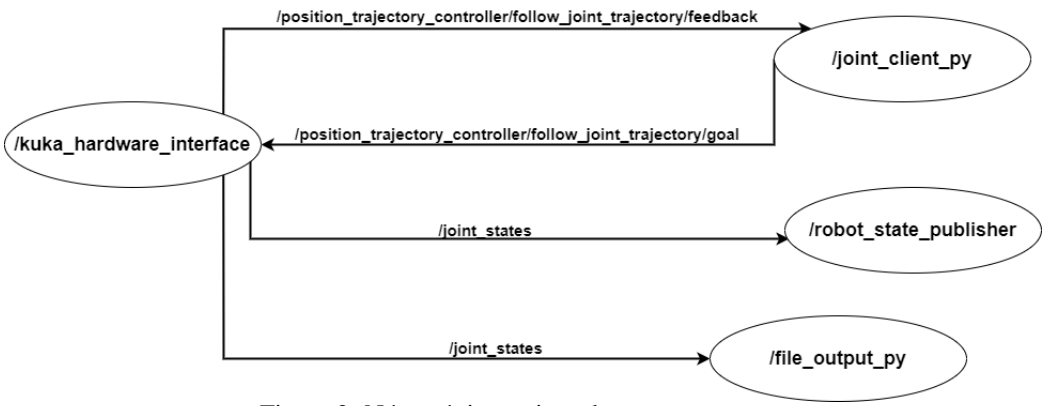


Figura 3: Nós e tópicos ativos durante a operação

3.4 Trajetórias

O objetivo do experimento é simular uma máquina-ferramenta em operação. Na simulação, limita-se isso à realização de uma trajetória simples: o robô inicia em uma posição segura, a uma altura livre; reduz a sua altura; entra no plano de trabalho lateralmente e descreve uma trajetória quadrangular; e então retorna para a posição inicial, na posição livre.

Como temos por objetivo realizar movimentações cartesianas em um robô composto exclusivamente por juntas de revolução, não podemos controlar apenas seus pontos inicial e final de sua trajetória. O controle realiza uma interpolação da trajetória, determinando pontos intermediários, de modo a aproximar o máximo a movimentação de uma trajetória linear.

4. RESULTADOS

A trajetória planar descrita pela ponta da ferramenta do robô no plano de trabalho desejado está apresentada na Fig. (4). Pode ser visto que o robô foi capaz de criar uma movimentação linear, como desejado para o experimento. Ao

comparar as sub-figuras da Fig. (5), pode-se reparar que a movimentação linear da ferramenta resulta em variações de ângulo bem diversas em cada uma das juntas do robô. A diferença entre os ângulos desejados definidos pelo controle e os ângulos reais obtidos no experimento podem ser vistos na Fig. (6), que demonstra os erros dos ângulos ao longo do experimento.

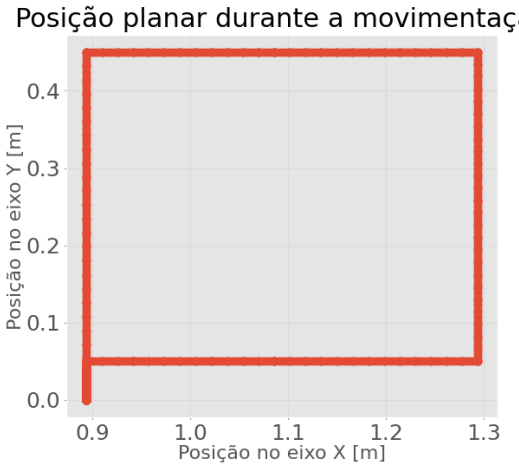
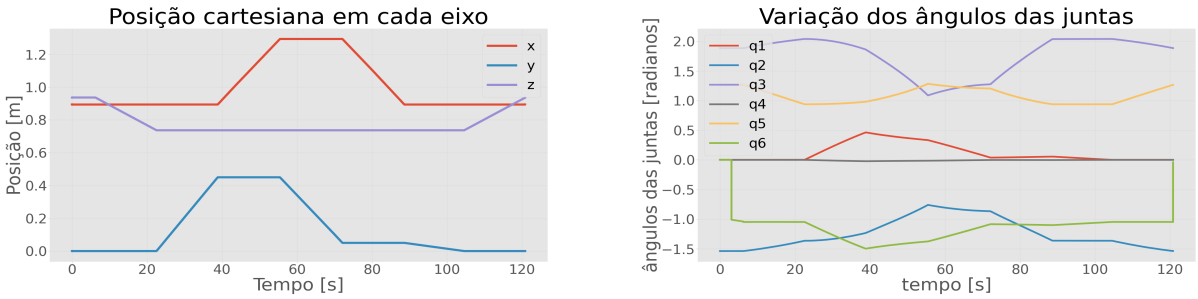


Figura 4: Posição planar da ferramenta durante a movimentação



(a) Trajetória nos eixos cartesianos

(b) Variação dos ângulos das juntas

Figura 5: Comportamento do robô durante os experimentos

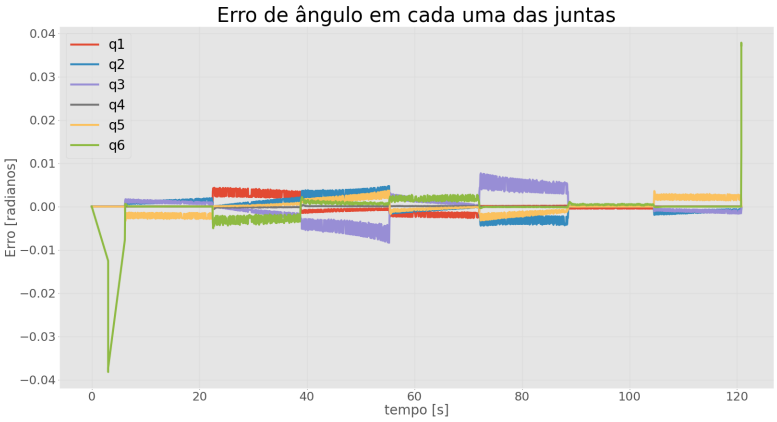


Figura 6: Diferenças entre ângulos desejados e atuais no experimento

5. CONCLUSÕES E DISCUSSÕES

Este trabalho teve por objetivo descrever o processo realizado para controlar um robô manipulador através do *framework* ROS, visando utilização futura em simulação de operação de máquina-ferramenta. Os resultados do experimento demonstram que esta é uma maneira viável de se controlar o robô manipulador. O *framework* ROS cumpre as funções desejadas, conseguindo transmitir as posições de junta determinadas pelo controle para o robô através do RSI. Também encapsula todas as informações da rede e permite sua transmissão para outras redes facilmente. Assim, este sistema cumpre os requisitos propostos, e é capaz de ser integrado à estrutura de redes.

Futuramente, planeja-se conectar esta rede com uma rede na nuvem, e realizar a comunicação entre o ROS na rede do robô e uma central de messagens MQTT na rede remota, o que é facilitado pelo sistema de comunicação produtor-consumidor a que ambos aderem. O experimento com o robô será aprimorado, utilizando como ferramentas na ponta do robô canetas que permitirão traçar a trajetória em folhas de papel, comprovando a adequação da movimentação também pelo resultado físico, e comparando-os com os dados coletados pelo ROS. Serão coletadas ainda mais informações dos sensores do robô através da interface com o RSI, como informações de velocidade e aceleração, bem como esforços nas juntas.

6. REFERÊNCIAS

ARBO, MH ; ERIKSEN, I ; SANFILIPPO, F ; GRAVDAHL, JT: Comparison of KVP and RSI for Controlling KUKA Robots Over ROS. In: *IFAC-PapersOnLine* 53 (2020), Nr. 2, S. 9841–9846

CORKE, Peter: *Robotics, vision and control: fundamental algorithms in MATLAB® second, completely revised*. Bd. 118. Springer, 2017

CORKE, Peter ; HAVILAND, Jesse: Not your grandmother's toolbox–the Robotics Toolbox reinvented for Python. In: *IEEE International Conference on Robotics and Automation*, 2021

HERMANN, Mario ; PENTEK, Tobias ; OTTO, Boris: Design principles for industrie 4.0 scenarios. In: *2016 49th Hawaii international conference on system sciences (HICSS)* IEEE (Veranst.), 2016, S. 3928–3937

KAGERMANN, Henning ; WAHLSTER, Wolfgang ; HELBIG, Johannes et. al.: Recommendations for implementing the strategic initiative Industrie 4.0: Final report of the Industrie 4.0 Working Group. In: *Forschungsunion: Berlin, Germany* (2013)

LEVENBERG, Kenneth: A method for the solution of certain non-linear problems in least squares. In: *Quarterly of applied mathematics* 2 (1944), Nr. 2, S. 164–168

MQTT: *MQTT : The Standard for IoT Messaging.* Disponível em: https://mqtt.org/, Acessado em 25 de fevereiro de 2021. 2021

RAJKUMAR, Ragunathan ; GAGLIARDI, Michael ; SHA, Lui: The real-time publisher/subscriber inter-process communication model for distributed real-time systems: design and implementation. In: *Proceedings Real-Time Technology and Applications Symposium* IEEE (Veranst.), 1995, S. 66–75

ROS: *ROS*. Disponível em: https://ros.org/, Acessado em 25 de fevereiro de 2021. 2021

ROS-INDUSTRIAL: *ROS-industrial*. Disponível em: https://rosindustrial.org/,Acessado em 20 de maio de 2021. 2021

SICILIANO, Bruno ; SCIAVICCO, Lorenzo ; VILLANI, Luigi ; ORIOLO, Giuseppe: Modelling, planning and control. In: *Advanced Textbooks in Control and Signal Processing. Springer,* (2009)

7. AGRADECIMENTOS

Os autores gostariam de agradecer à AWS, CNPQ processo 131354/2020-5, CAPES processo 1825953, CEPOF, e FINEP, que financiaram parcialmente este trabalho.

8. RESPONSABILIDADE PELAS INFORMAÇÕES

Os autores são os únicos responsáveis pelas informações incluídas neste trabalho.