UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Câmpus de Ilha Solteira

VINÍCIUS MACHADO OTOBONI

CINEMÁTICA INVERSA E PROGRAMAÇÃO DE UM MANIPULADOR ROBÓTICO COM 3 GRAUS DE LIBERDADE

Ilha Solteira, SP
Fevereiro de 2022

UNIVERSIDADE ESTADUAL PAULISTA
"JÚLIO DE MESQUITA FILHO"
Câmpus de Ilha Solteira

VINÍCIUS MACHADO OTOBONI

CINEMÁTICA INVERSA E PROGRAMAÇÃO DE UM MANIPULADOR ROBÓTICO COM 3 GRAUS DE LIBERDADE

Trabalho de Graduação apresentado à Faculdade de Engenharia - UNESP – Campus de Ilha Solteira, para cumprimento de requisito para obtenção do Grau de Engenheiro Mecânico

Prof. Dr. Márcio Antonio Bazani
Orientador

Ilha Solteira, SP
Fevereiro de 2022

FICHA CATALOGRÁFICA
Desenvolvido pelo Serviço Técnico de Biblioteca e Documentação

O88c

Otoboni, Vinícius Machado.
Cinemática inversa e programação de um manipulador robótico com 3 graus de liberdade / Vinícius Machado Otoboni. -- Ilha Solteira: [s.n.], 2022
38 f. : il.

Trabalho de conclusão de curso (Graduação em Engenharia Mecânica) - Universidade Estadual Paulista. Faculdade de Engenharia de Ilha Solteira, 2022

Orientador: Márcio Antonio Bazani
Inclui bibliografia

1. Cinemática inversa. 2. Internet das coisas. 3. Manipulador robótico.

Raiane da Silva Santos
Supervisora Técnica de Seção
Seção Técnica de Referência, Atendimento ao usuário e Documentação
Diretoria Técnica de Biblioteca e Documentação
CRB/8 - 9999

MODELO DE ATA

UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"

FACULDADE DE ENGENHARIA - CAMPUS DE ILHA SOLTEIRA

CURSO DE ENGENHARIA MECÂNICA

ATA DA DEFESA – TRABALHO DE GRADUAÇÃO

TÍTULO: Cinemática inversa e programação de um manipulador robótico com três graus de liberdade

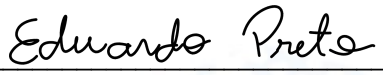
ALUNO: Vinícius Machado Otoboni RA:141052181

ORIENTADOR: Prof. Doutor Márcio Antonio Bazani

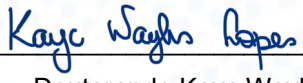
Aprovado (X) - Reprovado ( ) pela Comissão Examinadora

Comissão Examinadora:

Prof. Dr. Márcio Antonio Bazani
Presidente (Orientador)



Mestrando Eduardo Preto



Doutorando Kayc Wayhs Lopes

Ilha Solteira(SP) 11 de Fevereiro de 2022.

AGRADECIMENTOS

Agradeço à minha família e amigos, pelo apoio e por acreditarem em mim ao longo desses muitos anos de faculdade.

Agradeço aos professores e funcionários da FEIS, pelo excelente ensino e serviço prestado aos alunos.

Agradeço ao prof. Dr. Márcio Antonio Bazani, por ter me orientado neste trabalho e durante o estágio, além das aulas ministradas.

Agradeço à Equipe de Robótica T.E.R.A., pelo crescimento proporcionado e pelas amizades feitas ao longo dos meus anos como parte da equipe.

RESUMO

O objetivo deste trabalho é obter a cinemática inversa de um manipulador robótico do tipo RRR com três graus de liberdade e então, propor um algoritmo em linguagem C para que o manipulador alcance dada posição e orientação. Este algoritmo será executado em um microcontrolador ESP32, que transmitirá os ângulos de juntas obtidos por modulação LoRa para outro ESP32, que atua como receptor. Para a cinemática inversa, são obtidos os parâmetros de Denavit-Hartenberg e então os ângulos de juntas, através de uma solução algébrica. A programação é feita utilizando a Arduino IDE e dois módulos WiFi LoRa 32 (V2).

Palavras-chave: Cinemática inversa, Denavit-Hartenberg, LoRa

ABSTRACT

The goal of this paper is to solve the inverse kinematics of a RRR robotic manipulator with three degrees of freedom and then propose an algorithm in C language so that the manipulator can reach given position and orientation. This algorithm will run on an ESP32 microcontroller that will transmit the obtained joint angles by LoRa modulation to another ESP32, acting as a receiver. For the inverse kinematics, Denavit-Hartenberg parameters and joint angles are obtained, by an algebric solution. Programming is done using Arduino IDE and two WiFi LoRa 32 (V2) modules.

Key-words: Inverse kinematics, Denavit-Hartenberg, LoRa

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 – INTRODUÇÃO

Nos últimos anos, devido principalmente ao desenvolvimento da Internet das Coisas (IoT), a produção industrial tem passado por diversas mudanças e avanços tecnológicos. Na Indústria 4.0, conceitos como Redes de Comunicação Industrial, Comunicação Máquina a Máquina, Automação Totalmente Integrada, entre outros, se tornam mais presentes no meio fabril (STEVAN, LEME e SANTOS, 2018). Nesse contexto, a mecatrônica ganha um importante papel: fazer a integração entre a mecânica e as tecnologias que surgem com essa nova Indústria.

Microcontroladores são circuitos integrados presentes em sistemas embarcados. Normalmente, um microcontrolador inclui um processador, memória e periféricos de entrada e saída (do inglês, *input* e *output*, respectivamente) (LUTKEVICH, 2019). Estes circuitos integrados estão presentes em uma série de aplicações e produtos, como computadores, celulares, veículos, robôs, entre outros e vem se tornando ainda mais presentes devido ao avanço da Indústria 4.0.

Dispositivos com microcontroladores utilizam seus periféricos para trocar informações e dados com o ambiente à sua volta e tomar decisões baseadas nestes dados e no código que foi escrito para o mesmo. Esta troca de dados ocorre, por exemplo, na leitura de um sensor de umidade ou no acionamento de um servo motor de um manipulador industrial.

Entre os microcontroladores utilizados para a IoT há o ESP32. Lançado pela empresa Espressif, este microcontrolador possui um chip com rede *WiFi*, podendo ter também *Bluetooth* (OLIVEIRA, 2021). Devido à sua popularidade, é possível encontrar diversos módulos de desenvolvimento que utilizam este chip.

Há diversas tecnologias e protocolos utilizados na Indústria 4.0 e na IoT e, recentemente, LoRa vem ganhando destaque. Trata-se de uma tecnologia de rádio frequência criada pela empresa Semtech para redes LPWANs (l*ow-power wide area networks*), que são redes de área ampla de baixa potência. Seu nome é uma referência a "*long range*", ou seja, longa distância. A tecnologia LoRa permite, nas condições ideais, alcances de até 5 km em

áreas urbanas e até 15 km em áreas rurais, com um baixo consumo de energia (SEMTECH, 2019).

Enquanto LoRa é considerada a camada física que permite a comunicação à longa distância, LoRaWAN define o protocolo de comunicação e a arquitetura de sistema para a rede. Em uma rede LoraWAN, os dispositivos LoRa transmitem seus dados para diversos *gateways*. Esses *gateways* são responsáveis por transmitir as informações recebidas para algum servidor por meio de outro protocolo de comunicação (por exemplo, via satélite ou ethernet). Dessa forma, a complexidade e tratativa das informações recebidas é responsabilidade do servidor. (LORA ALLIANCE, 2015).

Este trabalho tem como objetivo principal obter a cinemática inversa de um manipulador robótico com 3 graus de liberdade e com isso desenvolver um programa em C para que o manipulador alcance determinada coordenada. Também será estudada a viabilidade de transmitir estas coordenadas utilizando a tecnologia LoRa, fazendo com que o manipulador possa ser controlado à distância.

2- FUNDAMENTAÇÃO TEÓRICA

2.1- ELEMENTOS DO MANIPULADOR

Manipuladores robóticos são constituídos por uma série de corpos, denominados elos, interligados por juntas. Normalmente possuem em seu último elo um efetuador, que os permite interagir com o meio à sua volta. O efetuador depende da função do robô, podendo ser uma garra ou um maçarico, por exemplo (CRAIG, 2012).

Para que um manipulador possa se movimentar, são necessários atuadores, como motores de passo ou servo motores. Os atuadores atuam nas juntas, proporcionando movimento relativo entre os elos. As juntas, por sua vez, determinam a quantidade de graus de liberdade (G.D.L) que um manipulador possui. A figura abaixo traz os tipos de juntas mais comuns e seus G.D.L.

Figura 1: Tipos de juntas e graus de liberdade (dof)

Joint type	dof f	Constraints c between two planar rigid bodies	Constraints c between two spatial rigid bodies
Revolute (R)	1	2	5
Prismatic (P)	1	2	5
Helical (H)	1	N/A	5
Cylindrical (C)	2	N/A	4
Universal (U)	2	N/A	4
Spherical (S)	3	N/A	3

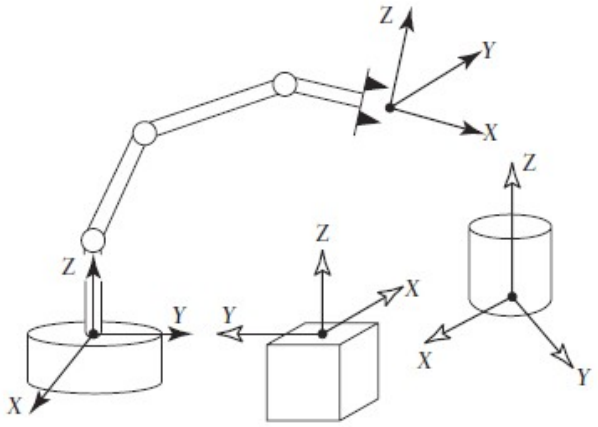
Fonte: LYNCH, e PARK (2017).

A quantidade de graus de liberdade de um robô indica a quantidade de variáveis independentes necessárias para definir por completo a posição do efetuador no espaço (LYNCH e PARK, 2017). Um robô do tipo RRR, por exemplo, possui três juntas rotacionais e consequentemente 3 G.D.L., necessitando de três variáveis independentes para descrever seu movimento.

2.2- POSIÇÃO E ORIENTAÇÃO

A localização do manipulador, seus elos, efetuador e objetos ao seu redor é extremamente importante para que a interação entre esses elementos ocorra conforme desejado. Para tanto, deve-se conhecer a posição e orientação de um corpo no espaço. Assim, um sistema de coordenadas, ou sistema de referência, é atrelado ao objeto de interesse. Então, é possível descrever este sistema de referência em relação à outros sistemas de coordenadas. A Figura 2 demonstra como alguns sistemas de referência podem ser atribuídos.

Figura 2: Manipulador robótico e sistemas de referência em sua base, efetuador e objetos ao redor



Fonte: CRAIG (2012).

2.3- CINEMÁTICA DIRETA E INVERSA

A cinemática é a ciência que estuda os movimentos sem considerar as forças que o causam (CRAIG, 2012). Em geral, a posição do manipulador é reproduzida por uma descrição do sistema de referência da ferramenta, que é ligado ao efetuador, com relação ao sistema de referência da base, que é ligado à base fixa do manipulador.

Dessa forma, a cinemática direta trata de computar a posição e orientação do sistema de referência da ferramenta com relação ao sistema da base. Já a cinemática inversa consiste em, dada a posição e orientação do efetuador, encontrar todas as possíveis combinações de ângulos de juntas que poderiam ser utilizados para se obter a posição e orientação desejadas.

2.4- MICROCONTROLADORES

Microcontroladores são pequenos circuitos integrados que possuem processador, memória (RAM e Flash) e periféricos em um único chip. Em sistemas embarcados, são programados para fazer uma função específica (LUTKEVICH, 2019). No contexto deste trabalho, microcontrolador pode ser entendido como o "cérebro" do manipulador robótico, sendo responsável pela resolução das equações da cinemática inversa e pela comunicação. A Figura 3 traz um exemplo de microcontrolador.

Figura 3: Microcontrolador ATMEGA328, fabricado pela Microchip.

Fonte: MICROCHIP (2022)

O processador é responsável por executar as instruções, aritmética e lógica, operações de periféricos, leitura de dados e transferência de dados responsáveis pelo funcionamento do sistema embarcado, de acordo com o código que está programado em sua memória. Além de conter o código que será executado, a memória pode guardar dados que foram coletados durante a execução do microcontrolador.

Os periféricos de um microcontrolador são, basicamente, entrada (*input*) ou saída (*output*). Um pino programado como *input* pode ser responsável pela medição de temperatura ou para indicar se um botão foi pressionado ou não, por exemplo. Já um pino programado como *output* tem como responsabilidade acender um LED, ou controlar um servo motor, entre outras funções.

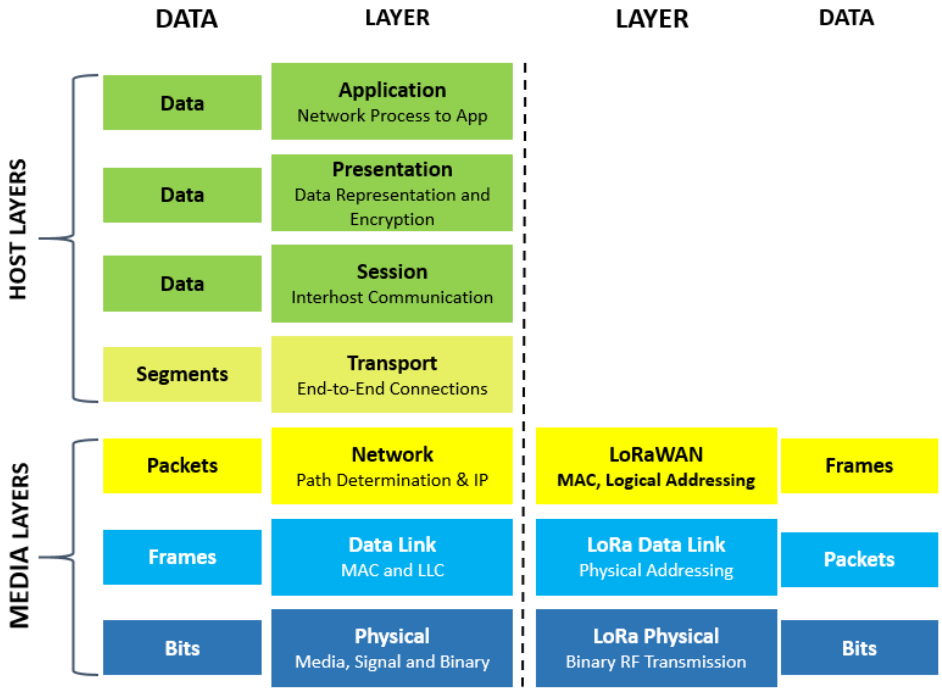
Comumente encontra-se no mercado, módulos (ou kits) de desenvolvimento. Esses módulos permitem acesso fácil aos periféricos do microcontrolador, permitindo uma avaliação mais rápida de suas funções.

2.5- LORA E LORAWAN

LoRa significa *Long Range* e é uma tecnologia de modulação de radiofrequência para redes LPWAN, criada pela empresa Semtech (SEMTECH, 2019). O grande diferencial de LoRa para outras soluções que utilizam Internet das Coisas é seu baixo consumo e longo alcance. Ainda, o protocolo LoRaWAN facilita a utilização desta tecnologia e a comunicação entre diversos dispositivos.

A modulação utilizada no LoRa é derivada da tecnologia CSS (do inglês, *Chirp Spread Spectrum*) e utiliza fatores ortogonais de espalhamento (LORA ALLIANCE, 2015). Na prática, permite um alcance maior, com um menor consumo de energia porém com uma taxa de transmissão de dados menor. No Modelo OSI de sete camadas, LoRa é a camada física por onde os dados são transmitidos, e LoRaWAN define o protocolo de comunicação e a arquitetura de rede. A Figura 4 apresenta o Modelo OSI e as camadas ocupadas pela LoRa e LoRaWan.

Figura 4: Modelo OSI e camadas ocupadas pela LoRa e LoRaWAN.



Fonte: SEMTECH (2019).

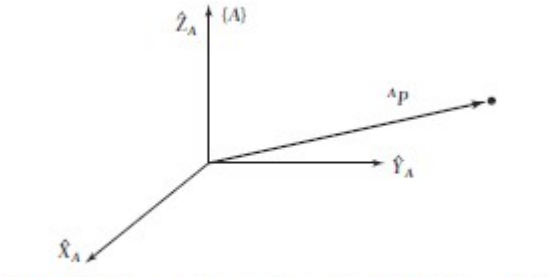
3- CINEMÁTICA

Conforme explicado na seção anterior, a cinemática estuda o movimento de corpos sem se preocupar com as forças que o causam. No escopo deste trabalho, obter a cinemática inversa é o ponto de maior importância. Para tanto, primeiro é necessário o entendimento de conceitos fundamentais, que serão apresentados neste capítulo.

3.1- POSIÇÕES, ORIENTAÇÕES E SISTEMAS DE REFERÊNCIA

Uma vez estabelecido um sistema de coordenadas, um ponto qualquer pode ser localizado por um vetor de posição de dimensões 3x1, sempre descrito em relação a algum sistema de coordenadas. Desta forma a notação ${}^{A}P$ indica que o vetor de posição P está descrito em relação ao sistema *{A}*, como demonstra a Figura 5.

Figura 5: Vetor P descrito em relação ao sistema de coordenadas {A}.



Fonte: CRAIG (2012)

Os elementos individuais do vetor ${}^{A}P$ são dados com os subscritos x, y, z, conforme a equação (1):

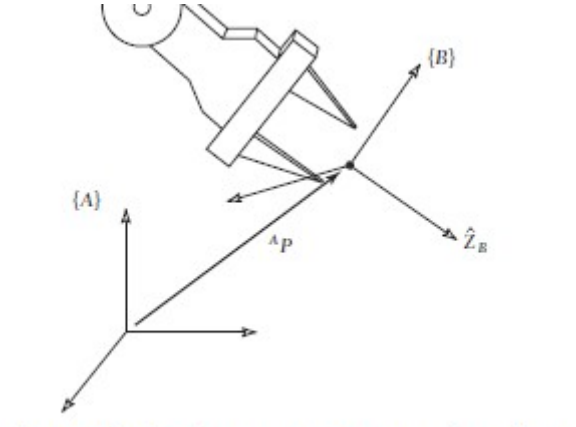
$$ {}^{A}P = \begin{bmatrix} p_x \\ p_y \\ p_z \end{bmatrix} \tag{1} $$

No caso de manipuladores, além da posição também é necessário definir sua orientação. Para obter a orientação de um corpo, um sistema de coordenadas é fixado a este e então este sistema é descrito em relação ao sistema de referência.

> "Assim, posições de pontos são descritas com vetores e orientações de corpos são descritas com um sistema de coordenadas fixado ao corpo." (CRAIG, 2012)

Utilizando como exemplo a figura abaixo, uma forma de descrever o sistema de coordenadas {B} é escrever os vetores unitários de seus eixos principais em relação ao sistema de coordenadas {A}, através da matriz rotacional R, composta pelos vetores unitários de um sistema de coordenadas em relação ao outro.. No caso da Figura 6, a matriz rotacional é dada pela equação (2).

Figura 6: Sistema de coordenadas {B} em relação a {A}.



Fonte: CRAIG (2012).

$$ {}^{A}_{B}R = [{}^{A}\hat{X}_B \; {}^{A}\hat{Y}_B \; {}^{A}\hat{Z}_B] = \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix} \tag{2} $$

Para descrever totalmente onde se encontra o manipulador da imagem acima é necessário uma posição e orientação. Por conveniência, o ponto cuja posição será descrita é escolhido como a origem do sistema de referência fixado ao corpo.

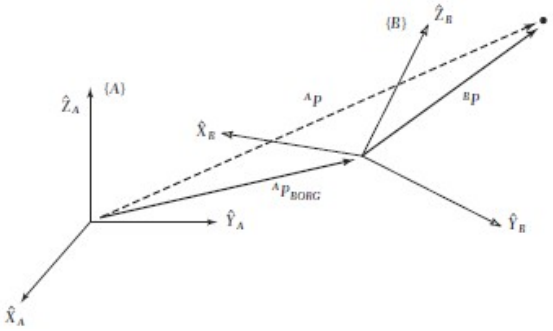
Dessa forma, define-se sistema de referência ou referencial como sendo um conjunto de quatro vetores que fornecem a orientação e posição de um corpo. Neste caso, temos que o sistema de referência {B} é descrito por:

$$\{B\}=\{{}^{A}_{B}R, {}^{A}P_{BORG}\} \tag{3}$$

sendo ${}^{A}P_{BORG}$ é o vetor que localiza a origem do sistema {B}.

Quando expressamos uma quantidade de um sistema de coordenas em relação a outro, estamos fazendo o mapeamento. Para mapeamentos genéricos, podemos utilizar como exemplo o caso da Figura 7:

Figura 7: Exemplo de mapeamento genérico.



Fonte: CRAIG (2012)

Dado ${}^{B}P$, queremos obter ${}^{A}P$. Para tanto, podemos multiplicar ${}^{B}P$ pela matriz de rotação de {B} em relação a {A} e entra somar com a translação entre as origens, ${}^{A}P_{BORG}$, conforme apresentado a seguir.

$${}^{A}P={}^{A}_{B}R\,{}^{B}P+{}^{A}P_{BORG} \tag{4}$$

A Equação (4) trata o mapeamento de transformação geral de um vetor, ou seja, de um sistema de referência para outro.

Para realizar a matemática proposta na equação (4), é necessário definir uma matriz 4x4. Dessa forma, adiciona-se 1 ao último elemento de cada vetor (${}^{A}P$ e ${}^{B}P$) e adiciona-se a linha 0 0 0 1 à matriz 3x3(${}^{A}_{B}R$). A equação pode, então, ser escrita na forma:

$$ {}^{A}P = {}^{A}_{B}T\,{}^{B}P \tag{5}$$

A matriz 4x4 ${}^{A}_{B}T$ é chamada matriz de transformação homogênea e permite, no contexto deste trabalho, dispor a rotação e a translação da transformação geral em uma única matriz.

3.2- DESCRIÇÕES DE UM ELO E SUAS CONEXÕES

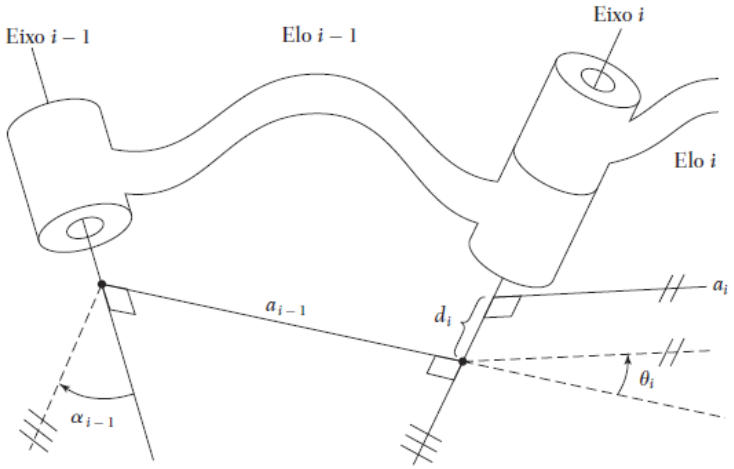
Para se obter as equações cinemáticas, devemos considerar um elo como sendo "u*m corpo rígido que define a relação entre eixos de duas juntas vizinhas, de um manipulador*" (CRAIG, 2012). O eixo das juntas são linhas retas no espaço em torno da qual um eixo rotaciona em relação ao eixo anterior. Um elo pode ser especificado com dois parâmetros que definem a localização relativa de dois eixos no espaço, e sua relação com elos vizinhos por mais dois parâmetros.

O primeiro parâmetro a ser considerado é o comprimento de elo (*a*). Esta distância é medida por uma linha mutualmente perpendicular aos dois eixos. O segundo parâmetro é a torção do elo (α). Para definir esse parâmetro, primeiro devemos imaginara um plano cuja normal é a linha perpendicular entre os eixos. Devemos então medir o ângulo entre os eixos utilizando a regra da mão direita, do eixo *i -1* ao eixo *i*.

O deslocamento de elo, também conhecido como distância entre elos ou *offset* (*d*), é medido ao longo do eixo da junta *i*, como sendo a distância entre o ponto onde o comprimento de elo a_{i-1} e o comprimento de elo a_i cruzam o eixo *i*. Essa distância é variável apenas em eixos prismáticos. O último parâmetro, que é variável em juntas rotacionais, é o ângulo de junta (θ). Este é o ângulo entre o comprimento de elo a_{i-1} e o comprimento de elo a_i .

Esses quatro parâmetros acima descritos e apresentados na Figura 8, quando aplicados a todos os elos, podem descrever cinematicamente por completo um robô. No caso específico de juntas rotacionais, o ângulo de junta θ é chamado de variável de junta, e os outros três parâmetros se mantém fixos. A definição de mecanismos utilizando essas quatro quantidades é chamada de notação de Denavit-Hartenberg.

Figura 8: Parâmetros de elo.



Fonte: CRAIG (2012).

3.3- CONVENÇÃO PARA FIXAR SISTEMAS DE REFERÊNCIA A ELOS

A convenção utilizada é que, dado um sistema de referência {i}, este tem sua origem onde a perpendicular a_i cruza o eixo de junta *i*. O eixo *Z* deste sistema é coincidente com eixo da junta *i* e *X* aponta ao longe de ai na direção da junta *i-1*. Caso a_i seja 0, *X* é normal ao plano de Z_i e Z_{i+1}.

α_i é medido utilizando a regra da mão direta no eixo *X*. Por fim, Y_i é formado pela regra da mão direita para completar o sistema de referência.

O primeiro elo de um robô é chamado de elo 0, e seu sistema de referência é {0}. Este sistema de referência é fixo na base do robô e é imóvel. Neste caso, podemos escolher a direção *Z* ao longo do eixo 1 e localizar {0} de forma que este coincida com {1} quando a

variável de junta 1 é 0. Dessa forma, a_0 e α_0 serão 0, e d_1 será 0 se for junta rotacional, e θ_1 será 0 se for junta prismática.

No caso da última junta *N*, se for rotacional, *X* é escolhido para se alinhar com X_{N-1} quando θ_N for 0 e então {N} é escolhido para que d_N seja 0. Caso *N* seja prismática, X_N é escolhido de forma que θ_N seja 0 e a origem de {N} é escolhido no cruzamento de X_{N-1} com o eixo da junta *N* quando $d_N=0$. A Figura 9 demonstra como fixar os sistemas de referência.

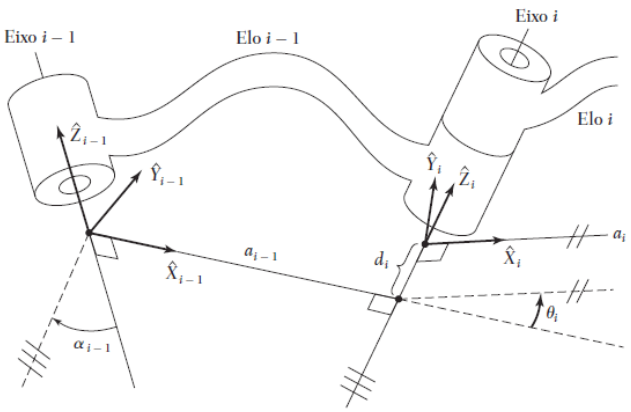
Uma vez tendo a descrição dos elos e fixados os sistemas de referência, a matriz de transformação homogênea pode ser obtida por:

$$ {}^{i-1}_{i}T=\begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1}d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6} $$

onde *c* é abreviação para cosseno e *s* para seno.

Com a matriz de transformação homogênea e os parâmetros de Denavit-Hartenberg, é possível obter tanto a cinemática inversa quanto a cinemática direta de um manipulador.

Figura 9: Fixando sistemas de referência a elos.

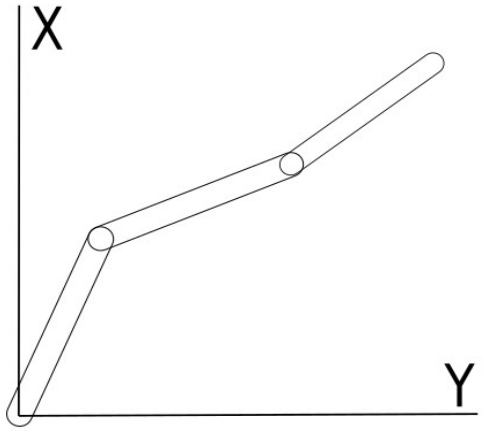


Fonte: CRAIG (2012).

4- MODELAGEM DO MANIPULADOR

O manipulador robótico proposto neste trabalho é do tipo RRR planar, ou seja, com juntas rotacionais proporcionando três graus de liberdade em um único plano. Ainda, não há nenhum efetuador. Este manipulador não apresenta uma grande complexidade em sua modelagem, e foi escolhido pois o objetivo desta monografia é envolver a programação à cinemática inversa. Na prática, um manipulador como este tem apenas utilidade didática, servindo como base para problemas mais complexos envolvendo a cinemática. Um esquema simplificado para o manipulador proposto se encontra na Figura 10.

Figura 10: Esquema simplificado do manipulador RRR.

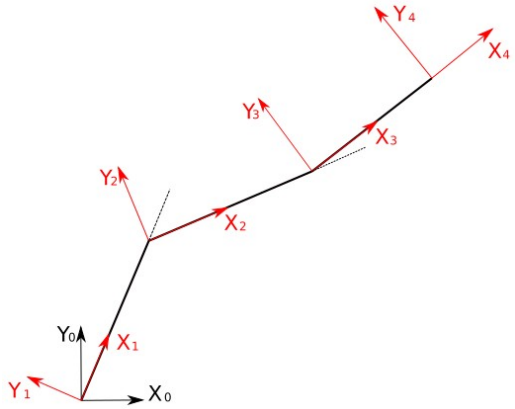


Fonte: Próprio autor.

4.1- PARÂMETROS DE DENAVIT-HARTENBERG E SISTEMAS DE REFERÊNCIA

Como apresentado no Capítulo 3, para obtermos a Cinemática Inversa e Direta, é preciso antes determinar os parâmetros de Denavit-Hartenberg (DH) e escolher um sistema de referência para cada elo. A Figura 11 traz uma representação do manipulador com os sistemas de referência.

Figura 11: Manipulador robótico e sistemas de referência



Fonte: Próprio autor.

Então, é possível obter os parâmetros de DH, conforme a tabela abaixo:

Tabela 1: Parâmetros de Denavit-Hartenberg para o manipulador proposto.

i	α_{i-1}	a_{i-1}	d_i	θ_i
1	0	0	0	θ_1
2	0	L_1	0	θ_2
3	0	L_2	0	θ_3
4	0	L_3	0	0

Fonte: Próprio autor.

Para resolvermos a cinemática inversa e direta do mecanismo, devemos encontrar a matriz de transformação homogênea do sistema de referência 0 para o sistema de referência 4, ${}^{0}_{4}T$.

$$ {}^{0}T_{4} = {}^{0}_{1}T\,{}^{1}_{2}T\,{}^{2}_{3}T\,{}^{3}_{4}T \tag{7} $$

Utilizando a equação (6) e os parâmetros de DH da Tabela 1, podemos obter as matrizes de transformação homogênea da equação (4):

$$ {}^{0}_{1}T = \begin{bmatrix} c\theta_1 & -s\theta_1 & 0 & 0 \\ s\theta_1 & c\theta_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{8} $$

$$ {}^{1}_{2}T = \begin{bmatrix} c\theta_2 & -s\theta_2 & 0 & L_1 \\ s\theta_2 & c\theta_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{9} $$

$$ {}^{2}_{3}T = \begin{bmatrix} c\theta_3 & -s\theta_3 & 0 & L_2 \\ s\theta_3 & c\theta_3 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{10} $$

$$ {}^{3}_{4}T = \begin{bmatrix} 1 & 0 & 0 & L_3 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{11} $$

Assim, obtemos ${}^{0}_{4}T$:

$$ {}^{0}_{4}T = \begin{bmatrix} c\theta_{123} & -s\theta_{123} & 0 & L_1 c\theta_1 + L_2 c\theta_{12} + L_3 c\theta_{123} \\ s\theta_{123} & c\theta_{123} & 0 & L_1 s\theta_1 + L_2 s\theta_{12} + L_3 s\theta_{123} \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{12} $$

onde θ_{12} significa ($\theta_1 + \theta_2$) e θ_{123} significa($\theta_1 + \theta_2 + \theta_3$).

A multiplicação das matrizes e simplificação dos resultados foi feito utilizando a linguagem de programação Python. O código da programação encontra-se no Apêndice A.

4.2- CINEMÁTICA DIRETA

A partir da matriz de transformação homogênea ${}^{0}_{4}T$ da equação (12), podemos obter a cinemática direta:

$$ x = L_1 c\theta_1 + L_2 c\theta_{12} + L_3 c\theta_{123} \tag{13} $$

$$ y = L_1 s\theta_1 + L_2 s\theta_{12} + L_3 s\theta_{123} \tag{14} $$

Ainda, a somatória dos ângulos θ nos dá a orientação do manipulador, ϕ .

$$\phi = \theta_1 + \theta_2 + \theta_3 \tag{15}$$

Estas três equações constituem a cinemática direta. Dados os valores de θ, e conhecendo as dimensões do manipulador, podemos obter sua posição e orientação.

4.3- CINEMÁTICA INVERSA

Na cinemática inversa queremos obter os valores de θ, dada uma posição e orientação. Há duas possíveis abordagens: algébrica e geométrica. Nesta monografia, será utilizada a algébrica.

Primeiro, devemos substituir ϕ nas equações (13) e (14) e isolar as variáveis, obtendo:

$$x - L_3 c\phi = L_1 c\theta_1 + L_2 c\theta_{12} \tag{16}$$

$$y - L_3 s\phi = L_1 s\theta_1 + L_2 s\theta_{12} \tag{17}$$

Para facilitar os cálculos, definimos:

$$x' = x - L_3 c\phi \tag{18}$$

$$y' = y - L_3 s\phi \tag{19}$$

Substituindo nas equações (16) e (17):

$$x' = L_1 c\theta_1 + L_2 c\theta_{12} \tag{20}$$

$$y' = L_1 s\theta_1 + L_2 s\theta_{12} \tag{21}$$

Elevando ao quadrado as equações (20) e (21) e realizando a soma das duas, obtemos:

$$(x')^2 + (y')^2 = (L_1 c\theta_1 + L_2 c\theta_{12})^2 + (L_1 s\theta_1 + L_2 s\theta_{12})^2 \tag{22}$$

Simplificando a equação (22), obtemos:

$$(x')^2+(y')^2=(L_1)^2+2L_1L_2c\theta_2+(L_2)^2 \tag{23}$$

A simplificação da equação (23) foi feita utilizando Python e a biblioteca SymPy. O código se encontra no Apêndice A.

Utilizando a equação (23), podemos isolar θ_2, obtendo

$$\cos(\theta_2)=\frac{(x')^2+(y')^2-(L_1)^2-(L_2)^2}{2L_1L_1} \tag{24}$$

$$sen(\theta_2)=\pm\sqrt{1-(\cos\theta_2)^2} \tag{25}$$

Dessa forma, θ_2 pode ser obtido por

$$\theta_2=atan2(sen(\theta_2),\cos(\theta_2)) \tag{26}$$

Uma vez encontrado θ_2, o próximo passo é obter θ_1.

As equações (20) e (21) podem ser expandidas, respectivamente, para

$$x'=L_1c\theta_1+L_2c\theta_1c\theta_2-L_2s\theta_1s\theta_2 \tag{27}$$

$$y'=L_1s\theta_1+L_2s\theta_1c\theta_2+L_2c\theta_1s\theta_2 \tag{28}$$

sendo possível obter as seguintes equações

$$x'=c\theta_1(L_1+L_2c\theta_2)-s\theta_1L_2s\theta_2 \tag{29}$$

$$y'=s\theta_1(L_1+L_2c\theta_2)+c\theta_1L_2s\theta_2 \tag{30}$$

Definindo k_1 e k_2 como

$$k_1=L_1+L_2c\theta_2 \tag{31}$$

$$k_2 = L_2 s\theta_2 \tag{32}$$

e substituindo em (29) e (30):

$$x' = k_1 c\theta_1 - k_2 s\theta_1 \tag{33}$$

$$y' = k_1 s\theta_1 + k_2 c\theta_1 \tag{34}$$

Então, θ_1 pode ser obtido através da equação

$$\theta_1 = atan(k_1 y' - k_2 x', k_1 x' + k_2 y') \tag{35}$$

Por fim, tendo obtido θ_1 e θ_2, podemos obter θ_3 a partir da equação (15):

$$\theta_3 = \phi - \theta_2 - \theta_1 \tag{36}$$

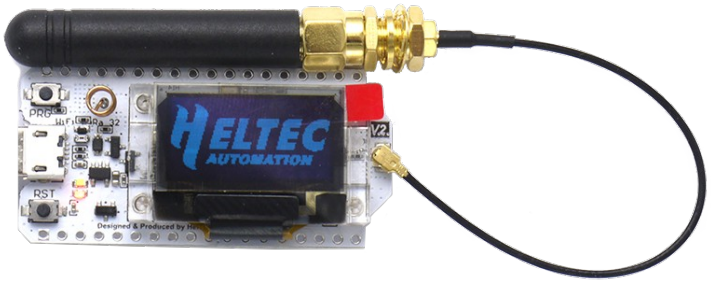
Dessa forma, a cinemática inversa do manipulador em questão pode ser obtida utilizando as equações (18), (19), (24), (25), (26), (31), (32), (35) e (36). É importante notar que essas equações proporcionam duas soluções possíveis, chamadas de “cotovelo para cima” ou “cotovelo para baixo”. Com a cinemática inversa solucionada, podemos resolver essas equações no microcontrolador e obter os ângulos de junta para dada posição e orientação.

5- PROGRAMAÇÃO

Conforme abordado no Capítulo 2, há uma série de microcontroladores e módulos de desenvolvimento no mercado. No contexto de Internet das Coisas, módulos utilizando o ESP32 são extremamente comuns.

Além de ter uma extensa documentação disponível gratuitamente, é possível programar módulos que utilizam o ESP32 pela Arduino IDE, facilitando a programação devido seu alto nível de abstração. Como o objetivo desta monografia é abordar a tecnologia LoRa, o módulo escolhido foi o WiFi LoRa 32 (V2), desenvolvido pela empresa Heltec Automation e apresentado na Figura 12.

Figura 12: Módulo de desenvolvimento WiFi LoRa 32 (V2), da Heltec Automation.



Fonte: (HELTEC, 2022).

Para o trabalho proposto, a conexão WiFi e o display não serão utilizados. Nesta etapa, serão utilizados dois módulos WiFi LoRa, um atuando como transmissor e outro como receptor. O transmissor resolverá as equações da cinemática inversa e enviará para o receptor os ângulos de junta obtidos. O receptor recebe as variáveis e então aciona os atuadores para movimentar o manipulador. A Figura 13 traz este esquema de forma simplificada.

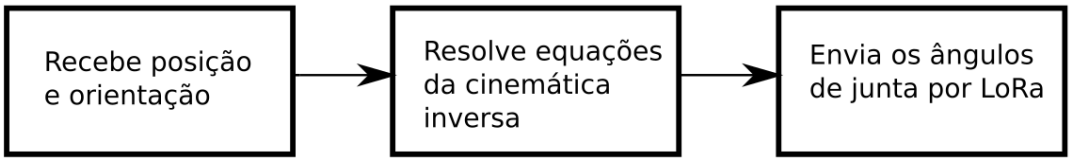
Figura 13: Esquema com módulo transmissor e receptor.

Fonte: Próprio autor

5.1- TRANSMISSOR

O seguinte fluxograma mostra o algoritmo utilizado no transmissor:

Figura 14: Fluxograma para o algoritmo do transmissor



Fonte: Próprio autor.

A posição e orientação (x, y e ϕ) são fornecidos pelo usuário, conforme a Figura 15.

Figura 15: Trecho do código onde são fornecidos x, y e ϕ **.**

```
41  // recebendo x, y e phi
42  Serial.print("Entre com x: ");
43  while (Serial.available() == 0){
44
45  }
46  x = Serial.parseFloat();
47  Serial.println(x);
48  Serial.flush();
```

Fonte: Próprio autor.

Este trecho de código é repetido para y e ϕ também. As funções da biblioteca Serial permitem receber os valores através do Monitor Serial da Arduino IDE, bastando conetar o módulo com um cabo USB ao computador. O valor recebido é do tipo *double*, ou seja, pode receber valores não inteiros.

A cinemática inversa é resolvida utilizando as equações obtidas no Capítulo 4 com o código apresentado na Figura 16. A função *resolveCinematicaInversa* recebe como parâmetro a posição e orientação fornecidas e então obtém θ_1, θ_2 e θ_3. Nota-se que só está sendo resolvida apenas uma solução das duas possíveis.

Figura 16: Resolvendo a cinemática inversa, conforme as equações do Capítulo 4.

```
83  void resolveCinematicaInversa(int x, int y, double phi) {
84    phi = phi * (M_PI/180); // converto phi em radianos
85
86    // obtendo x' e y'
87    xl = x - L3*(cos(phi));
88    yl = y - L3*(sin(phi));
89
90    // obtendo theta2
91    ctheta2 = (pow(xl, 2) + pow(yl, 2) - pow(L1, 2)- pow(L2, 2))/(2*L1*L2); // cosseno de theta2
92    stheta2 = sqrt(1 - pow(ctheta2,2)); // seno de theta2
93    theta2 = atan2(stheta2, ctheta2); // retorna o valor positivo de theta2 em rad
94
95    // obtendo theta1
96    k1 = L1 + L2*ctheta2;
97    k2 = L2*stheta2;
98    theta1 = atan2(k1*yl - k2*xl, k1*xl + k2*yl); // retorna theta1
99
100   // obtendo theta3
101   theta3 = phi - theta1 - theta2;
```

Fonte: Próprio autor.

Uma vez obtido os ângulos de junta, o próximo passo é enviar os valores através de LoRa. O código presente na Figura 17 serve este propósito.

Figura 17: Enviando os ângulos de junta por LoRa.

```
118 // essa funcao inicia o modulo lora e envia o angulo
119 void enviaLoRa (double theta){
120   LoRa.beginPacket();
121   LoRa.setTxPower(14,RF_PACONFIG_PASELECT_PABOOST);
122   LoRa.print(theta);
123   LoRa.endPacket();
124   delay(100);
125 }
```

Fonte: Próprio autor.

A função *enviaLora* recebe um ângulo de junta como parâmetro, empacota a variável e a envia. É possível escolher a potência do LoRa com a função *LoRa.setTxPower.* Uma potência maior permite um alcance maior, mas consome mais energia.

O código completo para o transmissor está disponível no Apêndice B.

5.2- RECEPTOR

O código para o receptor é mais simples, já que seu objetivo principal é receber os dados. A Figura 18 mostra como a recepção dos dados é realizada.

Figura 18: Recepção dos dados enviador pelo transmissor.

```
28  void recebeLoRa (void){
29    Pacote = LoRa.parsePacket();
30    if (Pacote) { // se ha algum dado
31      while (LoRa.available()) {
32        theta[Contador] = LoRa.read(); // salva o angulo recebido
33        Serial.print((char)theta[Contador]);
34      }
35      Serial.println();
36      Serial.flush();
37      movimentaAtuador(theta[Contador]);
38      Contador += 1;
39
40      if (Contador == 3){ // reseta o contador
41        Contador = 0;
42      }
43    }
44  }
```

Fonte: Próprio autor.

Conforme os ângulos são recebidos, a função *LoRa.read()* grava os dados em um vetor θ. No vetor, θ_1 ocupa a posição zero, θ_2 a posição 1 e θ_3 a posição 2. A variável *Contador* serve como índice para o vetor. Uma vez recebido os três ângulos, volta seu valor para zero.

Na figura 16, a função *movimentaAtuador* recebe como parâmetro um ângulo. Esta função é uma abstração, uma vez que não estão sendo utilizados atuadores neste trabalho. Dependendo do tipo de atuador, a função será diferente. Servo motores são mais simples de utilizar no Arduino IDE, graças à biblioteca *servo.h*. Porém, a maioria dos servos disponíveis tem sua movimentação limitada de 0° a 180°. Motores de passo e motores DC não têm essa limitação, mas são mais complexos do ponto de vista de programação e eletrônica.

O código completo do receptor se encontra no Apêndice C.

6- RESULTADOS

Considerando um manipulador com distância entre elos de 8 cm, foram fornecidos dez valores de x, y e ϕ no código do transmissor. A Tabela 2 traz os valores fornecidos e os ângulos de junta obtidos.

Tabela 2: Valores para posição e orientação e os ângulos de junta obtidos.

Posição e orientação			**Ângulos de junta**		
X	**Y**	ϕ	θ_1	θ_2	θ_3
5	5	0	52,34	137,25	-189,59
5	5	90	-99,59	137,25	52,34
5	5	180	-8,44	58,96	129,48
2,5	5	0	79,41	121,56	-200,98
2,5	5	90	-133,29	153,95	69,33
2,5	5	180	-19,11	91,34	107,76
5	7,5	0	51,62	123,15	-174,78
5	7,5	90	-82,73	142,83	29,89
5	7,5	180	5,64	45,32	129,04
7,5	7,5	45	-38,18	166,36	-83,18

Fonte: Próprio autor.

Dependendo do tipo de atuador, os valores obtidos para os ângulos devem ser arredondados. No caso de servo motores, que normalmente atuam de 0° a 180°, valores com 10° de proximidade dos extremos não são lidos corretamente.

O teste de distância do LoRa foi feito em um galpão industrial com 100 metros de comprimento, com máquinas de usinagem no caminho e estoque de lâminas de alumínio. No envio de 100 pacotes, todos foram recebidos. Em teste com distâncias maiores, ocorreu perda de informações e demora no recebimento.

7- CONCLUSÕES

O objetivo deste trabalho foi resolver a cinemática inversa de um manipulador robótico com três graus de liberdade, propor um programa para este e abordar a tecnologia LoRa. Como notado no Capítulo 4, um manipulador como o proposto não possui um fim prático, sendo então a maior finalidade desta monografia o aprendizado sobre Internet das Coisas, programação e robótica, mais especificamente manipuladores.

Neste contexto, a Tabela 2 mostra que a solução proposta produz resultados para uma série de valores. No entanto, é importante frisar que microcontroladores tem performance limitada quando comparado a computadores e notebooks. Dessa forma, um programa em Python poderia gerar resultados diferentes, mas próximos.

Em relação ao LoRa, a documentação oficial promete alcances de até 5 km em áreas urbanas. O teste em galpão industrial produziu resultados inferiores. Isso se deve à antena utilizada no módulo ESP32 e ao seu posicionamento. Em condições ideais, a tecnologia LoRa é excelente para IoT.

Distâncias maiores são alcançadas com melhores antenas e módulos posicionados em locais altos. Ainda, é possível reduzir a perda de dados utilizando um sistema de “acknowledge”, ou seja, caso não seja confirmado o recebimento da informação, o transmissor envia novamente o pacote. No entanto, a tecnologia é promissora e kits de desenvolvimento a tornam acessível.

Como proposta de melhoria e continuidade a este trabalho, há a adição de um efetuador ao manipulador e a construção do mesmo, utilizando motores de passo como atuadores (e sua programação). Ainda, propõe-se a adição de uma confirmação ao receber os dados.

8- REFERÊNCIAS

CRAIG, JOHN J. **Robótica**. 3.ed. Pearson education do Brasil, 2012.

HELTEC. Heltec Automation, c2022. Página do produto WiFi LoRa 32 (V2). Disponível em: https://heltec.org/project/wifi-lora-32/. Acesso em: 07 jan. 2022.

LORA ALLIANCE. **What is Lorawan**. San Ramon, 2015. Disponível em: https://lora-alliance.org/resource_hub/what-is-lorawan/. Acesso em: 26 jul. 2021.

LUTKEVICH, Ben. Microcontroller (MCU). TechTarget, 2019. Disponível em: https://internetofthingsagenda.techtarget.com/definition/microcontroller. Acesso em: 24 jul. 2021

LYNCH, KEVIN M.; PARK, FRANK C. **Modern Robotics: Mechanics, Planning and Control**. 1st ed. Cambrige: Cambrige University Press, 2017.

MICROCHIP. Microchip, c2022. Página do produto ATmega328. Disponível em: https://www.microchip.com/en-us/product/ATMEGA328. Acesso em: 07 jan. 2022.

OLIVEIRA, SÉRGIO. **Internet das Coisas com ESP8266, Arduino e Raspberry PI**. 2.ed. São Paulo: Novatec Editora, 2021.

ROSÁRIO, JOÃO M. **Princípios de Mecatrônica**. 9.ed. Pearson education do Brasil, 2013.

SEMTECH CORPORATION. **Lora and LoraWAN: a technical overview**. Camarillo, 2019. Disponível em: https://lora-developers.semtech.com/library/tech-papers-and-guides/lora-and-lorawan/. Acesso em: 25 jul. 2021.

STEVAN, SERGIO L.; LEME, MURILO O.; SANTOS, MAX M. D. **Indústria 4.0 : fundamentos, perspectivas e aplicações**. 1.ed. São Paulo: Érica, 2018.

APÊNDICE A – Programa em Pyrthon para obter a cinemática inversa

Este apêndice apresenta o código de programação feito em Python utilizando a biblioteca SymPy para obter as matrizes de transformação homogênea. A Figura 19 apresenta o código completo.

Figura 19: Código de programação em Python para obter as matrizes de transformação homogênea.

```
[1]: from sympy import *

[2]: # definindo as variaveis
     theta1, theta2, theta3, l1, l2, l3 = symbols('theta1 theta2 theta3 l1 l2 l3')

[3]: # matriz transformação
     # Transformação de 1 para 0
     M01T = Matrix([[cos(theta1), -sin(theta1), 0, 0], [sin(theta1), cos(theta1), 0, 0], [0, 0, 1, 0], [0, 0, 0, 1]])
     # transformação de 2 para 1
     M12T = Matrix([[cos(theta2), -sin(theta2), 0, l1], [sin(theta2), cos(theta2), 0, 0], [0, 0, 1, 0], [0, 0, 0, 1]])
     # transformação de 3 para 2
     M23T = Matrix([[cos(theta3), -sin(theta3), 0, l2], [sin(theta3), cos(theta3), 0, 0], [0, 0, 1, 0], [0, 0, 0, 1]])
     # transformação de 4 para 3
     M34T = Matrix([[1, 0, 0, l3], [0, 1, 0, 0], [0, 0, 1, 0], [0, 0, 0, 1]])

[4]: M03T = M01T*M12T*M23T # obtendo transformação de 3 para 0
     M03T = M03T.applyfunc(simplify) # simplificando

[5]: M04T = (M03T*M34T).applyfunc(simplify) # obtendo a transformação de 0 para 4 e simplificando

[6]: X, Y, exp1, exp2, exp3 = symbols('X Y exp1 exp2 exp3')
     exp1 = l1*cos(theta1) + l2*cos(theta1 +theta2) # X
     exp2 = l1*sin(theta1) + l2*sin(theta1 + theta2) # Y
     exp3 = exp1**2 + exp2**2 # elevando ao quadrado e somando

[7]: exp3 = expand(exp3) # expandindo a equação

[8]: exp3 = simplify(exp3) # simplificando

[9]: equacao1 = Eq(X**2 + Y**2, exp3)
     # utilizada para encontrar theta2
```

Fonte: Próprio autor.

APÊNDICE B – Programa em C para obter o módulo transmissor

Este apêndice contém o código de programação feito em C utilizando a Arduino IDE para o transmissor. A Figura 20 apresenta todo o código.

Figura 20: Código de programação em C utilizado no transmissor.

```
#include <math.h>
#include <stdlib.h>
#include "heltec.h"


#define BAND    915E6  // frequencia do LoRa, no caso 915MHz

// comprimento das juntas
int L1 = 8;
int L2 = 8;
int L3 = 8;

// variaveis de posicao e orientacao
double x;
double y;
double phi;

// x' e y'
double x1;
double y1;

// angulos de junta e variaveis auxiliares
double theta1, theta2, theta3;
double ctheta2, stheta2; // cos e sen
double k1, k2;

// prototipos de funcao
void resolveCinematicaInversa(int x, int y, double phi);
double radianosParaGraus (double radiano);
void enviaLoRa (double theta);

void setup() {

Heltec.begin(true, true, true, true, BAND); // iniciando o modulo
Serial.begin(115200); // iniciando a serial

}

void loop() {

  // recebendo x, y e phi
  Serial.print("Entre com x: ");
  while (Serial.available() == 0){

  }
  x = Serial.parseFloat();
  Serial.println(x);
  Serial.flush();

  Serial.print("Entre com y: ");
  while (Serial.available() == 0){

  }
  y = Serial.parseFloat();
  Serial.println(y);
  Serial.flush();
```

```
57
58    Serial.print("Entre com phi: ");
59    while (Serial.available() == 0){
60
61    }
62    phi = Serial.parseFloat();
63    Serial.println(phi);
64    Serial.flush();
65
66
67   resolveCinematicaInversa(x, y, phi);
68
69   enviaLoRa(theta1);
70   enviaLoRa(theta2);
71   enviaLoRa(theta3);
72
73   Serial.print("theta1 = ");
74   Serial.print(theta1);
75   Serial.print(", theta2 = ");
76   Serial.print(theta2);
77   Serial.print(", theta3 = ");
78   Serial.print(theta3);
79   Serial.print("\n");
80
81 }
94
95    // obtendo theta1
96    k1 = L1 + L2*ctheta2;
97    k2 = L2*stheta2;
98    theta1 = atan2(k1*y1 - k2*x1, k1*x1 + k2*y1); // retorna theta1
99
100   // obtendo theta3
101   theta3 = phi - theta1 - theta2;
102
103   // convertendo para graus
104   theta1 = radianosParaGraus(theta1);
105   theta2 = radianosParaGraus(theta2);
106   theta3 = radianosParaGraus(theta3);
107
108 }
109
110 double radianosParaGraus (double radiano){
111   double graus;
112
113   graus = radiano*(180/M_PI);
114
115   return graus;
116 }
117
118 // essa funcao inicia o modulo lora e envia o angulo
119 void enviaLoRa (double theta){
120   LoRa.beginPacket();
121   LoRa.setTxPower(14,RF_PACONFIG_PASELECT_PABOOST);
122   LoRa.print(theta);
123   LoRa.endPacket();
124   delay(100);
125 }
126
```

Fonte: Próprio autor.

APÊNDICE C – Programa em C para o receptor

Este apêndice apresenta o código de programação feito em C utilizando a Arduino IDE para o receptor. A Figura 21 contém todo o código.

Figura 21: Código de programação em C para o receptor.

```
#include "heltec.h"

#define BAND    915E6  // frequencia do LoRa, no caso 915MHz

double theta[3]; // vetor com 3 posicoes
// theta[0] recebe theta1
// theta[1] recebe theta2
// theta[2] recebe theta3

int Pacote;
int Contador = 0; // utilizado como posicao do vetor theta

// prototipos de funcao
void recebeLoRa (void);
void movimentaAtuador (double angulo);

void setup() {

Heltec.begin(true, true, true, true, BAND); // iniciando o modulo
Serial.begin(115200); // iniciando a serial

}

void loop() {
  recebeLoRa();
}

void recebeLoRa (void){
  Pacote = LoRa.parsePacket();
  if (Pacote) { // se ha algum dado
    while (LoRa.available()) {
      theta[Contador] = LoRa.read(); // salva o angulo recebido
      Serial.print((char)theta[Contador]);
    }
    Serial.println();
    Serial.flush();
    movimentaAtuador(theta[Contador]);
    Contador += 1;

    if (Contador == 3){ // reseta o contador
      Contador = 0;
    }
  }
}

void movimentaAtuador(double angulo){
  // dependendo do tipo de atuador funcoes diferentes devem ser utilizadas
}

```

Fonte: Próprio autor.