***Relatório de prática da disciplina de Robótica – 2025.1***

**João Gustavo Cavalcanti Beltrão da Silva1,2**  [orcid.org/0000-0001-8392-3115](https://orcid.org/0000-0001-8392-3115)

**Davi Viana Gouveia** **1,2**  <https://orcid.org/0009-0008-8432-8391>

**José Lucas Bessa de Oliveira 1,2**  [orcid.org/0000-0002-5830-1143](http://orcid.org/0000-0002-5830-1143)

1 Escola Politécnica de Pernambuco, Universidade de Pernambuco, Recife, Brasil,

2  Graduação em Engenharia de Controle e Automação, Escola Politécnica de Pernambuco, Pernambuco, Brasil,

**Git Hub do projeto:** [**https://github.com/jgc77/Manipulador-2EE**](https://github.com/jgc77/Manipulador-2EE)

**Resumo**

Este artigo apresenta o desenvolvimento de um manipulador robótico com três graus de liberdade, configurado com três juntas rotacionais (RRR), de configuração antropomórfica e controlado por meio de um Arduino. O sistema foi estruturado com uma arquitetura modular, separando os componentes de hardware dos cálculos de cinemática, o que facilitou a organização e o desenvolvimento do projeto. Foram realizadas medições dos elos do robô para permitir a aplicação correta dos cálculos de cinemática direta e inversa, além da montagem mecânica das juntas e elos utilizando peças impressas em 3D. O controle dos motores de passo foi feito com drivers dedicados, sendo que todos dos motores (28BYJ-48) foram modificado para funcionar como motor bipolar, ampliando sua precisão e controle. Para organizar e documentar o projeto, foi criado um repositório no GitHub, e bibliotecas específicas, como controlMotor e math.h, foram desenvolvidas para facilitar o controle e a reutilização do código.

**Palavras-Chave*:*** manipulador robótico, juntas rotacionais, Arduino, cinemática direta e inversa, motores de passo.

***Abstract***

*This article presents the development of a robotic manipulator with three degrees of freedom, configured with three rotational joints (RRR) and controlled using an Arduino platform. The system was structured with a modular architecture, separating hardware components from kinematic calculations, which facilitated the organization and development of the project. Measurements of the robot's links were carried out to enable the correct application of forward and inverse kinematics calculations, along with the complete mechanical assembly of the joints and links using 3D-printed parts. Stepper motors were controlled using dedicated drivers, and one of the motors (28BYJ-48) was modified to operate as a bipolar motor, improving its precision and control. A GitHub repository was created to organize and document the project, and specific libraries, such as main and controlMotor, were developed to simplify control and enable code reuse.*

***Key-words:*** *robotic manipulator, rotational joints, Arduino, forward and inverse kinematics, stepper motors.*

## Introdução

Os manipuladores robóticos são sistemas mecânicos desenvolvidos para reproduzir movimentos semelhantes muitas vezes de um braço humano, sendo amplamente utilizados em processos automatizados que exigem precisão e repetibilidade. Um tipo comum de manipulador é o que possui três graus de liberdade com juntas rotacionais (RRR), de configuração antropomórfica, permitindo que o sistema realize movimentos em diferentes direções por meio da rotação de seus elos. Esse tipo de configuração é especialmente útil em aplicações que exigem movimentação em planos tridimensionais, como em linhas de montagem e tarefas de manipulação leve.

Para projetar e controlar corretamente um manipulador (RRR), de configuração antropomórfica é essencial aplica os conceitos de cinemática direta e inversa. A cinemática direta permite determinar a posição final do atuador (ou ferramenta) a partir dos ângulos das juntas, enquanto a cinemática inversa realiza o processo inverso: calcula os ângulos necessários das juntas para alcançar uma determinada posição no espaço. Para isso, foram feitas medições precisas dos elos do robô, servindo como base para os cálculos matemáticos.

Durante o desenvolvimento do projeto, foram utilizadas técnicas de programação no Arduino para integrar os motores de passo ao controle das juntas, além do usa de bibliotecas específicas para organizar o código e facilitar a modularização do sistema.

## Materiais e Métodos

Para a realização da prática referente ao manipulador robótico com três graus de liberdade com juntas rotacionais (RRR), de configuração antropomórfica foi utilizado os seguintes componentes:

* 1x Arduino uno
* 3x Drive a4988
* 3x Capacitor eletrolítico 100uf 50v
* 1x Fonte 12v 2a
* 1x Protoboard
* 1x Conector P4 fêmea
* 1x Conector p4 macho
* 1x Chave alavanca
* 1x Chave gangorra 2 terminais
* 1x Kit jumpers macho-fêmea
* 3x Motores de passo 28BYJ-48
* 3x Elos impressos em 3d

### Análise Teórica

### 

Para compreender o comportamento do manipulador robótico RRR, de configuração antropomórfica é fundamental realizar a análise teórica baseada nos princípios da cinemática. Essa análise permite descrever matematicamente os movimentos do sistema e prever com precisão a posição do efetuador final com base nas configurações das juntas. A cinemática é dividida em duas etapas principais: cinemática direta e cinemática inversa.

Na cinemática direta, determina-se a posição e a orientação do efetuador a partir dos ângulos das juntas e dos comprimentos dos elos do robô. Essa análise é essencial para saber onde o braço robótico estará localizado no espaço tridimensional após receber determinados comandos.

Para realizar a **cinemática direta** do manipulador, o processo é relativamente simples, já que o objetivo do código não é calcular movimentos, mas apenas determinar a posição x, y, z do efetuador final em relação à base do robô. Seguimos os seguintes passos:

* **Definição dos parâmetros de Denavit-Hartenberg (D-H):**  
  Modelamos o manipulador considerando-o na posição “em pé”, ou seja, vertical em relação à base.
* **Montagem das matrizes de transformação:**  
  A partir dos parâmetros D-H, calculamos as matrizes de transformação entre os elos.
* **Implementação no código:**  
  Pegamos a equação final da cadeia de transformações e a implementei no código.

A entrada do sistema são os três ângulos das juntas , , , e a saída é a posição cartesiana x, y, z do manipulador.

Já a cinemática inversa é o processo de calcular os ângulos das juntas que o manipulador deve adotar para alcançar uma posição desejada no espaço. Essa etapa é especialmente desafiadora, pois pode haver múltiplas soluções (ou nenhuma), dependendo da posição alvo e das limitações físicas do robô.

Lógica da Cinemática Inversa (cin\_inv)

Dado um ponto no espaço cartesiano definido por coordenadas (x, y, z), como foi dito o objetivo da cinemática inversa é encontrar os ângulos das juntas do manipulador:

θ₁: ângulo da base

θ₂: ângulo do ombro

θ₃: ângulo do cotovelo

Cálculo de θ₁ (ângulo da base):

Este é o ângulo mais simples de determinar. Basta utilizar a função arco-tangente para encontrar o ângulo entre o eixo X e o ponto projetado no plano XY:

Redução para manipulador plano:

Com θ₁ determinado, o problema tridimensional é reduzido a um plano bidimensional (plano R-R), considerando apenas os comprimentos dos elos e a altura. Define-se dois valores auxiliares:

r: distância horizontal no plano XY

s: altura relativa ao elo L₁

Cálculo de θ₃ (ângulo do cotovelo):

Utiliza-se a Lei dos Cossenos no triângulo formado pelos elos L₂, L₃ e a linha direta entre o ombro e o ponto final (r, s):

O valor de θ₃ é obtido com:

Observação: É adotada a convenção do cotovelo voltado para cima (solução "cotovelo para cima"), portanto escolhe-se a solução com o maior valor possível de θ₃ (ângulo interno obtuso).​

Cálculo de θ₂ (ângulo do ombro):

Para encontrar θ₂, utiliza-se dois ângulos auxiliares:

β (beta): ângulo interno entre os elos L₂ e L₃

γ (gama): ângulo formado entre a linha do ombro ao ponto (r, s) e o eixo horizontal

Com isso:

Para viabilizar a implementação prática dessa lógica no sistema embarcado, foram desenvolvidas duas bibliotecas principais: ControlMotor e math.h.

A biblioteca ControlMotor é responsável por integrar a biblioteca AccelStepper.h com os drivers A4988, além de automatizar toda a configuração inicial do sistema de controle dos motores. Ela define os pinos dos drivers, configura a aceleração e a velocidade dos motores, inicializa os sinais de ativação dos drivers como LOW (nível lógico 0) e implementa a lógica da chave seletora (chave gangorra), utilizando um atraso controlado para evitar efeitos de bouncing. Além disso, essa biblioteca converte os valores angulares calculados pela cinemática em valores discretos de passos para o acionamento preciso dos motores, e também implementa os limites físicos do manipulador e o modo de calibração manual.

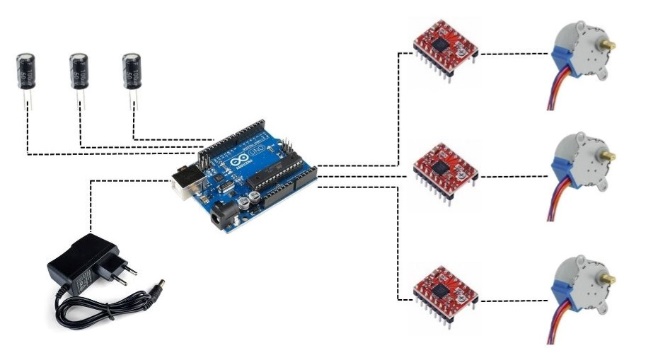
Já a biblioteca padrão math.h é utilizada para realizar operações matemáticas fundamentais no contexto do cálculo da cinemática direta, como funções trigonométricas (sin, cos, atan2) e manipulação de radianos. Essas operações são essenciais para a conversão de ângulos em coordenadas cartesianas e para o correto funcionamento dos algoritmos que envolvem movimentos coordenados dos elos do manipulador.

### Simulação

Para validar o funcionamento do manipulador robótico RRR antes da montagem final, foi desenvolvida uma representação teórica e visual do sistema. Essa representação buscou simular o comportamento do robô com base nos componentes reais utilizados.

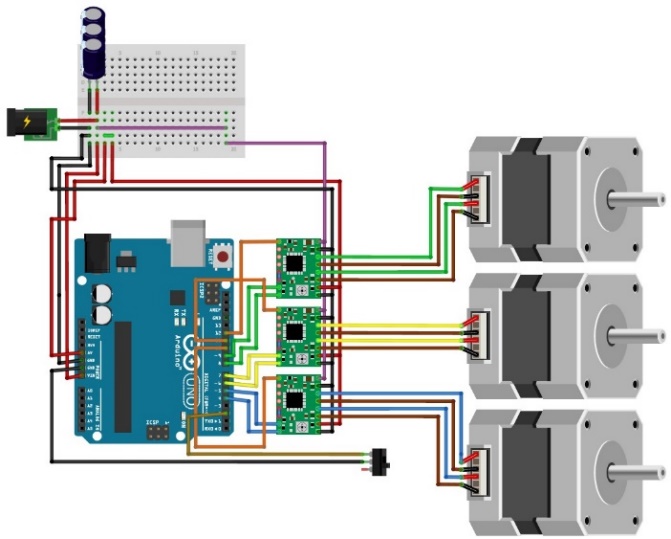
Na simulação, consideramos o uso de três motores de passo 28BYJ-48, que foram modificados para funcionar no modo bipolar. Cada motor foi conectado a um driver A4988, que realiza o controle de corrente e direção com maior estabilidade. A comunicação e o controle dos motores foram gerenciados por meio de uma placa Arduino Uno, que executa os códigos implementados nas bibliotecas main e ControlMotor.

Abaixo, apresentamos a representação teórica criada para ilustrar o funcionamento do sistema com os três motores conectados ao Arduino e aos drivers:

****

**Figura 01:** Representação da estrutura do manipulador robótico.

Além da representação teórica, também foi elaborado um diagrama elétrico completo do sistema, utilizando um fritzing de simulação específico para projetos com Arduino. Esse diagrama detalha todas as conexões entre os componentes, incluindo os pinos digitais do Arduino, os sinais de controle dos A4988, a alimentação dos motores e os demais elementos que compõem a arquitetura do projeto.

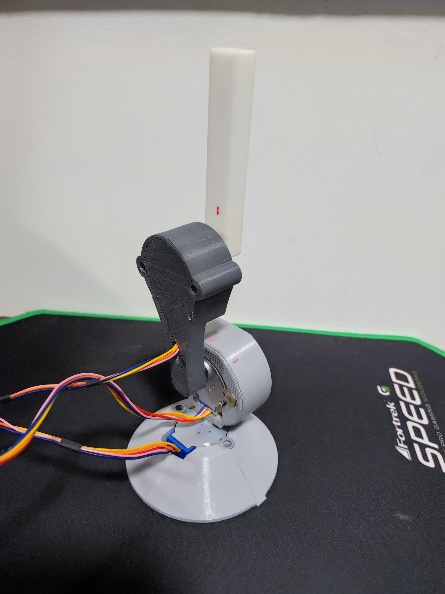
****

**Figura 02:** Representação da estrutura do manipulador robótico.

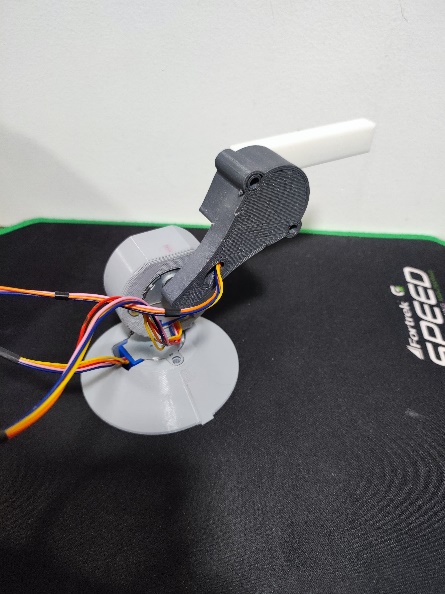
## Resultados

Durante os testes práticos, o manipulador foi colocado em diferentes posições de forma controlada, demonstrando a eficácia do sistema tanto na parte mecânica quanto na lógica de movimentação. As posições foram calculadas com base na representação teórica e executadas com boa estabilidade, respeitando os limites físicos definidos no código.

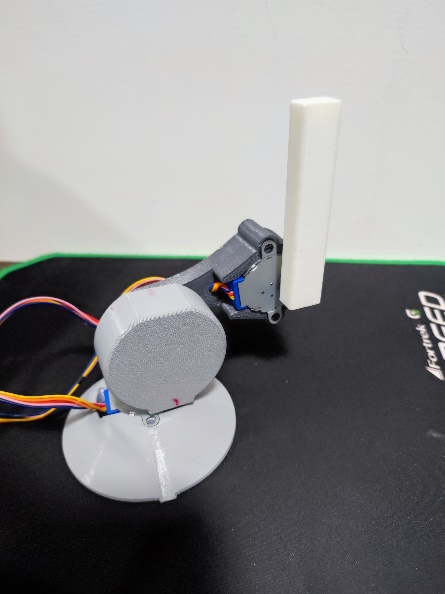
A seguir, apresentamos imagens da montagem final do manipulador, bem como exemplos das posições alcançadas pelo braço robótico durante a execução dos testes:

****

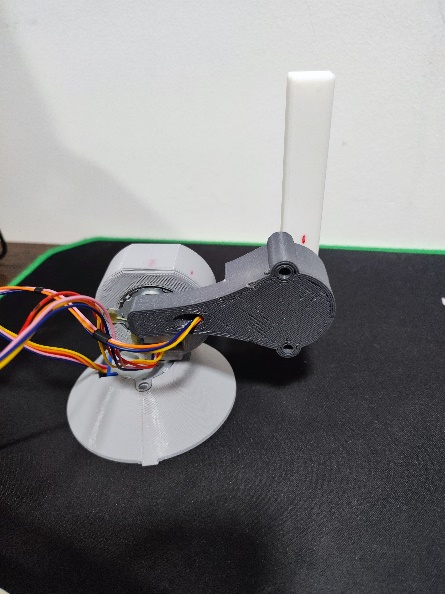
**Figura 03:** Manipulador na posição inicial dos nossos cálculos (0, 0, 0).

****

**Figura 04:** Manipulador na posição (90, -45, -45).

****

**Figura 05:** Manipulador na posição (-130, 90, -80).

****

**Figura 06:** Manipulador na posição (130, -80, 90).

1. **Discussão**

Após realizar uma análise comparativa dos resultados teóricos, práticos e simulados, constatou-se que a organização modular do software separando leitura de comandos, lógica de controle e acionamento de motores foi fundamental para o bom desempenho do sistema. Implementar a cinemática direta foi relativamente rápido: bastou codificar as transformadas homogêneas e verificar se a posição calculada batia com a observada.

Já a cinemática inversa tem sido mais trabalhosa. Converter um alvo cartesiano em três ângulos viáveis, sem infringir limites mecânicos ou gerar soluções ambíguas, exigiu um refinamento do modelo matemático e tratamento cuidadoso de singularidades.

A etapa de simulação mostrou‑se essencial para evitar erros de fiação ou escolha de pinos. Mesmo assim, alguns ajustes finos como correntes dos A4988 e compensação de jogo mecânico foram feitos apenas após os primeiros testes reais.

Em resumo, o projeto atingiu os objetivos iniciais: montar um manipulador funcional, validar a cinemática direta e criar a base para a cinemática inversa.

## Conclusões

Neste artigo, discutimos o funcionamento do manipulador robótico com três graus de liberdade. O mesmo foi finalizado com sucesso, cumprindo os objetivos propostos desde a fase de planejamento até os testes práticos. Conseguimos montar toda a estrutura física, implementar o controle dos motores via Arduino e aplicar tanto a cinemática direta quanto a inversa de forma funcional. Apesar dos desafios, especialmente na parte da cinemática inversa, o sistema se mostrou preciso.

A separação do código em bibliotecas facilitou bastante a organização do projeto e a realização de ajustes durante os testes. Além disso, o uso da simulação e do diagrama elétrico ajudou a montagem na prática. No final, o manipulador conseguiu executar movimentos programados de forma controlada, validando toda a lógica construída ao longo do processo.

**Referências**

**[1]** ARDUFOCUS. Como modificar o motor de passo 28BYJ-48 para bipolar. ArduFocus, [s.d.]. Disponível em: https://ardufocus.com/howto/28byj-48-bipolar-hw-mod/. Acesso em: 7 jul. 2025.

**[2]** ELETROGATE. Tudo sobre DRIVER A4988 e motor de passo usando o Arduino. Blog Eletrogate, 29 mar. 2018 (atualizado em 9 set. 2022). Disponível em: https://blog.eletrogate.com/driver-a4988-motor-de-passo-usando-o-arduino/. Acesso em: 7 jul. 2025.

**[3]** AIRSPAYCE. AccelStepper Class Reference. [S.l.]: Airspayce, [s.d.]. Disponível em: https://www.airspayce.com/mikem/arduino/AccelStepper/classAccelStepper.html. Acesso em: 7 jul. 2025.