Universidade de Aveiro
2025

Beatriz
Marques

Desenvolvimento e Operacionalização de uma Mão Robótica Antropomórfica

Development and Operationalization of an Anthropomorphic Robotic Hand

Universidade de Aveiro
2025

Beatriz
Marques

Desenvolvimento e Operacionalização de uma Mão Robótica Antropomórfica

Development and Operationalization of an Anthropomorphic Robotic Hand

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestre em Robótica e Sistemas Inteligentes, realizada sob a orientação científica do Doutor Vítor Manuel Ferreira dos Santos, Professor associado c/ agregação do Departamento de Engenharia Mecânica da Universidade de Aveiro, do Doutor Filipe Miguel Teixeira Pereira da Silva, Professor auxiliar do Departamento de Electrónica, Telecomunicações e Informática da Universidade de Aveiro.

o júri / the jury

presidente / president

Professor Doutor José Nuno Panelas Nau
Professor Associado da Universidade de Aveiro

vogais / examiners committee

Professor Doutor João Paulo Morais Ferreira
Professor Coordenador do Instituto Superior de Engenharia de Coimbra (Arguente Principal)

Professor Doutor Vítor Manuel Ferreira dos Santos
Professor Associado C/ Agregação da Universidade de Aveiro (Orientador)

agradecimentos /
acknowledgements

Gostaria de expressar a minha sincera gratidão a todas as pessoas que contribuíram, direta ou indiretamente, para a concretização deste trabalho.

Aos meus pais, pelo apoio incondicional, por estarem sempre ao meu lado e por acreditarem em mim em todas as etapas deste percurso. Ao meu namorado, pelo carinho, paciência e constante incentivo, especialmente nos momentos mais desafiantes.

Agradeço aos meus professores pelas sugestões, orientações e disponibilidade ao longo do desenvolvimento deste trabalho, que foram fundamentais para alcançar os melhores resultados possíveis.

Um agradecimento especial ao Engenheiro Rui Heitor, pelo apoio técnico e pelos esclarecimentos prestados nas questões relacionadas com fabrico e soldadura.

À Professora Petia Georgieva, agradeço pelos conselhos valiosos e pela orientação na fase final do projeto, em particular no que diz respeito à componente de aprendizagem automática.

A todos, o meu muito obrigada.

palavras-chave

Aprendizagem Automática, Arquitetura ROS2, Controlo Baseado em Corrente, Deteção de Sobreposição de Dedos, Mão Robótica Multi-Dedos, Perceção tátil

resumo

Esta dissertação apresenta como objetivo o desenvolvimento e operacionalização de uma mão robótica antropomórfica. O projeto incluiu o fabrico e montagem da estrutura mecânica, a integração de sensores piezoresistivos para deteção de contacto, e a implementação de um sistema de controlo modular em ROS2, capaz de executar movimentos básicos de manipulação. Durante a fase de testes, identificou-se a ocorrência de sobreposições entre dedos durante a abertura da mão, resultantes de uma ordem de ativação inadequada, com risco de colisões internas e esforço excessivo nos motores. Para resolver este problema, foi desenvolvido um sistema de aquisição de dados e treinado um modelo de aprendizagem automática para classificar automaticamente diferentes tipos de sobreposição. Entre os modelos testados, o *Support Vector Machine* (SVM) revelou o melhor desempenho. Os resultados demonstram elevada precisão na deteção de configurações críticas, contribuindo para a robustez do sistema e abrindo caminho à implementação futura de estratégias de controlo preventivo.

keywords

Current-Based Control, Finger Overlap Detection, Machine Learning, Multi-Finger Robotic Hand, ROS2 Architecture, Tactile Perception

abstract

This dissertation aims to develop and operationalize an anthropomorphic robotic hand. The project included the fabrication and assembly of the mechanical structure, the integration of piezoresistive sensors for contact detection, and the implementation of a modular control system using ROS2, capable of executing basic manipulation movements. During the testing phase, finger overlaps were observed during the hand opening sequence, caused by an inadequate activation order, leading to risks of internal collisions and excessive stress on the motors. To address this issue, a data acquisition system was developed, and a machine learning model was trained to automatically classify different types of finger overlap. Among the tested models, the Support Vector Machine (SVM) achieved the best performance. The results demonstrate high accuracy in detecting critical configurations, contributing to the system's robustness and paving the way for future implementation of preventive control strategies.

acknowledgement of use of AI tools

Recognition of the use of generative Artificial Intelligence technologies and tools, software and other support tools.

Reconheço a utilização do ChatGPT 4.0 (Open AI, https://chat.openai.com) para para melhoria e auxílio na revisão da escrita da dissertação.

Conteúdo

Lista de Figuras

Lista de Tabelas

Lista de Excertos de Código

Lista de Siglas e Acrónimos

CAD	Computer-Aided Design
DDS	Data Distribution Service
DIP	Distal Interphalangeal Joint
DIP	Dual In-line
FSR	Force Sensing Resistor
MCP	Metacarpophalangeal Joint
PCB	Printed Circuit Board
PIP	Proximal Interphalangeal Joint
PVDF	Polyvinylidene fluoride
RBF	Radial Basis Function
ROS	Robot Operating System
SVM	Support Vector Machine
URDF	Unified Robot Description Format

CAPÍTULO 1

Introdução

1.1 Enquadramento

O avanço contínuo da robótica e da inteligência artificial tem impulsionado o desenvolvimento de sistemas cada vez mais sofisticados e adaptáveis, capazes de interagir de forma segura e eficiente com o ambiente envolvente. Entre estes sistemas, destacam-se as mãos robóticas antropomórficas, cujo *design* e funcionalidades procuram aproximar-se das capacidades motoras e sensoriais da mão humana. Estas estruturas apresentam um elevado potencial de aplicação em áreas como a indústria, a reabilitação, a assistência pessoal e a investigação científica.

A manipulação com múltiplos dedos representa um dos maiores desafios em robótica, tanto pela complexidade dos mecanismos envolvidos como pela elevada dimensionalidade do espaço de controlo. Ao longo das últimas décadas, diversos autores têm contribuído para a compreensão e desenvolvimento de mãos robóticas com capacidades semelhantes às humanas. Cutkosky [1] estabeleceu as bases para a escolha de tipos de preensão em contextos industriais, enquanto Bicchi [2] realçou a dificuldade inerente à criação de mãos robóticas que conciliem destreza com robustez, propondo abordagens que privilegiem a simplicidade mecânica sem comprometer a funcionalidade. Mais recentemente, Feix et al. [3] apresentaram uma taxonomia abrangente de preensões humanas, que serve como guia para o planeamento e avaliação de estratégias de preensão robótica.

O progresso em manipulação robótica não se limita apenas à engenharia mecânica e controlo, mas também à integração de métodos de aprendizagem e perceção inteligente. Billard e Kragic [4] discutem tendências emergentes na área, incluindo a aprendizagem baseada em demonstração e a fusão sensorial, enquanto Piazza et al. [5] fornecem uma perspetiva histórica sobre a evolução das mãos robóticas ao longo de um século. Trabalhos mais recentes, como os de Shang et al. [6] e Li et al. [7], demonstram como a aprendizagem profunda tem sido utilizada para melhorar a manipulação com mãos multifuncionais, especialmente em ambientes não estruturados.

No entanto, a construção e controlo destas mãos robóticas continuam a apresentar desafios significativos. A complexidade mecânica e eletrónica dos atuadores e sensores requer soluções modulares e escaláveis, capazes de integrar diferentes fontes de informação e adaptar-se a variações nas interações físicas. Em particular, a perceção tátil e a interpretação de contactos são essenciais para evitar colisões indesejadas e otimizar a execução de tarefas de preensão.

Neste contexto, o presente trabalho insere-se no desenvolvimento de uma mão robótica antropomórfica, equipada com sensores de força para deteção de contacto, e controlada por um sistema modular. Durante o processo de desenvolvimento, verificou-se que a sobreposição dos dedos ocorre durante o fecho da mão, o que posteriormente compromete a abertura adequada e pode gerar tensões excessivas nos motores, afetando a integridade do sistema. Este problema motivou a exploração de técnicas de aprendizagem automática para a classificação automática de diferentes tipos de sobreposições, permitindo otimizar o controlo da mão e prevenir situações de bloqueio ou colisão entre os dedos.

1.2 Objetivos

O principal objetivo deste trabalho é o desenvolvimento de uma mão robótica antropomórfica equipada com sensores de contacto e sistemas de controlo adequados, capaz de realizar movimentos básicos de manipulação. Para alcançar este objetivo global, foram definidos os seguintes objetivos específicos:

- **Seleção, fabrico e montagem da estrutura da mão robótica** — Identificar e documentar a solução a implementar, selecionando os componentes necessários para o fabrico da mão, e proceder à montagem, considerando as diretivas de documentação existentes e as necessidades de adaptação ao robô colaborativo já disponível.
- **Incorporação de sensores** — Integrar sensores de contacto ou força na mão robótica, complementando o *feedback* de posição fornecido pelos servomotores, de forma a permitir a deteção de forças aplicadas nos dedos.
- **Incorporação da unidade de processamento** — Definir e implementar a unidade de processamento responsável pelo controlo dos motores, pela recolha e processamento de dados sensoriais.
- **Desenvolvimento do programa principal** — Desenvolver o *software* necessário para o controlo da mão, assegurando a integração com um ambiente adequado.
- **Implementação de ações básicas de movimentação de dedos** — Criar programas ou macros para movimentos específicos de cada dedo, como abrir ou fechar, posicionar em ângulos predefinidos, formar gestos e executar ações de agarrar ou largar objetos com dois ou mais dedos.

Durante o processo de desenvolvimento e implementação dos objetivos, identificou-se um problema relevante relacionado com a sobreposição de dedos durante o fecho da mão, o que compromete a abertura subsequente e pode afetar o funcionamento seguro do sistema. Como resposta a esta necessidade, foi definida uma tarefa adicional: o desenvolvimento de uma solução baseada em aprendizagem automática para a classificação automática das

configurações de sobreposição de dedos, de forma a otimizar a ordem de abertura e garantir o funcionamento seguro e eficiente da mão robótica.

1.3 Estrutura do documento

Este documento organiza-se em seis capítulos, estruturados da seguinte forma:

- O **Capítulo 2** descreve os trabalhos relacionados, abordando as soluções existentes no domínio das mãos robóticas antropomórficas e tecnologias de sensores de contacto.
- O **Capítulo 3** detalha as ferramentas de suporte utilizadas, incluindo *hardware*, *software* e bibliotecas aplicadas no desenvolvimento do sistema.
- O **Capítulo 4** expõe o processo de desenvolvimento da mão robótica, desde a montagem física, configuração dos motores e implementação do sistema sensorial, até à programação do sistema completo.
- O **Capítulo 5** concentra-se na classificação de sobreposições de dedos com aprendizagem automática, apresentando o processo de aquisição de dados, os modelos desenvolvidos, a avaliação de desempenho e os resultados obtidos. Este capítulo resultou diretamente da necessidade de resolver o problema identificado durante o desenvolvimento do projeto.
- O **Capítulo 6** apresenta as conclusões gerais e propõe linhas de trabalho futuro para a evolução do sistema.

CAPÍTULO 2

Trabalhos relacionados

2.1 Introdução

A área da manipulação robótica apresenta inúmeros desafios, particularmente em tarefas que envolvem preensão precisa e troca física de objetos, tanto entre robôs como entre robôs e seres humanos. Nestes contextos, a configuração e natureza da extremidade manipuladora desempenham um papel crucial na eficácia da interação. Para alcançar níveis mais elevados de destreza, flexibilidade e adaptabilidade, o uso de mãos robóticas antropomórficas tem-se revelado uma abordagem promissora. No entanto, estas soluções implicam desafios significativos, desde a conceção e fabrico até à complexidade do controlo, devido ao elevado número de graus de liberdade e à redundância cinemática associada.

Paralelamente, a integração de sensores de contacto é essencial para dotar as mãos robóticas de perceção tátil, permitindo-lhes detetar interações físicas com o ambiente ou entre os próprios dedos. Esta capacidade é fundamental para melhorar o controlo em tempo real, garantir uma manipulação mais segura e possibilitar a implementação de estratégias adaptativas baseadas na perceção sensorial.

Este capítulo apresenta uma análise de diferentes mãos robóticas antropomórficas, incluindo soluções comerciais e plataformas *open-source*, com o objetivo de fundamentar a seleção do modelo adotado neste projeto. Adicionalmente, é feita uma revisão das principais tecnologias de sensores de contacto, culminando na escolha dos mais adequados para integração na mão desenvolvida.

2.2 Mãos Robóticas Antropomórficas

O desenvolvimento de mãos robóticas antropomórficas tem sido alvo de grande interesse na comunidade científica e industrial, devido ao seu potencial para executar tarefas complexas de manipulação com maior destreza e versatilidade. Estas mãos procuram reproduzir, de forma funcional, algumas das capacidades da mão humana, beneficiando de um elevado número de graus de liberdade e de uma estrutura mecânica inspirada na anatomia biológica.

Esta secção apresenta e analisa diferentes soluções de mãos robóticas antropomórficas existentes, tanto comerciais como *open-source*, com o objetivo de identificar características relevantes para o projeto em desenvolvimento.

2.2.1 Soluções comerciais

Existem várias mãos robóticas antropomórficas disponíveis comercialmente, desenvolvidas com foco na destreza, robustez e integração em ambientes industriais ou de investigação. Estas soluções oferecem alto desempenho, mas geralmente implicam custos elevados, com valores que podem ultrapassar os 10 000€, o que as torna menos acessíveis para projetos académicos ou de prototipagem. Nesta subsecção são apresentadas algumas das opções disponíveis no mercado, como a Oceantrix Hand e a Agile Hand.

OceanTrix Hand

A OceanTrix Dexterous Hand (Figura 2.1) é uma mão robótica de sétima geração que apresenta uma arquitetura mecânica com 6 graus de liberdade (DOF), permitindo movimentos coordenados e versáteis dos dedos, adequados para tarefas complexas de manipulação.

Figura 2.1: OceanTrix Hand desenvolvida pela OceanTrix Robotics[8].

O sistema de atuação é totalmente elétrico, utilizando motores de alto desempenho encapsulados em componentes metálicos de grau aeronáutico (alumínio AL6061), o que assegura robustez estrutural e resistência a ambientes exigentes.

A mão incorpora 13 sensores táteis distribuídos estrategicamente nos dedos, proporcionando *feedback* sensorial detalhado que permite um controlo em malha fechada e uma interação com objetos de diferentes formas e texturas. No entanto, esta característica é opcional.

Em termos de integração, a OceanTrix Dexterous Hand é compatível com interfaces de comunicação RS485 e CAN, suportando protocolos como Modbus-RTU. A programação e controlo podem ser realizados através de linguagens como Python e C, oferecendo flexibilidade para integração em diversos sistemas robóticos.

A compatibilidade mecânica com manipuladores específicos não é diretamente indicada nas fontes disponíveis. No entanto, a utilização de interfaces mecânicas padrão e a flexibilidade de

comunicação sugerem que a integração com manipuladores existentes é possível, desde que sejam realizadas as adaptações necessárias[8].

Agile Hand

A *Agile Hand* (Figura 2.2), desenvolvida pela Agile Robots SE, é uma mão robótica antropomórfica com 15 a 16 graus de liberdade, distribuídos por 20 articulações, permitindo movimentos complexos e precisos. Cada dedo é modular e atuado por motores elétricos integrados, capazes de gerar até 10 N de força na ponta dos dedos e velocidades de até 360°/s. A mão incorpora sensores de torque e posição em todas as articulações, permitindo controlo em malha fechada com conformidade ativa. É compatível com manipuladores que seguem a norma ISO 9409-1-50-4-M6 e oferece suporte a C++, Python e ROS, com comunicação a 1 kHz por protocolo proprietário[9].

Figura 2.2: Agile Hand desenvolvida pela Agile Robots SE [9].

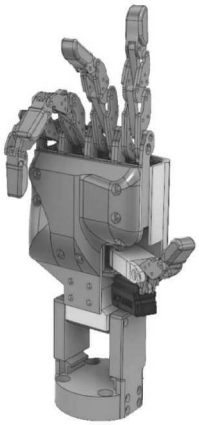
2.2.2 Soluções *open-source*

As mãos robóticas *open-source* surgem como alternativas acessíveis e personalizáveis às soluções comerciais, especialmente em contextos de investigação e desenvolvimento. Entre as suas principais vantagens destacam-se o baixo custo, a flexibilidade de modificação e a comunidade ativa de utilizadores e desenvolvedores, que facilita a partilha de conhecimento e melhorias contínuas. No entanto, estas soluções apresentam também desvantagens, como a menor robustez mecânica, limitações em precisão e desempenho, e falta de suporte técnico especializado, o que pode representar um desafio em aplicações industriais exigentes.

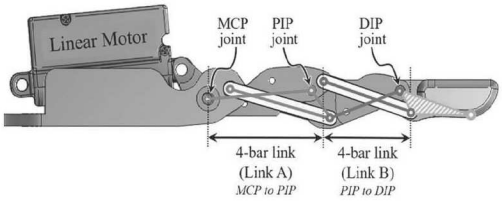
Para este projeto, a escolha de construir uma mão robótica *open-source* revela-se mais vantajosa, uma vez que será utilizada em contexto de investigação e desenvolvimento, onde a possibilidade de realizar modificações estruturais e funcionais é essencial. Para além disso, a natureza aberta do projeto permite uma maior adaptabilidade aos requisitos específicos da aplicação.

HRI Hand

A HRI Hand [10] (Figura 2.3a) é uma mão robótica antropomórfica *open-source* desenvolvida para atuar como *end-effector* em manipuladores robóticos. Esta mão possui 15 graus de liberdade (sendo apenas 6 juntas ativas), distribuídos entre os 5 dedos, com cada dedo controlado por um motor linear dedicado, enquanto o polegar utiliza dois motores adicionais para realizar movimentos de abdução e adução. A mão utiliza um mecanismo subatuado baseado num sistema de duas barras articuladas (*four-bar linkage*, ilustrada na figura 2.3b), permitindo que as articulações Proximal Interphalangeal Joint (PIP) e Dual In-line (DIP) sejam movidas de forma dependente a partir do motor que controla a articulação Metacarpophalangeal Joint (MCP).



(a) HRI Hand desenvolvida por Park, et al. [10].



(b) Sistema de *four-bar linkage* desenvolvida na HRI Hand [10].

Figura 2.3: Mão HRI e seu mecanismo

Em termos de compatibilidade de software, a HRI Hand é totalmente integrada com o Robot Operating System (ROS) e segue a norma ISO 9409-1-50-4-M6, garantindo compatibilidade com diversos manipuladores robóticos, como o UR3.

Robot Nano Hand

A Robot Nano Hand [11] (Figura 2.4) é uma mão robótica antropomórfica *open source*, concebida para replicar o tamanho e os movimentos de uma mão humana adulta, oferecendo um total de 11 graus de liberdade.

A sua estrutura inclui 4 dedos independentes e 1 polegar parcialmente opositor, com cada dedo capaz de realizar movimentos de flexão e laterais, garantindo uma ampla gama de gestos. O sistema de atuação é baseado no uso de tendões, que percorrem os dedos e são controlados por servomotores.

A mão conta também com uma câmara Raspberry Pi V2 integrada na palma, o que a torna adequada para aplicações em visão computacional, incluindo reconhecimento de gestos e análise de objetos. O controlo é gerido por um NVIDIA Jetson Nano Development Kit, um dispositivo robusto que suporta algoritmos de inteligência artificial, como redes neuronais para classificação de imagens e segmentação de objetos. Embora projetada como uma solução

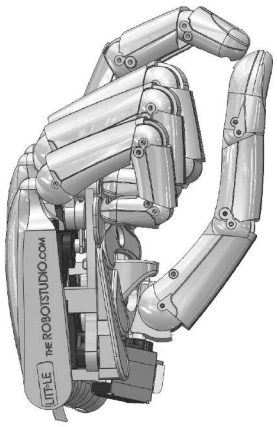


Figura 2.4: Robot Nano Hand desenvolvida por The Robot Studio[11].

autónoma, com os servos incorporados diretamente na estrutura e um servo adicional que fornece rotação no eixo de inclinação para o conjunto da mão, o design modular da Robot Nano Hand permite adaptações que podem facilitar a sua integração com manipuladores robóticos ou outros sistemas robóticos.

Alaris Hand

A Alaris Hand [12] (Figura 2.5) é uma mão robótica antropomórfica *open source*, concebida para reproduzir as funcionalidades e movimentos de uma mão humana. Este dispositivo apresenta um total de 6 graus de liberdade, permitindo movimentos versáteis dos dedos, que incluem flexão, extensão e manipulação de objetos. O sistema de atuação baseia-se em atuadores lineares miniaturizados, acoplados a mecanismos de quatro barras acionados por transmissões de parafuso sem-fim e cremalheira, dispensando o uso de cabos que simulem tendões artificiais. Este método garante um controlo eficaz das articulações e proporciona uma força de preensão robusta. Além disso, a Alaris Hand está equipada com sensores de posição que monitorizam os movimentos articulares em tempo real.

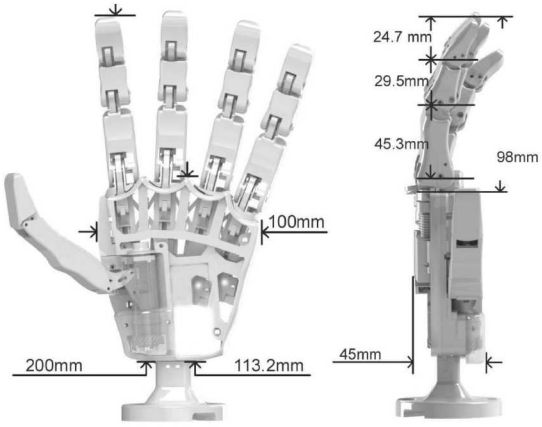


Figura 2.5: Alaris Hand desenvolvida por Nurpeissova, et al.[12]

LEAP Hand

A LEAP Hand [13] (Figura 2.6)é uma mão robótica antropomórfica de baixo custo, comparado com as soluções comerciais, projetada para aplicações com aprendizagem automática. Esta mão possui 4 dedos e um total de 16 graus de liberdade, com um mecanismo inovador de abdução-adução que mantém todos os graus de liberdade disponíveis em diversas posições das articulações, o que contribui para a sua alta agilidade. O método de atuação utiliza motores diretos Dynamixel, o que proporciona resistência a torques elevados e aumenta a durabilidade em tarefas de manipulação prolongada.

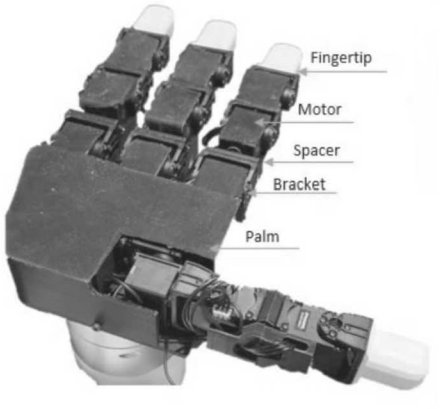


Figura 2.6: Leap Hand desenvolvida por Shaw, et al.[13]

No que diz respeito aos sensores, a LEAP Hand é capaz de integrar sensores externos para teleoperação ou aprendizagem, mas não possui sensores embutidos nativamente. Em termos de software, ela é compatível com diversas plataformas, incluindo ROS, Python e C++, além de oferecer um simulador baseado no Isaac Gym e Pybullet, o que permite a realização de experiências de simulação para o mundo real (*sim2real*).

A LEAP Hand é projetada para ser modular e facilmente reparável, utilizando peças impressas em 3D e componentes disponíveis no mercado. A sua compatibilidade com manipuladores robóticos inclui braços como o UR5, o que amplia as suas aplicações em investigação e desenvolvimento.

2.2.3 Seleção da mão a construir

Após a análise comparativa das diferentes mãos robóticas open-source disponíveis, sumarizada na Tabela 2.1, optou-se pela LEAP Hand [13] como a solução mais adequada para replicação e adaptação no âmbito deste projeto. Esta decisão fundamenta-se principalmente no seu design de construção simples, baseado em servomotores em vez de sistemas de tendões, o que reduz a complexidade mecânica, facilita a montagem e minimiza potenciais pontos de falha.

Apesar da sua simplicidade, a LEAP Hand oferece um elevado número de graus de liberdade (16 DoF), superior à maioria das alternativas analisadas, permitindo uma manipulação

Modelo	DoF	Atuação	Sensores	Compatibilidade com manipulador	Software
HRI Hand	6	Motores Lineares	Nenhum	UR3	ROS
Robot Nano Hand	11	Tendões	Câmera	Não mencionado	Não mencionado
Alaris Hand	6	Tendões	Posição	Não	Não mencionado
LEAP Hand	16	Servomotores	Posição	UR5	ROS / Python / C++ / simulação

Tabela 2.1: Comparação entre diferentes mãos robóticas *open-source*

mais versátil. A presença de sensores de posição integrados garante um controlo eficaz dos movimentos articulares, fator essencial para aplicações em tarefas complexas de manipulação.

Adicionalmente, destaca-se a sua compatibilidade com o sistema ROS, bem como com linguagens de programação como Python e C++, o que proporciona uma plataforma de desenvolvimento acessível e extensível. A possibilidade de integração com manipuladores como o UR5 reforça ainda mais a sua aplicabilidade prática. Por fim, o seu design modular e totalmente imprimível em 3D permite uma manutenção económica e favorece alterações e melhorias iterativas durante o desenvolvimento.

2.3 Sensores de Contacto

A percepção tátil desempenha um papel vital ao permitir que mãos robóticas interajam com o ambiente de forma adaptativa, imitando as capacidades da mão humana. Esta tecnologia é essencial em áreas como a robótica industrial e a aprendizagem por reforço, que se beneficia do *feedback* do mundo real para melhorar o desempenho robótico.

Ao longo dos anos foram propostas diversas tecnologias para sensores táteis, sendo as mais relevantes, no contexto deste trabalho, os sensores baseados em tecnologia magnética, polímeros condutivos e materiais piezoresistivos. Esta secção abordará as principais características e aplicações destas tecnologias em mãos robóticas, destacando as suas vantagens e limitações.

Apesar dos avanços, apenas um número limitado dessas tecnologias tem sido implementado com sucesso em plataformas robóticas reais. Algumas empresas já produzem sensores táteis para aplicações robóticas, como o sensor de ponta de dedo BioTac da SynTouch (ilustrado na figura 2.7) [14]. No entanto, estes sensores continuam a apresentar custos elevados e, frequentemente, requerem a integração com outras tecnologias para otimização do desempenho.

Figura 2.7: Sensor Biotac desenvolvido pela SynTouch [14].

2.3.1 Sensores baseados em Tecnologia Magnética

Os sensores baseados em tecnologia magnética têm ganho bastante destaque na área da robótica tátil devido à sua elevada precisão, ausência de desgaste e boa durabilidade. Estes sensores destacam-se também por permitirem a deteção de forças através da variação de campos magnéticos, o que reduz a necessidade de contacto mecânico direto com os elementos sensíveis, tornando-os particularmente adequados para aplicações em ambientes adversos.

Filmes magnéticos flexíveis, como os apresentados por Jamone et al. [15] (esquema de funcionamento do sensor presente na figura 2.8) e Yan et al.[16], consistem numa camada de elastómero incorporando partículas magnéticas e sensores de efeito Hall colocados abaixo dessa camada. Esta configuração permite medir forças normais e de cisalhamento com elevada sensibilidade, sendo particularmente útil para aplicações em superfícies curvas. No entanto, este tipo de sensor tende a saturar em forças superiores a 4 N, o que limita a sua utilização em tarefas que envolvem forças mais intensas.

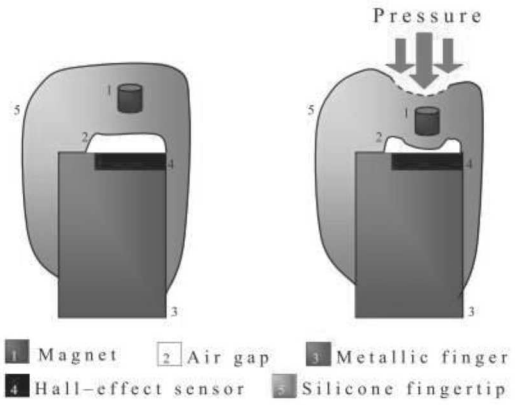


Figura 2.8: Sensor com filme magnético flexível desenvolvido por Jamone, et al. [15].

Outra abordagem promissora é o uso de *magneto-elastomer composites*, que combinam materiais elásticos com partículas magnéticas dispersas de forma controlada. Estes sensores,

como os utilizados nos trabalhos de Kawasetsu et al. (Figura 2.9)[17], demonstram elevada capacidade de deteção mesmo em situações de contacto rápido e com elevada aceleração, para além de apresentarem uma resposta estável a forças contínuas. Contudo, o processo de fabrico pode ser complexo, o que pode implicar maiores custos e limitações em termos de flexibilidade geométrica.

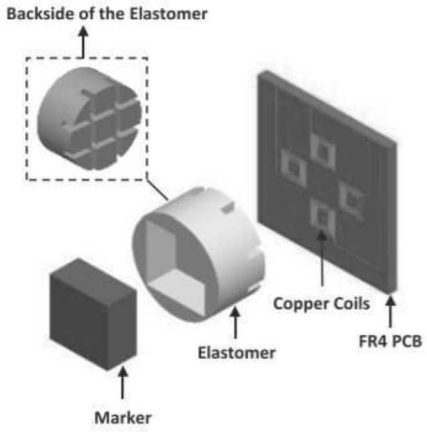


Figura 2.9: Composição do sensor desenvolvido por Kawasetsu et al. [17].

Os sensores magnetoresistivos representam uma alternativa particularmente robusta, sendo eficazes mesmo em condições adversas como altas temperaturas, humidade ou ambientes com poeiras. O sensor desenvolvido por Alfadhel et al. (cujo princípio de operação se encontra visível na Figura 2.10)[18] utiliza uma estrutura em espiral com materiais magnetoresistivos sensíveis ao campo magnético gerado por uma fonte próxima. Embora a sua robustez seja uma vantagem, estes sensores exigem elevada precisão na montagem e estabilidade dos campos magnéticos externos, o que pode complicar a sua integração em sistemas compactos e móveis [19].

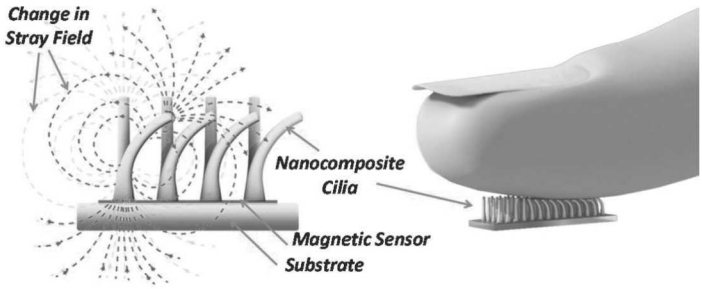


Figura 2.10: Princípio de operação do sensor baseado em materiais magnetoresistivos [18].

Apesar dos desafios, os sensores magnéticos continuam a evoluir e a mostrar grande potencial, sobretudo em aplicações onde a precisão e a resistência a condições ambientais extremas são fundamentais.

2.3.2 Sensores baseados em Polímeros Condutivos

Os sensores baseados em polímeros condutivos têm-se destacado como soluções promissoras para aplicações táteis em sistemas robóticos, sobretudo pela sua leveza, flexibilidade e facilidade de adaptação a superfícies irregulares. Estes dispositivos exploram a variação das propriedades elétricas de materiais condutivos em resposta a deformações mecânicas, permitindo a detecção eficiente de toque e pressão.

Os sensores piezoelétricos utilizam materiais como o PVDF, capazes de gerar sinais elétricos sob estímulos dinâmicos. São valorizados pela elevada sensibilidade, baixo peso e flexibilidade [20], [21], sendo eficazes na deteção de vibrações e forças variáveis. No entanto, não são apropriados para medições de forças estáticas e requerem circuitos de condicionamento de sinal mais complexos. Na figura 2.11 é possível observar a estrutura de um sensor piezoelétrico.

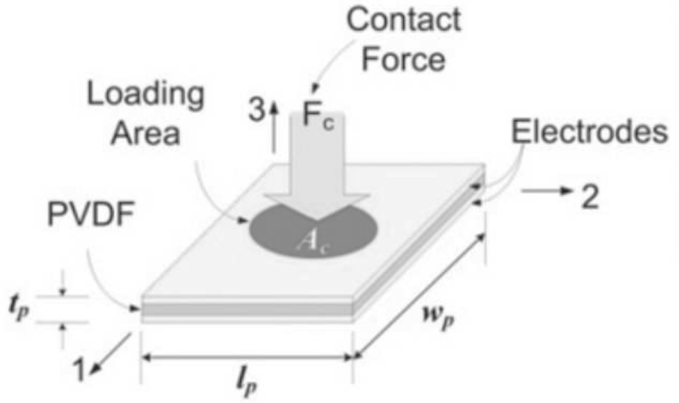


Figura 2.11: Estrutura de um sensor baseado em PVDF desenvolvido por Seminara et al.[21]

Adicionalmente, os sensores triboelétricos operam com base na eletrificação por contato, sendo capazes de gerar sinais elétricos de forma autossuficiente, sem necessidade de alimentação externa. Trabalhos recentes [22], [23] demonstram o seu bom desempenho em ambientes adversos. Ainda assim, enfrentam desafios associados à fragilidade mecânica, complexidade de fabrico e sensibilidade a fatores ambientais.

2.3.3 Sensores baseados em Materiais Piezoresistivos

Os sensores baseados em materiais piezoresistivos têm sido amplamente utilizados em aplicações de sensoriamento tátil, especialmente no domínio da robótica, devido à sua elevada sensibilidade à aplicação de forças e pressões. O seu funcionamento baseia-se na variação da resistência eléctrica resultante de deformações mecânicas, permitindo medições precisas e localizadas.

Um exemplo representativo desta tecnologia são os sensores de resistência sensível à força (FSRs, representados na Figura 2.12), que combinam um design compacto e robusto com uma elevada adaptabilidade a superfícies curvas e complexas, como as encontradas em mãos robóticas [24], [25]. Estas características tornam-nos particularmente adequados para sistemas de deteção tátil de baixo custo e fácil integração.

Figura 2.12: Sensores FSR desenvolvidos pela Interlink [26].

Entre as principais vantagens dos sensores piezoresistivos destacam-se o custo reduzido de produção, o peso reduzido, a flexibilidade e a possibilidade de personalização da sensibilidade em função dos requisitos da aplicação [27]. Para além disso, a simplicidade do seu princípio de funcionamento facilita a integração com circuitos eletrónicos de leitura e controlo.

No entanto, estes sensores apresentam também algumas limitações. A sua precisão é, por vezes, inferior à de outras tecnologias sensoriais, e a resposta não linear pode dificultar a calibração. Adicionalmente, a durabilidade e a estabilidade da resposta ao longo do tempo podem ser comprometidas, especialmente em aplicações sujeitas a esforços repetidos ou prolongados [24].

2.3.4 Escolha dos sensores a incorporar no projeto

A seleção dos sensores táteis para o presente projeto foi fundamentada numa análise comparativa entre as diferentes tecnologias disponíveis, com base em critérios como princípio de funcionamento, vantagens técnicas e limitações práticas. A Tabela 2.2 apresenta uma síntese dos principais tipos de sensores considerados.

Sensor	Princípio	Vantagens	Desvantagens
Magnético	Variação de campo magnético	Sem desgaste, durável	Custo elevado, interferência magnética, complexidade
Polímero Condutivo	Variação da condutividade com deformação	Sensível, flexível, leve	Menor durabilidade, sensível ao ambiente, resposta lenta
Piezoresistivo (FSR)	Redução da resistência com pressão	Baixo custo, fácil integração, compacto	Não linear, vida útil limitada

Tabela 2.2: Comparação entre tipos de sensores táteis

Com base na análise comparativa realizada, os sensores piezoresistivos do tipo FSR foram selecionados como a solução mais adequada para o projeto em desenvolvimento. Esta decisão

fundamenta-se num conjunto de fatores que favorecem a sua aplicação em contextos de robótica tátil, nomeadamente em mãos robóticas.

Em primeiro lugar, destaca-se a sua estrutura fina, leve e flexível, que permite uma integração eficiente em superfícies curvas, como as pontas dos dedos e a palma da mão robótica. Para além disso, os FSRs oferecem um equilíbrio vantajoso entre desempenho e custo.

Adicionalmente, o seu design compacto e a simplicidade de integração com os sistemas eletrónicos permitem reduzir significativamente a complexidade de implementação, fator particularmente relevante em projetos com restrições de espaço físico, como é o caso da LEAP Hand [13].

2.4 Conclusão

Ao longo deste capítulo foram analisadas várias opções tecnológicas com vista à definição da arquitetura sensorial do projeto. Numa primeira fase, avaliou-se um conjunto de mãos robóticas antropomórficas, tanto comerciais como open-source, com o objetivo de selecionar uma solução adequada aos requisitos de integração tátil. Desta análise resultou a escolha da LEAP Hand[13], cuja estrutura modular, dimensões compactas e compatibilidade com sensores externos a tornam particularmente apropriada para o presente projeto.

De seguida, procedeu-se ao estudo comparativo de diferentes tecnologias sensoriais, nomeadamente sensores baseados em princípios magnéticos, polímeros condutivos e materiais piezoresistivos. A avaliação teve em consideração fatores como sensibilidade, facilidade de integração e custo.

Com base nesta análise, foram selecionados sensores piezoresistivos do tipo FSR, devido ao seu formato compacto, baixo custo, facilidade de aplicação em superfícies curvas e capacidade de resposta adequada aos requisitos de deteção tátil em mãos robóticas.

Em suma, as decisões tomadas relativamente à mão robótica e à tecnologia sensorial asseguram uma base sólida para o desenvolvimento do sistema de perceção tátil, alinhando-se com os objetivos técnicos e funcionais definidos para o projeto.

CAPÍTULO 3

Ferramentas de suporte

3.1 Introdução

A adoção de uma mão robótica *open-source* apresenta diversas vantagens, entre as quais se destacam a disponibilização de código de apoio e a existência de decisões previamente fundamentadas relativamente à seleção de componentes mecânicos e eletrónicos. Estas características permitem reduzir significativamente o tempo necessário para o desenvolvimento de um sistema funcional, evitando o esforço associado ao desenho e especificação de soluções desde a parte inicial do projeto.

Neste capítulo descrevem-se os principais elementos já existentes e utilizados como base no presente trabalho, com particular destaque na LEAP Hand, desenvolvida por Shaw et al. [13]. Serão abordadas as escolhas efetuadas por estes autores ao nível dos motores, do controlador de motores, da fonte de alimentação e das ferramentas de software disponibilizadas, que serviram de ponto de partida para o desenvolvimento realizado neste projeto.

3.2 Hardware

A construção de um sistema robótico funcional requer a seleção criteriosa dos componentes físicos que garantem a sua operacionalidade e fiabilidade. Assim, nesta secção procede-se à descrição dos principais elementos de hardware adotados, com base na arquitetura da LEAP Hand. Serão abordadas as características dos atuadores utilizados, o sistema de controlo associado, a fonte de alimentação dimensionada para o conjunto de motores, bem como a integração com o braço robótico UR10, que desempenha um papel fundamental na realização de ensaios experimentais.

3.2.1 Motores Dynamixel

No desenvolvimento da LEAP Hand, Shaw et al. [13] optaram por utilizar os motores Dynamixel XC330-M288-T, amplamente utilizados em aplicações robóticas devido à sua fiabilidade, eficiência energética e dimensões compactas. Estes atuadores oferecem controlo

detalhado sobre posição, velocidade e corrente, permitindo gerar movimentos suaves, reproduzíveis e bem definidos — uma característica fundamental em sistemas de manipulação, como mãos robóticas antropomórficas.

Os motores Dynamixel XC330-M288-T (Figura 3.1) operam a 5V e apresentam uma corrente nominal de 1.8A, o que os torna compatíveis com fontes de alimentação de baixa potência, mantendo um desempenho consistente. Adicionalmente, a comunicação digital via protocolo TTL permite a ligação e controlo eficiente de múltiplos motores em rede, facilitando a sua integração com microcontroladores e promovendo uma arquitetura modular e escalável.

Figura 3.1: Motor Dynamixel XC330-M288-T utilizado na construção da LEAP Hand[13]

O controlo destes motores pode ser realizado por diferentes meios, nomeadamente através da ferramenta gráfica Dynamixel Wizard 2.0, útil para configuração e diagnóstico, e da biblioteca Dynamixel SDK, que permite o controlo programático e a integração com sistemas desenvolvidos em diversas linguagens de programação. Estas opções tornam o sistema versátil e facilmente adaptável a diferentes contextos de desenvolvimento, incluindo ambientes compatíveis com ROS.

Os motores Dynamixel armazenam e gerem os seus parâmetros internos através de uma tabela de controlo de memória, onde cada parâmetro — como posição, velocidade, corrente ou temperatura — está associado a um endereço específico. Essa tabela encontra-se dividida em duas secções principais: a **EEPROM**, que contém parâmetros permanentes (como o ID do motor, taxa de transmissão, limites de corrente ou posição) e só pode ser modificada após a reinicialização do motor; e a **RAM**, onde se encontram os parâmetros dinâmicos, como a posição atual, velocidade, temperatura e estado do motor, permitindo atualizações em tempo real durante a operação. Esta estrutura torna o controlo dos motores altamente flexível e eficiente, uma vez que permite o acesso direto e segmentado aos dados necessários para a gestão do comportamento do sistema robótico.

3.2.2 Controlador e Fonte de Alimentação

Para além da seleção dos motores, a escolha do controlador de comunicação e da fonte de alimentação tem um impacto significativo no desempenho e na fiabilidade do sistema. No que respeita à interface de comunicação com os motores, os autores da LEAP Hand [13] optaram

por utilizar o **Dynamixel U2D2** (Figura 3.2), um conversor USB para TTL *half-duplex*. Este dispositivo permite estabelecer comunicação entre um computador e múltiplos motores Dynamixel através de uma única porta serial, utilizando um barramento em cadeia (*daisy chain*). Este método de ligação, possível graças à arquitetura *half-duplex* dos motores, permite que vários atuadores sejam ligados em paralelo, reduzindo significativamente a complexidade do sistema e facilitando a sincronização entre motores. O U2D2 suporta taxas de transmissão elevadas (até 4,5 Mbps), possui um conector JST de 3 pinos para comunicação TTL, e inclui um terminal de alimentação auxiliar (VDD), permitindo a alimentação dos motores a partir de uma fonte externa apropriada.

Relativamente à fonte de alimentação, foi necessário considerar que cada motor Dynamixel XC330-M288-T pode atingir um consumo máximo de 1.8A em condições de carga. Tendo em conta a utilização de 16 motores, a corrente total exigida pode atingir os 28.8A. Assim, foi selecionada uma fonte de alimentação industrial (Figura 3.3) com saída estabilizada de 5V — a mesma tensão nominal dos motores — e capacidade de fornecimento de até 30A, assegurando uma margem de segurança adequada para o funcionamento simultâneo de todos os atuadores.

Figura 3.2: Controlador Dynamixel U2D2 utilizado para controlar os motores do projeto

Figura 3.3: Fonte de Alimentação industrial com saída a 5V e capacidade de 30A utilizada neste projeto

3.2.3 UR10

O braço robótico UR10 (Figura 3.4), desenvolvido pela Universal Robots, é um manipulador colaborativo de seis graus de liberdade amplamente utilizado em aplicações industriais e de investigação. A sua versatilidade e facilidade de integração com outros sistemas tornam-no uma ferramenta valiosa para apoiar o desenvolvimento e a validação de tecnologias robóticas.

No contexto deste projeto, o UR10 será utilizado como estrutura de suporte para a Leap Hand, beneficiando de uma peça de adaptação já existente no modelo da mão que permite a sua montagem direta no flange do braço robótico. Esta integração permite fixar a mão de

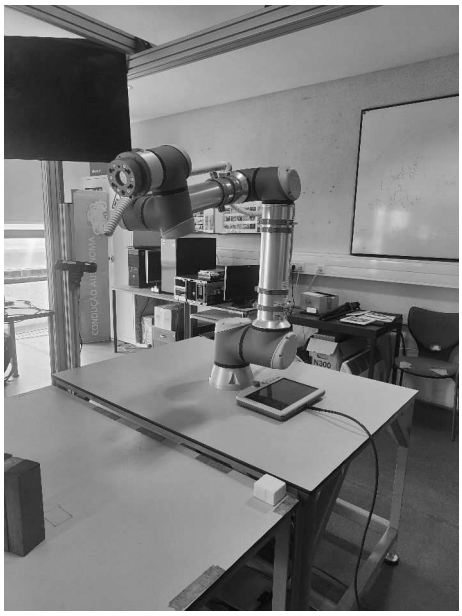


Figura 3.4: Braço robótico UR10 utilizado no projeto

forma segura e estável, garantindo uma base robusta para a realização de testes e recolha de dados experimentais.

A principal motivação para a utilização do UR10 está relacionada com o processo de aquisição de dados necessário ao treino de um modelo de classificação de sobreposições de dedos. Com o auxílio do braço, é possível posicionar a mão robótica em configurações específicas de forma repetível e controlada, bem como simular interações físicas com o ambiente.

Além disso, a utilização do UR10 permite explorar movimentos mais complexos e controlados da mão em diferentes orientações espaciais, o que facilita a geração de um conjunto de dados mais diversificado e representativo de cenários reais de sobreposição entre dedos. Esta abordagem visa melhorar a generalização do modelo e a sua robustez perante variações no posicionamento.

3.3 Software

Para além da infraestrutura física, o desenvolvimento de sistemas robóticos exige o suporte de um conjunto abrangente de ferramentas de software que possibilitem a configuração, controlo e monitorização dos componentes de hardware. Esta secção descreve as soluções de software utilizadas ao longo do projeto, incluindo aplicações para configuração dos motores, bibliotecas de comunicação, frameworks de desenvolvimento modular, bem como o ambiente de programação adotado para o microcontrolador. A seleção destas ferramentas teve como objetivo garantir a flexibilidade, escalabilidade e compatibilidade com os requisitos específicos do sistema implementado.

3.3.1 Dynamixel Wizard 2.0

A Dynamixel disponibiliza diversas ferramentas para configuração, monitorização e controlo dos seus motores, entre as quais se destaca o Dynamixel Wizard 2.0, uma interface gráfica

intuitiva (Figura 3.5) que permite ao utilizador interagir diretamente com cada motor. Antes de iniciar o controlo coordenado de um sistema com múltiplos atuadores, é essencial garantir que cada motor possui um identificador único. Por defeito, todos os motores Dynamixel são fornecidos com o mesmo identificador (ID = 1), o que inviabiliza a sua utilização simultânea num mesmo barramento de comunicação. Assim, uma das primeiras tarefas consiste na atribuição de um ID exclusivo a cada motor, procedimento que pode ser facilmente realizado através do Dynamixel Wizard 2.0, modificando o parâmetro correspondente no firmware do atuador.

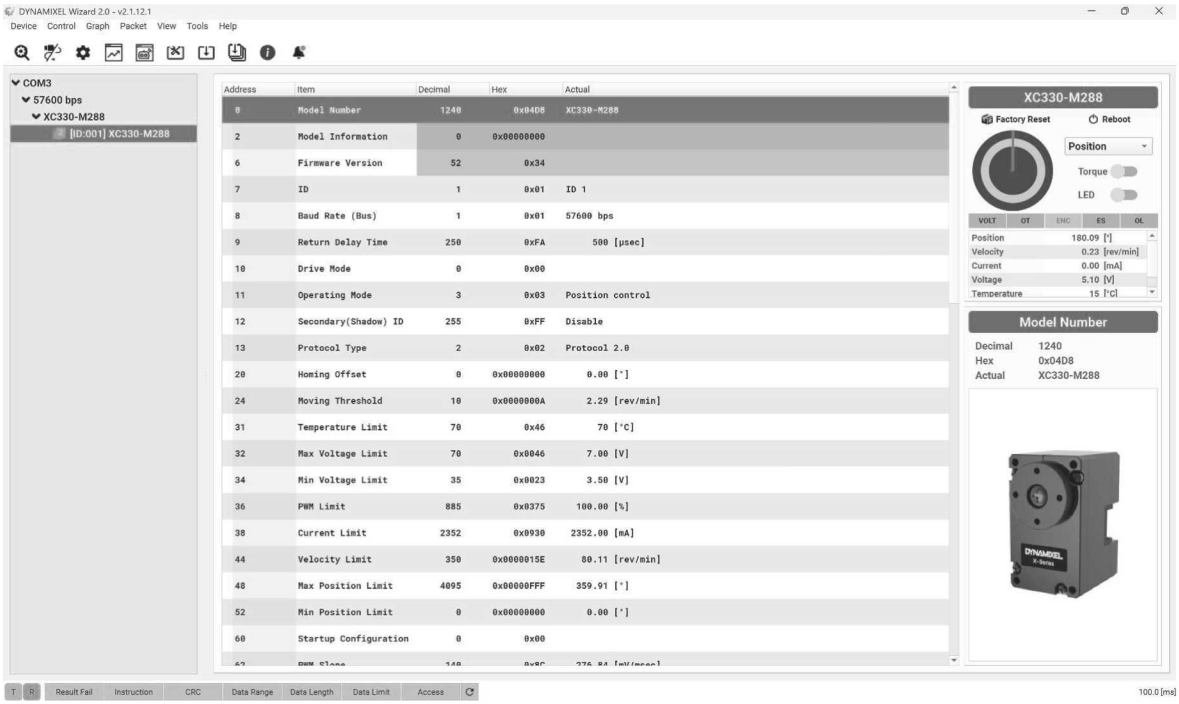


Figura 3.5: Interface Gráfica da ferramenta Dynamixel Wizard 2.0 utilizada para alterar o ID de cada motor

Para além da atribuição de IDs individuais, esta ferramenta permite ainda a definição de grupos de motores com IDs secundários, úteis para cenários em que se pretende acionar vários motores em simultâneo e de forma idêntica. No entanto, esta funcionalidade não será explorada neste projeto dado que o controlo será efetuado de forma independente para cada atuador.

Adicionalmente, o Dynamixel Wizard 2.0 oferece funcionalidades de diagnóstico e monitorização em tempo real, possibilitando a visualização de parâmetros como a posição atual, velocidade, corrente consumida, entre outros. Esta capacidade é particularmente útil durante as fases de teste, permitindo validar o comportamento dos motores e detetar eventuais anomalias no sistema.

3.3.2 Biblioteca Dynamixel SDK

O Dynamixel SDK é uma biblioteca de software multiplataforma que permite a comunicação e o controlo eficiente dos motores Dynamixel em diversas linguagens de programação, tais

como Python, C, C++ e Java. Concebido para proporcionar uma interface unificada e de alto nível, o SDK abstrai as complexidades da comunicação série, simplificando as operações de leitura e escrita de dados nos motores.

Esta biblioteca suporta os principais protocolos de comunicação Dynamixel, incluindo o Protocolo 2.0, utilizado no presente projeto para os motores XC330-M288-T, o que assegura compatibilidade com as funcionalidades mais recentes dos atuadores, como o controlo de corrente, modos operacionais configuráveis e diagnósticos internos. Através do SDK, é possível configurar parâmetros como posição, velocidade, corrente máxima, limites de movimento e modo de operação, permitindo adaptar o comportamento dos motores às exigências específicas de cada aplicação.

3.3.3 ROS2

O ROS é uma *framework* amplamente utilizada no desenvolvimento de sistemas robóticos, proporcionando um ambiente modular que facilita o desenvolvimento, gestão e interligação de múltiplos nós que executam tarefas específicas em paralelo. Esta arquitetura distribuída permite uma organização estruturada do código, promovendo a escalabilidade e a reutilização de componentes, o que é particularmente vantajoso em projetos complexos, como o desenvolvimento de mãos robóticas antropomórficas.

No âmbito da LEAP Hand, Shaw et al. [13] disponibilizaram documentação e exemplos compatíveis com Python, ROS 1 e ROS 2, conferindo flexibilidade na escolha da infraestrutura de desenvolvimento. Neste projeto, optou-se pela utilização do ROS 2, em detrimento do ROS 1, com base nas vantagens técnicas e estruturais que a nova versão oferece.

Entre os principais fatores que justificam esta escolha destacam-se a utilização do *middleware Data Distribution Service (DDS)*, que melhora substancialmente a comunicação entre nós, tornando-a mais eficiente e robusta — um requisito fundamental em sistemas com múltiplos atuadores e sensores a operar em tempo real. Para além disso, o ROS 2 elimina a necessidade de um nó central como o *roscore*, permitindo uma arquitetura verdadeiramente distribuída e mais resiliente a falhas. Esta mudança traduz-se numa maior robustez e flexibilidade na gestão da rede de nós, especialmente em contextos com comunicações complexas ou heterogéneas.

Outro fator relevante é o fim gradual do suporte ao ROS 1, cuja manutenção oficial está em fase de descontinuação, sendo que muitas das novas bibliotecas e ferramentas estão a ser exclusivamente desenvolvidas para o ROS 2. A adoção do ROS 2 assegura, portanto, maior longevidade, compatibilidade com futuras atualizações e acesso a funcionalidades mais recentes da comunidade de robótica.

Por fim, o suporte nativo a sistemas real-time, a maior portabilidade entre plataformas e o foco na segurança e escalabilidade reforçam a decisão de utilizar o ROS 2 neste projeto. Esta escolha visa garantir um desenvolvimento mais moderno e sustentável, mantendo a compatibilidade com sistemas robóticos avançados e preparados para integração em ambientes distribuídos.

3.3.4 Documentação da LEAP Hand

Os autores da LEAP Hand [13] disponibilizam documentação detalhada e exemplos de código para facilitar a reprodução e o desenvolvimento com base na sua plataforma. Embora essa documentação suporte a utilização em Python, ROS 1 e ROS 2, o código fornecido foi desenvolvido com um foco específico: a execução de políticas previamente treinadas em simulação (utilizando o simulador incluído no projeto) e a realização de movimentos predefinidos da mão robótica.

No entanto, esta abordagem não contempla a possibilidade de controlo direto e individualizado dos motores, nem permite uma personalização completa do comportamento da mão, o que limita a sua aplicabilidade em contextos que exigem maior flexibilidade no desenvolvimento de novos controladores ou algoritmos de manipulação.

Tendo em conta estas limitações, e considerando os objetivos deste projeto, optou-se por não utilizar diretamente o código disponibilizado pelos autores da LEAP Hand, desenvolvendo-se, em alternativa, uma nova infraestrutura de controlo que permite acesso completo e individualizado a cada motor, facilitando o desenvolvimento de estratégias de controlo personalizadas e adaptadas às necessidades específicas deste trabalho.

3.3.5 Arduino IDE

O *Arduino Integrated Development Environment* (Arduino IDE) é uma plataforma de desenvolvimento amplamente utilizada para programação de microcontroladores. Inicialmente concebida para os dispositivos da família Arduino, esta ferramenta tem vindo a expandir o seu suporte, sendo atualmente compatível com uma grande variedade de placas e microcontroladores de diferentes fabricantes.

A interface do Arduino IDE destaca-se pela sua simplicidade e acessibilidade, tornando possível escrever, compilar e carregar código para o microcontrolador com apenas alguns passos. A linguagem de programação utilizada baseia-se em C/C++, e a plataforma disponibiliza uma extensa biblioteca de funções que facilita o acesso a funcionalidades como leitura de sensores, controlo de atuadores ou comunicação com outros dispositivos.

Neste projeto, o Arduino IDE foi utilizado como ambiente principal de desenvolvimento para a programação do microcontrolador selecionado, permitindo implementar o sistema de aquisição de dados e a leitura dos sensores de contacto.

3.4 Conclusão

Ao longo deste capítulo foram apresentadas as principais ferramentas de suporte utilizadas no desenvolvimento do projeto, com especial destaque nos componentes de hardware e software que viabilizam a implementação da mão robótica.

Inicialmente identificaram-se os motores Dynamixel XC330-M288-T e o controlador U2D2 como soluções adequadas para assegurar movimentos compatíveis com a estrutura da LEAP Hand. A fonte de alimentação foi selecionada com base nas exigências energéticas do sistema, garantindo estabilidade e segurança durante a operação. Adicionalmente, foi integrado o braço

robótico UR10 como estrutura auxiliar para a recolha de dados experimentais, permitindo a realização de testes de forma controlada e repetível.

Paralelamente, foram descritas as ferramentas de software de apoio à configuração e controlo dos motores, nomeadamente o Dynamixel Wizard 2.0 e a biblioteca Dynamixel SDK, bem como o ambiente de desenvolvimento baseado em ROS 2, que oferece escalabilidade e integração com outros módulos do sistema. Para a programação do microcontrolador selecionado, recorreu-se à plataforma Arduino IDE, que se destaca pela compatibilidade com diferentes dispositivos e pela sua facilidade de utilização.

Adicionalmente, a documentação da LEAP Hand foi analisada como referência complementar, optando-se por não utilizar o código disponibilizado de forma a permitir o controlo individualizado de cada motor.

Em suma, a seleção e integração das ferramentas de suporte descritas neste capítulo estabeleceram uma infraestrutura adaptada às necessidades do sistema, contribuindo para a robustez, modularidade e evolução futura do desenvolvimento da mão robótica.

CAPÍTULO 4

Projeto e Implementação da Solução

4.1 Introdução

Este capítulo apresenta o processo de desenvolvimento da solução proposta, desde a construção física da mão robótica até à implementação do sistema sensorial. Inicia-se com a apresentação das etapas associadas ao fabrico e montagem da estrutura da mão robótica, seguindo-se a configuração dos motores e o desenvolvimento da lógica de controlo, desde o acionamento de motores individuais até ao controlo coordenado da mão completa.

É igualmente descrita a arquitetura de software desenvolvida, com especial destaque para as funcionalidades avançadas implementadas, como o ajuste dinâmico de corrente durante o contacto com objetos, a definição de velocidades relativas entre dedos e falanges, e a introdução de atrasos temporais entre movimentos.

Adicionalmente, aborda-se a implementação do sistema sensorial de deteção de contacto, incluindo a seleção do microcontrolador, o desenvolvimento físico da unidade de aquisição e os testes realizados com diferentes sensores.

4.2 Desenvolvimento da Mão Robótica

4.2.1 Fabrico e Montagem da Estrutura

A construção da estrutura da mão robótica seguiu integralmente as indicações fornecidas na documentação da LEAP Hand [13], a qual disponibiliza de forma aberta e acessível todos os ficheiros CAD necessários, bem como instruções detalhadas de montagem. Esta abordagem permitiu realizar uma primeira montagem da estrutura de forma fiel ao projeto original, facilitando a replicação inicial da arquitetura mecânica.

As peças foram produzidas por impressão 3D com recurso a uma Creality K1, uma impressora de alto desempenho que assegura uma boa qualidade dimensional e tempos de fabrico reduzidos. Para a estrutura principal foi utilizado o filamento Hyper PLA da Creality, selecionado pela sua rigidez e facilidade de impressão. Já para as pontas dos dedos, onde se

pretende maior flexibilidade e capacidade de aderência ao contacto com objetos, foi utilizado o Python Flex TPU da FormFutura.

A construção dos dedos revelou-se a fase mais demorada do processo de montagem devido ao elevado número de componentes envolvidos e à complexidade dos passos necessários. Todos os dedos seguem a mesma configuração estrutural, com exceção do polegar que apresenta um desenho distinto adaptado à sua função anatómica e posição na mão.

Na Figura 4.1 é apresentado o polegar já montado, enquanto na Figura 4.2 observa-se um dos restantes dedos, com estrutura idêntica entre si. A montagem de cada dedo envolveu operações detalhadas e sequenciais que exigiram precisão e atenção redobrada, especialmente na organização dos cabos e no alinhamento das juntas móveis.

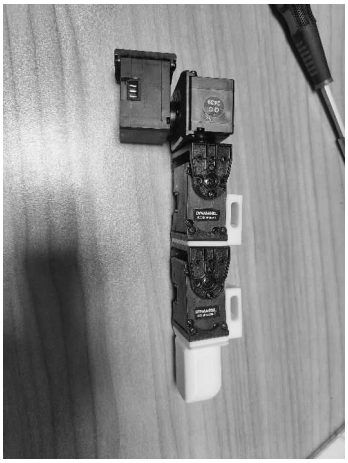


Figura 4.1: Polegar construído de acordo com a Documentação da Leap Hand

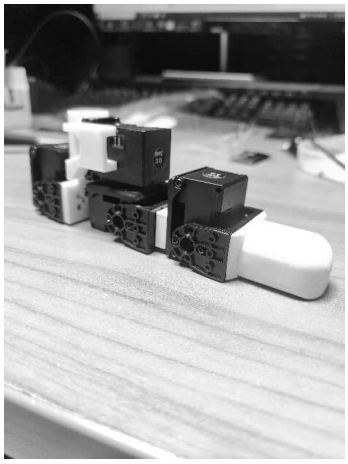


Figura 4.2: Dedo construído de acordo com a documentação da Leap Hand correspondente à estrutura do dedo indicador, médio e anelar

No final do processo obteve-se a mão totalmente montada de acordo com a estrutura proposta na LEAP Hand, conforme ilustrado na Figura 4.3.

Para complementar a descrição deste processo, no Apêndice A.1.1 e A.1.2 e são apresentadas imagens sequenciais representativas da construção de cada dedo, bem como hiperligações (A.1.3) para vídeos que documentam o procedimento completo de montagem, constituindo assim um apoio visual adicional à documentação escrita.

Embora a montagem inicial tenha seguido rigorosamente as orientações da documentação da LEAP Hand, durante o desenvolvimento do projeto surgiram necessidades específicas que motivaram adaptações à estrutura original. Estas modificações serão detalhadas nas secções seguintes deste capítulo.

Figura 4.3: Mão robótica completamente montada, seguindo a estrutura proposta na documentação da LEAP Hand.

4.2.2 Configuração dos Motores

Antes de iniciar o desenvolvimento do código responsável pelo controlo dos movimentos da mão robótica, foi necessário proceder à configuração inicial dos motores Dynamixel. Esta fase incluiu a atribuição de identificadores únicos (IDs) a cada motor, uma vez que todos os motores, por defeito, vêm configurados com o mesmo ID. Para isso, recorreu-se à ferramenta gráfica Dynamixel Wizard 2.0 que permite, de forma intuitiva, alterar os parâmetros fundamentais de cada atuador.

Nesta configuração inicial os IDs atribuídos seguiram a mesma lógica utilizada na documentação da LEAP Hand, garantindo assim compatibilidade com a estrutura de referência proposta pelos autores.

Modos de Operação do Motores Dynamixel

Após a atribuição dos IDs a cada motor, foi necessário configurar o modo de operação correspondente. Os motores Dynamixel XC330-M288-T suportam diversos modos de funcionamento, permitindo adaptá-los a diferentes tipos de movimentos e estratégias de controlo.

Na presente subsecção são descritos os principais modos de operação disponibilizados por estes atuadores, destacando as suas características e o comportamento em situações com ou sem contacto com obstáculos.

1. **Position Control Mode**
   Este modo permite definir uma posição angular específica que o motor deve atingir, dentro dos limites físicos configurados. Caso surja um obstáculo durante o movimento, o motor continuará a aplicar torque máximo na tentativa de alcançar a posição desejada, o que pode levar ao sobreaquecimento dos motores.

2. **Extended Position Control Mode**
   Uma extensão do modo anterior permite rotações superiores a 360°, tornando-se útil para aplicações que requerem movimento contínuo em rotação. No entanto, não é relevante no contexto deste projeto, dado que as juntas dos dedos não realizam rotações completas.
3. **Velocity Control Mode**
   Este modo permite definir uma velocidade angular constante, sem especificar uma posição final. O motor acelera até atingir a velocidade pretendida e mantém-se em rotação. Na presença de um obstáculo, o motor continua a tentar girar, aplicando o torque máximo, o que pode resultar em sobreaquecimento.
4. **PWM Control Mode**
   Permite controlar diretamente a potência fornecida ao motor através de modulação por largura de pulso (PWM). Este modo oferece um controlo mais direto e flexível, mas não garante precisão de movimento, nem proteção contra obstáculos, podendo causar sobreaquecimento em situações de bloqueio mecânico.
5. **Current Control Mode**
   Neste modo controla-se diretamente o torque aplicado ao definir a corrente desejada. Se não houver resistência, o motor continua a girar. Na presença de um obstáculo, o motor não tenta forçar o movimento, e mantém um torque constante correspondente ao valor de corrente definido, oferecendo uma resposta segura e previsível.
6. **Current-Based Position Control Mode**
   Combina as vantagens do controlo de posição com limitação de corrente. O motor tenta alcançar uma determinada posição, mas limita o torque máximo aplicado, de acordo com o valor de corrente permitido. Se o obstáculo exigir mais torque do que o configurado, o motor interrompe o movimento, mantendo a força aplicada constante.

A Tabela 4.1 resume os diferentes modos de operação, destacando os parâmetros controlados, as aplicações ideais e o comportamento em situações de contacto com obstáculos:

A escolha do modo de operação adequado é essencial para garantir o desempenho e a segurança do sistema. No contexto deste projeto, onde o contacto com objetos e a necessidade de limitar a força aplicada pelos dedos são aspetos fundamentais, foi escolhido o *Current-Based Position Control Mode*, uma vez que este modo permite controlar a posição de cada articulação, ao mesmo tempo que impõe um limite ao torque aplicado, o que evita danos em situações de bloqueio ou contacto inesperado.

Modo	Controla	Reação a Obstáculos
Position Control	Posição	Aplica torque máximo até atingir posição
Extended Position	Posição (>360º)	Idêntico ao Position Control
Velocity Control	Velocidade	Aplica torque máximo para manter velocidade
Current Control	Corrente (torque)	Mantém torque constante, sem tentar forçar
Current-Based Position	Posição + Corrente	Tenta atingir posição, mas respeita limite de torque
PWM Control	Potência (PWM)	Não protege contra sobrecarga ou obstáculos

Tabela 4.1: Resumo dos modos de operação dos motores Dynamixel

4.3 Programação e Controlo

A implementação do controlo dos motores Dynamixel exige, numa fase inicial, a correta definição dos parâmetros de comunicação, de forma a garantir a fiabilidade da ligação entre o sistema computacional e os atuadores.

Entre os elementos essenciais a configurar, destaca-se a especificação da porta série utilizada, tipicamente `/dev/ttyUSB0` em sistemas operativos baseados em Linux, a taxa de transmissão de dados (*baud rate*), o protocolo de comunicação (sendo utilizada a versão 2.0), a série dos motores (neste caso, a série X, correspondente aos modelos Dynamixel XC330-M288-T) e o identificador único (*ID*) atribuído previamente a cada motor.

A correta parametrização destes elementos é fundamental para que seja possível estabelecer comunicação com os motores, enviar comandos de controlo e ler os parâmetros relevantes de operação. Nos pontos seguintes, descreve-se de forma progressiva a programação de um motor individual, o controlo coordenado de um dedo completo e, por fim, a implementação da lógica de controlo da mão robótica na sua totalidade.

Nesta secção, apresenta-se de forma estruturada a abordagem seguida para o desenvolvimento do sistema de controlo da mão robótica, descrevendo-se de forma progressiva a programação de um motor individual, o controlo coordenado de um dedo completo e, por fim, a implementação da lógica de controlo da mão robótica na sua totalidade. Seguidamente, é descrita a arquitetura de *software* desenvolvida que suporta a coordenação entre os diferentes componentes do sistema. Por fim, são detalhadas as funcionalidades avançadas implementadas, nomeadamente o ajuste dinâmico de corrente no contacto com objetos, a implementação de velocidades relativas entre dedos e falanges, e os atrasos temporais programáveis entre movimentos de diferentes dedos, permitindo um controlo mais seguro e adaptável.

Importa ainda referir que foi adotada uma estratégia base de fecho da mão, em que

todos os dedos se movimentam para posições pré-definidas com a mesma velocidade. Esta abordagem simples e eficaz explora a capacidade adaptativa da mão robótica, permitindo-lhe conformar-se à geometria de diferentes objetos durante o processo de preensão.

4.3.1 Programação de um Motor Individual

A programação de um motor individual constitui o primeiro passo no desenvolvimento da lógica de controlo da mão robótica. O processo tem início com a importação da biblioteca e a configuração da interface de comunicação série. Tal configuração é efetuada através da criação dos objetos `portHandler` e `packetHandler`, com as instruções `portHandler = PortHandler(DEVICENAME)` e `packetHandler = PacketHandler(PROTOCOL_VERSION)`. O primeiro estabelece a ligação física à porta série especificada (por exemplo, `/dev/ttyUSB0`), enquanto o segundo gere a codificação e descodificação dos pacotes de dados, de acordo com o protocolo de comunicação adotado (neste projeto, a versão 2.0).

Após a abertura da porta e a verificação do sucesso da ligação, é necessário ativar o torque do motor para que este possa responder aos comandos enviados. A partir deste momento, torna-se possível controlar diversos parâmetros do motor, como a posição, a velocidade e a corrente máxima, dependendo do modo de operação previamente selecionado.

A leitura e escrita desses parâmetros nos registos internos do motor são realizadas através das funções disponibilizadas pelo SDK. A leitura pode abranger dados como a posição atual, velocidade de rotação, corrente consumida ou valor de PWM. A escolha da função adequada depende da dimensão dos dados a transferir, sendo comum a utilização de `read2ByteTxRx()` e `read4ByteTxRx()` para valores de 2 e 4 bytes, respetivamente.

É importante salientar que os motores Dynamixel não fornecem diretamente a posição angular em graus ou radianos. Em vez disso, utilizam um sistema interno de representação baseado em *ticks*, no qual a posição é expressa por um valor inteiro. No caso do modelo `XC330-M288-T`, a resolução do encoder é de 12 bits, o que resulta em $2^{12} = 4096$ valores distintos ao longo de uma rotação completa de 360°. Deste modo, cada unidade (*tick*) corresponde a um incremento angular de aproximadamente 0.088°.

A conversão do valor lido, em *ticks*, para a correspondente posição angular em graus é efetuada através da equação (4.1):

$$\theta = \frac{360 \times \text{ticks}}{4095} \tag{4.1}$$

onde θ representa a posição angular em graus, e `ticks` é o valor devolvido diretamente pelo motor.

De forma análoga, a velocidade de rotação também é representada em *ticks*. Neste caso, a relação com as unidades físicas é definida por um fator de conversão fornecido pelo fabricante. Para o motor em questão, cada unidade corresponde a aproximadamente 0.229 rotações por minuto (RPM). Assim, a velocidade real pode ser obtida através da equação (4.2):

$$\text{Velocidade (RPM)} = \text{ticks} \times 0.229 \tag{4.2}$$

Adicionalmente, o sinal do valor atribuído à velocidade determina o sentido de rotação: valores positivos indicam rotação no sentido horário, enquanto valores negativos correspondem a uma rotação no sentido anti-horário.

No que diz respeito à corrente elétrica consumida pelo motor, esta é diretamente disponibilizada pelo SDK em miliamperes (mA), não sendo necessária qualquer conversão adicional. Este parâmetro está diretamente relacionado com o torque aplicado pelo motor sendo, por isso, relevante para o diagnóstico do esforço mecânico exigido durante o funcionamento.

4.3.2 Programação e Controlo de um Dedo

Após a compreensão do funcionamento individual dos motores Dynamixel e da configuração dos seus parâmetros, procedeu-se ao desenvolvimento do controlo coordenado de um dedo completo da mão robótica. Cada dedo integra quatro motores, tornando necessário garantir a leitura sincronizada dos seus estados e o envio simultâneo de comandos de controlo, de forma eficiente e escalável. Para tal, recorreu-se às funcionalidades de comunicação em bloco disponibilizadas pelo Dynamixel SDK, nomeadamente as operações `sync_read`, `sync_write`, `bulk_read` e `bulk_write`.

No que respeita à leitura dos parâmetros dos motores — posição, velocidade e corrente — a função `sync_read` foi a selecionada, por permitir aceder a um mesmo conjunto de registos em múltiplos motores de forma síncrona. Esta abordagem revelou-se eficiente tanto em termos de desempenho como de utilização de memória, além de respeitar as limitações de tamanho de mensagem impostas pelo protocolo de comunicação, evitando os riscos associados à utilização do `bulk_read`.

Na Tabela 4.2, apresenta-se uma comparação entre as duas funções de leitura disponibilizadas pelo Dynamixel SDK, evidenciando os critérios que suportaram a decisão de utilizar a função `sync_read`.

Critério	**`Sync Read`**	**`Bulk Read`**
Registos lidos por motor	Iguais (obrigatório)	Diferentes (flexível)
Tempo médio de leitura (12 motores, 3 registos)	3.5 ms	4.7 ms
Utilização de buffer de comunicação	~60 bytes	~85–100 bytes
Tamanho máximo de mensagem (versão 2.0)	143 bytes	143 bytes

Tabela 4.2: Comparação entre as funções `sync_read` e `bulk_read` do Dynamixel SDK

Relativamente à escrita de comandos, optou-se pela função `bulk_write`, que permite enviar dados distintos para registos diferentes de múltiplos motores numa única operação. Apesar de o registo alvo ser frequentemente o mesmo (como a posição ou o limite de corrente), esta escolha confere maior versatilidade ao sistema, permitindo a sua adaptação a cenários futuros em que diferentes parâmetros possam ser configurados seletivamente em cada motor.

Na Tabela 4.3, apresenta-se uma comparação entre as duas funções de escrita disponibilizadas pelo Dynamixel SDK, evidenciando os critérios que suportaram a decisão de utilizar a função `bulk_write`.

Critério	**Sync Write**	**Bulk_Write**
Registos enviados por motor	1	Vários
Tempo médio de envio (8 motores)	1.85 ms	2.10 ms
Utilização do buffer de comunicação	96 bytes	104 bytes
Tamanho máximo de mensagem (protocolo 2.0)	1472 bytes	1472 bytes

Tabela 4.3: Comparação entre as funções `sync_write` e `bulk_write` do Dynamixel SDK

Assim, a combinação de `sync_read` para a leitura periódica do estado dos motores e `bulk_write` para o envio de comandos de controlo assegura simultaneidade, escalabilidade e robustez no controlo coordenado de um dedo da mão robótica, servindo como base sólida para a extensão posterior ao controlo integrado de múltiplos dedos. Com esta abordagem foi possível atingir uma taxa de leitura estável de aproximadamente 50 Hz, garantindo uma monitorização suficientemente rápida para aplicações em tempo quase real.

Após implementar com sucesso o envio de comandos e a leitura simultânea dos parâmetros dos motores, procedeu-se à aplicação do modo de operação selecionado: o Current-Based Position Control Mode. A adaptação do código previamente desenvolvido foi direta, bastando configurar o registo correspondente ao modo de operação e definir um limite máximo de corrente na zona de inicialização do sistema. Esta configuração é realizada através da escrita nos endereços apropriados de cada motor Dynamixel.

Nos primeiros testes foi definido um limite de corrente de 100 mA. Embora este modo de operação não permita o controlo direto da velocidade, é possível estabelecer uma velocidade máxima por meio da configuração do parâmetro `Profile Velocity`. Assim, foram enviadas posições de destino previamente definidas para provocar o movimento de fecho e posterior abertura do dedo, com o objetivo de avaliar o comportamento dos motores em condições ideais. Os gráficos resultantes desta experiência encontram-se nas Figuras 4.4, 4.5 e 4.6.

Como se pode observar, durante todo o movimento, os valores de corrente e velocidade mantiveram-se dentro dos limites especificados. A única exceção ocorreu no motor 1 onde se registou uma ligeira oscilação de velocidade, ultrapassando o valor definido em apenas $0.07\,\mathrm{rad\,s^{-1}}$. Esta variação é considerada normal e esperada no funcionamento real dos motores, não comprometendo a estabilidade nem a segurança do sistema.

Na experiência seguinte foram mantidas as mesmas posições de destino, mas introduziu-se um obstáculo físico que impedia a concretização do movimento pretendido. Tal como ilustrado nas Figuras 4.7, 4.8 e 4.9, a análise dos dados revela diferenças claras entre os dois cenários.

Na presença do obstáculo o dedo não consegue atingir a posição alvo, resultando numa estabilização precoce da posição e no corte da velocidade. Simultaneamente, verifica-se um

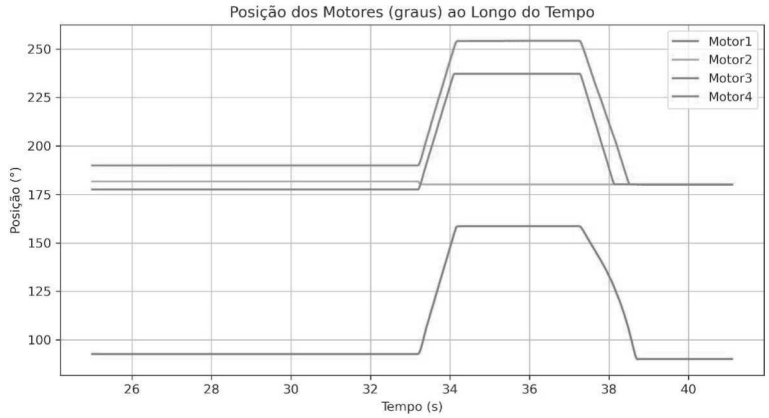


Figura 4.4: Posições dos motores ao longo do tempo durante o fecho e abertura do dedo sem nenhuma obstrução

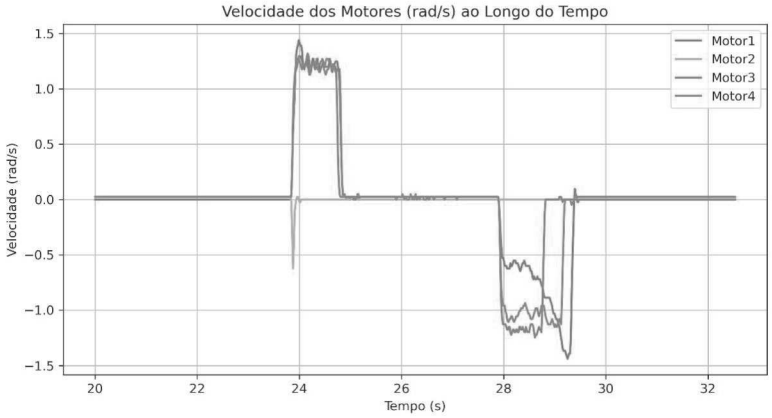


Figura 4.5: Velocidades dos motores ao longo do tempo durante o fecho e abertura do dedo sem nenhuma obstrução

aumento significativo na corrente consumida, evidenciando o esforço do motor perante a resistência sem nunca ultrapassar o limite de corrente definido. Após a remoção do obstáculo, o sistema retoma o movimento normal, alcançando a posição desejada com maior estabilidade e menor consumo energético.

Estes resultados comprovam a utilidade e a eficácia do *Current-Based Position Control Mode*, permitindo limitar a força exercida pelos motores durante a interação com o ambiente. Esta funcionalidade é fundamental para aplicações em que a segurança e a adaptabilidade ao contacto com objetos são essenciais, como é o caso da manipulação robótica.

Para além das duas experiências iniciais foram conduzidos testes adicionais com o objetivo de aprofundar a análise do comportamento dos motores em diferentes condições operacionais. Numa dessas experiências o dedo foi submetido a movimentos de abertura e fecho com diferentes velocidades máximas configuradas. Os resultados, apresentados no Apêndice A.2.1, demonstram que velocidades mais reduzidas conduzem a um menor consumo energético e a uma variação mais suave da posição ao longo do tempo. A inclinação menos acentuada da curva de posição indica um movimento mais progressivo e controlado, confirmando a influência

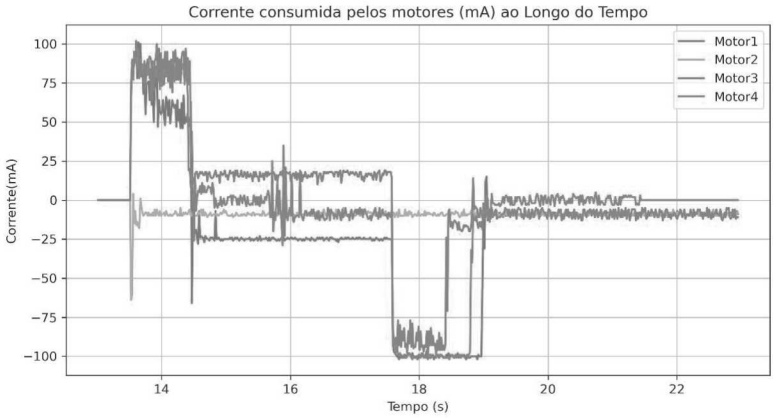


Figura 4.6: Correntes consumidas pelos motores ao longo do tempo durante o fecho e abertura do dedo sem nenhuma obstrução

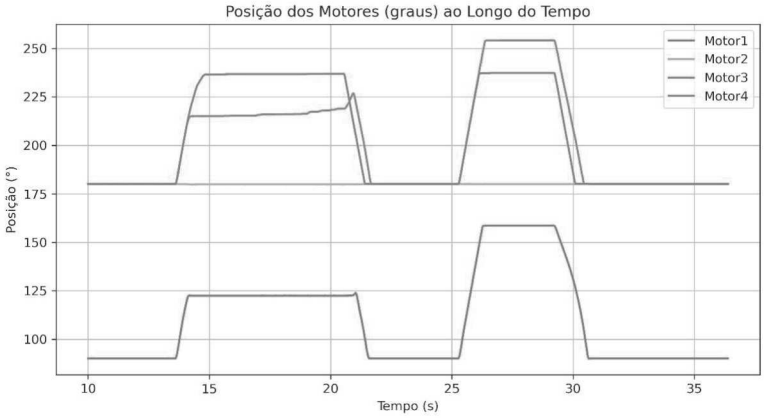


Figura 4.7: Posições dos motores ao longo do tempo durante o fecho e abertura do dedo com obstáculo

direta da velocidade sobre a dinâmica do sistema.

Outra experiência relevante consistiu em introduzir obstáculos que interrompessem o movimento do dedo em diferentes posições, com o intuito de estudar o efeito da localização do obstáculo no comportamento individual dos motores. Verificou-se que, quando o obstáculo atua sobre a base do dedo (afetando os motores inferiores), apenas o motor bloqueado entra em esforço, enquanto os restantes continuam a tentar atingir as suas posições alvo. Em contraste, quando o obstáculo é colocado na extremidade do dedo, todos os motores são simultaneamente afetados, ficando em esforço e impedidos de completar o movimento. Esta experiência, cujos gráficos se encontram no Apêndice A.2.2, reforça ainda a eficácia do controlo baseado em corrente, uma vez que a corrente consumida em nenhuma circunstância ultrapassou o limite máximo previamente definido.

Os dados experimentais obtidos encontram-se acompanhados de gráficos e ligações para vídeos demonstrativos no Apêndice A.2.1 e A.2.2, permitindo uma análise visual clara dos diferentes comportamentos observados. Estas experiências complementares contribuem significativamente para a validação do sistema de controlo implementado, demonstrando não

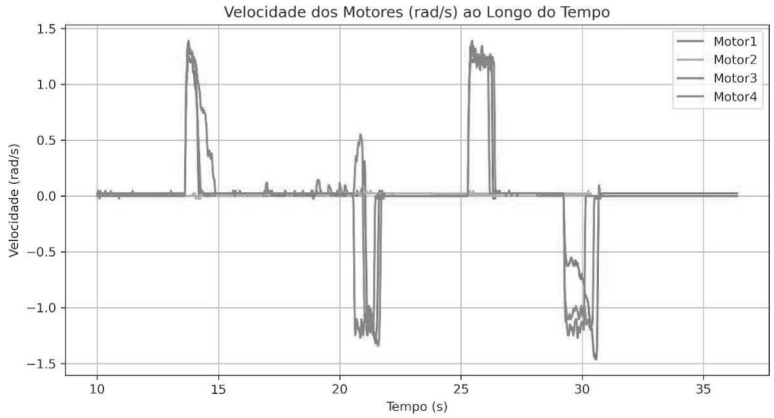


Figura 4.8: Velocidades dos motores ao longo do tempo durante o fecho e abertura do dedo com obstáculo

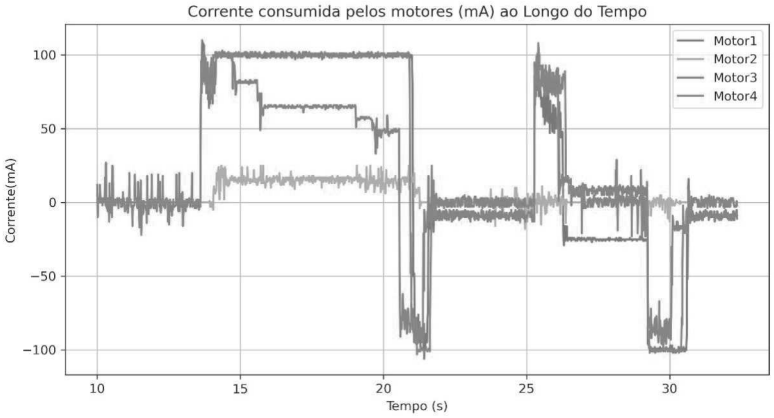


Figura 4.9: Correntes consumidas pelos motores ao longo do tempo durante o fecho e abertura do dedo com obstáculo

só a sua eficácia em cenários ideais, mas também a sua robustez perante interações físicas imprevistas.

Na Tabela 4.4 apresenta-se um resumo das duas experiências realizadas, incluindo uma breve descrição dos seus objetivos e a indicação da localização dos gráficos correspondentes que ilustram os resultados obtidos para uma melhor compreensão dos diferentes comportamentos da mão robótica sob as condições testadas.

Experiência	Conclusões	Gráficos
Movimentos de abertura e fecho com velocidades máximas variadas	Velocidades menores mostraram menor consumo e movimentos mais controlados, evidenciados por curvas de posição com inclinação reduzida.	Apêndice A.2.1
Interrupção do movimento por obstáculos em posições distintas	Obstáculos na base afetam apenas motores específicos, enquanto na extremidade todos os motores entram em esforço. A corrente nunca ultrapassou o limite definido, validando o controlo baseado em corrente.	Apêndice A.2.2

Tabela 4.4: Resumo das experiências complementares realizadas para análise do comportamento dos motores, com indicação da localização dos respetivos gráficos

4.3.3 Programação da Mão Completa

Para permitir o controlo simultâneo da mão robótica completa, bem como a leitura contínua dos valores de posição, velocidade e corrente consumida de todos os motores, foram necessárias apenas pequenas adaptações ao código previamente desenvolvido. Este já tinha sido concebido de forma modular e escalável, incluindo a deteção automática dos motores conectados, o que facilitou a sua extensão para múltiplos dedos.

A implementação consistiu essencialmente na inclusão das posições adicionais dos motores dos dedos restantes no nó responsável pelo envio das posições alvo. Paralelamente, foi necessário instanciar os nós correspondentes aos novos dedos, assegurando que cada conjunto de motores pudesse ser controlado de forma independente mas coordenada.

Estas alterações permitiram que o sistema mantivesse a mesma lógica de funcionamento utilizada no controlo de um único dedo, agora aplicada a toda a mão, garantindo a consistência e sincronização entre todos os atuadores. A lógica de comunicação e o tratamento dos dados lidos dos motores mantiveram-se inalterados, validando a robustez da abordagem inicialmente adotada.

Importa referir que a organização dos nós e a sua interligação são parte integrante da arquitetura de software desenvolvida, que será detalhada na secção seguinte.

4.3.4 Arquitetura de Software Desenvolvida

A arquitetura de software desempenha um papel fundamental no desenvolvimento de sistemas robóticos complexos, como é o caso da operação de uma mão robótica antropomórfica. Uma arquitetura bem definida não só influencia diretamente o desempenho e a escalabilidade do sistema, como também facilita a sua compreensão, manutenção e reutilização por outros utilizadores ou desenvolvedores. Uma estrutura modular clara e coerente é, por isso, essencial para garantir um desenvolvimento eficiente e uma experiência de utilização positiva.

Neste projeto optou-se por uma arquitetura modular baseada em múltiplos nós ROS de

forma a garantir modularidade, clareza e separação de responsabilidades.

A interação com o sistema inicia-se no nó `set_positions`, responsável por receber as posições alvo que o utilizador pretende que a mão atinja. Este nó funciona como a interface de alto nível do sistema e comunica essas posições ao `manager_node`, o nó central responsável por toda a interação com o hardware.

O `manager_node` desempenha um papel crítico, funcionando como elo de ligação entre o software e os motores Dynamixel. Por um lado, recebe as posições alvo (ou limites de corrente) enviadas pelos restantes nós e envia-as para os motores. Por outro lado, lê periodicamente o estado atual de todos os motores, incluindo posição, velocidade e corrente consumida.

Os dados lidos pelo `manager_node` são então distribuídos para quatro nós específicos, um para cada dedo da mão robótica: `thumb_node`, `index_node`, `middle_node` e `ring_node`. Cada um destes nós processa exclusivamente os dados do respetivo conjunto de motores, permitindo não só uma análise mais localizada e específica de cada dedo, como também a possibilidade de controlo independente por dedo. A Figura 4.10 ilustra a mão robótica com os dedos devidamente identificados, de forma a facilitar a compreensão da estrutura modular dos nós responsáveis pelo controlo de cada dedo.

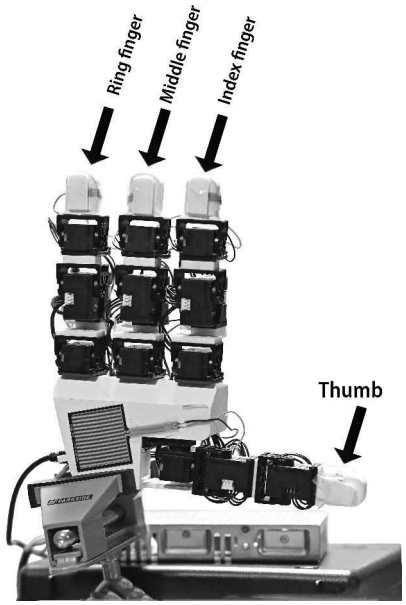


Figura 4.10: Fotografia da mão robótica desenvolvida, com a identificação dos dedos correspondentes a cada nó de controlo.

Adicionalmente, estes nós de dedo podem publicar mensagens para o nó `set_currents`, responsável por definir os limites de corrente para os motores. O `set_currents` envia os valores definidos de volta ao `manager_node`, que por sua vez os comunica aos motores.

Esta arquitetura modular e hierarquizada assegura uma operação fiável, facilita a expansão para novas funcionalidades e permite isolar falhas ou testar componentes individuais de forma independente. O diagrama da Figura 4.11 ilustra a interação entre os diferentes nós e a comunicação com o hardware, proporcionando uma visão clara do funcionamento global do sistema.

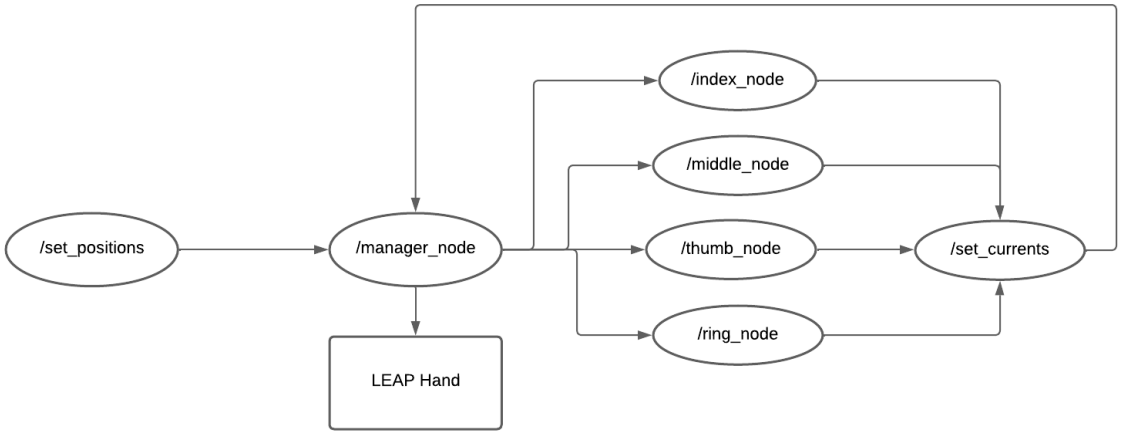


Figura 4.11: Diagrama da arquitetura de software desenvolvida

Com o objetivo de simplificar a utilização do sistema e facilitar o processo de inicialização, foi desenvolvido um *launch file* (apresentado no Bloco de Código 4.1) que automatiza a execução dos principais componentes da arquitetura. Este ficheiro lança automaticamente três terminais distintos:

- Um terminal dedicado à execução dos nós de controlo da mão, incluindo o `manager_node` e os nós de cada dedo;
- Um terminal responsável pela execução do nó de leitura dos sensores, permitindo a monitorização em tempo real dos valores adquiridos pelos sensores FSR;
- Um terceiro terminal que corre o nó de controlo da mão, no qual o utilizador pode inserir comandos diretamente no terminal, como por exemplo abrir ou fechar a mão completa, ou controlar dedos de forma individual.

```
def generate_launch_description():
    return LaunchDescription([
        # Terminal 1 - Managers
        ExecuteProcess(
            cmd=['gnome-terminal', '--', 'bash', '-c',
                 'ros2 run leap_hand_control hand_manager & '
                 'ros2 run leap_hand_control thumb_manager & '
                 'ros2 run leap_hand_control index_manager & '
                 'ros2 run leap_hand_control ring_manager & '
                 'ros2 run leap_hand_control middle_manager; exec bash'],
            shell=False
        ),

        # Terminal 2 - set_fingers_position
        ExecuteProcess(
            cmd=['gnome-terminal', '--', 'bash', '-c',
                 'ros2 run leap_hand_control set_fingers_position; exec bash'],
```

```
            shell=False
        ),

        # Terminal 3 - read_sensors
        ExecuteProcess(
            cmd=['gnome-terminal', '--', 'bash', '-c',
                 'ros2 run leap_hand_control read_sensors ; exec bash'],
            shell=False
        ),
    ])
```

Listagem 4.1: Launch File desenvolvido para automatizar a execução do sistema

Esta abordagem facilita significativamente a operação do sistema, reduz o tempo necessário para iniciar todos os componentes e proporciona uma experiência de utilização mais fluida, mesmo para utilizadores com menor familiaridade com a infraestrutura ROS.

4.3.5 Funcionalidades Avançadas

A arquitetura modular desenvolvida para o controlo da mão robótica não só garante uma operação robusta e eficiente, como também facilita significativamente a integração de novas funcionalidades. Esta capacidade de expansão é fundamental para adaptar o sistema a cenários de maior complexidade, melhorar a segurança da interação com o ambiente e oferecer maior expressividade nos movimentos da mão.

Tirando partido dessa flexibilidade, foram implementadas três funcionalidades avançadas que ampliam o potencial de controlo e aumentam a segurança operacional do sistema:

- **Ajuste dinâmico da corrente limite**: Permite adaptar automaticamente o limite de corrente dos motores em resposta ao contacto com objetos, evitando a aplicação de força excessiva e protegendo tanto o sistema robótico como os objetos manipulados.
- **Velocidades relativas entre dedos e falanges**: Introduz a possibilidade de definir velocidades relativas entre dedos e entre diferentes falanges de um mesmo dedo, possibilitando movimentos mais naturais, coordenados e eficientes.
- **Atrasos temporais programáveis entre movimentos**: Permite sequenciar ações de forma controlada e adaptada a diferentes tarefas ou contextos.

Estas funcionalidades são exploradas nas subsecções seguintes, onde se descreve o seu funcionamento, integração no sistema e os testes realizados para validar a sua eficácia.

Ajuste Dinâmico de Corrente no Contacto com Objetos

Um dos principais desafios na área da manipulação com mãos robóticas antropomórficas consiste em conseguir agarrar objetos sensíveis ou deformáveis sem os danificar, mantendo uma preensão estável. A aplicação de forças excessivas pode levar ao esmagamento ou deformação irreversível do objeto sendo, por isso, essencial implementar mecanismos de controlo adaptativo que regulem a força aplicada em tempo real. Com esse objetivo, foi desenvolvida uma funcionalidade de ajuste dinâmico da corrente limite nos motores, de forma a

restringir o esforço exercido sempre que é detetado contacto com um objeto. Para tal, adaptou-se o código dos nós individuais de cada dedo, permitindo que estes ajustem automaticamente a corrente limite com base em dois critérios principais: o aumento da corrente consumida e a diminuição significativa da aceleração do motor. A Figura 4.12 ilustra uma situação em que, após a deteção de contacto com o objeto, foi aplicada uma corrente limitada de forma a prevenir a aplicação excessiva de força e, consequentemente, o risco de esmagamento.

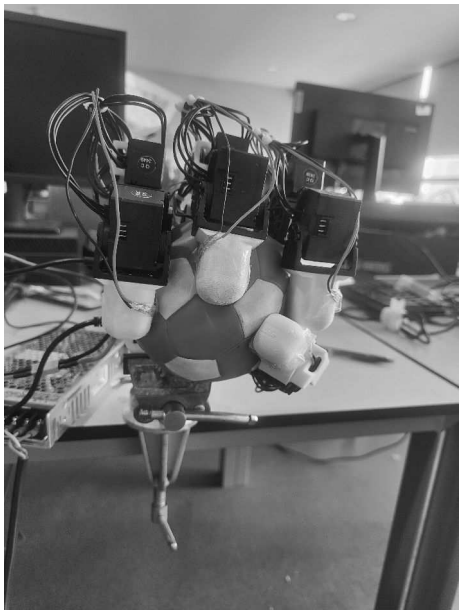


Figura 4.12: Exemplo de controlo de corrente onde, após o contacto entre o dedo robótico e o objeto, é imposta uma corrente limite para evitar forças excessivas que possam danificar o objeto.

A deteção de um aumento de corrente baseia-se na comparação das correntes lidas para cada motor do dedo com um valor máximo pré-definido no nó, denominado `GOAL_CURRENT_VALUE`. Se alguma das correntes medidas exceder este valor, considera-se que o motor está em carga excessiva. Inicialmente a redução de velocidade era avaliada diretamente através da diferença entre a velocidade atual e a registada no instante anterior. Contudo, devido à variação do intervalo temporal entre amostras consecutivas, essa abordagem revelou-se inconsistente. Assim, optou-se por considerar a aceleração como métrica mais fiável, tendo-se adicionado à mensagem enviada para os dedos um parâmetro correspondente ao intervalo de tempo entre amostras. Com essa informação, torna-se possível calcular a aceleração de forma precisa.

Desta forma define-se que a deteção de um contacto com obstáculo ocorre quando se verifica simultaneamente um aumento significativo da corrente e uma redução expressiva da aceleração, indicando que o motor está a aplicar força sem conseguir avançar, ou seja, em contacto com um objeto. De forma a visualizar mais facilmente o processo descrito, na Figura 4.13 apresenta-se o diagrama referente à lógica de implementação da funcionalidade de ajuste dinâmico da corrente no contacto com objetos.

Para validar esta funcionalidade, foi realizada uma experiência em que se introduziu um obstáculo a impedir o movimento de um dos dedos da mão robótica. Nas Figuras 4.14

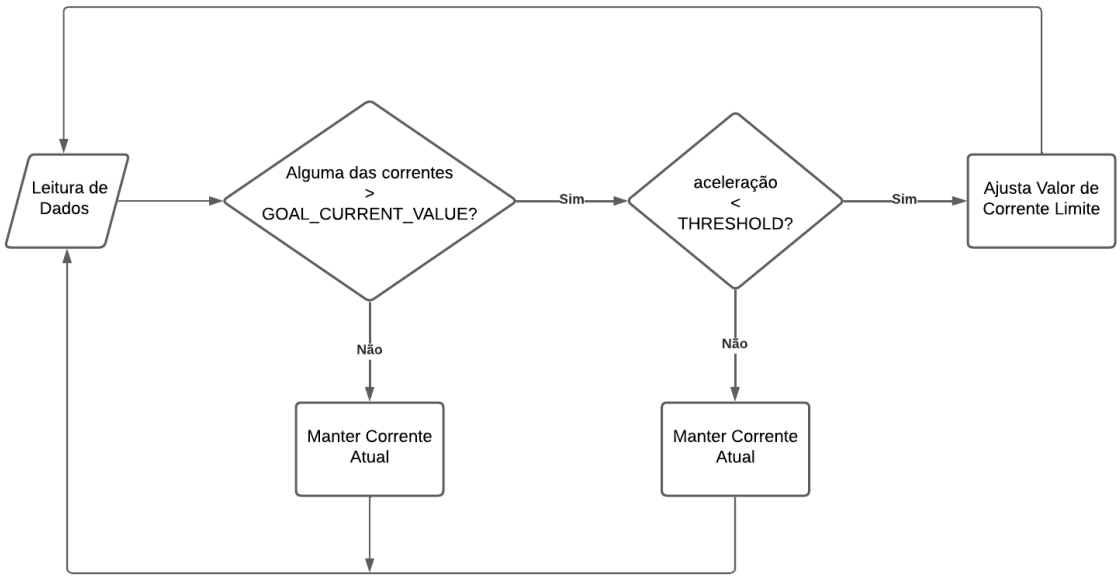


Figura 4.13: Diagrama da lógica de ajuste dinâmico da corrente dos motores da mão robótica no contacto com objetos

apresentam-se os gráficos relativos às posições e correntes consumidas pelos motores de um dedo em duas situações: com e sem obstáculo. Nesta experiência a corrente limite inicial foi definida como 200 mA, enquanto o valor para o qual a corrente era ajustada dinamicamente, aquando da deteção de contacto, foi fixado em 100 mA.

A análise dos gráficos mostra que durante o movimento livre do dedo (sem obstáculo), ocorrem picos de corrente superiores a 100 mA, o que é normal dado o esforço requerido pelos motores para vencer a resistência mecânica do sistema. No entanto, no momento em que o dedo encontra o obstáculo, observa-se uma estabilização da corrente nos 100 mA, refletindo a deteção de contacto e a consequente adaptação do valor limite de corrente. Após a remoção do obstáculo voltam a registar-se picos superiores a 100 mA, permitindo ao dedo retomar o seu movimento normal com a corrente limite restabelecida.

Esta funcionalidade representa um avanço significativo na segurança e adaptabilidade da mão robótica, permitindo interações mais delicadas e controladas com objetos de diferentes propriedades físicas.

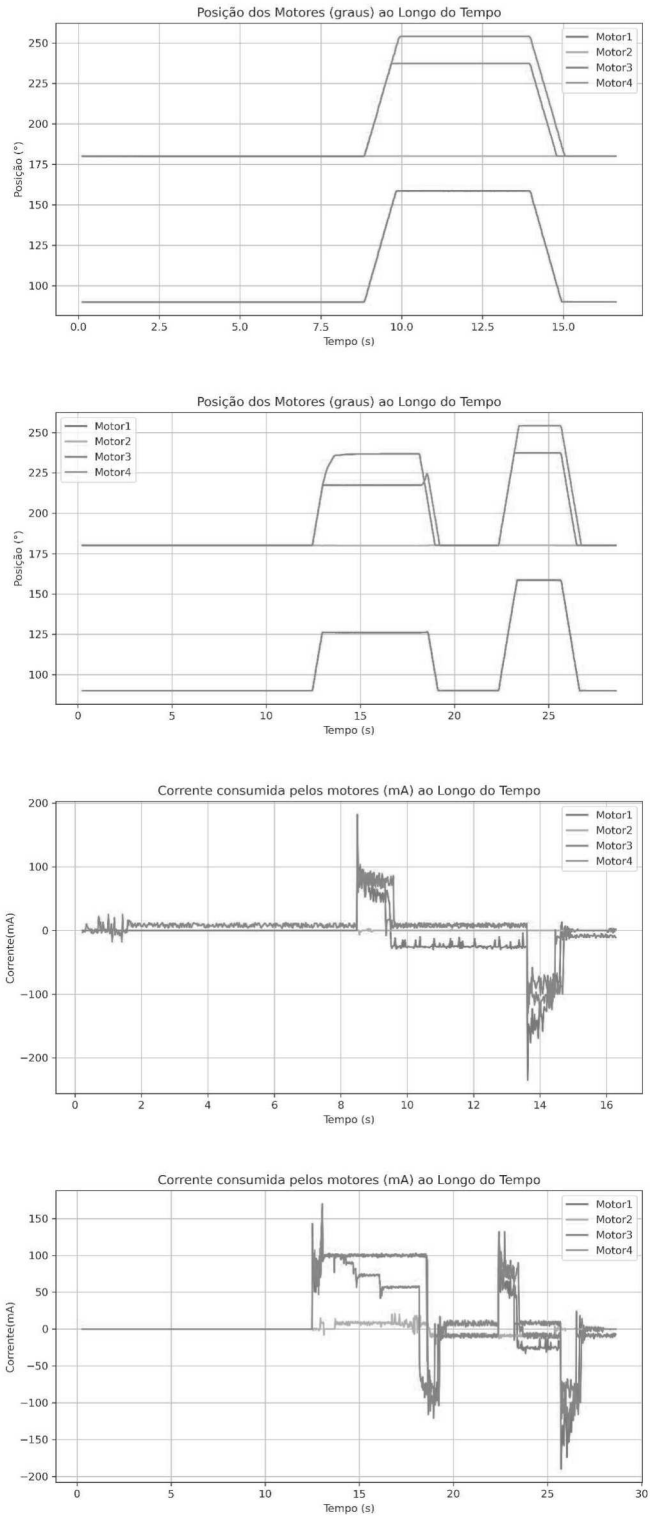


Figura 4.14: Gráficos da posição e da corrente consumida pelos motores de um dedo em dois cenários distintos: sem obstáculo (primeiro e terceiro gráfico) e com obstáculo a impedir o movimento (segundo e quarto gráfico).

Implementação de Velocidades Relativas entre Dedos e Falanges

O ser humano possui uma notável capacidade de controlar a sua mão de forma precisa e adaptativa, aplicando diferentes velocidades tanto entre os dedos como entre as falanges de um mesmo dedo. Esta habilidade permite executar gestos variados, aparentemente semelhantes, mas com propósitos distintos e ajustados ao contexto, como agarrar, ajustar ou largar objetos de diferentes formas.

Com o objetivo de aproximar o comportamento da mão robótica antropomórfica do seu equivalente biológico, foi implementado neste projeto um mecanismo que permite a definição de fatores de velocidade relativa, tanto entre os diferentes dedos como entre as várias falanges de cada dedo. Esta funcionalidade possibilita, por exemplo, que o polegar alcance a sua posição de destino antes dos restantes dedos, ou que a ponta de um dedo se mova mais rapidamente do que a sua base, permitindo movimentos mais articulados e naturais.

A implementação desta funcionalidade realizou-se com uma modificação ao nó `hand_manager`, onde se definiu a classe `Finger`. Esta classe inclui como atributos o nome do dedo, o fator de velocidade relativa do dedo face aos restantes, e um dicionário com os fatores de velocidade relativa entre as diferentes juntas (motores) do mesmo dedo. Por exemplo, a construção `Finger("index", 1, "0": 1, "1": 1, "2": 1, "3": 1)` representa o dedo indicador com fator de velocidade relativa global 1 e com velocidades iguais entre todas as suas juntas. Este sistema permite configurar rapidamente diferentes padrões de movimento através da alteração dos fatores no dicionário de dedos.

Para validar esta funcionalidade realizaram-se duas experiências distintas, cujos resultados são apresentados de seguida.

Na primeira experiência atribuiu.se ao dedo médio o dobro da velocidade do polegar. A Figura 4.15 apresenta os gráficos das velocidades dos motores de ambos os dedos durante o movimento de fecho e abertura da mão. Para além dos gráficos também é possível visualizar os vídeos referente a esta experiência no Apêndice A.3.1

A análise dos gráficos e a visualização do vídeo revela que, conforme esperado, o dedo médio alcança a posição de destino mais rapidamente do que o polegar, devido ao seu fator de velocidade mais elevado. Esta diferença de sincronização resulta num posicionamento em que o polegar termina o seu movimento por cima do dedo médio.

Além disso, é possível observar um pico de corrente no polegar durante o movimento de abertura da mão. Este pico ocorre porque, ao estar o polegar posicionado por cima do dedo médio, e sendo o dedo médio mais rápido (devido ao seu fator de velocidade superior), este tenta abrir antes do polegar. Como resultado, o dedo médio empurra mecanicamente o polegar que se encontra por cima, provocando resistência à sua própria abertura.

Na segunda experiência, testou-se o impacto da variação de velocidade entre falanges de um único dedo. Apenas um dedo foi movimentado, sendo atribuído à sua primeira falange (motor mais próximo da base) o dobro da velocidade das restantes. A Figura 4.16 apresenta os gráficos das velocidades dos motores ao longo do tempo para duas situações distintas: na primeira, todos os motores do dedo operam com a mesma velocidade; na segunda, o motor da primeira falange executa o movimento com o dobro da velocidade dos demais. Tal como

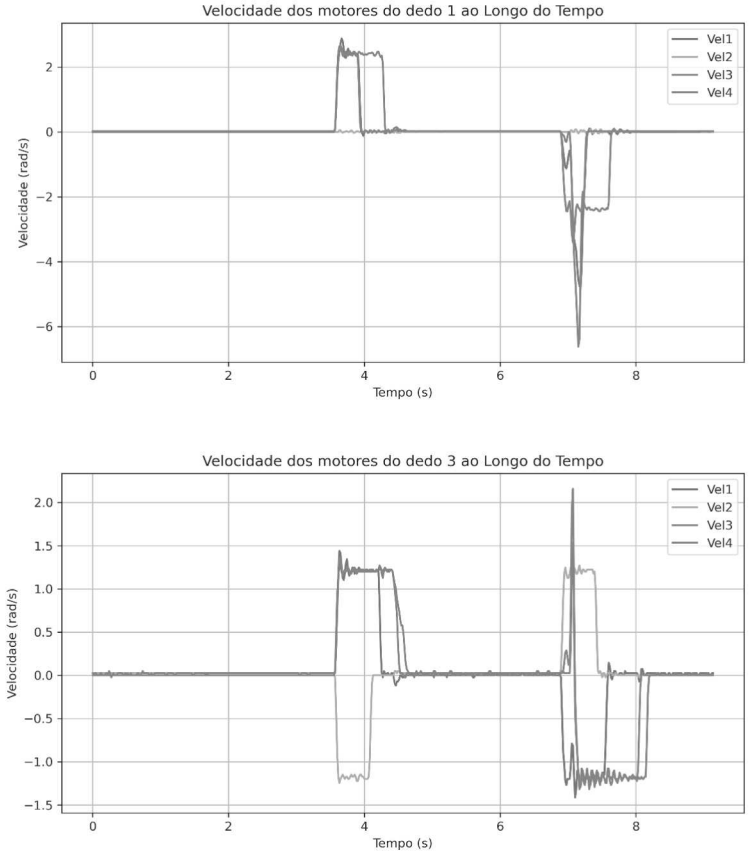


Figura 4.15: Gráficos das velocidades para o dedo médio e para o polegar quando se aplicou o dobro da velocidade do polegar ao dedo médio

na primeira experiência, os vídeos referentes a esta experiência podem ser consultados no Apêndice A.3.1 para observação complementar dos resultados.

A análise dos gráficos e a visualização dos vídeos indica que a primeira falange inicia e termina o seu movimento antes das restantes, conduzindo o movimento do dedo como um todo. Este comportamento faz com que o dedo feche de forma mais rápida na base, completando o gesto de agarrar apenas na fase final do movimento, quando as falanges mais distais se aproximam da posição final.

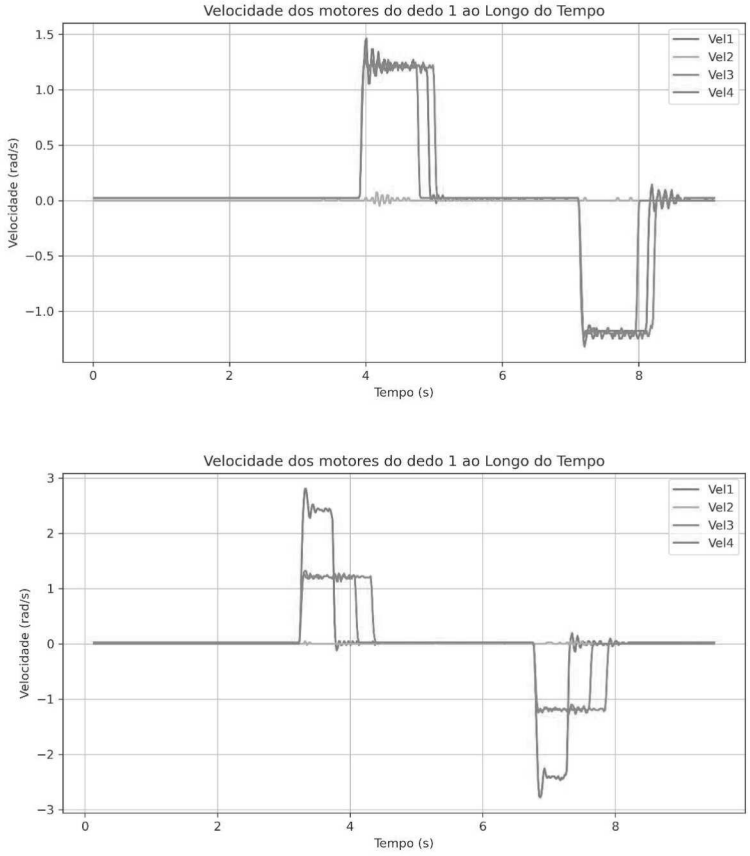


Figura 4.16: Gráficos das velocidades dos motores de um dedo em duas configurações distintas: em cima, todos os motores operam com a mesma velocidade; em baixo, o motor da primeira falange (mais próximo da base) opera com o dobro da velocidade das restantes.

Implementação de Atrasos Temporais Programáveis entre Movimentos de Dedos

A mão humana é capaz de executar movimentos coordenados entre os dedos que, embora relacionados, não ocorrem necessariamente de forma simultânea. De forma a replicar essa capacidade na mão robótica desenvolvida, implementou-se um mecanismo de controlo que permite definir atrasos temporais programáveis entre os movimentos dos diferentes dedos. Para tal, foi desenvolvida a função `publish_ordered_positions` no nó `set_positions.py`, que permite o envio escalonado de comandos de posição para diferentes dedos, com base em offsets temporais definidos previamente. Esta funcionalidade acrescenta flexibilidade ao sistema, possibilitando a criação de movimentos mais expressivos, adaptados ao contexto da tarefa executada.

O funcionamento da função `publish_ordered_positions` pode ser descrito da seguinte forma:

1. As entradas (pares de dedo e posição) são ordenadas por ordem crescente de offset;
2. Num primeiro momento, são enviadas todas as posições correspondentes aos dedos cujo offset é igual a zero;

3. Em seguida, o sistema aguarda um intervalo de tempo calculado como a diferença entre o offset do último grupo de dedos enviado e o offset do próximo grupo;
4. Após esse intervalo, são publicadas as posições dos dedos seguintes;
5. Este processo é repetido até que todas as posições tenham sido enviadas, respeitando os offsets temporais definidos para cada dedo.

Para exemplificar a utilização desta funcionalidade, foi criado o movimento designado por *wave*, no qual a mão fecha de forma sequencial, iniciando pelo polegar, seguido sucessivamente pelos restantes dedos. Neste caso, o offset temporal entre o polegar e o dedo seguinte foi fixado em 0.5 s, enquanto os offsets entre os restantes dedos foram definidos como 0.2 s.

Na fase de reabertura da mão optou-se por inverter a ordem dos offsets, garantindo assim uma simetria no movimento — ou seja, os dedos que fecharam por último passam a ser os primeiros a abrir. Para isso, os offsets de abertura foram calculados com base na expressão presente em 4.3.

$$\text{offset}_{\text{abertura}} = \text{offset}_{\text{máximo}} - \text{offset}_{\text{fecho}} \tag{4.3}$$

Apesar de não terem sido adquiridos gráficos nesta experiência, o movimento pode ser visualizado no vídeo correspondente, cuja ligação se encontra no Apêndice A.3.1. O vídeo demonstra claramente o efeito dos atrasos programados na execução do gesto, reforçando o potencial desta funcionalidade para gerar interações mais naturais e contextualmente adaptadas.

Na Figura 4.17a, é possível observar uma situação em que o polegar ficou por baixo dos restantes dedos, enquanto na Figura 4.17b se verifica uma configuração em que o polegar ficou por cima. Estas variações ilustram a flexibilidade do sistema em adaptar a ordem dos movimentos, resultando em diferentes gestos e formas de interação com objetos.

(a) Exemplo de execução do gesto com o polegar a posicionar-se por baixo dos restantes dedos.

(b) Exemplo de execução do gesto com o polegar a posicionar-se por cima dos restantes dedos.

Figura 4.17: Variações de gesto obtidas através da definição de diferentes offsets temporais entre os dedos.

4.4 Implementação dos Sensores de Contacto

A capacidade de detetar contacto físico com objetos é um elemento crucial para o desenvolvimento de sistemas robóticos mais seguros, precisos e adaptáveis ao ambiente. No contexto deste projeto a integração de sensores de contacto visa não apenas a perceção de toque, mas também a monitorização da distribuição de forças ao longo dos dedos, permitindo implementar estratégias de controlo baseadas em *feedback* sensorial. Este tipo de funcionalidade é particularmente relevante para tarefas que envolvem manipulação delicada ou interação com objetos de diferentes formatos e texturas, sendo um importante contributo para a robustez e versatilidade da mão robótica.

Esta secção descreve detalhadamente o processo de implementação dos sensores de contacto, desde a seleção do microcontrolador até à montagem física dos componentes e ao desenvolvimento de um PCB. Para além dos aspetos de hardware, foram também realizados diversos testes com sensores do tipo FSR, com o objetivo de validar a sua resposta e sensibilidade ao contacto.

4.4.1 Seleção do Microcontrolador

A seleção do microcontrolador responsável pela aquisição dos sinais provenientes dos sensores de contacto é um passo fundamental para garantir a fiabilidade e eficiência do sistema sensorial da mão robótica desenvolvida. Para além da capacidade de leitura adequada dos sensores, esta escolha deve também considerar as restrições físicas impostas pela estrutura da mão robótica, bem como a compatibilidade com o ecossistema de desenvolvimento adotado ao longo do projeto, o ROS.

O modelo sensorial inicialmente previsto contempla a utilização de quatro sensores FSR 400, instalados nas pontas dos dedos, e um sensor FSR 406 localizado na região da palma da mão. Esta configuração requer, no mínimo, cinco entradas analógicas disponíveis no microcontrolador. Contudo, esta configuração poderá sofrer alterações em trabalhos futuros, pelo que é importante considerar alguma margem de expansão na escolha do microcontrolador, bem como a eventual integração de um PCB.

Recorrendo ao software FreeCAD foi possível determinar as dimensões máximas permitidas para a instalação do microcontrolador na palma da mão, tal como ilustrado na Figura 4.18, onde a zona de alojamento se encontra destacada a vermelho.

As medições, visíveis na Figura 4.19 indicam que a unidade de controlo não deverá exceder $23.40\,\text{mm} \times 52.63\,\text{mm}$.

Com base nestas dimensões e nos requisitos técnicos do sistema, foi realizada uma análise comparativa de diferentes microcontroladores compatíveis, resumida na Tabela 4.5. Esta análise teve em consideração o número de entradas analógicas, a frequência de operação e o tamanho físico do dispositivo.

O Teensy 4.0 (Figura 4.20a) destaca-se como a opção mais robusta, combinando uma elevada frequência de operação com um número generoso de entradas analógicas (14), o que assegura não só a leitura precisa dos sensores FSR, como também permite futuras expansões. Adicionalmente, a sua compatibilidade com o ambiente de desenvolvimento Arduino IDE,

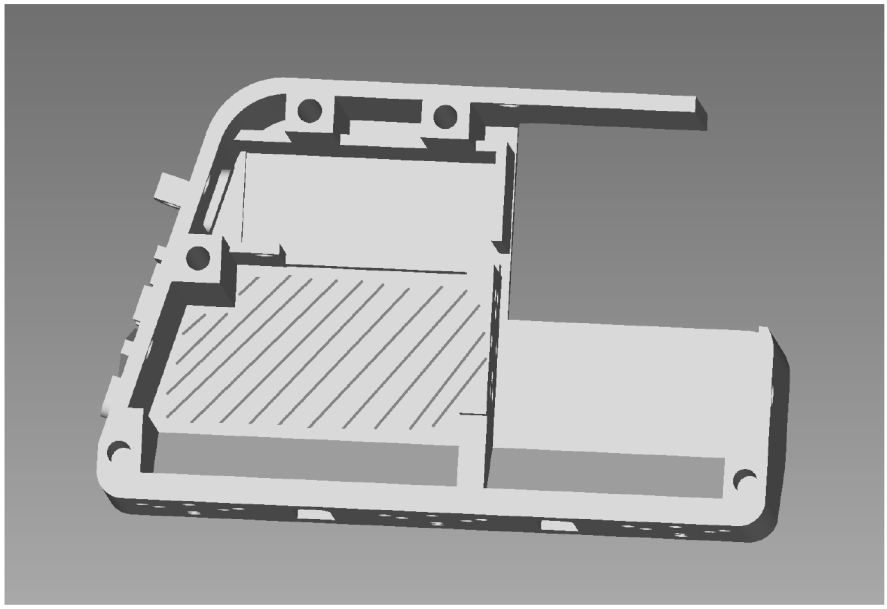


Figura 4.18: Peça da palma da mão com o espaço reservado para o microcontrolador destacado a vermelho. Figura obtida com o software FreeCad.

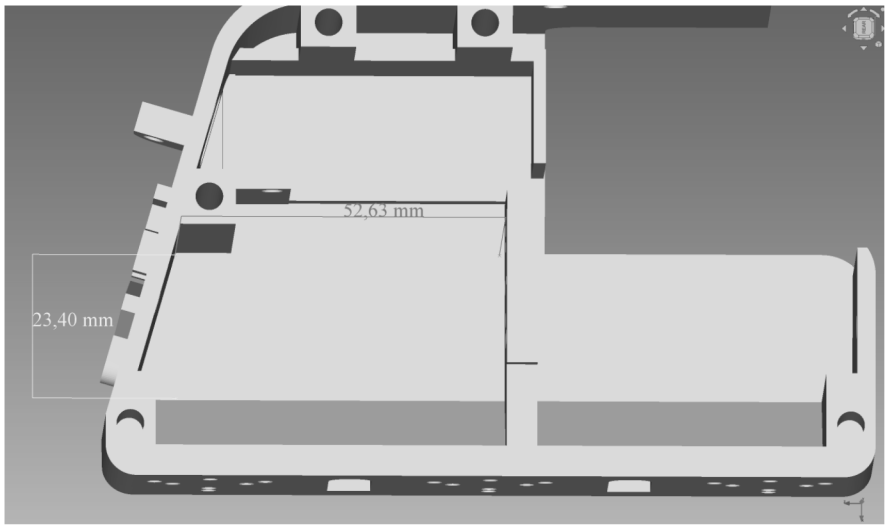


Figura 4.19: Medições efetuadas com a ferramenta FreeCad

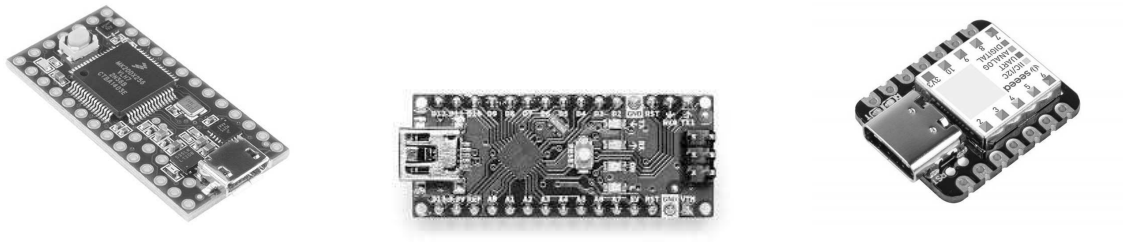
PlatformIO e bibliotecas como *rosserial* torna-o particularmente adequado à integração com o sistema baseado em ROS. A resolução analógica de 12 bits e o processador ARM Cortex-M7 conferem-lhe ainda vantagens em termos de desempenho e resposta temporal.

O Arduino Nano (Figura 4.20b), embora amplamente documentado e suportado pela comunidade, apresenta limitações ao nível do desempenho e número de entradas analógicas. Já o Seeeduino XIAO (Figura 4.20c), apesar de ser a solução mais compacta, oferece menos suporte na integração com ROS e menor desempenho comparativamente ao Teensy.

Assim, a escolha recaiu sobre o Teensy 4.0 por reunir um conjunto equilibrado de características técnicas, dimensão física adequada e excelente compatibilidade com o ecossistema de desenvolvimento utilizado neste projeto.

Modelo	Entradas Analógicas	Frequência	Dimensões (mm)
Teensy 4.0	14	600 MHz	19 × 36
Arduino Nano	8	16 MHz	19 × 43
Seeeduino XIAO	11	48 MHz	18 × 20

Tabela 4.5: Comparação entre microcontroladores compatíveis com as dimensões disponíveis.



(a) Teensy 4.0 **(b)** Arduino Nano **(c)** Seeeduino XIAO

Figura 4.20: Microcontroladores considerados para análise.

4.4.2 Implementação Física do Sistema sensorial

A implementação física do sistema sensorial é uma etapa essencial para garantir a integração eficiente dos sensores de contacto com a arquitetura eletrónica e mecânica da mão robótica. Esta implementação envolveu não só a preparação do circuito de aquisição dos sinais provenientes dos sensores, mas também a conceção de soluções mecânicas para a fixação segura e funcional dos componentes eletrónicos no interior da palma da mão desenvolvida.

Nesta subsecção são detalhadas três vertentes fundamentais do processo: a **Unidade de Aquisição dos Sensores**, onde se descreve a necessidade da utilização de divisores de tensão para permitir a leitura correta dos sinais analógicos pelo microcontrolador Teensy 4.0; o **Desenho da Peça para Fixação**, que corresponde à modelação 3D da estrutura de suporte do microcontrolador e do circuito impresso, de modo a garantir a sua instalação estável dentro da mão; e por fim, o **Desenvolvimento do PCB**, onde se apresenta o PCB concebido para acomodar as ligações entre todos os sensores FSR utilizados e os pinos analógicos disponíveis, assegurando uma montagem compacta e funcional.

Esta abordagem integrada permitiu construir um sistema sensorial modular e escalável, capaz de suportar as necessidades do projeto e de se adaptar a futuras modificações na disposição ou quantidade de sensores.

Unidade de Aquisição dos sensores

Os sensores de força resistivos (FSR) são dispositivos cuja resistência elétrica varia inversamente com a força aplicada sobre a sua superfície sensível. Ou seja, quanto maior a força exercida, menor a sua resistência. Esta característica permite a medição de pressões ou contactos físicos de forma relativamente simples, sendo ideal para aplicações em robótica onde se pretende detetar interações entre a mão robótica e objetos do ambiente.

Os microcontroladores não conseguem medir diretamente a resistência elétrica. Para converter esta variável resistiva num sinal analógico de tensão mensurável, é necessário implementar um circuito divisor de tensão. Este circuito consiste em ligar o sensor FSR em série com uma resistência fixa, aplicando uma tensão constante nas extremidades da série. A tensão de saída é então medida no ponto comum entre a resistência fixa e o sensor FSR. Quando o sensor é pressionado, a sua resistência diminui, fazendo com que a tensão no ponto de leitura aumente. Esta variação de tensão pode então ser lida por uma entrada analógica do microcontrolador.

A Figura 4.21 ilustra o esquema de um circuito divisor de tensão desenhado com a ferramenta Fritzing, utilizando um sensor FSR, uma resistência e o microcontrolador Teensy 4.0. Este tipo de montagem foi replicado para cada sensor de contacto utilizado no projeto.

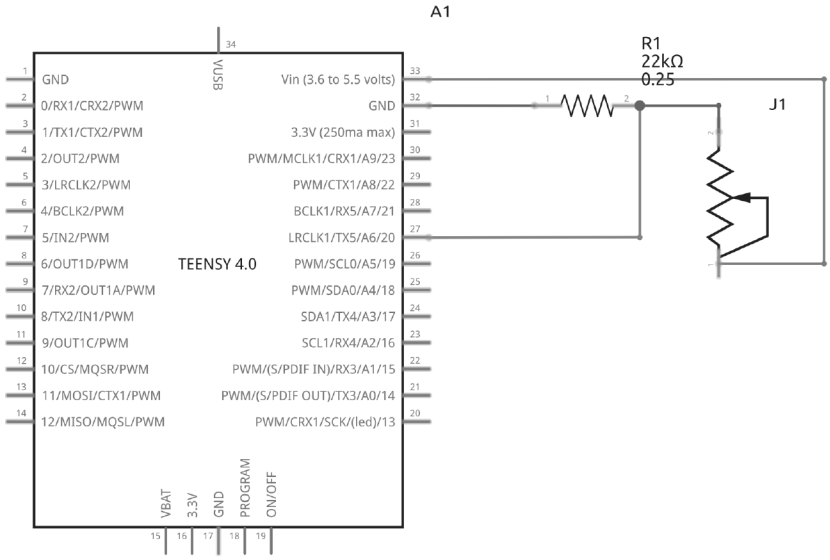


Figura 4.21: Esquema de circuito com divisor de tensão utilizando um sensor FSR (representado por J1), uma resistência e um Teensy 4.0, criado com o Fritzing.

O valor da resistência no divisor de tensão é um parâmetro crítico no circuito de leitura dos sensores FSR, uma vez que influencia diretamente a sensibilidade do sistema. Em geral, para circuitos com sensores de força resistivos, os valores de resistência utilizados situam-se entre 1 kΩ e 100 kΩ, dependendo da aplicação e do intervalo de forças a ser medido.

Para um divisor de tensão, o cálculo da tensão de saída é dado pela expressão presente em 4.4, onde V é a tensão de alimentação (neste caso, 3.3 V), R é a resistência fixa, R_{FSR} é a resistência variável do sensor, e V_{out} é a tensão de saída lida pelo microcontrolador.

$$V_{\text{out}} = V \cdot \frac{R}{R + R_{\text{FSR}}} \tag{4.4}$$

Na Figura 4.22 é possível observar a curva característica do sensor FSR 400, disponibilizada pela Interlink Electronics[26]. Esta curva representa a resistência do sensor em função da força aplicada (em gramas), e permite estimar a gama de valores que o sensor irá apresentar no contexto da aplicação.

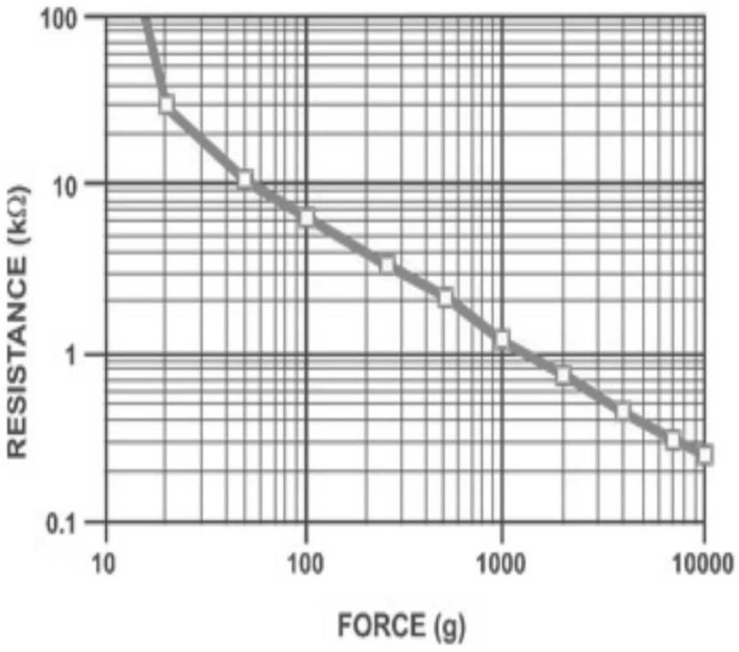


Figura 4.22: Curva característica do sensor FSR 400, indicando a resistência em função da força aplicada. Retirada da ficha técnica do fabricante (Interlink Electronics[26]).

Tendo em consideração que neste projeto não se esperam forças superiores a 2000 g, é possível, a partir da curva, estimar que para essa força a resistência do sensor se situa entre aproximadamente 0.8 kΩ e 0.6 kΩ. Com base na Equação 4.4, e assumindo uma tensão de alimentação de 3.3 V e um valor de saída desejado próximo de 3.2 V (sem atingir a saturação), é possível calcular os valores de resistência fixa ideais para o divisor.

Substituindo os valores na equação para os dois extremos de resistência do sensor, obtêm-se os seguintes resultados:

- Para $R_{\text{FSR}} = 0.8\,\text{k}\Omega$:

$$R_{\text{M}} = 25.6\,\text{k}\Omega$$

- Para $R_{\text{FSR}} = 0.6\,\text{k}\Omega$:

$$R_{\text{M}} = 19.2\,\text{k}\Omega$$

Deste modo selecionou-se um valor intermédio de 22 kΩ para a resistência fixa, de forma a obter uma resposta adequada do sistema para o intervalo de forças esperado, sem comprometer a resolução da leitura analógica nem atingir valores de saturação.

Desenho da peça para fixação

A estrutura da LEAP Hand [13] não foi originalmente concebida com sensores de contacto integrados. No entanto, conforme descrito na Subsecção 4.4.1, existe algum espaço disponível

no interior da palma da mão, o que possibilitou a implementação de um sistema sensorial adicional.

De forma a aproveitar o espaço disponível e garantir a correta fixação dos componentes eletrónicos, nomeadamente o microcontrolador Teensy 4.0 e o circuito impresso (PCB), foi necessário desenhar uma peça em 3D utilizando a ferramenta Tinkercad. Após uma análise cuidada do espaço interno disponível, optou-se por posicionar tanto o microcontrolador como o PCB na vertical, uma vez que esta orientação maximiza a utilização do espaço e facilita a sua remoção para futuras modificações ou manutenções.

Para o Teensy 4.0, foi desenvolvida uma ranhura com 1.58 mm de largura, tendo em conta a sua espessura de 1.57 mm, de forma a garantir um encaixe justo. O comprimento da ranhura foi definido em 36mm, ligeiramente superior ao comprimento do microcontrolador (35.56 mm), e a altura de 5 mm foi escolhida para proporcionar estabilidade suficiente. A peça de fixação dedicada ao Teensy pode ser observada na Figura 4.23.

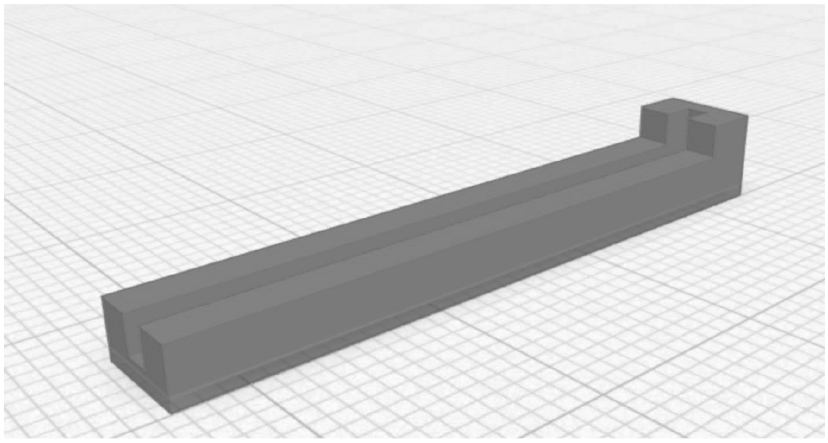


Figura 4.23: Peça desenvolvida com a ferramenta TinkerCad para fixar o Teensy 4.0 na palma da mão

Para o PCB, aplicou-se a mesma lógica de construção. Foi desenhada uma ranhura com 1.48 mm de largura (para acomodar um PCB de 1.47 mm de espessura) e 44 mm de comprimento. Contrariamente ao suporte do Teensy, não foi necessário incluir elementos adicionais para impedir o deslizamento vertical, uma vez que a própria estrutura da palma da mão já possui duas paredes laterais que garantem a contenção do PCB. A peça de fixação do PCB encontra-se representada na Figura 4.24.

Por fim, para assegurar uma montagem precisa e integrada, foi desenvolvida uma peça única com a geometria correspondente ao espaço disponível na palma da mão. Nessa base comum foram integradas as duas estruturas anteriormente desenvolvidas para a fixação do microcontrolador e do PCB nas posições definidas. O resultado final da peça completa pode ser visualizado na Figura 4.25.

A peça final foi fixada na estrutura da palma da mão com recurso a cola epóxi, garantindo uma ligação firme e duradoura entre os componentes e a estrutura da mão robótica. Adicionalmente, foi realizado um furo lateral na própria peça da palma da mão, de modo a permitir a passagem do cabo USB responsável pela alimentação e programação do microcon-

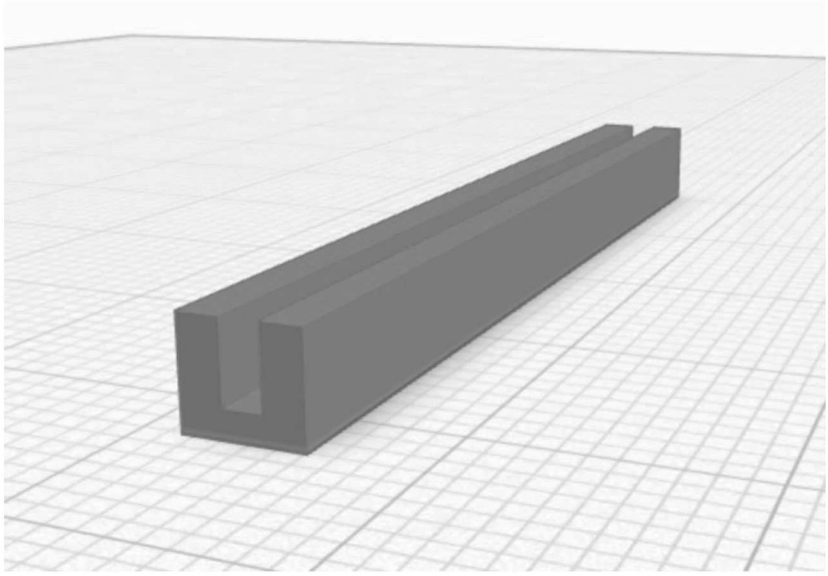


Figura 4.24: Peça desenvolvida com a ferramenta TinkerCad para fixar o PCB na palma da mão

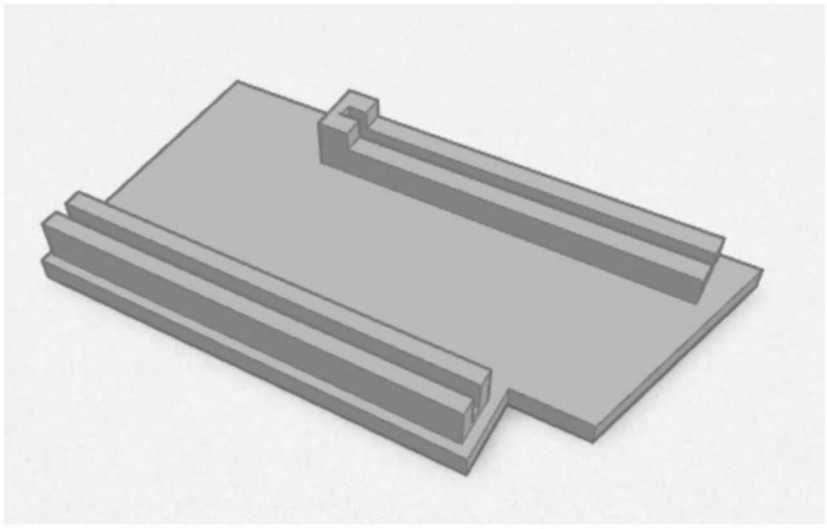


Figura 4.25: Peça final desenvolvida com a ferramenta TinkerCad para fixar o o Teensy 4.0 e o PCB na palma da mão

trolador, assegurando o acesso contínuo à porta de comunicação do Teensy sem comprometer a integridade estrutural do sistema.

Desenvolvimento do PCB

Tendo em conta a utilização de 5 sensores de contacto, surgiu a necessidade de desenvolver uma solução de interligação eficiente entre os sensores e o microcontrolador. Para tal, foi construído um circuito em *protoboard* com vista à criação de um PCB funcional que garantisse a correta distribuição de sinais e alimentação elétrica.

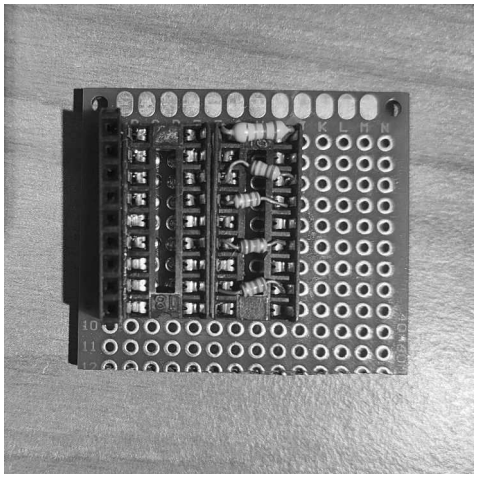
A organização do circuito baseou-se na utilização de dois DIP *sockets* de 9 pinos e uma barra de pinos fêmea de 9 pinos. No canto inferior direito da placa, foi colocada a barra de pinos na vertical. Esta barra foi totalmente soldada na parte traseira, permitindo a ligação

de um dos pinos ao terminal de 3.3 V do microcontrolador através de um cabo jumper. Os restantes pinos da barra foram conectados às extremidades de alimentação dos sensores, fornecendo assim a tensão necessária para o seu funcionamento.

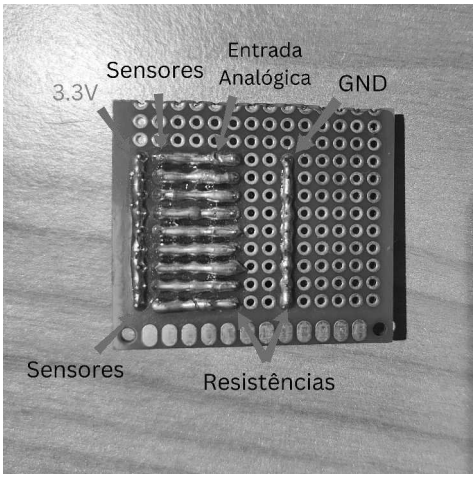
Imediatamente ao lado da barra de pinos, foram posicionados os dois DIP *sockets*, também na vertical, mas sem ligação direta à barra de pinos fêmea. Os pinos mais à esquerda do primeiro DIP *socket* foram todos interligados por soldadura na parte traseira da placa, permitindo ligar o terminal GND do microcontrolador a um dos pinos e garantir que todos os restantes fiquem igualmente conectados ao terra do sistema. Neste ponto, foram também inseridas as resistências necessárias à implementação dos divisores de tensão, conectando uma das extremidades de cada resistência a estes pinos comuns de GND e a outra extremidade aos pinos correspondentes do lado direito do mesmo DIP *socket*.

Os pinos mais à direita do primeiro DIP *socket* foram individualmente ligados aos pinos mais à esquerda do segundo DIP *socket*. Estes últimos, por sua vez, foram interligados com os pinos mais à direita do segundo DIP *socket*. Esta configuração permitiu criar os circuitos necessários à implementação de divisores de tensão, tal como descrito na Secção 4.4.2. Assim, os pinos mais à esquerda do primeiro DIP *socket* correspondem às saídas de sinal dos divisores e foram conectados às entradas analógicas do microcontrolador. Já os pinos mais à direita do segundo DIP *socket* foram utilizados para ligar as extremidades livres dos sensores de contacto.

Na Figura 4.26a encontra-se representada a vista frontal do PCB, enquanto que a Figura 4.26b mostra a vista traseira da placa com indicações adicionais que facilitam a compreensão das ligações realizadas.



(a) Vista frontal do PCB desenvolvido, com a disposição dos DIP sockets e da barra de pinos fêmea



(b) Vista traseira do PCB com destaque para as soldaduras e interligações entre os componentes

Figura 4.26: Vista frontal e traseira do PCB desenvolvido para melhor compreensão das ligações realizadas.

4.4.3 Programação do Microcontrolador

A leitura dos sensores FSR foi realizada através de um código desenvolvido na plataforma Arduino IDE, recorrendo à linguagem C/C++. O objetivo principal deste código é efetuar a leitura contínua das entradas analógicas a que os sensores se encontram ligados e calcular a correspondente tensão para posterior transmissão via porta série.

Para tal, os cinco sensores FSR foram conectados a diferentes entradas analógicas do microcontrolador, sendo estas definidas no início do código com as constantes `fsrPin1` a `fsrPin5`, associadas respetivamente aos pinos `A9`, `A8`, `A7`, `A6` e `A5`.

O código implementado encontra-se organizado em duas partes principais: a função `setup()`, onde é inicializada a comunicação série a uma taxa de 115200 bauds, e a função `loop()`, que corre continuamente durante a execução do programa. Dentro da função `loop()`, são efetuadas as leituras dos sensores através da função `analogRead()`, que retorna um valor inteiro proporcional à tensão medida em cada entrada analógica.

Os valores obtidos são depois convertidos em tensão (em volts), utilizando a fórmula presente em 4.5:

$$V = \frac{\text{valor\_lido} \times 3.3}{1023} \tag{4.5}$$

Neste contexto, `valor_lido` representa o valor devolvido pela função `analogRead()`, que no caso do Teensy 4.0, cuja resolução do conversor analógico-digital (ADC) é de 10 bits, pode variar entre 0 e 1023. Esta gama de valores decorre da capacidade do ADC representar $2^{10} = 1024$ níveis discretos de tensão. Assim, o valor 0 corresponde a uma tensão de 0 V, e o valor 1023 corresponde a 3.3V, que é a tensão de referência do microcontrolador.

A seguir à conversão, os cinco valores de tensão são enviados através da porta série, sendo apresentados numa única linha e separados por tabulações, facilitando a sua leitura e processamento posterior. A utilização da função `Serial.println()` no último valor garante a quebra de linha entre conjuntos sucessivos de amostras. Foi ainda introduzido um pequeno `delay` de 1ms entre ciclos para evitar sobrecarga na transmissão dos dados.

4.4.4 Testes com sensores

Até ao momento, tinha sido realizado apenas um teste preliminar com um único sensor. Nesta subseção, apresentam-se testes adicionais realizados para avaliar de forma mais abrangente o desempenho dos sensores incorporados no sistema.

O primeiro teste consistiu em ligar quatro sensores FSR ao microcontrolador em simultâneo. Foram aplicadas forças individualmente em cada um dos sensores, de forma a analisar a resposta de cada sensor de maneira isolada. De seguida, aplicaram-se forças em mais do que um sensor ao mesmo tempo, permitindo avaliar o comportamento do sistema quando várias entradas são acionadas simultaneamente, como acontece durante o funcionamento típico da mão robótica. Esta etapa foi essencial para compreender a capacidade do sistema em distinguir forças aplicadas em diferentes pontos ou limitações na leitura dos sinais.

Adicionalmente, foi realizada uma experiência específica utilizando o sensor FSR 406, com o objetivo de aprofundar a análise do seu comportamento. Este sensor, devido à sua maior

área de detecção, apresenta características distintas em relação aos sensores FSR 400, sendo, portanto, fundamental compreender o seu desempenho em condições práticas.

Para integrar os dados dos sensores no sistema robótico baseado em ROS 2, foi desenvolvido um nó denominado `fsr_sensor_node`. Este nó estabelece uma ligação com o microcontrolador via comunicação série, lê continuamente os dados provenientes dos cinco sensores e publica-os no tópico `fsr_values` . Para além disso, os valores adquiridos são gravados num ficheiro CSV, com o respetivo carimbo temporal, permitindo a posterior análise offline.

Por fim, é descrito o desenvolvimento de uma peça de interface que foi projetada com o objetivo de propagar a força exercida na ponta do dedo para o sensor. Esta peça, posicionada entre a superfície externa da falange e o sensor FSR, permite uma distribuição mais homogénea da força e contribui para melhorar a fiabilidade das medições.

Testes iniciais com sensores FSR

Para a realização dos testes iniciais com os sensores de contacto, foi preparada uma montagem experimental composta por quatro sensores a operar em simultâneo: três sensores FSR 400 e um sensor FSR 406. O principal objetivo destes testes foi avaliar o comportamento individual de cada sensor, assim como o desempenho do sistema quando várias forças são aplicadas de forma simultânea.

Na primeira experiência foi aplicada força apenas sobre um dos sensores, de forma a analisar a sua resposta individual e a verificar se a ativação desse sensor poderia influenciar as leituras dos restantes. Os gráficos obtidos para esta experiência encontram-se apresentados nas Figuras 4.27.

A análise dos resultados permitiu concluir que os sensores respondem de forma adequada à aplicação de carga, apresentando um aumento claro da tensão de saída quando submetidos a força. Simultaneamente, verificou-se que os sensores não acionados mantiveram os seus sinais estáveis ao longo de toda a experiência, confirmando que a aplicação de carga num sensor não interfere com o comportamento dos restantes. Foi ainda registada uma ligeira oscilação de aproximadamente 0,01 V na ausência de carga, o que indica uma variação residual no sinal. No entanto, tendo em conta a sua amplitude reduzida, considerou-se que esta flutuação não compromete o desempenho do sistema, pelo que não foi aplicada qualquer filtragem adicional.

Na segunda experiência, aplicaram-se forças em vários sensores em simultâneo, com o objetivo de estudar o comportamento do sistema em situações de utilização mais representativas das condições de operação da mão robótica. Os gráficos resultantes desta experiência encontram-se apresentados nas Figuras 4.28.

A análise dos resultados confirma as conclusões retiradas da primeira experiência de que os sensores apresentam um comportamento independente, sem sinais de interferência mútua. Além disso, foi possível observar que o sistema responde de forma satisfatória à aplicação de força, validando o desempenho adequado dos sensores nas condições de funcionamento previstas.

Estes testes permitiram, assim, validar o correto funcionamento do sistema de sensores, tanto em situações de carga individual como de carga múltipla, constituindo uma etapa

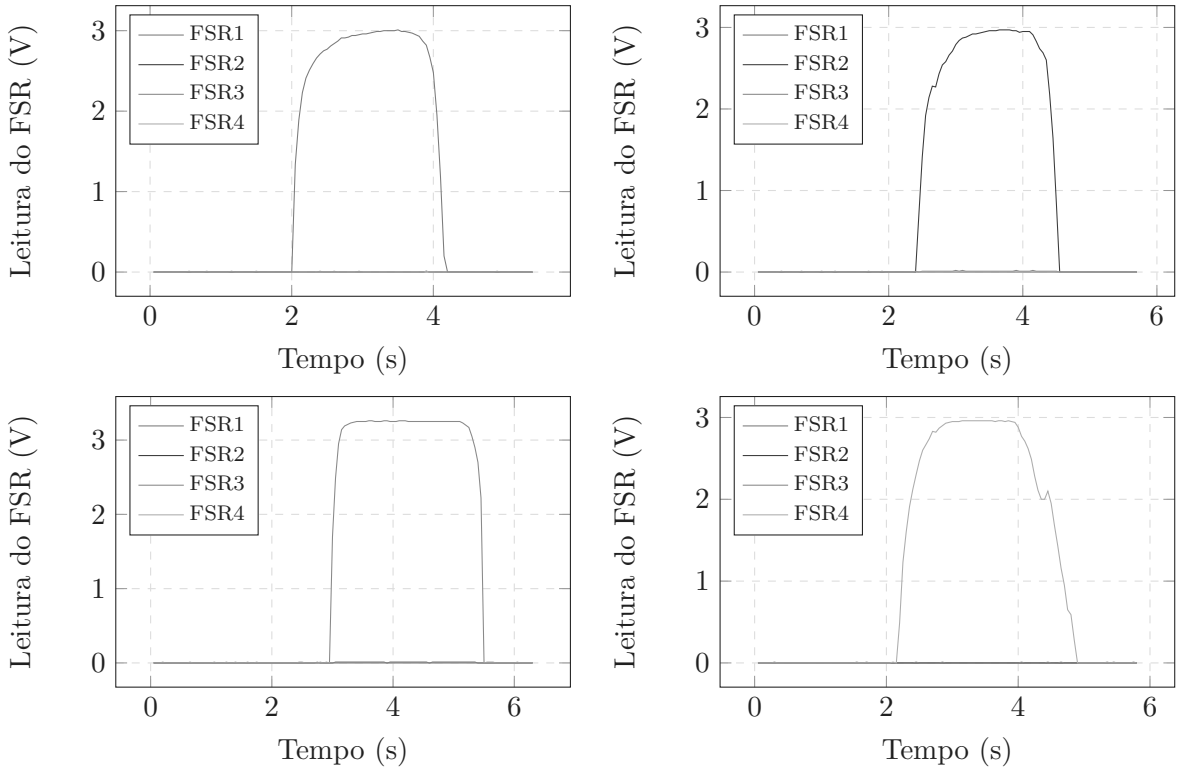


Figura 4.27: Gráfico da tensão de saída dos sensores durante a aplicação de força apenas sobre um dos sensores.

fundamental para a integração dos sensores na estrutura da mão robótica. Adicionalmente, verificou-se que a frequência de leitura dos valores provenientes dos sensores FSR é de aproximadamente 1000 Hz, valor que garante uma amostragem suficientemente elevada para aplicações em tempo real e deteção de contactos rápidos.

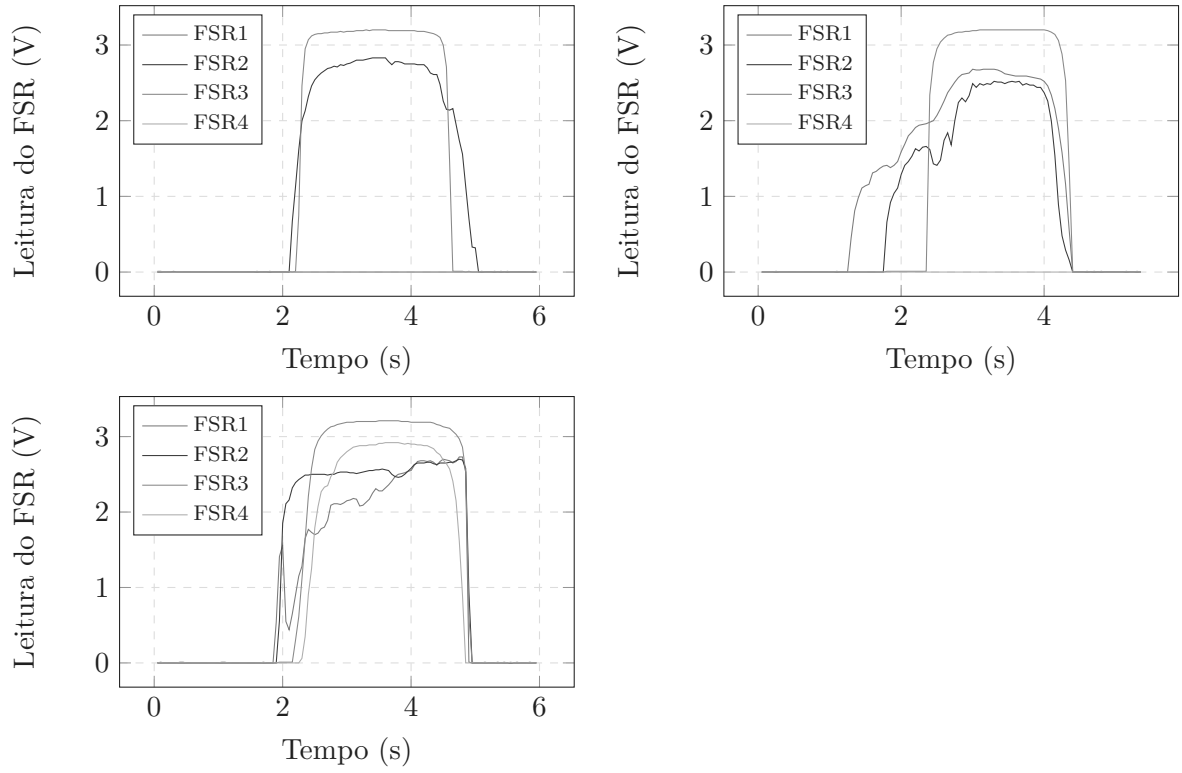


Figura 4.28: Gráfico da tensão de saída dos sensores durante a aplicação de força sobre 2, 3 e 4 sensores em simultâneo.

Teste com sensor FSR 406

O sensor FSR 406 possui uma geometria quadrada e uma área de deteção consideravelmente maior do que os sensores FSR 400, o que o torna particularmente interessante para aplicações onde se pretende medir forças aplicadas de forma mais distribuída.

De forma a avaliar o comportamento específico deste sensor, nomeadamente a sua capacidade de resposta à aplicação de cargas em diferentes zonas da superfície sensível, realizou-se outra experiência onde foi inicialmente aplicada uma carga sobre o sensor, colocando-se um objeto de massa conhecida numa região específica da sua área ativa. Após esta primeira aplicação, procedeu-se à aplicação de uma força, posicionada noutro ponto distinto do sensor, de forma a analisar a resposta do sistema à distribuição de forças em múltiplas zonas.

O gráfico obtido para esta experiência encontra-se apresentado na Figura 4.29 e a sua análise permite observar que aquando da aplicação da primeira carga, o sinal de saída apresenta alguma oscilação, possivelmente associada à instabilidade do objeto colocado, dado o seu peso reduzido, ou à falta de fixação adequada do sensor à superfície de teste. Este comportamento evidencia a sensibilidade do sensor a fatores externos, como a estabilidade da carga e as condições de montagem.

Após a aplicação da segunda carga, observou-se um aumento adicional na tensão de saída, indicando que o sinal final reflete a soma das forças aplicadas em diferentes pontos da superfície sensível do sensor. Estes resultados demonstram a capacidade do sensor FSR 406 para detetar forças em múltiplas zonas da sua área ativa.

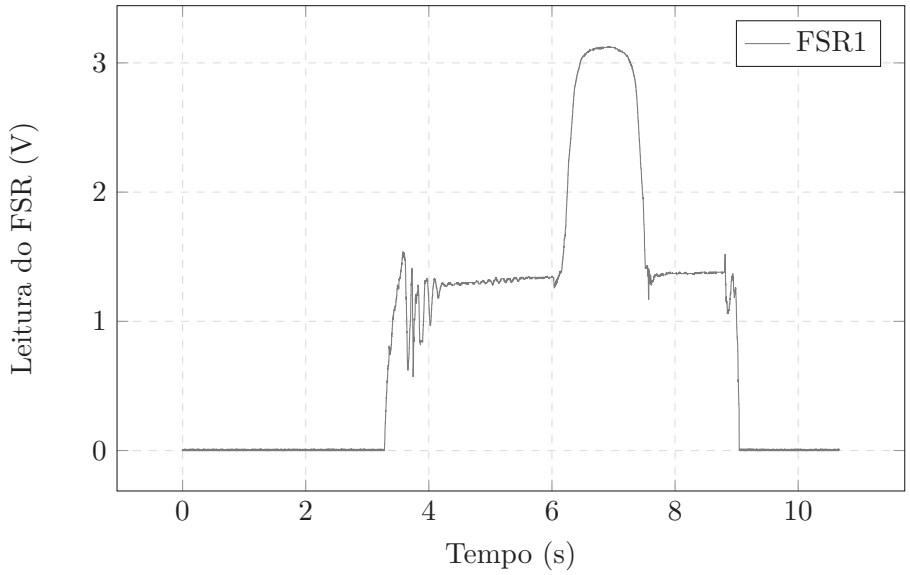


Figura 4.29: Gráfico da tensão de saída do sensor FSR 406 durante a aplicação de duas cargas em pontos distintos da sua superfície sensível.

Aumento da Zona Sensível do Sensor

Os testes preliminares realizados em bancada permitiram avaliar o comportamento dos FSR em condições ideais. No entanto, quando se integraram na estrutura da mão robótica, verificou-se que os sensores raramente registavam valores significativos durante os testes de contacto. Este fenómeno deve-se principalmente à discrepância entre o tamanho reduzido e a geometria plana dos sensores FSR e a forma volumosa e irregular das pontas dos dedos da *LEAP Hand* [13]. Assim, a força exercida pelos objetos nem sempre é aplicada diretamente sobre o sensor, comprometendo a transferência eficaz da força.

Para ultrapassar o problema identificado, foi desenhada, com recurso à ferramenta Tinkercad, uma pequena peça em 3D composta por uma base com espessura de 0.5 mm que replica aproximadamente a forma da superfície da ponta do dedo. No centro desta base foi adicionada uma saliência com geometria semiesférica, com cerca de 1.5 mm de altura e 8 mm de diâmetro, valor próximo do diâmetro útil do sensor FSR. Esta semiesfera é orientada de forma a ficar em contacto direto com o sensor, permitindo concentrar e propagar a força proveniente de diferentes pontos da superfície do dedo para a zona ativa do sensor. Na Figura 4.30 é possível observar o esquema de aplicação da peça desenvolvida para melhor compreensão da utilidade da mesma.

A utilização desta peça garante uma melhor transferência de força, mesmo quando o objeto interage com áreas da ponta do dedo que não se encontram diretamente sobre o sensor. Na Figura 4.31 é possível visualizar a peça desenhada. A eficácia desta solução será discutida no capítulo seguinte, com base nos resultados obtidos durante os testes experimentais.

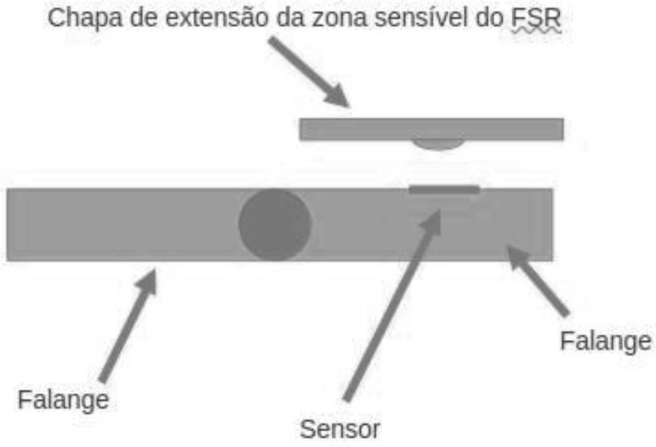


Figura 4.30: Esquema de utilização da peça desenvolvida para aumento da zona sensível do sensor.

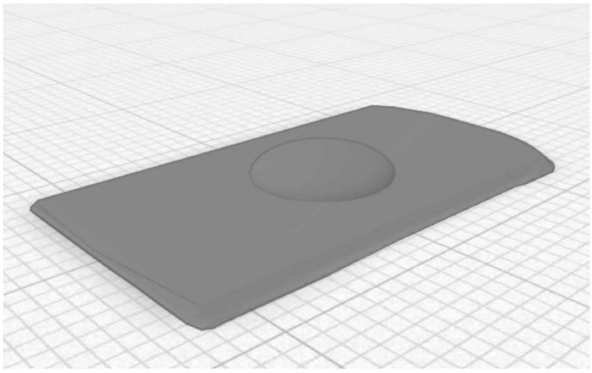


Figura 4.31: Peça desenhada em Tinkercad para propagação da força até ao sensor FSR.

4.5 Conclusão

Este capítulo apresentou de forma sistemática o processo de desenvolvimento, montagem e programação da mão robótica, bem como a sua integração com um sistema sensorial de contacto.

Na primeira parte abordou-se a construção da estrutura física da mão robótica, incluindo o fabrico, montagem e integração dos diferentes componentes. Foram configurados os motores *Dynamixel XC330-M288-T*, com destaque para a seleção dos modos de operação e parâmetros que asseguram um controlo eficiente e compatível com a cinemática da mão.

Seguidamente, desenvolveu-se a programação necessária para o controlo dos movimentos, iniciando-se pela ativação de um único motor, passando pelo controlo completo de um dedo, até à coordenação dos quatro dedos da mão. Este processo resultou na conceção de uma arquitetura de software modular e escalável, que suporta o controlo coordenado da mão e funcionalidades avançadas, como a definição de velocidades relativas entre dedos e falanges, bem como a introdução de atrasos temporais entre movimentos, possibilitando gestos mais fluidos e semelhantes aos de uma mão humana.

Na fase final do capítulo, detalhou-se a implementação do sistema sensorial baseado em

sensores FSR, começando pela escolha do microcontrolador e das resistências adequadas até ao desenvolvimento físico do sistema, com a criação de um PCB para facilitar as ligações e a alimentação. Foram também desenhadas peças em 3D para fixar os componentes eletrónicos na palma da mão, bem como uma peça adicional que permite propagar a força exercida sobre os dedos até ao sensor, tendo em conta a geometria da estrutura.

A nível de software, foi implementada a leitura dos sinais analógicos dos sensores e sua transmissão via comunicação série, utilizando o microcontrolador Teensy 4.0, cuja frequência de operação até 600 MHz permitiu atingir uma elevada taxa de amostragem. Os testes realizados demonstraram que o sistema é capaz de adquirir dados dos sensores FSR a uma frequência aproximada de 1000 Hz, o que assegura uma perceção tátil responsiva e adequada a aplicações em tempo real. Paralelamente, avaliou-se também que os dados dos motores podem ser lidos a uma frequência de 50 Hz de forma estável, permitindo a monitorização contínua do estado da mão durante o seu funcionamento.

Em suma, este capítulo consolidou os principais elementos físicos e funcionais da mão robótica desenvolvida, dotando-a não só de capacidade de movimento articulado, mas também de perceção tátil básica. O sistema encontra-se, assim, preparado para fases posteriores de integração com estratégias de controlo sensório-motor e testes experimentais mais avançados.

CAPÍTULO 5

Classificação de Sobreposições de Dedos com Aprendizagem Automática

5.1 Introdução

O fecho da mão robótica para agarrar objetos pode, em muitos casos, ser realizado de forma uniforme, uma vez que a sua estrutura adaptativa permite uma conformação passiva às geometrias dos objetos manipulados. Contudo, no momento da libertação, surgem frequentemente situações em que os dedos se encontram sobrepostos ou em contacto entre si. Nestes casos, é essencial determinar a ordem mais adequada para a abertura dos dedos, de forma a evitar esforços excessivos nos motores e garantir movimentos suaves e controlados.

Tradicionalmente, a deteção de sobreposições entre componentes de sistemas robóticos é realizada através de métodos de cinemática direta. No entanto, em mãos robóticas antropomórficas, esta abordagem apresenta limitações relevantes. Estes sistemas incluem frequentemente componentes com geometrias irregulares e não modeladas como juntas convencionais, dificultando a aplicação de métodos cinemáticos clássicos. Em ambientes ROS é possível utilizar ficheiros Computer-Aided Design (CAD) ou modelos Unified Robot Description Format (URDF) para verificar a existência de sobreposições através da deteção de interseções de malhas. Apesar de eficazes, estes métodos requerem tempo de processamento considerável e poder computacional elevado, o que os torna pouco adequados para aplicações em tempo real ou em sistemas com recursos limitados.

Neste contexto, este capítulo propõe uma abordagem alternativa baseada em técnicas de aprendizagem automática. Serão treinados e comparados diferentes modelos de classificação, nomeadamente regressão logística, Support Vector Machine (SVM) e redes neuronais com o objetivo de identificar o tipo de sobreposição entre dedos. Estes modelos utilizarão como entrada os dados adquiridos diretamente dos motores e sensores integrados na mão robótica

desenvolvida neste projeto. Esta abordagem visa não só aumentar a eficiência e robustez da deteção de sobreposições, como também abrir caminho para a sua aplicação em tempo real no controlo da mão, promovendo interações mais seguras, inteligentes e adaptativas.

5.2 Aquisição de dados experimentais

Nesta secção descreve-se o processo de aquisição dos dados utilizados para o treino dos modelos de deteção de sobreposições. Inicialmente, apresenta-se o tipo de dados recolhidos a partir dos motores e sensores da mão robótica, detalhando os parâmetros monitorizados. Em seguida, são explicados os procedimentos experimentais adotados durante a recolha, incluindo os cenários de teste e as condições de sobreposição simuladas. Por fim, é apresentado o *dataset* obtido, contendo as diferentes classes de sobreposição e respetivos dados associados.

5.2.1 Descrição dos dados adquiridos

Os motores Dynamixel disponibilizam uma vasta gama de dados em tempo real, incluindo posição, velocidade, corrente, tensão, temperatura, entre outros parâmetros internos. No entanto, para a construção de um *dataset* adequado ao treino de uma rede neuronal, é fundamental selecionar cuidadosamente quais os dados a recolher. Utilizar todos os parâmetros disponíveis pode não só introduzir redundância como também aumentar significativamente a dimensionalidade do problema, o que pode levar a *overfitting*, aumento do tempo de treino, necessidade de maior capacidade computacional e dificuldades na generalização do modelo.

Assim, nesta fase, foi efetuada uma seleção dos parâmetros mais relevantes para a tarefa de deteção de sobreposições entre os dedos da mão robótica. Foram recolhidas as posições angulares de todos os motores, por representarem diretamente o estado postural da mão, bem como as correntes consumidas, uma vez que aumentos súbitos ou valores anómalos de corrente podem indicar contacto físico ou resistência ao movimento.

Adicionalmente, foram incluídos os valores provenientes dos sensores de contacto integrados na mão robótica desenvolvida neste projeto, por fornecerem uma informação complementar crucial sobre a interação física entre os dedos ou com o ambiente. Estes dados sensoriais são particularmente úteis para confirmar a presença de contacto e, em conjunto com os dados dos motores, fornecer uma caracterização mais robusta da interseção.

A combinação destas três fontes de informação foi considerada suficiente para capturar os elementos mais relevantes à deteção e classificação das sobreposições, garantindo um bom equilíbrio entre riqueza informativa e complexidade computacional.

5.2.2 Definição das Classes de Sobreposição

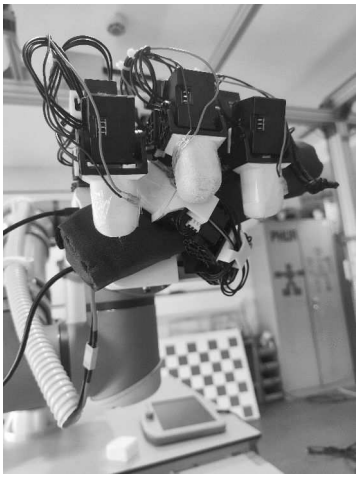
Para que um modelo de aprendizagem automática consiga distinguir diferentes situações de contacto entre os dedos da mão robótica, é fundamental definir previamente as classes que representam os diferentes tipos de sobreposição a detetar. No presente trabalho foram consideradas três classes distintas, que cobrem os cenários mais relevantes observados durante a interação entre os dedos, nomeadamente entre o polegar e os restantes dedos:

- **Classe 0 – Sem sobreposição:** corresponde ao cenário em que os dedos da mão robótica executam os movimentos de fecho e abertura sem qualquer interferência física entre si. Esta classe representa o comportamento desejado em operações normais, como o agarrar e largar de objetos, sem sobreposição de dedos.
- **Classe 1 – Sobreposição com o polegar por baixo:** nesta classe ocorre contacto entre dedos, sendo o polegar o dedo que se encontra fisicamente por baixo de outro(s) dedo(s) durante a interseção.
- **Classe 2 – Sobreposição com o polegar por cima:** nesta situação também existe sobreposição entre dedos, mas com o polegar localizado por cima do(s) dedo(s) com o(s) qual/quais colide.

Estas classes foram escolhidas por refletirem casos típicos e relevantes para o controlo seguro e eficiente da mão robótica antropomórfica. A sua correta identificação permite tomar decisões informadas durante a execução de tarefas, como ajustar a ordem de abertura dos dedos para evitar esforços desnecessários ou interseções indesejadas.

As situações descritas para cada classe podem ser observadas na Figura 5.1. A Figura 5.1a ilustra um exemplo da Classe 0, onde não existe qualquer sobreposição entre os dedos. A Figura 5.1b mostra a situação correspondente à Classe 1, na qual o polegar se encontra por baixo de outro dedo. Por fim, a Figura 5.1c apresenta a Classe 2, caracterizada pela sobreposição do polegar por cima dos restantes dedos.

(a) Exemplo de situação para a classe 0



(b) Exemplo de situação para a classe 1

(c) Exemplo de situação para a classe 2

Figura 5.1: Situações de utilização da mão para as diferentes classes de sobreposições dos dedos descritas

5.2.3 Procedimentos de Aquisição

Para a aquisição dos dados utilizados no treino da rede neuronal, partiu-se da funcionalidade de ajuste dinâmico de corrente no contacto com objetos, anteriormente descrita na Secção 4.3.5. Esta funcionalidade permite detetar, em tempo real, quando um dedo entra em contacto com um objeto através do aumento da corrente consumida pelos motores e da diminuição da sua velocidade.

A partir desta base, foi desenvolvida uma estratégia para automatizar o processo de recolha de dados. O nó `set_positions` foi adaptado para permitir a execução automática de movimentos específicos da mão, seguidos de um curto período de espera e posterior reabertura. Este nó aceita, via terminal, os comandos 0, 1 ou 2, cada um associado a uma classe distinta de movimento:

- **0**: a mão fecha e abre com velocidades iguais em todos os dedos;
- **1**: o polegar inicia o fecho primeiro e é o último a abrir;
- **2**: o polegar é o último a fechar e o primeiro a abrir.

Quando um destes comandos é introduzido, o nó `set_positions` executa automaticamente o movimento correspondente, aguardando 2 segundos após o fecho para iniciar a abertura. Simultaneamente, este nó publica a classe correspondente num novo tópico, ao qual o nó `collect_data` está subscrito. Ao receber esta informação, o nó `collect_data` atualiza automaticamente a variável `self.my_class`, associando a classe correta aos dados que venham a ser recolhidos durante essa interação.

Além disso, sempre que os motores de um dedo publicam uma mensagem com os novos valores de corrente limite no nó `set_currents`, o código publica também uma mensagem no tópico `state`. Esta mensagem binária indica o estado da mão: **0** quando nenhum dedo está em contacto (sem esforço e sem ajuste de corrente), e **1** quando pelo menos um dedo está em carga e com a corrente ajustada.

O nó `collect_data` atua com base neste sinal. Está subscrito ao tópico `state` e só entra em funcionamento quando o valor é 1. Nessa situação, guarda automaticamente os valores das posições e correntes de todos os motores, bem como os dados obtidos pelos sensores de contacto da mão robótica. Quando o valor é 0, o nó permanece inativo.

Este sistema revelou-se particularmente útil, pois permite automatizar e otimizar o processo de aquisição de dados, tornando possível recolher vários conjuntos de dados dentro da mesma experiência, sempre que ocorrer um contacto relevante. O diagrama da Figura 5.2 ilustra a estratégia desenvolvida para automatizar a recolha de dados, facilitando a compreensão das condições que ativam ou inibem o armazenamento da informação.

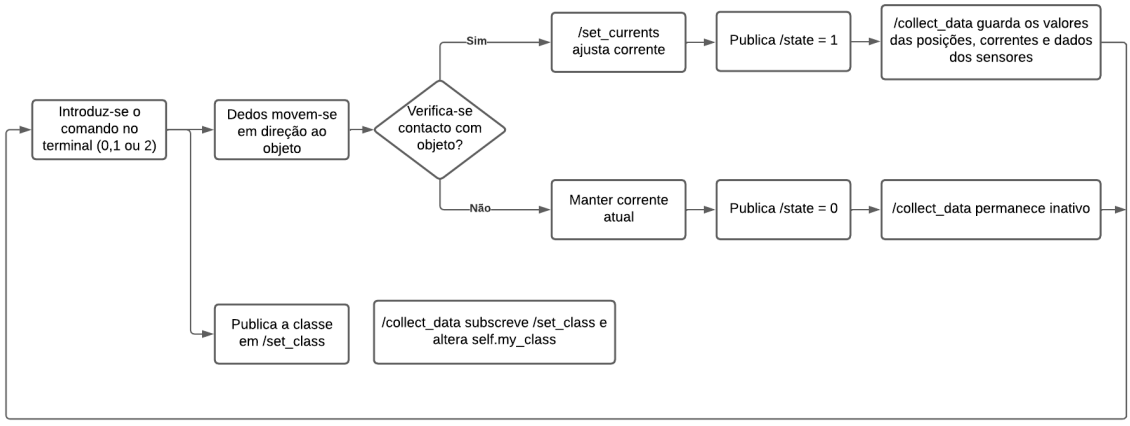
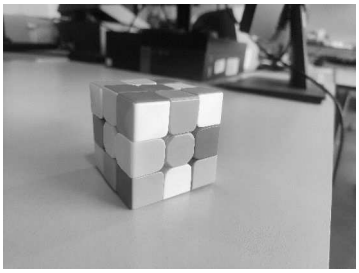


Figura 5.2: Fluxograma da estratégia de automatização da recolha de dados.

Para a construção do *dataset*, foram utilizados três objetos com diferentes formas e propriedades físicas: uma bola (Figura 5.3a), um cubo de *Rubik* (Figura 5.3b) e um estojo com geometria aproximadamente cilíndrica (Figura 5.3c). Cada experiência de contacto foi realizada em oito posições distintas, correspondentes aos oito octantes do espaço tridimensional, de forma a representar diferentes configurações de interação entre os dedos e os objetos, considerando também os efeitos da gravidade. Para garantir a variabilidade e robustez dos dados, cada experiência foi repetida dez vezes por objeto e por posição, permitindo obter um conjunto diversificado e representativo de situações reais de utilização.

(a) Bola utilizada na aquisição de dados



(b) Cubo utilizado na aquisição de dados

(c) Estojo utilizado na aquisição de dados

Figura 5.3: Objetos utilizados na aquisição de dados.

Esta abordagem permitiu recolher dados de forma sistemática e eficiente, facilitando a criação de um *dataset* adequado para o treino de modelos de deteção de sobreposições entre dedos.

5.2.4 *Dataset* adquirido

A partir do processo de aquisição automatizado desenvolvido e descrito anteriormente, e após a realização das experiências com todas as combinações de posições e número de repetições indicados na Subsecção 5.2.3, foi possível construir um *dataset* composto por um total de 6234 amostras, todas distintas entre si. Cada entrada no *dataset* corresponde a uma configuração única de interação entre a mão robótica e os objetos utilizados, refletindo variações nas posições relativas, forças de contacto e estados dos motores, o que resulta numa base de dados diversificada e representativa de diferentes cenários de sobreposição entre dedos.

Adicionalmente, os vídeos exemplificativos das aquisições realizadas para cada um dos objetos testados podem ser consultados no Apêndice A.4, permitindo visualizar o contexto físico associado a cada tipo de interação registada.

5.3 Desenvolvimento e Avaliação de Modelos de Deteção de Sobreposições

Nesta secção é apresentado o processo de desenvolvimento e avaliação de modelos de aprendizagem automática aplicados à deteção de sobreposições entre os dedos da mão robótica. O principal objetivo consiste em identificar automaticamente a classe de sobreposição em que a mão se encontra, com base nos dados adquiridos.

A secção está organizada em duas partes principais. Na primeira descreve-se o processo de preparação dos dados, incluindo as matrizes de correlação entre os dados. Na segunda parte são apresentados os diferentes algoritmos de classificação utilizados, nomeadamente a Regressão Logística, SVM e uma Rede Neuronal.

As implementações dos modelos de Regressão Logística e SVM foram realizadas com recurso à biblioteca *Scikit-learn*, amplamente utilizada para tarefas de aprendizagem supervisionada. A Rede Neuronal foi desenvolvida com a biblioteca *TensorFlow*, que permite maior flexibilidade na definição e treino de arquiteturas neuronais. A comparação dos desempenhos destes modelos visa selecionar a abordagem mais adequada para integrar no sistema de controlo da mão robótica, contribuindo para uma atuação mais inteligente, segura e adaptativa.

5.3.1 Preparação dos dados

Na aplicação de modelos de aprendizagem automática, uma etapa essencial consiste na análise preliminar do *dataset*, com o objetivo de compreender a natureza das variáveis, identificar possíveis desequilíbrios entre classes e detetar padrões ou correlações relevantes. Esta análise permite não apenas melhorar o desempenho dos modelos, mas também garantir uma interpretação mais robusta dos seus resultados.

Um dos primeiros passos realizados consiste na verificação do balanceamento das classes. Para tal, foi construído um gráfico circular, representado na Figura 5.4, que permite visualizar a distribuição das amostras por classe.

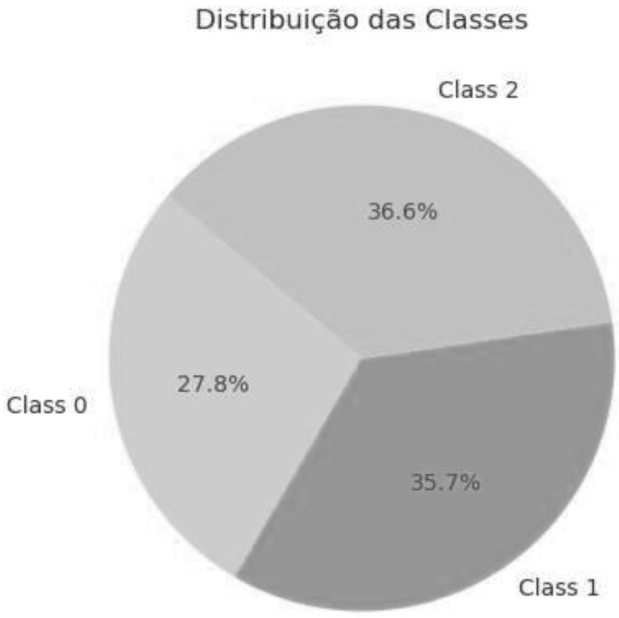


Figura 5.4: Gráfico com a distribuição das classes do *dataset*.

Através da análise do gráfico observa-se que aproximadamente 27.8% dos dados pertencem à Classe 0 (sem sobreposição), 35.7% à Classe 1 (polegar por baixo) e 36.6% à Classe 2

(polegar por cima). Esta distribuição revela que o *dataset* está relativamente bem balanceado, o que é um fator importante para evitar possíveis falhas durante o treino dos modelos.

Outro aspeto crucial na preparação dos dados consistiu na análise da correlação entre as *features* e a variável de classe. Esta análise tem como objetivo principal identificar que variáveis estão mais associadas a cada uma das classes, podendo assim contribuir para a seleção de *features* relevantes e para uma melhor compreensão dos padrões existentes nos dados.

Para este efeito, foi utilizado o coeficiente de correlação de Pearson, que avalia a intensidade e direção da relação linear entre duas variáveis numéricas cuja expressão é apresentada na Equação 5.1:

$$r_{X,Y} = \frac{\sum_{i=1}^{n}(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \bar{X})^2} \cdot \sqrt{\sum_{i=1}^{n}(Y_i - \bar{Y})^2}} \tag{5.1}$$

onde:

- X_i e Y_i são os valores das variáveis X e Y na i-ésima observação;
- $\bar{X}$ e $\bar{Y}$ são as médias das variáveis X e Y;
- n é o número total de observações.

Este coeficiente assume valores entre -1 e 1, sendo que valores próximos de 1 indicam uma forte correlação positiva, valores próximos de -1 indicam uma forte correlação negativa, e valores próximos de 0 indicam ausência de correlação linear.

No entanto, como a variável de classe é categórica, contendo três categorias distintas, não é possível calcular diretamente a correlação de Pearson com as *features* numéricas. Para contornar esta limitação, aplicou-se a técnica de *one-hot encoding*, que consiste em transformar uma variável categórica em múltiplas variáveis binárias. No caso específico deste trabalho, a variável de classe foi convertida em três novas colunas binárias, cada uma representando uma das classes possíveis. Cada coluna assume o valor 1 se a amostra pertence à classe correspondente, e 0 caso contrário.

Dado que o *dataset* contém 37 variáveis numéricas, incluindo posições dos motores, valores de corrente e leituras dos sensores de força, a representação de todas as correlações numa única matriz resultaria numa visualização demasiado densa e de difícil interpretação. Por este motivo, optou-se por apresentar os resultados em três matrizes de correlação distintas:

- A Figura 5.5 apresenta a matriz de correlação referente às posições dos motores;
- A Figura 5.6 mostra a matriz de correlação para os valores de corrente;
- A Figura 5.7 exibe a matriz de correlação correspondente às leituras dos sensores FSR.

Em cada uma destas matrizes, as três últimas linhas correspondem às variáveis geradas pelo *one-hot encoding* da classe e são de particular importância. Estas linhas mostram, para cada *feature*, o grau de correlação com cada uma das classes, permitindo identificar as variáveis mais relevantes para distinguir entre as diferentes classes. Esta análise é especialmente útil para orientar a seleção de *features* e melhorar o desempenho dos modelos de classificação desenvolvidos posteriormente caso os resultados obtidos com os modelos iniciais não sejam satisfatórios

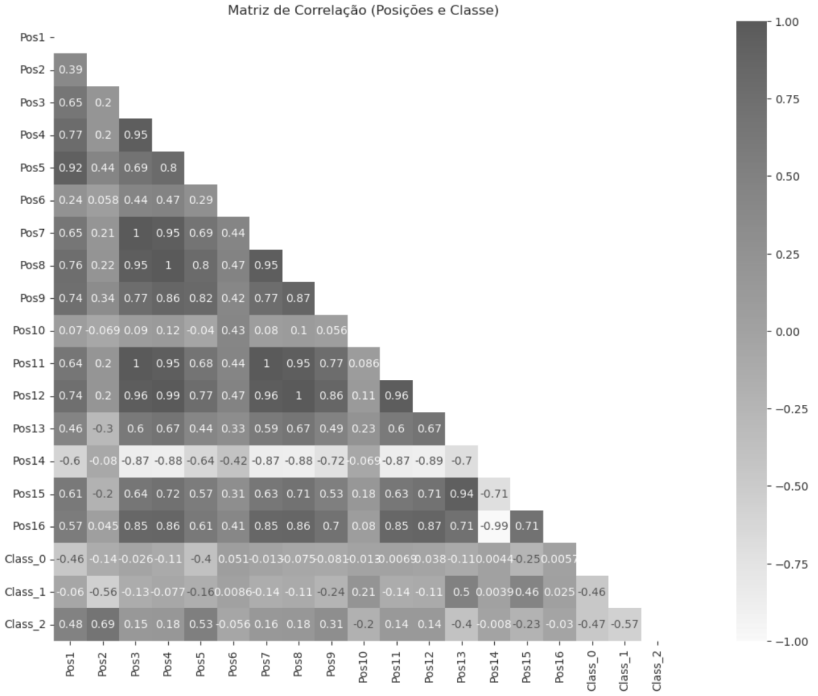


Figura 5.5: Matriz de correlação obtida para as posições e classes

A partir da análise da matriz de correlação das posições dos motores (Figura 5.5) é possível verificar que as variáveis com maior correlação com a **classe 0** são as **Posições 1, 5 e 15**, correspondentes, respetivamente, ao primeiro motor do *dedo indicador* (*index finger*), ao primeiro motor do *dedo médio* (*middle finger*) e ao terceiro motor do *polegar* (*thumb*). Para a **classe 1**, destacam-se as **Posições 2, 13 e 15**, que representam o segundo motor do *dedo indicador*, o primeiro motor do *polegar* e novamente o terceiro motor do *polegar*, sugerindo uma forte associação entre esta classe e os movimentos deste dedo. No caso da **classe 2**, as posições com maior correlação são as **Posições 1, 2, 5 e 9**, relativas ao primeiro e segundo motores do *dedo indicador*, ao primeiro motor do *dedo médio* e ao primeiro motor do *dedo anelar* (*ring finger*). Estes resultados evidenciam que o *dedo indicador* participa de forma relevante em todas as classes, o que é coerente do ponto de vista biomecânico, dado que este dedo desempenha frequentemente um papel central em tarefas manuais e de preensão por ser o mais próximo do polegar.

Relativamente à matriz de correlação das correntes dos motores (Figura 5.6) verifica-se que as variáveis com maior correlação com a **classe 0** são as **Correntes 3, 4, 8 e 11**, que correspondem, respetivamente, ao terceiro e quarto motores do *dedo indicador* (*index finger*), ao quarto motor do *dedo médio* (*middle finger*) e ao terceiro motor do *dedo anelar* (*ring finger*). Para a **classe 1**, as correntes mais correlacionadas são as **Correntes 2, 3 e 4**, que abrangem praticamente todo o *dedo indicador*, com exceção do primeiro motor, indicando que este dedo tem um papel predominante na execução dos gestos associados a esta classe. No que respeita à **classe 2**, destacam-se as **Correntes 2, 8 e 14**, que dizem respeito ao segundo motor do *dedo indicador*, ao último motor do *dedo médio* e ao segundo motor do *polegar* (*thumb*).

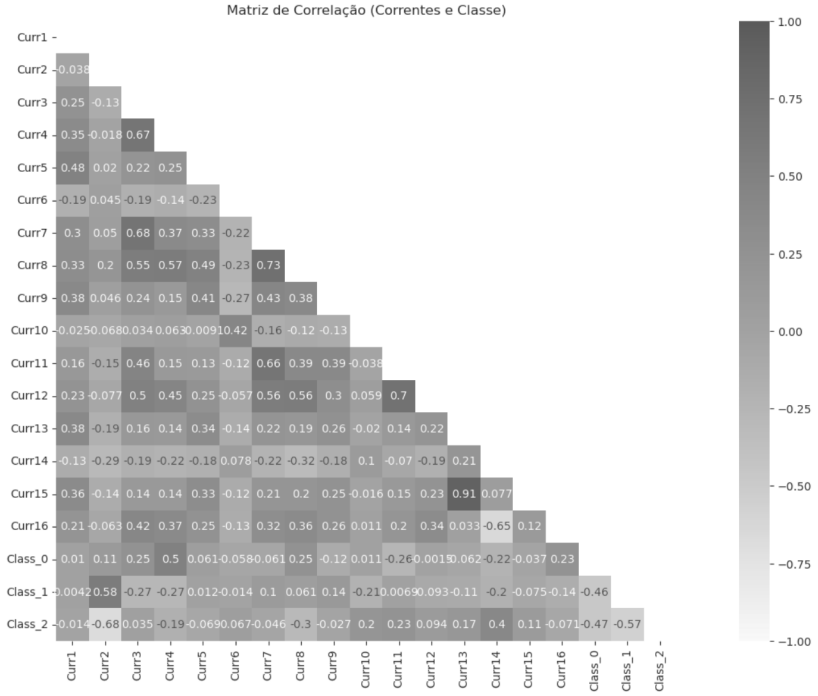


Figura 5.6: Matriz de correlação obtida para as correntes e classes

Por fim, na análise da matriz de correlação das leituras dos sensores (Figura 5.7) verifica-se que, para a **classe 0**, os sensores com maior correlação são os **Sensores 1, 3 e 5**, correspondentes, respetivamente, aos sensores posicionados no *dedo anelar* (*ring finger*), no *dedo indicador* (*index finger*) e na *palma da mão*. Para a **classe 1**, os sensores mais relevantes em termos de correlação são os **Sensores 3 e 4**, localizados no *dedo indicador* e no *polegar* (*thumb*), o que evidencia a importância do polegar na execução dos gestos associados a esta classe. Por fim, no que respeita à **classe 2**, os sensores mais correlacionados são os **Sensores 1 e 5**, ambos relacionados com o *dedo anelar* e a *palma da mão*, respetivamente.

Esta análise exploratória permite, portanto, obter uma compreensão mais profunda da influência de cada variável, podendo futuramente apoiar estratégias de seleção de *features* ou de interpretação do comportamento dos modelos treinados.

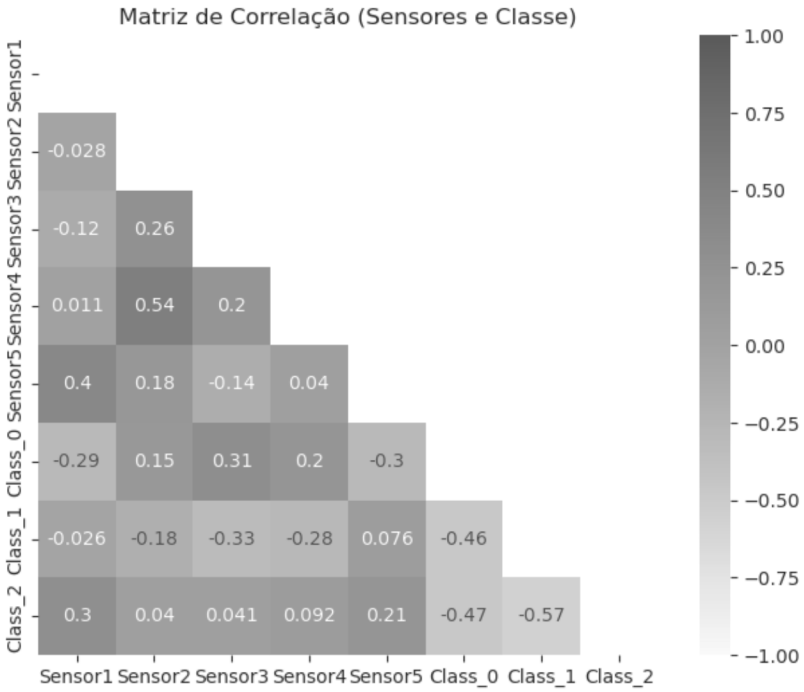


Figura 5.7: Matriz de correlação obtida para os sensores e classes

5.3.2 Métodos Testados

Após a análise e preparação do *dataset* adquirido, procede-se à implementação dos métodos de aprendizagem automática selecionados para o problema de deteção de sobreposições entre os dedos da mão robótica. Nesta subsecção descreve-se a aplicação de três abordagens distintas: Regressão Logística, SVM e uma Rede Neuronal. Para cada uma destas abordagens é apresentada uma breve descrição do seu funcionamento, bem como os principais parâmetros considerados na sua implementação.

Na fase de testes, o conjunto de dados é dividido em dois subconjuntos: 80% dos dados são utilizados para treino dos modelos e os restantes 20% para avaliação do seu desempenho. Esta divisão garante que a avaliação é realizada com dados que não foram previamente vistos pelos modelos, contribuindo para uma estimativa mais realista da sua capacidade de generalização.

Adicionalmente, recorreu-se à técnica de *cross-validation* com 10 *folds*, um método amplamente utilizado para avaliar a performance de modelos de forma mais robusta. Nesta técnica, o *dataset* de treino é dividido em 10 subconjuntos (ou *folds*) aproximadamente iguais. Em cada iteração, nove *folds* são usados para treinar o modelo e o *fold* restante é utilizado para validação. Este processo repete-se até que cada *fold* tenha sido usado como conjunto de validação uma vez e os resultados obtidos são então agregados. Esta abordagem permite reduzir a variância associada a uma única divisão dos dados e fornece uma avaliação mais estável do desempenho dos modelos [28]. Na figura 5.8 observa-se um esquema de funcionamento do *cross-validation*.

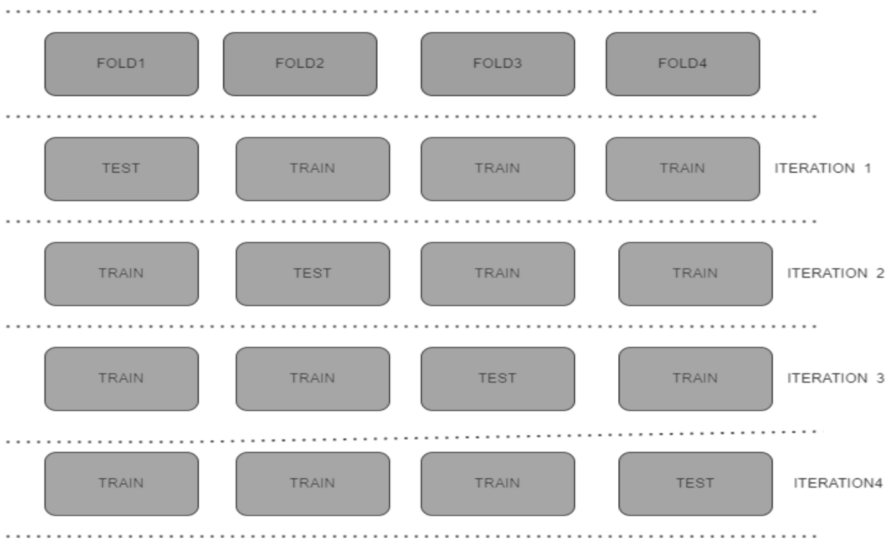


Figura 5.8: Esquema com a lógica de funcionamento do *cross-validation*, presente no trabalho de Meenu et al.[28].

Regressão Logística

A regressão logística é um modelo de classificação linear amplamente utilizado em tarefas supervisionadas, particularmente devido à sua simplicidade, interpretabilidade e desempenho competitivo em muitos contextos. Embora seja originalmente concebida para problemas binários, a regressão logística pode ser estendida para problemas multiclasse através de diferentes estratégias. Neste trabalho, foi utilizada a abordagem *multinomial*, também conhecida como regressão softmax, que modela diretamente a probabilidade de cada classe com base numa generalização da função sigmoide, a função softmax[29].

Na abordagem softmax, para um conjunto de classes $\{1, 2, ..., K\}$, a probabilidade predita de uma amostra $\mathbf{x}$ pertencer à classe k é dada pela expressão presente na Equação 5.2:

$$P(y = k \mid \mathbf{x}) = \frac{e^{\mathbf{w}_k^\top \mathbf{x}}}{\sum_{j=1}^{K} e^{\mathbf{w}_j^\top \mathbf{x}}} \tag{5.2}$$

Onde $\mathbf{w}_k$ é o vetor de pesos associado à classe k.

Nesta implementação recorreu-se à biblioteca `scikit-learn`[30], utilizando um `Pipeline` que inclui a normalização das variáveis de entrada através da técnica `StandardScaler`, uma etapa fundamental para garantir um bom desempenho da regressão logística, seguida pela aplicação do modelo de regressão logística com um número máximo de iterações definido em 5000 para assegurar a convergência. No bloco de código, 5.1, apresenta-se o código desenvolvido para inicializar o *pipeline* da regressão logística.

```
pipeline_logreg = Pipeline([
    ('scaler', StandardScaler()),
    ('logreg', LogisticRegression(max_iter=5000))
])
```

Listagem 5.1: Pipeline da Regressão Logística Multiclasse

De forma a encontrar os melhores hiperparâmetros, recorreu-se a um processo de *Grid Search*, uma técnica de otimização que testa exaustivamente combinações de parâmetros especificadas pelo utilizador. Esta procura foi realizada com validação cruzada de 10 *folds*, o que permite avaliar a robustez do modelo reduzindo o risco de *overfitting* e garantindo que cada classe está representada em todas as divisões do conjunto de treino.

Os parâmetros testados incluíram diferentes valores para o termo de regularização C, diferentes opções para ponderação de classes, penalização e algoritmos de otimização. Os valores testados encontram-se representados no excerto de código 5.2

```
param_grid = {
    'logreg__C': [0.001, 0.01, 0.1, 1, 10, 100],
    'logreg__class_weight': [None, 'balanced'],
    'logreg__penalty': ['l2'],
    'logreg__solver': ['lbfgs', 'newton-cg', 'saga'],
}
```

Listagem 5.2: Parâmetros utilizados no Grid Search para a Regressão Logística

Após a execução do processo de afinação, a melhor configuração foi selecionada com base na métrica de *accuracy* e os melhores hiperparâmetros encontrados foram:

- `C`: 100
- `class_weight`: `balanced`
- `penalty`: `l2`
- `solver`: `newton-cg`

O modelo foi então treinado no conjunto de treino, tendo convergido ao fim de 12 iterações.

Support Vector Machine

As SVM são algoritmos supervisionados amplamente utilizados para tarefas de classificação e regressão, sendo particularmente eficazes em problemas com elevada dimensionalidade e estruturas complexas de dados [31]. O seu princípio fundamental consiste na identificação do **hiperplano ótimo** que separa os dados de diferentes classes, maximizando a margem entre os pontos de dados mais próximos desse hiperplano, denominados *vetores de suporte*.

A ideia subjacente é que ao maximizar esta margem, o modelo se torna mais robusto a novas observações e menos suscetível ao sobreajuste. Nos casos em que os dados não são linearmente separáveis no espaço original, a SVM recorre ao uso de funções de transformação denominadas *kernels*, que projetam os dados para um espaço de maior dimensionalidade onde a separação linear é viável. Esta técnica, conhecida como *kernel trick*, permite à SVM resolver problemas não lineares sem aumentar explicitamente a complexidade computacional do modelo.

Apesar de a SVM ter sido originalmente concebida para problemas de classificação binária, é possível adaptá-la a problemas multiclasse através de abordagens como *one-vs-rest* (OvR), onde são treinados K classificadores binários (um por classe), sendo cada classificador responsável por distinguir uma classe contra todas as outras. Esta abordagem foi automaticamente utilizada na implementação com a biblioteca `scikit-learn`.

O funcionamento da SVM é controlado por um conjunto de hiperparâmetros que influenciam diretamente a sua capacidade de generalização:

- C: parâmetro de regularização que controla o equilíbrio entre a maximização da margem e a penalização de erros de classificação.
- γ: parâmetro utilizado nos *kernels* não lineares, como o Radial Basis Function (RBF), que determina a influência de cada ponto de treino.
- `kernel`: função de transformação usada para mapear os dados para um espaço de maior dimensionalidade. Os principais tipos de *kernel* disponíveis são:
  - `linear` – adequado quando os dados são aproximadamente linearmente separáveis.
  - `rbf` (Radial Basis Function) – indicado para separações complexas e não lineares.
  - `poly` – aplica uma transformação polinomial ao espaço de entrada.
  - `sigmoid` – semelhante às funções de ativação usadas em redes neuronais.
- `degree`: define o grau do polinómio utilizado no *kernel* `poly`. Quanto maior o grau, maior a complexidade do modelo.
- `coef0`: parâmetro que representa o termo independente nas funções `poly` e `sigmoid`, influenciando a forma da superfície de decisão.

A *pipeline* desenvolvida para aplicar a SVM encontra-se descrita no bloco de código 5.3. Numa primeira etapa, os dados são normalizados utilizando `StandardScaler`, dado que a SVM é sensível à escala das variáveis. De seguida, é aplicado o classificador `SVC` da `scikit-learn`, permitindo configurar diferentes *kernels* e hiperparâmetros.

```
pipeline_svm = Pipeline([
    ('scaler', StandardScaler()),
    ('svm', SVC())
])
```

Listagem 5.3: Pipeline da SVM com normalização

Para encontrar a combinação de hiperparâmetros mais adequada foi utilizada a técnica de *Grid Search* com validação cruzada de 10 *folds*, apresentada no código 5.4. Foram considerados diferentes *kernels* e valores para os parâmetros `C`, `gamma`, `degree` e `coef0`.

```
param_grid_svm = {
    'svm__kernel': ['linear', 'rbf', 'poly', 'sigmoid'],
    'svm__C': [0.01, 0.1, 1, 10, 100],
    'svm__gamma': ['scale', 'auto', 1, 0.1, 0.01, 0.001],
    'svm__degree': [2, 3, 4],
    'svm__coef0': [0.0, 0.5, 1.0]
}
```

Listagem 5.4: Parâmetros utilizados no Grid Search para a SVM

O critério de avaliação utilizado durante a otimização foi a `accuracy`. Após a execução da pesquisa exaustiva, os melhores hiperparâmetros encontrados foram:

- `C`: 100

- `kernel`: `rbf`
- `gamma`: 0.1
- `degree`: 2
- `coef0`: 0.0

Esta abordagem permitiu encontrar uma configuração adequada para a tarefa de deteção de sobreposição entre os dedos da mão robótica, assegurando uma elevada capacidade preditiva e uma boa generalização em validação cruzada.

Rede Neuronal

As redes neuronais artificiais são modelos computacionais inspirados no funcionamento do cérebro humano. São compostas por camadas de unidades interligadas, denominados neurónios artificiais, capazes de aprender padrões complexos a partir dos dados. Cada neurónio recebe um conjunto de entradas ponderadas, aplica uma função de ativação e transmite a sua saída para os neurónios da camada seguinte. O processo de aprendizagem realiza-se através de algoritmos de retropropagação do erro e técnicas de otimização, que ajustam iterativamente os pesos das ligações com base na discrepância entre a saída prevista e o valor real [32].

A arquitetura implementada neste trabalho corresponde a uma **rede neuronal totalmente conectada** (do inglês, *fully connected neural network*), na qual cada neurónio está ligado a todos os neurónios da camada anterior e da camada seguinte. Este tipo de arquitetura é particularmente adequado para tarefas de classificação com dados tabulares, como os utilizados neste projeto.

O modelo foi desenvolvido com recurso à biblioteca `TensorFlow`, recorrendo à interface de alto nível `Keras`. A estrutura da rede inclui:

- Uma camada oculta densa, com número variável de unidades, cuja função de ativação pode ser `ReLU` ou `tanh`, determinadas durante a otimização dos hiperparâmetros.
- Uma **camada de Dropout**, que desativa aleatoriamente uma fração das unidades durante o treino. Esta técnica, proposta por Srivastava et al. [33], tem como objetivo reduzir o *overfitting*, impedindo que o modelo dependa excessivamente de caminhos específicos e incentivando a aprendizagem de representações mais generalizáveis e robustas.
- Uma camada de saída com três neurónios (correspondentes às três classes alvo), utilizando a função de ativação `softmax`, que converte os valores de saída num vetor de probabilidades cuja soma é igual a 1, adequado para classificação multiclasse.

Para treinar a rede, foi utilizada a função de custo `sparse_categorical_crossentropy`, apropriada para problemas de classificação com rótulos inteiros. O otimizador escolhido foi o `Adam`, amplamente utilizado pela sua eficácia em cenários com muitos parâmetros e por adaptar dinamicamente a taxa de aprendizagem durante o treino.

A seleção dos hiperparâmetros foi realizada com a biblioteca `Optuna`, utilizando uma abordagem de otimização bayesiana para explorar de forma eficiente o espaço de busca. O processo considerou os seguintes hiperparâmetros:

- Número de unidades na camada oculta (`units`);

- Função de ativação (`activation`);
- Taxa de `dropout`;
- Taxa de aprendizagem (`learning_rate`);
- Tamanho do `batch`.

A avaliação de cada combinação de hiperparâmetros foi realizada através de validação cruzada estratificada com 10 *folds*, garantindo uma distribuição equilibrada das classes em cada subdivisão. Esta abordagem proporciona uma estimativa mais fiável da performance do modelo em dados não vistos e ajuda a mitigar o sobreajuste.

O código responsável por este processo está representado nos Blocos 5.5 e 5.6. No Bloco 5.5 apresenta-se a função de criação dinâmica do modelo, enquanto o Bloco 5.6 mostra a definição da função objetivo e a configuração do estudo com 120 tentativas de otimização.

```
def create_model(trial, input_dim):
    units = trial.suggest_int('units', 1, 71, step=5)
    activation = trial.suggest_categorical('activation', ['relu', 'tanh'])
    learning_rate = trial.suggest_float('learning_rate', 1e-4, 1e-1, log=True)
    dropout_rate = trial.suggest_float('dropout_rate', 0.0, 0.5, step=0.1)

    model = Sequential()
    model.add(Dense(units, activation=activation, input_shape=(input_dim,)))
    model.add(Dropout(dropout_rate))
    model.add(Dense(NUM_CLASSES, activation='softmax'))

    model.compile(
        optimizer=tf.keras.optimizers.Adam(learning_rate),
        loss='sparse_categorical_crossentropy',
        metrics=['accuracy']
    )
    return model
```

Listagem 5.5: Função de criação dinâmica do modelo com a ferramenta *Optuna*

Durante a otimização, a função objetivo executava a validação cruzada para cada tentativa (*trial*) e calculava a média das *accuracies* obtidas. O `Optuna` foi configurado para realizar 120 tentativas, explorando diferentes combinações de hiperparâmetros.

```
study = optuna.create_study(direction='maximize')
study.optimize(objective, n_trials=120, n_jobs=4)
```

Listagem 5.6: Criação do estudo Optuna

Após a conclusão do estudo, os melhores hiperparâmetros encontrados encontram-se apresentados na Tabela 5.1.

Com base nesses valores, o modelo final foi treinado utilizando 80% dos dados, aplicando validação cruzada com 10 *folds*, enquanto os 20% restantes foram reservados para teste.

Hiperparâmetro	Valor Ótimo
Número de neurónios (`units`)	*66*
Função de ativação (`activation`)	*relu*
Taxa de aprendizagem (`learning_rate`)	*0.002*
Taxa de `dropout`	*0.0*
Tamanho do batch (`batch_size`)	*16*

Tabela 5.1: Melhores hiperparâmetros encontrados para a rede neuronal

5.4 Resultados Obtidos

Esta secção apresenta os resultados obtidos com a aplicação dos modelos desenvolvidos para o problema em estudo. A análise está dividida em duas partes principais.

Na primeira parte realiza-se uma avaliação individual de cada modelo, Regressão Logística, SVM e Rede Neuronal, com o objetivo de analisar o desempenho de cada abordagem. Para isso, são apresentadas as matrizes de confusão e os valores de *accuracy* obtidos nos conjuntos de treino e teste, de forma a avaliar a capacidade de generalização de cada modelo para dados não vistos. Esta análise permite identificar potenciais sinais *overfitting* ou subajuste *underfitting*.

Na segunda parte procede-se à comparação entre os modelos desenvolvidos, com base nas métricas de desempenho mais relevantes para problemas de classificação, nomeadamente *accuracy*, precisão, *recall* e F1-score. Por fim, para reforçar a avaliação da capacidade de generalização dos modelos, é realizado um teste adicional com um objeto nunca antes apresentado durante o treino, simulando um cenário realista de previsão em dados nunca antes vistos.

5.4.1 Análise de Resultados

De forma a avaliar o desempenho dos modelos desenvolvidos foram calculadas métricas de desempenho padrão em problemas de classificação multiclasse, bem como as matrizes de confusão associadas a cada modelo.

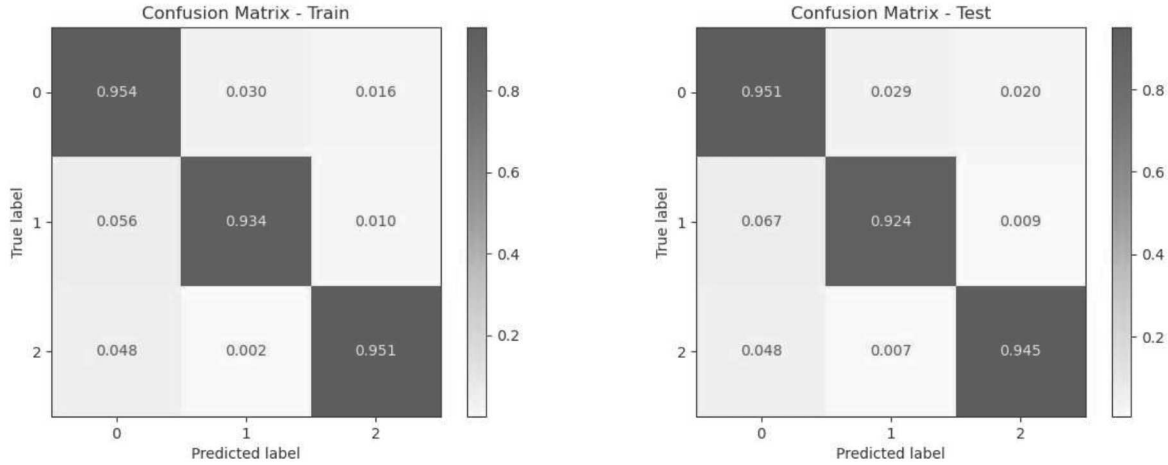
As métricas consideradas incluem *accuracy*, *precision*, *recall* e *F1-score*, fundamentais para avaliar a capacidade de cada modelo em distinguir corretamente as classes. A *accuracy* representa a proporção global de classificações corretas, enquanto a *precision* mede a proporção de classificações positivas que estavam corretas (evitando falsos positivos). O *recall*, por sua vez, indica a capacidade do modelo em identificar corretamente as instâncias de cada classe (evitando falsos negativos). Já o *F1-score* corresponde à média harmónica entre *precision* e *recall*, proporcionando uma visão equilibrada entre ambas as métricas.

As matrizes de confusão apresentadas para os conjuntos de treino e teste, permitem uma análise detalhada dos erros cometidos por cada modelo, mostrando a distribuição das previsões entre as diferentes classes. Através dessas matrizes, é possível verificar padrões de erro específicos, identificar classes mais suscetíveis a confusões e avaliar a capacidade de generalização dos modelos ao comparar os resultados obtidos no treino e no teste.

A avaliação detalhada dos três modelos, com as respetivas matrizes de confusão e métricas de desempenho, permite uma comparação direta entre as abordagens e uma análise crítica sobre a eficácia de cada uma na resolução do problema em estudo.

Regressão Logística

O modelo de Regressão Logística foi desenvolvido com o objetivo de distinguir entre as três classes do problema. Para avaliar a sua capacidade de generalização, foram analisadas as matrizes de confusão normalizadas para os conjuntos de treino e teste, apresentadas nas Figuras 5.9a e 5.9b.



(a) Matriz de confusão normalizada para o conjunto de treino da Regressão Logística

(b) Matriz de confusão normalizada para o conjunto de teste da Regressão Logística

Figura 5.9: Matrizes de confusão obtidas para o conjunto de treino e de teste para a Regressão Logística

No treino, o modelo apresenta elevados índices de acerto para todas as classes, com valores superiores a 93%. A classe 2 é a melhor discriminada (95.1% de acertos), enquanto a classe 0 apresenta ligeiras confusões com a classe 1 (4.6%). Já no conjunto de teste, observa-se uma ligeira diminuição no desempenho, especialmente no *recall* da classe 1 (92.4%) e da classe 2 (94.5%). No entanto, o modelo mantém uma boa consistência entre treino e teste, não evidenciando sinais de *overfitting*.

Complementando a análise das matrizes de confusão, a Tabela 5.2 apresenta as métricas de desempenho calculadas para o conjunto de teste. A *accuracy* global do modelo foi de 94%, e as médias macro e ponderada confirmam a robustez do modelo, com valores de precisão, *recall* e *F1-score* próximos entre si.

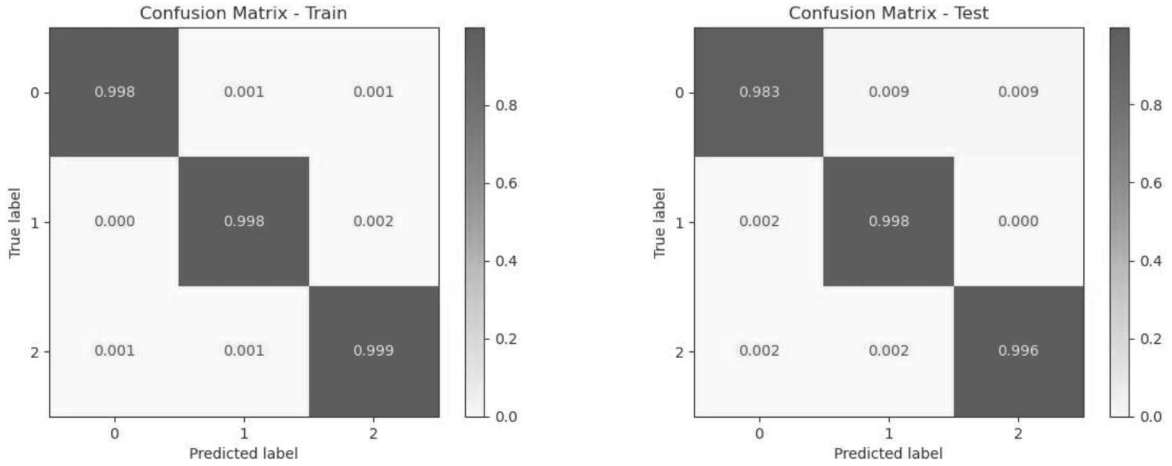
Destaca-se a elevada precisão para a classe 2 (0.98) e o *recall* elevado para a classe 0 (0.95), indicando que o modelo raramente confunde estas classes com as restantes. Por outro lado, a classe 0 apresenta a menor precisão (0.86), sugerindo uma maior tendência para falsos positivos nesta classe.

Classe	Precisão	Recall	F1-score
0	0.864	0.951	0.905
1	0.969	0.924	0.946
2	0.975	0.945	0.960
Média (Macro)	0.936	0.940	0.937
Média (Ponderada)	0.942	0.939	0.940

Tabela 5.2: Métricas de desempenho para a Regressão Logística

SVM

O modelo baseado *Support Vector Machine* obteve resultados notáveis, com uma percentagem de classificações corretas extremamente elevada em ambos os conjuntos de treino e teste. A análise das matrizes de confusão normalizadas (Figuras 5.10a e 5.10b) revela que a SVM consegue distinguir as três classes de forma muito precisa, com apenas pequenas confusões residuais.



(a) Matriz de confusão normalizada para o conjunto de treino da SVM

(b) Matriz de confusão normalizada para o conjunto de teste da SVM

Figura 5.10: Matrizes de confusão obtidas para o conjunto de treino e de teste para a SVM

No treino, observa-se que a SVM atinge uma *accuracy* de 99.8%, com a maioria das amostras corretamente classificadas. As confusões entre classes são quase inexistentes, limitando-se a frações inferiores a 0.2%. No teste, a SVM mantém um desempenho muito elevado, com *acuuracy* de 99.3%. As confusões residuais surgem principalmente na classe 0, onde cerca de 0.9% das amostras foram classificadas como pertencentes às classes 1 ou 2, enquanto as restantes classes apresentam uma percentagem de classificações corretas quase perfeita.

O relatório de classificação obtido para o conjunto de teste confirma a consistência do modelo, com *precision*, *recall* e *F1-score* acima dos 99% para todas as classes. Estes resultados demonstram que a SVM não apenas ajusta bem os dados de treino, mas também consegue generalizar com eficácia para novos dados, sem sinais evidentes de *overfitting*.

A Tabela 5.3 resume os valores das métricas de desempenho calculadas para o conjunto de teste.

Classe	Precisão	Recall	F1-score
0	0.994	0.983	0.988
1	0.991	0.998	0.994
2	0.993	0.996	0.995
Média (Macro)	0.993	0.992	0.992
Média (Ponderada)	0.993	0.993	0.993

Tabela 5.3: Métricas de desempenho no conjunto de teste para a SVM

De forma geral, a SVM demonstrou ser um modelo altamente eficaz e confiável, com excelente capacidade de discriminação entre as classes e resultados consistentes entre treino e teste.

Rede Neuronal

A rede neuronal também se revelou eficaz na tarefa de classificação, apresentando resultados de elevada precisão e robustez. A *accuracy* global obtida foi de 99.2% no conjunto de treino e de 98.5% no conjunto de teste, o que evidencia uma boa capacidade de generalização para novos dados.

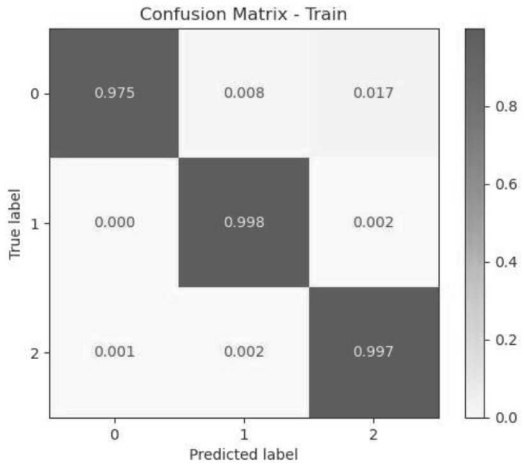
As matrizes de confusão, apresentadas nas Figuras 5.11a e 5.11b, permitem analisar de forma mais detalhada o desempenho do modelo em cada classe. No conjunto de treino, observa-se uma taxa de classificação correta superior a 97% para todas as classes, com confusões residuais de baixo impacto. A classe 1, em particular, apresenta uma taxa de acerto muito elevada (99.8%), enquanto a classe 0 regista uma ligeira dispersão, com aproximadamente 1.7% dos exemplos a serem classificados como pertencentes à classe 2.

No conjunto de teste, verifica-se uma ligeira redução da precisão, especialmente para a classe 0, onde cerca de 3.5% dos exemplos foram classificados incorretamente como pertencentes à classe 2. No entanto, as restantes classes mantêm taxas de classificação superiores a 99%, o que demonstra a consistência e a eficácia do modelo.

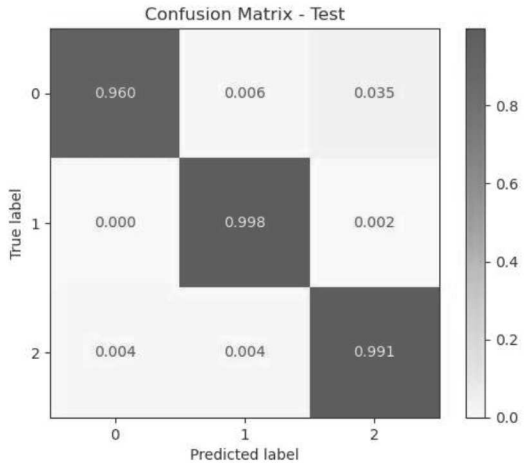
O relatório de classificação para o conjunto de teste, apresentado na Tabela 5.4, permite uma análise mais detalhada das métricas de desempenho. Observa-se que a classe 1 apresenta resultados praticamente perfeitos, com uma *precision* de 0.99, um *recall* de 1.00 e um F1-score de 0.99, evidenciando a capacidade do modelo em identificar corretamente a maioria dos exemplos desta classe, sem gerar falsos positivos relevantes. A classe 2 também apresenta valores muito elevados, com um F1-score de 0.98, apesar de uma ligeira redução na precisão (0.97), o que indica que o modelo tende a incluir alguns exemplos de outras classes como pertencentes a esta. A classe 0 é a que apresenta o desempenho ligeiramente inferior, com um F1-score de 0.98 e uma precisão de 0.99, mas com um *recall* de 0.96, sugerindo que alguns exemplos desta classe não foram corretamente identificados, sendo atribuídos a outras categorias.

No geral, o modelo apresenta valores médios de precisão, *recall* e F1-score bastante elevados, superiores a 97%, o que reflete um desempenho consistente e robusto em todas as classes.

De um modo geral, a rede neuronal destaca-se pelo seu desempenho robusto, mostrando uma



(a) Matriz de confusão normalizada para o conjunto de treino da Rede Neuronal



(b) Matriz de confusão normalizada para o conjunto de teste da Rede Neuronal

Figura 5.11: Matrizes de confusão obtidas para o conjunto de treino e de teste para a Rede Neuronal

Classe	Precisão	Recall	F1-score
0	0.994	0.960	0.976
1	0.991	0.998	0.994
2	0.972	0.991	0.982
Média (Macro)	0.986	0.983	0.984
Média (Ponderada)	0.985	0.985	0.985

Tabela 5.4: Métricas de desempenho no conjunto de teste para a Rede Neuronal

ligeira diminuição de desempenho no conjunto de teste, mas mantendo níveis de classificação bastante elevados. Estes resultados sugerem que o modelo é capaz de captar padrões relevantes nos dados, mantendo uma boa capacidade de generalização.

5.4.2 Comparação de desempenho

Após a análise individual de cada modelo é possível realizar uma comparação direta dos seus desempenhos, de forma a identificar qual abordagem apresenta melhores resultados para a tarefa de classificação em estudo.

A Tabela 5.5 sintetiza os principais indicadores de desempenho, precisão, *recall*, F1-score e *accuracy*, obtidos para cada modelo no conjunto de teste. Estes valores permitem uma avaliação quantitativa clara das diferenças entre os três modelos desenvolvidos: Regressão Logística, SVM e Rede Neuronal.

Modelo	Precisão	Recall	F1-score	Accuracy
Regressão Logística	0.94	0.94	0.94	0.94
SVM	0.99	0.99	0.99	0.99
Rede Neuronal	0.99	0.98	0.98	0.98

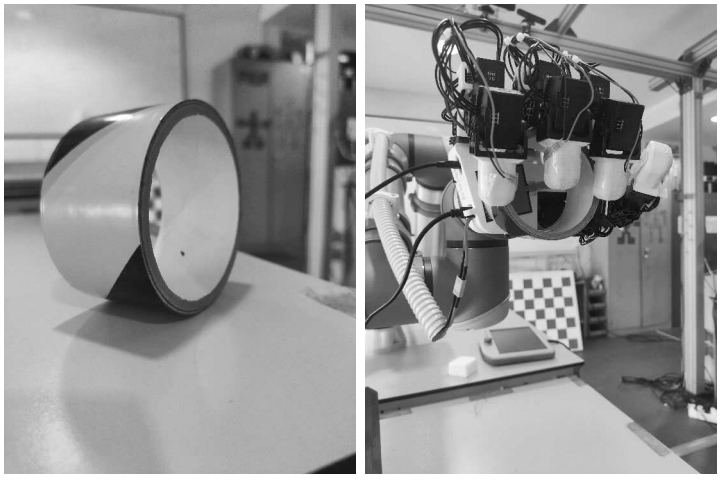
Tabela 5.5: Comparação das métricas de desempenho dos três modelos avaliados

Analisando os resultados apresentados, verifica-se que o modelo de Regressão Logística apresenta um desempenho global inferior em comparação com os outros modelos, com uma *accuracy* de 94% e valores de precisão, *recall* e F1-score médios de 0.94. Esta diferença é particularmente notória na classe 0, onde o *recall* foi de 0.95 e a precisão de 0.86, revelando alguma dificuldade em identificar corretamente esta classe.

Por outro lado, o modelo SVM destacou-se como o que apresentou o melhor desempenho, com uma acurácia de 99% e métricas médias de 0.99 para precisão, *recall* e F1-score. Este desempenho excecional confirma a elevada capacidade discriminativa da SVM para este problema, conseguindo classificar corretamente quase todos os exemplos, sem desvios significativos entre as classes.

A rede neuronal também apresentou resultados muito satisfatórios, com uma *accuracy* de 98% e métricas médias próximas de 0.98. Embora ligeiramente inferior ao desempenho da SVM, a rede neuronal demonstrou uma performance bastante consistente e equilibrada entre as diferentes classes. Destaca-se, no entanto, uma ligeira diminuição do *recall* para a classe 0 (0.96), o que indica que o modelo, ocasionalmente, não conseguiu identificar corretamente alguns exemplos desta classe.

Para além do cálculo das métricas de desempenho, foi também testada a capacidade de generalização dos modelos com um novo conjunto de dados, correspondente a um objeto que, ao ser agarrado pela mão robótica, gera uma configuração de dedos associada à classe 2 (situação em que o polegar fica por cima dos restantes dedos). Este novo objeto é ilustrado na Figura 5.12a e na Figura 5.12b é possível observar a posição distinta assumida pelos dedos da mão ao agarrar o objeto, em comparação com os exemplos utilizados durante o treino dos modelos.



(a) Objeto utilizado para recolher novos dados

(b) Configuração de dedos associada ao novo objeto

Figura 5.12: Fotografias do novo objeto utilizado para avaliar o desempenho dos modelos desenvolvidos e respetiva configuração de dedos.

A Tabela 5.6 resume as classificações atribuídas pelos diferentes modelos ao novo conjunto de dados. Através da sua análise verifica-se que a Regressão Logística classificou incorretamente

o exemplo como pertencente à classe 0, enquanto a SVM e a Rede Neuronal foram capazes de identificar corretamente a classe 2, demonstrando assim uma maior capacidade de generalização para dados não vistos durante o treino.

Modelo	**Classe Prevista**
Regressão Logística	0
SVM	2
Rede Neuronal	2

Tabela 5.6: Classificação do novo objeto por cada modelo

A análise destes resultados revela informações importantes sobre o comportamento dos modelos. A Regressão Logística, embora com desempenho global razoável, mostrou fragilidades ao classificar novos dados que não pertencem ao conjunto de treino, cometendo erros ao identificar objetos da classe 2. Por outro lado, a SVM e a Rede Neuronal demonstraram uma capacidade de generalização mais robusta, conseguindo prever corretamente a classe do novo objeto, mesmo sem terem sido explicitamente treinadas com ele. Este resultado confirma a superioridade destes modelos, não apenas nas métricas de teste, mas também na capacidade de extrapolar para dados desconhecidos.

Em resumo, a SVM demonstrou o desempenho mais robusto e equilibrado, enquanto a rede neuronal surge como uma alternativa viável com resultados muito próximos. Já a Regressão Logística, apesar de apresentar resultados satisfatórios, ficou aquém do desempenho dos outros dois modelos, sendo mais sensível a desequilíbrios entre as classes.

Esta análise permite concluir que, para este problema específico de classificação, a SVM é o modelo mais adequado, dado o seu desempenho superior em todas as métricas avaliadas. A rede neuronal surge como uma segunda opção viável, e a Regressão Logística, embora funcional, revela algumas limitações no reconhecimento de certas classes.

5.5 Conclusão

Neste capítulo foi explorada a aplicação de técnicas de aprendizagem automática para a deteção de sobreposições entre os dedos da mão robótica desenvolvida. Foram treinados três modelos distintos (Regressão Logística, SVM e Rede Neuronal)e os seus desempenhos foram comparados através de métricas como precisão, *recall*, *f1-score* e *accuracy*. As tabelas e análises obtidas demonstraram que tanto a SVM como a Rede Neuronal superaram a Regressão Logística, alcançando valores mais elevados em todas as métricas de desempenho, com a SVM a apresentar resultados ligeiramente superiores.

Para além das métricas padrão de avaliação foi realizado um teste adicional para validar a capacidade de generalização dos modelos, através da classificação de um novo objeto cuja configuração dos dedos correspondia à classe 2 (situação em que o polegar fica por cima dos restantes dedos). Este exercício revelou limitações da Regressão Logística, que classificou o novo objeto incorretamente como pertencente à classe 0, enquanto a SVM e a Rede Neuronal

conseguiram identificar corretamente a classe 2, demonstrando maior robustez na generalização para novos cenários.

Os resultados obtidos neste estudo reforçam a importância de escolher cuidadosamente o modelo de aprendizagem automática, em função das exigências do problema em questão. Em particular, para o caso da deteção de sobreposições em mãos robóticas antropomórficas, a SVM e a Rede Neuronal revelaram-se soluções mais eficazes e fiáveis, sendo por isso as opções mais recomendadas para a implementação final deste sistema.

CAPÍTULO 6

Conclusões e Trabalho Futuro

6.1 Conclusões

O trabalho apresentado nesta dissertação permitiu desenvolver uma mão robótica antropomórfica funcional, integrando capacidades de controlo, sensorização de contacto e funcionalidades avançadas. Desde a seleção, fabrico e montagem da estrutura da mão até à incorporação de sensores de força (*FSR*) e à implementação da unidade de processamento baseada em *ROS*, foram concretizados todos os objetivos delineados no início do projeto.

Destaca-se a implementação de funcionalidades avançadas, como o ajuste dinâmico de corrente durante o contacto com objetos, a definição de velocidades relativas entre dedos e falanges, e a introdução de atrasos temporais programáveis entre os movimentos dos dedos. Estas funcionalidades conferem à mão robótica a capacidade de realizar movimentos mais naturais, seguros e adaptáveis, aproximando o seu comportamento ao de uma mão humana.

No decorrer dos testes de validação do sistema, foi possível aferir o desempenho da arquitetura desenvolvida em termos de aquisição de dados. Verificou-se que os sensores FSR podem ser lidos a uma frequência de aproximadamente 1000 Hz, valor viabilizado pela utilização do microcontrolador Teensy 4.0, com frequência de operação até 600 MHz. Esta elevada taxa de amostragem garante uma perceção tátil responsiva e adequada a aplicações em tempo real. Em paralelo, confirmou-se que a leitura do estado dos motores pode ser realizada a uma frequência de 50 Hz de forma estável, permitindo monitorizar continuamente variáveis como posição, velocidade e corrente, essenciais para controlo e diagnóstico do sistema.

Durante o desenvolvimento do projeto, foi identificado um problema crítico: a ocorrência de sobreposições entre os dedos durante o movimento de abertura, potencialmente geradora de bloqueios mecânicos e esforços excessivos nos motores. Para resolver este desafio, foi concebida e validada uma abordagem baseada em *Machine Learning* para a classificação de configurações de sobreposição, recorrendo a modelos como Regressão Logística, SVM e Redes Neuronais. A avaliação comparativa demonstrou que a SVM apresentou os melhores resultados, com elevada precisão e robustez, validando a eficácia da abordagem proposta.

No seu conjunto, este trabalho contribui para o avanço no desenvolvimento de mãos robóticas antropomórficas modulares, com capacidades de perceção tátil e controlo baseado em corrente, abrindo novas perspetivas para aplicações futuras em cenários reais, como a robótica assistiva, próteses ativas ou manipulação colaborativa em ambientes industriais.

6.2 Contribuições

O trabalho desenvolvido nesta dissertação resultou num conjunto de contribuições relevantes no domínio do controlo e perceção em mãos robóticas antropomórficas, das quais se destacam:

- **Desenvolvimento completo da arquitetura de software da mão robótica**, com uma estrutura modular e escalável, permitindo fácil adaptação a diferentes configurações e requisitos funcionais.
- **Desenvolvimento de funcionalidades avançadas de controlo da mão**, como a implementação de offsets temporais entre os dedos, permitindo movimentos mais fluidos e naturais, aproximando o comportamento da mão robótica ao de uma mão humana.
- **Integração de sensores de contacto piezoresistivos (FSR)**, incluindo o desenho e fabrico de suportes personalizados para a sua correta fixação na estrutura da mão, assegurando robustez mecânica e qualidade de medição.
- **Desenvolvimento de uma solução para aumento da zona sensível dos sensores FSR**, através da conceção e fabrico de peças específicas que permitem distribuir melhor a força aplicada, aumentando a fiabilidade da leitura sensorial.
- **Implementação de um sistema de leitura e integração de dados sensoriais**, que combina a informação dos motores e dos sensores de contacto, permitindo a execução de gestos mais controlados e a prevenção de forças excessivas.
- **Conceção de um sistema automatizado de aquisição de dados**, recorrendo ao braço robótico UR10 para gerar movimentos repetíveis da mão em múltiplas configurações, possibilitando a criação de um dataset diversificado e representativo.
- **Treino e avaliação de modelos de aprendizagem automática para deteção de sobreposições de dedos**, com destaque para o modelo baseado em *Support Vector Machines* (SVM), que apresentou melhor desempenho na classificação de sobreposições dos dedos.

6.3 Trabalho Futuro

O trabalho realizado estabelece uma base sólida para projetos futuros, abrindo diversas possibilidades de investigação e desenvolvimento. Um dos passos mais imediatos será a integração da solução de classificação de sobreposições no ciclo de controlo em tempo real da mão robótica, permitindo detetar e corrigir situações de sobreposição à medida que estas ocorrem, de forma autónoma e sem necessidade de intervenção externa.

Além disso, será importante aprofundar a integração entre o sistema sensorial e o controlo motor, explorando a utilização dos dados provenientes dos sensores FSR para ajustar dinamicamente as trajetórias e forças aplicadas em função das características do objeto manipulado.

Esta integração permitirá implementar estratégias de controlo mais avançadas, otimizando a interação com objetos de diferentes formas, tamanhos e materiais.

Outra linha de trabalho relevante poderá ser a integração de sensores adicionais, como sensores de posição de alta resolução ou sensores de pressão distribuída, para enriquecer a perceção tátil do sistema.

Por fim, será interessante avaliar a aplicabilidade da solução desenvolvida em contextos práticos, como o controlo de próteses robóticas, ou a colaboração segura em ambientes industriais. Estes cenários representam um campo de aplicação desafiante e de elevado impacto, onde soluções como a desenvolvida neste trabalho podem contribuir significativamente para aproximar as capacidades das mãos robóticas às da mão humana, promovendo uma interação mais segura, eficiente e inteligente.

Referências

[1] M. R. Cutkosky, «On grasp choice, grasp models, and the design of hands for manufacturing tasks,» *IEEE Transactions on Robotics and Automation*, vol. 5, n.º 3, pp. 269–279, 1989. (acedido em 03/06/2025).

[2] A. Bicchi, «Hands for dexterous manipulation and robust grasping: A difficult road toward simplicity,» *IEEE Transactions on Robotics and Automation*, vol. 16, n.º 6, pp. 652–662, 2000. (acedido em 03/06/2025).

[3] T. Feix, J. Romero, H. Schmiedmayer, A. M. Dollar e D. Kragic, «The grasp taxonomy of human grasp types,» *IEEE Transactions on Human-Machine Systems*, vol. 46, n.º 1, pp. 66–77, 2015. (acedido em 03/06/2025).

[4] A. Billard e D. Kragic, «Trends and challenges in robot manipulation,» *Science*, vol. 364, n.º 6446, eaat8414, 2019. (acedido em 03/06/2025).

[5] C. Piazza, G. Grioli, M. G. Catalano e A. Bicchi, «A century of robotic hands,» *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 2, pp. 1–32, 2019. (acedido em 03/06/2025).

[6] W. Shang, F. Song, Z. Zhao, H. Gao, S. Cong e Z. Li, «Deep learning method for grasping novel objects using dexterous hands,» *IEEE Transactions on Cybernetics*, vol. 52, n.º 5, pp. 2750–2762, 2022. (acedido em 03/06/2025).

[7] Y. Li, P. Wang, R. Li, M. Tao, Z. Liu e H. Qiao, «A survey of multifingered robotic manipulation: Biological results, structural evolvements, and learning methods,» *Frontiers in Neurorobotics*, vol. 16, p. 843 267, 2022. (acedido em 03/06/2025).

[8] *Dexterous Robotic Hand - OceanTrix Robotics Since 1996*. URL: `https://oceantrix.com/robotic-hand/` (acedido em 02/05/2025).

[9] *Agile Hand – the human-like robotic hand*. URL: `https://www.agile-robots.com/en/solutions/agile-hand` (acedido em 04/05/2025).

[10] H. Park e D. Kim, «An open-source anthropomorphic robot hand system: HRI hand,» *HardwareX*, vol. 7, e00100, fev. de 2020, ISSN: 2468-0672. DOI: `10.1016/J.OHX.2020.E00100`. (acedido em 29/10/2024).

[11] *Robot Nano Hand - open source robot hand project*. URL: `https://robotnanohand.com/` (acedido em 10/01/2025).

[12] A. Nurpeissova, T. Tursynbekov e A. Shintemirov, «An Open-Source Mechanical Design of Alaris Hand: A 6-DOF Anthropomorphic Robotic Hand,» *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 1177–1183, mai. de 2021, ISSN: 10504729. DOI: `10.1109/ICRA48506.2021.9561977`. (acedido em 10/01/2025).

[13] K. Shaw, A. Agarwal e D. Pathak, «LEAP Hand: Low-Cost, Efficient, and Anthropomorphic Hand for Robot Learning,» *Robotics: Science and Systems (RSS)*, 2023. (acedido em 02/05/2025).

[14] Syntouch, *Biotac*, 2018. URL: `%5Curl%7Bhttps://www.syntouchinc.com/wp-content/uploads/2018/08/BioTac-Manual-V.21.pdf%7D` (acedido em 24/10/2024).

[15] L. Jamone, L. Natale, M. Giorgio e G. Sandini, «Highly sensitive soft tactile sensors for an anthropomorphic robotic hand,» *IEEE Sensors Journal*, vol. 15, pp. 4226–4233, 8 mar. de 2015. DOI: `10.1109/JSEN.2015.2417759`. URL: `https://ieeexplore.ieee.org/abstract/document/7070742/` (acedido em 24/10/2024).

[16] Y. Yan, Z. Hu, Z. Yang et al., «Soft magnetic skin for super-resolution tactile sensing with force self-decoupling,» *Science Robotics*, vol. 6, 51 fev. de 2021, ISSN: 24709476. DOI: 10.1126/SCIROBOTICS.ABC8801. (acedido em 28/11/2024).

[17] T. Kawasetsu, T. Horii, H. Ishihara e M. Asada, «Mexican-Hat-Like Response in a Flexible Tactile Sensor Using a Magnetorheological Elastomer,» *Sensors*, vol. 18, p. 587, 2 fev. de 2018, ISSN: 1424-8220. DOI: 10.3390/S18020587. URL: https://www.mdpi.com/1424-8220/18/2/587/htm%20https://www.mdpi.com/1424-8220/18/2/587 (acedido em 04/12/2024).

[18] A. Alfadhel, M. A. Khan, S. Cardoso, D. Leitao e J. Kosel, «A Magnetoresistive Tactile Sensor for Harsh Environment Applications,» *Sensors*, vol. 16, p. 650, 5 mai. de 2016, ISSN: 1424-8220. DOI: 10.3390/S16050650. URL: https://www.mdpi.com/1424-8220/16/5/650/htm%20https://www.mdpi.com/1424-8220/16/5/650 (acedido em 05/12/2024).

[19] H. Yang, L. Weng, B. Wang e W. Huang, «Design and Characterization of High-Sensitivity Magnetostrictive Tactile Sensor Array,» *IEEE Sensors Journal*, vol. 22, pp. 4004–4013, 5 mar. de 2022, ISSN: 15581748. DOI: 10.1109/JSEN.2022.3145822. (acedido em 05/12/2024).

[20] F. Yin, H. Niu, E. S. Kim, Y. K. Shin, Y. Li e N. Y. Kim, «Advanced polymer materials-based electronic skins for tactile and non-contact sensing applications,» *InfoMat*, vol. 5, e12424, 7 jun. de 2023, ISSN: 2567-3165. DOI: 10.1002/INF2.12424. URL: https://onlinelibrary.wiley.com/doi/full/10.1002/inf2.12424%20https://onlinelibrary.wiley.com/doi/abs/10.1002/inf2.12424%20https://onlinelibrary.wiley.com/doi/10.1002/inf2.12424 (acedido em 14/12/2024).

[21] L. Seminara, L. Pinna, M. Capurro e M. Valle, «A Tactile Sensing System Based on Arrays of Piezoelectric Polymer Transducers,» em *Smart Actuation and Sensing Systems - Recent Advances and Future Challenges*. IntechOpen, out. de 2012, pp. 611–638, ISBN: 978-953-51-0798-9. DOI: 10.5772/50112. URL: https://www.intechopen.com/chapters/39972%20undefined/chapters/39972 (acedido em 14/12/2024).

[22] Y. Zhong, J. Wang, L. Han et al., «High-performance flexible self-powered triboelectric pressure sensor based on chemically modified micropatterned PDMS film,» *Sensors and Actuators A: Physical*, vol. 349, p. 114 013, jan. de 2023, ISSN: 0924-4247. DOI: 10.1016/J.SNA.2022.114013. (acedido em 14/12/2024).

[23] K. Tao, Z. Chen, J. Yu et al., «Ultra-Sensitive, Deformable, and Transparent Triboelectric Tactile Sensor Based on Micro-Pyramid Patterned Ionic Hydrogel for Interactive Human–Machine Interfaces,» *Advanced Science*, vol. 9, p. 2 104 168, 10 jan. de 2022, ISSN: 2198-3844. DOI: 10.1002/ADVS.202104168. URL: https://onlinelibrary.wiley.com/doi/full/10.1002/advs.202104168%20https://onlinelibrary.wiley.com/doi/abs/10.1002/advs.202104168%20https://onlinelibrary.wiley.com/doi/10.1002/advs.202104168 (acedido em 14/12/2024).

[24] A. Ke, J. Huang, L. Chen et al., «Fingertip Tactile Sensor with Single Sensing Element Based on FSR and PVDF,» *IEEE Sensors Journal*, vol. 19, pp. 11 100–11 112, 23 dez. de 2019, ISSN: 15581748. DOI: 10.1109/JSEN.2019.2936304. URL: https://ieeexplore.ieee.org/document/8805357 (acedido em 02/11/2024).

[25] X. Lu, D. Sun, H. Yin et al., «3-D Tactile-Based Object Recognition for Robot Hands Using Force-Sensitive and Bend Sensor Arrays,» *IEEE Transactions on Cognitive and Developmental Systems*, vol. 15, pp. 1645–1655, 4 dez. de 2023, ISSN: 23798939. DOI: 10.1109/TCDS.2022.3215021. (acedido em 15/12/2024).

[26] I. I. Electronics, *FSR Sensor, Force Sensing Resistor | Interlink Electronics*. URL: https://www.interlinkelectronics.com/force-sensing-resistor (acedido em 24/10/2024).

[27] S. Stassi, V. Cauda, G. Canavese e C. F. Pirri, «Flexible Tactile Sensing Based on Piezoresistive Composites: A Review,» *Sensors*, vol. 14, pp. 5296–5332, 3 mar. de 2014, ISSN: 1424-8220. DOI: 10.3390/S140305296. URL: https://www.mdpi.com/1424-8220/14/3/5296/htm%20https://www.mdpi.com/1424-8220/14/3/5296 (acedido em 16/12/2024).

[28] B. Meenu e B. Bakariya, «A Comprehensive Review of Cross-Validation Techniques in Machine Learning,» *International Journal on Science and Technology*, vol. 16, 1 jan. de 2025, ISSN: 2229-7677. DOI: 10.71097/IJSAT.v16.i1.1305. URL: https://www.ijsat.org/research-paper.php?id=1305 (acedido em 31/05/2025).

[29] A. Géron, *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow*. O'Reilly Media, Inc., 2019.

[30] P. et al., *scikit-learn: Machine Learning in Python*, 2011. URL: `https://scikit-learn.org/stable/` (acedido em 31/05/2025).

[31] C. Cortes e V. Vapnik, «Support-vector networks,» *Machine learning*, vol. 20, n.º 3, pp. 273–297, 1995. (acedido em 31/05/2025).

[32] I. Goodfellow, Y. Bengio e A. Courville, *Deep Learning*. MIT Press, 2016. URL: `https://www.deeplearningbook.org` (acedido em 01/06/2025).

[33] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever e R. Salakhutdinov, «Dropout: A Simple Way to Prevent Neural Networks from Overfitting,» *Journal of Machine Learning Research*, vol. 15, n.º 1, pp. 1929–1958, 2014. (acedido em 31/05/2025).

APÊNDICE A

Apêndice

A.1 Montagem dos Dedos

A.1.1 Montagem do Polegar

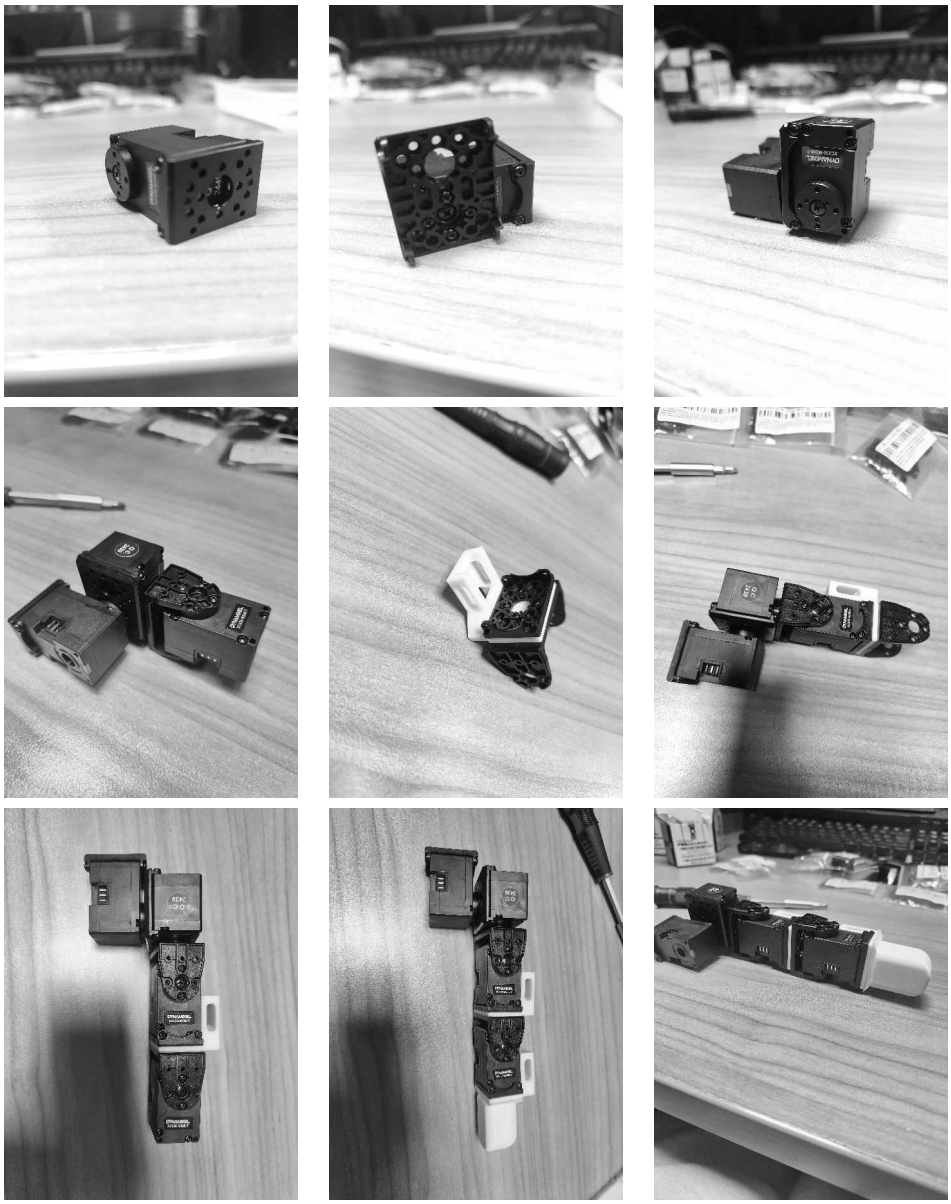


Figura A.1: Fotografias sequenciais da montagem do polegar

A.1.2 Montagem dos Restantes Dedos

Figura A.2: Fotografias sequenciais da montagem do dedo referente ao dedo indicador, médio ou anelar

A.1.3 Hiperligações

Hiperligação para vídeo de montagem de um dedo: `https://youtu.be/oWXucHKOkyo`
Hiperligação para vídeo de montagem do polegar: `https://youtu.be/FOSO4HZ6_bg`

A.2 Testes Realizados com Um Dedo

A.2.1 Teste com Duas Velocidades Diferentes

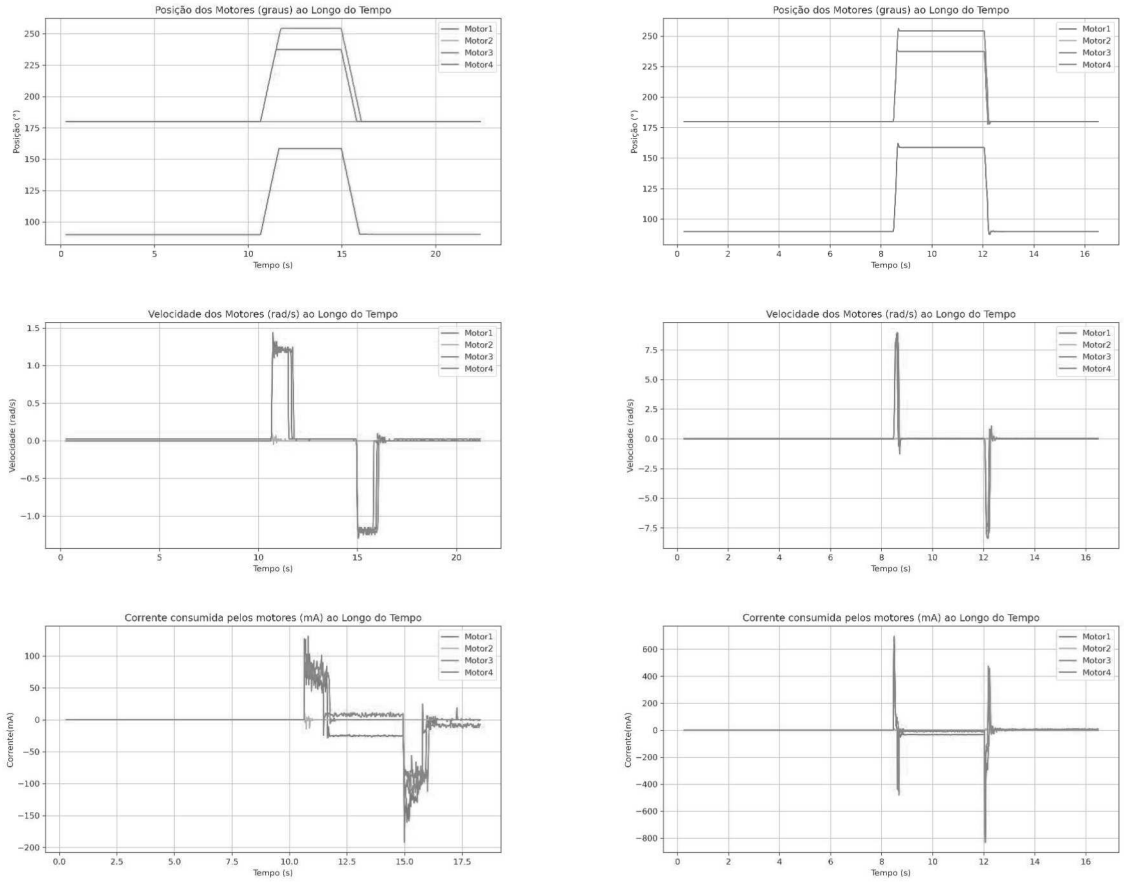


Figura A.3: Gráficos da posição, velocidade e corrente consumida pelos motores de um dedo. A coluna da esquerda corresponde a uma velocidade máxima configurada de 1.2 rad/s e a coluna da direita a 23.98 rad/s.

Hiperligação para vídeo: `https://www.youtube.com/shorts/_3l9AzUPu9A`

A.2.2 Teste com Obstáculo em Posições Diferentes

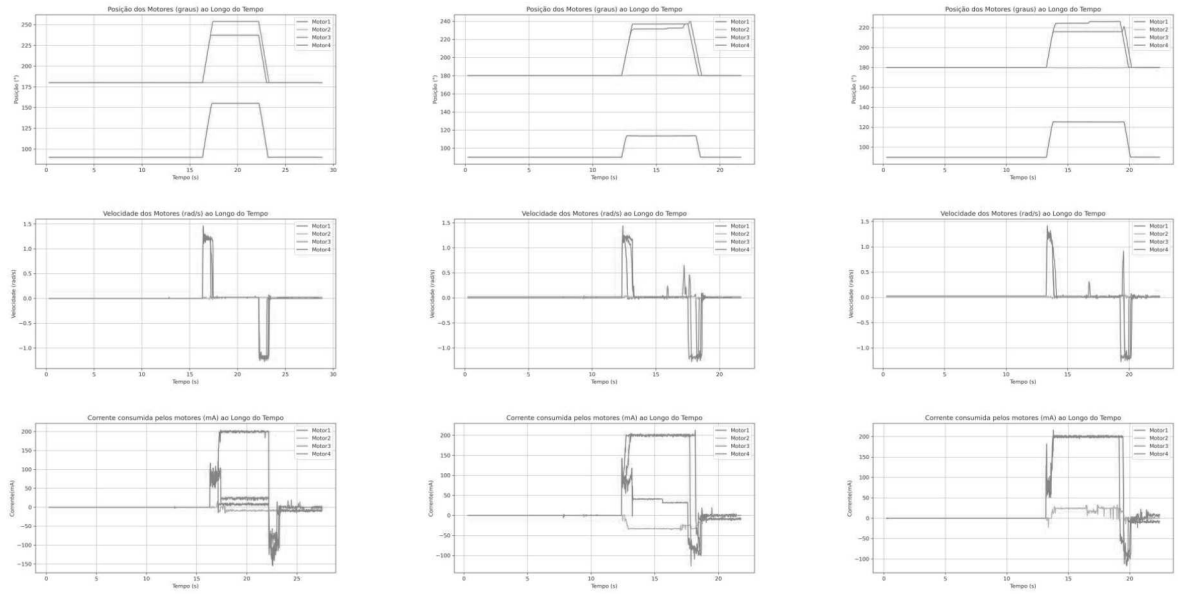


Figura A.4: Gráficos da posição, velocidade e corrente consumida pelos motores de um dedo, em diferentes cenários de obstrução. A coluna da esquerda representa a presença de um obstáculo junto ao motor mais próximo da base do dedo, a coluna central refere-se à obstrução a meio do dedo, e a coluna da direita corresponde à interferência na ponta do dedo.

Hiperligação para vídeo: `https://www.youtube.com/shorts/Qy4lXXRUP4Q`

A.3 Funcionalidades Avançadas

A.3.1 Implementação de Velocidades Relativas entre Dedos e Falanges

Velocidades Relativas entre Dedos

Hiperligação para o vídeo com os dois dedos com velocidade igual: `https://www.youtube.com/watch?v=40anKBzSLEo`

Hiperligação para o vídeo com o dedo médio com o dobro da velocidade do polegar: `https://www.youtube.com/watch?v=y8ddleaoB3U`

Velocidades Relativas entre Falanges

Hiperligação para o vídeo com a mesma velocidade para todos os motores do dedo: `https://www.youtube.com/watch?v=8vk7IcXj0z4`

Hiperligação para o vídeo com o dobro da velocidade para o primeiro motor do dedo: `https://www.youtube.com/watch?v=QenLMoWv7jk`

Atrasos Temporais Programáveis entre Movimentos de Dedos

Hiperligação para o vídeo do movimento *wave*: `https://www.youtube.com/watch?v=BEH4XPx-fHk`

A.4 Aquisição de Dados para o *Dataset*

Hiperligação para o vídeo de aquisição de dados: `https://youtube.com/shorts/tQWnxhsu6h4?feature=share`