

DESENVOLVIMENTO DE INTERFACE GRÁFICA E INTERAÇÃO HOMEM-MÁQUINA PARA O MANIPULADOR ED7220C

Development Of Graphic Interface and Man-Machine Interaction for The ED7220C Manipulator

Matheus R. S. Maia ^{1*} e Matheus Romão ¹

¹ Universidade Federal de São João del-Rei, Ouro Branco, Brasil

*Autor correspondente: matheusrodriguesmaia0318@gmail.com

RESUMO

As chamadas Interfaces Homem-Máquina desempenham um papel crucial em diversos setores, como o industrial e da biotecnologia, permitindo o controle e interação de humanos para com máquinas de forma que trabalhem de forma conjunta e eficaz para atingir determinados objetivos. O presente trabalho tem como objetivo aprimorar o aprendizado pela utilização do manipulador ED7220C presente no laboratório de robótica da Universidade Federal de São João del-Rei (UFSJ) – Campus Alto Paraopeba, justificando-se pelo fato de que através da interface gráfica disponibilizada pelo fabricante há, por diversas vezes, um prejuízo no aprendizado dos alunos por erros de múltiplas naturezas apresentados ao longo dos processos práticos com o equipamento. Além disso, utilizando-se do manipulador referido, o projeto em questão propõe-se a consolidar, por meio de leituras de movimentos de um braço real realizadas posteriormente através de uma série de sensores, a movimentação do ED7220C, atuando assim como um intermediário entre o movimento biológico e o robótico, o que permite uma série de aplicações no setor industrial e de saúde / biotecnologia.

Palavras-chave: Robótica, ED7220C, Interface, Computação, Python, Educação, Bioengenharia.

ABSTRACT

The so-called Human-Machine Interfaces play a crucial role in several sectors, such as industrial and biotechnology, allowing the control and interaction of humans with machines so that they work together and effectively to achieve certain objectives. The present work aims to improve learning through the use of the ED7220C manipulator present in the robotics laboratory of the Federal University of São João del-Rei (UFSJ) – Campus Alto Paraopeba, justified by the fact that through the graphical interface provided by the manufacturer There is, on several occasions, a loss in students' learning due to errors of multiple natures presented throughout the practical processes with the equipment. Furthermore, using the aforementioned manipulator, the project in question proposes to consolidate, through readings of movements of a real arm carried out later through a series of sensors, the movement of the ED7220C, thus acting as an intermediary between biological and robotic movement, which allows for a series of applications in the industrial and health/biotechnology sector.

Keywords: Robotics, ED7220C, Interface, Computing, Python, Education, Bioengineering.

1. Introdução

A utilização de tecnologias para a otimização de tarefas e expansão no horizonte de possibilidades é um fato indiscutivelmente crescente ao longo do tempo nas mais diversas áreas imagináveis, fazendo uso da computação, mecânica, elétrica, eletrônica e tudo o que for correlato para alcançar os objetivos determinados. Um grande exemplo desta aplicação crescente da tecnologia é no ramo industrial. Em um cenário onde cada segundo de tempo, cada grama de material, cada mínima possibilidade de otimização representa, no longo prazo, um grande impacto econômico, o investimento na aplicação de tecnologias que revolucionam o mercado é extremamente presente nesta área. Um grande exemplo de aplicação ampla da tecnologia no ramo industrial é a implementação do braço robótico para a execução de tarefas que antes eram manuais de maneira muito mais segura e otimizada.

Este tipo de mecanismo desempenha um papel crucial em tarefas que envolvem riscos à saúde dos operadores humanos ou quando as atividades são excessivamente repetitivas, comprometendo a qualidade quando realizadas por pessoas. Além disso, funções que requerem rapidez, controle preciso e uniformidade de produção são mais eficientemente executadas por esses dispositivos. Exemplos dessas atividades incluem soldagem, pintura, montagem mecânica e eletrônica, bem como a manipulação de materiais, como carga e descarga de peças em máquinas ou estoques [1].

Diante deste cenário, a formação de profissionais da área da tecnologia que são capazes de lidar com maestria com estes avanços é imprescindível para que a constante evolução e implementação de técnicas deste cunho sejam constantemente realizadas da melhor forma possível, se adaptado a todos os cenários que virão ao longo do tempo.

Deste modo, instituições universitárias como a Universidade Federal de São João del-Rei (UFSJ – Campus Alto Paraopeba) desempenham um papel imprescindível para o fomento do desenvolvimento de tais tecnologias a partir da formação de profissionais capacitados. Tratando-se da universidade em questão (UFSJ – CAP), com o objetivo de oferecer experiências práticas aos alunos, há no laboratório de robótica alguns exemplares do manipulador ED-7220C que são utilizados para experimentos de controle dos mesmos para a execução de rotinas de movimentação para

interação com objetos reais, atuando desta forma como uma forma de os alunos experienciarem um pouco do cenário de aplicação industrial.

De modo geral, ao se trabalhar com um equipamento interativo, é essencial que haja uma comunicação de qualidade estabelecida por meio de uma interface homem-máquina, e neste cenário não seria diferente.

Embora o fabricante forneça um software que permite a interação com o equipamento, que aqui será chamado de “braço robótico” ou de “manipulador ED-7220C”, este software mostrou-se insuficientemente produtivo a nível de aprendizado, uma vez que por vezes acabou apresentando falhas de execução que comprometem a realização das atividades práticas durante os experimentos. Desta forma, neste sentido, a proposta do presente trabalho é de ter, ao final, um software de fácil execução que possibilitará o controle do manipulador ED-7220C através de uma interface gráfica para tal interação homem-máquina, onde será possível visualizar o movimento do braço robótico replicado em um software de simulação computacional, no caso, o CoppeliaSim. Esta proposta justifica-se, pois, além de aprimorar a experiência com o controle do braço robótico por meio da interface, facilitando o aprendizado pelas experiências em laboratório, ainda atuará, de certa forma, como um sistema supervisor para o manipulador, devido à natureza de visualização da máquina no software computacional.

Além do cenário de aplicação industrial descrito, dentre diversos outros, destaca-se neste projeto, a implementação da tecnologia no ramo da bioengenharia, mais especificamente na mimetização de movimentos de um membro real por parte de um equipamento robótico. Esta técnica não se limita apenas ao uso em ambientes de produção, mas também possui aplicações significativas na área da saúde. Exemplos incluem sua utilização em hospitais para a realização de procedimentos cirúrgicos de forma remota, onde traz diversos benefícios à modalidade padrão, como o fato de que com o procedimento de cirurgia remota as incisões são significativamente menores, há uma maior precisão cirúrgica por eliminar erros causados por movimentos involuntários humanos, e traz maior ergonomia ao médico responsável [2]. Além disso, destaca-se também o uso da mimetização de movimentos no desenvolvimento de próteses que respondem a sinais eletromiográficos emitidos pelo corpo do paciente. Tais próteses tem como objetivo

principal minimizar os prejuízos gerados pela amputação, melhorando a qualidade de vida do paciente ao retomar, ao menos parcialmente, a liberdade de movimentos do mesmo [3].

Tendo em vista o cenário descrito, este trabalho também propõe-se a gerar um sistema que receba a leitura de sinais de movimentos de um membro biológico fornecidos através de sensoriamentos diversos e seja capaz de mimetizar esta movimentação, replicando-a no manipulador ED-7220C, atuando assim como um sistema de mimetização completo, embora relativamente rudimentar, mas que abre uma vasta gama de possibilidades para aprimoramentos futuros e aplicações realmente viáveis nos setores da bioengenharia citados.

2. Materiais

Para o desenvolvimento das propostas descritas anteriormente, o projeto foi segmentado em duas vertentes, onde a primeira tratará da interface gráfica de controle do manipulador ED-7220C, aprimorando sua utilização em experimentos práticos a fins de aprendizado, e será chamada de “Interface Gráfica Para o Manipulador ED7220C”, enquanto a segunda irá realizar o controle do braço robótico utilizando-se da mimetização dos movimentos descritos pelos sinais fornecidos por sensores que exercem a função de captar dados de um braço biológico, e será chamada de “Mimetização via Interface Homem-Máquina Para o Manipulador ED7220C”.

2.1. Manipulador ED7220C

O componente central das propostas aqui definidas é o Manipulador ED7220C. Desenvolvido pela Sul Coreana ED Corporation, este manipulador é amplamente utilizado para pesquisa, desenvolvimento e ensino, uma vez que possibilita a observação clara do comportamento mecânico de cada junta perante aos comandos recebidos via comunicação serial [4].

A construção do ED7220C permite que ele tenha 6 graus de liberdade, sendo geometricamente configurado para se assemelhar às juntas presentes em um braço biológico humano, contendo articulações de corpo, ombro, cotovelo e pulso (Figura 1), que funcionam por meio de servos motores DME 38B50G de acionamento óptico.

No contexto do presente projeto, o manipulador em questão será o último componente dos

sistemas aqui propostos, o objeto cujo o movimento realizado conforme previsto é o resultado esperado.

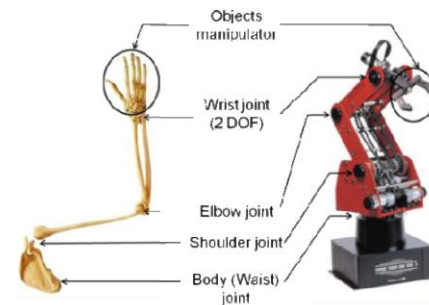


Figura 1. Representação das Juntas do ED7220C em comparação com o braço biológico. Fonte: [4]

2.2. Simulador Computacional CoppeliaSim

Para a utilização de um modelo 3D que pudesse representar de forma gráfica a movimentação do manipulador real, foi utilizado o software de simulação computacional CoppeliaSim em sua versão 4.5.1 – rev. 4 Education, onde utilizou-se da cena correspondente à montagem do ED7220C (Figura 2), obtida através do download do arquivo de extensão ‘.ttx’.

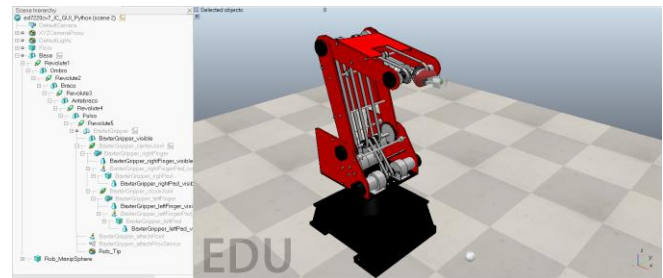


Figura 2. A esquerda a árvore de componentes da montagem do Manipulador ED7220C que está à direita. Fonte: Dados da pesquisa, 2024

Como será melhor descrito posteriormente, para realizar a comunicação entre o software externo ao programa que foi desenvolvido ao longo deste projeto e o próprio CoppeliaSim [5] foi utilizada a funcionalidade de “Remote API” [6] disponibilizada entre os arquivos base que são instalados juntamente com o software em “... \Program Files \CoppeliaRobotics \CoppeliaSimEdu \programming \legacyRemoteApi”. O pacote “Remote API” permite que programas externos controlem e interajam com simulações do CoppeliaSim, facilitando a comunicação entre o software de simulação e outras aplicações, como scripts em Python, C/C++, Java, MatLab, entre outros.

2.3. Linguagem de Programação Python

Para ambas as vertentes aqui propostas, a linguagem de programação Python (versão 3.11.2) foi a principal tecnologia utilizada para o seu desenvolvimento. A escolha desta linguagem justifica-se devido ao fato de possuir diversos recursos que facilitam a implementação das funcionalidades requeridas no desenvolvimento de cada aplicação que será descrita posteriormente. Dentre estes recursos, destacam-se:

- **Simplicidade e Legibilidade:** Python é conhecido por sua sintaxe de leitura simples [7], o que facilita a escrita e a manutenção do código. Isso é particularmente útil em projetos de hardware, onde a clareza do código pode ajudar a diagnosticar e corrigir problemas mais rapidamente.
- **Bibliotecas e Pacotes:** Esta linguagem de programação conta com uma vasta gama de bibliotecas e pacotes que são amplamente suportados pela comunidade, facilitando a implementação de funcionalidades e integrações que em outros cenários, traria um grau de complexidade muito maior ao projeto. Um grande exemplo disso é a biblioteca “*pySerial*” [8] que está sendo utilizada neste projeto para facilitar a comunicação entre o programa em Python com dispositivos que se comunicam pela porta serial com a máquina, no caso específico, uma placa Arduino e o próprio ED7220C. Outro exemplo importante a ser citado é a utilização do pacote *RemoteAPI* disponibilizado com o Coppeliasim para que seja realizada a comunicação com o software de simulação.
- **Ferramentas Gráficas:** É possível utilizar bibliotecas que permitem a criação de interfaces gráficas para interagir com os dados processados no programa de forma facilitada, como a *Tkinter* [9], que está sendo implementada aqui como uma das ferramentas principais para o desenvolvimento da Interface Gráfica Para o Manipulador ED7220C.
- **Multiplataforma:** O Python é uma linguagem interpretada, desta forma, tendo sua execução funcionando bem em diversos ambientes e sistemas operacionais [10].

2.4. Dados dos Sensores

Assim como citado anteriormente, para o desenvolvimento do sistema de Mimetização via Interface Homem-Máquina Para o Manipulador ED7220C, foram utilizados dados providos de uma

série de sensores que realizam a leitura de sinais de natureza eletromiográfica e cinética realizados por um braço biológico real e processados numa placa Arduino Uno, sendo eles:

- **AD8232:** Este sensor é um módulo que foi utilizado para monitoramento da atividade elétrica em um determinado ponto do braço biológico. Este sensor extrai, amplifica e filtra sinais biopotenciais envolvidos em ruídos, facilitando a captura de um sinal claro [11].
- **MPU6050:** O sensor MPU6050 é um dispositivo que combina um acelerômetro que mede a aceleração em três eixos (X, Y, Z) e um giroscópio que realiza a medição da rotação em 3 eixos, fornecendo medições de movimento em seis graus de liberdade (6 DoF) de forma precisa a respeito do movimento e orientação do dispositivo [12].

A figura 3 mostra o circuito de captação dos sinais de movimentos, composto pelo Arduino Uno, o AD8232 e os dois MPU6050.

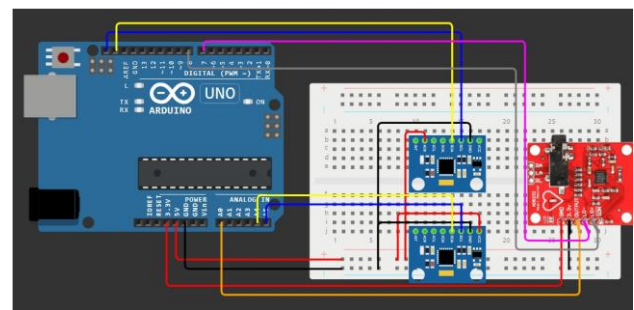


Figura 3. Conexões do AD8232 e MPU6050 ao Arduino Uno. Fonte: Mileib, Kaleb Aquino. Carvalho, Guilherme Matheus Sabino de. Romão, Matheus; 2024 [13].

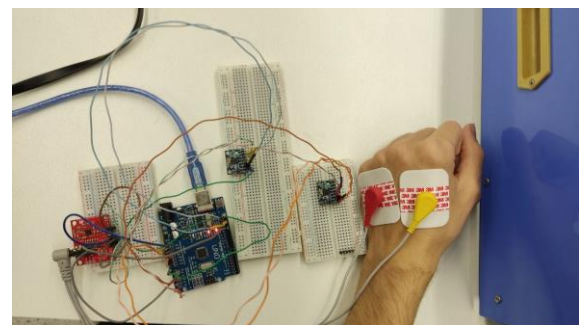


Figura 4. Conexões do AD8232 (e terminais posicionados no braço humano) e 2 MPU6050 ao Arduino Uno. Fonte: Teste prático da mimetização de movimentos por parte do ED7220C; 2024

Ao longo do trabalho desempenhado por [13] foi realizado um extenso trabalho envolvendo

localização dos sensores para captação de sinais (Figura 4) e processamento dos mesmos utilizando de diversas técnicas como o filtro de Kalman implementadas em um software Arduino que teve como resultado final o envio destes dados captados, processados e filtrados via porta serial pela placa e que serão, neste projeto, manipulados de forma a serem utilizados para definir a movimentação do braço ED7220C.

3. Metodologia

Assim como foi citado anteriormente, em ambas as vertentes de desenvolvimento a principal tecnologia utilizada foi a linguagem de programação Python, onde por meio desta foi possível realizar a conexão com o CoppeliaSim a partir do pacote “Remote API” para que houvesse o controle do modelo tridimensional simulado do ED7220C, bem como a utilização da biblioteca “pySerial” para receber dados e enviá-los via porta serial ao manipulador real.

Levando em consideração que ambas as vertentes dependem de uma conexão estabelecida tanto com o CoppeliaSim quanto com o manipulador ED7220C, algumas técnicas serão comuns entre os softwares desenvolvidos, enquanto outras serão individuais a cada aplicação.

3.1. Controle do Modelo Virtual no CoppeliaSim

O primeiro passo para o desenvolvimento das aplicações aqui propostas é possibilitar plena interação com o modelo virtual presente no CoppeliaSim. Para isso, serão implementados à aplicação Python alguns dos recursos disponibilizados pelo próprio simulador, presentes no pacote “Remote API” que foi descrito anteriormente [6] sendo as seguintes funcionalidades:

- **simxStart:** Inicia um thread de comunicação com o servidor do CoppeliaSim.
- **simxFinish:** Fecha quaisquer comunicações antes estabelecidas para garantir o bom funcionamento da aplicação.
- **simxGetObjectHandle:** Recupera o identificador de um objeto da cena ativa no software.
- **simxSetJointTargetPosition:** Define a posição da junta referenciada nos parâmetros da função (obtida pela *simxGetObjectHandle*).

Além disso, é estabelecido no lado do servidor a porta de conexão onde haverá a comunicação com os softwares externos. Desta forma, na função principal da

cena ativa no CoppeliaSim, a “*function sysCall_init()*”, utilizou-se a sintaxe “*simRemoteApi.start(19998)*” para definir a porta de comunicação sendo a ‘19998’ [14].

Desta forma, a estrutura básica do algoritmo de interação da aplicação Python com a simulação do CoppeliaSim é basicamente, em resumo, composta pelo estabelecimento da comunicação com o modelo, a captura dos dados de cada junta que se deseja controlar e então, o envio dos valores desejados para cada junta. Assim, pode-se estabelecer que a Figura 5 representa bem um resumo da utilização do plugin Remote API integrado a aplicações Python.

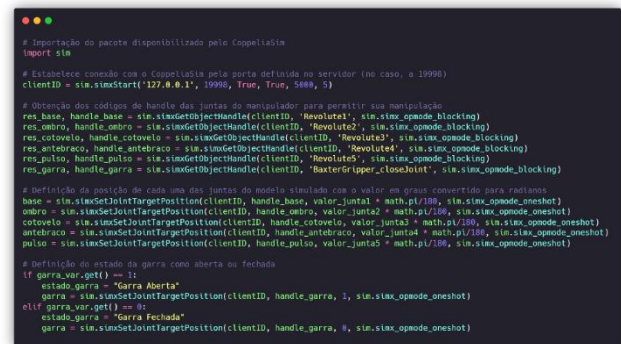


Figura 5. Representação da implementação do Remote API na aplicação Python para controle do modelo do ED7220C. Fonte: Dados da pesquisa, 2024 / Escrito utilizando a ferramenta online ‘carbon.now.sh’

Vale ressaltar que a imagem acima é uma estrutura representativa da lógica que foi implementada nas aplicações Python, de forma que nas funções de definição da posição de cada junta, o valor em si é o parâmetro ‘valor_juntaX’ que é convertido de graus para radiano segundo a Equação 1, e para a garra, haverão apenas os estados de aberta (1) ou fechada (0).

$$rad = graus \cdot \frac{\pi}{180} \quad (\text{Equação 1})$$

3.2. Controle do Manipulador ED7220C

O manipulador ED7220C opera por meio de comandos recebidos através da sua conexão serial, possuindo uma configuração estabelecida de fábrica, onde a sintaxe específica definida pelo fabricante (Minipa) deve ser rigidamente seguida para que os comandos enviados ao manipulador resultem nos comportamentos esperados. Os comandos utilizados serão melhor descritos posteriormente.

Tratando-se da aplicação Python, levando-se

em conta que a comunicação serial é essencial para que haja a devida interação entre os comandos solicitados e a resposta do manipulador, foram utilizados os recursos disponibilizados através da biblioteca “pySerial” para estabelecer esta comunicação. Os principais recursos utilizados nas aplicações desempenhadas neste projeto foram os de abertura da porta serial correspondente à ocupada pelo ED7220C e o envio da lista de comandos ao manipulador utilizando a função “write()” da biblioteca referente [8]

A respeito do envio dos comandos a serem executados pelo manipulador, deve-se seguir de forma rígida as regras estabelecidas pelo manual do fabricante [15], onde os principais comandos utilizados no presente projeto foram os seguintes:

- **TH:** Garante a permissão de controle ao host, que no caso, é a máquina onde a aplicação Python está sendo executada e onde o ED7220C está plugado.
- **PD:** Define a posição absoluta de um motor específico a partir da quantidade de passos fornecidos juntamente com os parâmetros do comando.
- **PR:** Define a posição relativa de um motor específico a partir da quantidade de passos fornecidos juntamente com os parâmetros do comando.
- **GO:** Realiza a abertura da garra.
- **GC:** Realiza o fechamento da garra.
- **MC:** Movimenta todos os motores de forma coordenada seguindo as regras estabelecidas através dos comandos enviados pela porta serial.

Estabeleceu-se de forma empírica que os comandos enviados ao manipulador pela aplicação Python através da porta serial deveriam antes serem codificados em uma sequência de bytes no padrão de formatação UTF-8 [16] definido ainda que para que os comandos sejam de fato enviados (após o envio dos caracteres ‘MC’), deve-se inserir uma quebra de linha através da sequência de caracteres correspondente, seja o ‘\n’ ou ‘\r’.

As instruções de movimento seguem a sintaxe de “PD, nome do motor, quantidade de passos”, e para identificar quais juntas devem ser movidas ao enviar o comando, o fabricante atribui uma letra do alfabeto a cada um dos motores presentes no equipamento. Além disso, foi estabelecida também uma faixa de movimentação em graus para cada um dos motores, onde por meio de medições realizadas de forma empírica, foi possível estabelecer uma relação entre as medidas em graus e os passos correspondentes. As informações correspondentes às suas respectivas juntas

podem ser visualizadas na Tabela 1.

Tabela 1. Informações sobre as juntas do ED7220C.

Junta	Descrição	Faixa Operacional (°)	Faixa Operacional (passos)	Relação de passos por ângulo em graus
B	Junta de Rotação do pulso	$+165^{\circ} \leq B \leq -40^{\circ}$	$+1500 \leq B \leq -1500$	$\alpha > 0, 9$ passos $\alpha < 0, 37$ passos
C	Junta de Rotação do pulso	$+165^{\circ} \leq C \leq -40^{\circ}$	$+3000 \leq C \leq -900$	18 ~ 22 passos
D	Junta de Rotação do Cotovelo	$+55^{\circ} \leq D \leq -5^{\circ}$	$+2000 \leq D \leq -200$	35 ~ 45 passos
E	Junta de Rotação do Ombro	$+10^{\circ} \leq E \leq -60^{\circ}$	$+450 \leq E \leq -2100$	35 ~ 45 passos
F	Junta de Rotação da Base	$+135^{\circ} \leq F \leq -135^{\circ}$	$+6500 \leq F \leq -6500$	48 ~ 50 passos

Ou seja, a estrutura básica para realizar o controle do manipulador ED7220C, considerando comandos enviados com medidas em graus, é em resumo de transferir o controle para o host com o comando “TH”, e criar uma mensagem do tipo lista de *strings* contendo todos os comandos direcionados a cada uma das juntas e seu respectivo número de passos a serem movimentados, bem como o comando para abrir ou fechar a garra, tendo ao final o indicativo “MC” para consolidar a movimentação e um caractere de quebra de linha para enviar os comandos que serão antes, criptografados como uma sequência de bytes com formatação UTF-8. A Figura 6 ilustra a sintaxe para a rotina básica de controle do manipulador conforme foi definido anteriormente.

```

# Importação da Biblioteca PySerial
import serial
from serial import Serial

# Estabelecimento de conexão com o ED7220C na porta COM3
arm_ED7220C = serial.Serial(
    port='COM3',
    baudrate=9600,
    timeout=1
)

# Envio do comando "TH" para transferir o controle para o host (a máquina rodando a aplicação)
command = "TH\n"
arm_ED7220C.write(command.encode())

# Estruturação da mensagem com os comandos de movimentos dados a todas as juntas
mensagem = f"
    PD,F,(passos_base),
    PD,E,(passos_ombro),
    PD,D,(passos_cotovelo),
    PD,C,(passos_antebraço),
    PD,B,(passos_pulso),
    MC,
    status_garra # GO ou GC
"

# Envio da lista de comandos ao manipulador ED7220C
arm_ED7220C.write(mensagem + '\n'.encode())

```

Figura 6. Representação da implementação da comunicação via porta serial com o manipulador ED7220C. Fonte: Dados da pesquisa, 2024 / Escrito utilizando a ferramenta online ‘carbon.now.sh’.

3.3. Desenvolvimento da Interface Gráfica para o ED7220C

Para o desenvolvimento da proposta de interface gráfica como um software Python, foi utilizada a biblioteca *tkinter*, que é a biblioteca padrão do Python para a criação de GUI (*Graphical User Interface* (Interfaces Gráficas de Usuário)) nesta linguagem, oferecendo uma maneira simples e eficiente de desenvolver aplicativos com janelas, botões, campos de texto e outros elementos gráficos [9].

```
import tkinter as tk
from tkinter import ttk, messagebox

# Cria a janela principal
root = tk.Tk()
root.title("ED-7220C")

# Variáveis de controle para os sliders e botões
slider1_var = tk.IntVar()
slider2_var = tk.IntVar()
garra_aberta_var = tk.IntVar(value=-1) # Inicialmente indefinido (-1)

# Adiciona um label para o slider1
label_slider1 = ttk.Label(root, text="Ângulo Base (graus):")
label_slider1.grid(row=0, column=0, padx=10, pady=10, sticky=tk.W)

# Adiciona um slider (Scale) associado à variável de controle slider1 = Base = Junta F
slider1 = tk.Scale(root, from_=-180, to=180, orient='horizontal', variable=slider1_var)
slider1.grid(row=0, column=1, padx=10, pady=10, sticky="ew")

# Adiciona um label para o slider2
label_slider2 = ttk.Label(root, text="Ângulo ombro (graus):")
label_slider2.grid(row=1, column=0, padx=10, pady=10, sticky=tk.W)

# Adiciona um slider (Scale) associado à variável de controle slider2 = Ombro = Junta E
slider2 = tk.Scale(root, from_=-21, to=54, orient='horizontal', variable=slider2_var)
slider2.grid(row=1, column=1, padx=10, pady=10, sticky="ew")

# Adiciona um label descritivo para os botões
label_botaoes = ttk.Label(root, text="Controle da Garra:")
label_botaoes.grid(row=5, column=0, colspan=2, padx=10, pady=(20, 5))

# Botão "Garra Aberta"
botao_abrir_garra = ttk.Button(root, text="Garra Aberta", command=garra_aberta)
botao_abrir_garra.grid(row=6, column=0, padx=10, pady=10)

# Botão "Garra Fechada"
botao_fechar_garra = ttk.Button(root, text="Garra Fechada", command=garra_fechada)
botao_fechar_garra.grid(row=6, column=1, padx=10, pady=10)

# Botão "Enviar" que chama a função "enviar_valores" e executa o envio ao CoppeliaSim e ED7220C
botao_enviar = ttk.Button(root, text="Enviar", command=enviar_valores)
botao_enviar.grid(row=7, column=0, colspan=2, padx=10, pady=10)

# Executa o loop principal
root.mainloop()

def move_coppelia(moveSet):
    # CONTROLE DOS VALORES DE MOVIMENTO DA BASE E OMBRO DE GRAUS PARA RADIANOS
    baseCoppelia = moveSet[0] * (math.pi / 180)
    print("BaseCoppeliaValue: (baseCoppelia)")
    shoulderCoppelia = moveSet[1] * (math.pi / 180)
    print("ShoulderCoppeliaValue: (shoulderCoppelia)")

    # ENVIO DO COMANDO VALIDO DE MOVIMENTAÇÃO PARA A BASE
    if (abs(moveSet[0]) < 3):
        returnCodeBase = sim.simGetObjectHandle(clientID, "Revolute1", sim.sim_opmode_blocking)
        returnCodeBase = sim.simSetJointTargetPosition(clientID, ed7220_Base, baseCoppelia, sim.sim_opmode_oneshot)
    else:
        print("A Base real permanece imóvel")

    # ENVIO DO COMANDO VALIDO DE MOVIMENTAÇÃO PARA O OMBRO
    if (abs(moveSet[1]) < 3):
        returnCodeShoulder = ed7220_Shoulder = sim.simGetObjectHandle(clientID, "Revolute2", sim.sim_opmode_blocking)
        returnCodeShoulder = sim.simSetJointTargetPosition(clientID, ed7220_Shoulder, shoulderCoppelia, sim.sim_opmode_oneshot)
    else:
        print("O Braço real permanece imóvel")

    # CONTROLE DE ABERTURA DO FECHAMENTO DA GARRA
    res_gripper, handle_gripper = sim.simGetObjectHandle(clientID, "BaxterGripper.claseJoint", sim.sim_opmode_blocking)
    gripper = sim.simSetJointTargetPosition(clientID, handle_gripper, moveSet[2], sim.sim_opmode_oneshot)
```

Figura 7. Trechos do código em Python para implementação da interface gráfica com *tkinter*.

Fonte: Dados da pesquisa, 2024 / Escrito utilizando a ferramenta online ‘carbon.now.sh’.

Dentro do algoritmo, assim como evidenciado na Figura 7, para o desenvolvimento da aplicação gráfica que realiza o controle do manipulador ED7220C real e no ambiente virtual do CoppeliaSim

simultaneamente, tratando-se da utilização da biblioteca *tkinter*, primeiramente foi criada a janela principal com dimensões responsivas aos elementos internos. Em seguida, foram declaradas as variáveis correspondentes aos valores medidos em graus dos ângulos requeridos em cada uma das juntas do mecanismo, que serão definidas a partir do arraste de elementos de slide na interface gráfica. Para controle do estado da garra (aberta ou fechada) foi utilizado um sistema de seleção por botões, assim como o mecanismo de envio dos comandos agregados, que é por meio de um botão “Enviar”.

Ao enviar os dados, a função responsável é chamada, onde ela capta o valor atual de cada um dos slides com a função “*get()*” e armazena em uma variável, que será enviada como parâmetro de valor convertido em radianos à junta correspondente do modelo simulado no CoppeliaSim seguindo as diretrizes e o modelo estipulado na seção 3.1. Os valores serão também enviados à cada uma das juntas do ED7220C via comandos nos padrões do fabricante pela porta serial, assim como demonstrado na seção 3.2 e com os valores convertidos em passos, assim como visto na Tabela 1. O estado da garra será definido conforme a opção selecionada na interface gráfica.

3.4. Desenvolvimento da Interface Gráfica para o ED7220C

Assim como foi citado anteriormente, no desenvolvimento da aplicação Python responsável por replicar os movimentos de um braço biológico no manipulador ED7220C, são recebidos como dados de entrada os resultados obtidos a partir do trabalho realizado em [13].

Estes dados consistem na leitura em tempo real dos sensores listados na seção 2.4, que enviam, a cada iteração, um buffer de dados para cada propriedade de cada um dos sensores, onde especificamente serão utilizadas as informações a respeito das acelerações (Figura 8) e a leitura de sinais eletromiográficos (Figura 9).

A aplicação em questão tem como objetivo realizar a mudança de posição do ED7220C de acordo com o movimento dos sensores dentro do mais próximo possível do que possa ser considerado como “em tempo real”, sendo assim, tanto a aplicação Arduino que está lendo, tratando e enviando os dados brutos, quanto a aplicação em Python que está trabalhando e manipulando estes dados para consolidar as

movimentações requeridas, estão em execução de forma constante. Desta forma, para descrever a lógica de funcionamento da aplicação em questão deve ser definido o seguinte:

- **Lista de Buffers:** Deve-se levar em consideração que há 13 fontes de dados de entrada no programa, sendo eles os 3 eixos dos sensores de acelerômetro e giroscópio de cada um dos MPU6050 (3 eixos $\times$ 2 sensores $\times$ 2 MPU6050 = 12 dados), e por fim, a leitura do AD8232. No algoritmo do Arduino, foi definido que para separar cada um dos 13 buffers de dados, é inserido um valor de pico entre eles. Desta forma, na aplicação Python há um laço de repetição de loop infinito “while True:” para garantir o funcionamento constante, e inserido nele está o programa principal, que se inicia com a lógica de identificação do valor recebido como o pico definido ou não. Caso não seja o pico definido, ele irá realizar a inclusão dos valores no buffer correspondente ao dado em questão até que seja detectado o pico. Quando o pico é definido, o programa muda o buffer de dados a ter os valores incluídos, e esta sequência segue até completar os 13 buffers de dados, e então a sequência recomeça, por exemplo: Supondo que são enviados 10 valores por buffer, assim que o programa se inicia os primeiros dados recebidos são correspondentes aos dados do eixo X do acelerômetro do primeiro AD8232, quando completam as 10 posições e detecta-se o pico, passa-se então a inserir dados no buffer do eixo Y do mesmo acelerômetro, e assim por diante até finalizar todos os buffers de dados, armazenando-os em uma lista nomeada aqui de ‘moveSet’ e então manipulando estes dados e enviando os comandos correspondentes, tal qual será descrito a seguir.

- **Manipulações Realizadas nos Buffers de Dados:** Com todos os buffers de dados devidamente preenchidos, são acionadas então as seções do programa que irão fazer com que estes dados sejam de fato utilizáveis. Primeiramente levando em consideração que os 13 dados estão armazenados como buffers, deve-se então convertê-los em um único valor inteiro que possa representar todo o conjunto de cada uma das informações. Para isso, foi utilizado o arredondamento da média aritmética para cada um dos conjuntos de dados, formando assim uma nova lista ‘moveSet’ contendo valores inteiros.

Após a obtenção dos valores individuais, é necessário realizar a conversão dos valores obtidos em cada eixo dos acelerômetros dos sensores MPU6050

para uma unidade de medida utilizável no contexto em questão, que no caso será convertido para radianos (BEZERRA NETO, 2023) e posteriormente, para graus, conforme evidenciado nas equações seguintes.

$$\theta_x = \arctan \left(\frac{Ac_x}{\sqrt{Ac_y^2 + Ac_z^2}} \right) \cdot \frac{180}{\pi} \quad (\text{Equação 2})$$

$$\theta_y = \arctan \left(\frac{Ac_y}{\sqrt{Ac_x^2 + Ac_z^2}} \right) \cdot \frac{180}{\pi} \quad (\text{Equação 3})$$

$$\theta_z = \arctan \left(\frac{\sqrt{Ac_x^2 + Ac_y^2}}{Ac_z} \right) \cdot \frac{180}{\pi} \quad (\text{Equação 4})$$

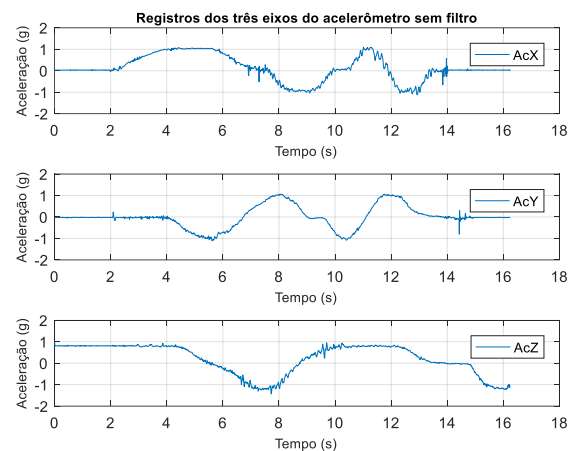


Figura 8. Aceleração nos eixos X, Y e Z obtidas de um MPU6050 em movimento. Fonte: Mileib, Kaleb Aquino. Carvalho, Guilherme Matheus Sabino de. Romão, Matheus; 2024 [13]

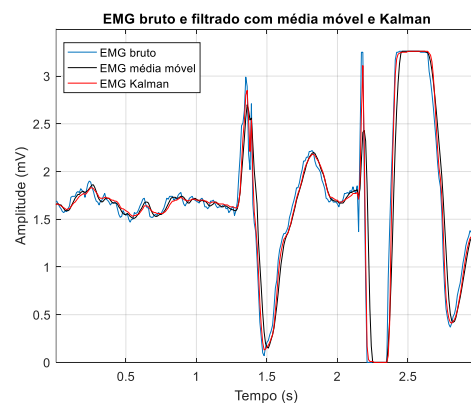


Figura 9. Variação na leitura de sinais eletromiográficos obtidos por um AD8232. Fonte: Mileib, Kaleb Aquino. Carvalho, Guilherme Matheus Sabino de. Romão, Matheus; 2024 [13]

A estrutura básica do processo aqui descrito desde a recepção dos dados dos sensores até a conversão para um valor em ângulos codificado na linguagem de programação Python encontra-se ilustrada na Figura 10.

```
# Append Values to The MoveSet History
for i in range(len(moveSet)):
    moveSetHistory[i].append(moveSet[i])

acix = []
acy = []
aciz = []
gyroix = []
gyroy = []
gyroz = []
ac2x = []
ac2y = []
ac2z = []
gyro2x = []
gyro2y = []
gyro2z = []
emgl = []

if data != '':
    match list_count:
        case 1:
            acix.append(float(data))
        case 2:
            acy.append(float(data))
        case 3:
            aciz.append(float(data))
        case 4:
            gyroix.append(float(data))
        case 5:
            gyroy.append(float(data))
        case 6:
            gyroz.append(float(data))
        case 7:
            ac2x.append(float(data))
        case 8:
            ac2y.append(float(data))
        case 9:
            ac2z.append(float(data))
        case 10:
            gyro2x.append(float(data))
        case 11:
            gyro2y.append(float(data))
        case 12:
            gyro2z.append(float(data))
        case 13:
            emgl.append(float(data))
```

Figura 10. Representação de trechos da implementação em Python das leituras dos sensores. Fonte: Dados da pesquisa, 2024.

Vale ressaltar que para o valor do EMG, é realizado um cálculo de média aritmética que é arredondado, onde caso seja arredondado para 1, a garra abrirá, e caso seja 0, ela fechará.

Manipulação e envio dos dados para a simulação no CoppeliaSim: A partir da conversão dos dados para seu respectivo valor representativo, pode-se então realizar o envio dos comandos correspondentes à movimentação do ED7220C. Para tal, desenvolveu-se uma estrutura aos moldes do que foi evidenciado na seção 3.1, onde primeiramente é estabelecida a conexão com o endereço IP e porta ativa da simulação no CoppeliaSim, em seguida é realizado o envio dos dados de posição. Vale ressaltar que foi estabelecida a diretriz de que caso a movimentação realizada seja menor do que 3 rad., o algoritmo não deve considerar como um movimento válido. Além disso, os dados foram convertidos da medida de graus para radianos, já que é com esta medida que a simulação no CoppeliaSim atua. Nota-se que no presente algoritmo foi realizado o controle apenas das articulações ‘Revolute2’ (ombro), ‘Revolute1’ (base) e controle de abertura e fechamento da garra, a fins de simplificar a visualização da resposta do braço robótico nos experimentos práticos de observação. A estrutura básica da seção no algoritmo

Python correspondente à função de controle do modelo simulado no CoppeliaSim está evidenciada na Figura 11.

```
def move_coppelia(moveSet):
    # CONVERSÃO DOS VALORES DE MOVIMENTO DA BASE E OMBRO DE GRAUS PARA RADIANOS
    baseCoppelia = moveSet[0] * (math.pi / 180)
    print(f'BaseCoppeliaValue: {baseCoppelia}')
    shoulderCoppelia = moveSet[1] * (math.pi / 180)
    print(f'ShoulderCoppeliaValue: {shoulderCoppelia}')

    # ENVIAR DO COMANDO VÁLIDO DE MOVIMENTAÇÃO PARA A BASE
    if abs(moveSet[0]) > 3:
        returnCodeBase = sim.simsGetObjectHandle(clientID, 'Revolute1', sim.slm_opmode_blocking)
        returnCodeBase = sim.simsSetJointTargetPosition(clientID, ed7220_Base, baseCoppelia, sim.slm_opmode_oneshot)
    else:
        print('A Base real permanece imóvel')

    # ENVIAR DO COMANDO VÁLIDO DE MOVIMENTAÇÃO PARA O OMBRO
    if abs(moveSet[1]) > 3:
        returnCodeShoulder = sim.simsGetObjectHandle(clientID, 'Revolute2', sim.slm_opmode_blocking)
        returnCodeShoulder = sim.simsSetJointTargetPosition(clientID, ed7220_Shoulder, shoulderCoppelia, sim.slm_opmode_oneshot)
    else:
        print('O Braço real permanece imóvel')

    # CONTROLE DE ABERTURA DO FECHAMENTO DA GARRA
    res_gripper = sim.simsGetObjectHandle(clientID, 'BaxterGripper_closeJoint', sim.slm_opmode_blocking)
    gripper = sim.simsSetJointTargetPosition(clientID, handle_gripper, moveSet[12], sim.slm_opmode_oneshot)
```

Figura 11. Controle via software Python do modelo simulado do manipulador ED7220C no CoppeliaSim. Fonte: Dados da pesquisa, 2024 / Escrito utilizando a ferramenta online ‘carbon.now.sh’.

Manipulação e envio dos dados para o manipulador ED7220C: Com os valores convertidos para a medida em graus e a conexão devidamente estabelecida com a porta serial do manipulador, é definida então uma função que irá utilizá-los para realizar o controle da garra e das juntas E (ombro) e F (base) do manipulador ED7220C.

```
def move_ed7220c(moveSet):
    # CONVERSÃO DOS VALORES DAS JUNTAS 'E' E 'F' DE GRAUS PARA PASSOS DO BRAÇO ROBÓTICO
    jointFMovement = moveSet[0] * 40
    jointEMovement = moveSet[1] * 35

    if abs(jointFMovement) > 180 or abs(jointEMovement) > 180:
        # ESCOLHA DO COMANDO DE MOVIMENTAÇÃO DA JUNTA 'F' (BASE) EM CASO DE VALOR VÁLIDO
        if abs(jointFMovement) > 180:
            if jointFMovement < -180 and jointEMovement < 180:
                print('Comando Feito (Base)')
                command = f'F {jointFMovement}'
                arse7220c.write(command.encode())
            else:
                print('O movimento solicitado na Base excede os limites físicos do robô')
        else:
            print('A Base real permanece imóvel')

    # ESCOLHA DO COMANDO DE MOVIMENTAÇÃO DA JUNTA 'E' (OMBRIO) EM CASO DE VALOR VÁLIDO
    if abs(jointEMovement) > 180:
        if jointEMovement < -180 and jointFMovement < 180:
            print('Comando Feito (Braço)')
            command = f'E {jointEMovement}'
            arse7220c.write(command.encode())
        else:
            print('O movimento solicitado no Braço excede os limites físicos do robô')
    else:
        print('O Braço real permanece imóvel')

    # ENVIAR DO COMANDO DE MOVIMENTO PARA SEREM DE FATO EXECUTADOS PELO ED7220C
    print('Enviando dados (Base e Braço)')
    command = 'MCLN'
    arse7220c.write(command.encode())

    # CONTROLE DE ABERTURA E FECHAMENTO DA GARRA
    if moveSet[12] == 0.5:
        print('Comando Feito (Garra)')
        command = 'gpn'
        arse7220c.write(command.encode())
    elif moveSet[12] == 0.1:
        print('Comando Feito (Garra)')
        command = 'gpn'
        arse7220c.write(command.encode())
```

Figura 12. Representação da implementação do controle via software Python do manipulador ED7220C. Fonte: Dados da pesquisa, 2024 / Escrito utilizando a ferramenta online ‘carbon.now.sh’.

Nesta função, primeiramente há a conversão dos valores em graus para o correspondente em quantidade de passos, uma vez que o manipulador em questão necessita da passagem do parâmetro de

movimentação nesta medida para que a atuação ocorra conforme esperado. Para tal conversão, utilizou-se dos dados presentes na coluna ‘relação de passos por ângulo em graus’ da Tabela 1. Vale ressaltar que foram considerados como valores válidos apenas aqueles que excediam 100 passos de movimentação. Em seguida, foram escritos os comandos de movimentação das juntas correspondentes de acordo com a validação dos seus valores, e por fim, o envio de tais comandos via a solicitação de execução ao manipulador. Por fim, analisa-se se o valor correspondente a garra encontra-se no estado de aberto (0) ou fechado (1) e então é executado o comando correspondente. Vale ressaltar que todos os comandos foram escritos e codificados de acordo com as diretrizes evidenciadas na seção 3.2. A estrutura da seção no algoritmo Python correspondente à função de controle do manipulador ED7220C está evidenciada na Figura 12.

4. Resultados

Após o desenvolvimento das aplicações especificadas na seção “3. Metodologia”, foi possível alcançar resultados razoavelmente satisfatórios dentro das propostas definidas para este projeto.

A respeito da interface gráfica, ao definir os valores em ângulos nos *sliders* e enviar para a função especificada controlar o modelo virtual e o ED7220C real, a movimentação ocorre de forma extremamente condizente com o que foi especificado, possuindo um comportamento de movimentação observável bastante preciso.

Desta forma, a utilização deste software em experimentos práticos mostrou-se significativamente interessante, uma vez que possibilita um controle rápido e preciso dos movimentos do robô ED7220C, refletindo estes movimentos de maneira fiel no modelo virtual, facilitando a execução de experimentos com visualização gráfica dos resultados quase em tempo real.

A Figura 13 apresenta, à esquerda, a interface gráfica em destaque. É possível observar os *sliders* para ajustes dos ângulos das juntas, os botões para abertura e fechamento da garra e o botão para enviar os comandos para o simulador e o manipulador real. À direita são apresentadas, como exemplo, duas configurações ajustadas para o manipulador. É possível observar que as posições na tela do CoppeliaSim e na bancada do laboratório são coerentes.

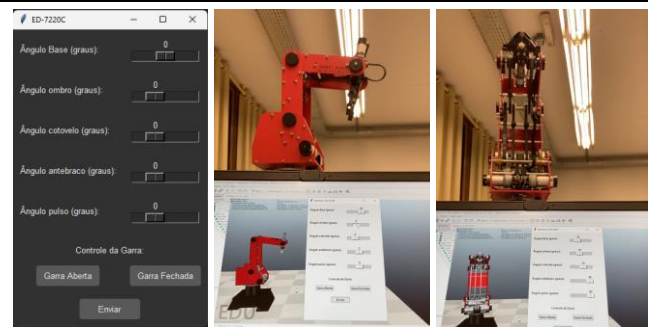


Figura 13. Interface gráfica resultante (à esquerda) e demonstração da sincronização de movimentos entre o modelo simulado e o braço robótico real. Fonte: Experimento realizado com o professor orientador em laboratório

A respeito da aplicação de controle do ED7220D acionado pelas leituras providas dos sensores, houve a obtenção de um modelo razoavelmente responsivo e que é capaz de realizar os movimentos das juntas testadas durante os experimentos de forma consideravelmente condizentes com as mudanças angulares aplicadas aos sensores.

A Figura 14 mostra a montagem da interface homem-máquina, na qual, tanto o simulador, quanto o manipulador real respondem aos sinais provenientes do circuito composto pelo Arduino e pelos sensores AD8232 e MPU6050. À direita é possível observar que a mão está fechada, o que provocou o fechamento da garra.

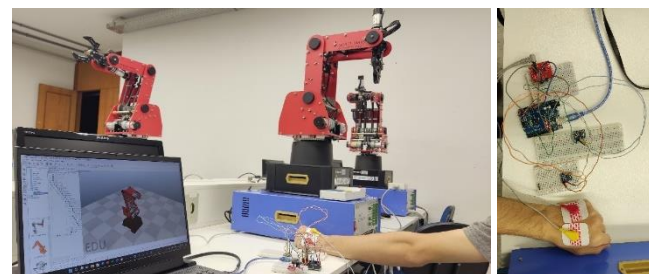


Figura 14. Fotografias representativas de todo o sistema de funcionamento do modelo que mimetiza movimentos reais. Fonte: Experimento realizado com o professor orientador em laboratório

É válido ressaltar que ambas as aplicações tem como característica centralizar o processamento de dados, conexões estabelecidas e o núcleo de controle dos componentes finais (braço robótico e modelo simulado) num mesmo software Python, o que otimiza significativamente o nível de complexidade e

manutenção dos modelos, além de reduzir o risco de falhas por incompatibilidades que poderiam vir a acometer o funcionamento do sistema.

5. Conclusão

Levando em conta a natureza das propostas estabelecidas para este projeto, pode-se afirmar que os resultados obtidos são satisfatórios e atendem as expectativas das propostas, onde a aplicação de “Interface Gráfica Para o Manipulador ED7220C” apresentou um grande potencial para, posteriormente, se tornar uma grande ferramenta auxiliar no aprendizado dos estudantes de robótica que fazem uso deste tipo de equipamento para adquirir conhecimentos práticos a respeito do funcionamento do mesmo.

Além disso, a aplicação “Mimetização via Interface Homem-Máquina Para o Manipulador ED7220C” apresenta o princípio de uma vasta gama de possibilidades a serem exploradas em pesquisas futuras para aplicação deste tipo de tecnologia tanto na indústria com robôs controlados por movimentação humana, quando em ramificações da bioengenharia, como a de cirurgias remotas e desenvolvimento de próteses e equipamentos de reabilitação.

Embora os resultados obtidos tenham se mostrado interessantes, ainda há muito espaço para melhorias em ambas as aplicações, como por exemplo:

- Interface permitir a mudança da posição de forma absoluta, relativa, ou via velocidade;
- Interface gráfica passar a funcionar em tempo real ao invés de depender do botão “enviar”;
- Tornar o design de interface gráfica mais amigável ao usuário;
- Implementar gráficos de monitoramento dos dados dos sensores em tempo real;
- Melhoria no tempo de resposta de forma geral, de comunicações e respostas;
- Otimização no mapeamento / relacionamento das juntas do ED7220C com os dados providos dos sensores, visando atender da melhor forma possível aos 6 graus de liberdade que este equipamento permite utilizar.

Todos estes aspectos deverão ser trabalhados posteriormente como pontos de melhoria nas aplicações, viabilizando ainda mais a implementação de cada uma delas nos contextos a que se destinam.

Referências

- [1] CARRARA, V. **INTRODUÇÃO À ROBÓTICA INDUSTRIAL**. [s.l.: s.n.]. Disponível em: <<http://urlib.net/8JMKD3MGP3W34P/3K5JPL8>>.
- [2] MOHAN, A. et al. Telesurgery and Robotics: An Improved and Efficient Era. *Cureus*, 26 mar. 2021.
- [3] SONO, T. S. P. **Projeto de uma prótese de mão sub-atuada: mecanismo, interface e sistema de controle**. 2012. Pós Graduação em Engenharia Mecânica, Escola de Engenharia, Universidade Federal de Minas Gerais, Belo Horizonte, MG.
- [4] IQBAL, J. et al. **Modeling and analysis of a 6 DOF robotic arm manipulator** *Canadian Journal on Electrical and Electronics Engineering*. [s.l.: s.n.]. Disponível em: <<https://www.researchgate.net/publication/280643085>>.
- [5] **regular API reference**. Disponível em: <<https://manual.coppeliarobotics.com/en/apiFunctions.htm>>. Acesso em: 1 jul. 2024.
- [6] **Remote API functions (Python)**. Disponível em: <<https://manual.coppeliarobotics.com/en/remoteApiFunctionsPython.htm>>. Acesso em: 1 jul. 2024.
- [7] **Python Tutorials – Real Python**. Disponível em: <<https://realpython.com/>>. Acesso em: 2 jul. 2024.
- [8] **Welcome to pySerial’s documentation – pySerial 3.0 documentation**. Disponível em: <<https://pythonhosted.org/pyserial/>>. Acesso em: 2 jul. 2024.
- [9] **tkinter – Python interface to Tcl/Tk – Python 3.12.4 documentation**. Disponível em: <<https://docs.python.org/3/library/tkinter.html>>. Acesso em: 2 jul. 2024.
- [10] **Python Geral – Documentação Python 3.14.0a0**. Disponível em: <<https://docs.python.org/pt-br/dev/faq/general.html>>. Acesso em: 2 jul. 2024.
- [11] **AD8232 ECG Module Pinout, Interfacing with Arduino, Applications**. Disponível em: <<https://microcontrollerslab.com/ad8232-ecg-module-pinout-interfacing-with-arduino-applications-features/>>. Acesso em: 2 jul. 2024.
- [12] **How To Use An MPU6050 With Arduino - Makerguides.com**. Disponível em: <<https://www.makerguides.com/how-to-use-an-mpu6050-3-axis-accelerometer-and-3-axis-gyrosensor-with-arduino/>>. Acesso em: 2 jul. 2024.
- [13] MILEIB, K. A.; CARVALHO, G. M. S. DE; ROMÃO, M. **Protótipo para captação e processamentos de sinais eletromiográficos e de movimento**. [s.l.: s.n.].
- [14] **Enabling the remote API - server side**. Disponível em: <<https://manual.coppeliarobotics.com/en/remoteApiServerSide.htm>>. Acesso em: 1 jul. 2024.

[15] ROBOT TRAINER WITH ED-MK4 COMMAND SET MANUAL ED-7220C. . [s.l: s.n.].

[16] **Built-in Types — Python 3.12.4 documentation**. Disponível em: <<https://docs.python.org/3/library/stdtypes.html#str.encode>>. Acesso em: 2 jul. 2024.

[17] BEZERRA NETO, H. P. Use of Arduino for monitoring angles on oil platforms for didactics purposes. **Diversitas Journal**, v. 8, n. 2, p. 840–854, 2023.

[18] BANDARA, D. S. V. et al. Development of a multi-DoF transhumeral robotic arm prosthesis. **Medical Engineering & Physics**, v. 48, p. 131-141, 2017/10/01/ 2017. ISSN 1350-4533. Disponível em: <<http://www.sciencedirect.com/science/article/pii/S135045331730173X>>.