UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA ELÉTRICA
GRADUAÇÃO EM ENGENHARIA DE CONTROLE E AUTOMAÇÃO

ADRIEL LUIZ MARQUES

PROPOSTA DE DESENVOLVIMENTO DE UMA PLATAFORMA ROBÓTICA PARA USO EDUCACIONAL

UBERLÂNDIA
2022

ADRIEL LUIZ MARQUES

PROPOSTA DE DESENVOLVIMENTO DE UMA PLATAFORMA ROBÓTICA PARA USO EDUCACIONAL

Trabalho de Conclusão de Curso de Engenharia de Controle e Automação da Universidade Federal de Uberlândia - UFU - Campus Santa Mônica, como requisito para a obtenção do título de Graduação em Engenharia de Controle e Automação.

Universidade Federal de Uberlândia–UFU

Faculdade de Engenharia Elétrica

UBERLÂNDIA

2022

ADRIEL LUIZ MARQUES

PROPOSTA DE DESENVOLVIMENTO DE UMA PLATAFORMA ROBÓTICA PARA USO EDUCACIONAL

Trabalho de Conclusão de Curso de Engenharia de Controle e Automação da Universidade Federal de Uberlândia - UFU - Campus Santa Mônica, como requisito para a obtenção do título de Graduação em Engenharia de Controle e Automação.

Universidade Federal de Uberlândia–UFU

Faculdade de Engenharia Elétrica

Uberlândia, 2022

Banca Examinadora:

_____

Prof. Dr. Renato Santos Carrijo
Orientador

_____

Prof. Dr. Renato Ferreira Fernandes Jr
Membro

_____

Prof. Dr. Marcelo Barros de Almeida
Membro

Dedico este trabalho aos meus pais, pelo
apoio, carinho e motivação.

AGRADECIMENTOS

Agradeço primeiramente a Deus, o qual nunca esteve distante e é a razão de toda a esperança.

Aos meus pais, André e Carolina, que sempre fizeram todo o possível para que este momento se realizasse. Às minhas irmãs, Lorraine e Luana, que sempre estiveram ao meu lado me auxiliando.

Agradeço ao meu professor e orientador Renato Santos Carrijo, o qual auxiliou o meu progresso dentro da Universidade anos antes deste projeto ser iniciado e sempre orientou o meu trabalho de forma profissional e dedicada.

Agradeço ao Laboratório de Acionamentos Sistemas Eletrônicos e Controle (LASEC) da Universidade Federal de Uberlândia e ao seu coordenador, o professor Renato Ferreira Fernandes Jr. Neste local desenvolvi pesquisas de iniciação científica que contribuíram para este projeto e encontrei o espaço físico necessário para o desenvolvimento deste trabalho.

Agradeço também a todos os amigos que tive a felicidade de encontrar nessa jornada, os quais tornam os meus dias mais significativos.

“A gratidão é a consciência de que ninguém vai a qualquer lugar significativo sozinho.”

(Ed René Kivitz)

RESUMO

Este trabalho propõe o desenvolvimento de uma plataforma de robótica para uso educacional no nível de graduação. Tendo em vista que o estudo da área de robótica é facilitado quando se utiliza simulações que permitem observar a aplicação dos conceitos teóricos, além do uso de equipamentos físicos que permitem analisar a influência das diversas variáveis presentes no mundo real, este projeto apresenta o desenvolvimento de um *software* central com interface gráfica, a elaboração da conexão deste *software* com o programa de simulação V-REP e a construção de um robô físico controlado por uma placa embarcada que se comunica com o *software* central utilizando o protocolo MQTT.

O *software* foi desenvolvido na linguagem Python com a utilização da biblioteca PyQt5 para a construção da interface gráfica. A placa embarcada foi projetada para a utilização de um módulo ESP32, o qual gerencia o acionamento dos servo motores do robô e possibilita a comunicação com o *software* central via MQTT. O *firmware* para o chip ESP32 foi desenvolvido em C/C++ utilizando o ambiente Arduino. Deste modo, é possível observar o desenvolvimento de uma plataforma muito abrangente, a qual pode auxiliar em estudos na área de robótica e no desenvolvimento de aplicações futuras.

Palavras-chave: robótica; automação; simulação de robôs; software com interface gráfica; sistema embarcado; ESP32.

ABSTRACT

This project aims the development of a robotics platform for educational use in undergraduate courses. In the study of robotics, it is easy to observe the benefits of the use of simulations that allow an application of theoretical concepts, in addition to the use of equipment that allows analyzing the influence of the various variables present in the real world. Therefore, this project presents the development of central software with a graphical interface, the elaboration of the connection of this software with the simulation program V-REP and the construction of a physical robot controlled by an embedded board that communicates with the central software using the protocol MQTT.

The software was developed in Python language using the PyQt5 library to build the graphical interface. The embedded system was designed to use an ESP32 module, which drives the servomotors to move the robot and enables communication with the software using MQTT. The firmware for the ESP32 chip was developed in C/C++ using the Arduino framework. In this way, it is possible to observe the development of a very comprehensive platform, which can help in robotics studies and the development of future applications.

Keywords: robotics; automation; robot simulation; software with graphical interface; embedded system; ESP32.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

API	*Application Programming Interface*
CA	Corrente Alternada
CC	Corrente Contínua
CSS	*Cascading Style Sheets*
HTML	*HyperText Markup Language*
IFR	*International Federation of Robotics* (Federação Internacional de Robótica)
LED	*Light Emitting Diode*
MQTT	*Message Queuing Telemetry Transport*
OTA	*Over-the-air*
PCB	Placa de Circuito Impresso
PWM	Modulação por Largura de Pulso
QoS	Qualidade de Serviço
SoC	*System-on-a-chip* (Sistema-em-um-chip)
UML	*Unified Modeling Language*

SUMÁRIO

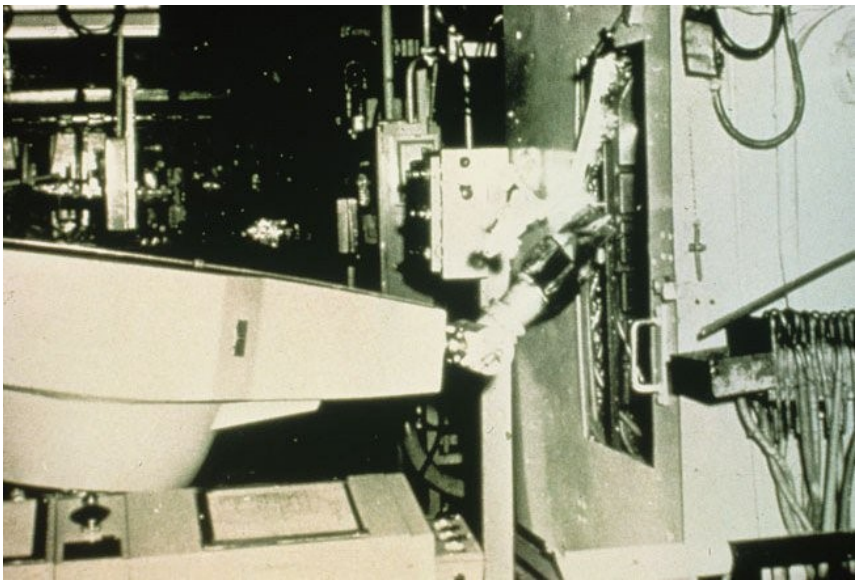
1 INTRODUÇÃO

A utilização de máquinas para o auxílio de atividades humanas ou simplesmente para o entretenimento é realizada desde tempos remotos. Os gregos há dezenas de séculos, desenvolveram estátuas operadas hidraulicamente, as quais não tinham finalidades práticas além do entretenimento e do exercício didático. Contudo, a idealização do conceito moderno de robôs é recente (PAZOS, 2002).

O termo "robô" foi criado somente em 1921, por Karen Capek e deriva das palavras *robota* e *robotnik* de origem eslava, que significam "trabalho forçado" e "servo", respectivamente. Curiosamente, o inventor do termo era um dramaturgo e o criou para uma peça teatral chamada "Os Robôs Universal de Russum" (MATARIC, 2014; ROMANO, 2002).

Todavia, o conceito moderno de robô foi elaborado no início da década de 1960. Chamado de UNIMATE, o primeiro robô moderno foi desenvolvido pela empresa UNIMATION Inc e instalado na Ford Motor Company. O mesmo tinha a função de auxiliar na linha de montagem automotiva (PAZOS, 2002; ROMANO, 2002). Na Imagem 1 é apresentada a foto deste robô:

Imagem 1 – Robô UNIMATE



Fonte: IFR (2022).

Os benefícios industriais da utilização de robôs foram observados logo nos primeiros anos. Em 1969, a General Motors Corporation utilizou os seus primeiros robôs de solda a

ponto na sua fábrica em Lordstown (Estados Unidos) e observou que com o uso de robôs foi possível automatizar mais de 90% das operações de soldagem de carrocerias, enquanto que a porcentagem de automação das fábricas tradicionais variava de 20% a 40% (IFR, 2022).

Portanto, tem-se observado que a utilização de robôs traz diversos benefícios para a produção industrial, tais como: melhora da qualidade do produto, pois os robôs podem executar tarefas cumprindo uma precisão determinada; capacidade de operar em ambientes perigosos; melhora do gerenciamento da produção, pois em um sistema automatizado as tarefas de obter e organizar informações são simplificadas (PAZOS, 2002).

Tendo em vista os benefícios trazidos pela robótica para a sociedade, o presente trabalho propõe a criação de uma plataforma educacional de robótica, composta por um ambiente de simulação de robótica e outro de acionamento de robôs físicos, de modo a auxiliar no estudo deste tema e no desenvolvimento de novas aplicações.

1.1 Justificativa

Este trabalho tem como motivação auxiliar no desenvolvimento de profissionais e projetos na área de robótica, de modo com que os benefícios obtidos pela utilização de robôs no ambiente industrial sejam difundidos, especialmente no cenário brasileiro.

De acordo com dados do relatório desenvolvido pela International Federation of Robotics (IFR) no ano de 2020, o número de robôs instalados no Brasil era de aproximadamente 15 mil unidades, número muito inferior ao encontrado em outros países do continente americano, como Estados Unidos que contam com mais de 293 mil unidades e México que possui cerca de 40 mil unidades (IFR, 2020).

Portanto, tendo em vista a importância da robótica para a produção industrial, o presente trabalho visa auxiliar no desenvolvimento de estudos e ferramentas na área da robótica, incentivando a participação de novos indivíduos neste setor.

1.2 Objetivos

Este trabalho tem como tema a robótica aplicada em ambiente educacional no nível de graduação. Diante disso, os objetivos deste projeto são:

- Desenvolver uma plataforma que, empregada em cursos de graduação, auxilie no ensino da disciplina de robótica e outras disciplinas aderentes.
- Elaborar um software que forneça uma interface gráfica para permitir que o usuário acesse um ambiente de simulação de robótica e outro ambiente de acionamento de robôs físicos.
- Construir uma interligação entre a interface gráfica e um simulador de robótica.
- Elaborar cenas no simulador de robótica que permita o usuário visualizar conceitos de robótica como rotação, translação e cinemática direta.
- Desenvolver uma placa embarcada que permita o acionamento de um braço robótico e se comunique por meio de um protocolo industrial.
- Elaborar um *firmware* que controle o funcionamento do sistema embarcado e se comunique com o software de interface gráfica.

2 REFENCIAL TEÓRICO

Neste capítulo são apresentados os conceitos fundamentais para a compreensão deste trabalho. Portanto, são abordados os seguintes conceitos: Robótica, Softwares de simulação e Protocolos industriais.

2.1 Robótica

O termo “robótica” refere-se ao estudo de robôs. Portanto, estuda a capacidade de um robô “sentir e agir no mundo físico de forma autônoma e intencional” (MATARIC, 2014). A primeira utilização do termo “robótica” com base no termo “robô” da obra de Capek é creditada ao escritor Isaac Asimov, que em 1940 para criou o conceito para “designar a ciência que se dedica ao estudo dos robôs” (ROMANO, 2002).

No ambiente industrial, há diversas definições para o termo “robô”. Pazos (2002) define robô como “máquina automática programável”. O termo “máquina” refere-se à conversão de energia em trabalho útil. Além disso, a expressão “automática” indica que a sua energia de trabalho não se origina do operador. Por fim, o termo “programável” adiciona a ideia de que a sua operação depende de um conjunto de instruções previamente elaboradas (programa) que pode ser alterado de acordo com a atividade.

A norma técnica ISO (International Organization for Standardizatio)10218 elaborada no ano de 2006, na sua primeira parte, define as diretrizes centrais para robôs industriais e define robô como "uma máquina manipuladora com vários graus de liberdade controlada automaticamente, reprogramável, multifuncional" (ISO 10218, apud PAZOS, 2002).

Desse modo, observa-se que dois atributos básicos para a categorização de um robô é a sua capacidade de realizar tarefas de forma automática (sem a necessidade de um operador) e a sua capacidade de reprogramação para atender atividades diversas. Portanto, a utilização de robôs na indústria é fundamental para permitir a produção de bens variados e em médios e pequenos lotes (ROMANO, 2002).

2.1.1 Aspectos físicos

Fisicamente, um robô é basicamente constituído por elos (estruturas rígidas) que são interligados entre si por juntas, formando uma cadeia. As juntas interligam pares de elos vizinhos. Para que haja a movimentação do robô, um atuador (como um motor elétrico) deve

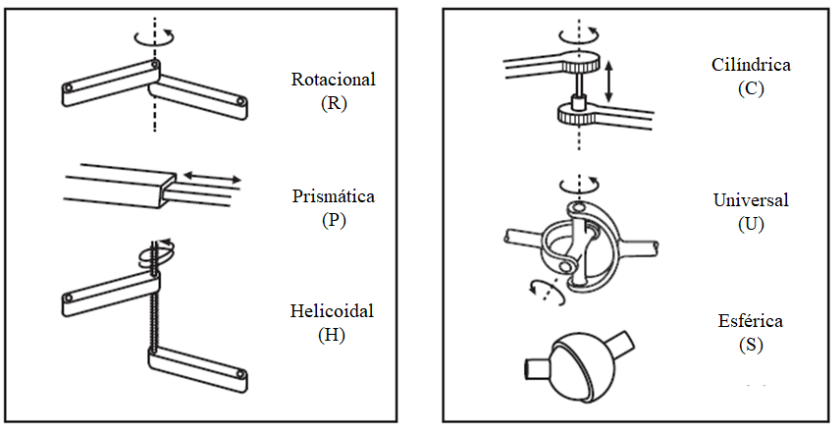
fornecer o torque necessário para a movimentação dos elos. Com o fim de possibilitar o controle da posição, velocidade ou força do robô, sensores devem ser adicionados na estrutura. Por fim, um efetuador (como uma garra ou ferramenta) é associado para que o robô possa realizar algum tipo de trabalho em objetos externos (CRAIG, 2013; LYNCH; PARK, 2017).

Os elos do robô são projetados para apresentarem alta rigidez aos esforços de torção e flexão e alguma flexibilidade quando submetidos a esforços provenientes das tarefas que estão exercendo. Portanto, o alumínio e o aço são os materiais comumente utilizados na fabricação destes itens (ROMANO, 2002).

As juntas do robô são os elementos essenciais para a determinação do número de graus de liberdade de um robô. De acordo com Romano (2002), o número de graus de liberdade representa o "número de variáveis independentes de posição que precisam ser especificadas para se definir a localização de todas as partes do mecanismo, de forma inequívoca".

Na Imagem 2 são apresentados os seis principais tipos de juntas. A junta de revolução (R) permite movimentos sobre o eixo da junta. A junta prismática (P) possibilita uma movimentação retilínea na direção do eixo da junta. A junta helicoidal (H) permite, simultaneamente, a rotação e a translação sobre o seu eixo. Todos esses três tipos de juntas apresentam um grau de liberdade. Todavia, há juntas que apresentam mais de um grau de liberdade. A junta cilíndrica (C) possui dois graus de liberdade e permite rotações e translações sobre um ponto fixo do seu eixo. A junta universal (U) também apresenta dois graus de liberdade e consiste de um par de juntas de revolução postos ortogonalmente. Por fim, a junta esférica (S), apresenta três graus de liberdade a atua semelhantemente a um ombro humano (LYNCH; PARK, 2017).

Imagem 2 – Tipos de juntas de um robô

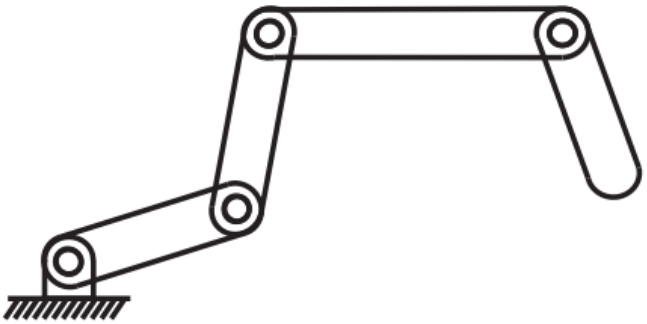


Fonte: Adaptado de LYNCH; PARK (2017).

Ademais, outro conceito necessário para definir o número de graus de liberdade de um robô é a distinção entre dois tipos de mecanismo: cadeia cinemática aberta e cadeia cinemática fechada. Um mecanismo de cadeia cinemática aberta é aquele em que a disposição de seus componentes forma um ciclo completo. Um exemplo desse tipo de mecanismo é uma pessoa sentada com os dois pés no chão (um ciclo pode ser traçado do chão ao pé esquerdo, do pé esquerdo ao direito, e do pé direito de volta ao chão). Um mecanismo de cadeia cinemática aberta é aquele em que a disposição dos seus componentes não forma um ciclo fechado (LYNCH; PARK, 2017).

Desse modo, considerando que geralmente um robô industrial é formado por uma combinação de elos e juntas que forma uma cadeia cinemática aberta, o número de graus de liberdade de todo o robô equivale à soma do número de graus de liberdade de cada junta (ROMANO, 2002). Desse modo, um robô como o apresentado na Imagem 3, composto por uma cadeia cinemática aberta e quatro juntas de revolução apresentará quatro graus de liberdade.

Imagem 3 – Exemplo de robô com quatro graus de liberdade



Fonte: LYNCH; PARK (2017).

A informação do número de graus de liberdade de um robô é importante para determinar se ele consegue alcançar qualquer ponto dentro de um espaço limitado em seu entorno. Conforme demonstrado por Lynch; Park (2017), um corpo rígido (objeto cuja distância entre quaisquer dois pontos no seu interior é constante) possui seis graus de liberdade. Portanto, caso um robô tenha menos de seis graus de liberdade, é possível que o seu efetuador não atinja uma posição e uma orientação arbitrária dentro do seu espaço de trabalho.

Os atuadores de um robô são os elementos que convertem energia elétrica, hidráulica ou pneumática, em potência mecânica para movimentar o robô. Os atuadores hidráulicos utilizam um fluido incompressível (óleo hidráulico) para realizar trabalho, desta forma é possível implementar controles contínuos e acurados de posicionamento, porém o controle da força é instável. Já os atuadores pneumáticos utilizam um fluido compressível (ar comprimido) para realizar trabalho, desta forma é possível realizar operações suaves, porém, o controle do posicionamento é pouco preciso (ROMANO, 2002).

Outro grupo de atuadores são os eletromagnéticos, composto principalmente por motores de corrente contínua e motores de passo. Este é o grupo de atuadores mais utilizados em robôs, apresentando como vantagens a grande variedade de fabricantes, a possibilidade de controle de força e posição (quando associado a sensores) e a facilidade de programação (ROMANO, 2002).

Os sensores podem ser definidos como “dispositivos físicos que permitem a um robô perceber seu ambiente físico, a fim de obter informações sobre si mesmo e sobre os objetos que o cercam” (MATARIC, 2014). Com as informações dos sensores, o robô pode controlar aspectos do seu funcionamento, como posição e velocidade. Os sensores mais comuns para controlar a posição do robô são o potenciômetro (sensor resistivo) e o encoder (dispositivo que avalia a passagem de pulsos luminosos que passam por um disco com marcações) (SICILIANO et al., 2009).

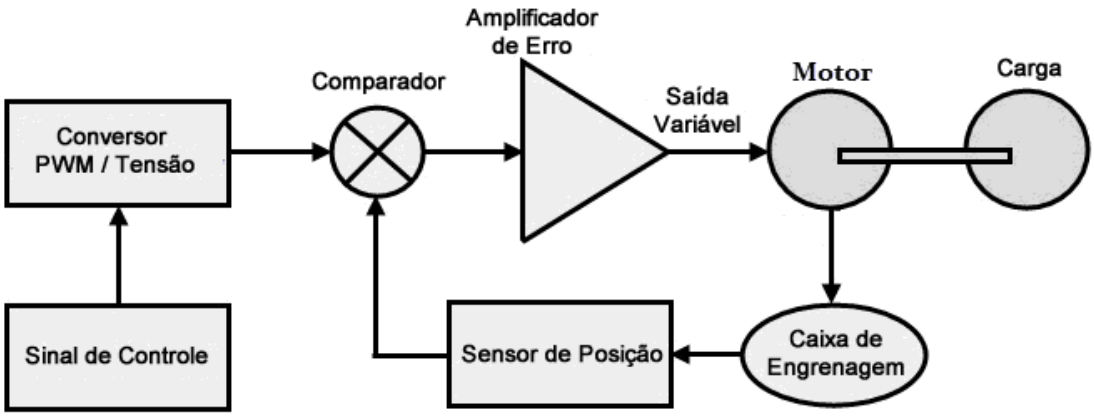
Uma alternativa para se utilizar um único componente com a função de atuador e sensor é o uso de servo motores. Um servo motor é um equipamento que une um motor elétrico e um sensor de posição em um mesmo encapsulamento, de forma que, internamente, a leitura da posição momentânea é utilizada para controlar a velocidade e a posição final do motor (SILVEIRA, 2016).

De acordo com a natureza da alimentação, os servo motores podem ser de dois tipos: corrente alternada (CA) e corrente contínua (CC). Os servos CC são empregados em atividades que exigem menor potência (menos que 500 W) e, geralmente, tem um custo reduzido. Já os servos CA, são utilizados em aplicações que demandam maior potência e precisão, além de menor taxa de manutenção (SILVEIRA, 2016).

O funcionamento detalhado de um servo motor segue conforme o descrito na Imagem 4. Um sinal de entrada, geralmente um valor de tensão modulado com determinada frequência, técnica conhecida como Modulação por Largura de Pulso (PWM), é aplicado em um circuito conversor, o qual transmite o sinal convertido para um circuito comparador. Dessa forma o sinal de entrada é comparado com o sinal lido do sensor de posição gerando

um erro. O valor do erro é amplificado e utilizado para mover o servo motor para a posição desejada (SILVEIRA, 2016).

Imagem 4 – Sistema de controle de um servo motor



Fonte: Silveira (2016).

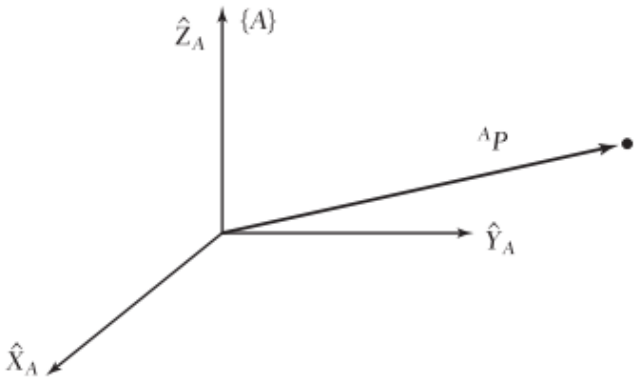
Portanto para movimentar um servo motor para uma posição desejada, basta enviar um sinal PWM com a largura de pulso correspondente à posição almejada e o circuito interno do servo motor irá movê-lo, realizando o menor caminho possível da posição inicial até a final. Para o servo motor MG995, utilizado neste projeto, o período base para avaliação do pulso de entrada é de 20 milissegundos. Um pulso com largura de 1 milissegundo irá movimenta-lo para a posição de 0 grau. Já um pulso de 2 milissegundos irá movimentá-lo para a posição de 180 graus. O mesmo vale para os valores intermediários (“MG995”, [s.d.]).

2.1.2 Descrições espaciais

A movimentação dos robôs no espaço implica, naturalmente, na necessidade de representar a posição e a orientação de determinados pontos do corpo do robô. Desse modo, é necessário estabelecer sistemas de coordenadas e as transformações que permitem movimentar um ponto ou vetor nesses sistemas (CRAIG, 2013).

A posição de um ponto P no espaço pode ser representada por um vetor associado a um sistema de coordenadas qualquer, denotado por {A}. Para indicar em qual sistema de coordenadas o ponto está representado, utiliza-se a notação ^{A}P (CRAIG, 2013). Na Imagem 5 é representado esse ponto:

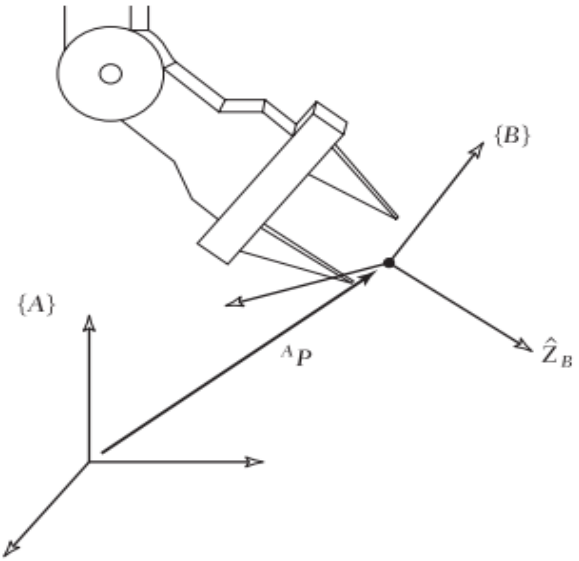
Imagem 5 – Representação do vetor ^{A}P



Fonte: Craig (2013).

Para se determinar o lugar de um objeto no espaço, como a garra de robô, é necessário especificar a sua posição e a sua orientação. Por conveniência, a posição do ponto do corpo escolhido para ser descrever o objeto é definida como a origem do sistema de referência fixado ao corpo (CRAIG, 2013). Na Imagem 6 é representado um ponto P com relação ao sistema {A}, o qual pode ser descrito por sua posição, vetor ^{A}P (o índice A indica o sistema a qual o vetor se refere), e a orientação de seu sistema de coordenadas {B}.

Imagem 6 – Localização da posição e orientação de um objeto



Fonte: Craig (2013).

Conforme apresentado por Craig (2013), a movimentação de um ponto no espaço por uma distância finita ao longo de uma dada direção vetorial é chamada de translação. A

translação de vetor ${}^{A}P_1$ para outro ${}^{A}P_2$ realizada pelo operador translação $D_Q(q)$ é apresentada na equação 1:

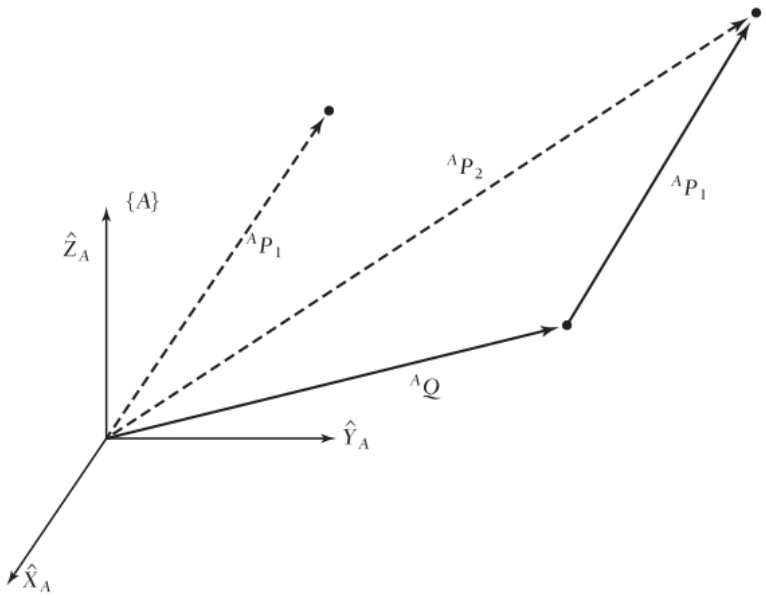
$$ {}^{A}P_2 = D_Q(q) * {}^{A}P1 \quad (1) $$

Na equação 1, q representa a magnitude da translação ao longo da direção vetorial $\hat{Q}$. O operador $D_Q(q)$ pode ser interpretado como uma transformação homogênea da forma apresentada na equação 2, onde q_x, q_y, e q_z são componentes do vetor translação Q e $q = \sqrt{{q_x}^2 + {q_x}^2 + {q_x}^2}$ (CRAIG, 2013).

$$ D_Q(q) = \begin{bmatrix} 1 & 0 & 0 & q_x \\ 0 & 1 & 0 & q_y \\ 0 & 0 & 1 & q_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2) $$

Na Imagem 7 é apresentada a translação de um vetor ${}^{A}P_1$ para outro ${}^{A}P_2$ realizada pelo operador translação $D_Q(q)$:

Imagem 7 – Operador translacional



Fonte: Craig (2013).

Outro operador importante é o rotacional, que transforma o vetor ${}^{A}P_1$ para outro vetor ${}^{A}P_2$ por meio de uma rotação R (CRAIG, 2013). Essa operação é descrita na equação 3:

$$^{A}P_2 = R_k(\theta) * {}^{A}P1 \quad (3)$$

O operador rotacional $R_k(\theta)$ realiza uma rotação em torno do eixo direção $\hat{K}$ em (θ) graus (CRAIG, 2013). O mesmo pode ser descrito nas direções dos eixos X, Y e Z de acordo com as equações 4, 5 e 6, respectivamente:

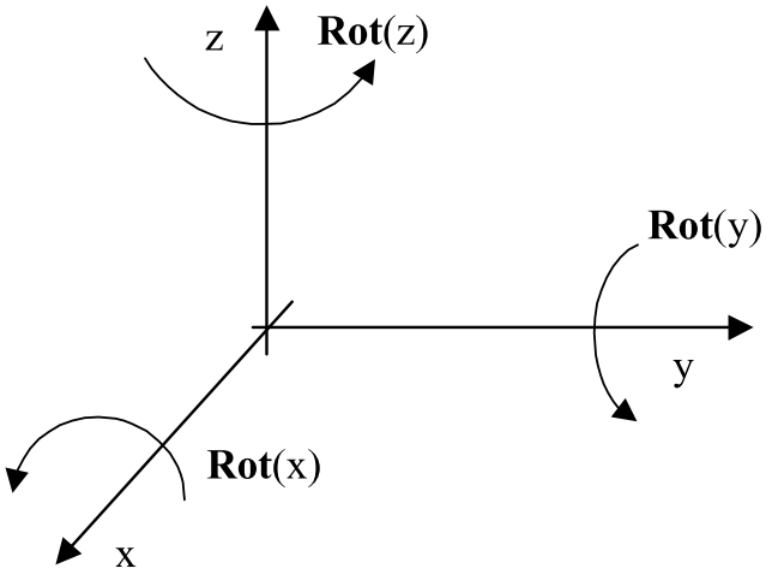
$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\Theta) & -\text{sen}(\Theta) & 0 \\ 0 & \text{sen}(\Theta) & \cos(\Theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (4)$$

$$R_y(\theta) = \begin{bmatrix} \cos(\Theta) & 0 & \text{sen}(\Theta) & 0 \\ 0 & 1 & 0 & 0 \\ -\text{sen}(\Theta) & 0 & \cos(\Theta) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (5)$$

$$R_z(\theta) = \begin{bmatrix} \cos(\Theta) & -\text{sen}(\Theta) & 0 & 0 \\ \text{sen}(\Theta) & \cos(\Theta) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (6)$$

Na Imagem 8 é apresentada a rotação na direção dos eixos X, Y e Z:

Imagem 8 – Operador rotacional



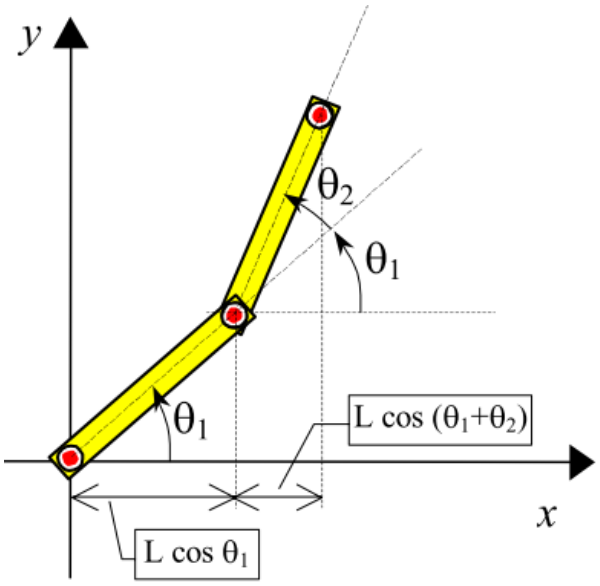
Fonte: Santos (2004).

2.1.3 Cinemática

De acordo com Santos (2004), a cinemática de um robô é "o estudo do conjunto de relações entre as posições, velocidades e acelerações dos seus elos". O estudo da cinemática é dividido em duas áreas: cinemática direta (quando se deseja encontrar a localização da posição do efetuador a partir da posição das juntas) e cinemática inversa (quando se deseja determinar as posições das juntas a partir da posição do efetuador). Como a solução para problemas de cinemática inversa nem sempre é unívoca, o que significa que há várias configurações de juntas que resultam na mesma posição do efetuador, este trabalho abordará apenas a cinemática direta.

Em situações nas quais a estrutura do manipulador é simples e o número de juntas é pequeno, é possível adotar uma solução geométrica direta para problemas de cinemática direta (SICILIANO et al., 2009). Na Imagem 9 é apresentado um exemplo no qual a abordagem geométrica direta é suficiente:

Imagem 9 – Robô com dois graus de liberdade



Fonte: Santos (2004).

Para se determinar as coordenadas da posição do efetuador do robô de dois graus de liberdade apresentado na Imagem 9, é necessário apenas decompor os vetores que representam os dois elos em relação aos eixos X e Y, resultando nas equações 7 e 8, onde L_1 refere-se ao comprimento do primeiro elo, L_2 o comprimento do segundo elo, e os ângulos Θ_1 e Θ_2, referem-se, respectivamente, aos ângulos formados entre os elos 1 e 2 e o eixo X.

$$x = L_1 * \cos(\theta_1) + L_2 * \cos(\theta_1 + \theta_2) \quad (7)$$

$$y = L_1 * \text{sen}(\theta_1) + L_2 * \text{sen}(\theta_1 + \theta_2) \quad (8)$$

Todavia, em situações na qual a estrutura do robô é complexa e o número de juntas é maior, torna-se necessária a utilização de soluções menos diretas (SICILIANO et al., 2009). Uma alternativa é a utilização do algoritmo de Denavit-Hartenberg apresentado em Santos (2004).

2.2 Protocolos de comunicação

A comunicação do robô com equipamentos externos é um requisito essencial para o seu funcionamento (AMARAN et al., 2015). O uso de protocolos *wireless* (sem fio) é uma abordagem que vem ganhando espaço no ambiente industrial. Entre as suas principais vantagens estão: redução do custo agregado, pois, mesmo que a sua instalação seja dispendiosa, a maior facilidade de expansão e a menor necessidade de manutenção reduz o custo geral; flexibilidade, pois dentro de uma área de cobertura os equipamentos podem ter as suas posições livremente alteradas. Em contrapartida, as suas principais desvantagens são: menor segurança intrínseca, pois os dados são mais suscetíveis a interceptadores; menor taxa de transferência de dados (MATHIAS, 2000).

Duas características importantes para que um protocolo de comunicação seja utilizado é a capacidade de transmitir pacotes de forma rápida, para que a operação do robô seja precisa, e a capacidade de transmitir dados com o menor tamanho de pacote possível, para que seja consumida a menor quantidade de energia possível (AMARAN et al., 2015).

Nesse sentido, o uso do protocolo *Message Queuing Telemetry Transport* (MQTT) é uma opção interessante. De acordo com experimento de envio de pacotes com intervalo de um segundo durante o período de uma hora, o protocolo MQTT apresentou um pacote médio de 78 bytes, enquanto que o protocolo *Hypertext Transfer Protocol* (HTTP), que também é utilizado para comunicação entre máquinas, apresentou um pacote médio de 228 bytes (FRUGOLI; PEREIRA, 2019).

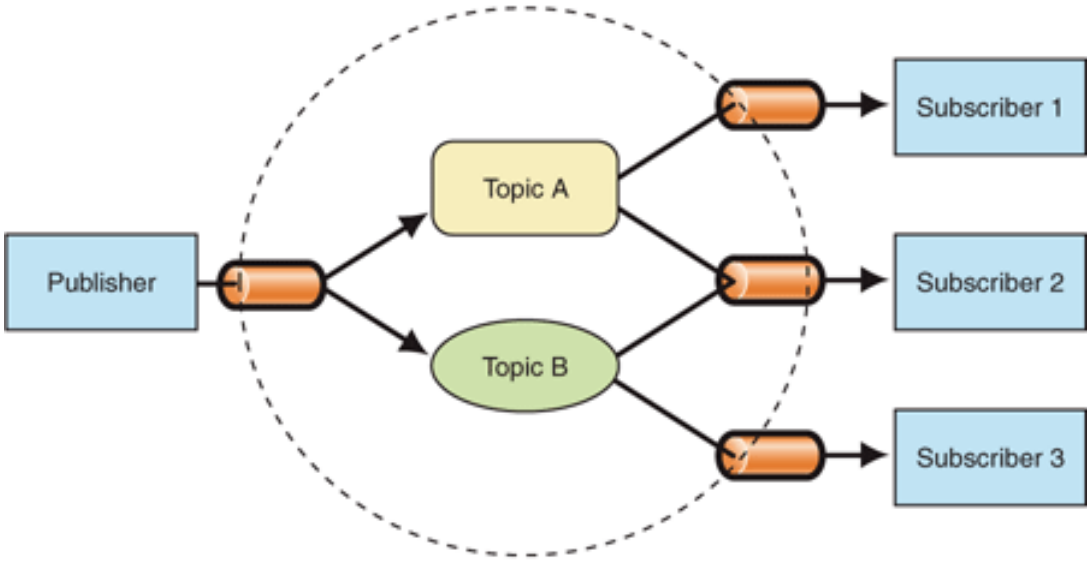
O protocolo MQTT foi criado na década de 1990 pela IBM. O objetivo original era vincular sensores em oleodutos de petróleo a satélites. Apesar de o nome transmitir a ideia

que o protocolo faz alguma espécie de enfileiramento de mensagens, isso não ocorre na prática (YUAN, 2017). Em 2014, o MQTT se tornou um protocolo aberto, gerido pela MQTT ORG. Segundo a organização, o protocolo MQTT pode ser definido como um protocolo "projetado para um transporte de mensagens de *publisher*/*subscriber* extremamente leve, ideal para conectar dispositivos remotos com uma pequena área de cobertura ("MQTT: The Standard for IoT Messaging", 2022).

A transmissão de dados geralmente funciona em cima do protocolo TCP/IP de comunicação. Todavia, há uma especificação derivada, denominada MQTT-SN, que pode utilizar outros tipos de transporte como UDP ou Bluetooth ("MQTT", 2019).

Conforme anteriormente citado, a arquitetura do protocolo MQTT é do tipo *publisher*/*subscriber*. O mesmo é composto por duas entidades: o *broker* (servidor intermediário de mensagens) e o cliente (que envia e assina mensagens do *broker*). A relação entre o *broker* e o cliente é realizada por meio de tópicos. Desse modo, o cliente pode escrever ou assinar a leitura de tópicos que são geridos por um *broker*. Vale ressaltar que um cliente pode escrever ou assinar mais de um tópico durante a comunicação (YUAN, 2017). Na Imagem 10 é apresentada a estrutura de comunicação deste protocolo:

Imagem 10 – Estrutura de comunicação do protocolo MQTT



Fonte: ("MQTT", 2019).

O cabeçalho do protocolo MQTT pode variar de 2 a 5 bytes. No primeiro byte, os quatro primeiros bits referem-se ao tipo de mensagem; o quinto bit é o indicador de mensagem duplicada; os próximos dois bits identificam a QoS (qualidade de serviço) e o

último bit indica se a mensagem deve ser retida no *broker*. Nos próximos quatro bytes é informado o tamanho da mensagem. Por fim é enviada a própria mensagem e não há um padrão para ela (“MQTT”, 2019).

Um item que merece destaque no cabeçalho do MQTT é QoS. Esse campo aceita três valores: 0 indica que a mensagem vai ser enviada no máximo uma vez; 1 indica que a mensagem será enviada ao menos uma vez e 2 indica que a mensagem será enviada exatamente uma vez (“MQTT”, 2019).

2.3 Softwares de simulação

A utilização de robôs físicos no ambiente acadêmico nem sempre é possível, uma vez que o custo dos mesmos é elevado. Uma solução interessante é o uso de simuladores. Simuladores podem ser definidos como “ambientes computacionais que emulam o acontecimento de algum fenômeno real que os usuários conseguem manipular, explorar e experimentar” (JONASSEN, 1996, apud FERNANDES, 2013).

As principais vantagens da utilização de ambientes virtuais para simulação robótica são: baixo custo de desenvolvimento, pois, de início, só há o custo intelectual e não é necessário se fazer investimentos em equipamentos; disponibilidade de um maior acervo de robôs; economia de tempo, pois no ambiente virtual é mais simples a programação de robôs; baixo risco do ambiente, pois o controle do ambiente evita possíveis danos ao robô e ao operador (DE MELO et al., 2019; FERNANDES, 2013).

Diante dos benefícios da utilização de simuladores de robôs, a próxima etapa importante é analisar qual simulador disponível no mercado se adequa melhor para aplicações no ambiente universitário. De acordo com pesquisa desenvolvida por MELO et al. (2019), os simuladores mais utilizados pela comunidade da robótica são: V-REP (também conhecido por CoppeliaSim), Gazebo e Unity.

Visando utilizar os simuladores no ambiente de cursos de graduação, critérios importantes para avaliar a utilidade de um simulador são: custo da licença, usabilidade (quão bem os usuários podem desenvolver as suas necessidades utilizando o sistema) e facilidade de uso. O simulador V-REP apresenta uma licença livre para usos educacionais. O Gazebo é *open-source* (licença Apache 2.0). Já o Unity apresenta licença gratuita para usos educacionais, desde que os lucros obtidos com a aplicação não ultrapassem um limite estabelecido. Em termos de usabilidade, o V-REP e o Unity apresentam as interfaces mais amigáveis e maior liberdade de edição. Por fim, o V-REP e o Gazebo são os que apresentam

maior facilidade de uso (DE MELO et al., 2019). Conclusões semelhantes são apresentadas por (FREESE et at., 2010, apud DE MELO et al., 2019).

Dessa forma, o simulador V-REP foi escolhido para este trabalho. Uma característica interessante que ele apresenta é a portabilidade, apresentando suporte para os sistemas operacionais Windows, Linux e macOS. Além disso, o mesmo apresenta bibliotecas que permitem a programação de robôs em sete linguagens diferentes: Python, Lua, C/C++, Java, JavaScript, Matlab e Octave (“Coppelia Robotics”, 2022; ROHMER; SINGH; FREESE, 2013).

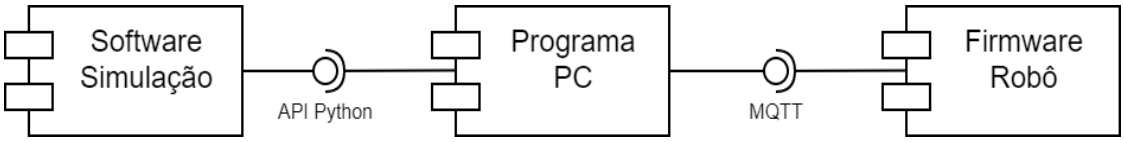
A comunicação do V-REP com softwares externos ocorre por meio das APIs (Application Programming Interface), ou Interface de Programação de Aplicação, em português, fornecidas pelo próprio programa. O V-REP apresenta três tipos de APIs, de acordo com a estratégia de comunicação utilizada. O primeiro, “ZeroMQ remote API” é o mais recente e utiliza a biblioteca ZeroMQ para a comunicação entre programas, estando disponível apenas para as linguagens C++ e Python. O segundo tipo, “WebSocket remote API” utiliza sockets e está disponível para a linguagem JavaScript. Já o terceiro grupo, “Legacy remote API” foi o primeiro a ser desenvolvido, utiliza comunicação por meio de socket e está disponível para C++, Python, Java, Matlab, Octave e Lua (“Remote API”, 2022).

Tendo em vista que algumas APIs do V-REP utilizam sockets para a comunicação entre programas, é necessário entender esse conceito. Um socket é uma interface de comunicação entre processos bidirecional que permite a comunicação entre programas que estejam na mesma máquina ou rede. Aos se utilizar a camada TCP para a comunicação, um socket deve ser vinculado a um endereço de IP e porta (ORACLE, [s.d.]).

3 DESENVOLVIMENTO

Este trabalho apresenta o desenvolvimento de uma plataforma robótica educacional com projeto de *software*, *hardware* e *firmware*. Para facilitar o entendimento das partes que compõem este projeto, na Imagem 11 é apresentado o diagrama de componentes da plataforma proposta. O diagrama segue as orientações da UML (Unified Modeling Language) (SEIDL et al., 2015).

Imagem 11 – Diagrama de componentes



Fonte: O autor.

De acordo com a Imagem 11, observa-se que o componente central do projeto é o programa para computador (*software*), o mesmo se comunica com o *software* de simulação por meio de uma biblioteca em Python, além de utilizar o protocolo MQTT como interface para se comunicar com o *firmware* do robô físico. Os detalhes do desenvolvimento de cada um dos componentes deste projeto são apresentados a seguir.

3.1 Software

O *software* central do projeto foi desenvolvido na linguagem Python. A motivação da escolha dessa linguagem foi a sua simplicidade de programação (principalmente se comparada com linguagens do paradigma funcional, como C), a sua compatibilidade com diferentes sistemas operacionais e a existência de um grande número de bibliotecas, que facilitam o desenvolvimento da interface gráfica e a comunicação com outros componentes.

Para o desenvolvimento da interface gráfica, foi utilizada a biblioteca PyQt5 (“PyQt5 5.15.6”, 2022). Uma característica interessante dessa biblioteca é a possibilidade de se utilizar uma interface gráfica chamada Qt Designer para auxiliar no desenvolvimento da aplicação. Essa interface permite a disposição de elementos como botões e caixas de texto diretamente em uma tela que simula o aplicativo em desenvolvimento, possibilitando uma rápida identificação de como está sendo desenvolvida a interface, além de permitir a alteração de propriedades dos elementos adicionados de forma simples e visual. O arquivo gerado pela

interface do Qt Designer possui uma extensão própria (ui), o qual pode ser traduzido para um código em Python por meio do comando “pyuic5”.

Diante disso, toda a aplicação visual do *software* foi desenvolvida com a utilização da interface Qt Designer e, posteriormente, traduzido o código gerado para a linguagem Python. No script em Python, todos os elementos, como caixas de texto e botões, foram programados para desenvolverem as suas ações. A tela inicial do programa é apresentada na Imagem 12:

Imagem 12 – Tela inicial do programa

Fonte: O autor.

De acordo com a Imagem 12, é possível observar que a tela inicial do programa contém dois botões, os quais permitem acessar as áreas destinadas ao controle do robô virtual e do robô físico. Para simplificar o gerenciamento da aplicação, os botões não abrem novas telas, mas alteraram o elemento exibido pelo objeto “stackedWidget”, de forma que apesar de ser exibida uma única tela, todo o seu conteúdo é alterado de acordo com o botão pressionado. Outra característica importante é que a tela possui um tamanho fixo de 480 x 640 pixels, o que evita a necessidade de alterar o tamanho dos elementos internos ao se modificar o tamanho da tela e facilita a visualização do software de simulação de robótica. Por fim, observa-se que a tela apresenta o título EdBot, o qual faz referência ao termo *Educational*

Robot em inglês (Robô Educacional, em português) e foi utilizado para identificar facilmente o projeto apresentado neste trabalho.

3.1.1 Robô virtual

A conexão entre o programa central desenvolvido e o *software* de simulação V-REP utiliza a biblioteca “Legacy remote API” fornecida pelo próprio simulador. O principal fator que motivou a sua utilização é a existência de um grande número de materiais a respeito da sua utilização, o que facilita o seu uso.

Para utilizar essa API no script em Python, é necessário copiar para a pasta de execução do script três arquivos disponíveis na pasta de instalação do V-REP: o arquivo “sim.py”, que contém todas as funções para comunicação com o simulador; o arquivo “simConst.py”, que contém todas as constantes utilizadas; e o arquivo “remoteApi.dll”, composto pela a biblioteca compilada dinamicamente para o sistema operacional Windows (também estão disponíveis opções para outros sistemas operacionais). É importante ressaltar que é fornecida a biblioteca compilada para Windows compatível com a versão do Python de 64 bits. Para utilizar outra versão do Python, é necessário recompilar a biblioteca com essa alteração.

Conforme citado anteriormente, a comunicação do script em Python com o simulador, por meio da biblioteca “Legacy remote API”, ocorre por meio de sockets. Desse modo, nas cenas da simulação é adicionado o comando *simRemoteApi.start(19999)*, que cria um socket no endereço de l27.0.0.1:1999. No script em Python, é adicionado o comando *simxStart('127.0.0.1', 19999, True, True, 5000, 5)*, onde os dois primeiros parâmetros se referem ao IP e porta do socket, o terceiro parâmetro indica se o código deve bloquear o seu fluxo até que uma respostas seja recebida, o quarto parâmetro indica se deve ocorrer uma tentativa de conexão em caso de desconexão, o quinto parâmetro indica o tempo máximo de espera para a conexão em milissegundos e, por fim, o sexto parâmetro indica a taxa de comunicação entre os processos.

Uma característica importante do uso dessa API é que no lado do cliente (código em Python) ela adiciona duas *threads*: uma principal para a chamada de funções e outra para gerenciar a comunicação entre os programas. Todavia, é possível criar mais *threads* no lado do cliente, nas quais o comando *sim.simxStart* deve ser adicionado (“Remote API modus operandi”, 2022).

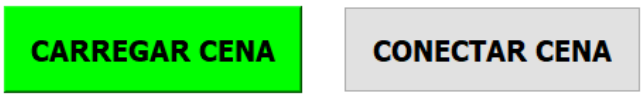
A área da interface gráfica referente à robótica virtual foi dividida em duas páginas. A primeira página permite a visualização de conceitos básicos de robótica, como translação e rotação. Na Imagem 13 é apresentada a página de conceitos básicos referente à translação:

Imagem 13 – Tela de conceitos básicos (translação)

Fonte: O autor.

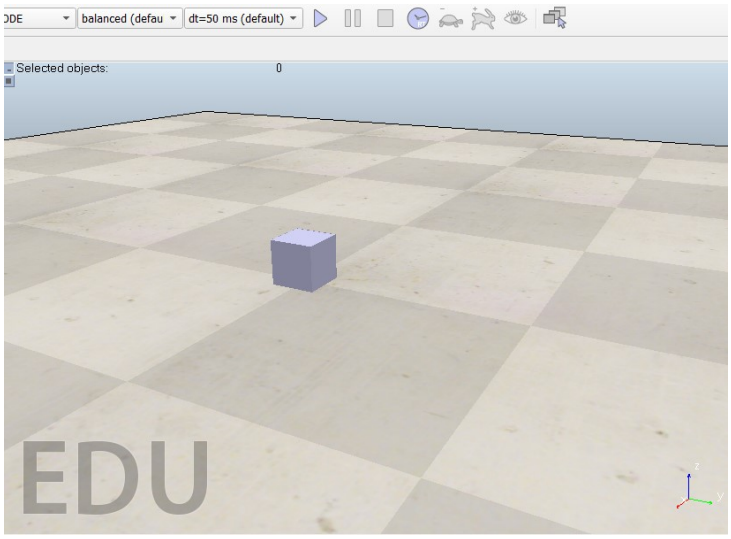
De acordo com a Imagem 13, é possível verificar a página de conceitos básicos com a guia translação selecionada. O botão “CARREGAR CENA” realiza a abertura do arquivo da cena no simulador utilizando a função *subprocess.call* da biblioteca *subprocess*. Os parâmetros para a abertura do arquivo variam de acordo com o sistema operacional do usuário, o qual é anteriormente verificado com a função *platform.system* da biblioteca *platform*. Dessa forma, a abertura da cena no sistema operacional Windows terá como função completa *subprocess.call(('start', filepath), shell=True)*, onde *filepah* é o caminho para o arquivo onde está a cena que será aberta. Ao abrir a cena, o botão terá a sua cor modificada para verde, indicando que a operação ocorreu com sucesso, conforme apresentado na Imagem 14. A cena carregada é apresentada na Imagem 15.

Imagem 14 – Indicação da cena carregada com sucesso



Fonte: O autor.

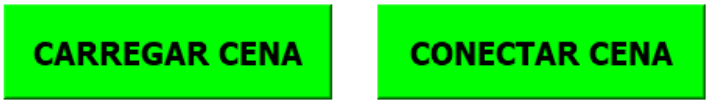
Imagem 15 – Cena para visualização dos conceitos básicos



Fonte: O autor.

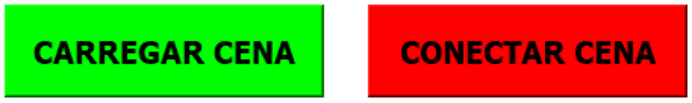
O botão “CONECTAR CENA” realiza a tarefa de conexão com a cena, chamando a função *simxStart* citada anteriormente. Para que a conexão seja bem sucedida, a cena precisa estar aberta e em execução (necessário clicar no botão executar do simulador). Em caso de sucesso, o botão terá a sua cor alterada para verde, conforme Imagem 16. Já em caso de falha, o mesmo terá a sua cor alterada para vermelho, conforme Imagem 17.

Imagem 16 – Indicação da cena conectada com sucesso



Fonte: O autor.

Imagem 17 – Indicação de falha ao conectar cena



Fonte: O autor.

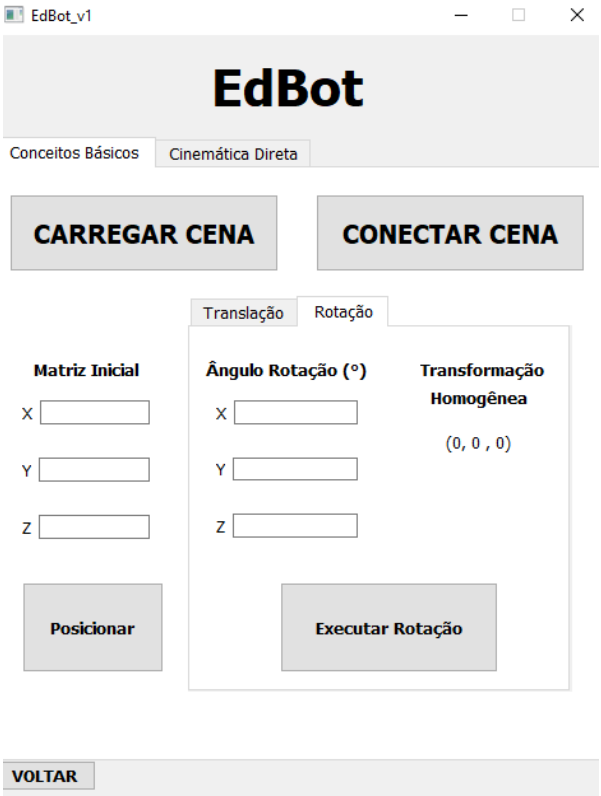
Posteriormente, o campo “Matriz Inicial” permite que o cubo da cena tenha a sua posição inicial alterada antes de realizar a transformação. Para isso, basta informar as novas coordenadas X, Y e Z e pressionar o botão “Posicionar”. Caso algum caractere diferente de um número seja digitado, o valor do campo é alterado para zero. O clique no botão “Posicionar” chama a função *simxSetObjectPosition* da biblioteca do V-REP, passando os valores das coordenadas digitadas como parâmetros. Essa etapa refere-se ao parâmetro ${}^{A}P_1$ apresentado na equação 1.

O operador translação, $D_Q(q)$, é preenchido de acordo com os campos da aba “Matriz Translação”. Os valores das coordenadas X, Y e Z (em metros), representam, respectivamente, as variáveis q_x, q_y, e q_z do operador translação.

O cálculo da matriz resultante da translação de ${}^{A}P_1$ pelo operador $D_Q(q)$ é realizado ao se clicar no botão “Executar Translação” e segue a equação 1. O resultado dessa operação, ${}^{A}P_2$, é utilizado na função *simxSetObjectPosition* para determinar a nova posição do objeto e apresentado no campo “Transformação Homogênea”. Os cálculos da multiplicação das matrizes são realizados no script em Python, utilizando a biblioteca numpy.

A tela para a execução da rotação é apresentada na Imagem 18:

Imagem 18 – Tela de conceitos básicos (rotação)



Fonte: O autor.

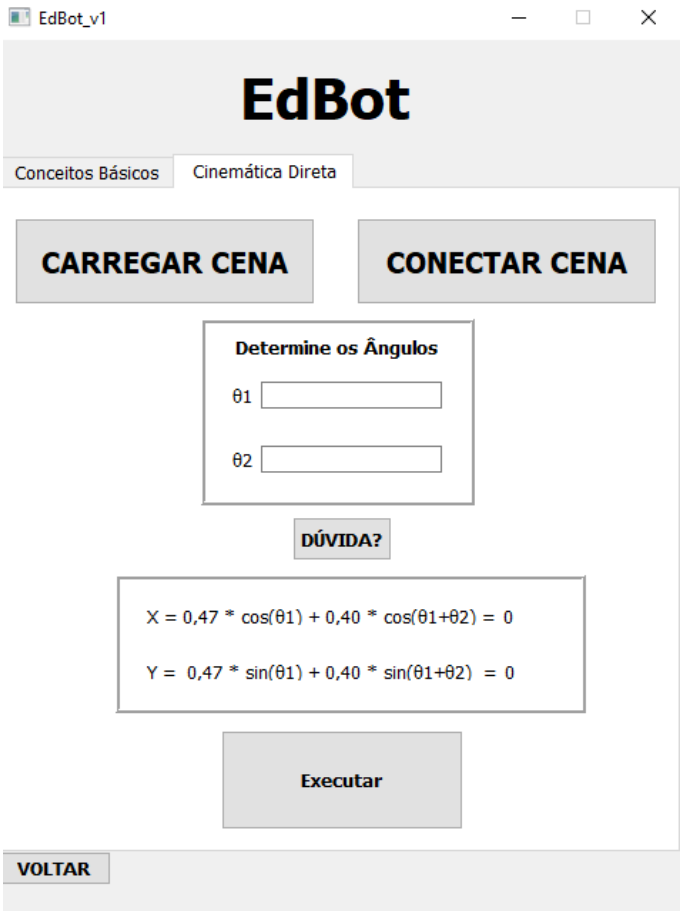
Comparando-se as telas para translação e rotação (Imagens 12 e 17), observa-se que o único campo alterado é o campo “Ângulo de Rotação”. Portanto, os demais campos e botões mantêm a mesma funcionalidade.

Dessa forma, o operador rotação, $R_k(\theta)$, é preenchido de acordo com os campos da aba “Ângulo de Rotação”. Os valores das coordenadas X, Y e Z (em graus), representam, respectivamente, as componentes $R_x(\theta)$, $R_y(\theta)$ e $R_z(\theta)$ do operador rotação.

O cálculo da matriz resultante da rotação de ${}^{A}P_1$ pelo operador $R_k(\theta)$ é realizado ao se clicar no botão “Executar Rotação” e segue a equação 3. O processo seguinte é semelhante ao efetuado na aba translação. O resultado dessa operação, ${}^{A}P_2$, é utilizado na função *simxSetObjectPosition* para determinar a nova posição do objeto e apresentado no campo “Transformação Homogênea”. Os cálculos da multiplicação das matrizes são realizados no script em Python, utilizando a biblioteca numpy.

Outra funcionalidade relacionada à simulação de robôs presenta no programa é a cinemática direta. Na Imagem 19 é apresentada a tela para essa função:

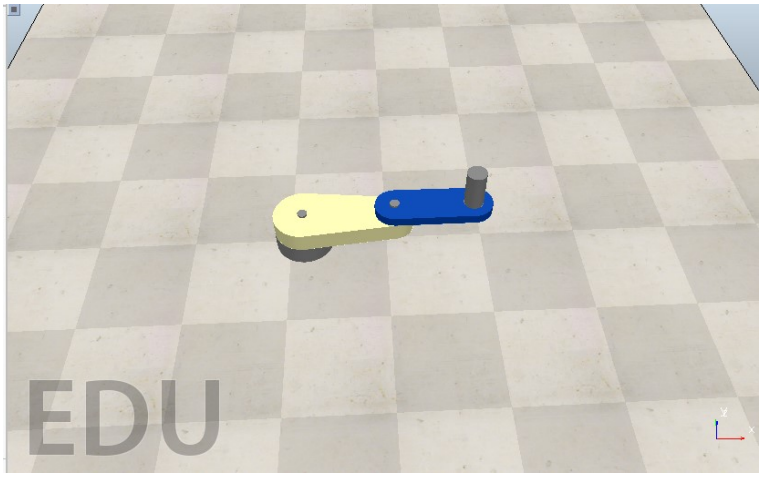
Imagem 19 – Tela de cinemática direta



Fonte: O autor.

De acordo com a Imagem 19, observa-se a permanência dos botões “CARREGAR CENA” e “CONECTAR CENA”. Estes dois botões continuam com o mesmo funcionamento anteriormente descrito. A única alteração é em relação a cena carregada. A mesma é exibida na Imagem 20:

Imagem 20 – Cena para visualização dos conceitos de cinemática direta



Fonte: O autor.

Na Imagem 20 é apresentado um robô de três elos e duas juntas, baseado no modelo “MTB robot” disponível no V-REP. As duas juntas movimentam os elos amarelo e azul com ângulos Θ_1 e Θ_2, respectivamente, em torno do eixo Z centrado no ponto em cinza do primeiro elo (amarelo). Dessa forma, o objetivo da cinemática direta é encontrar a posição do efetuador (cilindro cinza sobre o elo azul), em relação aos eixos X e Y, de acordo com a variação dos ângulos.

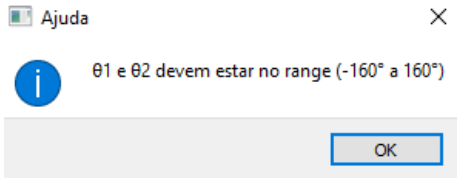
O robô apresentado na Imagem 20 apresenta dois graus de liberdade, semelhante ao robô descrito na Imagem 9. Portanto, as equações 7 e 8 são válidas para encontrar a posição do efetuador, de modo que o comprimento L_1 é igual a 0,47 metros e o comprimento L_2 é igual a 0,40 metros.

Portanto, a aplicação é executada da seguinte forma: o usuário informa os ângulos Θ_1e Θ_2 que ele deseja movimentar os elos e clica no botão “Executar”, a posição final do efetuador é calculada, no próprio programa em Python, e a função *simxSetJointPosition* é chamada para realizar o movimento. O resultado da operação também é indicado na tela.

Um detalhe importante relacionada à movimentação do robô é que os elos aceitam movimentações no intervalo de -160° e 160°. Caso o usuário informe valores fora desse

intervalo a movimentação não é realizada. O usuário pode conferir o intervalo válido ao clicar no botão “Dúvida ?”, o qual exibe a mensagem apresentada na Imagem 21:

Imagem 21 – Mensagem de indicação do intervalo válido para entrada de ângulos



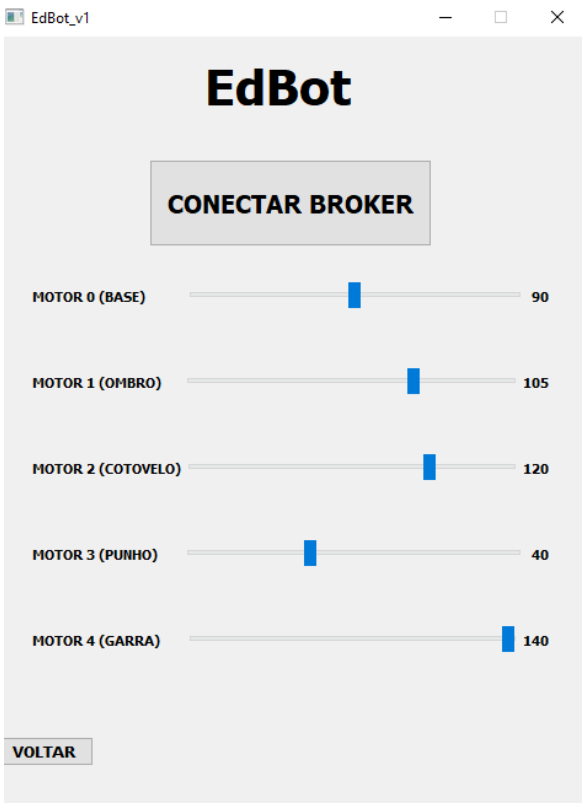
Fonte: O autor.

Em qualquer momento, caso o usuário deseje retornar para a tela inicial da aplicação, o mesmo deve clicar no botão “Voltar”.

3.1.2 Robô Físico

A tela de controle do robô físico pode ser acessada ao clicar no botão “ROBÔ FÍSICO” presente na tela inicial do programa (Imagem 12). Esta tela é apresentada na Imagem 22:

Imagem 22 – Tela de comando do robô físico



Fonte: O autor.

A comunicação com o robô é iniciada ao clicar no botão “CONECTAR BROKER”. O mesmo terá a sua cor alterada para verde em caso de conexão bem sucedida, conforme apresentado na Imagem 23:

Imagem 23 – Indicação de conexão estabelecida

Fonte: O autor.

Posteriormente, o usuário pode movimentar o robô de acordo com a alteração da posição da barra azul do componente *slider* (Imagem 22). A alteração da posição da barra publica uma mensagem no *broker* MQTT indicando qual elo deve se mover e para qual posição. A posição atual (em graus) de cada elo é exibida no canto direito da linha de cada elo. Maiores detalhes sobre o funcionamento da comunicação do programa via MQTT são apresentados no tópico sobre o *firmware*.

Caso o usuário deseje retornar para a tela inicial da aplicação, o mesmo deve clicar no botão “Voltar”.

3.2 Hardware

Este trabalho tem como foco os aspectos eletrônicos e a programação de um robô, portanto, decidiu-se por imprimir em 3D a estrutura de um robô de licença livre já disponível no mercado e projetar a placa de circuito impresso (PCB) responsável pelo acionamento do robô.

O modelo do robô escolhido para ser impresso em 3D é apresentado na Imagem 24. Este robô apresenta cinco graus de liberdade e uma garra. Nos três primeiros elos é prevista a instalação de servo motores do tipo MG996R. Já nos dois elos superiores é prevista a instalação de servo motores do tipo SG90, um modelo mais simples e com menos torque que o anterior (DEJAN, 2018). Este modelo de robô já foi utilizado em aplicação semelhante, na qual o mesmo era comandado remotamente por uma luva de dados (GOMES; MARQUES; RIBEIRO, 2021).

Imagem 24 – Estrutura física do robô impresso em 3D

Fonte: Dejan (2018).

Para o acionamento do robô físico foi projetada e desenvolvida uma placa de circuito impresso controlada por um chip ESP32. Dessa forma, foram realizadas as seguintes etapas: projeto do diagrama elétrico, elaboração do *layout*, pedido de fabricação e soldagem dos componentes.

A escolha da utilização do SoC (System-on-a-chip ou Sistema-em-um-chip, em português) ESP32 deve-se aos seguintes motivos: disponibilidade de Wi-Fi no mesmo chip do controlador, possibilidade de programação com diferentes ambientes, existência de uma grande comunidade e a alta disponibilidade de compra.

Para facilitar o processo de soldagem e a troca em possíveis manutenções, optou-se por utilizar, sempre que possível, módulos comerciais para compor o circuito. Portanto, foi escolhido o módulo conhecido como “ESP-WROOM-32 38 PIN Development Board” que utiliza o chip ESP-WROOM-32 para controlar os periféricos da placa. As suas principais características são apresentadas na Tabela 1:

Tabela 1 – Informações do chip ESP-WROOM-32

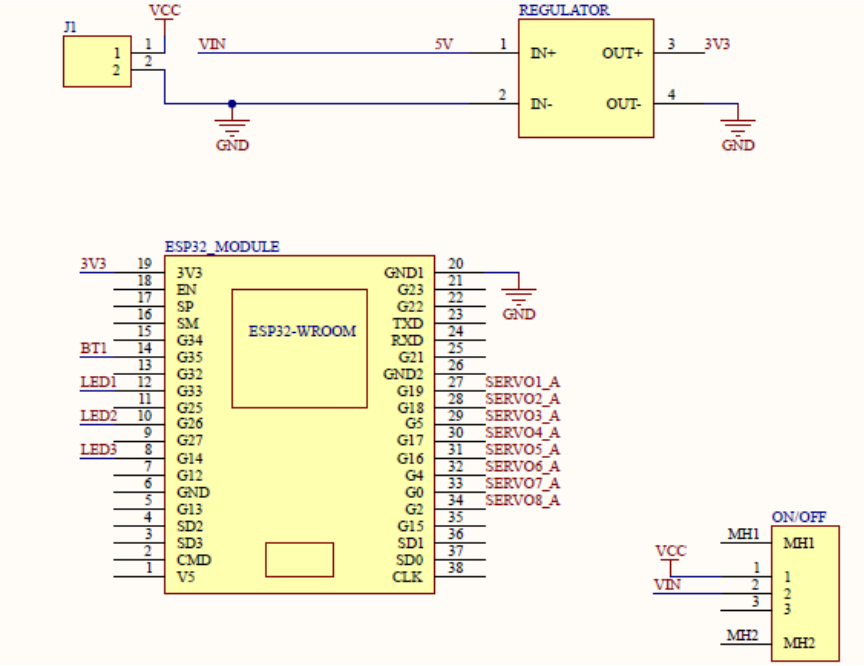
Tensão de Operação	3.0 V ~ 3.6 V
Corrente de Operação	Média de 80 mA
Cristal Interno	40 MHZ
Flash SPI Interna	4 MB
SRAM Interna	520 KB
ROM Interna	448 KB
Wi-Fi	802.11 b/g/n (802.11n até 150 Mbps)

Fonte: (ESPRESSIF, 2013).

O projeto do diagrama elétrico do circuito e o *layout* da PCB foram realizados utilizando o software Altium. Este programa, apesar de ser proprietário, disponibiliza licenças gratuitas para estudantes universitários (ALTIUM, 2022).

O circuito de alimentação é formado por um conector para o acoplamento dos conectores da fonte externa de 5 V e 5 A, uma chave para ligar e desligar a alimentação e um regulador de tensão de 5 V para 3,3 V (módulo com chip Ams1117). Desta forma, é possível fornecer uma alimentação de 5 V ou 3,3 V para todos os componentes. Na Imagem 25 é apresentado o circuito de alimentação e o módulo ESP32.

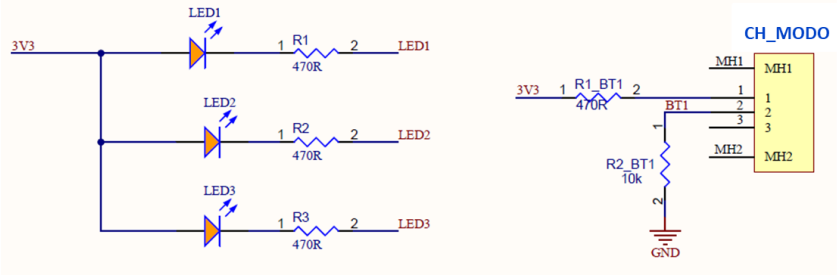
Imagem 25 – Circuito de alimentação e módulo ESP32



Fonte: O autor.

De modo a indicar os estados de operação do firmware, foi projetada a inclusão de três LEDs (*Light Emitting Diode*). Além disso, uma chave (CH_MODO) foi projetada para ter o seu nível de tensão lido pelo microcontrolador. Esta parte do circuito é apresentada na Imagem 26.

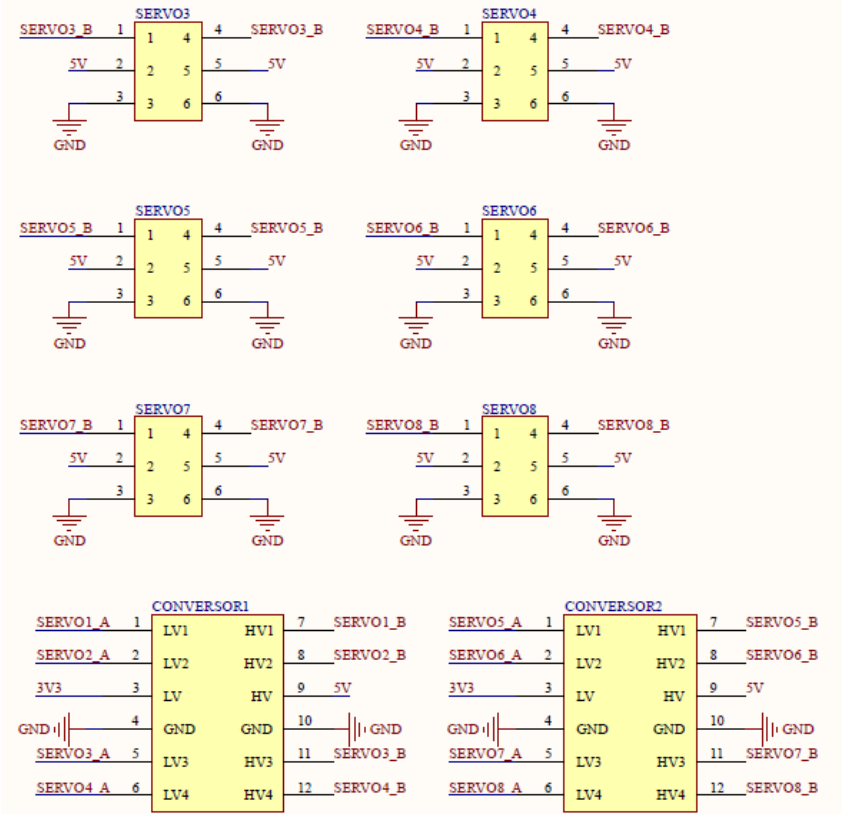
Imagem 26 – Circuito de LEDS e chave



Fonte: O autor.

Por fim, o circuito de acionamento é apresentado na Imagem 27. O mesmo é composto por dois módulos CNL35 conversores de nível lógico bidirecional (5 V e 3,3 V) de 4 canais, responsáveis pelo acionamento dos servo motores e conectores para o acoplamento de até 8 servo motores.

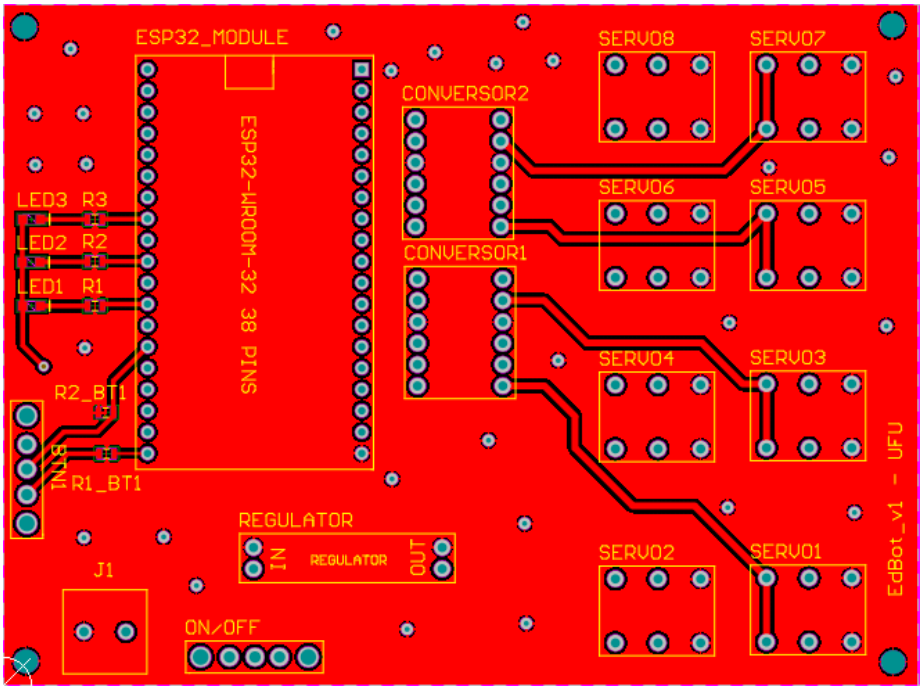
Imagem 27 – Circuito de acionamento



Fonte: O autor.

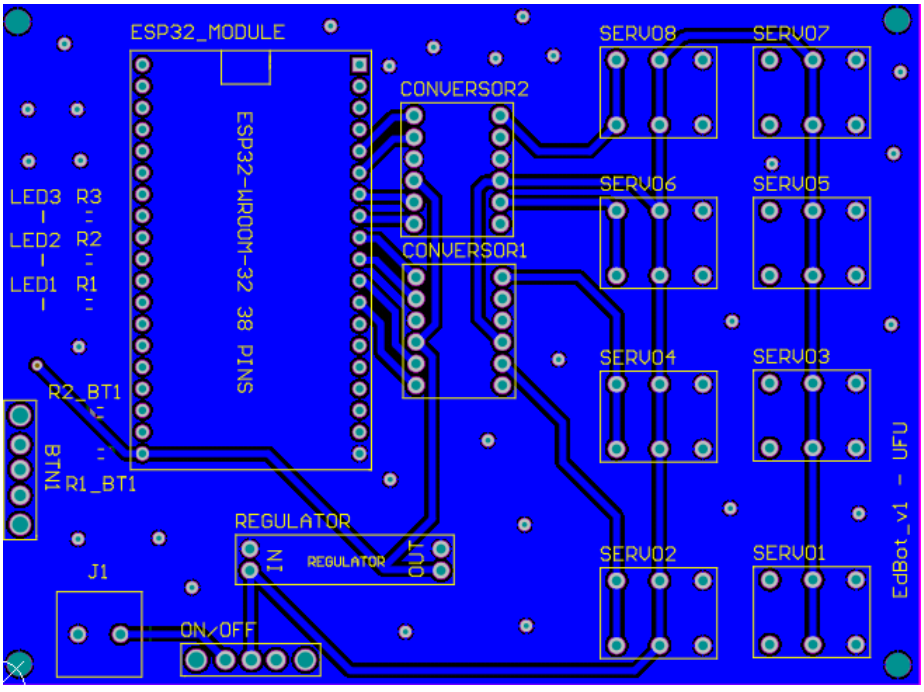
O *layout* do circuito foi realizado em uma placa de duas faces, com comprimento de 11 centímetros e altura de 8,1 cm. Na Imagem 28 é apresentada a visualização da camada superior da placa e na Imagem 29 é apresentada a visualização da camada inferior.

Imagem 28 – Projeto da camada superior da placa



Fonte: O autor.

Imagem 29 – Projeto da camada inferior da placa



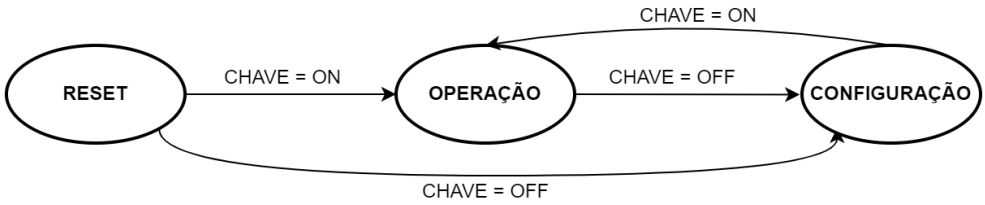
Fonte: O autor.

3.3 Firmware

O *firmware* para o controle do robô e comunicação com o *software* via MQTT foi elaborado em C/C++ com o *framework* Arduino disponível na extensão Platformio do IDE (Ambiente de Desenvolvimento Integrado) do Microsoft Visual Studio Code. Optou-se por utilizar o *framework* Arduino por conta do grande número de bibliotecas disponíveis e da facilidade da sua utilização, o que, além de possibilitar um desenvolvimento mais rápido, simplifica eventuais manutenções futuras. Todavia, para facilitar a portabilidade do código para outros ambientes de programação ou para chips diferentes, as chamadas que utilizam alguma função específica desse hardware foram adicionadas em um arquivo próprio, com exceção das funções de conexão Wi-Fi e MQTT, as quais são muito dependentes da biblioteca utilizada.

Tendo em vista a baixa complexidade do *firmware*, a estratégia escolhida para o seu desenvolvimento é um *loop* principal associado a duas interrupções, uma interrupção relacionada à comunicação via MQTT e outra responsável pela leitura da chave CH_MODO que indica o modo de operação. Dessa forma, é possível representar o funcionamento do sistema de acordo com a máquina de estados apresentada na Imagem 30:

Imagem 30 – Máquina de estados do *firmware*



Fonte: O autor.

Conforme o diagrama apresentado na Imagem 30 observa-se que o *firmware* é composto por três estados, sendo um estado de inicialização e dois estados que se alternam durante que se alternam durante a execução do loop. Destelhes sobre a execução de cada estado são apresentas na sequência.

3.3.1 Estado reset

Neste estado é realizada a inicialização das variáveis e dos periféricos. A primeira etapa é a configuração dos pinos de saída (3 LEDs) e a configuração da interrupção no pino

da chave CH_MODO. A função de cada LED é apresentada na Tabela 2. Uma vez que a chave pode assumir dois estados e o *framework* Arduino permite apenas a configuração de uma interrupção por pino, foi utilizada a estratégia de definir a interrupção na alteração de ambos os estados (parâmetro *CHANGE*), seguida da leitura, dentro da interrupção, de qual o estado atual da chave.

Tabela 2 – Indicação de cada LED

LED 1	Modo de Operação
LED 2	Status da conexão MQTT
LED 3	Status da conexão Wi-Fi STA

Fonte: O autor.

Em seguida é realizada a configuração do Wi-Fi. O chip da ESP32 apresenta dois modos de configuração de Wi-Fi: o modo STA (*Station*), no qual a ESP32 se conecta em alguma rede externa e o modo AP (*Acess Point*), no qual outros aparelhos se conectam na rede gerada pela ESP32 (ESPRESSIF, [s.d.]). Portanto, foi programada a habilitação dos dois modos. No modo STA, a ESP32 tenta se conectar com a rede informada no seu *firmware*. Caso a conexão seja bem sucedida, o LED 3 é acionado. Já no modo AP, a ESP32 cria a rede "EdBot_v1", definindo a senha "12345678" para a conexão.

Posteriormente, é realizada a inicialização do MQTT. Nessa etapa é utilizada a biblioteca "PubSubClient" para a comunicação e a biblioteca "ArduinoJson" para facilitar a leitura do texto no formato JSON (BLANCHON, [s.d.]; O'LEARY, [s.d.]). O *broker* escolhido para essa aplicação é o "Test Mosquitto" ("Test Mosquitto", [s.d.]), o qual tem apresenta a vantagem de ser gratuito, porém, todos os seus tópicos são públicos. Dessa forma, o tópico "edbotv1_ufu2022/position/set" foi definido para a assinatura pelo *firmware* e publicação pelo *software*, de forma a diminuir o risco de outra aplicação publicar no mesmo tópico. Além disso, a QoS foi definida para 0, uma vez que muitas mensagens são publicadas pelo software e a perda de uma mensagem não traz grandes riscos. Por fim, o *client ID* (identificação do cliente no *broker*) do *firmware* foi definido como "EdBotV1".

Uma vez configurada a comunicação com o *broker* MQTT, o LED 2 será ligado no momento em que a conexão for estabelecida. A partir desse momento, as mensagens que chegarem ao tópico assinado e tiverem um formato de JSON aceito ({"motor": número do servo motor a ser acionado, "pos": posição para mover o servo}) serão adicionadas em um *buffer* circular, o qual será consumido no modo operação. Portanto, a cada mensagem válida

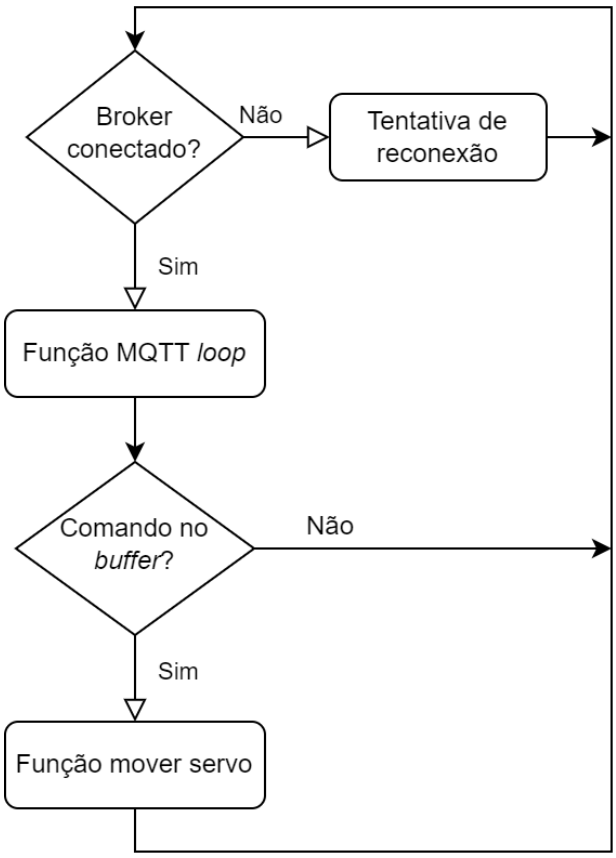
recebida serão adicionados dois bytes ao buffer, um relacionado ao número do motor e outro relacionado à posição.

A última configuração realizada nessa etapa é posicionamento dos servo motores em uma posição inicial segura. Essa etapa permite sincronizar a posição inicial do robô com a posição inicial apresentada nos *sliders* da interface gráfica e evita que algum servo inicie a sua movimentação em alguma região que possa danificar alguma estrutura do robô.

3.3.2 Estado operação

Este estado se inicia quando, após o estado reset, a chave CH_MODO estiver enviando um sinal alto de tensão no pino de leitura do ESP32, o que é indicado pelo LED 1 ligado. Neste estado é realizado um ciclo contínuo conforme o apresentado na Imagem 31:

Imagem 31 – Ciclo do modo operação



Fonte: O autor.

A primeira ação realizada é a verificação do estado da conexão com o *broker.* Caso não haja conexão, é chamada a função de conexão da biblioteca “PubSubClient”. Em seguida,

o comando *loop* dessa mesma biblioteca é chamado para enviar uma mensagem ao *broker* informando que o cliente está ativo.

Na sequência o *buffer* circular é observado. Caso haja algum byte disponível, a função de movimentação dos servos é chamada. Esta função verifica se o número do servo é válido (entre 0 e 4, de acordo com Imagem 22) e a posição pode ser alcançada (de acordo com os limites de operação definidos no *firmware* para cada elo, de forma a evitar que a movimentação para determinada posição possa causar algum dano ao robô). Em seguida, a posição do servo é incrementada unitariamente, com um intervalo de 10 milissegundos entre cada movimentação, até que se alcance a posição desejada.

3.3.3 Estado configuração

Este estado se inicia quando, após o estado reset, a chave CH_MODO estiver enviando um sinal baixo de tensão no pino de leitura do ESP32, o que é indicado pelo LED 1 desligado. Este estado é composto apenas pela função *handleClient* da biblioteca "WiFi" disponível no próprio ambiente do Arduino para esse chip. Esta função permite que a ESP32 atue com um servidor web. De modo que, estando conectado na mesma rede, é possível acessar a página apresentada na Imagem 32 por meio do endereço IP:80/atualizar.

Imagem 32 – Página de atualização do firmware

Fonte: O autor.

A página web apresentada na Imagem 32 foi desenvolvida, tendo como base o exemplo apresentado na biblioteca "WiFi", na linguagem *HyperText Markup Language*

(HTML) com o seu estilo modificado por meio da linguagem Cascading Style Sheets (CSS). Além disso, é utilizada a linguagem JavaScript e a biblioteca JQuery para permitir a atualização da mensagem de progresso sem que a página precise ser recarregada. Todo o código elaborado para essa página, incluindo a biblioteca JQuery, é salvo na memória não volátil da ESP32 e é enviado para o cliente após o recebimento da requisição HTTP.

O objetivo desta página web é permitir que o *firmware* da ESP32 seja atualizado de forma remota via Wi-Fi, procedimento conhecido como atualização *Over-the-air* (OTA). Portanto, ao clicar no botão “Escolher arquivo” apresentado na Imagem 32, o usuário pode escolher um novo firmware e iniciar a atualização clicando no botão “Update”. O progresso da atualização é informado na barra no canto inferior da página. Ao fim, se a operação for bem sucedida, a ESP32 será reinicializada e passará a utilizar o novo código. Esse procedimento é útil para atualizar parâmetros do *firmware* original.

Um detalhe importante em relação à formatação da página com CSS é que as dimensões dos elementos da página foram definidas em função do tamanho da tela do aparelho que acessa o servidor. Desse modo, caso o usuário deseje acessar a página com diferentes aparelhos, os elementos da página manterão a mesma proporção.

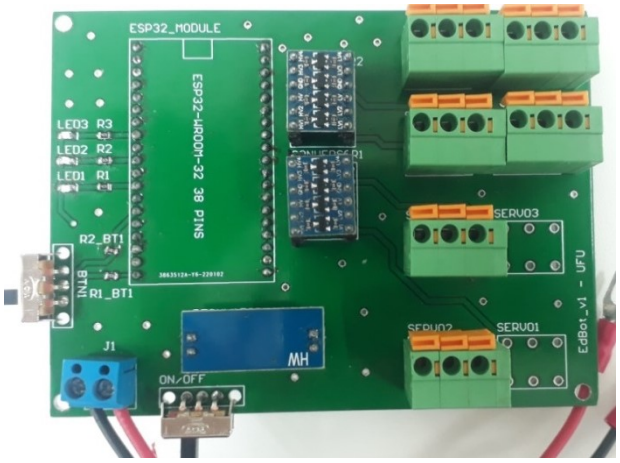
4 RESULTADOS

Neste capítulo são apresentados os resultados obtidos neste trabalho. Para facilitar o entendimento, primeiramente são apresentados os resultados referentes ao sistema embarcado desenvolvido e, em seguida, os resultados referentes ao *software*.

4.1 Sistema embarcado

O sistema embarcado desse projeto foi elaborado de acordo com os diagramas apresentados no item 3.2 (Imagens 24 a 28). A fabricação do mesmo foi realizada em uma empresa especializada na fabricação de PCB. Por fim, todos os componentes eletrônicos foram soldados manualmente. Na Imagem 33 é apresentada a camada superior da placa com os componentes soldados:

Imagem 33 – Vista superior da PCB com componentes soldados



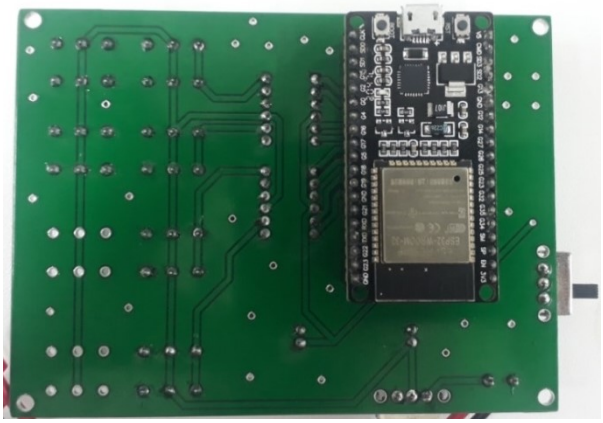
Fonte: O autor.

De acordo com a Imagem 33, é possível observar que foram soldados apenas seis conectores para os servo motores. Isso se deve ao fato de que o robô utiliza apenas cinco servos para a sua operação, sendo que um conector adicional foi soldado para ser utilizado em caso de algum incidente com outro conector.

Além disso, outro detalhe apresentado na Imagem 33 é a ausência do módulo ESP32 na camada superior. Esse fato se deve a uma falha durante a criação do componente no *software* Altium, o qual foi criado de forma espelhada. Portanto, tendo em vista à manutenção das trilhas originas, o chip foi soldado na camada inferior.

Na Imagem 34 é apresentada a visão da parte inferior da placa:

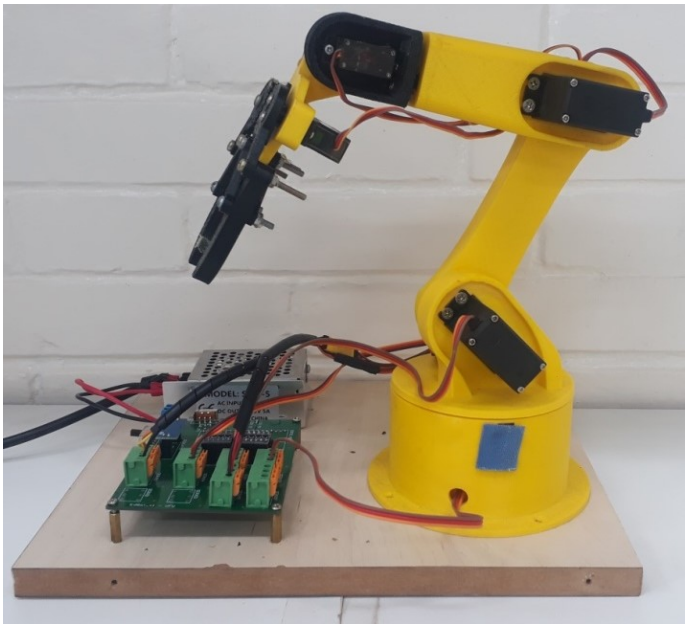
Imagem 34 – Vista inferior da PCB com componentes soldados



Fonte: O autor.

Posteriormente, a placa desenvolvida, a fonte externa e o robô foram fixados em uma base de madeira, de forma a permitir a operação de forma segura e facilitar o transporte da estrutura. Na Imagem 35 é apresentada a montagem final:

Imagem 35 – Estrutura final do robô



Fonte: O autor.

O erro em relação ao projeto do módulo ESP32 na PCB não trouxe problema para a fixação da estrutura, uma vez que os espaçadores utilizados permitem que o módulo não fique em contato direto com a estrutura. Na Imagem 36 é exibido esse detalhe da estrutura:

Imagem 36 – Detalhe da fixação do módulo ESP32

Fonte: O autor.

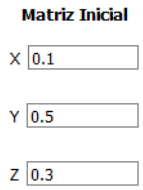
Para validar a estrutura, foi realizado um teste utilizando uma fonte de bancada e um *firmware* simples para o acionamento do servo motores de forma intercalada. Desse modo, foi observado que todos os servos se moveram conforme o esperado e o consumo de corrente durante a movimentação de cada servo é sempre inferior a 1 A. Desse modo, é possível concluir que a fonte externa de 5 A é suficiente para o projeto. Além disso, conclui-se que o projeto e a montagem do sistema embarcado cumpriu o obtivo de permitir o acionamento de um robô.

A compilação do *firmware* foi realizada com as configurações padrões da extensão Platformio. De acordo com a mensagem fornecida ao fim da compilação, o código ocupa aproximadamente 41 Kb de memória RAM e 785 Kb de memória Flash. Observando os dados do Soc ESP32 apresentados na Tabela 1, conclui-se que o chip escolhido é suficiente para a aplicação realizada.

4.2 Software

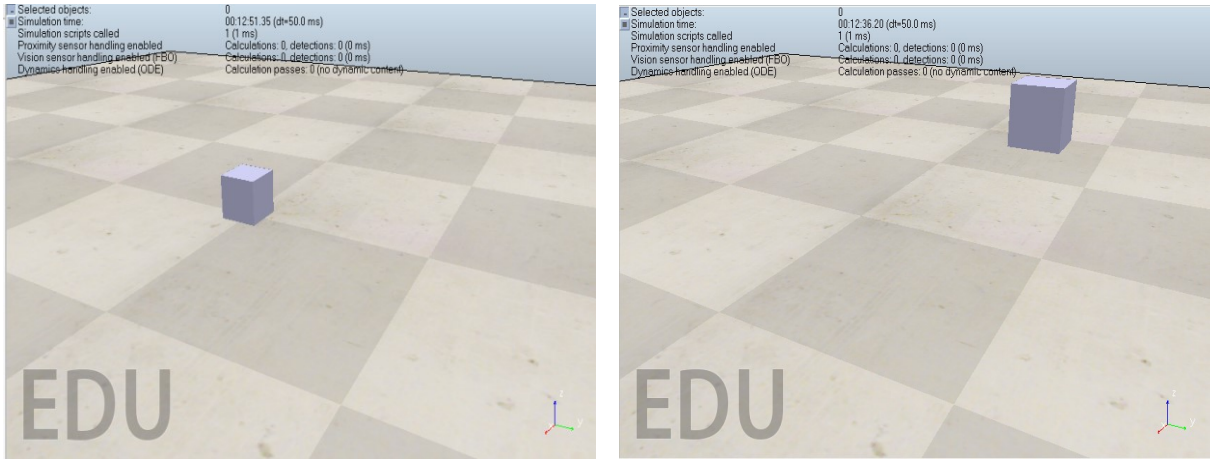
O *software* desse projeto foi elaborado de acordo as informações apresentados no item 3.1. O primeiro teste realizado foi em relação à translação na cena de conceitos básicos. A primeira etapa é o posicionamento do cubo em uma posição inicial. Na Imagem 38 é apresentada a transição do cubo da posição original da cena (imagem à esquerda) para a posição inicial (imagem à direita) apresentada na Imagem 37.

Imagem 37 – Definição da posição inicial do cubo



Fonte: O autor.

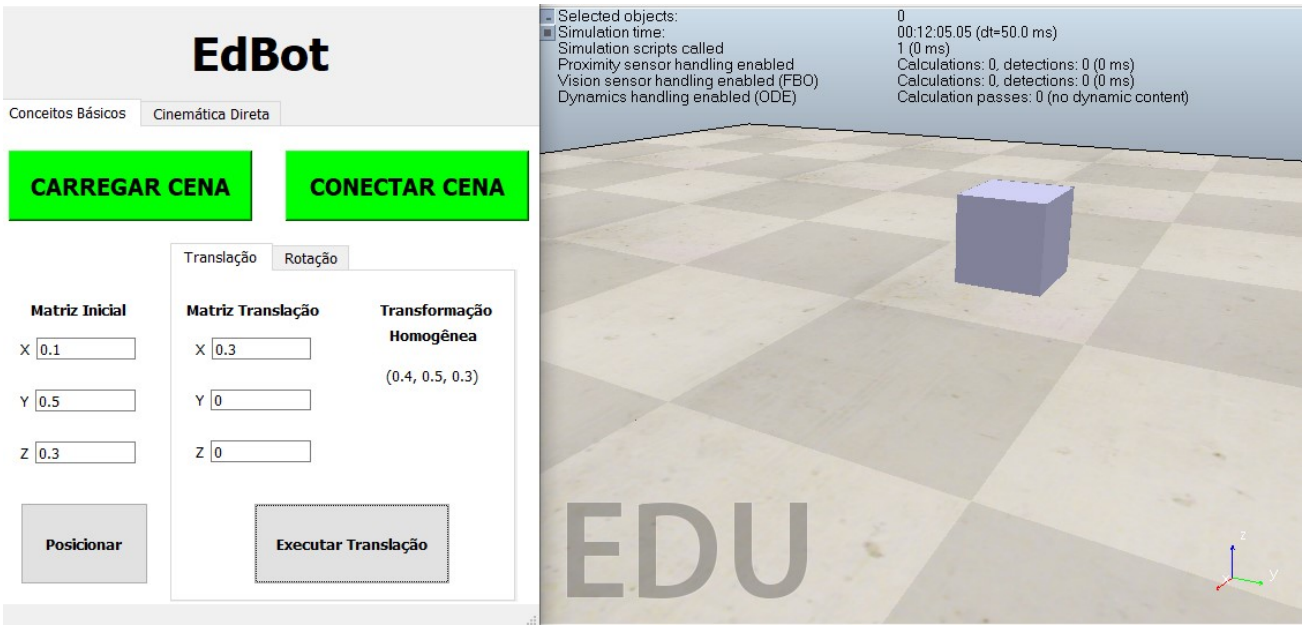
Imagem 38 – Posicionamento inicial do cubo



Fonte: O autor.

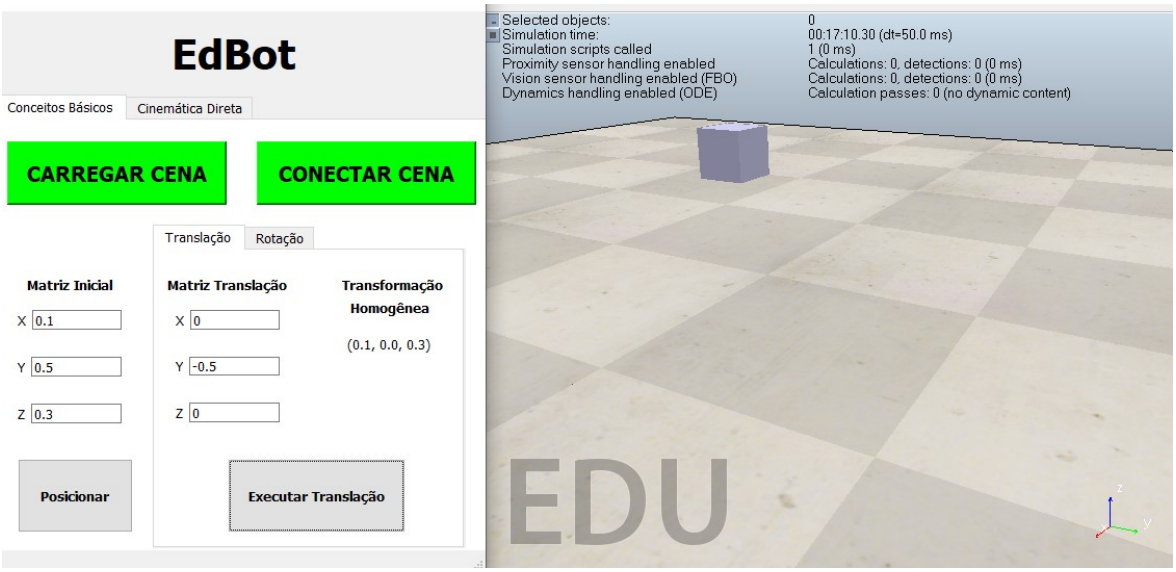
Em seguida é realizado o teste de translação. Para isso, são configuradas translações nas direções X, Y e Z, sempre em relação à posição inicial (0,1; 0,5; 0,3) apresentada na cena à direita da Imagem 38. Dessa forma, na Imagem 39 é apresentada a translação de 0,3 metros no eixo X, na Imagem 40 a translação de -0,5 metros no eixo Y e na imagem 41 a translação no de 0,1 metros no eixo Z.

Imagem 39 – Translação de 0,3 metros no eixo X



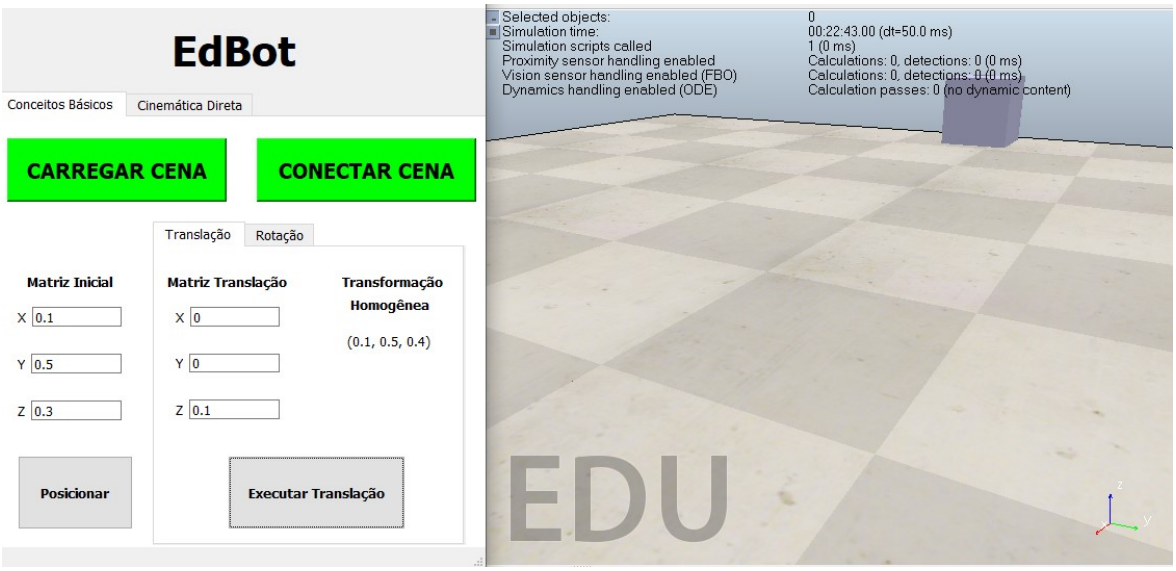
Fonte: O autor.

Imagem 40 – Translação de -0,5 metros no eixo Y



Fonte: O autor.

Imagem 41 – Translação de 0,1 metros no eixo Z

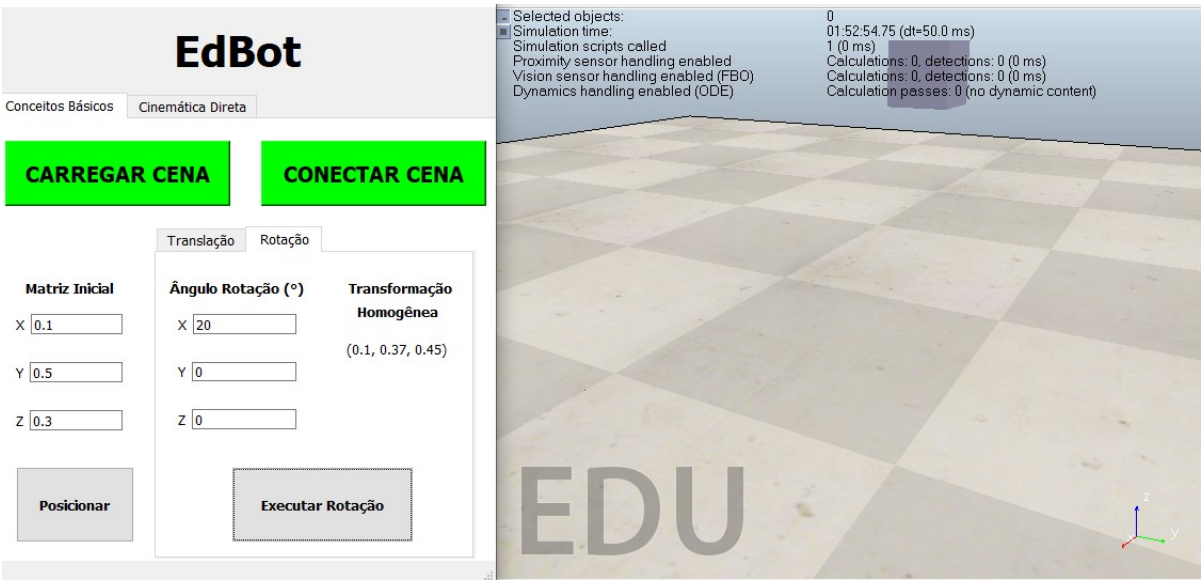


Fonte: O autor.

Diante dos testes realizados, observa-se que a função de translação funcionou corretamente, sendo possível visualizar a movimentação do cubo na cena e o resultado do campo “Transformação Homogênea”. É importante destacar a possibilidade de realizar as transformações nos três eixos simultaneamente, de modo que a escolha de exibir as transformações em cada eixo separadamente deve-se apenas a tentativa de facilitar a observação.

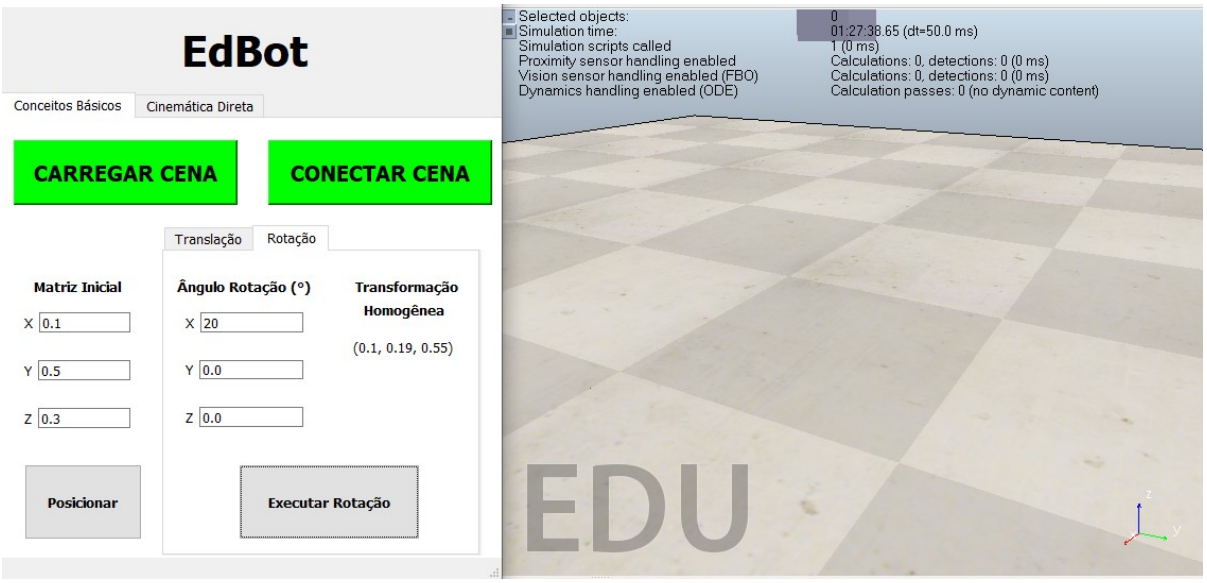
Na sequência é realizado o teste da função de rotação. De modo semelhante ao teste de translação, são configuradas rotações nas direções X, Y e Z, partindo-se da posição inicial (0,1; 0,5; 0,3) apresentada na cena à direita da Imagem 38. Dessa forma, duas rotações sucessivas são realizadas em relação ao eixo X (Imagens 42 e 43), duas rotações sucessivas em relação ao eixo Y (Imagens 44 e 45) e duas rotações sucessivas em relação ao eixo Z (Imagens 46 e 47).

Imagem 42 – Primeira rotação em relação ao eixo X



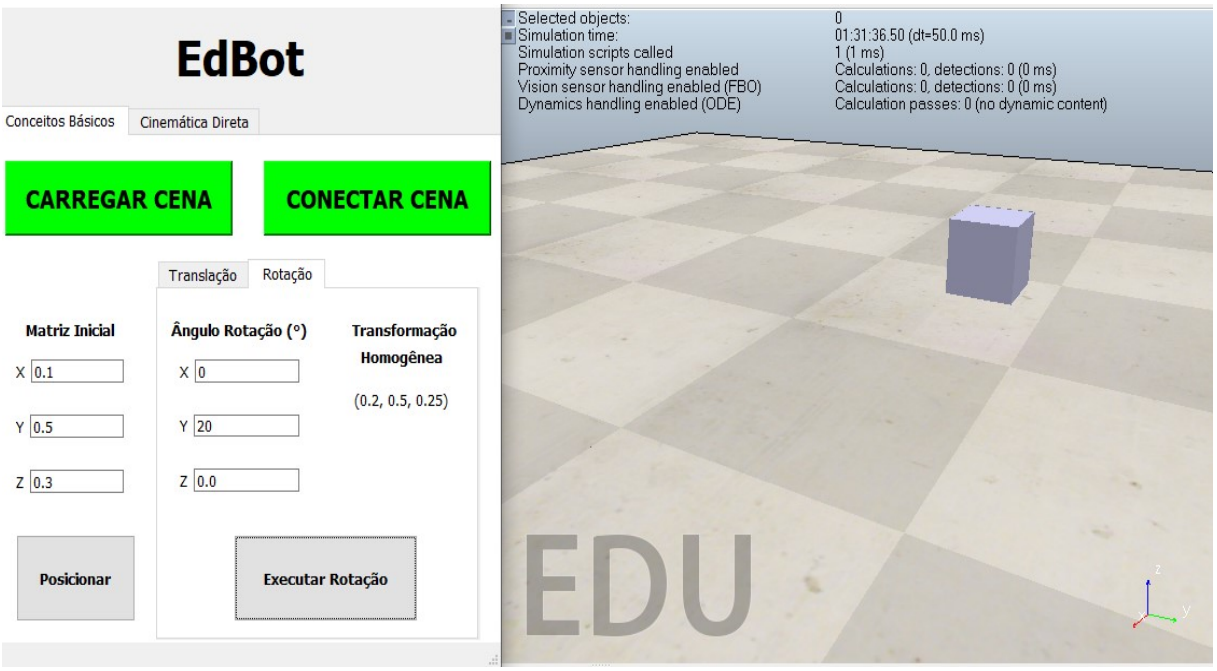
Fonte: O autor.

Imagem 43 – Segunda rotação em relação ao eixo X



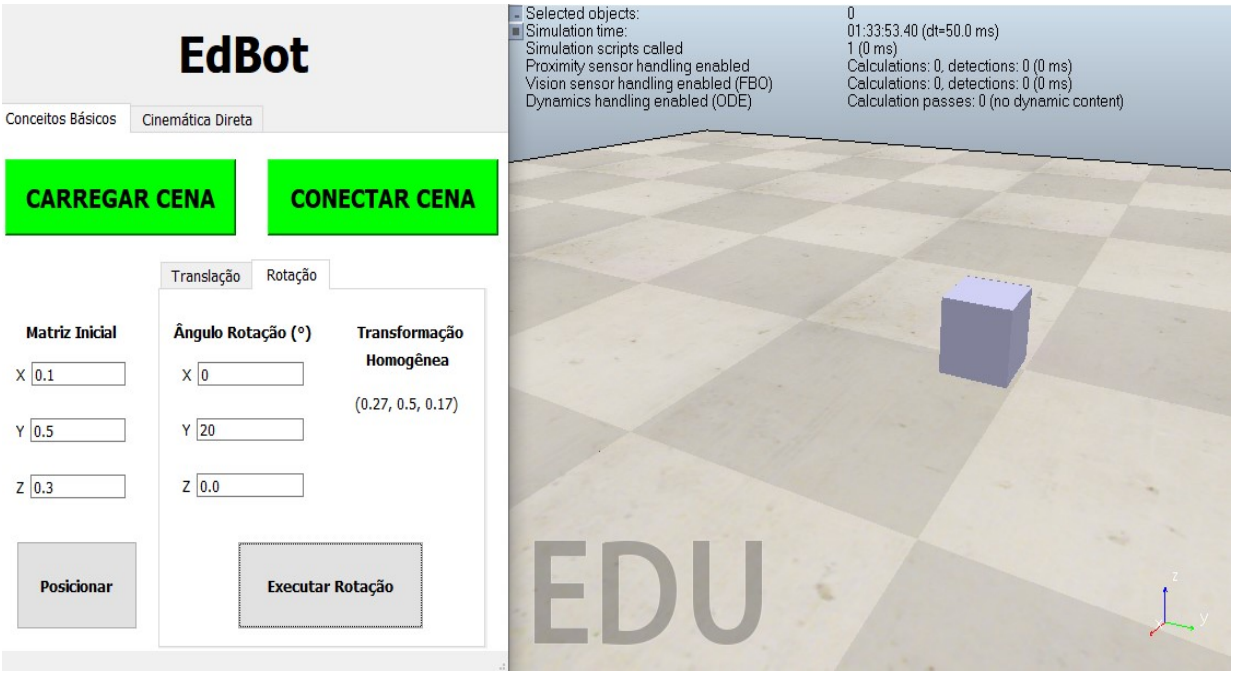
Fonte: O autor.

Imagem 44 – Primeira rotação em relação ao eixo Y



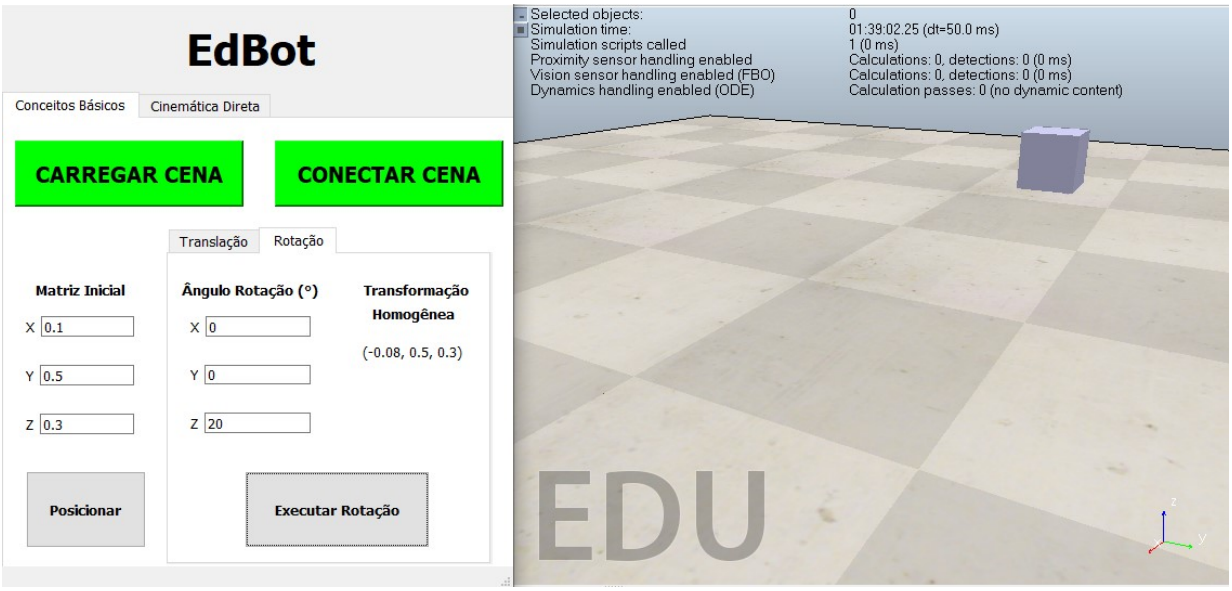
Fonte: O autor.

Imagem 45– Segunda rotação em relação ao eixo Y



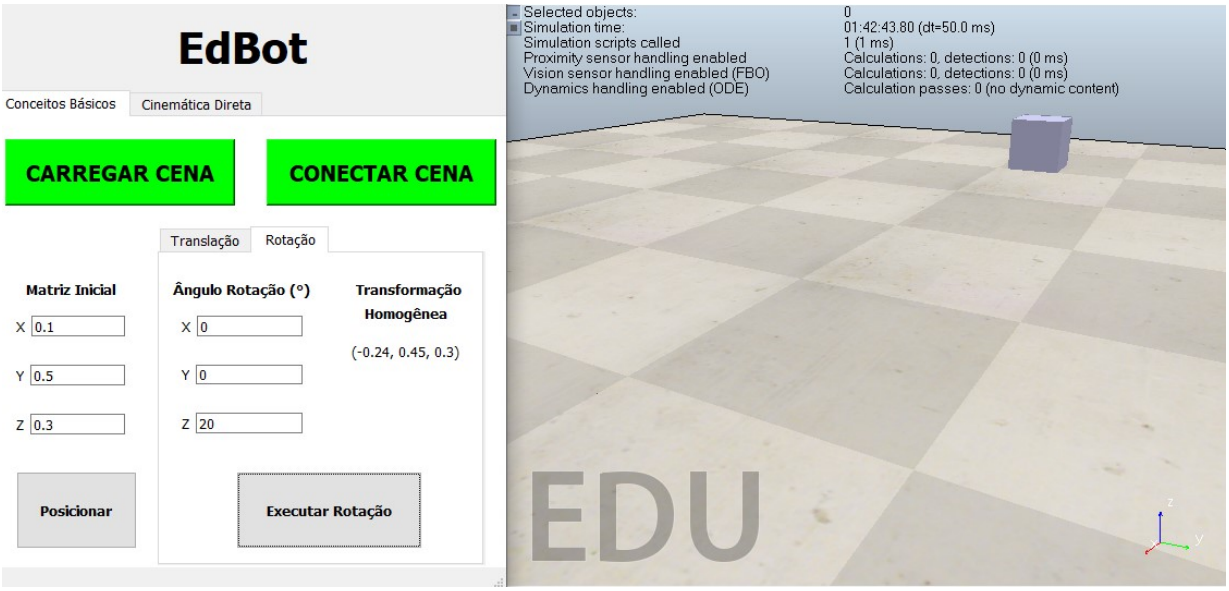
Fonte: O autor.

Imagem 46 – Primeira rotação em relação ao eixo Z



Fonte: O autor.

Imagem 47 – Segunda rotação em relação ao eixo Z



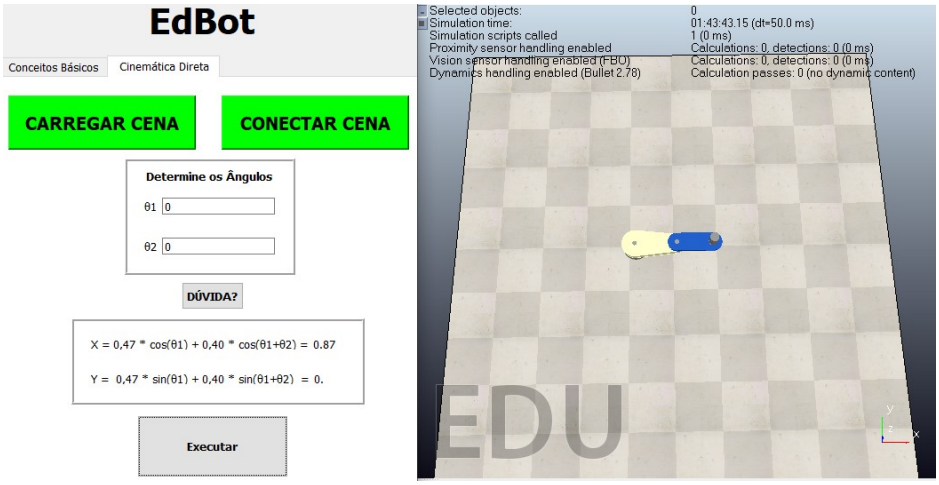
Fonte: O autor.

Diante dos resultados apresentados, observa-se que a função de rotação funcionou de forma satisfatória, sendo possível visualizar a movimentação do cubo na cena e o resultado do campo “Transformação Homogênea”, especialmente ao observar que a coordenada relacionada ao eixo sobre o qual o cubo se movimenta permanece constante durante as duas rotações. Assim como na função de translação, é importante destacar a possibilidade de realizar as transformações nos três eixos simultaneamente, de modo que a escolha de exibir as

transformações em cada eixo separadamente deve-se apenas a tentativa de facilitar a observação.

Na sequência é analisada a função de cinemática direta. Na Imagem 48 é apresentado o resultado da posição do efetuador quando os dois ângulos são iguais a 0. Conforme o esperado, a coordenada Y da posição efetuador é igual a 0, uma vez que esta depende do seno dos ângulos.

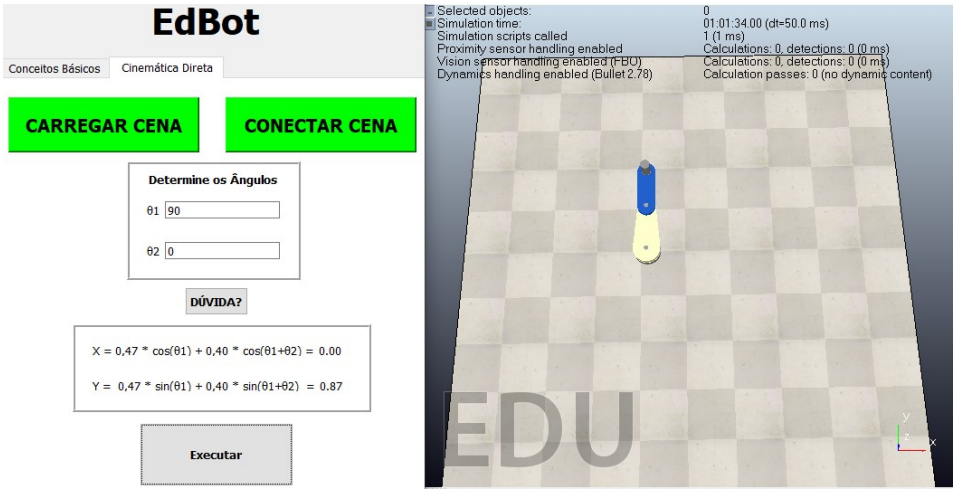
Imagem 48 – Cinemática direta com ângulos iguais a 0



Fonte: O autor.

Na Imagem 49 é apresentado o resultado da posição do efetuador quando o ângulo Θ_1 é igual a 90° e Θ_2 vale 0°. Conforme o esperado, a coordenada X da posição efetuador é igual a 0, uma vez que esta depende do cosseno dos ângulos.

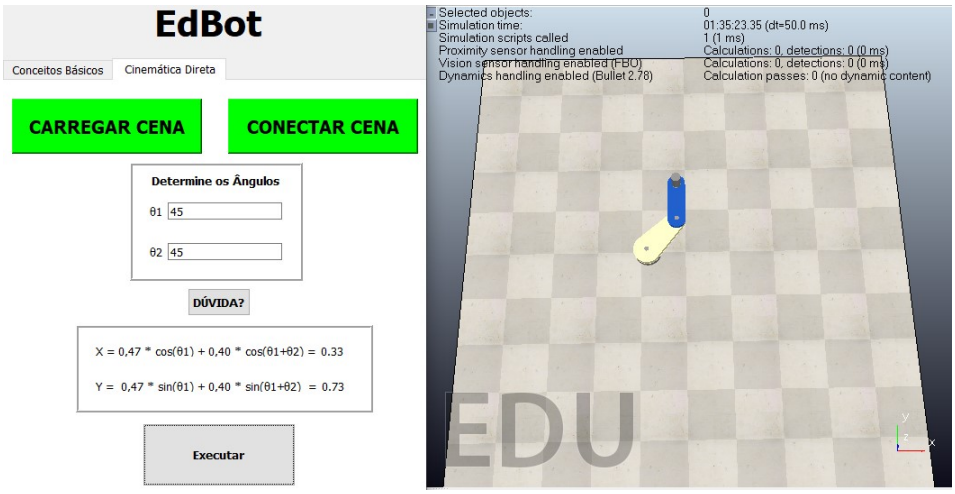
Imagem 49 – Cinemática direta com ângulo Θ_1= 90° e Θ_2 = 0°



Fonte: O autor.

Por fim, na Imagem 50 é apresentado o resultado da posição do efetuador quando o ângulo Θ_1 e Θ_2 valem 45°.

Imagem 50 – Cinemática direta com ângulo Θ_1= 45° e Θ_2 = 45°

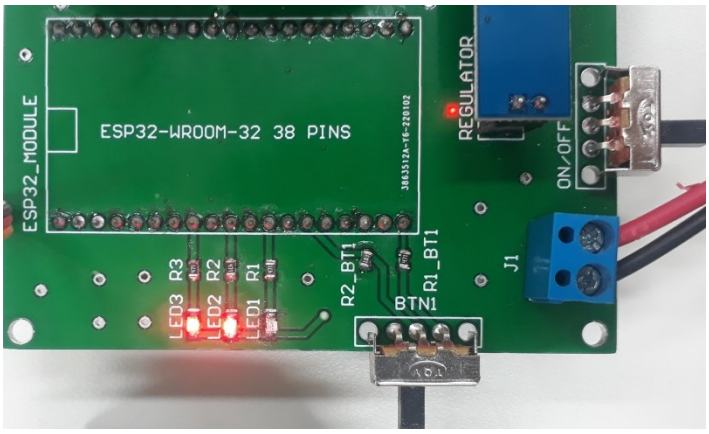


Fonte: O autor.

De acordo com os resultados apresentados, observa-se que a simulação funcionou corretamente, movendo o robô para a posição desejada, e os cálculos da posição do efetuador foram realizados de forma assertiva.

Terminada a apresentação dos resultados da parte de simulação de robótica, são apresentados os resultados relacionados ao robô físico. Primeiramente, são apresentados os estados do LEDs que indicam o estado de funcionamento do robô. Estando o robô conectado com a rede Wi-Fi e com o *broker*, na Imagem 51 é apresentada a placa em modo configuração (chave CH_MODO no nível baixo de tensão).

Imagem 51 – PCB no modo configuração

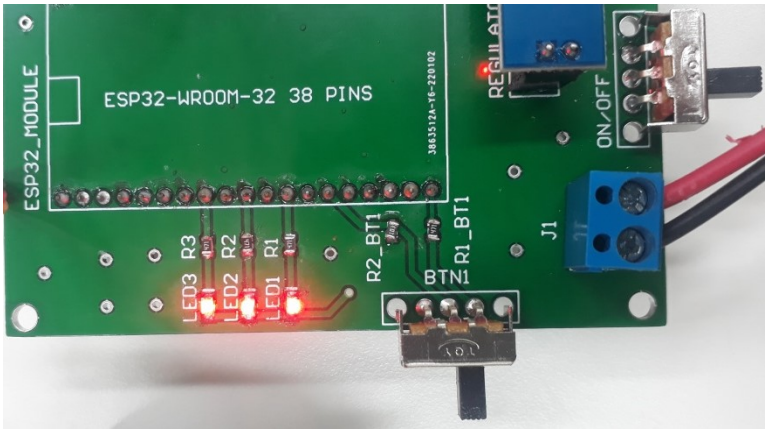


Fonte: O autor.

Uma vez que a placa está no modo de configuração, a atualização do *firmware* via OTA é habilitada e funcionou corretamente.

Na Imagem 52 é apresentada a placa em modo de operação (chave CH_MODO no nível alto de tensão).

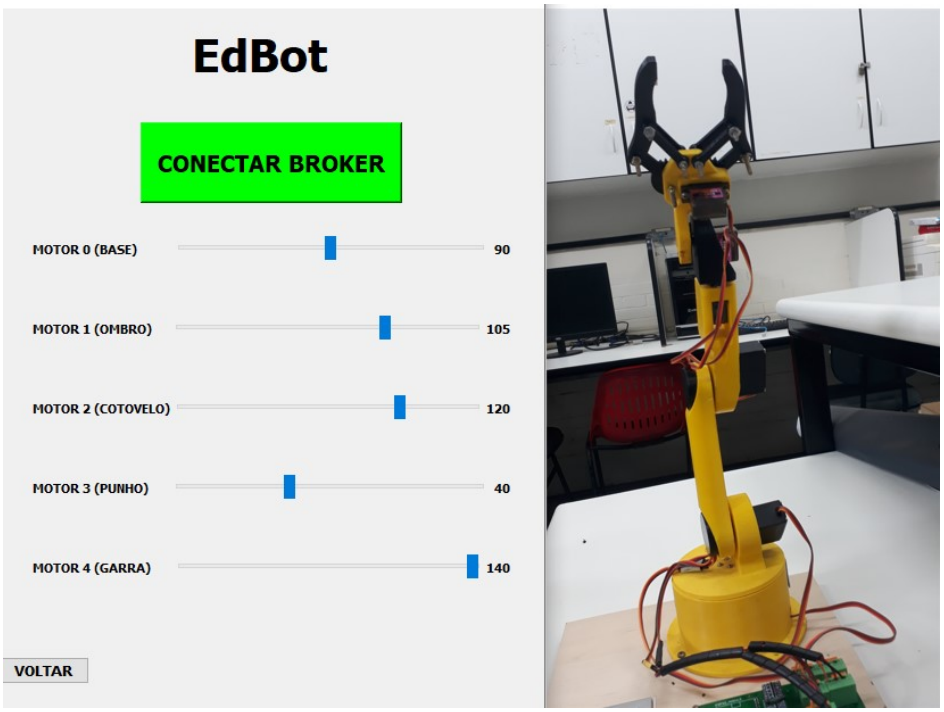
Imagem 52 – PCB no modo operação



Fonte: O autor.

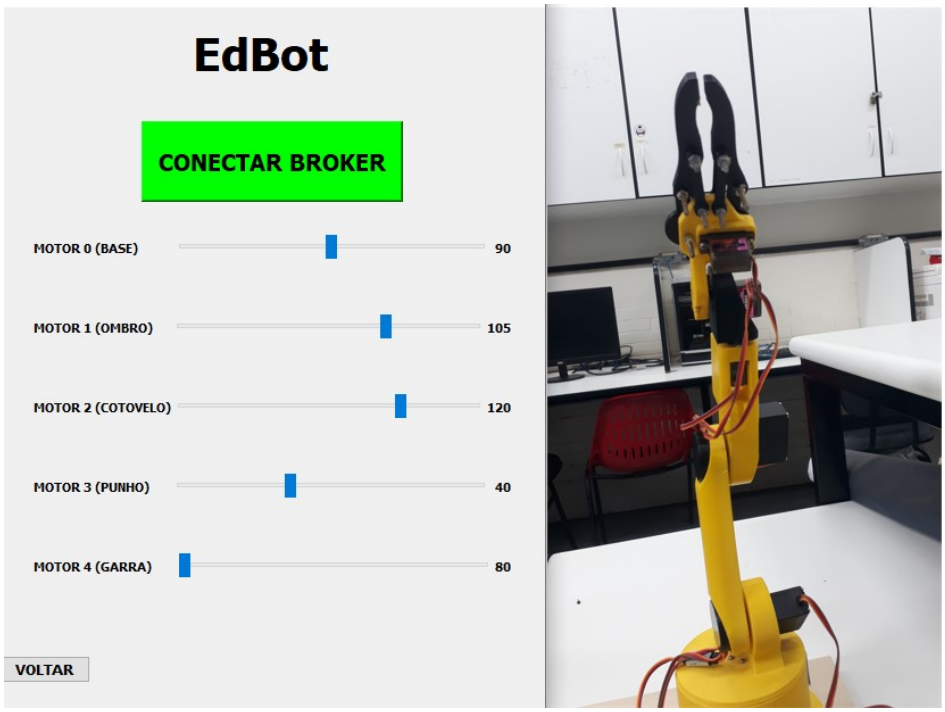
No modo de operação é possível controlar o robô via MQTT, utilizando a interface apresentada na Imagem 22. Foi realizado o teste de todos os motores do robô. Nas Imagens 53 e 54 é apresentada a movimentação de um dos elos testados, a garra.

Imagem 53 – Teste de abertura da garra



Fonte: O autor.

Imagem 54 – Teste de fechamento da garra



Fonte: O autor.

Utilizando o programa MQTTBox é possível verificar os comandos publicados no tópico MQTT. Na Imagem 55 é exibida a captura de tela apresentando a mensagem responsável por fechar totalmente a garra:

Imagem 55 – Mensagem MQTT observada no programa MQTTBox

Fonte: O autor.

De acordo com os resultados apresentados, observa-se que foi possível movimentar o motor da garra do robô via MQTT e, de modo semelhante, foi possível movimentar os demais elos. Um aspecto importante observado neste teste é que a movimentação dos elos ocorre com um pequeno intervalo após o comando ser enviado pela interface. Isto se deve,

principalmente, ao *delay* adicionado na função de movimentação dos motores. Além disso, observa-se que a movimentação do robô ocorre com algumas oscilações, o que também pode estar relacionado à estratégia de controle.

5 DISCUSSÃO

Por meio dos resultados apresentados, observa-se que todas as etapas planejadas foram executadas nos testes. Em relação ao teste do robô físico, é importante ressaltar que sempre foi testado o acionamento de um motor por vez. Essa estratégia diminui o consumo de corrente elétrica durante a operação se comparado ao acionamento de vários motores no mesmo momento. Todavia, esse procedimento aumenta o tempo para que o robô alcance determinado local do espaço.

Em relação à simulação, é importante destacar que todos os testes foram realizados no sistema operacional Windows. Além disso, observa-se que o programa V-REP permite a abertura de apenas uma instância para execução da simulação por vez. Desse modo, caso o usuário deseje utilizar as duas cenas apresentadas neste projeto, é necessário que ele se atente para fechar a cena anterior ao abrir uma nova.

6 CONCLUSÃO

Tendo em vista todas as informações apresentadas neste projeto, conclui-se que o mesmo alcançou o objetivo de desenvolver uma plataforma de robótica que auxilie no ensino do conteúdo de robótica em aulas de graduação. De modo que todas as partes da plataforma proposta foram desenvolvidas: o *software* central com interface gráfica, a conexão deste *software* com o programa de simulação V-REP e o robô físico controlado por uma placa embarcada que se comunica com o *software* central via MQTT.

O *software* desenvolvido simplifica o trabalho com o simulador V-REP. Dessa forma, as duas cenas desenvolvidas facilitam a visualização dos conceitos de robótica abordados, os quais, apesar de não serem muito complexos, são difíceis de compreender sem o auxílio de uma imagem.

O sistema embarcado projetado e desenvolvido permite o acionamento de robôs diversos. Portanto, é possível utilizar diferentes configurações de robôs de acordo com a exigência das tarefas. Uma vez que o firmware elaborado permite a atualização via OTA, o código pode ser facilmente alterado.

Ademais, o uso do protocolo MQTT para a comunicação entre o sistema embarcado e o software possibilita que o robô possa ser operado remotamente. Dessa forma, o operador pode acionar o robô mesmo estando em um ambiente separado do robô, o que é útil tanto no estudo futuro de aplicações teleoperadas como facilita o uso didático do robô em situações em que os alunos não estejam no mesmo ambiente do robô.

Devido à grande abrangência deste trabalho, o mesmo pode auxiliar no desenvolvimento de trabalhos futuros. Primeiramente, sugere-se que em trabalhos futuros a PCB seja corrigida, de forma que o módulo ESP32 possa ser soldado corretamente na camada superior da placa. Além disso, indica-se a adequação dos cabos dos motores do robô para que seja possível a movimentação do robô em uma área maior, pois no trabalho atual a área de movimentação foi reduzida para evitar situações em que os cabos possam ser muito esticados ou travarem um deslocamento.

Em relação ao *firmware*, sugere-se que em trabalhos futuros a estratégia de controle possa ser aprimorada com emprego de algum tipo de controlador mais robusto. Dessa forma, o deslocamento do robô poderá ser efetuado de forma mais suave. Ademais, indica-se a melhoria da função de acionamento dos servos motores para que sejam possíveis movimentações com passo menor do que um grau, o que aumentaria a precisão do deslocamento.

Além disso, sugere-se a adição de uma função no *software* que permita a movimentação do robô entre determinados pontos definidos. Dessa forma, o usuário informaria no programa os conjuntos de pontos em que o robô deve passar e eles seriam utilizados para determinar a rota do robô para o cumprimento de alguma tarefa.

Já em relação à simulação, indica-se o desenvolvimento de novas cenas que abordem outros conteúdos da área da robótica, facilitando o aprendizado deles. Outra adição interessante seria a criação de uma cena com um robô semelhante ao utilizado no experimento prático. Dessa forma, seria possível comparar com mais facilidade a simulação de um robô e o acionamento de um robô físico.

REFERÊNCIAS

ALTIUM. **Student License**. Disponível em: <https://www.altium.com/documentation/knowledge-base/altium-designer/student-license>. Acesso em: 1 mar. 2022.

AMARAN, M. H. et al. A Comparison of Lightweight Communication Protocols in Robotic Applications. **Procedia Computer Science**, v. 76, n. Iris, p. 400–405, 2015.

BLANCHON, B. **ArduinoJson**. Disponível em: <https://github.com/bblanchon/ArduinoJson>. Acesso em: 2 mar. 2022.
Coppelia Robotics. Disponível em: <https://coppeliarobotics.com/features>. Acesso em: 26 fev. 2022.

CRAIG, J. J. **Robótica**. 3. ed. São Paulo: Pearson, 2013.

DE MELO, M. S. P. et al. Analysis and comparison of robotics 3D simulators. **Proceedings - 2019 21st Symposium on Virtual and Augmented Reality, SVR 2019**, p. 242–251, 2019.

DEJAN. **DIY Arduino Robot Arm with Smartphone Control**. Disponível em: <https://howtomechatronics.com/tutorials/arduino/diy-arduino-robot-arm-with-smartphone-control/>. Acesso em: 1 mar. 2022.

ESPRESSIF. **Wi-Fi**. Disponível em: <https://docs.espressif.com/projects/esp-idf/en/latest/esp32/api-reference/network/esp_wifi.html>. Acesso em: 2 mar. 2022.

ESPRESSIF. ESP32-WROOM-32 Datasheet. 2013.

FERNANDES, C. C. **S-Educ : Um Simulador de Ambiente de Robótica Educacional em Plataforma Virtual**. [s.l.] Universidade Federal do Rio Grande do Norte, 2013.

FRUGOLI, P. F. F.; PEREIRA, I. P. Comparação dos protocolos MQTT e HTTP para IoT. 2019.

GOMES, L. G. D.; MARQUES, A. L.; RIBEIRO, L. Low-Cost Data Glove for Robotic Hand-Forearm Teleoperation Systems Applications. 2021.

IFR. **IFR presents World Robotics Report 2020**. Disponível em: <https://ifr.org/ifr-press-releases/news/record-2.7-million-robots-work-in-factories-around-the-globe>. Acesso em: 18 fev. 2022.

IFR. **Robot History**. Disponível em: <https://ifr.org/robot-history>. Acesso em: 17 fev. 2022.
LYNCH, K. M.; PARK, F. C. **Modern Robotics: Mechanics, Planning, and Control**. Cambridge: Cambridge University Press, 2017.

MATARIC, M. J. **Introdução à Robótica**. 1. ed. São Paulo: UNESP, 2014.

MATHIAS, A. P. **IEEE 802.11 - Redes sem Fio**. Disponível em: <https://www.gta.ufrj.br/grad/00_2/ieee/index.html>.

MG995. Disponível em: <https://www.electronicoscaldas.com/datasheet/MG995_Tower-Pro.pdf>. Acesso em: 23 fev. 2022.

MQTT: The Standard for IoT Messaging. Disponível em: <https://mqtt.org/>. Acesso em: 25 fev. 2022.

MQTT. Disponível em: <https://www.gta.ufrj.br/ensino/eel878/redes1-2019-1/vf/mqtt/>. Acesso em: 25 fev. 2022.

O'LEARY, N. **PubSubClient**. Disponível em: <https://github.com/knolleary/pubsubclient>. Acesso em: 2 mar. 2022.

ORACLE. **What Is a Socket?** Disponível em: <https://docs.oracle.com/javase/tutorial/networking/sockets/definition.html>. Acesso em: 28 fev. 2022.

PAZOS, F. **Automação de Sistemas e Robótica**. 1. ed. Rio de Janeiro: Axcel Books, 2002.
PyQt5 5.15.6. Disponível em: <https://pypi.org/project/PyQt5/>. Acesso em: 28 fev. 2022.

Remote API. Disponível em: <https://www.coppeliarobotics.com/helpFiles/en/remoteApiOverview.htm>. Acesso em: 28 fev. 2022.

Remote API modus operandi. Disponível em: <https://www.coppeliarobotics.com/helpFiles/en/remoteApiModusOperandi.htm>. Acesso em: 28 fev. 2022.

ROHMER, E.; SINGH, S. P. N.; FREESE, M. V-REP: A versatile and scalable robot simulation framework. **IEEE International Conference on Intelligent Robots and Systems**, p. 1321–1326, 2013.

ROMANO, V. F. **Robótica Industrial: Aplicação na Indústria de Manufatura e de Processos**. [s.l.] Editora Edgard Blücher LTDA, 2002.

SANTOS, V. M. F. **Robótica Industrial**. Aveiro: Universidade de Aveiro, 2004.

SEIDL, M. et al. **UML@Classroom: An introduction to object-oriented modeling**. [s.l.] Springer, 2015. v. 1555

SICILIANO, B. et al. **Robotics: Modeling, Planning and Control**. Glasgow: Springer, 2009.

SILVEIRA, C. B. **Servo Motor: Veja como Funciona e Quais os Tipos**. Disponível em: <https://www.citisystems.com.br/servo-motor/>. Acesso em: 23 fev. 2022.

Test Mosquitto. Disponível em: <https://test.mosquitto.org/>. Acesso em: 3 mar. 2022.

YUAN, M. **Conhecendo o MQTT**. Disponível em: <https://developer.ibm.com/br/articles/iot-mqtt-why-good-for-iot/>. Acesso em: 25 fev. 2022.