TDP IEEE OPEN 2023 - EQUIPE ATENA - EESC/USP

Amanda Tanaka, Diego Fleury, Fernando Cristóvão, Francisco Affonso,
Gabriel D'Acol, Gianluca Capezzuto, João Henrique Aléssio,
João Vitor de Oliveira, Tainara Mareco

***Abstract*— This document describes the main aspects of the solution proposed by the Atena Team for automation of warehouses and distribution centers, as found in companies such as Amazon and Alibaba.**

***Resumo*— Este documento descreve os principais aspectos da solução proposta pela Equipe Atena para a automação de armazéns, centros de distribuição e depósitos, como ocorre em empresas como Amazon e Alibaba.**

I. INTRODUÇÃO

Com o advento da globalização, há um forte estímulo para a busca incessante por maior eficiência nos custos, flexibilidade e autonomia em sistemas logísticos de larga escala. Exemplos de companhias que aplicam grande foco e esforços neste paradigma de *warehousing* são a Amazon e Alibaba. O uso responsável de recursos humanos é uma grande preocupação, dado isso, tarefas de alta repetibilidade, grande demanda de uso de tempo e com alta estruturação do ambiente, chamam a atenção à aplicação de sistemas robóticos autônomos, de modo a permitir uma alocação melhor de pessoas dentro das tarefas as quais são fundamentalmente necessárias, melhorando a eficácia do processo como um todo.

Sob este paradigma, algumas habilidades necessárias para os sistemas robóticos autônomos seriam o apanhamento, ordenação, transporte e armazenamento de objetos de interesse. A competição IEEE Open 2023 tem como objetivo a simulação de uma versão desta problemática, de modo a explorar as possibilidades de aferição de diferentes técnicas, ideias e conceitos que podem ser usadas para o crescente incremento de qualidade na realização desta automação tão crucial para a economia moderna.

Este TDP (*Team Description Paper*) contém a descrição das soluções propostas pela equipe Atena do grupo de robótica SEMEAR (Soluções em Engenharia Mecatrônica e Aplicação na Robótica), da Universidade de São Paulo (USP) - campus São Carlos. O grupo é constituído por estudantes de graduação dos mais diversos cursos oferecidos no campus citado, no envolvendo de engenharia e computação.

II. ESTRUTURA DO ROBÔ

Agora, será apresentado os módulos da estrutura do robô. Tal secção permitirá compreender todas as *features* implementadas para possibilitar a execução da estratégia de navegação e planejamento que serão apresentadas posteriormente.

A. Estrutura geral do robô

O robô possui uma cinemática omnidirecional, proporcionado por um conjunto de quatro rodas tracionadas independentemente, logo possuindo quatro motores.

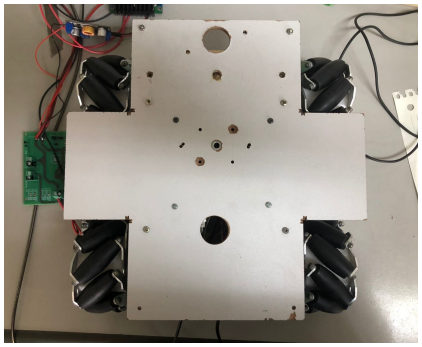

Fig. 1. Disposição das rodas

Os motores que dão tração às rodas são DC de 1030 RPM com tensão nominal de 12 V e torque em *stall* de 3,2 kgf.cm. Além disso, para possibilitar uma medição de sua rotação há acoplado um encoder de 48 CPR em cada um (modelo *Pololu* 9.7:1 HP 12V *with encoder*).

Fig. 2. Motor DC com encoder [8].

As rodas são do tipo *Mecanum Wheels* (Figura 3), que possuem rolamentos em todo o seu perímetro, com eixos de rotação que formam ângulos de 45° com o eixo de rotação da roda. Isso permite a navegação no sentido perpendicular ao habitual. Dessa forma, o robô tem a capacidade de transladar para os quadrantes vizinhos do cenário sem a necessidade realizar qualquer rotação sobre seu eixo. Portanto, possui três graus de liberdade e é holonômico.

Fig. 3. *Mecanum Wheel* [7].

Os dados de deslocamento serão enviados para o componente de *software* responsável pela tomada inteligente de decisões. Isso é feito através do uso de microcontroladores ESP32 (modelos *DevKit*), com o uso do protocolo de comunicação UART. O recebimento de valores de comando (providos pelo *software* que implementa a estratégia) de velocidade resulta em ações de controle a serem seguidas pelo programa presente em cada microcontrolador.

Além disso, a carcaça do robô conta com uma estrutura que permite o armazenamento dos circuitos eletrônicos e baterias em dois andares. Dessa forma, conseguimos dispor os componentes da seguinte maneira:

- Placa dos motores - composta pelos dois microcontroladores e pontes H
- Placa de potência - composta pelos reguladores de tensão e o microcomputador (Raspberry Pi 4)
- Placa de controle - composta por uma IMU (*Inertial measurement unit*), motor de passo e servo

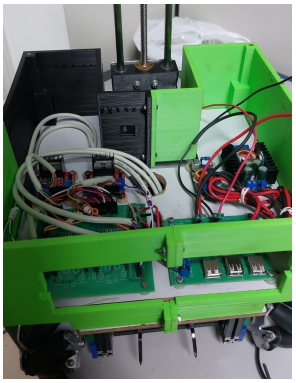

Fig. 4. Primeiro Andar.

A carcaça externa foi impressa em material ABS (Acrilonitrila Butadieno Estireno) por uma impressora 3D, servindo como suporte para o elevador da garra e para a fixação da câmera lateral.

Para o tratamento dos dados recebidos pela ESP32, foi idealizado uma metodologia de controle PID (Figura 7). A identificação dos valores adequados para o direcionamento

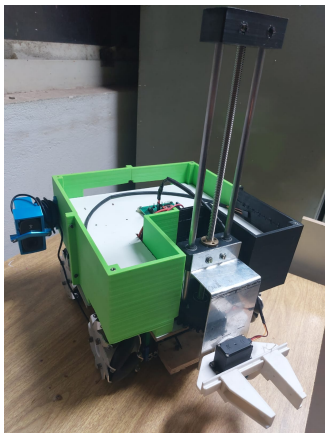

Fig. 5. Carcaça.

do sistema fora feita com vários testes, em condições diversificadas de velocidade, para abrangir uma gama maior de segurança e robustez. Depois de todo o processo, fora feita a modelagem de uma planta (em termos do jargão comumente utilizado no campo de controle) completa utilizando a informação conhecida de entrada e a resposta obtida pela leitura dos encoders, à fim de encontrar os melhores parâmetros (Proporcional, Integral e Derivativo) para a realização do controle em baixo nível.

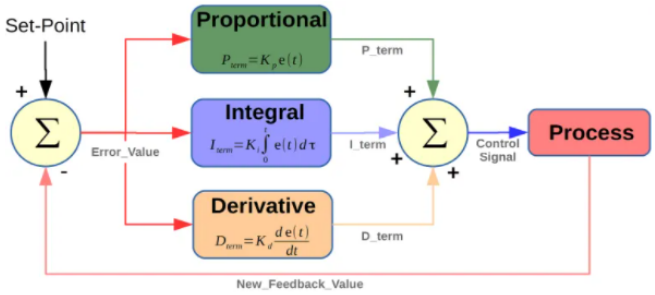

Fig. 6. Controlador PID [1].

B. Garra

O mecanismo de captura é constituído de uma plataforma de elevação na parte frontal do robô com uma garra fixada no elevador. A plataforma de elevação possui duas barras lisas para suporte do mecanismo da garra e uma barra roscada acoplada a um motor de passo (NEMA 17, Fonte: [11]) para maior precisão da posição da garra na subida e descida. A garra é acionada por um micro servo e possui um mecanismo semelhante ao de uma Morsa, podendo se ajustar de acordo com o tamanho do bloco a ser capturado, evitando uma possível perda ao longo do caminho.

O projeto proposto envolve a incorporação de uma Raspberry Pi Camera v1 ao mecanismo de elevação do robô. Essa câmera, quando posicionada voltada para baixo, desempenha uma função crucial ao identificar o tipo de bloco presente e sua localização na área de trabalho. Essa capacidade de

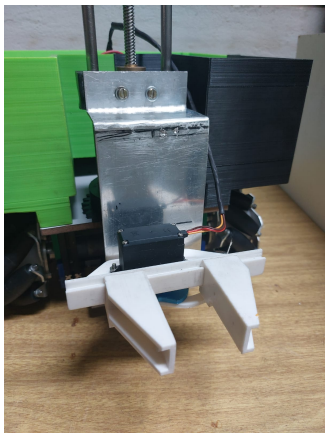

Fig. 7. Mecanismo de captura.

reconhecimento é de suma importância para aprimorar o desempenho geral do robô, permitindo uma locomoção mais eficiente e precisa.

Por meio da análise das imagens obtidas pela câmera, o robô utilizará técnicas de visão computacional para distinguir os diferentes tipos de blocos, considerando suas características únicas, como cor e marcações específicas.

Com os dados fornecidos pela câmera, o robô poderá centralizar adequadamente a garra em relação aos blocos, em um sistema de controle de malha fechada, garantindo assim uma captura precisa e eficiente dos objetos. Além disso, o robô poderá se movimentar de maneira otimizada.

Vale ressaltar que a implementação da câmera no mecanismo de elevação foi projetada de modo a garantir que a câmera esteja adequadamente posicionada para capturar as imagens necessárias de forma clara e sem obstruções.

C. Módulo de alimentação

Para o funcionamento adequado do robô, todo o seu circuito eletrônico será alimentado por duas baterias, de 2 células de polímero de lítio (Li-Po) cada, com tensões de $14,8V$ e $7,4V$.

A primeira bateria escolhida, de $14,8V$, será destinada para a alimentação dos circuitos dos 4 motores *DC* e do motor de passo (responsável pelo funcionamento do elevador frontal). Essa bateria possui capacidade de $5200mAh$ e corrente de descarga de $16,67mAh$ e foi escolhida para fornecer uma tensão de $12V$, o que torna necessária a utilização de um regulador de tensão para o uso adequado, e seguro, dessa bateria.

Já a outra bateria escolhida, de $7,4V$, será destinada para a alimentação dos seguintes componentes: um servo motor (em uso para o fechamento da garra), o componente *driver* do motor de passo, um sensor IMU, o microcomputador *Raspberry Pi 4*, 2 microcontroladores ESP32, uma câmera lateral, uma câmera frontal e um elemento de resfriamento (*cooler*) para o microcomputador. Essa bateria possui capacidade de $6000mAh$ e corrente de descarga de $8,33mAh$ e foi escolhida para suprir a necessidade de uma alimentação de $5V$, o que torna necessária a utilização de mais um regulador de tensão para converter a tensão de $7,4V$ para uma tensão de $5V$.

Para mitigar os riscos de acidente, foi adicionado um circuito de proteção para cada bateria. Dessa forma, caso ocorra algum curto em qualquer circuito, a alimentação será interrompida, podendo ser reconectada após a correção das falhas.

D. Sistema de sensoriamento global

Como será apontado na secção de estratégia e planejamento, a forma de modelagem do problema envolve o amplo uso de *features* marcantes do ambiente de trabalho, que servem de guia para obter-se um robusto senso de localização global. Dado isso, toda a projeção do sistema de sensoriamento fora centrada no aproveitamento de tais *features*.

Na lateral do robô há uma câmera (Webcam HD Logitech C270, Figura 8), cujo objetivo é visualizar o ambiente externo para então, através de visão computacional, extrair aspectos-chave do ambiente (como a prateleira, linhas no chão, blocos e seus tipos, etc.) e com isso não somente fornecer informações essenciais à resolução do problema, como também auxiliar na estimação da localização global do robô.

Fig. 8. Câmera Lateral [14].

Além disso, o robô conta com uma *Inertial Measurement Unit*, também chamada de IMU MPU-9250 (Figura 9), responsável por complementar a estimativa de localização global, por meio do uso de sua informação de angulação, que em conjunto com os dados dos encoders, selecionamos o tipo 48-CPR, apresentadas na Figura 2, trarão mais precisão à estimativa do deslocamento.

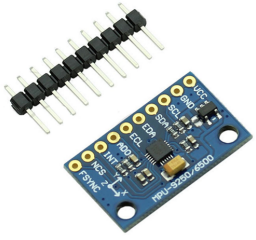

Fig. 9. *Inertial measurement unit* [13].

O sistema fora projetado para permitir o uso de sensores infravermelhos (com o sensor TCRT5000), fixados na lateral

e na frente do robô, de modo que possam detectar a passagem do robô pelos quadrantes, pelas linhas que os separam, sendo uma redundância para a localização global do robô.

III. NAVEGAÇÃO E PLANEJAMENTO

A. *Estratégia Empregada*

A estratégia proposta para o desafio consiste em mapear o cenário para um sistema de coordenadas, determinando um par ordenado (x, y) para cada quadrante delimitado pelas faixas pretas da plataforma do desafio. O sistema de coordenadas adotado pode ser observado na Figura 10.

(0,0)	(0,1)	(0,2)	(0,3)	(0,4)	(0,5)	(0,6)
(1,0)	(1,1)	(1,2)	(1,3)	(1,4)	(1,5)	(1,6)
(2,0)	(2,1)	(2,2)	(2,3)	(2,4)	(2,5)	(2,6)
(3,0)	(3,1)	(3,2)	(3,3)	(3,4)	(3,5)	(3,6)
(4,0)	(4,1)	(4,2)	(4,3)	(4,4)	(4,5)	(4,6)
(5,0)	(5,1)	(5,2)	(5,3)	(5,4)	(5,5)	(5,6)
(6,0)	(6,1)	(6,2)	(6,3)	(6,4)	(6,5)	(6,6)

Fig. 10. Sistema de coordenadas adotado para navegação no cenário.

A partir dessa definição é produzido um grafo utilizando representação por listas de adjacências, em que cada vértice do grafo corresponde a uma coordenada navegável (quadrados em branco).

Existem dois intuitos por trás desta decisão de particionamento do espaço de trabalho. A primeira diz respeito ao modelo de controle utilizado. Para a estimativa de posicionamento global, o problema torna-se mais simples se tratado em termos de deslocamentos e rotações simples entre dois instantes de tempo. A manutenção de estados internos de localização é assim facilitada, aumentando a robustez do controlador de alto nível. O segundo intuito desta escolha é conseguir determinar sempre o caminho mais curto entre dois pontos do cenário, utilizando para isso o algoritmo de Busca em Largura (*Breadth-First Search*). Dessa forma, armazenando sempre a posição atual do robô, e ponderando corretamente cada translação e rotação necessária para o movimento de um setor ao próximo, consegue-se obter uma sequência de ações de controle necessárias para alcançar certa posição, obtendo sempre um caminho ótimo.

O armazenamento (e manutenção) da posição atual é feito utilizando o sistema de sensoriamento global, que somado ao conhecimento da posição inicial (a ser determinada a partir da análise de estruturas no ambiente) no tabuleiro, é capaz de saber a sua posição no mapa a qualquer instante de tempo. Para cada movimento translacional em direção a um quadrado adjacente, são utilizados os sensores infravermelhos para identificar a linha preta entre os quadrados, contabilizando a coordenada da posição atual respectiva ao movimento realizado (sendo essa informação usada para o envio de comandos de controle para o módulo de locomoção do robô).

A Figura 11 representa o diagrama da estratégia geral empregada. O robô inicia em uma posição aleatória e, utilizando o grafo, calcula a distância entre sua posição inicial e as coordenadas (4,1), (4,2), (4,4) e (4,5), locomovendo-se para a mais próxima. Supondo que a coordenada mais próxima seja (4,4), o robô se posicionaria no centro da linha preta entre as coordenadas (4,4) e (4,5), utilizando um controle (a partir do sistema de sensoriamento global, somado aos encoders). A partir dessa posição, captura-se uma foto, através da câmera colocada na lateral do robô, de toda região cinza à direita na Figura 10. Através dessa imagem, é aplicado um algoritmo de visão computacional para mapear todos os pacotes existentes em cada coordenada dessa região. Posteriormente, o robô é deslocado de forma que seu centro seja posicionado na linha entre as coordenadas (4,1) e (4,2), realizando assim o mesmo procedimento de mapeamento dos pacotes para a região cinza à esquerda.

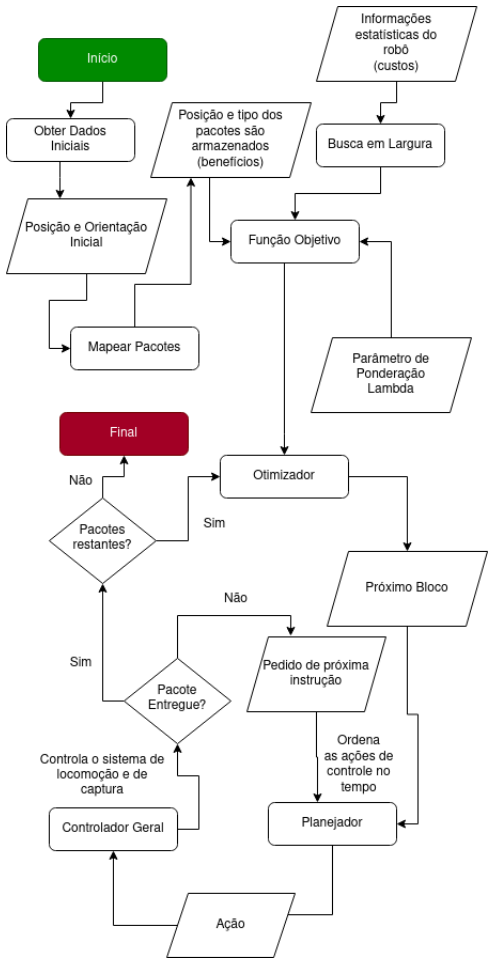

Fig. 11. Diagrama geral da estratégia.

A escolha de algoritmo para determinação da sequência de blocos é otimização binária, com uma formulação gulosa da

solução do caixeiro viajante, adicionando-se uma heurística à função objetiva para melhorar sua qualidade. Como o problema em sua forma geral exigiria uma complexidade computacional muito elevada (por ter de considerar todas as escolhas, em todos os instantes de tempo, e com todos os custos que vem junto de cada escolha) opta-se por uma versão simplificada do problema, considerando apenas a próxima decisão.

Tem-se, a cada escolha, dois conjuntos de valores a serem ponderados: os custos de se pegar um dado bloco (determinado pela busca em largura, assumindo o melhor caminho tomado) e os benefícios (pontuação) de optar por aquele bloco para aquele instante de tempo (isso naturalmente inclui casos a parte, como, por exemplo, o fato que, segundo o edital, cada primeiro bloco de cada tipo coletado e entregue vale o dobro de pontos, sendo isso naturalmente incorporado nessa solução). Deseja-se maximizar a quantidade de pontuação que conseguimos, dado o custo de escolher aquele bloco. Esta estratégia é muito simples, entretanto, levando a uma solução demasiada gananciosa.

$$f(\{X_b\}) = \sum_{b=0}^{B} \frac{Pontos(b)}{Custo(b)} \times X_b \tag{1}$$

Nesta equação, X_b é uma variável binária (0 ou 1) de escolha do bloco b como próximo a ser o capturado (restrições do problema omitidas por simplicidade), e deseja-se obter o melhor custo-benefício como explicitado para aquele bloco, naquele instante de tempo.

Para incorporar um elemento de escolha mais complexo, de forma a capturar uma melhor resolução do problema, optou-se pela adição de uma ponderação (do próximo custo-benefício, $novo\_CB(X_b)$) de modo a considerar escolhas que levem o nosso robô a uma situação ruim no futuro (por exemplo, longe de blocos valiosos). Resumidamente, caso nossa escolha de bloco no presente limite demais nosso custo-benefício na próxima decisão a ser tomada, talvez seja melhor escolher outro bloco. Isso é feito fazendo essa modificação na função objetivo de custo-benefício atual, com uma constante de proporcionalidade (λ) a ser determinada experimentalmente. Dessa forma, consegue-se modelar não somente o fato que todos os tipos de blocos possuem múltiplos locais de entrega (aumentando os caminhos viáveis para o robô) mas também adiciona-se um elemento de busca menos gulosa, com um parâmetro livre (λ) para adequar-se às necessidades de entrega e coleta particulares do cenário.

$$f(\{X_b\}, \lambda) = \left(\sum_{b=0}^{B} \frac{Pontos(b)}{Custo(b)} \times X_b + \lambda \times novo\_CB(X_b) \right) \tag{2}$$

B. ROS (Robotic Operating System)

Neste projeto, será utilizado a *framework Robot Operating System* (ROS, traduzido para "Sistema Operacional Robótico", Fonte: [9]), que auxilia o desenvolvimento de sistemas robóticos.

Uma das características mais notáveis dessa ferramenta é sua capacidade de concretizar a modularização de um sistema. ROS possui um sistema de "nós", arquivos executáveis que podem estar desenvolvidos em linguagens distintas. Cada nó é responsável por uma função diferente do robô, facilitando a execução de tarefas de forma paralela. Uma estrutura de programas chamada de *roscore* é responsável por gerenciar os diferentes nós de um mesmo sistema, além de armazenar parâmetros globais, que podem ser consultados por qualquer parte da organização.

Para trocarem dados, os nós utilizam uma estrutura chamada de Tópico ROS. Ele é um canal de comunicação entre diferentes partes do robô. Exemplificando, um nó responsável por fazer a leitura de sensores se inscreve no papel de publicador de um tópico, enquanto que outro, responsável pelo tratamento desses dados, se torna assinante do tópico, recebendo os dados. A inscrição ou subscrição a um tópico é informada e gerenciada pelo *roscore*. Vale ressaltar que as informações são estruturadas em Mensagens ROS. Assim, uma mensagem pode conter informações como translação e rotação do robô em vários eixos, enviadas de uma só vez. Esse processo pode ser visualizado na Figura 12.

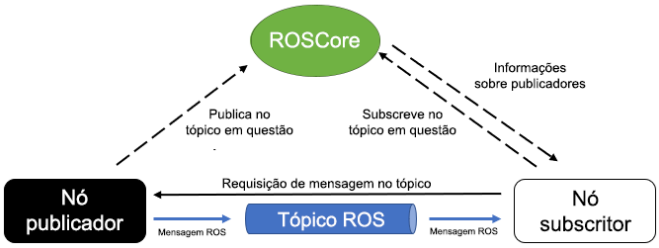

Fig. 12. Esquema de comunicação entre nós utilizando [10].

C. Visão Computacional

Vale a pena lembrar que a estimativa de posição inicial será feita com a utilização do OpenCV.

Para a detecção dos pacotes do cenário foi desenvolvido um algoritmo, com duas etapas distintas, utilizando a biblioteca de visão computacional e de processamento de imagens OpenCV, através da linguagem de programação C++.

A primeira etapa receberá como entrada as imagens capturadas pela câmera lateral do robô de cada uma das regiões que contém blocos. Estas imagens serão utilizadas para que o robô consiga realizar um mapeamento do espaço e encontrar a posição de cada um dos blocos dispersos nas áreas de captura do cenário e analisar quais são seus tipos, ou seja, se é um bloco vermelho, azul, verde, amarelo, branco ou preto. O reconhecimento prévio das características de posição e cor dos blocos, permitirá com que algoritmo de decisão de coleta dos pacotes e o de decisão da estratégia empregada, escolham quais são os melhores alvos para a captura.

A etapa seguinte receberá como entrada a imagem capturada pela câmera. Após a etapa de reconhecimento e mapeamento do cenário, segue-se com a identificação das

faces de cada uma das caixas coletadas (para aquelas que se vê necessário, ou seja, as não coloridas), a partir do tipo detectado na fase anterior, utilizando-se método de visão computacional independentes. Para os blocos brancos, o algoritmo será capaz de extrair as letras do pacote a partir de uma rede neural pré-treinada, denominada Tesseract. Para os blocos pretos o algoritmo será capaz de identificar estruturas de ArUco em suas faces, por meio de conceitos de estimativa de pose.

Estimativa de pose é uma das principais ferramentas utilizadas na visão computacional, esse método consiste em achar correspondências entre um objeto real e sua projeção de imagem 2D, a proposta do ArUco é simplificar esse processo. O OpenCV possui uma biblioteca específica para o ArUco e por meio dela é possível determinar a localização do marcador no espaço usando os seus quatro cantos. Para que o algoritmo funcione corretamente é necessário que o fundo onde o ArUco está seja branco, como o cenário da competição não proporciona essa condição, iremos tratar essa imagem a fim de obter o resultado adequado. Além disso, cada ArUco é formado por uma matriz binária interna que se refere a sua identificação (id=número) dentro do seu dicionário, permitindo diferenciar os blocos desse tipo. Uma vista de perspectiva dos blocos alfanuméricos e com arUco dispostos no cenário do desafio pode ser visto na Figura 13.

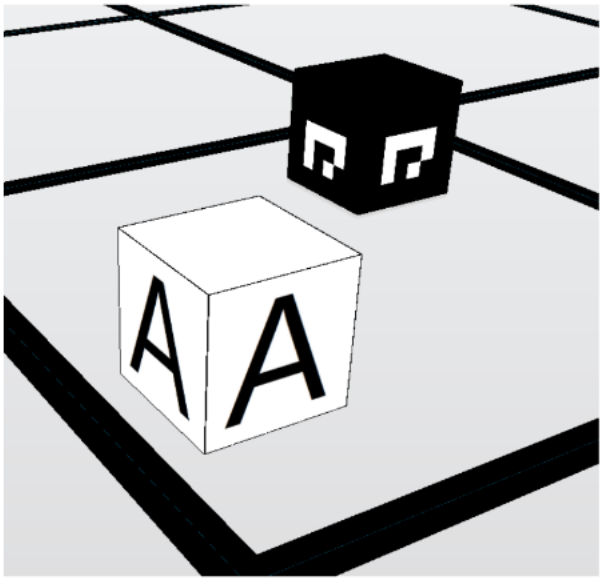

Fig. 13. Vista de perspectiva dos blocos alfanuméricos e com ArUco.

IV. CONCLUSÃO

O encadeamento das soluções propostas pelo grupo está segundo as regras impostas pela IEEE Open e está alinhado com o problema apresentado. Toda a estrutura robótica foi proposta para a maior rapidez e eficácia da operação de carregamento dos pacotes. Tudo isso levando em conta as respectivas pontuações de cada tipo de pacote.

O uso de visão computacional para determinar a posição inicial dos blocos e do robô em relação ao ambiente, aliado ao algoritmo de procura da melhor ordem de ações, garante uma operação estratégica e eficiente. Além disso, os dados obtidos pelos *encoders* e sistema de sensoriamento global, associados ao movimento omnidirecional das rodas escolhidas, permitem um ótimo controle da posição do robô.

Em relação ao mecanismo de captura, o sistema de pinças laterais fará com que a deposição dos pacotes nas prateleiras não encontre problemas de colisão com a estrutura. Além disso, com o motor de passo para controlar o deslocamento vertical, não haverá problemas em determinar a altura correta.

Em suma, a equipe desenvolveu um projeto pensando não apenas na competição e em suas regras, mas também no problema real de engenharia proposto. Todos os módulos e tecnologias usadas foram pensados para ampliar a eficiência e diminuir o tempo das operações de manejo de pacotes em um contexto de armazém automatizado.

AGRADECIMENTO

Os autores agradecem o apoio da EESC-USP com a sua infraestrutura e o suporte dos orientadores da equipe SEMEAR, Prof. Dr. Marcelo Becker, Profa. Dra. Maíra Martins da Silva e Prof. Dr. Rogério Andrade Flauzino.

REFERÊNCIAS

[1] LOOP TECNOLOGIA INDUSTRIAL. Disponível em : https://www.ltiengenharia.com.br/2021/05/11/sistemas-de-controle-controlador-pid-parte-02/. Acesso em: 29 de julho de 2023.

[2] MERCADOLIVRE, Webcam I374 3556 Notebook Positivo Premium S6055. Disponível em: https://produto.mercadolivre.com.br/MLB-1675561336-webcam-i374-3556-note-positivo-premium-s6055-cod-100877-_JM. Acesso em: 04 de julho de 2022.

[3] MERCADOLIVRE, Esp32 Doit Devkit Com Esp32-wroom-32. Disponível em : https://produto.mercadolivre.com.br/MLB-2200163889-esp32-doit-devkit-com-esp32-wroom-32-_JM#position=2&search_layout=grid&type=item&tracking_id=29694c7d-924c-4b7e-ac25-9f650fbef430. Acesso em: 28 de julho de 2023.

[4] FILIPEFLOP, Sensor de Obstáculo Infravermelho IR. Disponível em: https://www.filipeflop.com/produto/sensor-de-obstaculo-infravermelho-ir/. Acesso em: 31 de julho de 2023.

[5] RS COMPONENTS LTD. Magnatec L298N, Brushed Motor Driver IC, 46 V 4A 15-Pin, MULTIWATT V, 2020. Disponível em: https://uk.rs-online.com/web/p/motor-driver-ics/0636384/. Acesso em 27 de junho de 2022.

[6] ROBOCORE, Servo TowerPro MG996R Metálico. Disponível em: https://encurtador.com.br/iquxV. Acesso em: 04 de julho de 2022.

[7] ALIEXPRESS, Rodas Mecanum. Disponível em:https://encurtador.com.br/ciyB8. Acesso em: 27 de junho de 2022.

[8] POLOLU, Motor das rodas. Disponível em: Pololu. Acesso em: 27 de junho de 2022.

[9] ROBOTICS, O.ROS. 2021. Acessado em 5 jul. 2022. Disponível em: https://www.ros.org/

[10] ROBERT, J.Hands-On Introduction to Robot Operating System (ROS). 2020. Disponível em: https://trojrobert.github.io/hands-on-introdution-to-robot-operating-system(ros). Acesso em 05 de julho de 2022.

[11] Motor de Passo NEMA 17, Disponível em: Baú da Eletrônica. Acesso em: 28 de julho de 2023.

[12] Raspberry Pi camera, Disponível em: Proto-PIC. Acesso em: 26 de julho de 2023.

[13] MERCADOLIVRE, Mpu-9250, Disponível em: https://encurtador.com.br/bqEIR. Acesso em: 26 de julho de 2023.

[14] MERCADOLIVRE, Câmera web Logitech C270 HD 30FPS , Disponível em: https://encurtador.com.br/lotB4. Acesso em: 26 de julho de 2023.