15º Congresso de Inovação, Ciência e Tecnologia do IFSP - 2024

UMA PROPOSTA DE RESOLUÇÃO DA OBR RESGATE

ALVARO T. S. SOUZA[1], LUIZ E. HORTÊNCIO[2], MATHEUS R. OLIVEIRA[3], SILVIO V. JUNIOR[4], VERA L. SILVA[5]

[1] Estudante do 2º ano do Técnico em Automação Industrial Integrado ao Ensino Médio, Bolsista ICJ CNPQ, IFSP, Campus Suzano, alvaro.tadeu@aluno.ifsp.edu.b .
[2] Estudante do 2º ano do Técnico em Automação Industrial Integrado ao Ensino Médio, Bolsista ICJ CNPQ, IFSP, Campus Suzano, luiz.hortencio@aluno.ifsp.edu.br.
[3] Estudante do 2º ano do Técnico em Automação Industrial Integrado ao Ensino Médio, Bolsista ICJ CNPQ, IFSP, Campus Suzano, ramos.oliveira@aluno.ifsp.edu.br.
[4] Estudante do 1º ano do Técnico em Automação Industrial Integrado ao Ensino Médio, IFSP, Campus Suzano, s.valentini@aluno.ifsp.edu.br.
[5] Doutora em Ciências no Programa de Pós-Graduação em Engenharia Eletrônica e Computação - área de Informática pelo Instituto Tecnológico de Aeronáutica (ITA). Atua nas áreas de informática geral, tecnologia da informação, linguagens de programação, sistemas multi-agentes, robótica móvel e educacional.

Área de conhecimento (Tabela CNPq): 1.03.01.04-6 Lógicas e Semântica de Programas

RESUMO:

O artigo a seguir tem como objetivo disseminar os métodos de resolução dos desafios da Olimpíada Brasileira de Robótica (OBR), uma olimpíada que simula um cenário catastrófico onde um robô autônomo deverá atravessar um perigoso caminho para enfim resgatar vítimas de forma objetiva e aplicável, para que sejam usados como referência em diferentes projetos. O projeto é um robô de resgate seguidor de linha, confeccionado em lego e programado na Linguagem de programação Python para a modalidade busca e resgate da OBR. O projeto apresenta os múltiplos sistemas lógicos desenvolvido para a Zona de Resgate e captura de vítimas, para as curvas e suas variações e o controle a partir do algoritmo *proportional integral derivative controller* (PID), propondo maneiras inovadoras de superar os desafios propostos por esta olimpíada.

PALAVRAS-CHAVE: Ev3; Python; OBR; Zona Resgate; Lego; ev3dev.

A RESOLUTION PROPOSAL FOR THE OBR RESCUE

ABSTRACT: *The following article aims to disseminate the methods for addressing the challenges of the Olimpíada Brasileira de Robótica (OBR), a olympiad that simulates a catastrophic scenario in which an autonomous robot must traverse a perilous path to rescue victims, in an objective and applicable manner, so that they can be used as a reference in different projects. The project is a line-following rescue robot, made of Lego and programmed in the Python programming language for the OBR search and rescue category. The Project presents the multiple logical systems developed for the Rescue Zone and victim capture, for the curves and their variations and the control from the proportional integral derivative controller (PID) algorithm, proposing innovative ways to overcome the challenges proposed by this Olympiad.*

KEYWORDS: *Ev3; Python; OBR; Rescue Zone; Lego; ev3dev.*

INTRODUÇÃO

Este trabalho propõe o projeto de um robô móvel seguidor de linha, programado em Python e construído a partir dos kits lego Ev3 e NXT com desempenho desejável na resolução do percurso de obstáculos da OBR (Olímpiadas Brasileira de robótica). O artigo vem com a proposta de ajudar e disseminar lógicas e maneiras de solucionar os principais obstáculos da competição usando Python "A missão da Modalidade Prática - Robótica de Resgate caracteriza-se por realizar o resgate de vítimas de uma situação de desastre, utilizando robôs completamente autônomos e sem qualquer assistência humana" (OBR, 2024).

A OBR é composta por duas zonas, uma delas é chamada de zona de percurso onde o robô deverá seguir uma linha de variadas formas e obstáculos, como intersecções das linhas, curvas em ângulos retos, falhas na linha e paredes que obstruem o percurso. A outra área presente é denominada zona de resgate onde o robô deverá entrar, posicionar as vítimas (bolinhas de isopor pintadas de preto ou prata) em suas respectivas zonas, vermelho para a vítima morta (bolinha preta) e verde para a vítima viva (bolinha prata).

MATERIAL E MÉTODOS

Para o presente robô foram utilizados como materiais kits Lego Mindstorms EV3 e peças complementares providas dos kit Lego Mindstorms NXT. Todo o projeto teve como foco o microcontrolador EV3, utilizando sensores e atuadores compatíveis com o microcontrolador.

Para a programação foi utilizado a linguagem de programação Python, através da biblioteca pybricks2.0, uma biblioteca focada na programação de microcontroladores lego (PYBRICKS, 2020) Essa linguagem vem com uma gama de conteúdo, trazendo mais possibilidades para estratégias de desenvolvimento do projeto. Para utilizar esta biblioteca foi necessário uma alternativa ao sistema operacional padrão instalado, por isso utilizou-se o sistema operacional ev3dev. O sistema é uma distribuição Linux baseada no Debian para o EV3 (ev3dev,2024).

O projeto iniciou com a construção da arquitetura do robô móvel, utilizando as peças do Kit Lego. Fez-se necessário projetar a arquitetura de acordo com os desafios da OBR. Foram necessárias várias versões de arquitetura. Em seguida foram desenvolvidos os códigos de controle do robô, utilizando a linguagem de programação Python. O código foi documento através de explicações simples em fluxogramas, textos e desenhos.

LÓGICA PARA CURVAS DE ÂNGULOS RETOS

Na Olimpíada Brasileira de Robótica há dois tipos de curvas de ângulos retos, também conhecidas por curvas de múltiplos de noventa graus. Há aquelas anunciadas com uma fita de cor verde, estas podem ter múltiplas direções que o caminho de fita preta seguirá, porém apenas uma é a verdadeira que é sinalizada por um pequeno quadrado de cor verde adjacente à linha. Esta mostra-se um desafio para a calibração dos sensores, cuja altura não deverá atrapalhar a capacidade de passar o obstáculo "lombada". Há também as não anunciadas, nestas curvas não há bifurcações porém não há nenhum aviso prévio de sua existência. Este mostra-se um desafio pela lógica, que não deverá atrapalhar a capacidade de passar o obstáculo "*gap*".

Para realizar a primeira dessas curvas é necessário encontrar o sinalizador verde, e posteriormente checar a validez da curva. Caso não haja continuação de linha preta, virando para a direção sinalizada, a sinalização é falsa e portanto deve-se seguir em frente.

Para realizar ações posteriores dessas curvas, como há apenas dois sensores disponíveis, quando um desses detecta estar completamente sobre a linha preta, envia a instrução para o robô ir para frente. Se não há linha preta continuando, isto é, ambos sensores leem branco, deve-se virar para a direção. Caso haja ainda uma continuação, isto significa que o Robô está apenas desalinhado e deve ser realinhado.

PROPORTIONAL INTEGRAL DERIVATIVE CONTROLLER (PID)

Responsável por manter uma movimentação linear e fluida, o controle PID é extremamente necessário para que alguns movimentos como: a curva de ângulos contínuos e para passar por linhas circulares, desafios que, para se ter um bom desempenho na OBR, é essencial superá-los. Seu cálculo por ser preciso e eficaz consegue contornar o problema do robô móvel, devido o projeto contar com apenas dois sensores de cor, problema que dificulta mais ainda uma boa movimentação.

Cálculo de derivativo e proporcional: O cálculo para realização desse controle ocorre com 4 variáveis, a proporcional (KP), o derivativo (KD), integral (Ki) (que não foram usados devido às necessidades da implementação) e o erro, a diferença das leituras anteriores do sensor com as atuais, conforme Figura 1.

Diferentes leituras requisitarão diferentes valores do derivativo e proporcional. Como exemplo, tem-se os valores presentes no código do robô a seguir que foram adquiridos empiricamente:

KP = 2.1

KD = KP*4.7

KI = 0

Posteriormente os valores KP e KD, serão responsáveis por corrigir a leitura do erro.

$$K_p E_{rro} + K_i \int E_{rro} dt + K_d \frac{dE_{rro}}{dt}$$

FIGURA 1 - Fórmula do *PID*

A Figura 2 contém o gráfico que demonstra a variação dos valores das leituras durante a movimentação do robô durante uma curva, mostrando a relação entre as leituras do erro e o retorno da função de correção.

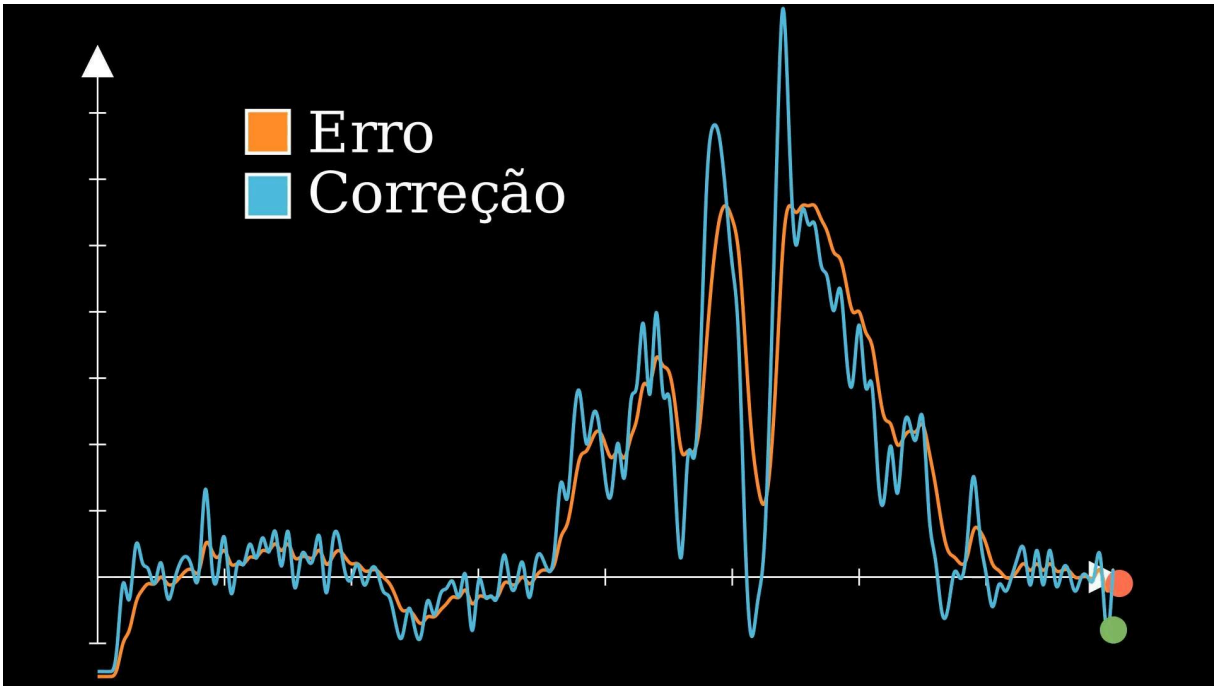

FIGURA 2 - Medições empíricas do algoritmo PID agindo sobre uma curva

LÓGICA PARA A ZONA RESGATE

Para um bom desempenho na zona de resgate foi pensado o método de varredura para que o robô fizesse a identificação da Zona, áreas de resgate e vítimas, através do sensor ultrassônico e das distâncias lidas, o robô se torna capaz de realizar os desafios. Esse método se mostrou aplicável.

A lógica de zona de resgate foi aplicada no robô da seguinte maneira: antes de entrar na sala pode ocorrer uma situação de o robô ler a cor branca no solo durante um longo período. Após rotacionar-se, o sensor ultrassônico detecta se há as paredes que formam a Zona de Resgate para que continue a lógica.. Uma série de comandos minuciosos começam a ser executados, para que o robô, se direcione ao centro da sala, e inicie um mapeamento de áreas que checa todos os cantos a fim de encontrar áreas de interesse. Posteriormente o robô começa a fazer uma varredura de identificação de

vítimas segundo a lógica “plot” onde o robô é continuamente rotacionado e as leituras de distância são tratadas. Através desta lógica pode-se encontrar os chamados pontos de “rachada”, conforme Figura 4, pontos com diferença significativa de distância com leituras anteriores e posteriores, e pela proximidade entre elas e sua direção o robô é capaz de localizar as vítimas.

Após, a computação das posições de cada vítima identificada, rotaciona-se para os seus ângulos relativos e inicia-se uma lenta captura. Após voltar a posição e orientação padrão verifica se a captura foi bem sucedida. Se sim lê a cor da vítima, a cor prata representa uma vítima viva e preta vítimas mortas, e a leva a sua zona respectiva, usando as posições encontradas pela lógica de mapeamento de áreas da zona explicada previamente. Toda essa lógica é exemplificada no fluxograma da Figura 3.

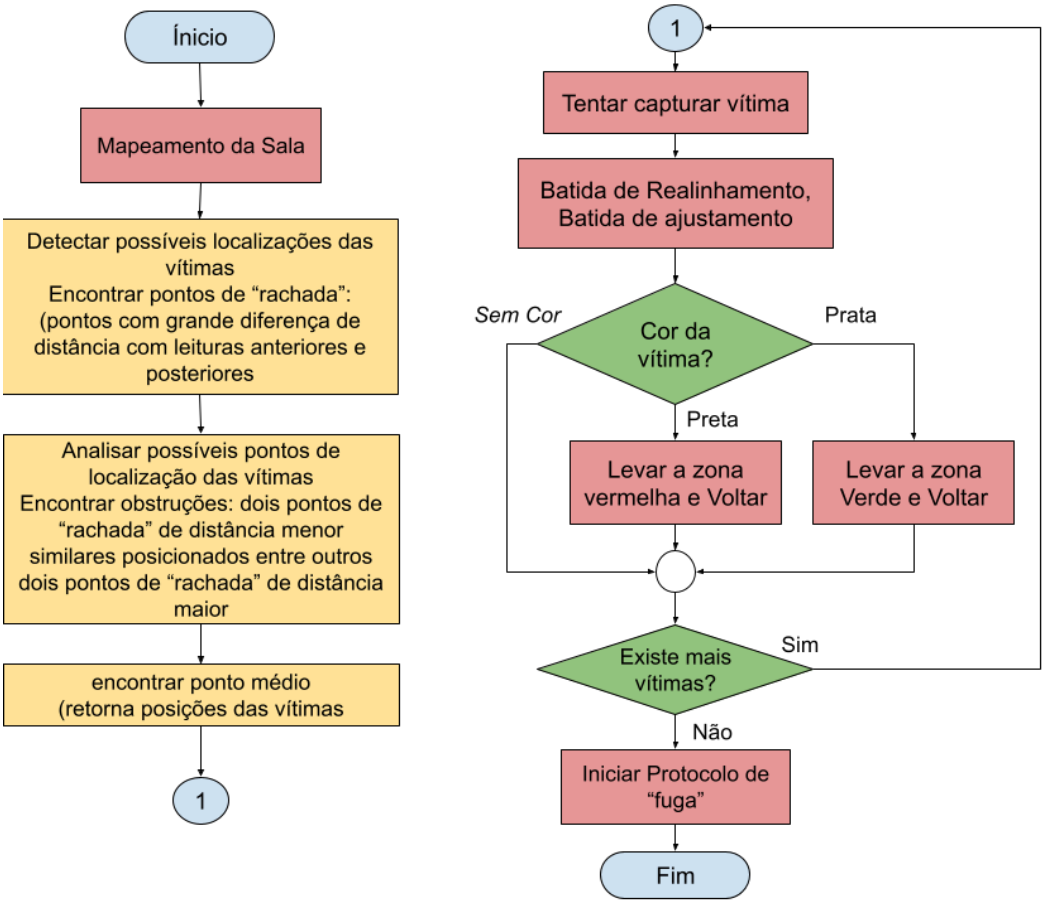

FIGURA 3 - Fluxograma da lógica da Zona de Resgate

O PLOT

Para detectar as vítimas usou-se primariamente o sensor de distância ultrassônico. Com os testes descobriu-se que as vítimas impedem que o sensor leia a parede, ou seja, obstruem o campo de visão do sensor. Pensando nisso, criou-se uma lógica que consiste em fazer uma revolução enquanto mapeia-se as distâncias lidas pelo sensor ultrassônico e as redimensiona para os pontos de grande diferença com as leituras anteriores ou posteriores. Após isso, itera-se por estes pontos a procura de uma malha que indica uma obstrução.

Na Figura 4 demonstra-se uma representação polar das leituras de distância do robô onde cada ponto representa uma leitura e os pontos laranjas representam os pontos de “rachada”.

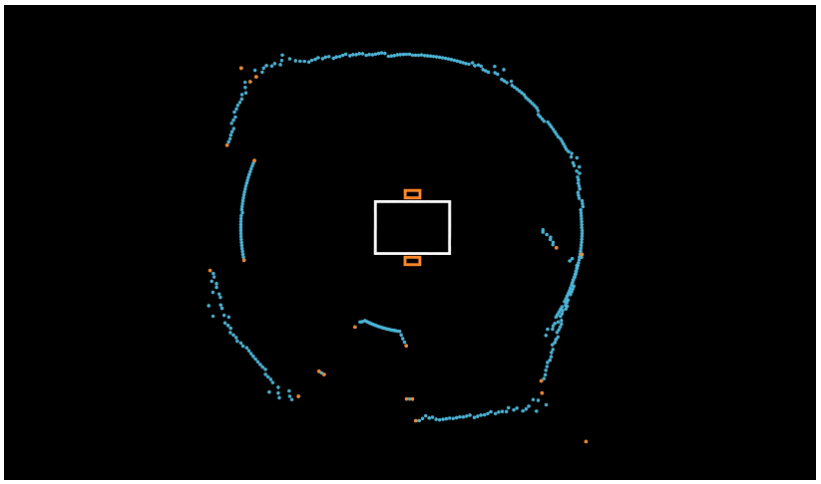

FIGURA 4 - Representação de uma varredura

RESULTADOS E DISCUSSÃO

O projeto do robô móvel apresentou um bom resultado na OBR, porém identificou-se algumas falhas. Por mais que o projeto seja capaz de realizar todos os desafios da competição, apresenta leves erros estruturais que atrapalharam as realizações dos desafios com excelência, porém tem-se a ciência de que esses pequenos contratempos foram causados por uma falta de comunicação e de testes específicos, sendo assim, as lógicas apresentadas são sim funcionais.

Com a Linguagem Python foi possível uma grande abrangência de controle do Ev3 através do Pybricks, biblioteca que permitiu que fosse desenvolvido a maioria das nossas lógicas com sucesso. Na própria competição se provou um bom projeto por conta da movimentação quase impecável por conta do controle do PID, fazendo com que a equipe alcançasse uma pontuação satisfatória. A figura 5 exibe o resultado final da arquitetura do robô.

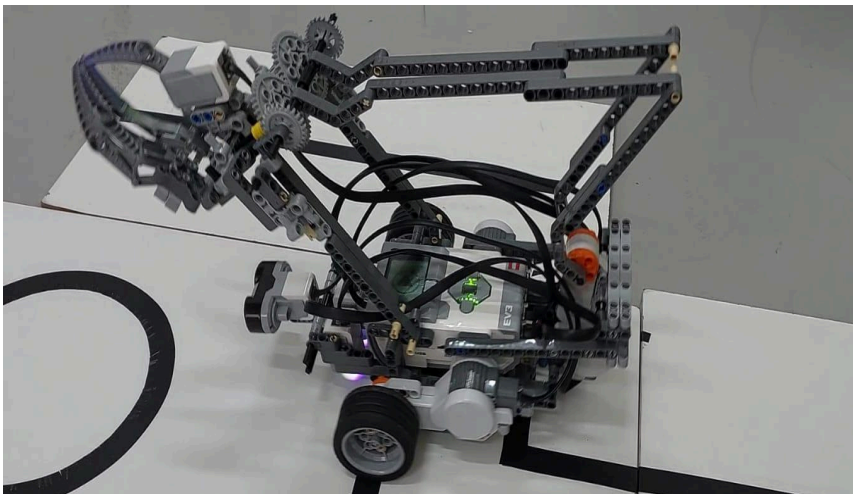

Figura 5 - Nosso projeto

Um dos principais problemas apresentado ao longo do projeto foi o estruturamento da roda boba, que por conta de seu modelo atrapalhou a passagem do carrinho sobre as lombadas. Este problema só foi detectado durante a competição da OBR. Está em andamento o desenvolvimento de um método para resolver este problema.

CONCLUSÕES

Este trabalho pode ser considerado como referência para a confecção de robôs móveis com Kit Lego e programação em linguagem de programação Python, utilizando o controle PID. Espera-se que as lógicas apresentadas possam ser entendidas e levadas como base ou referência para a programação de diferentes projetos de robótica.

O projeto ainda está sendo aprimorado e passará por diferentes inovações para continuar se desempenhando de forma significativa.

CONTRIBUIÇÕES DOS AUTORES

Luiz Eduardo Hortêncio e Matheus Ramos de Oliveira foram responsáveis pela criação e desenvolvimentos das lógicas e da programação.

Alvaro Tadeu Santos Souza e Silvio Valentini Junior foram encarregados da montagem de uma estrutura adequada para o robô móvel.

A equipe gostaria de agradecer à Vera Lucia da Silva que foi responsável pela orientação do projeto e participação na redação e revisão do trabalho.

Todos os autores participaram na redação e revisão do trabalho e aprovaram a versão submetida.

AGRADECIMENTOS

A equipe agradece a nossa professora orientadora do projeto, ao IFSP pelos equipamentos e laboratório utilizados e ao CNPQ pela bolsa de fomento à pesquisa.

A todos que participaram, direta ou indiretamente do desenvolvimento deste trabalho de pesquisa, enriquecendo o processo de aprendizado da equipe.

REFERÊNCIAS

Ev3dev. **ev3dev Home**. Disponível em: https://www.ev3dev.org/. Acesso em: 05 set. 2024.

Pybricks. **Getting started with LEGO® MINDSTORMS Education EV3 MicroPython**. Disponível em: https://pybricks.com/ev3-micropython/. Acesso em: 05 set. 2024.

CAMPOS, Augusto et al. UMA PROPOSTA DE RESOLUÇÃO DOS DESAFIOS DA ÁREA DE RESGATE DA OBR. Anais da XI Mostra Nacional de Robótica (MNR), [S. l.], p. 281-285, 15 ago. 2022. Disponível em: https://mnr.robocup.org.br/wp-content/uploads/2024/01/MNR-Anais-2021-1_compressed.pdf. Acesso em: 05 set. 2024.

OBR. Manual de Regras e Instruções – Resgate. **Documentos e Manuais – OBR – Olimpíada Brasileira de Robótica,** 2024. Disponível em: https://obr.robocup.org.br/documentos-e-manuais/. Acesso em: 05 set. 2024.

PYBRICKS. Ev3-micropython 2.0.0 documentation - Pybricks. **Getting started with LEGO® MINDSTORMS Education EV3 MicroPython — ev3-micropython 2.0.0 documentation,** 2020. Disponível em: https://pybricks.com/ev3-micropython/. Acesso em: 05 set. 2024.