PROBLEMA DO LABIRINTO: ESTUDO DE CASO DE SOLUÇÃO COM ROBÔ ZUMO POLOLU 32U4

Thayller Vilela Cintra[1]

Carlos Eduardo de França Roland[2]

Resumo

A tecnologia robótica desperta o interesse humano em todas as faixas etárias e para diferentes usos. Com a evolução da tecnologia, dispositivos microcontrolados ficaram acessíveis economica e tecnicamente, permitindo que pessoas sem profundos conhecimentos técnicos de eletrônica e linguagens de programação desenvolvam sistemas de automação e robótica. Este projeto foi concebido com os objetivos de permitir a compreensão do clássico Problema do Labirinto, de identificar os algoritmos de solução já estudados e divulgados, e projetar a implementação do Algoritmo Seguindo a Parede em um kit robótico da Pololu, o Zumo 32U4. Este kit oferece um conjunto eletromecânico sofisticado, microcontrolado com arquitetura Arduino, mas que abstrai toda essa complexidade em um conjunto pré montado, com facilidades de operação especificadas em linguagem de alto nível, que permitem o desenvolvimento e testes de rotinas de controle do movimento do robô. Tal exercício possibilitou conhecer os conceitos e métodos do controle de movimento de veículos autônomos, utilizando as funções de uma plataforma acessível para a implementação da possível solução, que permite desenvolver habilidades e competências técnicas para se construir *know-how* de desenvolvimento de veículos autônomos.

Palavras-chave: Algoritmo Seguindo a Parede. Arduino. Problema do Labirinto. Robótica autônoma. Zumo Pololu 32U4.

Abstract

Robotic technology arouses human interest in all age groups and for different uses. With the evolution of technology, microcontrollable devices became economically and technically accessible, allowing people without deep technical knowledge of electronics and programming languages to develop automation and robotics systems. This project was conceived with the objectives of understanding the classic Labyrinth Problem, identifying the solution algorithms already studied and disseminated, and designing the implementation of the Following-the-Wall Algorithm in a Pololu robotic kit, the Zumo 32U4. This kit offers a sophisticated electromechanical assembly, microcontrolled with Arduino architecture, but that abstracts all this complexity in a pre-assembled set, with operating facilities specified in high level language, that allow the development and testing of robot movement control routines. This exercise made it possible to know the concepts and methods of motion control of autonomous vehicles, using the functions of an accessible platform

[1]Graduando em Análise e Desenvolvimento de Sistemas pela Fatec Dr Thomaz Novelino – Franca/SP. Endereço eletrônico: thayllervilelacintra@yahoo.com.br
[2]Docente na Fatec Dr Thomaz Novelino – Franca/SP. Endereço eletrônico: carlos.roland@fatec.sp.gov.br

for the implementation of the possible solution, which allows the development of knowledge and technical skills to build know-how of autonomous vehicle development.

Keywords: *Arduino. Autonomous Robotics. Maze Problem. Wall Following Algorithm. Zumo Pololu 32U4.*

1 Introdução

Labirintos fazem parte da história da humanidade, sendo um dos mais conhecidos o Labirinto de Creta. Segundo a mitologia grega, ele foi construído por Dédalo por ordem do Rei Minos que recebeu como presente do deus Poseidon, um belissimo touro branco para que fosse sacrificado em sua homenagem. Como Minos não sacrificou o animal, Poseidon provocou a paixão de Pasífae, esposa de Minos, pelo touro e dessa união nasceu o Minotauro, personagem com corpo humano e cabeça de touro. O Labirinto de Creta então foi criado para manter o Minotauro preso.

Existem duas variações de labirintos: os chamados Meandros cujos caminhos são únicos e geralmente usados para meditação; e os que desafiam o viajante a encontrar a saída, chamados Maze. O fascínio por este segundo tipo motivou a busca por algotimos que pudessem descrever a solução para se encontrar a saída de qualquer labirinto. A partir do século XX este desafio ganhou relevância em função do desenvolvimento da robótica, especialmente a robótica autônoma.

O termo robô foi usado pela primeira vez na peça R.U.R (Rossum's Universal Robots) do escritor checo Karel Capek em 1921. O enredo era simples: um homem cria um robô para substituí-lo na linha de produção industrial e, em seguida, o robô se rebela e mata o homem (ZAMALLOA, 2017). Há uma referência ao primeiro robô mecânico da história como um soldado com um fole automático soprando um trumpete. Ele teria sido construído em 1810 por Friedrich Kaufman de Dresden, Alemanha, e seus movimentos eram realizados por molas de relógio (ROBOSHOP, 2008), mas este fato não pode ser verificado em outras referências. O termo robô tem origem na palavra tcheca *robota*, que significa trabalho forçado.

O presente artigo tem por objetivo apresentar os resultados da pesquisa bibliográfica exploratória realizada sobre Robótica Autônoma, para se aprofundar o conhecimento a respeito dos algoritmos que possam ser usados para resolver o Problema do Labirinto por um robô autônomo, conhecer as características

operacionais e as funcionalidades de controle do kit robótico Zumo Pololu 32U4, bem como escolher um dos algoritmos existentes para ser implementado no dispositivo para se verificar se e como o conjunto *hardware* e *software* resolvem um labirinto criado.

Assim, os resultados da pesquisa são apresentados nas seções a seguir, sendo os fundamentos teóricos sobre os elementos que contextualizam tema, questão problema e hipótese de solução, descritos na seção 2 Problema do Labirinto e Robótica Autônoma; na seção 3 são descritos os Métodos e o desenvolvimento da pesquisa; a seção 4 apresenta os Resultados e discussões, sendo seguida da seção de Considerações finais e das Referências bibliográficas utililizadas.

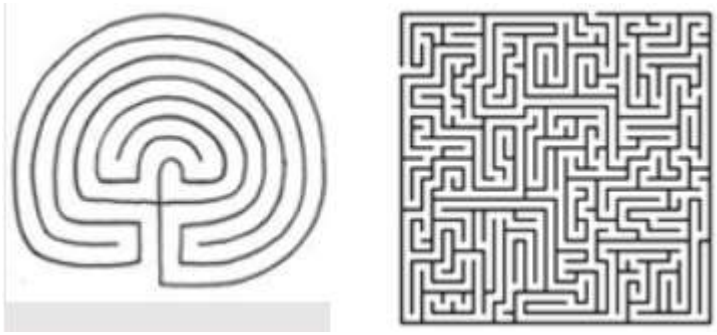
2 Problema do Labirinto e Robótica Autônoma

O labirinto é um símbolo ancestral presente nas culturas ocidentais e orientais. Os labirintos são conhecidos pela humanidade há mais de 4000 anos. Parece terem surgido e ressurgido, despertando o interesse em diversas ondas no tempo e de forma ligeiramente diferente através desse período (EDINBURGH, 2015). Segundo Algeo (2001), um labirinto é um caminho complexo em forma de circuito que leva de um ponto de partida para um destino estabelecido. Existem duas variações de labirintos: aqueles com um único e indivisível caminho, que não apresentam opções de escolha ao serem percorridos para se atingir a meta, os Meandros; e os que são formados por caminhos repetidamente divididos, que forçam o viajante a escolher uma das possíveis opções, algumas das quais levam a becos sem saída, enquanto outras fazem retornar ao mesmo ponto - os laços, de tal forma que o viajante não tem certeza de alcançar a saída (ou meta) e está constantemente frente a decisões e frustações, mas também experimenta o alívio e a surpresa de ter feito as escolhas certas.

Caminhando através do primeiro tipo, os Meandros (Figura 1a), a direção do percurso muda frequentemente, sem entretanto deixar o viajante perdido ou confuso à medida que percorre o espaço. Frequentemente este tipo de labirinto é propositalmente pensado para que se gaste mais tempo para atingir a meta, incentivando a contemplação lenta e meditativa enquanto se navega muitas voltas e reviravoltas (NBM, 2014).

Já o segundo tipo, designados como Maze (Figura 1b), possuem caminhos sem saída. Frequentemente existem enigmas que ajudam o viajante a encontrar o caminho certo aliviando frustrações, mas a ideia é deixá-lo perdido algumas vezes antes que encontre a saída. Estes labirintos, construídos em duas dimensões, oferecem a possibilidade de visualização de seu curso completo de uma vez, e os mais difíceis consomem tempo para sua solução. Enquanto Meandros são frequentemente vistos como espaços pensativos e tranquilos para reflexão silenciosa, os Mazes tendem a atrair aqueles com interesse em resolver enigmas desafiantes (NBM, 2014).

Figura 1 - Exemplos de labirintos: a) Meandro; b) Maze



(a) **(b)**

Fonte: NBM (2014, *online*)

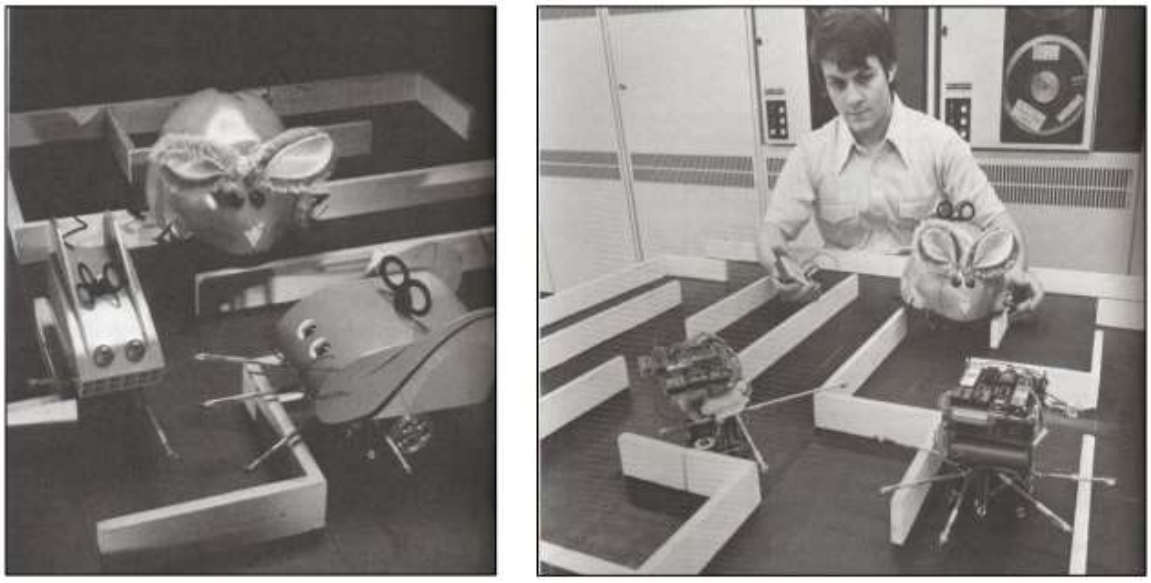
A palavra *maze* data do século XIII e deriva da palavra *maes* do Inglês Médio, denotando delírio ou desilusão. A palavra labirinto, do século XIV, deriva do Latim *labyrinthus* e do Grego *labýrinthos*, que significa uma construção com passagens intricadas (NBM, 2014). Neste trabalho, daqui em diante, as referências a labirintos sempre serão relativas aos do tipo Maze, que caracterizam um velho desafio a ser solucionado que ainda é considerado um importante campo da robótica, especialmente da robótica autônoma (RAHMAN, 2017).

As soluções do Problema do Labirinto são baseadas em Algoritmos de Tomada de Decisão. Em meados do século XX, as soluções do Problema do Labirinto alcançaram significância a partir da edição da revista IEEE Spectrum em 1972, na qual foi lançado o conceito do MicroMouse (Figura 2), que é um pequeno veículo microcontrolado com inteligência própria e com capacidade de navegar por um labirinto crítico. Em 1977 o Desafio do Labirinto do Surpreendente Micromouse (Amazing Micromouse Maze Contest) foi anunciado pela revista, e desde então este

tipo de desafio se tornou popular. Diversos tipos de robôs são desenvolvidos todos os anos para solucionar o labirinto (RAHMAN, 2017).

Em 1999, o *micromouse* desenvolvido na Universidade do Leste de Londres usou o Algoritmo Seguindo a Parede (Wall Following Algorithm) mas o robô não se moveu com inteligência suficiente para resolver um labirinto com um laço fechado. Em 2008 um estudante da Universidade de Tecnologia da Malásia projetou um *micromouse* usando o Algoritmo de Preenchimento por Inundação (Flood Fill Algorithm) mais robusto e eficiente. Este algoritmo usa a teoria dos grafos e é capaz de encontrar o menor caminho para a solução do labirinto, mas consome grande quantidade de memória, o que não é um forte dos *micromouses*.

Figura 2 - Robôs Micromouse da década de 1970



Fonte: Cyberneticzoo (2010, *online*)

O estudo realizado neste projeto usa o Algoritmo Seguindo a Parede (ASP) para implementar uma solução de labirinto usando o robô Zumo Pololu 32U4 (Figura 3), explorando seus sensores infravermelhos de linha (*line sensor array*) como robô seguidor de linha. A escolha deste algoritmo se deu em função dele consumir menos memória de programa e de dados para sua execução.

Revista EduFatec: educação, tecnologia e gestão
V.1 N.2 julho-dezembro/2018

Figura 3 - Zumo Pololu 32U4

Fonte: Pololu (2018)

O Zumo 32U4 é um versátil e completo robô controlado por um sistema ATmega32U4 compatível com Arduino. De dimensões reduzidas, aproximadamente 10 cm cada lado, é equipado com dois conjuntos micro motoredutores de metal 75:1 HP com velocidade de deslocamento e torque médios; a placa controladora gerencia os motores com dois *drivers* em Ponte-H, coletando dados de uma variedade de sensores incluindo um par de decodificadores em quadratura (*quadrature encoders*) para controle dos motores em laço-fechado (*closed-loop*); sensores inerciais (*3-axis accelerometer, gyro*, e *magnetometer*) na placa mãe, em conjunto com 5 sensores de refletância (*downward-facing reflectance*) para seguir linhas ou deteção de bordas; e sensores de proximidade (*front-and side-facing*) para deteção de obstáculos e variação. Três botões de pressão (*pushbuttons*) oferecem uma interface para entrada de dados, e um *display* de LCD, *buzzer*, e LEDs indicadores permitem que o robô forneça *feedback*. A carga de rotinas de operação do robô é feita por uma interface USB e o desenvolvimento em microcomputadores e notebooks é simplificado pela disponibilidade de bibliotecas de funções do *hardware on-board*, e o *software* adicional fornecido compatibiliza o ambiente de desenvolvimento IDE Arduino com a placa controladora (POLOLU, 2018).

Arduíno é plataforma eletrônica de código livre criada em 2005 na Itália para o desenvolvimento de projetos livres para automatizar processos e desenvolver dispositivos inteligentes que possam interagir com o meio ambiente através de

Revista EduFatec: educação, tecnologia e gestão
V.1 N.2 julho-dezembro/2018

sensores e atuadores e que tenham um baixo custo para seus desenvolvedores. O principal objetivo do Arduíno é o de oferecer um ambiente simples para que estudantes e amantes de eletrônica possam criar seus projetos de uma forma simples e descomplicada. O Arduíno consiste de uma placa controladora programável, e para diferentes interações o Arduino trabalha com placas que podem ser acopladas à controladora que são chamadas de *shields* (ARDUINO, 2018).

3 Métodos e desenvolvimento

O labirinto usado como base para os testes da implementação é o mostrado na Figura 4 que será implementado com fita isolante preta de 19 mm de largura colada sobre placas de EVA, para ser percorrido pelo robô como um veículo seguidor de linha, e não como detetor de obstáculos como os *micromouses*. Optou-se por essa configuração, em vez de construir o labirinto com paredes, para simplificação dos testes uma vez que o Zumo 32U4 dispõem de sensores de linha na sua configuração padrão. O círculo inferior à esquerda é a entrada do labirinto e o retângulo superior à direita é a saída.

O Algoritmo Seguindo a Parede é o mais comum para solucionar o labirinto, fazendo com que o robô decida a direção a tomar usando as regras mão esquerda-mão direita. Sempre que o robô encontrar uma junção ele irá ler a linha um passo a frente para decidir qual direção tomar. Este algoritmo tem restrição de solução quando o labirinto tiver uma região de laço fechado (*closed loop*) (RAHMAN, 2017).

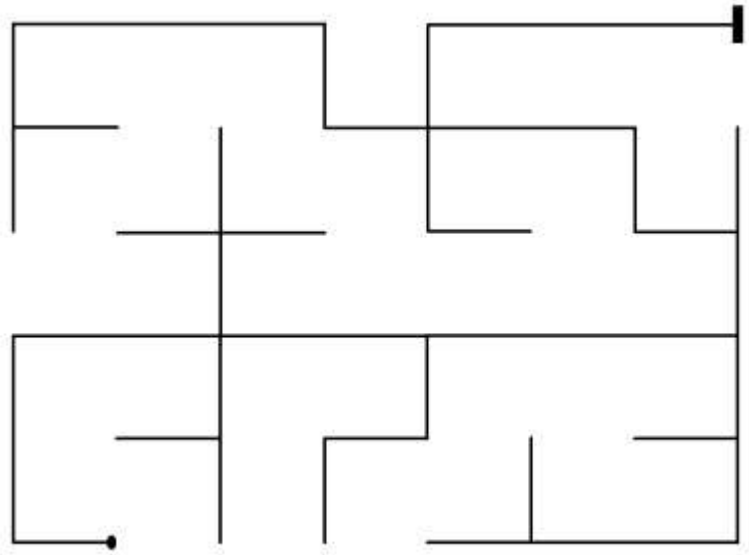
Existem basicamente duas atividades para resolver um labirinto por um robô. A primeira é fazer com que o robô percorra o labirinto até encontrar sua saída, respeitando as regras do ASP. A segunda atividade é otimizar a rota do caminho para que o robô possa sair do labirinto nos menores tempo e percurso possíveis. Foram adotadas, neste estudo, as regras da Mão Esquerda (LSRB - Left Straight Right Back) que são descritas como:

- Se puder virar à esquerda, então vire à esquerda (**L**)
- Se não puder, mas se puder seguir em frente, então siga (**S**)
- Se também não puder seguir em frente e se puder virar à direita, então vire à direita (**R**)
- Finalmente, se estiver em um ponto morto (beco sem saída), então gire 180º e volte (**B**)

Revista EduFatec: educação, tecnologia e gestão
V.1 N.2 julho-dezembro/2018

O robô tem que escolher o que fazer sempre que encontrar um entroncamento de rotas. Um entroncamento é um ponto no labirinto onde se pode mudar de direção.

Figura 4 - Labirinto projetado para os testes com o robô Zumo Pololu 32U4



Fonte: autoria própria

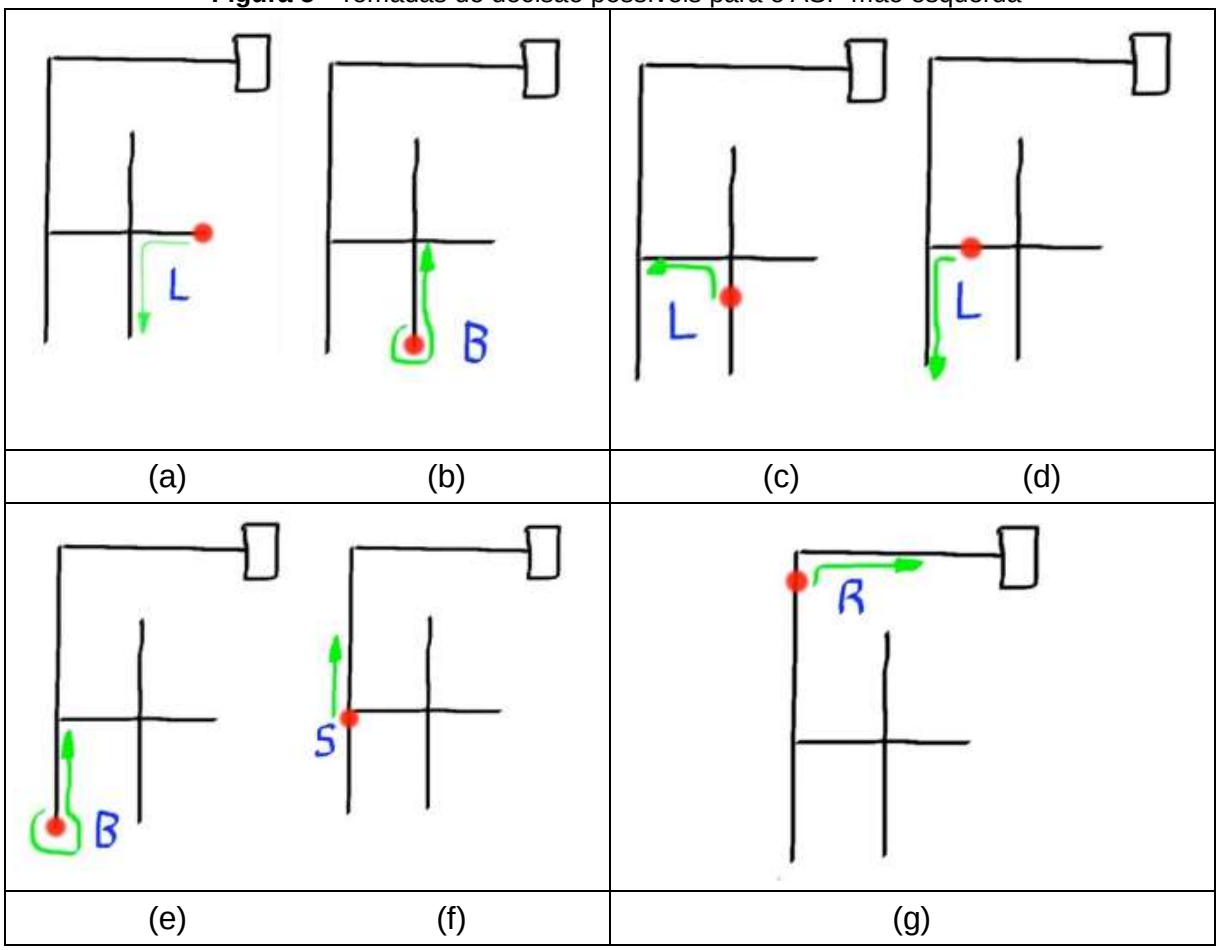
Para solucionar o labirinto é necessário armazenar todos os movimentos realizados a cada entroncamento. Considere a seguinte legenda para registro das decisões:

L = virar à esquerda (*left*)
S = seguir em frente (*straight*)
R = virar à direita (*right*)
B = voltar (*back*)

A Figura 5 mostra um exemplo simples de como se registrar uma rota, onde o robô seguidor de linha é representado pelo círculo vermelho e a saída do labirinto é representada pelo retângulo.

Revista EduFatec: educação, tecnologia e gestão
V.1 N.2 julho-dezembro/2018

Figura 5 - Tomadas de decisão possíveis para o ASP mão esquerda



Fonte: adaptado de Pandian, Karthick, e Karthikeyan (2013, p. 94)

O robô avança pela linha até encontrar o entroncamento. Como há caminho à esquerda, ele gira à esquerda e percorre o caminho até acabar a linha (ponto morto). Neste caso decide voltar. Ao chegar de volta ao entroncamento, pode rumar para esquerda e faz isso. No próximo entroncamento, pode novamente rumar para esquerda, e assim sucessivamente até encontrar a saída.

O registro do caminho percorrido pelo robô para resolver o labirinto da Figura 5 é o apresentado na Tabela 1. A rota inicial para o robô sair do labirinto é: **LBLLBSR**. Esta é uma solução, mas não é a solução ótima. A rota otimizada para a solução, visivelmente perceptível pelo desenho do labirinto, é: **SRR**.

Considerando as três primeiras figuras do exemplo (Figuras 5a, 5b e 5c), em vez do robô realizar a sequência **LBL** ele pode executar apenas **S**. Todas as vezes que ocorrer um **B** (voltar – *back*) significa que o robô escolheu um caminho errado. Esta é uma troca possível - **LBL** por **S** - mas não é a única. Na solução de labirintos pelo algoritmo da mão esquerda as possíveis trocas são mostradas na Tabela 2. Assim a sequência **LBL**LBSR pode ser otimizada para S**LBS**R da primeira troca de

Revista EduFatec: educação, tecnologia e gestão
V.1 N.2 julho-dezembro/2018

LBL por **S**, e a sequência **LBS** trocada por **R** na segunda operação, resultando SRR, que é a rota ótima.

Tabela 1 - Registro do caminho percorrido pelo robô

Figura 5a	**L**
Figura 5b	**LB**
Figura 5c	**LBL**
Figura 5d	**LBLL**
Figura 5e	**LBLLB**
Figura 5f	**LBLLBS**
Figura 5g	**LBLLBSR**

Fonte: autoria própria

Tabela 2 - Equivalência de movimentos para otimização

Ocorrência	Trocar por
LBR	**B**
LBS	**R**
RBL	**B**
SBL	**R**
SBS	**B**
LBL	**S**

Fonte: autoria própria

Para solucionar o labirinto da Figura 4 com o ASP usando as regras da Mão Esquerda a rota inicial é

RRLLBLBLBLLSLLRLLLBRLRLLLBLBSRRLLR

que deve ser otimizada substituindo as ocorrências, mostradas a seguir destacadas em negrito, sendo trocadas de acordo com a Tabela 2 em cada iteração, iniciando pela troca de **LBL** por **S**, seguida pela troca de **SBL** por **R**, e assim sucessivamente,

RRL**LBL**BLBLLSLLRLLLBRLRLLLBLBSRRLLR
RRL**SBL**BLLSLLRLLLBRLRLLLBLBSRRLLR
RRL**RBL**LSLLRLLLBRLRLLLBLBSRRLLR
RR**LBL**SLLRLLLBRLRLLLBLBSRRLLR
RRSSLLRLL**LBR**LRLLLBLBSRRLLR
RRSSLLRL**LBL**RLLLBLBSRRLLR
RRSSLLRLSRLL**LBL**BSRRLLR
RRSSLLRLSRLL**SBS**RRLLR
RRSSLLRLSRL**LBR**RLLR
RRSSLLRLSR**LBR**LLR
RRSSLLRLS**RBL**LR
RRSSLLRL**SBL**R

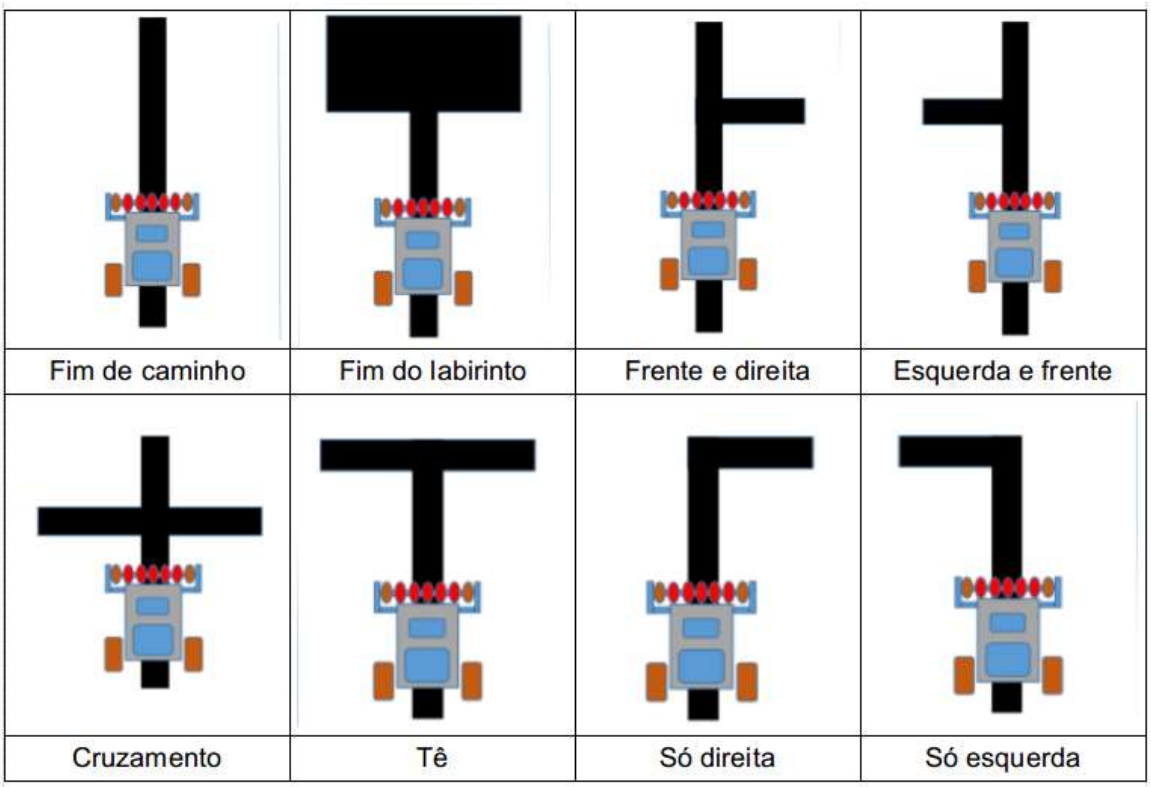
resultando a rota ótima como

RRSSLLRLRR

Revista EduFatec: educação, tecnologia e gestão
V.1 N.2 julho-dezembro/2018

Para implementar a rotina de movimentação do robô seguidor de linha para resolver o Problema do Labirinto, é necessário se identificar as interseções, definindo a opção de movimento com base nas regras LSRB descritas. Para solucionar um labirinto 2D ocorrem 8 tipos de interseções como as apresentadas na Figura 6.

Figura 6 - Possibilidades de interseções no labirinto 2G



Fonte: MJRoBot (2016)

As decisões que o robô pode assumir em cada uma destas condições são:

- Em um Fim de caminho - Volte (**B**)
- No Fim do Labirinto - **Pare**
- Em um Frente e direita - Siga (**S**)
- Em um Esquerda e frente - Esquerda (**L**)
- Em um Cruzamento - Esquerda (**L**)
- Em um Tê - Esquerda (**L**)
- Em um Só esquerda - Esquerda (**L**)
- Em um Só direita - Direita (**R**)

Revista EduFatec: educação, tecnologia e gestão
V.1 N.2 julho-dezembro/2018

Os 5 sensores de refletância, usados para controlar os movimentos do robô seguidor de linha são os destacados na Figura 7.

Os sensores da matriz geram o sinal elétrico ALTO (valor 1) quando sobre uma área escura e BAIXO (valor 0) quando sobre uma área clara, e são montados na matriz de tal forma que os espaços entre 2 dos sensores centrais possam cobrir a largura total da linha preta simultaneamente, produzindo o valor 1 em ambos.

Os possíveis valores lidos pela função da biblioteca de acesso à matriz de sensores estão representados na Tabela 3 com a descrição dos seus estados. Na matriz da Figura 7 os sensores (destacados em vermelho) são referenciados por DN1, DN2, DN3, DN4, e DN5 (da direita para a esquerda na imagem), ficando posicionados da esquerda para a direita do robô quando este é visto de cima sobre o labirinto.

Figura 7 - Matriz de Sensores Frontais do Zumo 32U4

Fonte: Pololu (2018b)

Para corrigir eventuais desalinhamentos do robô quando ele percorre a linha, pode-se usar uma variável ERRO que define o quanto o robô está fora de alinhamento. Tais valores, apresentados na Tabela 3, permitem aumentar ou diminuir a velocidade de cada motor do robô, corrigindo sua rota.

O robô deve realizar deslocamentos de 15 mm (bloco padrão) à frente e, a cada execução, deve-se ler o estado dos sensores para corrigir o alinhamento do robô e identificar interseções na pista do labirinto. Para identificar as interseções, os valores dos sensores são os mostrados na Tabela 4.

Revista EduFatec: educação, tecnologia e gestão
V.1 N.2 julho-dezembro/2018

Tabela 3 - Possíveis valores lidos dos sensores da matriz

DN1	DN2	DN3	DN4	DN5	Descrição	Erro
0	0	0	0	1	Robô à esquerda da linha	3
0	0	0	1	0	Robô sobre a linha, mas não alinhado	2
0	0	1	1	0	Robô alinhado sobre a linha, deslocado à esquerda	1
0	0	1	0	0	Robô perfeitamente alinhado	0
0	1	1	0	0	Robô alinhado sobre a linha, deslocado à direita	-1
0	1	0	0	0	Robô sobre a linha, mas não alinhado	-2
1	0	0	0	0	Robô à direita da linha	-3

Fonte: adaptado de MJRoBot (2016, *online*)

Tabela 4 - Possíveis valores lidos dos sensores da matriz, em interseções

DN1	DN2	DN3	DN4	DN5	Descrição (como possibilidades da Figura 6)
0	0	0	0	0	Fim de Caminho (beco sem saída)
1	1	1	1	1	Caminho à Esquerda e à Direita (Tê, Cruz, ou Fim do Labirinto)
0	0	1	0	0	Caminho à Frente
0	0	1	1	1	Caminho à Direita
1	1	1	0	0	Caminho à Esquerda

Fonte: adaptado de MJRoBot (2016, *online*)

Para o *software* decidir qual o próximo movimento do robô, quando encontrada uma interseção, é necessário avançar mais um bloco de deslocamento padrão e reler o estado dos sensores. As possibilidades de ação são:

1. Se Fim de Caminho:
   1.1. Volte (**B**)
2. Se Caminho à Esquerda e à Direita:
   2.1. Avance outro bloco padrão
   2.2. Se Caminho à Frente:
      2.2.1. É uma Cruz => vire à esquerda (**L**)
   2.3. Se Fim de Caminho:
      2.3.1. É um Tê => vire à esquerda (**L**)
   2.4. Se outro Caminho à Esquerda e à Direita:
      2.4.1. É Fim do Labirinto => Pare

Revista EduFatec: educação, tecnologia e gestão
V.1 N.2 julho-dezembro/2018

3. Se Caminho só à Esquerda:
   3.1. Avance outro bloco padrão
   3.2. Vire à esquerda (**L**) // não é necessário testar o estado dos sensores
4. Se Caminho à Direita:
   4.1. Avance outro bloco padrão
   4.2. Se Caminho à Frente:
      4.2.1. É Frente e Direita => seguir em frente (**S**)
   4.3. Se Fim de Caminho:
      4.3.1. É Só Direita => vire à direita (**R**)

A cada movimento decidido, a rotina deve salvar a decisão (estados mostrados em destaque e entre parênteses) em uma variável para, ao final do primeiro percurso, otimizar a rota para no percurso seguinte resolver o labirinto no menor caminho possível.

4 Resultados e discussão

O estudo bibliográfico exploratório realizado possibilitou conhecer com mais profundidade e detalhes o Problema do Labirinto, e alguns dos algoritmos de solução mais populares mostrados nas referências utilizadas. Foram também identificadas as características e funcionalidades do *hardware* e do *software* do robô Zumo Pololu 32U4, bem como escolhido um dos algoritmos para implementação e testes. A escolha foi o Algoritmo Seguindo a Parede com as regras da Mão Esquerda, o pseudocódigo foi definido e é mostrado nas Figuras 8 a 12. Foi utilizada a ferramenta *online* Portugol Webstudio[3] e partes são apresentadas.

A função de inicialização (Figura 8) define e inicializa as variáveis da rotina, calibra os sensores, e executa o laço de busca da solução do labirinto realizando a leitura dos sensores para decidir os movimentos. Ao chegar ao fim do labirinto, a rotina otimiza a rota percorrida.

A função de calibração dos sensores do robô é mostrada na Figura 9, e a função de leitura de linha é mostrada na Figura 10. A função decisão (Figura 11) é a que registra as decisões tomadas pelo robô a cada interseção encontrada, e parte

[3] Disponível em: <https://portugol-webstudio.now.sh/ide>

Revista EduFatec: educação, tecnologia e gestão
V.1 N.2 julho-dezembro/2018

da função refinaRota é mostrada na Figura 12. Ao final da execução deste trecho do algoritmo, a rota otimizada foi encontrada.

Figura 8 - Início do pseucódigo

```
programa {

    inclua biblioteca Texto --> tx

    funcao inicio() {


        logico labirinto = verdadeiro
        cadeia caminho = ""
        cadeia novo_caminho = ""
        inteiro linha

        inteiro sensores[]

        sensores = calibrar()

        enquanto(labirinto) {
           se sensores[0] == 0) { escreva = "fim da execução" } senao { caminho = ler_linha(linha_testar) }
        }

        novo_caminho = refinaRota(caminho)
    }
```

Fonte: autoria própria

Figura 9 - Função calibrar()

```
    funcao calibrar() {

        /*
        BE - girar a esquerda
        BD - girar a direita
        */
         inteiro sensores[3]
         inteiro tempo_limite = 0
         logico  bateria = verdadeiro // bateria carregada = verdadeiro ou bateria descarregada = falso

         cadeia caminho = ""

         se (bateria) {

             tempo_limite = 2000

             caminho += "BE"
             caminho += "BD"

             sensores = {1000,1000,1000}

         } senao {

             sensores = {0,0,0}
         }

         retorne sensores
     }
```

Fonte: autoria própria

A partir deste ponto pretende-se continuar a pequisa, como desenvolvimento futuro, com a construção do labirinto usando placas de EVA e fita isolante preta fosca de 19 mm; converter o pseudocódigo para um *sketch* a ser implementado na IDE Arduino, com uso das bibliotecas disponibilizadas pela Pololu para o modelo Zumo 32U4; e realizar mais testes com o robô, configurando diferentes pontos de partida no labirinto construído, para analisar as soluções iniciais de caminho de cada

Revista EduFatec: educação, tecnologia e gestão
V.1 N.2 julho-dezembro/2018

percurso e as consequentes otimizações de rota para o menor caminho possível de cada ponto de partida.

Figura 10 - Função ler_linha()

```
129     funcao ler_linha(linha_testar) {
130
131         cadeia caminho = ""
132         logico andou = verdadeiro
133
134         enquanto(andou == verdadeiro) {
135
136             se (linha_testar < 1000) {
137                 andou = falso
138                 caminho = decisao(falso,verdadeiro,falso)
139             } senao se (linha_stetar < 3000) {
140                 caminho = decisao(falso,falso,verdadeiro)
141             } senao se (linha_stetar < 4000) {
142                 caminho = decisao(verdadeiro,falso,falso)
143             } senao {
144                 caminho = decisao(falso,falso,falso)
145             }
146         }
147         retorne caminho
148     }
```

Fonte: autoria própria

Figura 11 - Função decisão()

```
77      funcao decisao(L,R,S) {
78
79          /*
80              L = virar à esquerda
81              S = seguir em frente
82              R = virar à direita
83              B = voltar
84          */
85
86          cadeia caminho = ""
87
88          se (L){
89              caminho += "L"
90          }
91          senao se (S == verdadeiro){
92              caminho += "S"
93          }
94          senao se (R == verdadeiro){
95              caminho += "R"
96          }
97          senao{
98              caminho += "B"
99          }
100         retorne caminho
101     }
```

Fonte: autoria própria

Revista EduFatec: educação, tecnologia e gestão
V.1 N.2 julho-dezembro/2018

Figura 12 - Parte da função refinaRota()

```
funcao refinaRota(caminho) {
    inteiro posicaoLBR = -1
    inteiro posicaoLBS = -1
    inteiro posicaoRBL = -1
    inteiro posicaoSBL = -1
    inteiro posicaoSBS = -1
    inteiro posicaoLBL = -1

    faca
    {
        posicaoLBR = tx.posicao_texto("LBR", caminho, posicaoLBR + 1)
        se (posicaoLBR >= 0)
        {
            caminho = tx.substituir(caminho, "LBR", "B")
        }

        posicaoLBS = tx.posicao_texto("LBS", caminho, posicaoLBS + 1)
        se (posicaoLBS >= 0)
        {
            caminho = tx.substituir(caminho, "LBS", "R")
        }

        posicaoRBL = tx.posicao_texto("RBL", caminho, posicaoRBL + 1)
        se (posicaoRBL >= 0)
        {
            caminho = tx.substituir(caminho, "RBL", "B")
        }

```

Fonte: autoria própria

Considerações finais

A tecnologia robótica desperta o interesse humano em todas as faixas etárias e para diferentes usos. Com a evolução exponencial da eletrônica nos séculos XX e XXI, dispositivos microcontrolados ficaram acessíveis economica e tecnicamente, motivando o movimento Faça Você Mesmo (DIY do termo em inglês Do It Yourself) no qual pessoas sem profundos conhecimentos técnicos de eletrônica e linguagens de programação desenvolvam sistemas de automação e robótica.

O robô Zumo Pololu 32U4 é um exemplo desses dispositivos, que contém um conjunto eletromecânico sofisticado controlado por um sistema eletrônico microcontrolado de difícil construção, mas que abstrai toda essa complexidade em um kit já montado, com facilidades de operação especificadas em linguagem de alto nível, que permitem o desenvolvimento e testes de elaboradas rotinas de controle do movimento do robô.

Um dos desafios do início do século XXI é o desenvolvimento e o lançamento em escala comercial de veículos autônomos. As soluções de *hardware* e *software* para a operação desses veículos é significativamente complexa, envolvendo conhecimentos multidisciplinares, e o exercício do desenvolvimento de uma solução

para o Problema do Labirinto é um caminho introdutório desafiador para se alcançar as habilidades e competências técnicas para se construir *know-how* para a produção destes veículos.

O projeto apresentado neste artigo foi concebido com a perspectiva de iniciar-se nos conceitos e métodos do controle de movimento autônomo em veículos, utilizando uma plataforma acessível para a implementação de possíveis soluções.

Espera-se com a continuidade da pesquisa em projetos futuros, implementar o código de controle do robô para resolver o Problema do Labirinto com o ASP, e aprimorar a inteligência do Zumo 32U4 para solucionar labirintos de diferentes configurações e com mais opções de algoritmos.

Neste contexto, o projeto atingiu seus objetivos permitindo a construção de conhecimento e capacitação técnica no âmbito da Robótica Autônoma.

Referências

ALGEO, J. The Laburinth: A Brief Introduction to its History, Meaning and Use. 2001. Disponível em: <https://www.theosophical.org/publications/1276>. Acesso em: 2.nov.2018.

ARDUINO. What is Arduino?. sd. Disponível em: <https://www.arduino.cc/en/Guide/Introduction>. Acesso em: 7.nov.2018.

CYBERNETICZOO. Amazing Micromouse Maze Contest. 2010. Disponível em: <http://cyberneticzoo.com/tag/amazing-micromouse-maze-contest/>. Acesso em: 3.nov.2018.

EDINBURGH. Origins of Labyrinths. 2015. Disponível em: <https://www.ed.ac.uk/labyrinth/historical/origins>. Acesso em 2.nov.2018.

MJROBOT. Robô explorador de labirintos, utilizando Inteligência Artificial com Arduino. 2016. Disponível em: <https://mjrobot.org/2016/04/28/robo-explorador-de-labirintos-utilizando-inteligencia-artificial-com-arduino/>. Acesso em: 4.nov.2018.

NBM. A Brief History od Maze. 2014. Disponível em: <https://www.nbm.org/brief-history-mazes/>. Acesso em: 2.nov.2018.

PANDIAN, J. A.; KARTHICK, R.; KARTHIKEYAN, B. Maze Solving Robot Using Freeduino ans LSRB Algorithm. 2013. International Journal of Modern Engineering Research (IJMER). National Conference on Architecture, Software System and Green Computing (NCASG). Disponível em: <http://www.ijmer.com/pages/NCASG.html>. Acesso em: 18.out.2018.

POLOLU. Zumo 32U4 Robot. sd. Disponível em: <https://www.pololu.com/category/170/zumo-32u4-robot>. Acesso em: 3.nov.2018.

_____. Zumo 32U4 Front Sensor Array. sd. Disponível em: <https://www.pololu.com/product/3122>. Acesso em: 5.nov.2018b.

RAHMAN, M. Autonomous Maze Solving Robot. Trabalho de Conclusão do curso de Engenharia Eletrônica e Telecomunicações. Bangladesh: University Of Liberal Arts, 2017.

ROBOSHOP. History of Robotics: Timeline. 2008. Disponível em: <https://www.robotshop.com/media/files/PDF/timeline.pdf>. Acesso em: 6.nov.2018.

ZAMALLOA, I.; *et al.* Dissecting Robotics - historical overview and future perspectives. 2017. Disponivel em: <https://arxiv.org/pdf/1704.08617.pdf>. Acesso em: 6.nov.2018.