Desenvolvimento de um Robô Seguidor de Linha Utilizando Arduino

Hilquias R. Oliveira[1], Evellyn Letícia Sales Telles Vaz[1], Flávio Pereira da Silva[1] e Daniel dos Anjos Costa[1]

[1]Instituto Federal da Bahia (IFBA) – Santo Amaro – BA – Brasil

hilquiasro10@gmail.com,evellynleticiasalestellesvaz@gmail.com,
{flavio.pereira,daniel.anjos}@ifba.edu.br

Abstract. *Line-following robots are widely used in education, robotics competitions, and industrial applications as the basis for autonomous navigation systems. This paper presents the development of a line-following robot using the Arduino Uno platform, TCRT5000 sensors, and an L298N Mini H-bridge. The project aimed to create a robot capable of identifying and following predetermined trajectories, using a control algorithm based on discrete conditional logic. The implementation considers aspects such as sensor selection, motor control, and algorithm adjustments to improve navigation accuracy.*

Resumo. *Robôs seguidores de linha são amplamente utilizados em educação, competições de robótica e aplicações industriais como base para sistemas de navegação autônomos. Este artigo apresenta o desenvolvimento de um robô seguidor de linha usando a plataforma Arduino Uno, sensores TCRT5000 e uma Mini ponte H L298N. O projeto teve como objetivo criar um robô capaz de identificar e seguir trajetórias pré-determinadas, usando um algoritmo de controle baseado em lógica condicional discreta. A implementação considera aspectos como seleção de sensores, controle de motor e ajustes de algoritmo para melhorar a precisão da navegação.*

1. Introdução

A automação e a robótica têm desempenhado um papel fundamental no desenvolvimento de sistemas autônomos, com aplicações que vão desde linhas de produção industriais até veículos autônomos [Fresnillo 2023]. Dentro desse contexto, os robôs seguidores de linha representam uma abordagem essencial para estudos de navegação autônoma, servindo como base para o aprendizado de controle preciso de motores, processamento em tempo real de dados sensoriais e desenvolvimento de algoritmos embarcados eficientes.

Um robô seguidor de linha é um sistema capaz de detectar e seguir um percurso definido por uma linha de alto contraste em relação ao fundo. Para isso, sensores infravermelhos identificam variações na superfície, enquanto um controlador processa essas informações e ajusta a direção e velocidade do robô. O uso da plataforma Arduino Uno nesse tipo de projeto é amplamente difundido devido à sua acessibilidade, facilidade de programação e grande disponibilidade de bibliotecas.

O desenvolvimento de um robô seguidor de linha, embora baseado em uma tecnologia já consolidada, possui valor significativo nas áreas da ciência e da educação como projeto introdutório de eletromecânica, sistemas embarcados, engenharia de controle, etc., com potencial evolutivo de ter suas bases convertidas em projetos mais

complexos como a produção de veículos autônomos ou automatizados. Serve também como plataforma experimental para implementação de algoritmos de controle, otimização de navegação e aplicações em ambientes automatizados. Além disso, sua simplicidade estrutural permite a democratização do ensino de robótica e programação, sendo ideal para contextos educacionais e ambientes de baixo custo. Ao integrar conceitos modernos, como controle adaptativo ou inteligência artificial embarcada, o projeto se posiciona como um laboratório de inovação acessível e replicável. Projetos semelhantes já demonstraram o uso do Arduino Uno como uma ferramenta eficaz para o ensino de eletrônica, programação e sistemas robóticos em contextos de ensino médio e universitário [Perez et al., 2013] sendo eficaz no desenvolvimento de habilidades de pensamento computacional, resolução de problemas e competências digitais em estudantes do ensino fundamental [Condori e Pozo 2023].

Assim, este trabalho apresenta o desenvolvimento de um robô seguidor de linha utilizando a plataforma Arduino Uno, sensores TCRT5000 e uma ponte H L298N controladora de motores CC. O objetivo é projetar um sistema eficiente capaz de seguir trajetórias pré-determinadas com precisão. Para isso, um algoritmo de controle baseado em lógica condicional discreta foi implementado, permitindo a interpretação das leituras dos sensores e o ajuste dos motores em tempo real.

2. Fundamentação Teórica

O desenvolvimento de um robô seguidor de linha envolve a integração de conceitos de eletrônica, automação e controle de sistemas embarcados. Esse tipo de robô é amplamente utilizado em ambientes educacionais e industriais como uma introdução à robótica móvel e à navegação autônoma [Reda 2024] e tem se mostrado especialmente útil para promover o aprendizado ativo e o desenvolvimento de habilidades como pensamento crítico, criatividade e resolução de problemas [Eduka Bytes, 2024]. Seu princípio de funcionamento baseia-se na detecção de uma linha de alto contraste em relação ao fundo, utilizando sensores ópticos para captar variações na superfície e um algoritmo de controle para ajustar a movimentação do robô em tempo real. Nesta seção estudaremos as principais tecnologias utilizadas no projeto.

A primeira tecnologia é o controlador do sistema robótico, que constitui o núcleo do sistema embarcado. Ele é responsável pelo processamento das leituras dos sensores e pelo controle dos atuadores. Entre as diversas opções disponíveis no mercado, optamos pelo Arduino Uno, equipado com o microcontrolador ATmega328P. Essa escolha se deu pela sua acessibilidade, facilidade de programação e pela ampla comunidade de suporte. A plataforma conta com 14 pinos digitais e 6 entradas analógicas, permitindo a comunicação com sensores e motores. Além disso, sua compatibilidade com bibliotecas pré-existentes facilita a implementação de algoritmos para navegação autônoma [Arduino 2025].

A segunda tecnologia abordada nesta seção é o sensor TCRT5000, classificado como um sensor óptico reflexivo devido à sua capacidade de detectar contrastes entre superfícies. Ele é composto por dois componentes principais: um emissor de luz infravermelha e um fototransistor. Quando a luz infravermelha emitida incide sobre uma superfície branca, a maior parte da luz é refletida e captada pelo fototransistor, resultando em uma leitura de alta intensidade. Em contraste, uma superfície preta absorve a luz infravermelha, gerando um sinal de menor intensidade [Vidal 2017].

A terceira tecnologia apresentada nesta seção é a Mini ponte H L298N. Na robótica de dispositivos móveis, a grande maioria dos controladores não conseguem garantir a potência necessária para tirar da inércia ou manter a movimentação de motores. Diante disso, tecnologias como a Mini ponte H L298N têm a capacidade de amplificar a corrente de sinais fracos vindo de controladores para atuar no controle de motores de corrente contínua (CC) [Straub 2019]. Essa tecnologia também possibilita a inversão da polaridade aplicada aos motores, permitindo que o robô avance, recue e realize curvas conforme o código de controle embarcado no Arduino [Straub 2019].

Para controlar a velocidade de motores é comum a utilização da técnica chamada Modulação por Largura de Pulso (*Pulse Width Modulation* - PWM). Essa técnica consiste em variar a largura dos pulsos de um sinal de controle digital para ajustar a velocidade e o torque de um motor [Boylestad 2013]. A largura dos pulsos determina a quantidade média de energia fornecida ao motor, o que, por sua vez, influencia a velocidade e o torque. Quanto maior a largura do pulso (ou seja, maior a proporção do tempo em que a tensão é aplicada), maior será a energia média fornecida ao motor, resultando em maior velocidade. É importante destacar que os pulsos são gerados com a amplitude máxima suportada pelo motor, o que impede a redução do torque. Por outro lado, um pulso de menor largura e com tensão máxima resultará em uma rotação mais lenta do motor, mas com torque máximo. Assim, mesmo com o eixo do motor girando lentamente, ele manterá sua força de torque no valor máximo.

3. Desenvolvimento

O desenvolvimento do robô seguidor de linha foi realizado em quatro etapas principais: seleção e preparação dos componentes, montagem do hardware, implementação do software embarcado e testes e calibração. Cada uma dessas fases foi essencial para garantir o funcionamento eficiente do sistema, desde a escolha dos elementos eletrônicos até os ajustes finais no algoritmo de controle.

3.1. Seleção e Preparação dos Componentes

A escolha dos componentes foi baseada na compatibilidade com a plataforma Arduino Uno, garantindo facilidade de integração e programação. Todos os componentes foram descritos na seção 2.0 deste trabalho. O microcontrolador foi responsável pelo processamento das leituras dos sensores e pelo controle dos motores. Para a detecção da linha, foram utilizados cinco sensores ópticos TCRT5000, posicionados na base frontal do robô. Esses sensores foram escolhidos devido à sua precisão na detecção de superfícies contrastantes, permitindo uma leitura confiável da trajetória.

Para a movimentação do robô, foram utilizados dois motores CC de 6V com caixa de redução, proporcionando torque adequado para a locomoção. Como o Arduino não possui capacidade para alimentar diretamente os motores, foi empregada Mini Ponte H L298N, um componente eletrônico que amplifica a corrente do sinal de controle. Esse circuito permite o controle independente da velocidade e da direção de rotação dos motores, garantindo um desempenho eficiente e preciso. A utilização do *shield* ponte H L298N para o controle dos dois motores CC é uma escolha viável pois evita os problemas com a soldagem de um circuito eletrônico, pois já possui terminais de conexão [Scherer et al., 2020].

Como fonte de alimentação, foram utilizadas duas baterias de lítio 18350

conectadas em série, fornecendo até 8,4V ao sistema. Essa configuração garantiu uma boa autonomia sem comprometer a eficiência energética do robô.

Antes da montagem, os sensores foram submetidos a uma calibração manual inicial para definir os valores ideais de limiar entre superfícies claras e escuras,. Esse processo foi essencial para minimizar erros de detecção causados por variações na iluminação ambiente.

3.2. Montagem Física

A estrutura do robô foi projetada para garantir estabilidade e distribuir adequadamente os componentes eletrônicos e mecânicos [Sgai et al., 2019]. A Figura 1 exibe o circuito eletrônico do sistema detalhando a interconexão entre os principais componentes. As Figuras 2a e 2b apresentam a montagem final em uma visão superior e lateral, respectivamente. A base foi confeccionada em acrílico, proporcionando leveza e resistência. Os motores CC foram fixados na parte traseira do robô, conectados diretamente às rodas de tração, no modelo específico de suporte para Mini Motor CC N20 e dimensões 11X25mm. Na parte frontal, foi instalada uma roda boba esférica, permitindo maior mobilidade e facilitando as correções de trajeto.

O conjunto de cinco sensores TCRT5000 foram posicionados na parte inferior frontal do robô, alinhados de maneira a abranger toda a largura da linha a ser seguida. Estes sensores infravermelhos reflexivos, alimentados pelo Arduino, enviam seus dados de leitura (digitais ou analógicos) para pinos específicos do microcontrolador. Com base nas informações desses sensores – que distinguem a superfície escura da linha clara – o Arduino processa e toma decisões, comandando os motores através da Ponte H para seguir a linha marcada no chão e corrigir o trajeto. Esse arranjo permite que o carro opere de forma totalmente autônoma, adaptando-se em tempo real à pista.

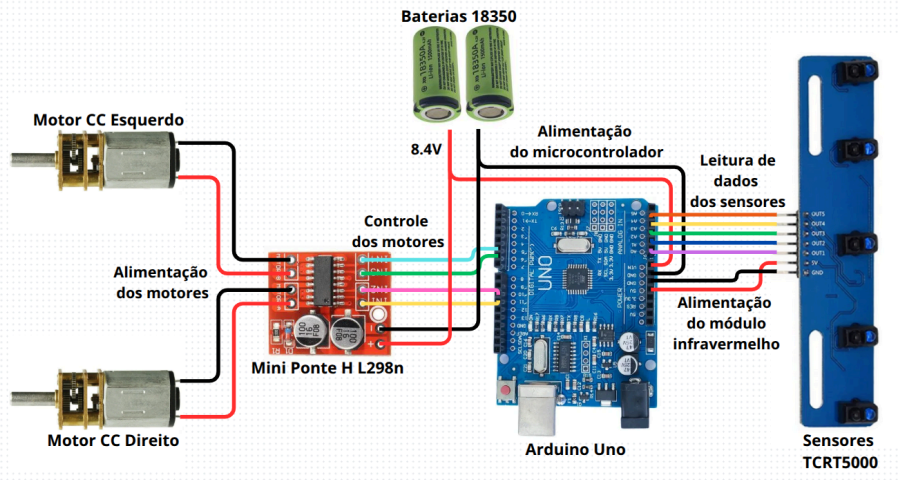
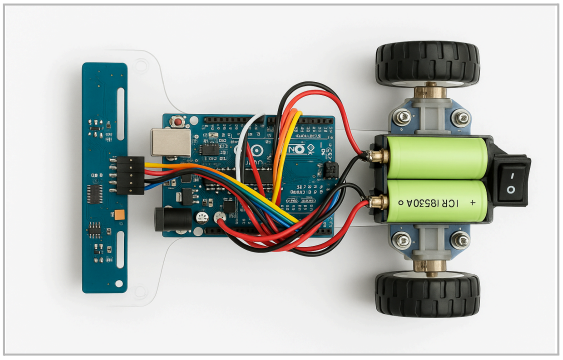


Figura 1. Esquema de montagem do circuito eletrônico do robô.

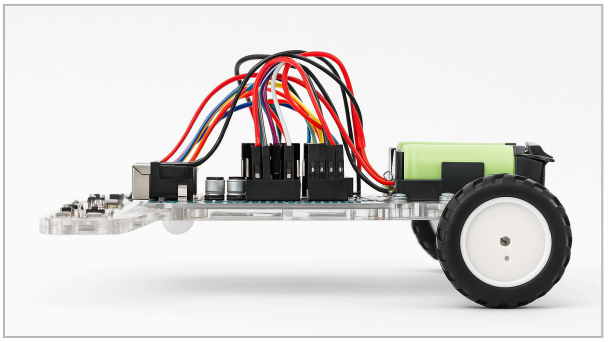
O sistema é alimentado por duas baterias 18350, conectadas em série para fornecer 8.4V. Essa energia é distribuída tanto para a placa controladora Arduino Uno (via pino VIN) quanto para o módulo Mini Ponte H L298N, que, por sua vez, recebe a tensão para acionar dois motores CC (um esquerdo e um direito) com torque adequado para a movimentação. O Arduino Uno, atuando como o controlador do robô, envia sinais de controle para o L298N, gerenciando a direção e velocidade dos motores

através de pinos digitais, enquanto os pinos GND garantem a referência comum para todo o circuito.

A ponte H L298N e o Arduino Uno foram montados na parte central do robô, permitindo uma melhor organização dos cabos, centralização do peso e reduzindo interferências elétricas. A fixação das baterias 18350 foi feita na parte superior da estrutura, garantindo fácil acesso para substituição ou recarga, e um interruptor foi adicionado ao circuito para ligar ou desligar energia do robô.



(a)



(b)

Figura 2. Robô seguidor de linha em (a) visão vertical e (b) visão lateral

3.3. Montagem Lógica

O software foi desenvolvido utilizando a IDE do Arduino, com a linguagem de programação C++. A biblioteca AFMotor foi empregada para facilitar o controle dos motores por meio do *driver* de controle L298N, que gerencia o acionamento dos motores CC do robô. O algoritmo de controle construído baseia-se em um conjunto de decisões condicionais, que são acionadas conforme as leituras dos sensores de linha e, para isso, foram utilizadas algumas variáveis importantes, apresentadas na Tabela 1, que definem a dinâmica do comportamento do robô.

Tabela 1. Variáveis usadas na construção do software embarcado.

Variável	Valor	Definição
motorSpeed	110	Define a velocidade dos motores quando o robô está seguindo a linha.
turningMotorSpeed	170	Ajusta a velocidade dos motores durante curvas bruscas.
lineThresh	200	Limite superior de leitura dos sensores para detectar a linha.
noLineThresh	20	Limite inferior de leitura dos sensores em situação de ausência de linha.

A simplicidade estrutural do algoritmo construído permite a fácil adaptação para diferentes configurações de pista, quantidades de sensores ou ajustes de velocidade, favorecendo também o aprendizado de iniciantes na programação embarcada e no controle de robôs móveis.

Na análise das variáveis, os valores de velocidade do motor (*motorSpeed* 110 e *turningMotorSpeed* 170) correspondem ao ciclo ativo do sinal PWM enviado à ponte H L298N. Como a saída PWM varia de 0 a 255, um valor de 110 equivale a aproximadamente 43% do ciclo de trabalho (*duty cycle*), enquanto 170 representa cerca de 67%. Isso significa que os motores recebem pulsos elétricos com essa proporção de tempo entre o estado de ligado e desligado, controlando a velocidade de rotação sem perder o torque. Os valores de limiar e linha (*lineThresh* 200 e *noLineThresh* 20) foram definidos com base nas leituras dos sensores reflexivos. Nos testes, foi observado que os valores para superfícies pretas ficaram entre 0 e 20, enquanto para superfícies brancas ficaram acima de 900. Para garantir uma margem de segurança, foi adotado 200 como limite para considerar a detecção da linha e 20 para ausência de linha, evitando leituras imprecisas causadas por variações na iluminação ou no material da pista. A Figura 3 ilustra a sequência de atividades executadas pelo robô com base na leitura dos sensores.

```
1  // IMPORTAR BIBLIOTECAS
2  void setup() {
3    // Configura os pinos dos sensores
4    // Configura os pinos de controle dos motores como saída
5  }
6  void moverParaFrente() {
7    // Função para mover o robô para frente (ambos os motores avançam)
8  }
9  void moverParaEsquerda() {
10   // Função de curva para esquerda (redução da velocidade motor esquerdo)
11 }
12 void moverParaDireita() {
13   // Função de curva para direita (redução da velocidade motor direito)
14 }
15 void pararMotores() {
16   // Função para parar todos os motores
17 }
```

Figura 3. Primeira parte do Pseudocódigo de controle do sistema.

No bloco *setup()* (linha 2), são definidos os pinos utilizados para entrada e saída digitais, garantindo a correta comunicação entre os sensores e motores. Em seguida, são definidas funções de movimentação: *moverParaFrente()* (linha 6) aciona ambos os motores com a mesma intensidade para que o carro avance para frente; *moverParaEsquerda()* (linha 9) e *moverParaDireita()* (linha 12) implementam as curvas, geralmente reduzindo a velocidade do motor correspondente e mantendo a velocidade mais alta do motor oposto; e *pararMotores()* (linha 15), que desativa ambos os motores, interrompendo o movimento do carro. A figura 4 apresenta o pseudocódigo parcial e simplificado de controle do carro.

O controle do carro reside na função *controleAutonomo()* (linha 19), que possui a lógica de decisão baseada nas leituras dos sensores. As pseudo condições IF e ELSE

IF (linhas 21-36) avaliam o estado dos sensores de linha (SENSOR_CENTRAL, SENSOR_ESQUERDO, SENSOS_DIREITO). Por exemplo, se apenas o SENSOR_CENTRAL detecta a linha (linha 21), o carro *moverParaFrente()*. Se o SENSOR_ESQUERDO é o único a detectar (linha 24), o carro *moverParaEsquerda()* para corrigir o trajeto, e de forma similar para o SENSOR_DIREITO (linha 27). Se TODOS_SENSORES estiverem como FALSE (linha 33), indicando que a linha foi perdida, o carro chama a função *pararMotores()*.

```
19 void controleAutonomo() {
20
21   if ( SENSOR_CENTRAL ==  TRUE e DEMAIS_SENSORES = FALSE ) {
22     moverParaFrente(); // Mover para frente
23   }
24   else if ( SENSOR_ESQUERDO ==  TRUE e DEMAIS_SENSORES = FALSE ) {
25     moverParaEsquerda(); // Mover para esquerda
26   }
27   else if ( SENSOR_DIREITO ==  TRUE e DEMAIS_SENSORES = FALSE ) {
28     moverParaDireita(); // Mover para direita
29   }
30   else if ( TODOS_SENSORES ==  TRUE ) {
31     moverParaFrente(); // Mover para frente
32   }
33   else if ( TODOS_SENSORES ==  FALSE ) {
34     pararMotores(); // Parar a movimentação
35   }
36   else {
37     moverParaFrente(); // Mover para frente
38   }
39 }
40
41 void loop() {
42   controleAutonomo(); // Chama a função principal de controle continuamente
43 }
```

Figura 4. Segunda parte do Pseudocódigo de controle do sistema.

Caso TODOS_SENSORES detectem a linha (linha 30) ou o ELSE final (linha 36) aconteça o carro segue o comportamento padrão de chamar a função *moverParaFrente()*, pois foi detectado alguma situação ambígua. Por fim, a função *loop()* (linha 41) executando repetidamente a chamada para *controleAutonomo()* (linha 42) de forma contínua. Essa execução constante garante que o carro seguidor de linha esteja sempre avaliando as leituras dos sensores e ajustando sua movimentação em tempo real para permanecer na linha, caracterizando sua operação autônoma.

A Figura 5 consiste em um diagrama de atividades que representa visualmente as etapas descritas nas Figuras 3 e 4 através do pseudocódigo. Essa representação visual facilita a compreensão da hierarquia e do fluxo de decisões que permitem ao carro seguidor de linha operar de forma autônoma e adaptativa em pistas variadas.

No primeiro nó de decisão, o sistema verifica se a linha está centralizada no sensor; em caso positivo, o robô avança em linha reta, mantendo o alinhamento. Se a linha não estiver no centro, a lógica prossegue para identificar se ela se encontra à esquerda ou à direita dos sensores. Conforme a detecção, o robô realiza ajustes suaves

em sua direção para a esquerda ou para a direita, respectivamente, garantindo a correção contínua da trajetória e o seguimento eficaz do percurso.

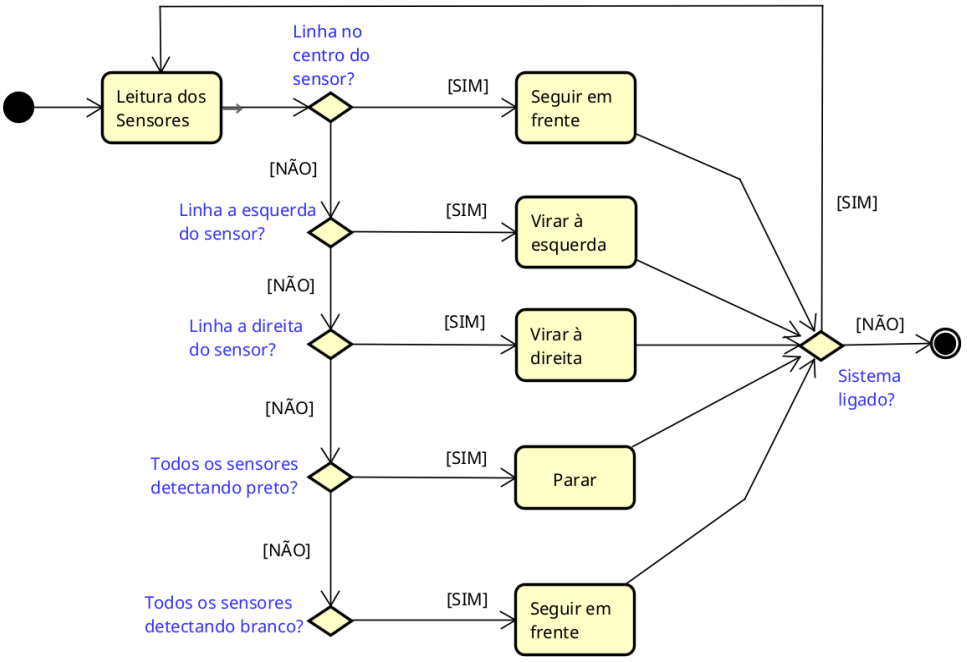


Figura 5. Diagrama de atividades do sistema.

O fluxograma também ilustra as condições para cenários mais complexos. Se todos os sensores detectam a cor preta simultaneamente, o robô interpreta a situação como uma possível interseção ou ponto de parada, acionando uma função para interromper seu movimento. Por outro lado, se todos os sensores detectam a cor branca simultaneamente, o robô assume que perdeu a linha e, em uma estratégia de recuperação, continua avançando em frente até que uma nova detecção o realinhe com o percurso.

4. Discussões e Resultados

O desempenho do robô seguidor de linha foi avaliado em diferentes trajetórias, considerando fatores como precisão na detecção da linha, tempo de resposta, estabilidade e eficiência do controle de motores. Durante os testes, foram identificados aspectos positivos e desafios que exigiram ajustes para otimizar o funcionamento do sistema, que estão listados abaixo:

- **Precisão na detecção e robustez do algoritmo** - Os sensores TCRT5000 apresentaram bom desempenho na identificação da linha, permitindo uma navegação confiável. No entanto, foi necessário um ajuste fino nos limiares de detecção, pois variações na iluminação ambiente impactavam as leituras. Após a calibração, a detecção tornou-se mais precisa, reduzindo falhas na interpretação do percurso. A lógica condicional discreta implementada demonstrou-se eficaz para trajetórias simples e moderadas. O algoritmo permitiu reações rápidas às leituras dos sensores, corrigindo pequenos desvios de forma eficiente. Entretanto, em curvas mais fechadas e bifurcações, o robô apresentou dificuldades ocasionais, exigindo refinamento nas condições de tomada de decisão para melhorar a

adaptação a diferentes percursos.

- **Tempo de Resposta e Estabilidade** - Em situações de correções rápidas de direção o robô foi ajustado para suavizar a transição das velocidades dos motores, melhorando a estabilidade do movimento. Além disso, observou-se que a velocidade do robô influenciava diretamente sua capacidade de realizar curvas. Em velocidades mais altas, ocorre um atraso na resposta, aumentando o risco de saída da trajetória. Para corrigir essa limitação, foram implementadas reduções dinâmicas de velocidade em trechos curvos, garantindo um controle mais preciso e eficiente.
- **Consumo Energético e Autonomia** - A alimentação por duas baterias 18350 mostrou-se suficiente para manter o robô operando por até duas horas contínuas, dependendo da quantidade de correções exigidas pelo percurso. O consumo energético pode ser otimizado com estratégias como a redução da velocidade em trechos retilíneos e a utilização de motores com menor dissipação de energia.
- **Trajetos** - Visando um nível satisfatório de autonomia e fluidez, foram realizados testes com diferentes trajetos, simulando pistas utilizadas em competições e demonstrações de robôs seguidores de linha anteriormente. Durante o processo, houve preocupação em representar percursos de natureza reta e prolongada, explorando a habilidade de aceleração durante linhas retas longas, com curvas acentuadas e em sequência, formando ondulações, cruzamentos bifurcados e trifurcados e seções com desvios repentinos. Assim, se obteve registros positivos acerca do domínio do robô em trajetos distintos em tipo, duração e dificuldade.

5. Conclusão

O desenvolvimento do robô seguidor de linha utilizando Arduino Uno, sensor TCRT5000 e a ponte H L298N demonstrou ser uma solução eficiente para navegação autônoma em trajetórias pré-definidas. A implementação da lógica condicional discreta permitiu correções rápidas e precisas, garantindo um bom desempenho do sistema.

O sistema foi testado em diferentes cenários de pista e iluminação para avaliar seu desempenho em condições variadas e garantir a precisão e eficiência em diversas situações de uso. Com este projeto, foi possível consolidar conhecimentos em sistemas embarcados, calibração e uso de sensores ópticos e controle de motores, além de abrir possibilidades para aperfeiçoamentos na navegação do robô em trajetórias complexas e iniciativas inovadoras utilizando o robô seguidor de linhas como estrutura base ou fundamental. Esse tipo de projeto também contribui para a democratização do conhecimento tecnológico e inclusão social, como discutido por Silva et. al (2025, p. 4), ao utilizar ferramentas de baixo custo e alto impacto educacional. O código de controle do carro se encontra em Oliveira (2025) para consulta.

A literatura e o universo científico como um todo ainda carecem de estudos sobre a escalabilidade desse tipo de tecnologia, porém há a possibilidade de integração de inteligência artificial e aprendizagem de máquina em projetos de robôs construídos a partir de arduíno, permitindo maior autonomia e a atribuição de habilidades de maior complexidade sem abandonar a proposta de leitura e seguimento de linhas [Brigido; Oliveira, 2025].

Desta forma, visando o aprimoramento do sistema, futuros desenvolvimentos podem incluir ajustes na lógica condicional para lidar melhor com curvas e bifurcações, além da aplicação de técnicas de filtragem de leitura dos sensores para aumentar a

confiabilidade em diferentes condições de iluminação. Outra possibilidade seria explorar um controle PID (Proporcional, Integral e Derivativo) como explorado por Oguten e Kabas (2021), permitindo o controle contínuo baseado em erro acumulado e derivado.

Referências

Arduino (2025). "Uno R3". Disponível em: https://docs.arduino.cc/hardware/uno-rev3/ Acesso em: 21 mar. 2025.

Boylestad, R. "Introdução à Análise de Circuitos". 12ª ed. LTC, 2013.

Brigido, W. J. H.; Oliveira, J. M. P. (2025). "The line follower robot: a meta-analytic approach". PeerJ Computer Science, v. 11, p. e2744, 19 mar. 2025.

Condori, K.; Pozo, S. (2023). "Educational Robotics in Action: Development of a Line Following Robot through STEAM Methodology to Address Traffic Issues". CITIE 2023. Disponível em: ceur-ws.org/Vol-3691/paper26.pdf. Acesso em: 17 jul. 2025.

Santos, L. C. et. al (2024). "Lógica de programação através da robótica: uso do Scratch e Arduino para criação de robôs e projetos interativos". *Lumen et Virtus*, Vol. 15 Nº 39, Vol. XV Núm. XXXIX p. 2408–2422, 2024. Disponível em: periodicos.newsciencepubl.com/LEV/article/view/207. Acesso em: 16 jul. 2025.

Fresnillo, P. et al. (2023). "A method for understanding and digitizing manipulation activities using programming by demonstration in robotic applications." Robotics and Autonomous Systems, Volume 170.

Oguten, S.; Kabas, B. (2021). "PID Controller Optimization for Low-cost Line Follower Robots". Disponível em: https://arxiv.org/abs/2111.04149. Acesso em: 17 jul. 2025.

Oliveira, H. R. (2025) *Line-Follower-Cart.* GitHub. Disponível em: https://github.com/OzyHil/Line-Follower-Cart.git. Acesso em: 19 jun. 2025.

Reda, M. et. al (2024). "Path planning algorithms in the autonomous driving system: A comprehensive review". Robotics and Autonomous Systems, Volume 174.

Scherer, D. et. at (2020). "Robótica educacional de baixo custo: Arduino como ferramenta pedagógica". Congresso sobre Tecnologias na Educação (Ctrl+E), 2020. SBC. Disponível em: https://sol.sbc.org.br/index.php/ctrle/article/view/11418. Acesso em: 16 jul. 2025.

Sgai, L. M. et al. (2013). "Uso da plataforma Arduino para o ensino e o aprendizado de robótica". Universidade Federal de Santa Catarina, 2013. Disponível em: https://online-engineering.org/icbl-archives/proceedings/2013/papers/Contribution77_a.pdf. Acesso em: 16 jul. 2025.

Silva, M. R. et al. (2025). "Arduino e Internet das Coisas aplicados na educação e inclusão social". *Revista Aracê*, São José dos Pinhais, v. 7, n. 7, p. 37579–37596, 2025. Disponível em: https://periodicos.newsciencepubl.com/arace/article/view/6537. Acesso em: 16 jul. 2025.

Straub, M. (2019). "Motor Shield L293D – Driver Ponte H no Controle de Motores" Disponível em: https://www.usinainfo.com.br/blog/motor-shield-l293d-driver-ponte-h-no-controle-de-motores/. Acesso em: 22 mar. 2025.

Vidal, V. (2017). "Sensor Óptico TCRT5000 com Arduino". Disponível em: https://blog.eletrogate.com/sensor-optico-tcrt5000-com-arduino/. Acesso em: 17 mar. 2025.