Plataforma autônoma para logística

Eliana So [1]; Pedro Bauke [1]; Andressa Martins [2]; Rodrigo França [2]

[1] Aluno do Instituto Mauá de Tecnologia (IMT);
[2] Professor do Instituto Mauá de Tecnologia (IMT).

Resumo. *Este projeto apresenta o desenvolvimento de um protótipo de Veículo Guiado Autonomamente (AGV) voltado para a logística interna, motivado pela crescente demanda do e-commerce no Brasil. A plataforma robótica foi construída utilizando materiais acessíveis, como MDF e impressão 3D, e é controlada por um microcontrolador RP2040. O sistema de navegação baseia-se na leitura de linha por meio de sensores de refletância e um algoritmo de controle PID, integrando também detecção de obstáculos via ultrassom e telemetria em tempo real via Bluetooth para um dashboard web. Durante os testes, desafios relacionados à aderência mecânica e ao driver de potência exigiram a substituição de componentes (do driver MX1508 para o L298N) e adaptações nas rodas. Apesar das dificuldades e da não implementação do módulo inercial (MPU) devido ao prazo, o protótipo final atingiu os objetivos propostos, demonstrando capacidade de navegar autonomamente, interromper o movimento ao detectar obstáculos e comunicar dados operacionais com sucesso.*

Introdução

O varejo online tem, cada vez mais, se tornado a opção primária para compras dos brasileiros, como mostra um estudo realizado pela FGV, que expõe que as vendas do e-commerce já representam mais de 10% do faturamento do setor desde outubro de 2021 e em fevereiro de 2025 chegaram a representar 17,8%. A Figura 1 apresenta um histórico da parcela do e-commerce no varejo.

Figura 1 – Fatia de e-commerce no varejo

Fonte: Valor Econômico (2025)

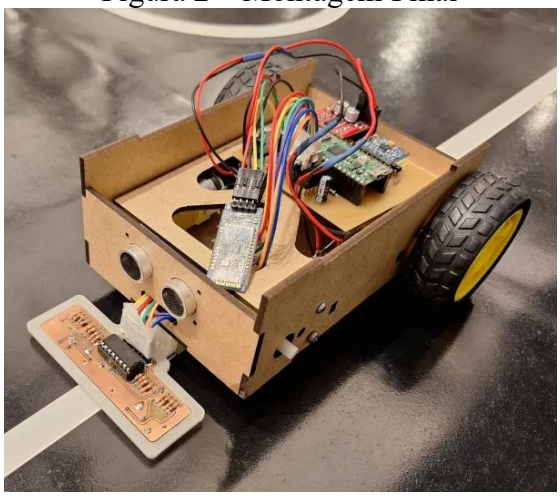
Olhando para esse cenário, diversas empresas estão investindo em centros de distribuição e logística, e maneiras de automatizar esses centros. Uma dessas maneiras é com a utilização de AGVs (Autonomous Guided Vehicles), ou robôs autônomos que são capazes de se movimentar e carregar cargas dentro dos armazéns, facilitando e agilizando a logística.

Observando essa tendência, optou-se pelo desenvolvimento de um protótipo de uma plataforma capaz de detectar obstáculos e navegar de forma autônoma em ambientes controlados, necessitando apenas de uma linha para poder seguir e andar de um ponto a outro.

Material e Métodos

Por se tratar de um protótipo, optou-se por utilizar materiais de fácil acesso, como MDF e filamento plástico, e métodos de fabricação rápidos, como corte à laser e impressão 3D, para poder iterar de maneira rápida entre versões. A Figura 2 apresenta a montagem final do protótipo.

Figura 2 – Montagem Final

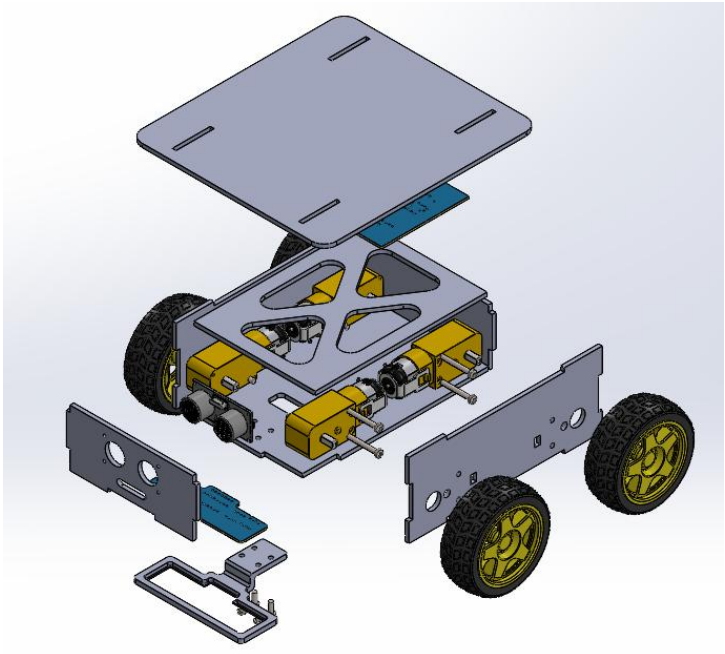


Fonte: Os Autores (2025)

Mecânica

Por se tratar de um protótipo, o robô possui dimensões pequenas (210 x 155 x 75 mm), a Figura 3 ilustra o projeto mecânico em CAD.

Figura 3 – Vista Explodida em CAD



Fonte: Os autores (2025)

O corpo principal, que possui um formato de caixa, é formado por chapas de MDF fabricadas por corte a laser. Dentro comporta-se os motores, a placa eletrônica principal e o sensor ultrassônico.

O robô utiliza-se de 4 motores DC com caixa de redução para ser capaz de transportar objetos mais pesados, devido à melhor distribuição de peso e torque. As rodas são de 65mm de diâmetro e possuem pneus de borracha com um encaixe para os motores.

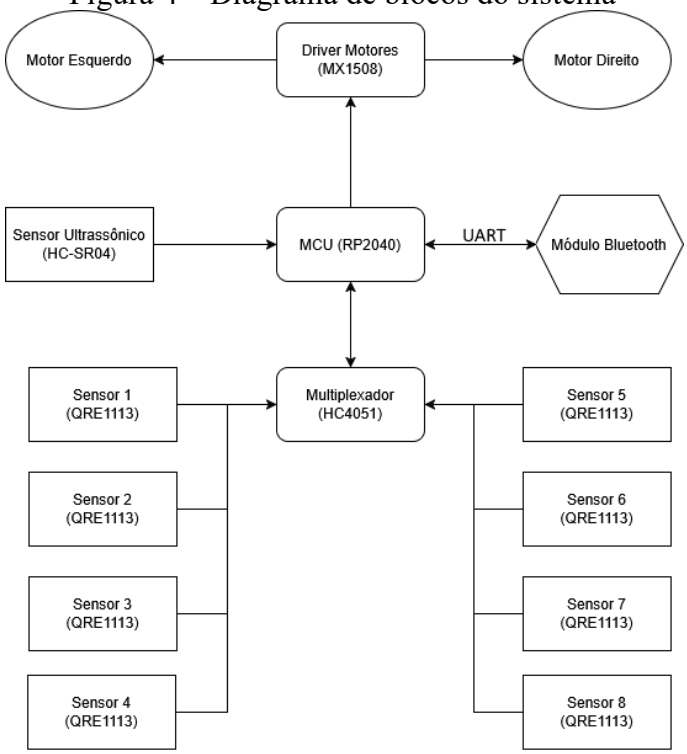
Foi feito um suporte em impressão 3D, preso por parafusos, para comportar a barra de sensores frontal e mantêm uma distância de 5mm entre o chão e os sensores para a diminuição de erros de leitura.

A mesa superior, onde ficam os itens os quais querem ser movidos, é feito de MDF de 3 mm e é modular, podendo-se criar outros formatos, desde que possuam os encaixes corretos.

Eletrônica

Para a criação e fabricação das placas de circuito impresso, foi montado o diagrama de blocos ilustrado pela Figura 4, que representa de forma simples e clara como cada elemento do circuito se comunica, explicitando como os componentes trocam informações entre si.

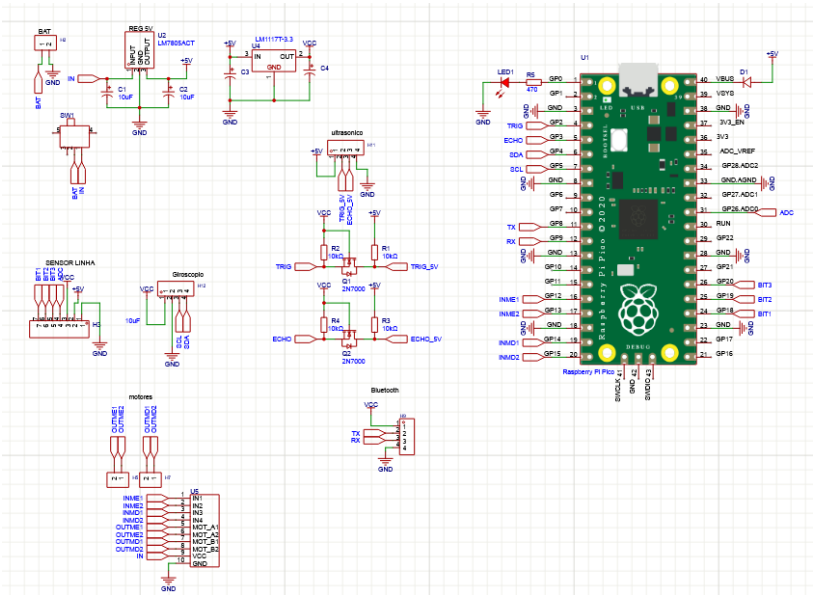
Figura 4 – Diagrama de blocos do sistema



Fonte: Os Autores (2025)

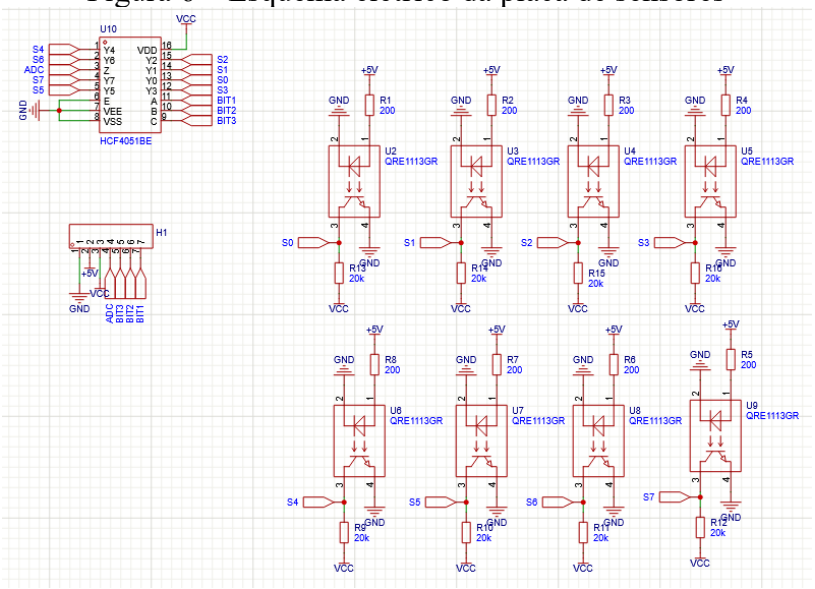
Optou-se por ter duas placas de circuito separadas, para facilitar a montagem, pois os sensores utilizados para "enxergar" a linha precisam estar próximos ao chão. As Figuras 5 e 6 apresentam os esquemas elétricos das duas placas.

Figura 5 – Esquema elétrico da placa de controle



Fonte: Os Autores (2025)

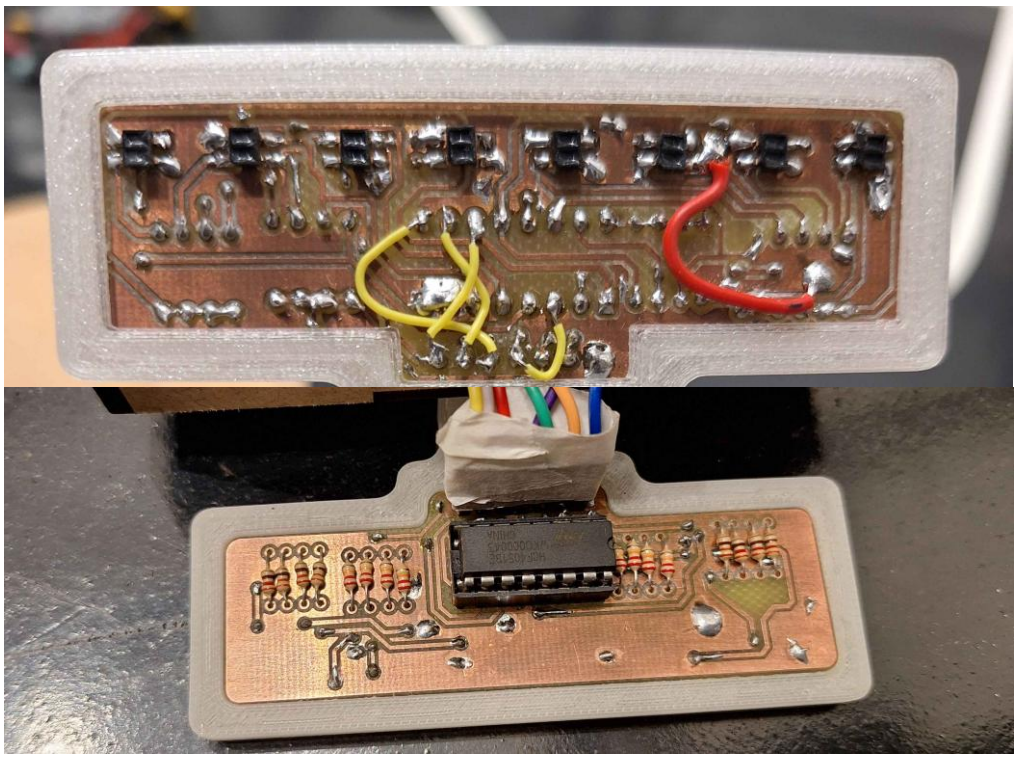
Figura 6 – Esquema elétrico da placa de sensores



Fonte: Os Autores (2025)

Na placa dos sensores estão localizados os 8 sensores de refletância QRE1113 e o multiplexador analógico HC4051, que possui 8 canais e é responsável por garantir que todos os sensores possam ser lidos pelo microcontrolador, uma vez que o RP2040 apresenta apenas 3 canais ADC disponíveis. A Figura 7 apresenta a barra dos sensores com os componentes já soldados.

Figura 7 – Barra dos Sensores

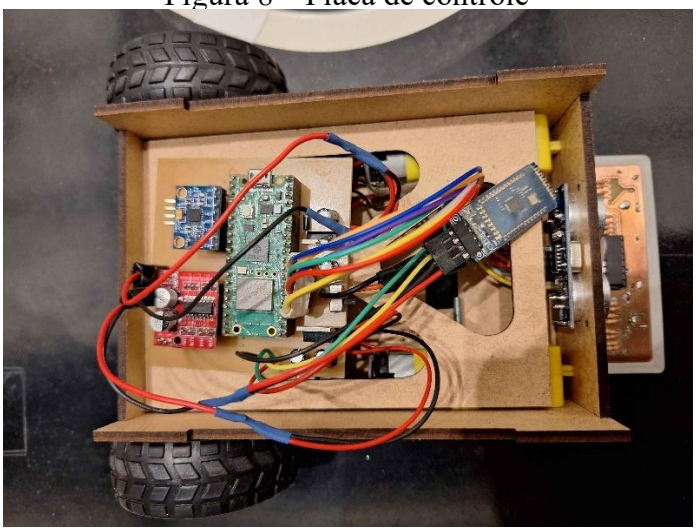


Fonte: Os Autores (2025)

Para comunicar com o multiplexador é preciso escolher qual porta se deseja ler, para isso, 3 GPIOs do microcontrolador são conectados aos pinos de controle do CI para realizar essa seleção via software.

A PCB de controle contém o microcontrolador e os demais componentes para o funcionamento do robô. Nessa placa também se conecta a bateria, LiPo de 2 células, que alimenta o projeto pois os motores escolhidos suportam até 9V. Para alimentar os demais componentes existem dois reguladores de tensão no circuito, utiliza-se do LM7805 para gerar uma tensão de 5V e um LM1117 que gera a tensão de 3,3V, como pode ser visto na Figura 8.

Figura 8 – Placa de controle



Fonte: Os Autores (2025)

Para o controle dos motores foi escolhido o Driver MX1508, é uma ponte H dupla onde recebe sinais PWM do microcontrolador e transforma em direção e velocidade aos motores DC.

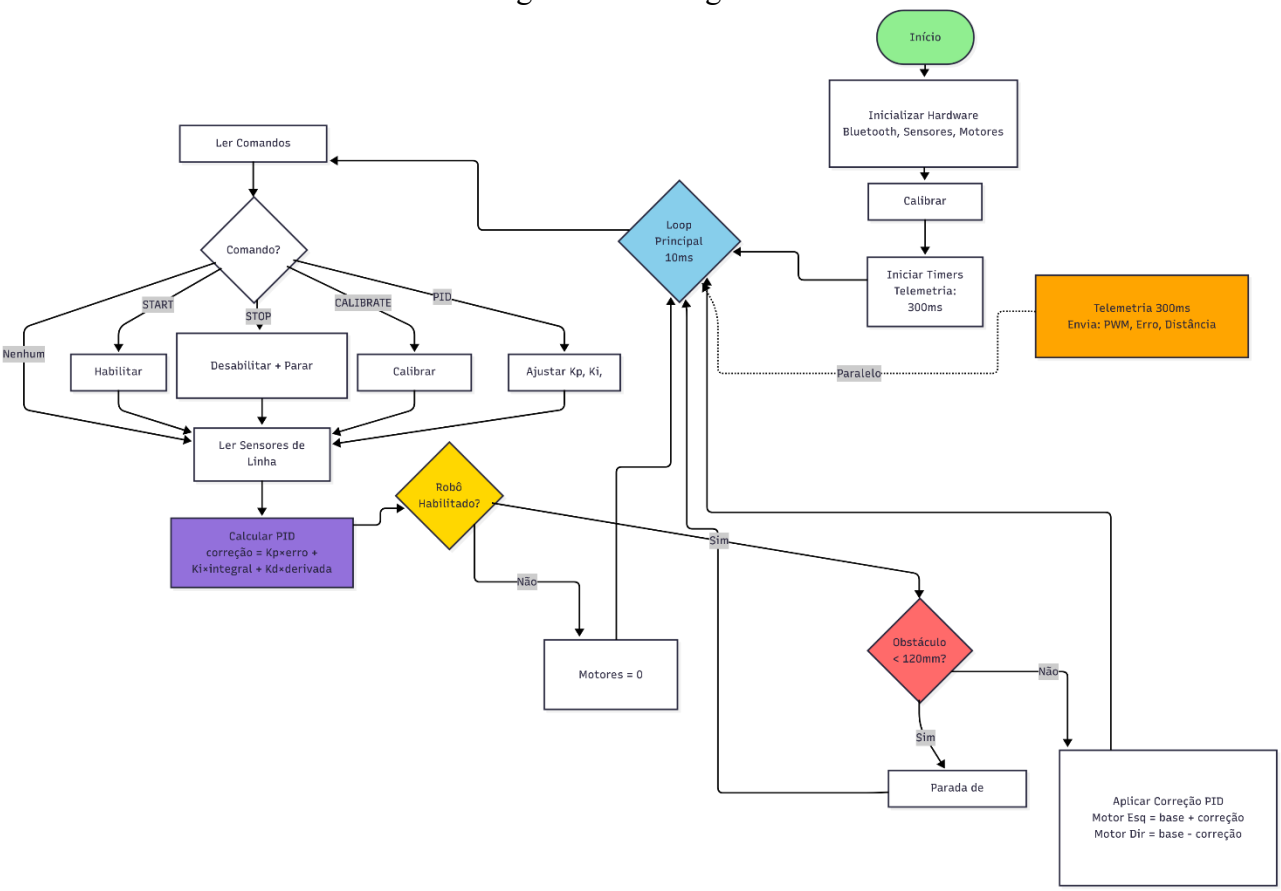
Adicionou-se uma MPU (acelerômetro e giroscópio) com o objetivo de obter a orientação do robô em um determinado percurso a partir da integração da aceleração obtida.

Na placa principal se encontra um módulo bluetooth, responsável por enviar informações do robô, como duty cicle do PWM de cada motor e o erro da leitura dos sensores de linha para um computador de apoio, que apresenta esses dados em um dashboard.

Código

O software do robô é escrito em C++, utilizando-se de classes e objetos para criar um código de fácil entendimento e manutenção. Além disso, cada classe está localizada em seu próprio par de arquivos header e cpp. O arquivo principal do programa contém a declaração de todas as classes utilizadas, uma interrupção de timer e o loop principal. O código completo pode ser visualizado no repositório do projeto no GitHub. A Figura 9 apresenta o fluxograma do sistema.

Figura 9 – Fluxograma



Fonte: Os Autores (2025)

Para controlar os motores, criou-se a classe Motor, que configura dois canais de PWM para atuarem a 20kHz de frequência, com um wrap máximo de 1023, para se ter uma resolução de aproximadamente 7,8mV.

O método *set_speed* recebe como argumento um número inteiro, checa se esse valor está dentro do intervalo permitido e a partir do sinal desse valor habilita um dos canais de PWM, uma vez que o driver utilizado possui apenas duas entradas de PWM, e a direção de rotação do motor é escolhida a partir de qual canal está ativo.

A classe SensorArray é responsável por controlar o multiplexador e realizar a leitura dos sensores. Para garantir uma leitura mais estável e homogênea entre os sensores, é realizada uma calibração no momento que o robô é ligado, guardando em arrays as leituras máximas e mínimas de cada um dos oito sensores, que são usadas posteriormente para realizar uma troca de escala na leitura dos sensores.

O controle do multiplexador é realizado pelo método *_set_mux_channel* que recebe o canal que se deseja ler como um número inteiro, e utilizando operações de bit shift para a direita, configura os GPIOs de controle para selecionar a porta correta.

O método que retorna a leitura dos sensores para o programa principal é o *calculate_error*, que realiza a leitura dos sensores e calcula uma média ponderada dos pesos de cada um dos sensores, a Equação 1 apresenta a fórmula utilizada para o cálculo dessa média.

$$error = \left(\frac{\sum_{i=0}^{7} value[i]*(i*1000)}{\sum_{i=0}^{7} value[i]}\right) - setpoint \quad \text{(I)}$$

O *value[i]* representa a leitura do sensor da posição *i*, onde o sensor mais à esquerda da placa se encontra na posição 0 e o mais à direita, na 7, normalizado para uma escala de 0 a 1000, e o termo $(i * 1000)$ apresenta o peso atribuído ao respectivo sensor. A equação retorna um valor que pode variar entre 0 e 7000, de acordo com o deslocamento do robô em relação a linha. A Equação 2 demonstra a normalização realizada com a leitura dos sensores.

$$value[i] = \frac{(raw[i] - \min[i])*1000}{(\max[i] - \min[i])} \quad \text{(II)}$$

Onde $value[i]$ apresenta o valor normalizado, $raw[i]$, a leitura bruta do ADC, e $\max[i]$ e $\min[i]$, os máximos e mínimos do sensor em questão, obtidos pela calibração.

O sensor ultrassônico é controlado pela classe Ultrasonic, que ativa o pino *trigger* do sensor por um breve período e calcula a distância dos objetos na frente do robô utilizando a Equação 3.

$$distance_{mm} = \frac{pulseDuration*343}{(2*1000)} \quad \text{(III)}$$

O valor de *pulseDuration* é obtido calculando a duração do sinal HIGH do pino *echo* do sensor, o valor de 343 representa a velocidade do som, em m/s. Divide-se por 2 para obter somente o tempo de volta da onda até o sensor.

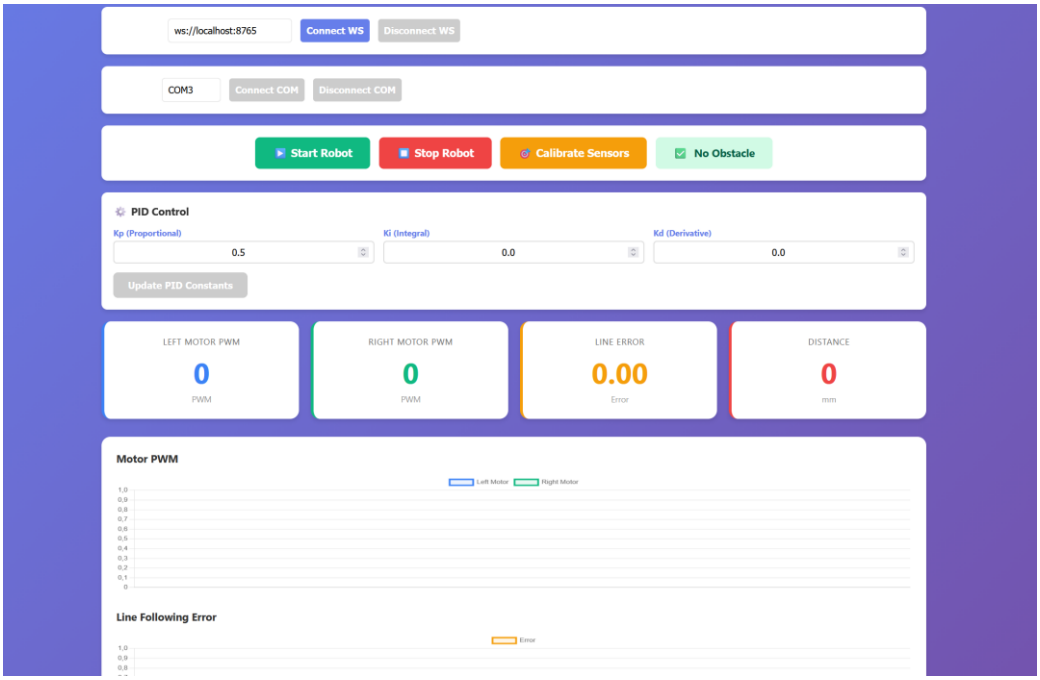
O loop principal consiste em uma malha de controle que atua para corrigir a posição da plataforma autônoma em relação a linha. A malha tem como período de atuação 10ms e possui um controlador PID que recebe o valor de retorno do método *calcuate_erro* como entrada. A Equação 4 apresenta a equações de diferenças do controlador utilizado.

$$correction = kp * error + kd * (error - lastError) + ki * integralError \quad \text{(IV)}$$

O sensor ultrassônico é lido no início do loop de controle e caso a distância a um objeto seja menor que 120mm, o PWM enviado aos motores é zerado, ou seja, o robô para. Optou-se por esse princípio para se ter uma resposta mais fluída da detecção de objetos.

O envio dos dados de telemetria ocorre através de uma interrupção de timer que acontece a cada 200ms, na qual são enviados os dados atuais de PWM de cada motor, distância medida pelo ultrassônico e o erro dos sensores de linha via Bluetooth (UART1 com 9600 de baudrate) para um computador de apoio. Este executa um servidor WebSocket local desenvolvido em Python que atua como ponte entre a comunicação serial Bluetooth e um dashboard web. O dashboard, hospedado localmente via servidor HTTP, utiliza WebSocket para receber telemetria em tempo real e enviar comandos de controle (START/STOP) de volta ao robô, exibindo gráficos dinâmicos atualizados através da biblioteca Chart.js. A Figura 10 apresenta a tela do dashboard.

Figura 10 – Dashboard



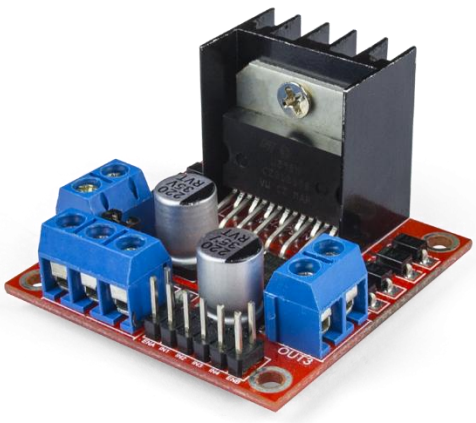
Fonte: Os Autores (2025)

Resultados e Discussão

Com as PCBs fabricadas, notou-se um pequeno erro no projeto da barra de sensores, algumas conexões do CI multiplexador estavam na camada errada, uma vez que as placas fabricadas no Instituto Mauá de Tecnologia não possuem as vias interligadas.

Foi inserido na placa principal um módulo de medição inercial, porém, optou-se por não o utilizar. Outro problema que foi encontrado durante os testes do protótipo foi a falta de aderência das rodas, fazendo com que o robô derrapasse e por algumas vezes, travasse enquanto tentava andar. Esse problema acabou queimando um dos canais do driver que estava sendo utilizado, por causa disso, ocorreu a troca do driver para um L298N. A Figura 11 apresenta o novo driver utilizado.

Figura 11 – Driver L298N



Fonte: RoboCore (2025)

A principal diferença entre o L298N e o MX1805 é a quantidade de pinos para controle dos drivers. Enquanto o MX1805 necessita de apenas 2 sinais de PWM para seu funcionamento, o L298N

necessita de 1 sinal PWM e dois sinais de direção, logo foi preciso reescrever a classe de controle de motor para poder controlar esse novo driver.

Para solucionar o problema de aderência das rodas, colocou-se elásticos nas rodas do projeto. Essa solução se mostrou eficaz, e após essa mudança, o AGV não apresentou mais dificuldades para se movimentar. A Figura 12 mostra como as rodas ficaram com o elástico alocado.

Figura 12 – Rodas com elástico

Fonte: Os Autores (2025)

O controle de quatro motores se mostrou mais difícil que o esperado, principalmente no que se diz respeito ao sincronismo dos motores, por isso, optou-se pelo uso de somente dois, na parte traseira do AGV. Para evitar que os sensores de linha raspassem no chão, um pedaço de feltro foi colado na barra de sensores para servir de terceiro apoio.

Apesar dos problemas encontrados, foi possível fazer com que o protótipo funcionasse da maneira esperada, sendo possível controlar e monitorar o robô a partir do dashboard, além da detecção de obstáculos estar funcionando adequadamente, fazendo com que o robô pare quando um objeto é detectado até 120mm a sua frente.

Por fim, a implementação da MPU não foi possível de ser realizada devido ao prazo de entrega do projeto, porém um software para a leitura dele foi escrito, mas não foi testado extensivamente.

Conclusões

Com base nos resultados apresentados e nas discussões realizadas ao longo do desenvolvimento do projeto, conclui-se que o objetivo de desenvolver uma plataforma autônoma para logística foi alcançado com êxito. O protótipo demonstrou ser capaz de seguir a trajetória estipulada e realizar a parada de segurança ao detectar obstáculos a uma distância de 120mm, validando a lógica de controle e a integração dos sensores.

O projeto evidenciou a importância da iteratividade na engenharia. Problemas críticos encontrados na fase de testes, como a falta de tração das rodas e a queima de canais do driver de motor original, foram solucionados de maneira eficaz através da implementação de elásticos para aumentar o atrito e da substituição do hardware de potência pelo driver L298N. Além disso, a simplificação do sistema de tração de quatro para dois motores traseiros mostrou-se uma decisão acertada para resolver as dificuldades de sincronismo encontradas.

No âmbito do software e eletrônica, a arquitetura baseada no microcontrolador RP2040 e a implementação de classes em C++ permitiram um código organizado e eficiente, facilitando a manutenção e a execução da malha de controle PID de 10ms. A criação de um dashboard web com comunicação via WebSocket provou ser uma ferramenta valiosa para monitoramento e telemetria em tempo real. Embora a integração do sensor inercial (MPU) não tenha sido concluída devido a restrições de prazo, o robô opera funcionalmente dentro das especificações fundamentais estabelecidas.

Dessa forma, o trabalho apresenta um protótipo funcional que serve como prova de conceito para a automação logística de baixo custo, com oportunidades futuras para o aprimoramento da fusão de sensores e navegação inercial.

Referências Bibliográficas

VALOR ECONÔMICO (Brasil). **Fatia do on-line no varejo quase dobra desde a pandemia**. 2025. Disponível em: https://valor.globo.com/empresas/noticia/2025/03/27/fatia-do-on-line-no-varejo-quase-dobra-desde-a-pandemia.ghtml. Acesso em: 23 nov. 2025.

POUNDER, Les. **How to Use an Ultrasonic Sensor with Raspberry Pi Pico**. 2021. Disponível em: https://www.tomshardware.com/how-to/raspberry-pi-pico-ultrasonic-sensor. Acesso em: 25 nov. 2025.

RASPBERRY PI FOUNDATION. **RP2040 Datasheet**: a microcontroller by raspberry pi. A Microcontroller by Raspberry Pi. 2025. Disponível em: https://pip-assets.raspberrypi.com/categories/814-rp2040/documents/RP-008371-DS-1-rp2040-datasheet.pdf?disposition=inline. Acesso em: 27 nov. 2025.