



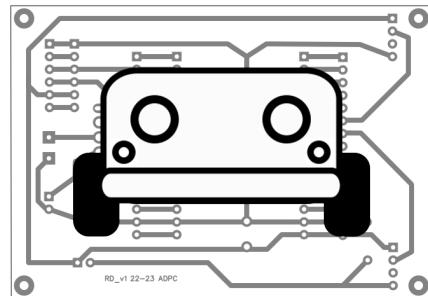
INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA (ISEL)

DEPARTAMENTO DE ENGENHARIA ELETRÓNICA E DE
TELECOMUNICAÇÕES E COMPUTADORES (DEETC)

LEIM

LICENCIATURA EM ENGENHARIA INFORMÁTICA E MULTIMÉDIA
UNIDADE CURRICULAR DE PROJETO

Desenvolvimento de robô didático



Alexandre Miguel Calejo de Figueiredo 48577

Daniela Filipa Valadas Gonçalves 48579

Orientadores

Professor Doutor

Carlos Jorge de Sousa Gonçalves

Professor Doutor

Pedro Miguel Florindo Miguens Matutino

8 de julho de 2023

Resumo

O projeto apresentado neste relatório descreve o desenvolvimento de um robô didático capaz de realizar movimentos no espaço 2D. O protótipo desenvolvido possui um conjunto de sensores e atuadores interligados. Além disso, o robô permite estabelecer comunicação com um computador/telemóvel através de comunicação Bluetooth.

O presente projeto tem como finalidade desenvolver um ambiente didático para a exploração de conceitos relacionados à robótica, como o controlo de movimento, comunicação sem fio e programação em Java. O protótipo desenvolvido é um robô didático funcional que pode ser utilizado como ferramenta de aprendizagem.

Palavras-chave: Robótica, plataforma didática, controlo remoto, microcontrolador, ESP32.

Abstract

The project presented in this report describes the development of an educational robot capable of performing movements in 2D space. The developed prototype consists of a set of interconnected sensors and actuators. Furthermore, the robot allows communication with a computer/mobile device through Bluetooth communication.

This project aims to create an educational environment for exploring concepts related to robotics, such as motion control, wireless communication, and Java programming. The developed prototype is a functional educational robot that can be used as a learning tool.

Keywords: Robotics, didactic platform, remote control, microcontroller, ESP32.

Agradecimentos

Gostaríamos de expressar o nosso sincero agradecimento a todas as pessoas que contribuíram para a conclusão bem-sucedida deste projeto final de licenciatura. Em particular, gostaríamos de agradecer:

Aos nossos orientadores, Doutor Carlos Gonçalves e Doutor Pedro Matutino, pelo apoio constante, orientação e valiosas sugestões ao longo de todo o processo. As suas disponibilidades foram fundamentais para o desenvolvimento deste trabalho.

Não poderíamos deixar de expressar o nosso agradecimento aos amigos que fizemos ao longo do nosso percurso académico nesta turma. Agradecemos por estarem sempre dispostos a ajudar, por partilharem conhecimento e por serem parte essencial do nosso crescimento pessoal e profissional.

Além disso, agradecer um ao outro, pela nossa parceria neste trabalho de grupo. A colaboração, comunicação aberta e trabalho em equipa foram fatores-chave para o sucesso deste projeto.

Por fim, gostaríamos de agradecer aos nossos familiares e entes queridos, pelo apoio incondicional, compreensão e paciência durante todo o período de realização deste projeto. As suas palavras encorajadoras e incentivo foram fundamentais para manter a nossa motivação.

Atenciosamente,

Alexandre Figueiredo

Daniela Gonçalves

Índice

| | |
|--|-------------|
| Resumo | i |
| Abstract | iii |
| Agradecimentos | v |
| Índice | vii |
| Lista de Figuras | ix |
| Lista de Tabelas | xi |
| Lista de Acrónimos | xiii |
| 1 Introdução | 1 |
| 1.1 Motivação | 1 |
| 1.2 Contextualização | 1 |
| 1.3 Objetivos | 2 |
| 1.4 Organização do Documento | 3 |
| 2 Trabalho Relacionado | 5 |
| 3 Modelo Proposto | 7 |
| 3.1 Requisitos | 7 |
| 3.2 Fundamentos | 9 |
| 4 Arquitetura do Sistema | 11 |
| 4.1 Estrutura | 11 |
| 4.2 Motor e Alimentação | 12 |
| 4.3 Controlador do Motor | 14 |
| 4.4 Sensores | 16 |
| 4.4.1 Sensor de Distância | 16 |
| 4.4.2 Sensor de Toque | 17 |
| 4.5 Microcontrolador | 17 |

| | | |
|----------|---|-----------|
| 5 | Implementação | 19 |
| 5.1 | Montagem | 20 |
| 5.2 | Programação | 23 |
| 5.3 | Construção da placa de circuito impresso | 32 |
| 6 | Validação e Testes | 35 |
| 6.1 | Desvio Antes da Paragem | 35 |
| 6.2 | Diferencial por Software | 35 |
| 7 | Conclusão e Trabalho Futuro | 41 |
| A | Esquemática da placa de circuito impresso | 43 |
| B | Diagrama de classes Unified Modeling Language (UML) completo | 45 |
| | Bibliografia | 47 |

Lista de Figuras

| | | |
|------|--|----|
| 1.1 | Esquema abstrato | 2 |
| 2.1 | Robô Lego Mindstorms | 5 |
| 2.2 | Robô autónomo | 5 |
| 3.1 | Casos de utilização | 8 |
| 4.1 | Chassi | 12 |
| 4.2 | Roda e motor <i>Direct current (DC)</i> | 12 |
| 4.3 | Roda com disco | 14 |
| 4.4 | <i>Duty cycle</i> a 50% | 15 |
| 4.5 | Esquema simplificado da Ponte H | 16 |
| 4.6 | Diagrama simplificado do funcionamento de microcontrolador | 17 |
| 5.1 | Diagrama do sistema em blocos | 19 |
| 5.2 | Montagem inicial | 20 |
| 5.3 | Diagrama ilustrativo do circuito | 21 |
| 5.4 | Esquema elétrico | 22 |
| 5.5 | Soldagem e finalização da montagem | 22 |
| 5.6 | Diagrama de classes UML simplificado | 24 |
| 5.7 | Diagrama de uma curva à esquerda | 29 |
| 5.8 | Layout da PCB | 32 |
| 5.9 | Revelação das pistas | 32 |
| 5.10 | Primeira iteração | 33 |
| 5.11 | Impressão final | 33 |
| 5.12 | Montagem da PCB | 33 |
| 5.13 | Furação da PCB | 33 |
| 6.1 | Testes práticos com rodas no ar | 36 |
| A.1 | Esquemática da PCB | 43 |
| B.1 | Diagrama de classes UML completo | 45 |

Lista de Tabelas

| | | |
|-----|--|----|
| 2.1 | Comparação entre Trabalhos Relacionados | 6 |
| 4.1 | Especificações técnicas do <i>chassi</i> – c/ motores e rodas | 11 |
| 4.2 | Especificações técnicas do motor DC | 12 |
| 4.3 | Especificações técnicas da bateria Gens Ace | 13 |
| 4.4 | Especificações técnicas do <i>encoder</i> H206 | 14 |
| 4.5 | Especificações técnicas do controlador L298N | 16 |
| 4.6 | Especificações técnicas do sensor de distância <i>Sound Navigation and Ranging</i> (SONAR) | 17 |
| 4.7 | Especificações técnicas do sensor de toque | 17 |
| 4.8 | Especificações técnicas do microcontrolador <i>Espressif32</i> (ESP32) . . | 18 |
| 5.1 | Valores lógicos para controlo | 26 |
| 6.1 | Testes práticos com rodas no ar | 36 |

Lista de Acrónimos

DC *Direct current.* ix, 12, 14, 16

ESP32 *Espressif32.* xi, 18, 30

GPIO *General Purpose Input/Output.* 18

I2C *Inter-Integrated Circuit.* 9, 18, 24

KB Kilobyte (10^3). 18

MB Megabyte (10^6). 18

PCB *Printed circuit board.* 3, 7, 9, 32, 33

PWM *Pulse Width Modulation.* 9, 14, 23–25, 37

RAM *Random Access Memory.* 18

SONAR *Sound Navigation and Ranging.* xi, 16, 21

UML *Unified Modeling Language.* viii, ix, 23, 24, 45, 46

Wi-Fi *Wireless Fidelity.* 18

Introdução

1.1 Motivação

O presente relatório enquadra-se na realização do projeto correspondente à unidade curricular Projeto (PRJ), integrante do plano de estudos da licenciatura em Engenharia Informática e Multimédia (LEIM), pertencente ao Departamento de Engenharia Eletrónica e de Telecomunicações e Computadores (DEETC).

Apesar dos conceitos relacionados com a eletrónica sejam abordados poucas vezes no curso mencionado, sendo o mesmo mais direcionado para a programação de software, a motivação para a realização deste projeto surgiu a partir do gosto dos estudantes pela robótica e também pelo objetivo de adquirir novos conhecimentos ao explorar esta área num projeto prático e relevante.

Os autores também perceberam que havia uma procura crescente por soluções de robótica educacional em escolas e universidades, e que muitos dos produtos disponíveis no mercado são caros, limitados em termos de funcionalidades e pouco flexíveis em relação à adição de novos componentes. Com base nisso, decidiu-se desenvolver uma plataforma didática que fosse acessível, versátil e de fácil utilização, capaz de ser adaptada para diferentes níveis de ensino e para diferentes propósitos pedagógicos. A plataforma desenvolvida é programável e a sua estrutura genérica permite a introdução de mais componentes, permitindo assim, que os estudantes possam explorar diferentes cenários e desafios e também personaliza-la de acordo com suas necessidades e interesses.

1.2 Contextualização

Na robótica, sensores e atuadores são elementos fundamentais para a criação de sistemas inteligentes. Sensores são utilizados para captar informações do ambiente, enquanto os atuadores são utilizados para interagir com o meio envolvente. Com a inclusão de um controlador, dispositivo que coordena as informações fornecidas pelos sensores e os sinais enviados aos atuadores, permite que os robôs possam

perceber o ambiente ao seu redor e executar tarefas, de forma autónoma sem intervenção Humana.

A inclusão de sensores e atuadores permite aos utilizadores explorarem conceitos como controlo de movimento e deteção de obstáculos. Neste projeto, o protótipo irá utilizar sensores de distância, velocidade e toque, controladores de motor, motores e rodas para permitir a movimentação do robô, e tecnologia de comunicação sem fios para permitir a comunicação com o computador.

Na Figura 1.1 encontra-se representado um esquema simples do funcionamento do sistema a implementar no presente projeto.

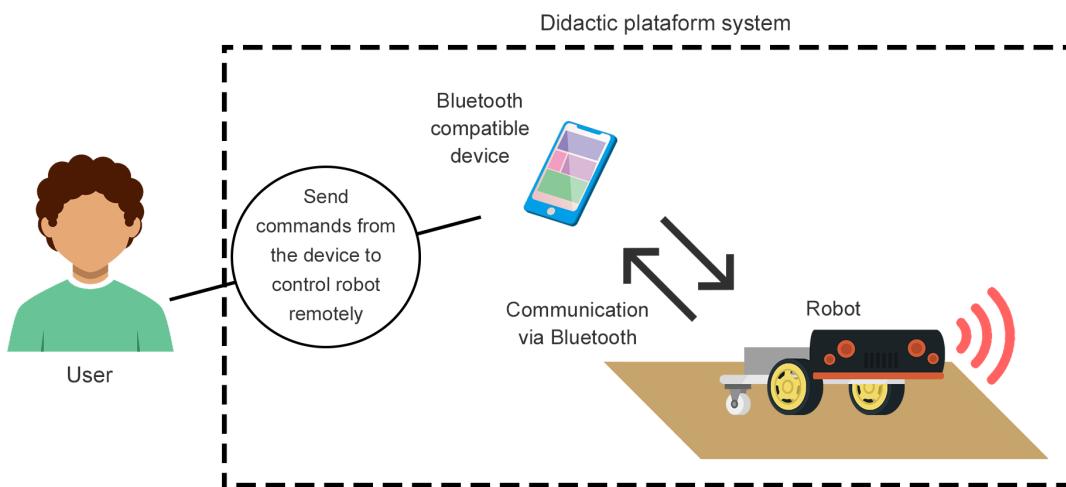


Figura 1.1: Esquema abstrato

1.3 Objetivos

Dessa forma, foram estabelecidos os principais objetivos deste projeto:

- 1) realização de pesquisa e estudo do estado da arte, com o objetivo de selecionar os componentes adequados e aprofundar o conhecimento sobre cada um deles;
- 2) montagem dos componentes eletrónicos, microcontrolador, expansor, controlador de motor, sensores de velocidade, motores e rodas, e programação dos componentes de forma a permitir a realização de movimentos no espaço 2D;

- 3) estabelecimento da comunicação entre o computador e a plataforma didática via comunicação Bluetooth, os movimentos do robô serão controlados a partir dos comandos enviados pelo computador;
- 4) desenvolvimento de uma *Printed circuit board* (PCB) para tornar a montagem e desmontagem mais fácil e acessível, garantindo um processo de aprendizagem mais célere. Ao utilizar blocos de interligação e camadas separadas na PCB;
- 5) inclusão de sensores de distância e toque, com o propósito de detetar obstáculos e evitar colisões que possam causar danos materiais.

1.4 Organização do Documento

O documento está organizado em sete capítulos. O primeiro capítulo é a introdução, que apresenta a motivação, os objetivos e a organização do presente documento.

O capítulo 2 é dedicado aos trabalhos relacionados, onde são descritos outros estudos e pesquisas realizadas nesta área.

O terceiro capítulo apresenta o modelo proposto, incluindo os requisitos, fundamentos e abordagem, de modo a detalhar a proposta de solução para o problema abordado.

O quarto capítulo, é discutida a arquitetura do sistema proposto, incluindo a estrutura geral, os motores, o controlador do motor e os sensores.

O capítulo 5 é dedicado à implementação do protótipo, incluindo a montagem e programação do sistema proposto.

No capítulo 6, são apresentados os resultados da validação e dos testes práticos realizados com o sistema, a fim de avaliar a eficácia e eficiência da solução proposta.

Por fim, no sétimo e último capítulo, são apresentadas as conclusões da pesquisa, bem como possíveis direções para trabalhos futuros sobre a plataforma desenvolvida.

Trabalho Relacionado

Uma referência importante para o desenvolvimento do presente projeto foi o robô da Lego Mindstorms utilizado nas aulas de Fundamentos de Sistemas Operativos, cuja biblioteca foi desenvolvida por [1].

Com o mesmo objetivo, a plataforma mencionada tem como intuito tornar as aulas mais interativas, de forma a ajudar a explorar conceitos relacionados com sistemas operativos. O robô referido, apresentado na Figura 2.1, era controlado remotamente pela biblioteca mencionada e capaz de se movimentar no espaço 2D.

Um outro projeto a referenciar é um robô feito por [2], realizado no âmbito de Projeto Final de Curso, do curso de licenciatura em Engenharia Eletrónica e Telecomunicações e de Computadores.

O robô em questão possui a capacidade de se movimentar no espaço 2D, de forma autónoma, evitando colisões com paredes e obstáculos. O mesmo encontra-se apresentado na Figura 2.2.



Figura 2.1: Robô Lego Mindstorms



Figura 2.2: Robô autônomo

Uma das principais vantagens do robô é sua acessibilidade económica. Enquanto, por exemplo, o robô da Lego Mindstorms é conhecido por sua qualidade e recursos avançados, o protótipo do robô aqui desenvolvido tem um foco especial na redução

Capítulo 2. Trabalho Relacionado

de custos. Isso torna-o uma opção mais acessível para estudantes e entusiastas de robótica que possuem orçamentos mais limitados.

Outra vantagem significativa do nosso robô é o facto de ser baseado em código aberto. Isso significa que o código-fonte do robô está disponível publicamente, permitindo que outros programadores e estudantes explorem, modifiquem e partilhem o código livremente.

Além disso, nosso robô é expansível. Projetou-se o sistema de forma modular, permitindo que outros componentes e módulos sejam facilmente integrados. Isso oferece flexibilidade aos utilizadores para adaptar e estender as funcionalidades do robô de acordo com as suas necessidades específicas. A capacidade de expansão torna o robô aqui apresentado uma plataforma versátil que pode ser usada numa ampla variedade de projetos e aplicações.

A Tabela 2.1 apresenta um sumário relativamente às características dos trabalhos relacionados e do robô didático.

Tabela 2.1: Comparação entre Trabalhos Relacionados

| | Lego Mindstorms | Robô Autônomo | Robô Didático |
|----------------|-------------------|---------------|---------------|
| Custo | Alto | Baixo | Baixo |
| Código aberto | Não | Sim | Sim |
| Conectividade | Bluetooth | Wi-fi | Bluetooth |
| Escalabilidade | Não | Sim | Sim |
| Plataforma | Intelligent Brick | ESP8266 | ESP32 |
| Linguagem | Java | C++ | C++ |

Modelo Proposto

A secção do modelo proposto começa por apresentar os requisitos, que englobam os requisitos funcionais, não funcionais e casos de utilização (Secção 3.1). De seguida, os fundamentos de alguns conceitos teóricos e tecnológicos referidos ao longo do relatório (Secção 3.2).

3.1 Requisitos

De forma a apresentar o modelo proposto primeiramente são listados os requisitos do projeto a desenvolver, as especificações técnicas e funcionais do robô didático, entre outras palavras, o que o sistema deve conseguir realizar:

- Movimentação retilínea para a frente/trás;
- Movimentação curvilínea para a direita/esquerda;
- Comunicação a partir de uma rede sem fios;
- Capacidade de leitura de dados dos sensores.

E os requisitos não funcionais, que tratam-se de critérios ou atributos que descrevem como o sistema deve operar, em vez de se concentrarem no que o sistema deve realizar. A lista de requisitos não funcionais, todos obrigatórios, no projeto são:

- Ser seguro e confiável de forma a evitar danos materiais;
- Ter uma montagem e desmontagem fácil, a partir de uma PCB que interliga todos os componentes eletrónicos;
- Ser expansível, com uma estrutura e código genérico que permita a adição de mais componentes.

Os casos de utilização são uma técnica de modelagem que descreve interações funcionais entre os utilizadores (atores) e o sistema. Descrevem como os utilizadores interagem com o sistema em situações específicas para alcançar um determinado objetivo. Em outras palavras, os casos de utilização fornecem uma representação visual das principais funcionalidades do sistema do ponto de vista do utilizador.

O diagrama dos casos de utilização encontra-se apresentado na Figura 3.1. No diagrama, o ator (sistema de controlo remoto) é representado como figura externa ao sistema, e as setas indicam as interações entre o ator e os diferentes casos de utilização.

- **Enviar comando para movimento retilíneo:** O utilizador utiliza o sistema de controlo remoto para enviar um comando ao robô, instruindo-o a mover-se em linha reta.
- **Enviar comando para virar:** O utilizador utiliza o sistema de controlo remoto para enviar um comando ao robô, instruindo-o a efetuar uma curva.
- **Enviar comando para parar:** O utilizador utiliza o sistema de controlo remoto para enviar um comando ao robô, instruindo-o a parar o movimento.
- **Ler dados do sensor de distância:** O sistema de controlo remoto lê os dados do sensor de distância do robô, permitindo ao utilizador obter informações sobre a distância entre o robô e um objeto próximo.
- **Ler dados do sensor de toque:** O sistema de controlo remoto lê os dados do sensor de toque do robô, permitindo ao utilizador saber se o robô está em contacto com algum objeto ou superfície.

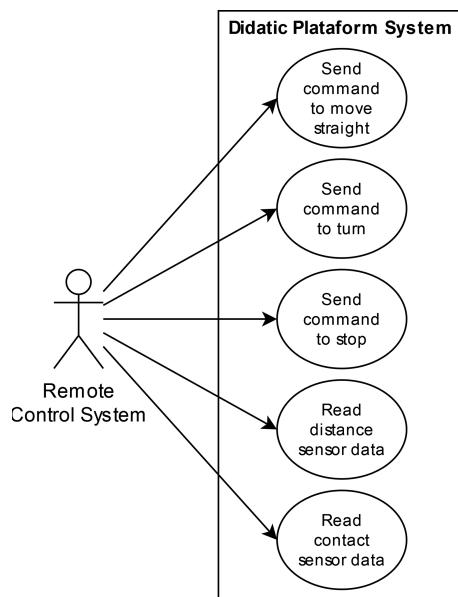


Figura 3.1: Casos de utilização

3.2 Fundamentos

De modo a facilitar a compreensão do projeto desenvolvido, é necessário explicitar detalhadamente alguns conceitos teóricos e tecnológicos que são abordados ao longo do projeto.

O Bluetooth é uma tecnologia de comunicação sem fio projetada para conectar dispositivos eletrónicos de curto alcance.

O protocolo *Inter-Integrated Circuit* (I2C) é um protocolo de comunicação serial utilizado para interligar dispositivos eletrónicos.

O *Pulse Width Modulation* (PWM) é uma técnica utilizada para controlar a quantidade média de energia entregue a um dispositivo. É comum ser usado em sistemas de controlo de velocidade de motores, controlo de luminosidade em LEDs, controlo de potência em fontes de alimentação e outras aplicações.

Uma PCB é uma placa plana, geralmente feita de material isolante, que contém trilhas condutoras e *pads* (áreas metálicas para soldagem de componentes eletrónicos). É usada para interligar e suportar componentes eletrónicos em dispositivos e sistemas.

Arquitetura do Sistema

O capítulo da arquitetura de sistema descreve a organização e os componentes da plataforma didática desenvolvida. Descreve-se em primeiro a estrutura da plataforma (Secção 4.1). De seguida aborda-se a seleção dos motores e alimentação do sistema (Secção 4.2). O controlador do motor, dispositivo responsável por controlar o funcionamento do motor é descrito na Secção 4.3. Os diferentes sensores presentes no sistema são analisados na Secção 4.4. Por fim o microcontrolador utilizado na Secção 4.5.

4.1 Estrutura

Começou-se por selecionar a estrutura para o protótipo do robô, a escolha do *chassi*, responsável por sustentar os componentes. Com o intuito de simplificar o processo inicial, optou-se por escolher o *chassi*, cujas especificações técnicas encontram-se representadas na Tabela 4.1, o qual possui suporte para duas rodas de tração dianteiras e uma roda de apoio na parte traseira.

Tabela 4.1: Especificações técnicas do *chassi* – c/ motores e rodas [3]

| Rodas | Rodas de apoio | Diâmetro das rodas [cm] | Peso [g] | Preço [€] |
|-------|----------------|-------------------------|----------|-----------|
| 2 | 1 | 6,5 | | 8,91 |

O *chassi* selecionado oferece uma base estável e robusta para a montagem dos componentes do robô. A sua configuração com duas rodas de tração proporciona uma adequada tração e mobilidade, enquanto a roda de apoio traseira contribui para a estabilidade do robô durante o movimento. Um dos objetivos futuros consiste na implementação de um protótipo com quatro rodas motrizes. O *chassi* selecionado para o protótipo do robô está representado na Figura 4.1.



Figura 4.1: Chassi [3]

4.2 Motor e Alimentação

Anteriormente descreveu-se que o robô é composto por duas rodas, as mesmas são acionadas por dois motores DC instalados na estrutura. É importante ressaltar que existem diversos tipos de motores DC, cada um com características específicas. Assim, inicialmente foi realizado um pequeno teste para determinar a corrente mínima necessária para iniciar o funcionamento dos motores. O motor DC utilizado, juntamente com a roda, encontra-se apresentado na Figura 4.2.



Figura 4.2: Roda e motor DC [4]

A seguir, apresentam-se as especificações técnicas do motor selecionado para o robô, conforme descritas na Tabela 4.2:

Tabela 4.2: Especificações técnicas do motor DC [4]

| Tensão de alimentação [V] | Velocidade [RPM] | Peso [g] | Preço [€] |
|---------------------------|------------------|----------|-----------|
| 3 – 9 | 208 | 58 | 3,44 |

Inicialmente, o robô estava a ser alimentado por uma pilha de 9 V, o que parecia ser uma escolha adequada. No entanto, durante os primeiros testes, observou-

se que, ocasionalmente, os motores não arrancavam. Com base na pesquisa e testes realizados em [2] constatou-se que o problema estava relacionado ao pico de corrente exigido pelos motores simultaneamente (8 A), o qual excedia a capacidade de fornecimento da pilha (2 A).

Foi adotada uma abordagem em que se tentou acionar um motor de cada vez, com um intervalo de 50 ms entre eles, a fim de evitar uma sobrecarga na corrente da pilha. Embora tenha sido bem-sucedida esta abordagem em cumprir esse objetivo, essa implementação resultou num problema adicional. Verificou-se que esta causava um desvio significativo no arranque do robô, fazendo com que ele se desviasse consideravelmente da trajetória desejada. Diante dessa situação, foi tomada a decisão de substituir a pilha por uma bateria que possui-se uma maior capacidade e corrente. Esta nova fonte de energia foi considerada suficiente para permitir o arranque simultâneo dos dois motores.

A bateria selecionada para substituir a pilha é a Gens Ace Bateria Lipo, com uma capacidade de 2200 mAh e uma tensão nominal de 7.4 V. Essa escolha foi feita devido à sua capacidade de fornecer uma corrente adequada para o arranque simultâneo dos dois motores, garantindo assim o desempenho adequado. As especificações técnicas e custos encontram-se apresentadas na Tabela 4.3.

Tabela 4.3: Especificações técnicas da bateria Gens Ace [5]

| Tensão [V] | Capacidade [mAh] | Peso [g] | Preço [€] |
|------------|------------------|----------|-----------|
| 7,4 | 2200 | 106 | 9,90 |

Os motores utilizados no robô estão equipados com *encoders*, que desempenham um papel importante no controlo e na leitura das informações relacionadas ao movimento dos motores. Um *encoder* é um dispositivo eletrónico que converte o movimento rotacional do eixo do motor em sinais elétricos.

Um tipo comum utilizado é o *encoder* óptico incremental. Esse tipo de *encoder* consiste num disco com furos, conhecidos como pulsos, dispostos em torno do eixo do motor. Na Figura 4.3, é ilustrado o posicionamento do *encoder* em relação à roda. Quando o motor gira, uma fonte de luz, geralmente um emissor de luz infravermelha, é projetada sobre o disco. Do outro lado do disco, um sensor óptico, composto por um fototransístor ou fotodíodo, deteta a presença ou ausência da luz através dos furos do disco, gerando assim um padrão de pulsos elétricos.

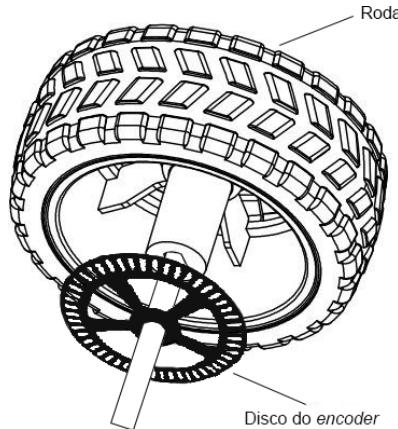


Figura 4.3: Roda com disco

No caso específico mencionado, o disco do *encoder* possui 20 furos, o que significa que a cada rotação completa do eixo do motor, serão gerados 20 pulsos elétricos. Esses pulsos são utilizados para medir a distância percorrida pelo motor. As especificações técnicas do *encoder* H206 encontram-se apresentadas na Tabela 4.4.

Tabela 4.4: Especificações técnicas do *encoder* H206 [6]

| Tensão de operação [V] | Peso [g] | Preço [€] |
|------------------------|----------|-----------|
| 3,3 – 5 | 1 | 2,25 |

4.3 Controlador do Motor

O controlador de motor é a unidade responsável pela gestão do funcionamento de um ou mais motores. Este controlador é responsável por regular diversos parâmetros, tais como velocidade, corrente elétrica, posição do rotor, entre outros, de acordo com as características específicas do controlador utilizado.

A velocidade de um motor DC pode ser manipulada através da alteração da tensão de entrada. A técnica *Pulse Width Modulation* (PWM) consiste em enviar uma série de pulsos de forma a ajustar o valor médio da tensão de entrada.

A largura dos pulsos, conhecida como *duty cycle*, é proporcional ao valor médio da tensão. Seguindo este raciocínio, reduzir o *duty cycle* resulta numa diminuição da velocidade do motor e consequentemente, ao aumentar o mesmo resulta num aumento da velocidade. Ao definir o *duty cycle* a 100% a tensão média ficará igual

à tensão de alimentação, fazendo com que o motor gire à velocidade máxima. Já com *duty cycle* a 0%, a tensão média ficaria igual a 0V, e consequentemente, o motor permanecerá parado.

De seguida encontra-se representado a formula matemática do *duty cycle*, com o tempo de pulsos ativos no valor lógico "1" t_{on} , o tempo de pulsos no valor lógico "0" t_{off} e período T . A Figura 4.4 representa um *duty cycle* a 50%, em que a linha tracejada consiste no valor de tensão média.

$$\text{duty cycle} = \frac{t_{on}}{T} \cdot 100\% \quad (4.1)$$

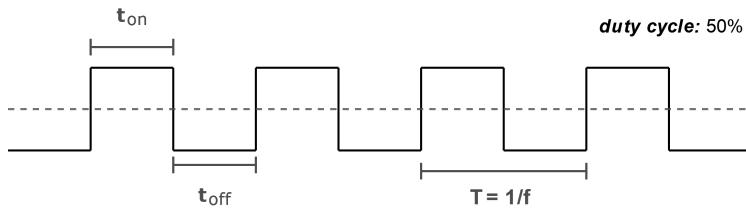


Figura 4.4: *Duty cycle* a 50%

Relativamente à alteração do sentido de rotação do motor é necessário introduzir uma outra técnica, cujo designação é ponte H. A ponte H, de uma forma geral, trata-se de um circuito eletrónico composto por quatro transístores interligados, numa disposição semelhante à letra H, de forma a permitir a inversão do sentido da corrente que alimenta o motor.

Ao ligar uma bateria a um motor, este rodará apenas num único sentido, posto isto, o circuito da ponte H é construído de forma a que quando dois transístores estão fechados, a corrente elétrica flui do terminal positivo da fonte de alimentação para o motor no sentido horário, e quando os outros dois transístores são fechados, a corrente flui no sentido anti-horário. Dessa forma, é possível controlar o sentido de rotação do motor. Na Figura 4.5 é apresentado um esquema simplificado do funcionamento de um circuito de Ponte H, com a rotação de um motor em sentido dos ponteiros do relógio e sentido contrário aos ponteiros do relógio, demonstrando o fluxo da corrente elétrica.

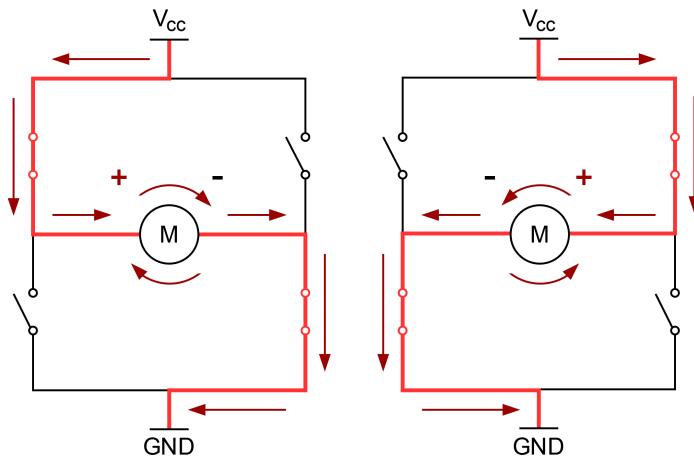


Figura 4.5: Esquema simplificado da Ponte H

O L298N DC Motor Driver é o módulo controlador de motores DC selecionado para controlar os motores do robô. O controlador em questão é um módulo de controlo bidirecional que permite acionar os dois motores DC simultaneamente. As especificações técnicas do módulo encontram-se apresentadas na Tabela 4.5.

Tabela 4.5: Especificações técnicas do controlador L298N [7]

| Tensão de operação [V] | Tensão lógica [V] | Peso [g] | Preço [€] |
|------------------------|-------------------|----------|-----------|
| 5 – 35 | 5 | 33 | 2,15 |

4.4 Sensores

Foram considerados para este protótipo dois sensores: um sensor de distância e um sensor de toque.

4.4.1 Sensor de Distância

O sensor de distância SONAR é um dispositivo que utiliza ondas sonoras para medir a distância entre o robô e um objeto próximo. Ele emite pulsos de som e mede o tempo que leva para o som refletido retornar ao sensor. Com base no tempo de retorno, é possível calcular a distância até o objeto.

As especificações técnicas do sensor de distância HC-SR04 utilizado no sistema estão apresentadas na Tabela 4.6.

Tabela 4.6: Especificações técnicas do sensor de distância SONAR [8]

| Tensão de alimentação [V] | Frequência ultrassónica [kHz] | Alcance [cm] | Preço [€] |
|---------------------------|-------------------------------|--------------|-----------|
| 5 | 40 | 3 – 400 | 2,25 |

4.4.2 Sensor de Toque

O sensor de toque é um dispositivo utilizado para detetar a interação física entre o robô e objetos ou superfícies. Ele possui um mecanismo sensível ao toque que é ativado quando há contacto físico. O sensor de toque é utilizado para detetar colisões ou interações com objetos externos.

As especificações técnicas do sensor de toque utilizado no sistema estão apresentadas na Tabela 4.7.

Tabela 4.7: Especificações técnicas do sensor de toque [9]

| Tensão de alimentação [V] | Durabilidade Mecânica [Ciclos] | Resistência máxima de contacto [cm] | Preço [€] |
|---------------------------|--------------------------------|-------------------------------------|-----------|
| 30 DC - 250 AC | 40 | 30 | 3,46 |

4.5 Microcontrolador

O microcontrolador é um componente eletrónico integrado num único chip que combina elementos de um microprocessador com periféricos, memória e interfaces de entrada/saída. O seu objetivo é executar tarefas específicas em sistemas embarcados, como controlo de dispositivos, recolha de dados, processamento de sinais, comunicação, entre outras funções. O diagrama simplificado do funcionamento de um microcontrolador de forma abstrata encontra-se apresentado na Figura 4.6.

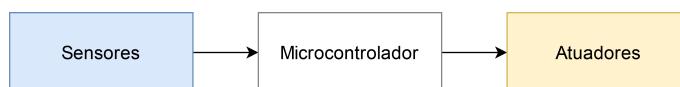


Figura 4.6: Diagrama simplificado do funcionamento de microcontrolador

O microcontrolador desempenha um papel fundamental no controlo e comportamento do robô. É responsável por executar o código e as instruções que controlam o

funcionamento do robô. Por meio do microcontrolador, é possível processar os dados dos sensores, tomar decisões e enviar comandos para os atuadores, permitindo assim a interação e movimentação do robô.

O ESP32 WROOM-32 DEVKIT V1 é um exemplo de microcontrolador, essa variante possui 320 Kilobyte (10^3) (KB) de *Random Access Memory* (RAM) e 4 Megabyte (10^6) (MB) de memória flash, usada para armazenar o programa que será executado.

Foi selecionado o ESP32 para microcontrolador do sistema devido a uma ampla variedade de periféricos integrados, como as interfaces *Wireless Fidelity* (Wi-Fi), Bluetooth, pinos *General Purpose Input/Output* (GPIO) e I2C. A interface Bluetooth possibilita a comunicação entre um dispositivo, como um computador ou telemóvel, e o robô, permitindo o envio dos comandos de movimento para o mesmo. Para além das especificações técnicas a escolha deste modelo é o fator económico, sendo uma solução de baixo custo atualmente.

Tabela 4.8: Especificações técnicas do microcontrolador ESP32 [10]

| Tensão de alimentação [V] | Tensão de operação [V] | Peso [g] | Preço [€] |
|---------------------------|------------------------|----------|-----------|
| 5 | 3,3 | 13 | 15,35 |

Implementação

Este capítulo tem como objetivo apresentar o processo de implementação do projeto, desde a montagem do protótipo (secção 5.1), programação do sistema (secção 5.2) e construção da PCB (secção 5.3).

A Figura representada na referência 5.1 ilustra um diagrama que apresenta a totalidade do sistema a ser implementado. No *software*, os blocos indicados correspondem aos diversos módulos do sistema, enquanto na parte do *hardware*, os blocos se referem aos diferentes componentes eletrónicos. O microcontrolador é responsável por interligar estas duas componentes.

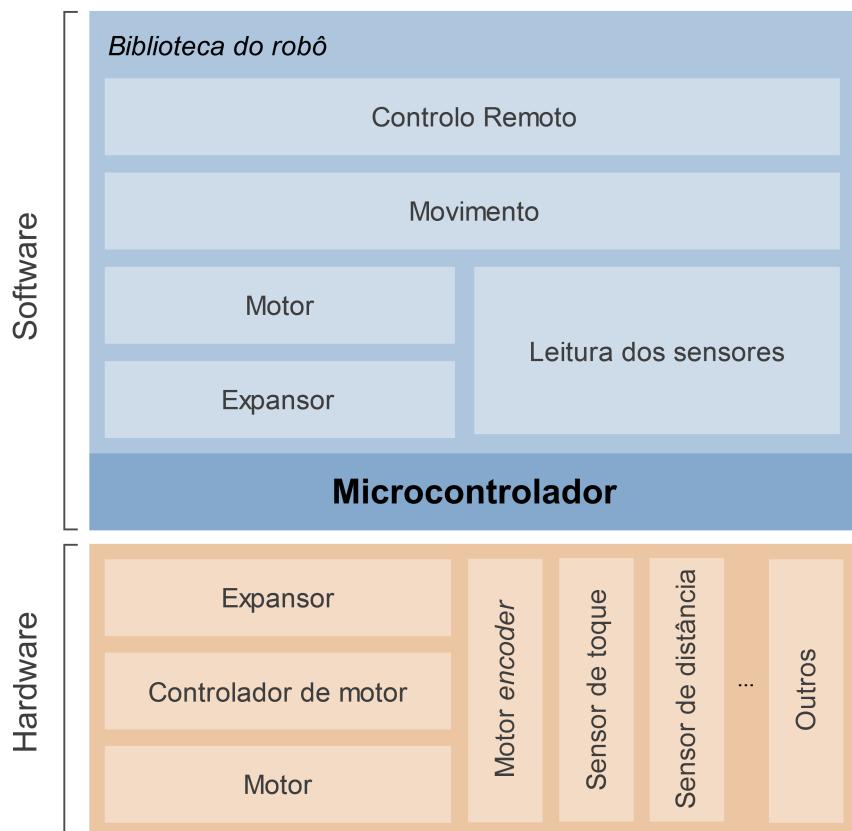


Figura 5.1: Diagrama do sistema em blocos

5.1 Montagem

Para começar, é importante salientar que a montagem da plataforma didática é a primeira etapa do projeto e requer a habilidade técnica e o conhecimento específico sobre os componentes do robô. Nesta etapa, recomenda-se que tenha lido rigorosamente os detalhes destacados de cada componente no capítulo 4.

Deu-se início à montagem dos componentes no *chassi*, começando pela fixação das rodas juntamente com os motores correspondentes à estrutura. Em seguida, os *encoders* foram montados nos respectivos orifícios, proporcionando a leitura precisa do movimento das rodas. O aspetto da plataforma nesta fase encontra-se apresentado na Figura 5.2.

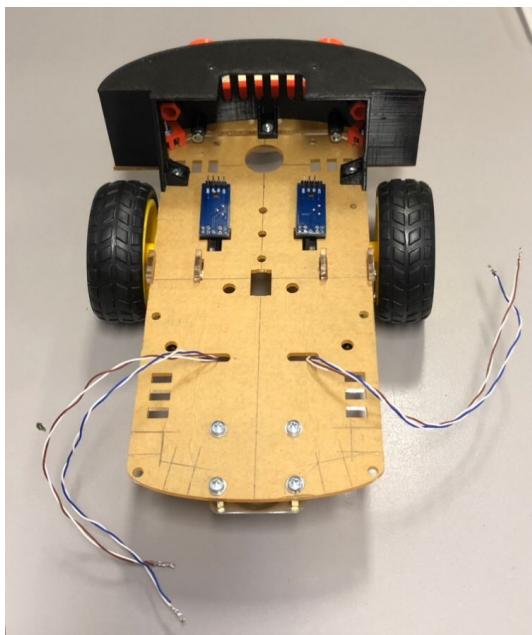


Figura 5.2: Montagem inicial

Em seguida, o próximo passo é a montagem do microcontrolador, controlador do motor, expansor e sensores. Para isso, é necessário planejar e pesquisar como esses componentes devem ser conectados para um funcionamento correto e integrado.

O controlador do motor desempenha um papel importante na alimentação do sistema. Ele é responsável por receber a carga total da bateria, pois possui uma entrada de alimentação que suporta uma faixa de tensão entre 5V e 35V. Além disso, tem uma saída de fornecimento de energia de 5V, que será utilizada para alimentar o microcontrolador. O interruptor deve ser introduzido entre o controlador do

motor e a bateria para controlar o fluxo de energia e ligar/desligar todo o sistema.

Os restantes componentes, como os *encoders* e o expansor, são alimentados pelo microcontrolador. O microcontrolador possui duas saídas de fornecimento de energia com diferentes níveis de tensão, sendo uma saída de 3,3V e outra com 5V.

No caso dos *encoders* e do expansor, eles são alimentados com uma tensão de 3,3V, que é fornecida diretamente pelo microcontrolador. Essa tensão é adequada para o funcionamento correto desses componentes, garantindo seu desempenho.

Foi elaborado um diagrama ilustrativo que representa a conexão dos componentes do sistema. Esse diagrama tem como objetivo visualizar de forma clara e organizada como os componentes serão interligados, apresentado na Figura 5.3.

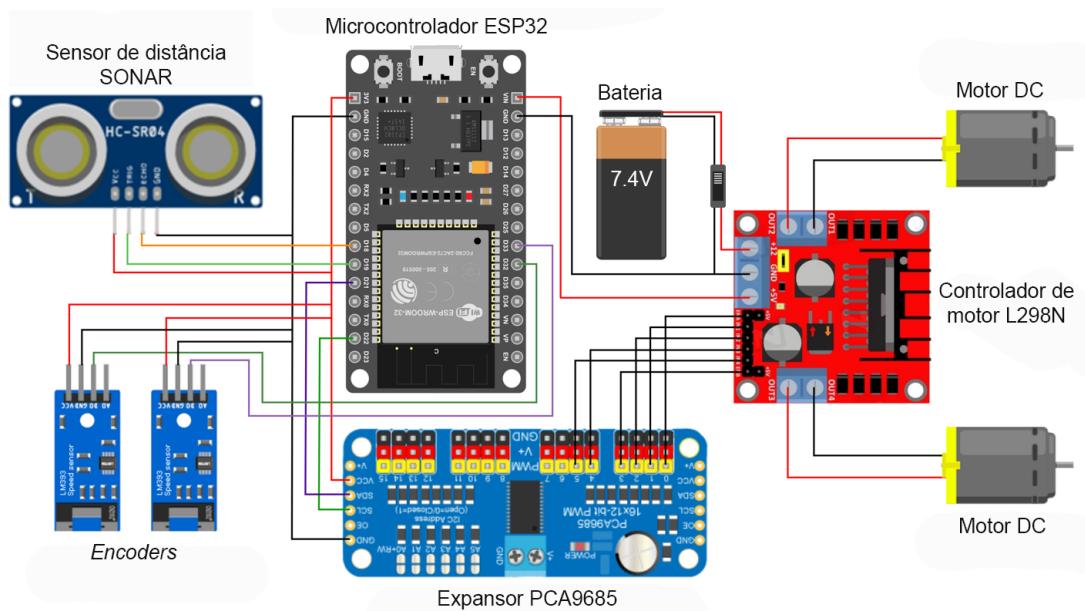


Figura 5.3: Diagrama ilustrativo do circuito

Cumpre salientar que, no lado esquerdo do esquema, encontram-se os sensores. Neste caso específico, foi adicionado um sensor de distância SONAR, no entanto, é possível adicionar outros sensores, como sensores de toque, conforme a preferência do utilizador. Por outro lado, os demais componentes são indispensáveis para o adequado funcionamento. É importante notar que a ausência de *encoders* no sistema compromete a capacidade de garantir o correto funcionamento dos motores durante a marcha.

Procedeu-se à implementação do esquema elétrico, isso envolveu a criação de um diagrama mais detalhado que representa a interconexão dos componentes e o fluxo de energia no sistema. O esquema encontra-se apresentado na Figura 5.4.

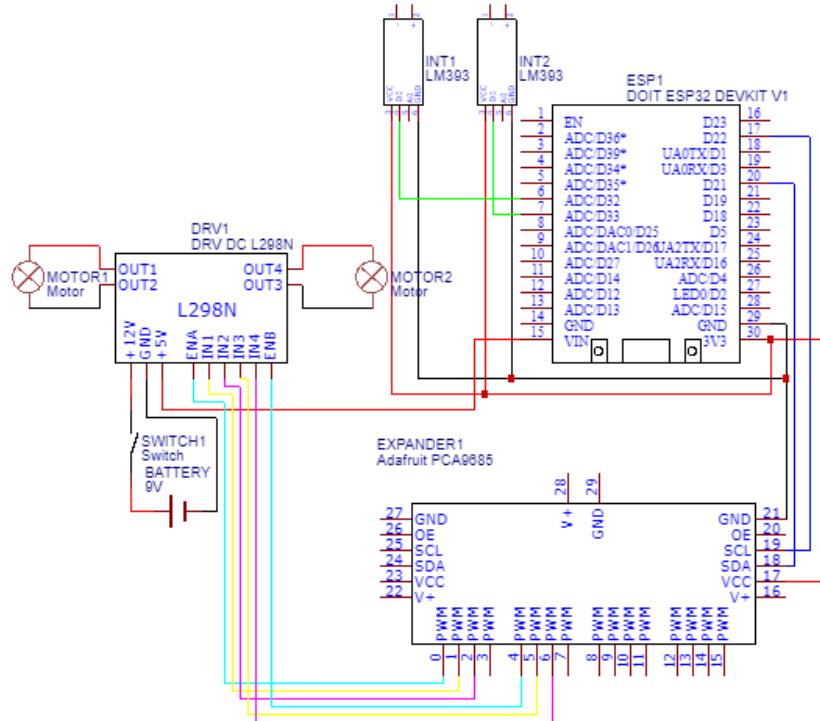


Figura 5.4: Esquema elétrico

Após planear as conexões entre os componentes e o devido esquema elétrico, soldaram-se os fios numa placa de circuito. Os demais componentes foram montados, finalizando a fase de montagem do projeto.

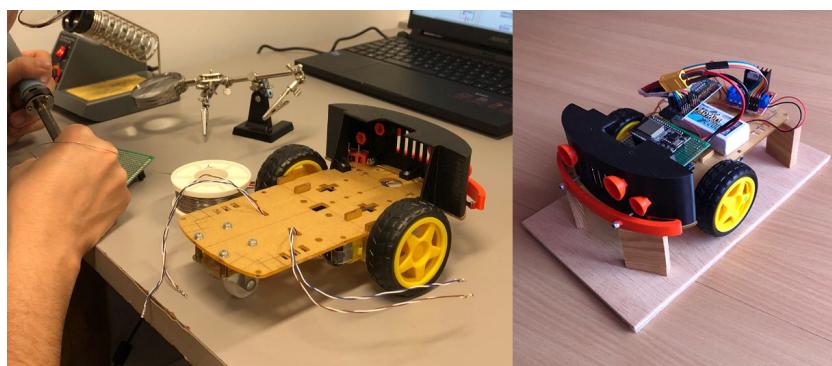


Figura 5.5: Soldagem e finalização da montagem

5.2 Programação

No âmbito da programação do sistema, foi elaborado o diagrama de classes utilizando a linguagem UML. O diagrama de classes aqui apresentado descreve as principais classes do sistema em questão.

A classe Expander representa um dispositivo de expansão, esta classe é responsável por manipular e enviar os sinais para um determinado pino do expensor, via PWM.

A classe Motor representa um motor físico no sistema. Esta classe recebe uma estrutura MotorController que descreve as características do motor. A classe Motor também recebe uma instância de Expander uma vez que o motor encontra-se ligado aos pinos do respetivo expensor.

A classe abstrata Movement representa um movimento genérico no sistema. Esta classe recebe uma lista de motores (classe Motor) e define métodos abstratos para os diferentes tipos de movimento a atuar em cada um dos motores. Nesta classe é criado a estrutura Data de forma a armazenar dados auxiliares a cálculos relacionados com o movimento.

A classe MovementTwoMotors é uma subclasse de Movement e representa um movimento que envolve dois motores. Essa classe implementa os métodos abstratos da classe pai, fornecendo implementações específicas para alguns tipos de movimento.

A classe Robot representa o robô em si, é responsável por iniciar a comunicação e processamento de comandos recebidos, via Bluetooth. Recebe uma instância de uma classe que estende de Movement, visto que cada comando definido no protocolo de comunicação está relacionado com um tipo de movimento definido na classe Movement.

Na Figura 5.6 está então representado este UML numa forma mais simplificada de modo a ser mais simples de ler. No entanto, em caso de ser necessário visualizar mais algum detalhe adicional, o diagrama de classes completo encontra-se disponível na Figura B.1 do anexo B.

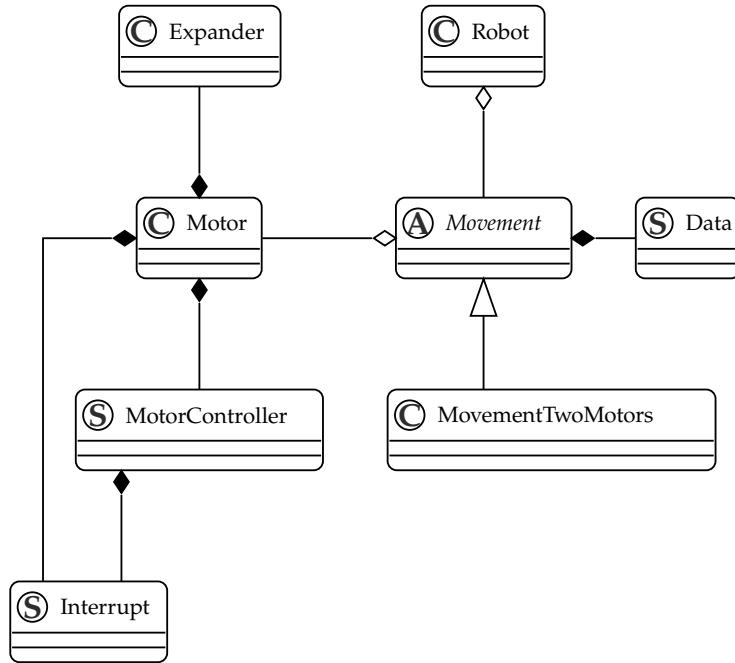


Figura 5.6: Diagrama de classes UML simplificado

Iniciou-se a implementação dos módulos, previamente apresentados na introdução desta secção, começando pelos mais específicos.

Expansor

Classe: Expander(byte address)

Primeiramente, deu-se início ao desenvolvimento do módulo do expansor do sistema. O expansor estabelece comunicação com o microcontrolador por meio do protocolo I2C, nesse sentido, para iniciar a comunicação é necessário fornecer o endereço do expansor, logo o mesmo é fornecido no construtor da classe Expander. As funções da classe baseiam-se na manipulação dos sinais nos pinos do expansor, especificamente as seguintes funcionalidades:

- Definir o PWM num certo pino;
- Definir o duty cycle num certo pino;

Em que os valores do PWM variam entre 0 e 4095, representando a amplitude do sinal modulado. Por outro lado, o duty cycle é expresso em percentagem, variando de 0% a 100%, cujo valores são mapeados para os valores correspondentes do PWM.

Motor

Classe: `Motor(Expander expander, MotorController controller)`

De seguida optou-se pelo desenvolvimento do módulo responsável pelo controlo do funcionamento do motor. No construtor desse objeto, é necessário fazer referência ao expansor correspondente, uma vez que os motores estão conectados ao controlador, que por sua vez está conectado ao expansor. Além da referência do expansor, é necessário fornecer a identificação dos pinos do expansor e do *encoder*, número de furos do disco do *encoder*, bem como o raio da roda, a fim de calcular o perímetro da mesma. Esses dados são enviados dentro de uma struct chamada `MotorController`. As principais funções desta classe baseiam-se em:

- Iniciar a marcha;
- Parar a marcha;
- Definir a direção (sentido dos ponteiros do relógio/contra-relógio);
- Definir a velocidade inicial v_0 ;
- Definir a distância d ;
- Obter o número de ticks lido;
- Obter o número de ticks pretendido.

Antes de implementar as funções do motor, é fundamental compreender o funcionamento dos três pinos do controlador: ENA, IN1 e IN2. Esses pinos desempenham papéis específicos no controlo do motor e as suas características são as seguintes:

ENA: Entrada PWM utilizada para controlar a velocidade do motor. Através desse pino, é possível enviar um sinal PWM para ajustar a amplitude do sinal de acionamento do motor, o que afeta diretamente a velocidade de rotação;

IN1/IN2: Estes pinos recebem um sinal digital e é usado para controlar a direção do motor. Dependendo do valor lógico (alto ou baixo) aplicado a esse pino, o motor pode ser acionado numa direção específica.

Tabela 5.1: Valores lógicos para controlo

| Estado | IN1 | IN2 |
|-----------------|-----|-----|
| Motor desligado | 0 | 0 |
| Em frente | 1 | 0 |
| Em marcha-atrás | 0 | 1 |
| Motor travado | 1 | 1 |

De acordo com a documentação do controlador, os valores lógicos a serem enviados para os pinos do controlador, que também foram utilizados nas funções correspondentes, estão apresentados na Tabela 5.1.

Outra funcionalidade é a capacidade de determinar o número de ticks que correspondem à distância pedida a percorrer. Para um determinado comprimento x em centímetros fornecido pelo utilizador, em que p corresponde ao perímetro da roda e a constante N ao número de furos do *encoder*, a expressão matemática utilizada para calcular o número de ticks t é a seguinte:

$$t = \frac{x}{p} \cdot N \quad (5.1)$$

Movimento

Classe: Movement(Motor *motors, float track)

O módulo do movimento, que abrange os módulos mencionados anteriormente, é responsável pela deslocação do robô, exercendo controlo sobre os motores para executar o trajeto desejado. A classe Movement trata-se de uma classe abstrata projetada para atuar como base de uma classe desenvolvida pelo utilizador, a fim de adaptar o módulo de movimento conforme os componentes do robô.

A classe Movement recebe o conjunto de motores e o track que consiste no comprimento entre as rodas, em centímetros, e cujo principais funcionalidades são as seguintes:

- Movimentar em linha reta, para a frente/trás, com velocidade inicial v_0 , até a uma distância de x cm;
- Movimentar em curva para a esquerda/direita, com velocidade inicial v_0 , raio r até ao ângulo α ;
- Compensar a direção;

No que diz respeito ao movimento em linha reta, o qual possui uma implementação inicial na classe, todos os motores da plataforma são acionados para mover-se para a frente/trás, tornando a função genérica e independente do número de motores. A movimentação em curva e a compensação da direção depende do número de motores instalados, nesse sentido, nesta classe essas funcionalidades encontram-se por implementar pelo utilizador.

Desta forma, foi criada a classe MovementTwoMotors com o propósito de implementar uma classe específica para o protótipo de duas rodas desenvolvido ao longo deste projeto. A mesma estende da classe Movement.

Classe: MovementTwoMotors(Motor *motors, float track) : Movement(
motors, track)

Nesta classe começou-se por implementar o sistema de compensação de direção ao movimentar-se em linha reta, para garantir que o robô se movimentasse numa trajetória correta, mantendo sua orientação ao longo do percurso. Todo o processo envolveu uma série de testes práticos que foram documentados na subsecção 6.2 da secção testes práticos.

Visto que o robô tem que ser capaz de realizar curvas para a esquerda e para a direita com um raio, ângulo e velocidade específicos. Para isso, é necessário calcular a velocidade correspondente a cada um dos motores.

Ao percorrer uma curva completa, ou seja, com um ângulo de 360° , o robô percorre toda a circunferência de raio r . Portanto, a distância total percorrida na curva é igual ao perímetro da circunferência, que é dado pela fórmula $2\pi r$. Considerando que o robô deve ser programado para parar num ângulo específico, representado por α , é essencial medir a distância até a esse ângulo. Nesse sentido, para calcular a distância que o robô deve percorrer até ao ângulo α , pode-se utilizar a regra de três simples. Sendo d a variável que representa essa distância, temos:

$2\pi r$ representa o perímetro da circunferência completa, correspondente a 360° . Assim, temos a seguinte proporção:

$$2\pi r \rightarrow 360^\circ d \rightarrow \alpha \quad (5.2)$$

Utilizando a regra de três simples, encontra-se o valor de d em função de α :

$$d = \frac{2\pi r \alpha}{360} \quad (5.3)$$

A velocidade v_0 especificada pelo utilizador corresponde à velocidade do centro do *chassi*. A partir da velocidade v_0 , é possível calcular o tempo t necessário para que o robô percorra a distância d . É importante destacar que o tempo para percorrer essa distância a partir do centro do *chassi* será igual para cada um dos motores do robô, visto que as rodas chegarão simultaneamente ao ponto de destino.

Considerando que a velocidade v representa a velocidade do robô no centro do *chassi*, como se fosse uma roda imaginária, é possível determinar o tempo necessário para que ele percorra a distância d utilizando a fórmula da velocidade. Essa fórmula estabelece uma relação entre a velocidade, a distância percorrida e o tempo de percurso. Isolando a variável tempo t na equação, podemos calcular o tempo necessário para que o robô alcance o ponto de destino.

$$v = \frac{d}{t} \Leftrightarrow t = \frac{d}{v} \quad (5.4)$$

Conhecer o tempo t permite determinar a velocidade ideal de cada um dos motores para o robô percorrer a curva pretendida. A Figura 5.7 apresenta um diagrama representativo de uma curva à esquerda do robô, com o respetivo raio r que representa o comprimento da circunferência até ao centro do *track* do *chassi*. É importante observar que o raio para o motor interno será igual a r subtraído à metade do comprimento do *track* do *chassi*. Já no motor exterior, o raio será igual a r somado à metade do comprimento do *track*.

Além disso, a Figura mostra o ângulo α , a distância d , a velocidade v , e os próximos valores a serem encontrados, a distância desde a roda interior e exterior até ao ponto de destino, variáveis d_{in} e d_{out} , e os valores de velocidade de cada um dos motores, variáveis v_{in} e v_{out} .

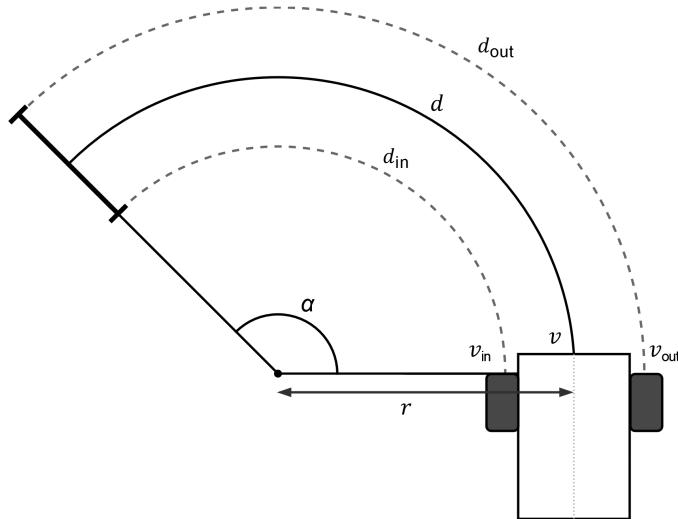


Figura 5.7: Diagrama de uma curva à esquerda

Nesta fase é calculado a distância desde a posição de cada um dos motores até ao ponto de destino. Para calcular os valores de distância, variáveis d_{in} e d_{out} , tendo em conta a variável Δ , que corresponde a metade do comprimento do *track*:

$$d_{in} = \frac{2\pi(r - \Delta)\alpha}{360} \quad (5.5)$$

$$d_{out} = \frac{2\pi(r + \Delta)\alpha}{360} \quad (5.6)$$

Observa-se que foi seguido o raciocínio anterior para o cálculo das distâncias, em que os valores são obtidos por meio da regra de três simples.

A partir da fórmula da velocidade, foi calculada a velocidade para cada motor, começando pela velocidade para o motor interno à curva, em que as variáveis v_0 e d_0 consistem na velocidade e distância respetivamente, calculadas anteriormente para o centro do *track* do *chassi*:

$$\begin{aligned} v_{in} &= \frac{d_{in}}{t} \Leftrightarrow \\ \Leftrightarrow v_{in} &= \frac{2\pi(r - \Delta)\alpha}{360} \cdot \frac{v_0}{d_0} \Leftrightarrow \\ \Leftrightarrow v_{in} &= \frac{2\pi(r - \Delta)\alpha}{360} \cdot \frac{360}{2\pi r \alpha} \cdot v_0 \Leftrightarrow \end{aligned} \quad (5.7)$$

Nesta fase é possível simplificar a equação, cortando os valores que aparecem tanto no numerador quanto no denominador:

$$\begin{aligned} \Leftrightarrow v_{in} &= \frac{2\pi(r - \Delta)\alpha}{360} \cdot \frac{360}{2\pi r \alpha} \cdot v_0 \Leftrightarrow \\ \Leftrightarrow v_{in} &= \frac{r - \Delta}{r} \cdot v_0 \end{aligned} \quad (5.8)$$

Ao observar a equação simplificada da velocidade do motor interno, podemos concluir facilmente que a equação correspondente para a velocidade do motor externo é dada por:

$$\Leftrightarrow v_{out} = \frac{r + \Delta}{r} \cdot v_0 \quad (5.9)$$

A partir destes cálculos foi possível determinar a velocidade e distância que cada motor deve ter definido para realizar a curva pedida. Desta forma apenas foi chamado as respetivas funções para definir os parâmetros corretos nos motores.

Controlo remoto

Após a implementação das retas e curvas, foi necessário desenvolver um sistema que permitisse o controlo remoto desses movimentos utilizando a função Bluetooth do ESP32. A utilização do *BluetoothSerial*, uma biblioteca específica para comunicação Bluetooth, proporcionou a capacidade de estabelecer uma conexão sem fio entre o ESP32 e outros dispositivos compatíveis com Bluetooth, como *smartphones*, *tablets* ou computadores.

A plataforma foi desenvolvida com uma classe que recebe mensagens no formato de string, utilizando a biblioteca *BluetoothSerial*. Esta classe permite a

recepção e interpretação das mensagens enviadas pelo dispositivo de controlo remoto. As mensagens são formatadas com base num protocolo específico, onde o primeiro carácter indica o tipo de movimento desejado, seguido pelos parâmetros necessários para definir completamente o movimento. Por exemplo, uma mensagem para realizar uma linha reta para frente pode ser enviada no formato "f,velocidade,comprimento". Para um movimento de linha reta para trás, basta substituir o primeiro carácter por b (de *back*). O mesmo princípio aplica-se às curvas, onde o comando deve ser formatado como "direção da curva [l ou r],raio,ângulo".

Com a conexão Bluetooth estabelecida e as mensagens interpretadas corretamente, tornou-se possível controlar remotamente os movimentos do sistema no espaço 2D. Essa abordagem oferece uma ampla variedade de movimentos possíveis e proporciona flexibilidade para explorar diferentes trajetórias.

5.3 Construção da placa de circuito impresso

Com o objetivo de fortalecer a base da plataforma didática, foi desenvolvida uma PCB personalizada nas instalações da instituição de ensino.

O processo teve início com a conceção do circuito da placa utilizando o software EasyEDA, que permitiu a criação de um esquema detalhado (anexo A). Em seguida, o *layout* da PCB foi preparado, aproveitando principalmente a funcionalidade de roteamento automático do software. Pequenos ajustes foram realizados na versão final do *layout*, como o aumento do tamanho das pistas para facilitar a impressão. Ao finalizar essa etapa, exportou-se o *layout* para um acetato (Figura 5.8), que foi utilizado posteriormente no processo de impressão da placa.

A fabricação da placa ocorreu a partir de uma combinação de cobre e fibra de vidro, com a aplicação de um verniz positivo sensível à luz (positivo 20). Esse verniz tem a capacidade de endurecer quando exposto à luz ultravioleta (UV). A placa foi então cortada nas dimensões desejadas, preparando-a para a etapa de impressão.

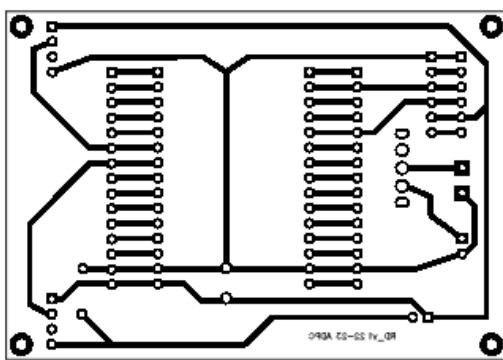


Figura 5.8: Layout da PCB

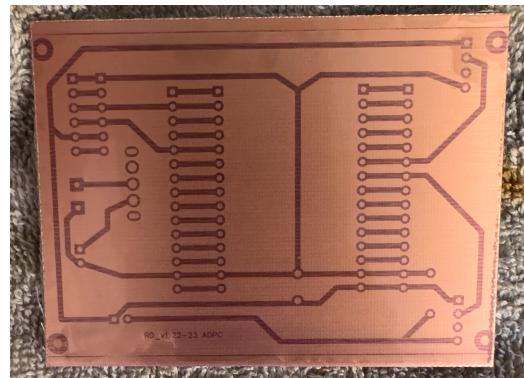


Figura 5.9: Revelação das pistas

O processo de impressão foi conduzido utilizando técnicas diversas. Primeiramente, a placa foi exposta à luz UV, com o acetato do layout posicionado sobre ela. Essa exposição permitiu revelar o circuito de forma precisa. Em seguida, foi aplicada uma solução reveladora composta por soda caustica e água, que removeu facilmente o verniz positivo das pistas, revelando-as nitidamente, como se consegue perceber pela Figura 5.9.

Após a etapa de revelação, o cobre que não estava protegido pelo verniz positivo foi removido através de um banho de corrosão com percloro de ferro. Essa ação garantiu a formação das pistas desejadas, mantendo-as isoladas e protegidas.

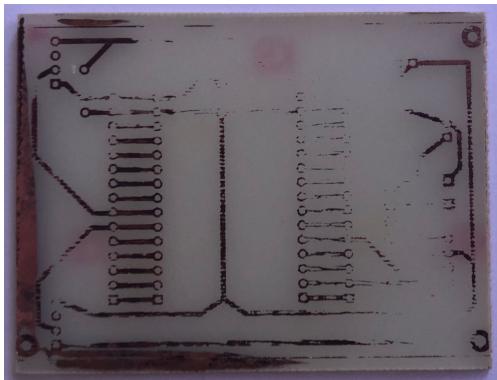


Figura 5.10: Primeira iteração

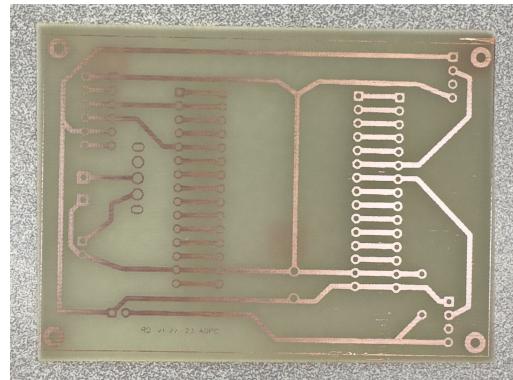


Figura 5.11: Impressão final

Durante a primeira iteração desse processo, ocorreram dificuldades, levando à necessidade de refazer todas as etapas com uma nova placa de cobre (Figura 5.10). Já na segunda iteração, as etapas foram concluídas com sucesso, e em seguida, a placa foi devidamente limpa utilizando álcool para eliminar qualquer resíduo indesejado, o final deste processo encontra-se representado pela Figura 5.11.

Por fim, utilizando uma broca específica para PCB, foram realizados os furos necessários na placa, que serão posteriormente utilizados para a instalação dos componentes eletrônicos (Figura 5.13).

O último passo envolveu a soldagem dos componentes eletrônicos na PCB e a sua montagem na plataforma didática, resultando no alcance dos objetivos previstos para o projeto.

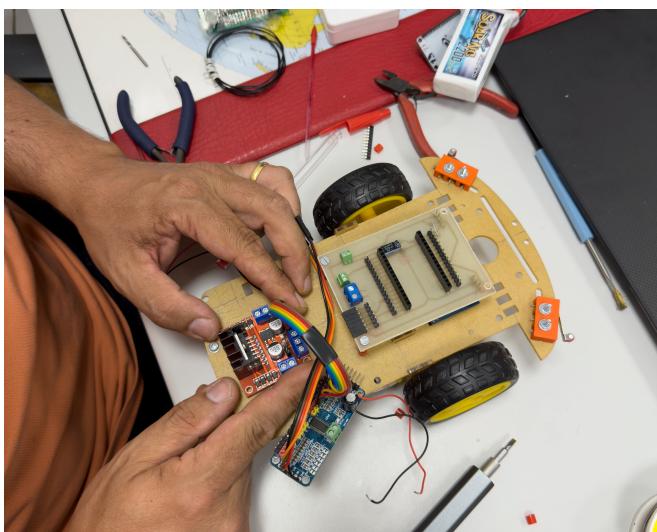


Figura 5.12: Montagem da PCB



Figura 5.13: Furação da PCB

Validação e Testes

Com o intuito de adquirir conhecimento sobre as tecnologias envolvidas e também resolução de alguns problemas encontrados, foram realizados os seguintes testes práticos, investigação de um desvio antes da paragem do robô (secção 6.1) e implementação de um diferencial por *software* (secção 6.2).

6.1 Desvio Antes da Paragem

Verificou-se que ao término da trajetória em linha reta do robô, ocasionalmente ocorria um deslizamento das rodas do mesmo, resultando na execução de uma curva característica.

O envio de comandos de paragem dos motores não ocorrem de forma simultânea. Primeiramente, é enviado um comando para interromper a receção de corrente do motor, resultando numa desaceleração gradual. Em seguida, são enviados dois comandos adicionais para travar completamente o motor. Essa sequência de três comandos é empregada para cada motor individualmente.

Contudo, devido ao motor esquerdo ser o primeiro a receber o conjunto de comandos de paragem, há um período significativo em que o motor direito ainda permanece ativo, resultando no deslizamento do robô.

6.2 Diferencial por Software

Enquanto o robô a movia-se em linha reta, notou-se que o mesmo começava a desviar e a realizar uma curva acentuada para a direita. Esse fenómeno é conhecido como desvio de trajetória e pode ser causado por várias razões, incluindo desequilíbrios na tração das rodas do robô, inclinação da superfície em que o robô se encontra, assimetria na geometria do robô, entre outros.

Dessa forma, foi conduzido o seguinte teste prático: para cada valor de *duty cycle* de 0% a 100%, com incrementos de 10, mediu-se o número de *ticks* contados por ambos

os *encoders*. Para minimizar possíveis interferências de inclinações ou deformações na superfície, esses testes foram realizados com o robô suspenso no ar, sem contacto com o solo. Observou-se que os motores só iniciam o movimento com *duty cycle* igual ou superior a 50%. Os resultados encontram-se apresentados na Tabela 6.1 e no gráfico da Figura 6.1.

Tabela 6.1: Testes práticos com rodas no ar

| Duty cycle | Teste 1 | | Teste 2 | | Teste 3 | | Média | |
|------------|---------|------|---------|------|---------|------|--------|--------|
| | Esq. | Dir. | Esq. | Dir. | Esq. | Dir. | Esq. | Dir. |
| 50 | 586 | 555 | 585 | 561 | 583 | 557 | 584,67 | 557,67 |
| 60 | 665 | 640 | 668 | 642 | 666 | 640 | 666,33 | 640,67 |
| 70 | 725 | 697 | 723 | 699 | 720 | 696 | 722,67 | 697,33 |
| 80 | 736 | 744 | 760 | 743 | 762 | 746 | 761,67 | 744,33 |
| 90 | 791 | 782 | 793 | 784 | 790 | 783 | 791,33 | 783,00 |
| 100 | 860 | 852 | 856 | 853 | 849 | 851 | 855,00 | 852,00 |

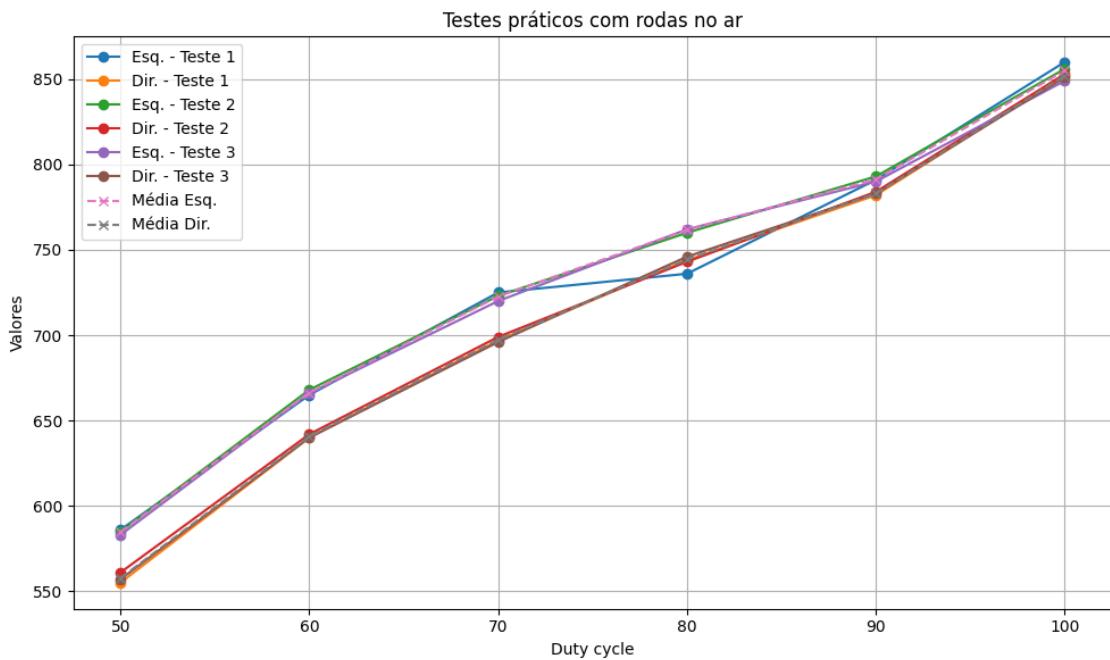


Figura 6.1: Testes práticos com rodas no ar

Conforme era esperado, os motores apresentaram uma diferença entre o número de *ticks*. Essa diferença é cada vez mais notável à medida que o *duty cycle* diminui.

Após alguma pesquisa e realização de testes práticos, concluiu-se que o motor esquerdo gira mais rápido que o motor direito. Durante a fabricação em série dos motores, apesar de cada motor passar por um processo de fabricação semelhante, pequenas variações podem ocorrer durante esse mesmo processo. Essas pequenas diferenças podem afetar a velocidade nominal de cada motor e, consequentemente, fazê-los girar em velocidades ligeiramente diferentes.

Diante disso, optou-se por implementar uma diferencial por *software*, consistindo num sistema de compensação de direção, o qual ajusta a velocidade dos motores conforme as exigências apresentadas.

O sistema funciona da seguinte maneira: armazena os valores de *ticks* obtidos por cada um dos *encoders* em intervalos de 50 milissegundos. Esse período foi estabelecido para alcançar um equilíbrio entre obter um número razoável de amostras e dispor de um tempo considerável para observar a evolução desde o último ajuste de velocidade dos motores.

A cada intervalo de amostragem de 50 milissegundos, é realizado o ajuste da velocidade dos motores, são calculadas as diferenças de *ticks* entre a amostra atual e a anterior, permitindo obter apenas os *ticks* ocorridos nos últimos 50 milissegundos.

Em seguida, a razão entre as diferenças dos *ticks* da amostra é calculada, possibilitando determinar o ajuste necessário. Se o motor esquerdo estiver mais rápido que o motor direito, o ajuste será aplicado de modo a reduzir o PWM do motor esquerdo e aumentar o PWM no motor direito. Essa estratégia visa corrigir eventuais desequilíbrios na velocidade dos motores. Todos estes passos encontram-se ilustrados no Código 1.

Código 1: Troço de código do diferencial por software

```

1 void MovementTwoMotors::directionLineCalibration()
2 {
3     unsigned long timeout = millis() + _PERIOD * ←
4         _SAMPLES_TO_SKIP;
5     unsigned long finalTime = millis() + _EXEC_TIME;
6     int currLeftCounter = _motors[MOTOR_LEFT].getCounter();
7     int currRightCounter = _motors[MOTOR_RIGHT].getCounter()←
8         ;
9     int leftTarget = _motors[MOTOR_LEFT].getTargetInterr();
10    int rightTarget = _motors[MOTOR_RIGHT].getTargetInterr()←

```

```

    ;
9     while ((currLeftCounter < leftTarget || currRightCounter<-
10      < rightTarget))
11     {
12         unsigned long currTime = millis();
13         if (currTime >= timeout)
14         {
15             currLeftCounter = _motors[MOTOR_LEFT].getCounter<-
16                 ();
17             currRightCounter = _motors[MOTOR_RIGHT].<-
18                 getCounter();
19             int diffLeft = currLeftCounter - dataLine[<-
20                 idxDataLine].ticksLeft;
21             int diffRight = currRightCounter - dataLine[<-
22                 idxDataLine].ticksRight;
23             timeout = currTime + _PERIOD;
24             int maxDiff = max(diffLeft, diffRight);
25             int minDiff = min(diffLeft, diffRight);
26             float ratio = 0.0f;
27             float motorDif;
28             float rightSpeed, leftSpeed;
29             if (minDiff > 0)
30             {
31                 ratio = (float)(maxDiff) / (float)(minDiff);
32                 float ratioAux = (ratio - 1.0) / 2.0;
33                 float ratioAdd = 1.0 + ratioAux;
34                 float ratioSub = 1.0 - ratioAux;
35                 if (diffLeft > diffRight)
36                 {
37                     rightSpeed = _motors[MOTOR_RIGHT].getPWM<-
38                         () * ratioAdd;
39                     leftSpeed = _motors[MOTOR_LEFT].getPWM()<-
40                         * ratioSub;
41                 }
42             }
43             else
44             {
45                 rightSpeed = _motors[MOTOR_RIGHT].getPWM<-
46                         () * ratioSub;
47                 leftSpeed = _motors[MOTOR_LEFT].getPWM()<-
48                         * ratioAdd;
49             }
50             float motorDif_right = rightSpeed < _MIN_PWM<-
51                 ? _MIN_PWM - rightSpeed : (rightSpeed > <-
52                 _MAX_PWM ? _MAX_PWM - rightSpeed : 0);
53             float motorDif_left = leftSpeed < _MIN_PWM ?<-
54                 _MIN_PWM - leftSpeed : (leftSpeed > <-
55                 _MAX_PWM ? _MAX_PWM - leftSpeed : 0);
56         }
57     }
58 }
```

```
42         rightSpeed = rightSpeed + motorDif_left < ↵
43             _MIN_PWM ? _MIN_PWM : (rightSpeed + ↵
44                 motorDif_left > _MAX_PWM ? _MAX_PWM : ↵
45                     rightSpeed + motorDif_left);
46         leftSpeed = leftSpeed + motorDif_right < ↵
47             _MIN_PWM ? _MIN_PWM : (leftSpeed + ↵
48                 motorDif_right > _MAX_PWM ? _MAX_PWM : ↵
49                     leftSpeed + motorDif_right);
50     _motors[MOTOR_RIGHT].setPWM(rightSpeed);
51     _motors[MOTOR_LEFT].setPWM(leftSpeed);
52     ++indxDataLine;
53     dataLine[indxDataLine].pwmLeft = _motors[←
54         MOTOR_LEFT].getPWM();
55     dataLine[indxDataLine].pwmRight = _motors[←
56         MOTOR_RIGHT].getPWM();
57     dataLine[indxDataLine].ticksLeft = ←
58         currLeftCounter;
59     dataLine[indxDataLine].ticksRight = ←
60         currRightCounter;
61     dataLine[indxDataLine].ratio = ratio;
62 }
63 }
64 }
65 }
```

Conclusão e Trabalho Futuro

Durante a realização deste projeto, foram estabelecidos objetivos que abrangiam desde a pesquisa e seleção de componentes até a montagem e programação do sistema. A plataforma desenvolvida incluiu sensores de distância e toque, controladores de motor, motores e rodas, além de tecnologia de comunicação sem fio para interação com um computador. Também foi desenvolvida uma PCB para facilitar a montagem e desmontagem do sistema.

Os resultados obtidos com a implementação e validação do sistema foram satisfatórios. A plataforma demonstrou ser capaz de realizar movimentos no espaço 2D, além de permitir o controlo remoto através de comandos enviados pelo computador.

Os objetivos futuros do projeto são ambiciosos e certamente levarão o projeto a um nível mais avançado. Os seguintes objetivos propostos encontram-se listados:

- Criação de uma aplicação para dispositivos móveis e computador para enviar comandos via Bluetooth: Ao desenvolver uma aplicação para dispositivos móveis e computador, permitirá que os utilizadores controlem o robô de forma mais conveniente e intuitiva.
- Criação de uma biblioteca em Java para controlar o robô: A criação de uma biblioteca em Java permitirá que outros programadores integrem facilmente a funcionalidade do robô nos seus próprios projetos e aplicações.
- Implementação de funcionalidades com inteligência artificial: A adição de inteligência artificial ao projeto é um passo significativo para melhorar as capacidades do robô. Ao analisar o ambiente ao redor por meio de uma câmera, o robô pode tomar decisões mais informadas e autónomas. As possibilidades de aplicação são vastas, como deteção e reconhecimento de objetos, navegação autónoma, interação com o ambiente e reconhecimento de comandos de voz.

Em suma, este projeto serviu como um elemento vivo, que combinou pesquisa,

Capítulo 7. Conclusão e Trabalho Futuro

desenvolvimento e aprendizagem, e pode servir de base para a construção de futuros projetos no campo da robótica educacional. A plataforma desenvolvida tem o potencial de auxiliar estudantes a explorar e compreender conceitos complexos de forma prática e interativa, contribuindo para o avanço da educação em ciência, tecnologia, engenharia e matemática.

Esquemática da placa de circuito impresso

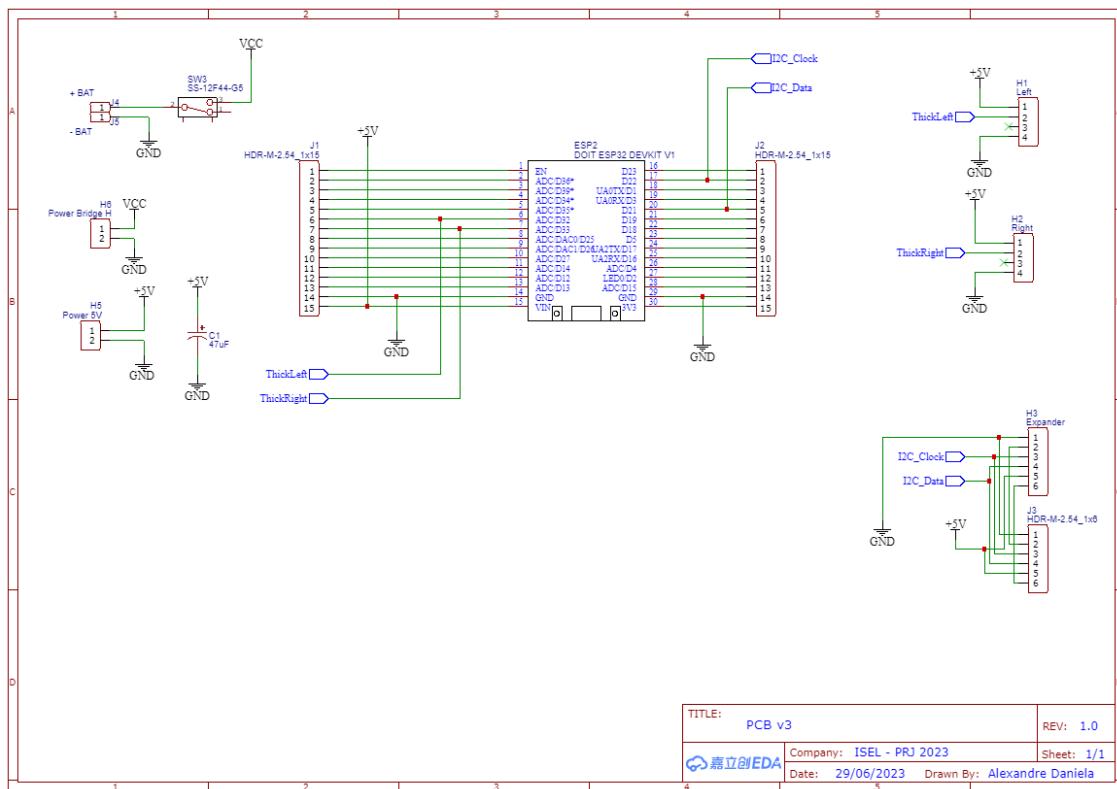


Figura A.1: Esquemática da PCB

Apêndice A. Esquemática da placa de circuito impresso

Diagrama de classes UML completo

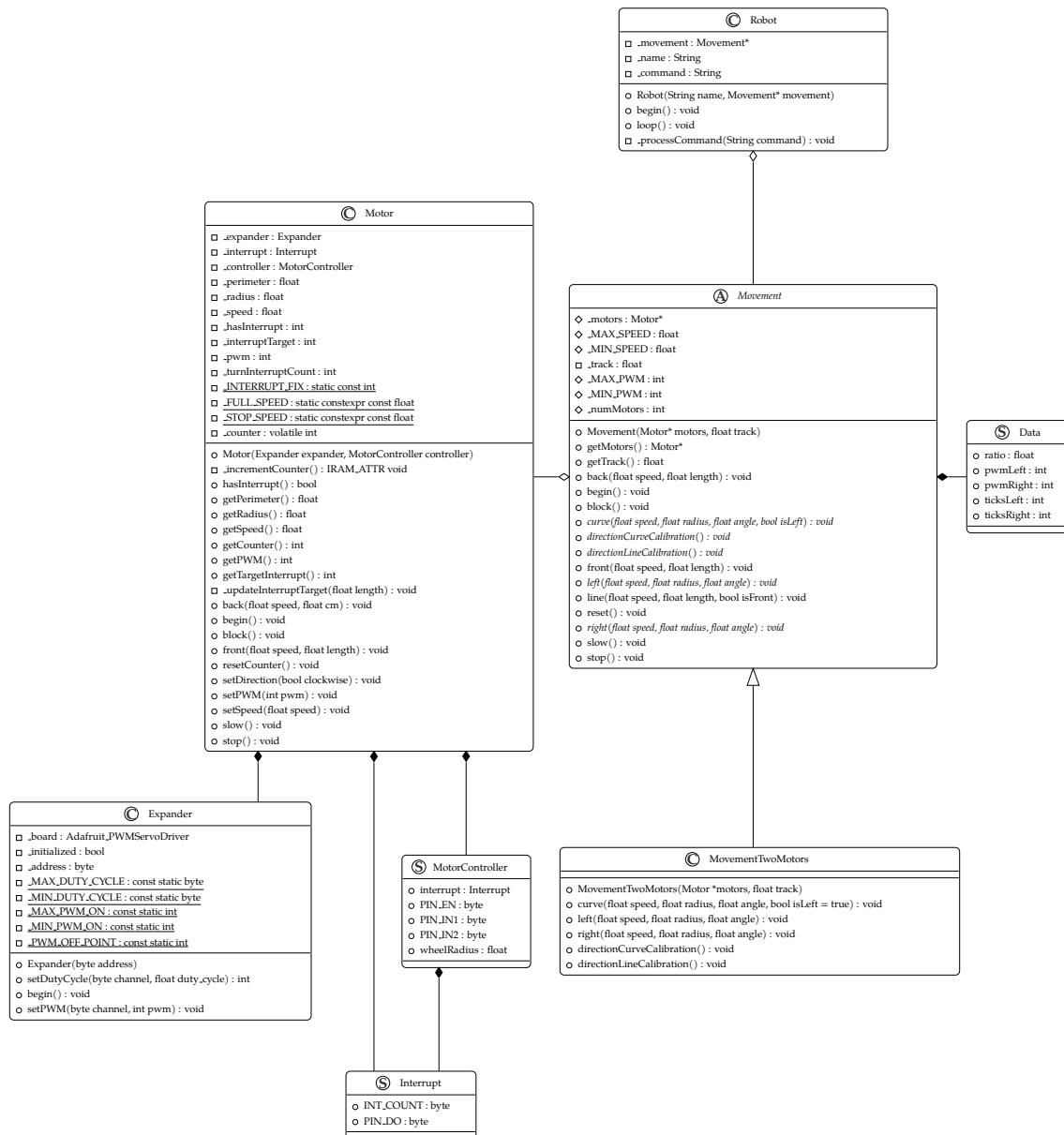


Figura B.1: Diagrama de classes UML completo

Apêndice B. Diagrama de classes UML completo

Bibliografia

- [1] Jorge Pais. Aulas de computação física, 2022.
- [2] Carolina Neves. Projeto final de curso - robô, Novembro 2020.
- [3] Mauser.pt. Kit para carro robot 2wd c/ motores e rodas. https://mauser.pt/catalog/product_info.php?products_id=096-7641. [Online: Acedido em junho 2023].
- [4] Mauser.pt. Motor com roda para carros robóticos - joy-it. https://mauser.pt/catalog/product_info.php?products_id=096-7645. [Online: Acedido em junho 2023].
- [5] SVmodelismo.net. Gens ace bateria lipo. <https://www.svmodelismo.net/pt/lipo/4769-gens-ace-soaring-mini-2200mah-74v-20c-2s1p-lipo-battery-pack-with-xt.html>. [Online: Acedido em junho 2023].
- [6] Loja. Encoder óptico h206. <https://www.google.com>. [Online: Acedido em junho 2023].
- [7] Mauser.pt. L298n dc motor driver module. https://mauser.pt/catalog/product_info.php?products_id=096-6807. [Online: Acedido em junho 2023].
- [8] Mauser.pt. Sensor ultrassónico (hc-sr04) compatível com arduino. https://mauser.pt/catalog/product_info.php?products_id=096-6220. [Online: Acedido em junho 2023].
- [9] Mauser.pt. Kit microswitch com 5 patilhas substituiveis (ip40) spdt 250vac 16a. https://mauser.pt/catalog/product_info.php?cPath=324_1401_707&products_id=010-1429. [Online: Acedido em junho 2023].
- [10] Mauser.pt. Esp32 wroom-32 devkit v1. https://mauser.pt/catalog/product_info.php?products_id=096-7620. [Online: Acedido em junho 2023].