



INSTITUTO SUPERIOR DE ENGENHARIA DE LISBOA (ISEL)

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

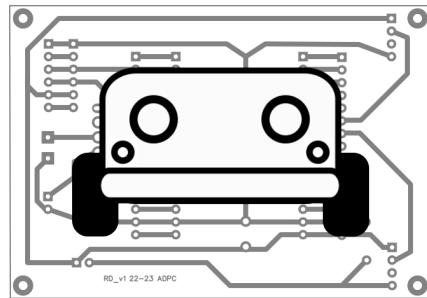
---

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

---

11 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. Além disso, o robô permite estabelecer comunicação com um computador/telemóvel através do protocolo 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, Comunicação Sem Fio, Bluetooth.



---

# 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 sensors and actuators. Furthermore, the robot allows communication with a computer/mobile device through Bluetooth protocol.

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, Didatic Plataform, Remote Control, Microcontroller, ESP32, Wireless Communication, Bluetooth.



---

# 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 Miguens 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 de Conteúdos

<b>Resumo</b>	i
<b>Abstract</b>	iii
<b>Agradecimentos</b>	v
<b>Índice de Conteúdos</b>	vii
<b>Lista de Figuras</b>	ix
<b>Lista de Tabelas</b>	xi
<b>Lista de Listagens</b>	xiii
<b>Lista de Acrónimos</b>	xv
<b>1 Introdução</b>	1
1.1 Motivação . . . . .	1
1.2 Contextualização . . . . .	1
1.3 Objetivos e Contributos . . . . .	2
1.4 Organização do Documento . . . . .	3
<b>2 Trabalho Relacionado</b>	5
<b>3 Modelo Proposto</b>	7
3.1 Requisitos . . . . .	7
3.1.1 Requisitos Funcionais . . . . .	7
3.1.2 Requisitos Não Funcionais . . . . .	8
3.2 Casos de Utilização . . . . .	8
3.3 Fundamentos . . . . .	9
<b>4 Arquitetura do Sistema</b>	11
4.1 Estrutura . . . . .	11
4.2 Motor e Alimentação . . . . .	12

4.3	Controlador do Motor . . . . .	14
4.4	Expansor . . . . .	16
4.5	Sensores . . . . .	17
4.5.1	Sensor de Distância . . . . .	17
4.5.2	Sensor de Toque . . . . .	17
4.6	Microcontrolador . . . . .	18
<b>5</b>	<b>Implementação</b>	<b>19</b>
5.1	Construção da Plataforma . . . . .	20
5.2	Programação . . . . .	22
5.2.1	Expansor . . . . .	24
5.2.2	Motor . . . . .	24
5.2.3	Movimento . . . . .	26
5.2.4	Robot . . . . .	29
5.3	Implementação do 2º Protótipo . . . . .	30
<b>6</b>	<b>Validação e Testes</b>	<b>35</b>
6.1	Desvio Antes da Paragem . . . . .	35
6.2	Diferencial por Software . . . . .	35
6.3	Custos do Protótipo e Custo de Produção . . . . .	41
<b>7</b>	<b>Conclusões e Trabalho Futuro</b>	<b>43</b>
<b>Bibliografia</b>		<b>45</b>
<b>A Esquema elétrico da PCB do 2º protótipo</b>		<b>47</b>
<b>B Diagrama de Classes UML</b>		<b>49</b>

---

# Listas de Figuras

1.1	Diagrama abstrato . . . . .	2
2.1	Robô LEGO® . . . . .	5
2.2	Robô autónomo . . . . .	5
3.1	Casos de utilização . . . . .	8
4.1	Vista interior do <i>chassi</i> escolhido . . . . .	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 do circuito em Ponte H . . . . .	16
4.6	Diagrama simplificado do funcionamento de um microcontrolador num sistema robótico . . . . .	18
5.1	Diagrama do sistema em blocos . . . . .	19
5.2	1 <sup>a</sup> Fase da montagem . . . . .	20
5.3	Diagrama ilustrativo do circuito . . . . .	21
5.4	Esquema elétrico do protótipo . . . . .	22
5.5	Soldagem e finalização da montagem . . . . .	22
5.6	Diagrama de classes UML simplificado . . . . .	23
5.7	Diagrama de uma curva à esquerda . . . . .	28
5.8	Layout da <i>Printed Circuit Board (PCB)</i> . . . . .	31
5.9	Revelação das pistas . . . . .	31
5.10	Impressão final . . . . .	32
5.11	Furação da PCB . . . . .	33
5.12	Montagem da PCB . . . . .	33
6.1	Testes práticos com rodas no ar sem compensação . . . . .	37
6.2	Testes práticos com rodas no ar com compensação . . . . .	40
A.1	Esquema elétrico da PCB do 2º protótipo . . . . .	47
B.1	Diagrama de classes UML completo . . . . .	49



---

# **Lista de Tabelas**

2.1	Características dos diferentes sistemas de robôs . . . . .	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> LM393 . . . . .	14
4.5	Especificações técnicas do controlador L298N . . . . .	16
4.6	Especificações técnicas do expansor PCA9685 . . . . .	16
4.7	Especificações técnicas do sensor de distância HC-SR04 . . . . .	17
4.8	Especificações técnicas do sensor de toque . . . . .	17
4.9	Especificações técnicas do microcontrolador <i>Espressif32</i> (ESP32) . .	18
5.1	Valores lógicos para controlo . . . . .	25
5.2	Tabela de comandos implementados . . . . .	30
6.1	Resultados dos testes práticos sem compensação . . . . .	36
6.2	Resultados dos testes práticos com compensação . . . . .	40
6.3	<i>Bill of Materials</i> (BOM) protótipo . . . . .	41
6.4	BOM produção . . . . .	42



---

# **Lista de Listagens**

1	Troço de código do diferencial por software . . . . .	38
---	---	----



---

# **Lista de Acrónimos**

**BOM** *Bill of Materials.* xi, 41, 42

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

**DEETC** *Departamento de Engenharia Eletrónica e de Telecomunicações e Computadores.*  
1

**ESP32** *Espressif32.* xi, 18, 29

**GPIO** *General Purpose Input/Output.* 18

**I2C** *Inter-Integrated Circuit.* 9, 16, 18, 24

**KB** *Kilobyte* ( $10^3$ ). 18

**LED** *Light-Emitting Diode.* 9

**LEIM** *Licenciatura em Engenharia Informática e Multimédia.* 1

**MB** *Megabyte* ( $10^6$ ). 18

**PCB** *Printed Circuit Board.* viii, ix, 3, 6, 8, 9, 19, 30–33, 43, 47, 48

**PWM** *Pulse Width Modulation.* 9, 14, 23–25, 38, 40, 41

**RAM** *Random Access Memory.* 18

**SONAR** *Sound Navigation and Ranging.* 17, 21

**UML** *Unified Modeling Language.* viii, ix, 22, 23, 49, 50

**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).

Os conceitos relacionados com a eletrónica são abordados algumas vezes durante o curso LEIM, no entanto este está mais direcionado para a programação de software. A maior 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 diferentes propósitos pedagógicos. A plataforma desenvolvida é programável e a sua estrutura genérica permite a introdução de mais sensores e atuadores, permitindo assim, que os estudantes possam explorar diferentes cenários e desafios e também personalizá-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. 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 o 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 de modo a permitir a movimentação do robô. Para além disso irá utilizar tecnologia de comunicação sem fios para permitir o controlo do robô com um computador/telemóvel.

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

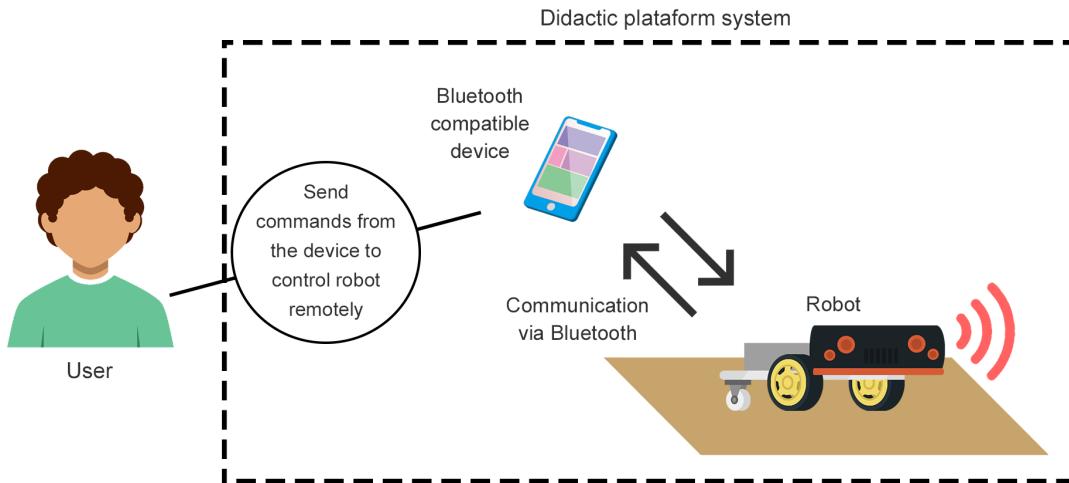


Figura 1.1: Diagrama abstrato

### 1.3 Objetivos e Contributos

Dessa forma, foram estabelecidos os principais objetivos e contributos 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 protocolo Bluetooth. Os movimentos do robô serão controlados a partir dos comandos enviados pelo computador/telemóvel;
4. desenvolvimento de uma *Printed Circuit Board* (PCB) para tornar a montagem 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.
6. disponibilização do material desenvolvido num repositório, bem como na forma de uma biblioteca disponível no Arduino IDE [1].

## 1.4 Organização do Documento

O documento está organizado em sete capítulos. O capítulo 1 é a introdução, que apresenta a motivação, a contextualização, 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 capítulo 3 apresenta o modelo proposto, incluindo os requisitos, fundamentos e abordagem, de modo a detalhar a proposta de solução para o problema abordado. O capítulo 4, 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 capítulo 7, são apresentadas as conclusões do trabalho desenvolvido, 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® [2] utilizado nas aulas de Fundamentos de Sistemas Operativos, cuja biblioteca foi desenvolvida por [3].

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 [2], representado na Figura 2.1, era controlado remotamente pela biblioteca [3] e capaz de se movimentar no espaço 2D.

Um outro projeto a referenciar é um robô desenvolvido por [4], 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 representado na Figura 2.2.



Figura 2.1: Robô LEGO® [2]



Figura 2.2: Robô autónomo

Uma das principais vantagens do robô autónomo é a sua acessibilidade económica. Enquanto, por exemplo, o robô da LEGO® MINDSTORMS® é referência pela sua qualidade de construção e recursos avançados, o protótipo do robô aqui desen-

## Capítulo 2. Trabalho Relacionado

---

volvido tem um foco especial na redução de custos. Isso torna-o uma opção mais acessível para estudantes e entusiastas de robótica que possuem orçamentos mais limitados.

A Tabela 2.1 apresenta uma comparação das principais características entre os trabalhos acima mencionados e o trabalho apresentado neste relatório.

Tabela 2.1: Características dos diferentes sistemas de robôs

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

Uma vantagem significativa do robô didático aqui proposto é o facto de ser baseado em código aberto. Isso significa que o código-fonte do robô está disponível publicamente, em [1], 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 sensores/atuadores 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.

Outra vantagem do sistema desenvolvido é o facto de se ter disponibilizado o *layout* da PCB facilitando assim a montagem de todo o circuito minimizando a quantidade de ligações a efetuar, bem como minimizando falhas devido a cabos partidos ou maus contactos.

# Modelo Proposto

O presente capítulo começa por apresentar os requisitos do projeto (Secção 3.1). De seguida, apresenta os casos de utilização (Secção 3.2). Por fim, os fundamentos teóricos (Secção 3.3).

## 3.1 Requisitos

Requisitos funcionais são descrições precisas e detalhadas das funcionalidades que um sistema deve oferecer, definindo as ações que o sistema deve ser capaz de executar e como este deve responder a diferentes interações com o utilizador. Por outro lado, os requisitos não funcionais são características do sistema que não se relacionam diretamente com as funcionalidades específicas, mas sim com as suas propriedades gerais, como por exemplo desempenho, segurança, usabilidade, escalabilidade, entre outros.

Na Secção 3.1.1 são apresentados os requisitos funcionais e na Secção 3.1.2 os requisitos não funcionais.

### 3.1.1 Requisitos Funcionais

No contexto deste projeto foram identificados os seguintes requisitos funcionais:

- Movimentação retilínea para a frente/trás (requisito visível);
- Movimentação curvilínea para a direita/esquerda (requisito visível);
- Previsão de movimentos (requisito invisível)
- Comunicação a partir de uma rede sem fios (requisito visível);
- Capacidade de leitura de dados dos sensores (requisito visível).

### 3.1.2 Requisitos Não Funcionais

Após identificar os requisitos funcionais foram identificados os requisitos não funcionais compostos por:

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

## 3.2 Casos de Utilização

Os casos de utilização são uma técnica de modelação 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 representado 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 casos de utilização.

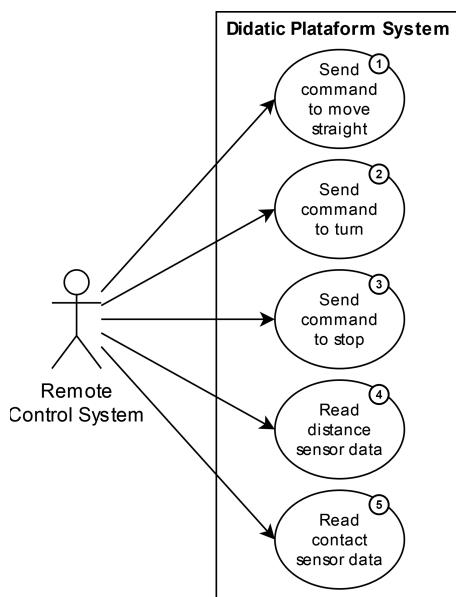


Figura 3.1: Casos de utilização

- **Enviar comando para movimento retilíneo (comando 1):** O sistema de controlo remoto envia um comando ao robô, instruindo-o a mover-se em linha reta.
- **Enviar comando para virar (comando 2):** O sistema de controlo remoto envia um comando ao robô, instruindo-o a efetuar uma curva.
- **Enviar comando para parar (comando 3):** O sistema de controlo remoto envia um comando ao robô, instruindo-o a parar o movimento.
- **Ler dados do sensor de distância (comando 4):** 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 (comando 5):** 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.

### 3.3 Fundamentos

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

**Bluetooth** é uma tecnologia de comunicação sem fio de curto alcance projetada para interligar dispositivos eletrónicos.

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

**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 Light-Emitting Diode (LED), controlo de potência em fontes de alimentação e outras aplicações.

**PCB** é uma placa plana, feita de material isolante, que contém pistas (*tracks*) condutoras e ilhas (*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 dos motores é descrito na Secção 4.3. Depois descreve-se o expansor utilizado (Secção 4.4). Os diferentes sensores presentes no sistema são analisados na Secção 4.5. Na Secção 4.6 descreve-se o microcontrolador utilizado. Ao longo deste capítulo são apresentados em complementos às especificações do material utilizado, o preço dos mesmos. Todas as características técnicas foram obtidas através das respetivas referências bibliográficas.

## 4.1 Estrutura

Começou-se por selecionar a estrutura para o protótipo do robô didático, a seleção 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 [5]

Rodas	Rodas de apoio	Diâmetro das rodas [cm]	Peso [g]	Preço [€]
2	1	6,5	314	14,00

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. O *chassi* selecionado para

o protótipo do robô está representado na Figura 4.1. Um dos objetivos futuros consiste na implementação de um protótipo com quatro rodas motrizes.



Figura 4.1: Vista interior do *chassi* escolhido [5]

## 4.2 Motor e Alimentação

Anteriormente descreveu-se que o robô é composto por duas rodas, acionadas por dois motores DC [6] que tipicamente já estão incluídos no conjunto do *chassi*. O motor DC utilizado, juntamente com a roda, encontra-se representado na Figura 4.2. 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 o arranque dos motores.



Figura 4.2: Roda e motor DC [6]

A Tabela 4.2 apresenta as especificações técnicas dos motores utilizados no robô.

Tabela 4.2: Especificações técnicas do motor DC [6]

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

Inicialmente, o robô foi alimentado por uma pilha de 9 V, o que aparentava 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 [4] constatou-se que o problema estava relacionado com o pico de corrente exigido pelos motores quando os mesmos eram ativados em simultâneo (2,8 A), o qual excedia a capacidade de corrente da pilha utilizada inicialmente neste projeto.

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 solicitada à pilha. Embora tenha sido bem-sucedida esta abordagem em cumprir esse objetivo, essa implementação resultou num problema adicional. Verificou-se que esta abordagem 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 corrente e capacidade. Esta nova fonte de energia já tem corrente suficiente para permitir o arranque simultâneo dos dois motores.

A bateria selecionada para substituir a pilha é a Gens Ace Bateria Lipo [7], com uma capacidade de 2200 mAh, classificação 20C na capacidade de descarga (*discharge rate*) e uma tensão nominal de 7,4 V. As especificações técnicas e custos encontram-se apresentadas na Tabela 4.3.

Tabela 4.3: Especificações técnicas da bateria Gens Ace [7]

Tensão [V]	Capacidade [mAh]	Capacidade Pico [A]	Peso [g]	Preço [€]
7,4	2200	44	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, 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.

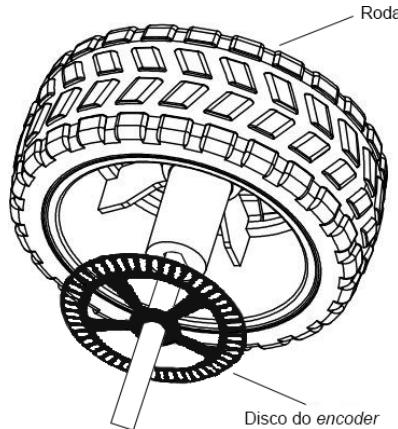


Figura 4.3: Roda com disco

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 impulsos elétricos. Esses impulsos são utilizados para medir a distância percorrida pelo motor. As especificações técnicas do *encoder* LM393 encontram-se descritas na Tabela 4.4.

Tabela 4.4: Especificações técnicas do *encoder* LM393 [8]

Tensão de operação [V]	Peso [g]	Preço [€]
3,3 – 5	1	8,14

### 4.3 Controlador do Motor

O controlador de motor é a unidade responsável pela gestão do funcionamento de um ou mais motores. Este componente é responsável por regular diversos parâmetros, tais como velocidade ou sentido de rotação de acordo com as características específicas de cada controlador.

Na versão aqui considerada opta-se por definir o valor médio da tensão aos terminais do motor através da técnica *Pulse Width Modulation* (PWM) que consiste em enviar uma série de impulsos de forma a ajustar o valor médio da tensão de entrada.

A largura dos impulsos, 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* (4.1) , com o tempo de pulsos ativos no valor lógico "1",  $t_{on}$ , o tempo de pulsos no valor lógico "0",  $t_{off}$  e período  $T$ .

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

A Figura 4.4 representa um *duty cycle* a 50%, em que a linha tracejada consiste no valor de tensão média.

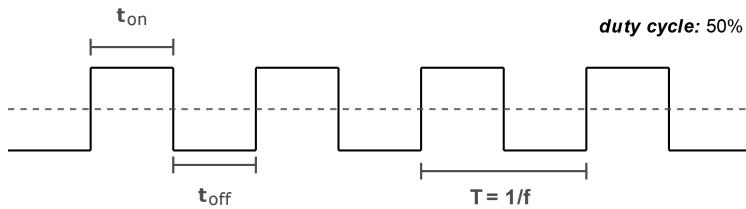


Figura 4.4: *Duty cycle* a 50%

Relativamente à alteração do sentido de rotação do motor é necessário introduzir um controlo através de uma 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 em 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 representado um esquema simplificado do circuito em 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.

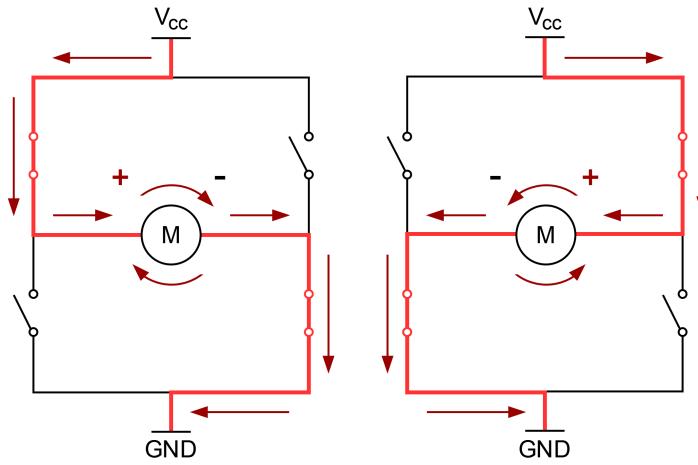


Figura 4.5: Esquema simplificado do circuito em Ponte H

O L298N DC Motor Driver foi selecionado como módulo controlador dos motores DC do robô didático aqui proposto. O controlador em questão é um módulo de controlo bidirecional que permite acionar os dois motores DC de forma independente. As especificações técnicas do módulo estão descritas na Tabela 4.5.

Tabela 4.5: Especificações técnicas do controlador L298N [9]

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

## 4.4 Expansor

Um expansor é um dispositivo eletrónico que permite expandir a capacidade de entradas e saídas de um sistema. Observa-se que o controlador [9] requer um total de 6 sinais para comandar dois motores. A fim de economizar pinos e permitir a adição de mais sensores e atuadores, optou-se por instalar o expansor [10] no sistema. Esse expansor é composto por 16 pinos e requer apenas 2 sinais do microcontrolador para estabelecer comunicação através do protocolo I2C. As especificações do expansor estão descritas na Tabela 4.6.

Tabela 4.6: Especificações técnicas do expansor PCA9685 [10]

Tensão de alimentação [V]	Número de pinos	Peso [g]	Preço [€]
3 – 5	16	11	13,40

## 4.5 Sensores

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

### 4.5.1 Sensor de Distância

O sensor de distância *Sound Navigation and Ranging* (SONAR) é um dispositivo que utiliza ondas sonoras para medir a distância entre o robô e um objeto próximo. Medindo o tempo que o sinal emitido demora a ser refletido e capturado novamente pelo sensor é possível calcular a distância do alvo ao sensor.

As especificações técnicas do sensor de distância [11] utilizado no sistema estão apresentadas na Tabela 4.7.

Tabela 4.7: Especificações técnicas do sensor de distância HC-SR04 [11]

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

### 4.5.2 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 4.8.

Tabela 4.8: Especificações técnicas do sensor de toque [12]

Tensão de alimentação [V]	Durabilidade Mecânica [Ciclos]	Resistência máxima de contacto [mΩ]	Preço [€]
30 DC - 250 AC	10 000 000	30	3,46

## 4.6 Microcontrolador

O microcontrolador é um componente eletrónico integrado num único circuito inteligente 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.

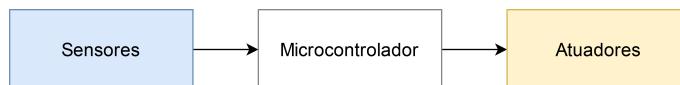


Figura 4.6: Diagrama simplificado do funcionamento de um microcontrolador num sistema robótico

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, esta 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 atualmente uma solução de custo mais baixo.

Tabela 4.9: Especificações técnicas do microcontrolador ESP32 [13]

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

# Implementação

Este capítulo descreve o processo de implementação do projeto, desde a montagem do protótipo (Secção 5.1), programação do sistema (Secção 5.2) e a construção da PCB (Secção 5.3). A Figura 5.1 apresenta o diagrama da arquitetura proposta.

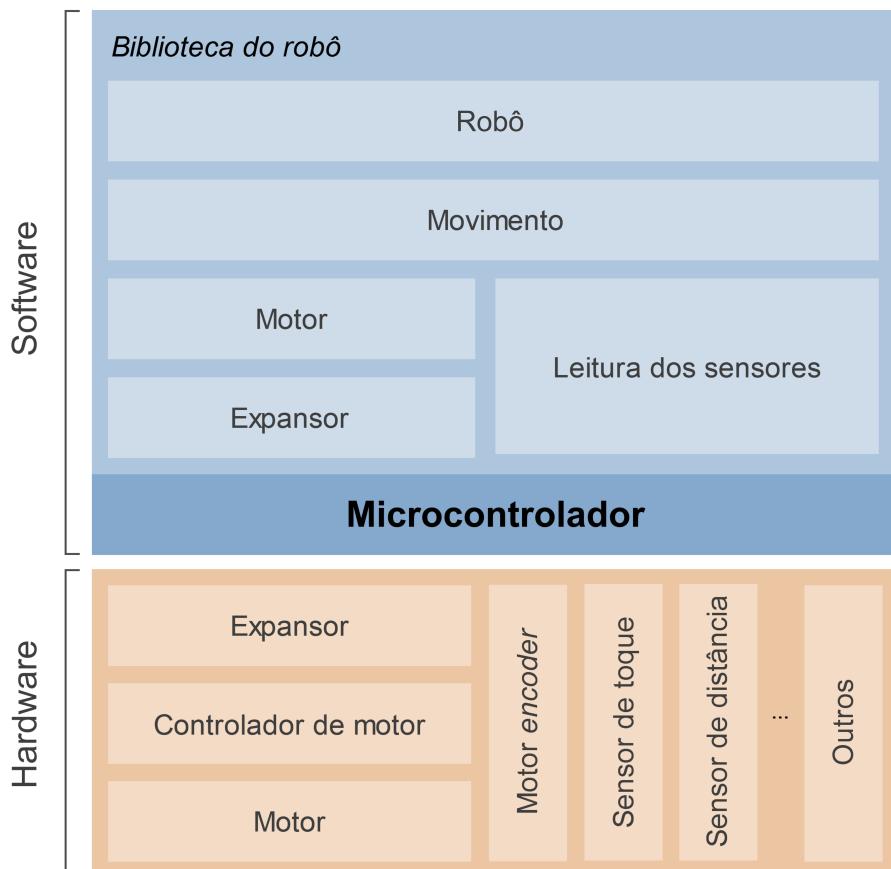


Figura 5.1: Diagrama do sistema em blocos

No *software*, os blocos indicados correspondem aos diversos módulos do sistema, enquanto na parte do *hardware*, os blocos referem-se aos diferentes componentes eletrónicos. O microcontrolador é o módulo que interliga estes componentes com os respetivos blocos de *software*.

## 5.1 Construção da Plataforma

Para começar, é importante salientar que a montagem da plataforma física do robô é a primeira etapa deste projeto, e requer a habilidade técnica e o conhecimento específico sobre os diferentes componentes do robô didático.

A primeira fase da construção inicia-se com os componentes associados ao *chassi*, começando pela fixação dos conjuntos motor/roda à estrutura. Em seguida, os *encoders* foram fixados nos respetivos orifícios, proporcionando a detação precisa do movimento das rodas. A Figura 5.2 ilustra a vista superior da plataforma nesta fase da montagem.

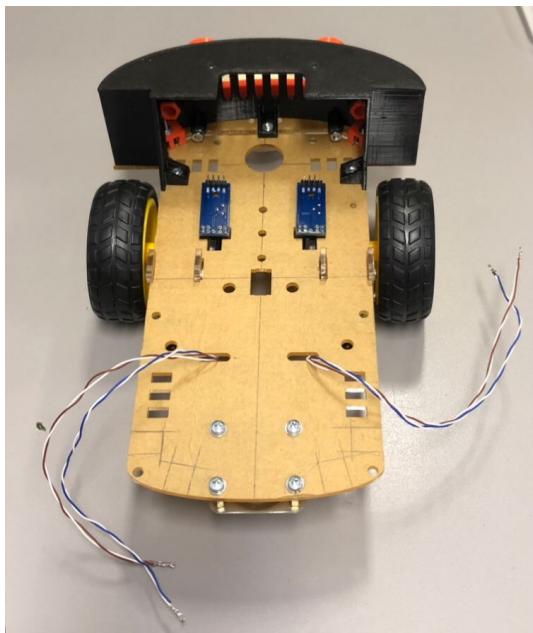


Figura 5.2: 1<sup>a</sup> Fase da montagem

Em seguida, o próximo passo é a montagem do microcontrolador, controlador do motor, expansor e sensores.

O controlador do motor desempenha um papel importante no sistema. Ele é responsável por receber a carga total da bateria, pois possui uma entrada de alimentação que suporta uma faixa de tensão entre 5 V e 35 V utilizada para alimentar os motores. Além disso, tem uma saída regulada de 5 V, que será utilizada para fornecer a tensão de alimentação do microcontrolador. Deve ser introduzido um interruptor entre a bateria e o controlador [9] de modo a desligar todo o sistema.

Os *encoders* e o expansor necessitam aproximadamente de 420 mA, 10 mA por

cada *encoder* e 400 mA para o expansor, desta forma poderão ser alimentados pelo regulador de tensão de 3,3 V do microcontrolador. Este possui uma saída de fornecimento de energia com 3,3V, com uma corrente máxima de 800 mA.

Foi elaborado um diagrama ilustrativo que representa a interligação dos componentes do sistema, apresentado na Figura 5.3.

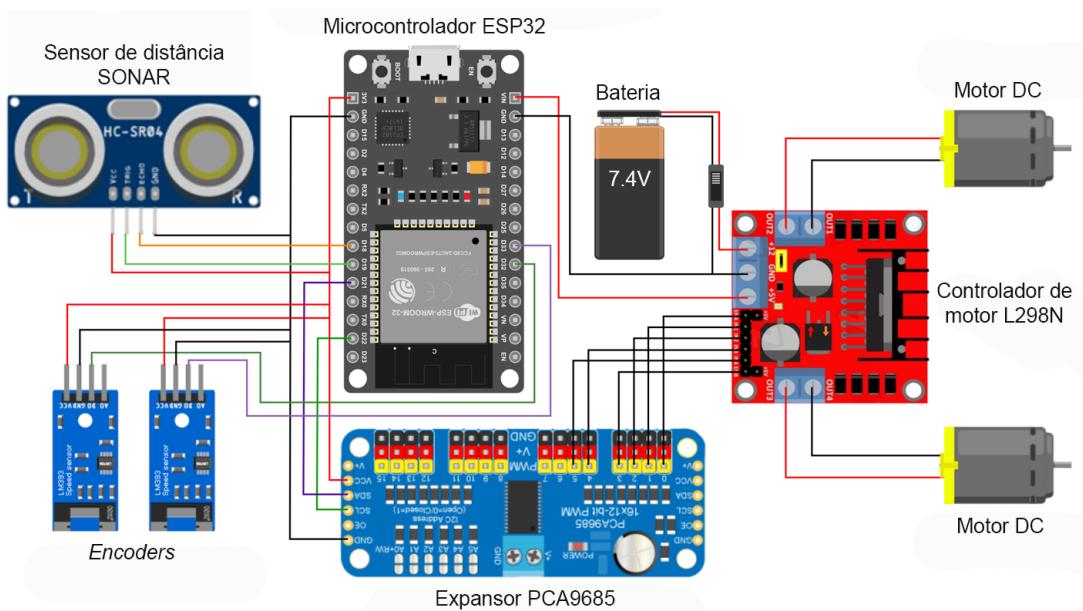


Figura 5.3: Diagrama ilustrativo do circuito

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 funcionamento adequado. É importante notar que a ausência de *encoders* no sistema compromete a capacidade de garantir o funcionamento mais preciso dos motores durante a marcha.

Procedeu-se em seguida à implementação do esquema elétrico, representado na Figura 5.4.

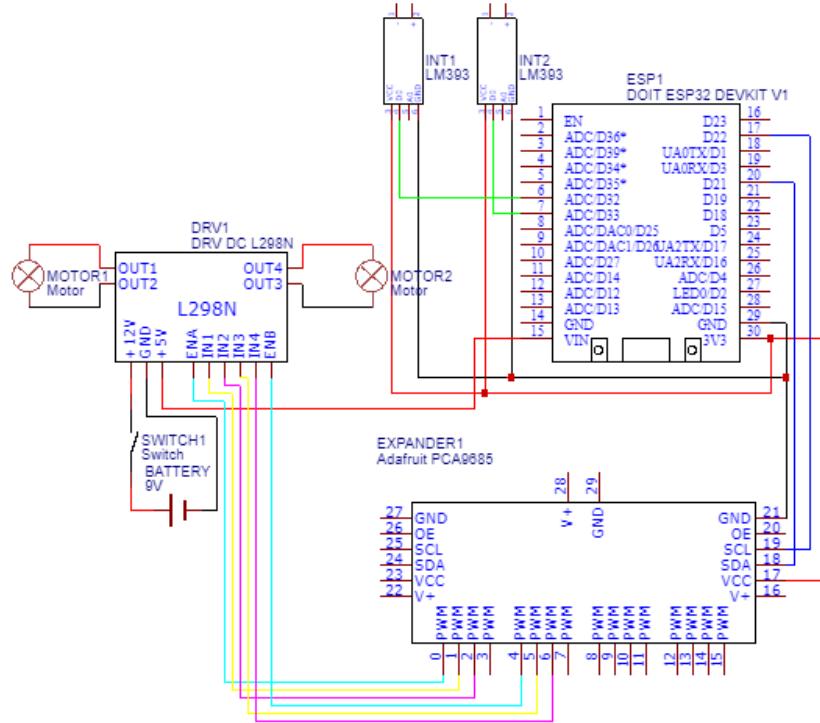


Figura 5.4: Esquema elétrico do protótipo

Com base no esquema elétrico implementou-se o primeiro protótipo conforme a Figura 5.5.

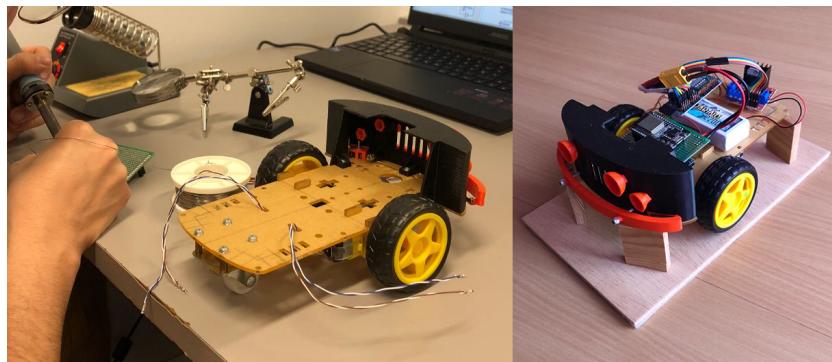


Figura 5.5: Soldagem e finalização da montagem

## 5.2 Programação

Na Figura 5.6 está representado o diagrama *Unified Modeling Language* (UML) proposto 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 está representado na Figura B.1 do anexo B.

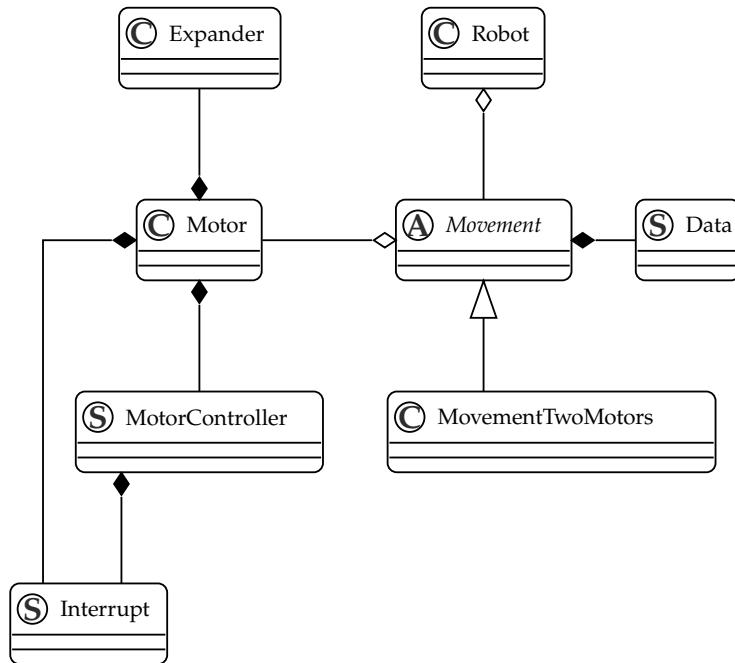


Figura 5.6: Diagrama de classes UML simplificado

De seguida apresentam-se os detalhes de cada uma das classes referidas na Figura 5.6.

A classe Expander representa um dispositivo de expansão, esta classe é responsável por manipular e enviar o 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 estrutura referida inclui na sua composição uma estrutura chamda Interrupt que define as características do respetivo encoder. 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`.

Iniciou-se a implementação dos módulos, previamente apresentados na introdução desta secção.

### 5.2.1 Expansor

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 `Expander` (`byte address`) 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.

Do ponto de vista do utilizador, o *duty cycle* é expresso numa gama de valores em percentagem, variando de 0% a 100%. Utilizando os extremos destes valores, ou seja, 0% e 100%, consegue-se então obter os valores lógicos "0" e "1" continuamente. Internamente, o expansor utiliza uma escala interna entre 0 e 4095 dos valores do PWM.

### 5.2.2 Motor

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 associados ao controlador, que por sua vez está associado a um 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 desta. Esses dados são enviados numa estrutura struct designada por `MotorController`. As principais funções desta classe `Motor` (`Expander expander, MotorController controller`) 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 que dão suporte ao funcionamento descrito na Secção 4.3. 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 binário ("0" ou "1") aplicado a esse pino, o motor pode ser acionado numa direção específica.

De acordo com a documentação do controlador, os valores lógicos a serem enviados para os pinos do controlador estão descritos na Tabela 5.1.

Tabela 5.1: Valores lógicos para controlo

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

Outra funcionalidade é a capacidade de determinar o número de ticks que correspondem à distância a percorrer. A fórmula é baseada numa relação proporcional entre o comprimento percorrido e o número de ticks registados pelo *encoder*. O comprimento é dividido pelo perímetro da roda para obter a proporção do círculo

percorrido. Em seguida, essa proporção é multiplicada pelo número de furos do *encoder* para obter o número correspondente de ticks.

Para uma determinada distância  $d$  a percorrer, 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  $\#tick$  é a seguinte:

$$\#tick = \frac{d}{p} \cdot N. \quad (5.1)$$

Resumindo, a fórmula matemática apresentada em (5.1) é utilizada para converter uma distância em centímetros num número de ticks correspondente, o qual é utilizado para determinar o momento em que o robô deve parar os motores ao atingir a referida distância  $d$ .

### 5.2.3 Movimento

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(*Motor \*motors*, *float track*) 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 cujas principais funcionalidades são as seguintes:

- Movimentar em linha reta, para a frente/trás, com velocidade inicial  $v_0$ , até a uma distância de  $d$  cm;
- Movimentar em curva para a esquerda/direita, com velocidade inicial  $v_0$ , raio  $r$  até ao ângulo  $\alpha$ ;
- 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 ou para 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 não são definidas na biblioteca aqui desenvolvida pelo que fica ao critério do utilizador, pois estas dependem do número de rodas motrizes.

Desta forma, foi concebida e implementada 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 `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 movimenta numa trajetória reta, mantendo sua orientação ao longo do percurso. Todo o processo envolveu um conjunto de testes práticos que foram documentados e descritos na Secção 6.2.

O robô tem que ser capaz de realizar curvas para a esquerda e para a direita com um raio, ângulo e velocidade específicos. Desse modo é necessário calcular a velocidade para cada um dos motores de forma a realizar a curva pretendida.

Ao percorrer uma curva completa, ou seja, um ângulo de  $360^\circ$ , 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  $\alpha$ , é essencial medir a distância até a esse ângulo. Nesse sentido, para calcular a distância que o robô deve percorrer até ao ângulo  $\alpha$ , utiliza-se a regra de três simples. Assim, temos a seguinte proporção:

$$\begin{aligned} 2\pi r &\rightarrow 360^\circ \\ d &\rightarrow \alpha. \end{aligned} \tag{5.2}$$

onde  $d$  a variável que representa essa distância e  $2\pi r$  representa o perímetro da circunferência completa, correspondente a  $360^\circ$ . Assim, o valor de  $d$  em função de  $\alpha$  é calculado por:

$$d = \frac{2\pi r \alpha}{360} = \frac{\pi r \alpha}{180}. \tag{5.3}$$

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 deverão atingir simultaneamente o 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 seja percorrido 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 determinar o tempo necessário para que o robô alcance o ponto de destino, através de (5.4).

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

Determinar o tempo  $t$  permite calcular 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*.

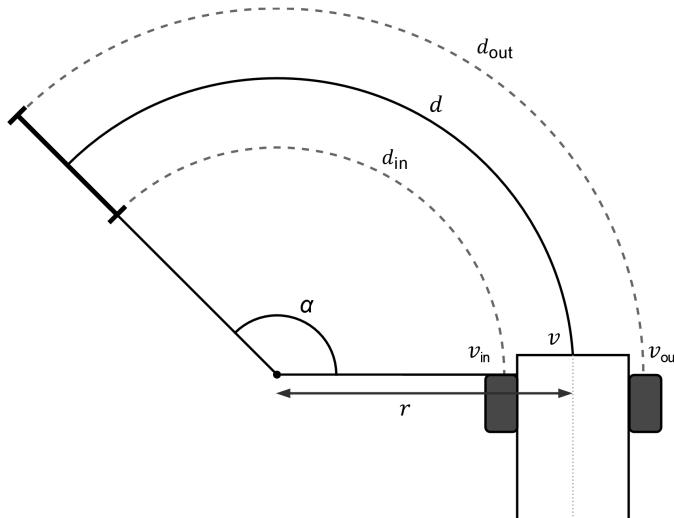


Figura 5.7: Diagrama de uma curva à esquerda

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 5.7 representa o ângulo  $\alpha$ , a distância  $d$ , a velocidade  $v$ , e os próximos valores a serem calculados, 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  $v_{out}$ .

Nesta fase é calculada a distância desde a posição de cada uma das rodas até ao ponto de destino. Para calcular os valores de distância, variáveis  $d_{in}$  e  $d_{out}$ , considera-se a variável  $\Delta$ , que corresponde a metade do comprimento do *track*:

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

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

A partir da fórmula da velocidade (5.4), foi calculada a velocidade para cada um dos motores, começando pela velocidade para o motor interno à curva (5.7), em que as variáveis  $v_0$  e  $d_0$  consistem na velocidade e distância respetivamente, calculadas anteriormente para o centro do *track* do *chassi*:

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

Nesta fase é possível simplificar a equação:

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

Ao observar a equação simplificada da velocidade do motor interno, podemos concluir facilmente que a equação correspondente para a velocidade do motor externo é calculada 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 solicitada.

#### 5.2.4 Robot

Após a implementação do movimento retilíneo e curvilíneo, foi necessário desenvolver um sistema que permitisse o controlo remoto desses movimentos utilizando a função Bluetooth do ESP32 essa classe é a `Robot` (`String name, Movement* movement`) que interliga todas as outras classes. A utilização do `BluetoothSerial`, uma biblioteca específica para comunicação Bluetooth, proporcionou a capacidade de estabelecer uma ligação sem fios 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 receção e interpretação das mensagens enviadas pelo dispositivo de controlo remoto.

As mensagens são formatadas com base numa trama específica, 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`". Para adicionar um intervalo de atraso entre movimentos utiliza-se o comando de atraso, identificado pelo carácter `d` (de *delay*), onde o comando deve ser formatado como "`d,tempo`", com o tempo definido em milissegundos.

Todos os comandos neste protótipo estão representados na Tabela 5.2.

Tabela 5.2: Tabela de comandos implementados

Comando	String
Frente	<code>f,speed,length</code>
Trás	<code>b,speed,length</code>
Esquerda	<code>l,speed,length,radius,angle</code>
Direita	<code>r,speed,length,radius,angle</code>
Atraso	<code>d,time</code>

Com a ligação 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. Além disso, esta opção permite testar todas as funcionalidades implementadas anteriormente.

### 5.3 Implementação do 2º Protótipo

Com o objetivo de desenvolver uma nova versão do 1º protótipo da base da plataforma didática, foi desenvolvida uma PCB.

O processo teve início com a conceção do circuito da placa utilizando o *software* [14], que permitiu a criação de um esquema elétrico (Anexo A). Em seguida, o *layout*

da PCB foi desenvolvido, 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) utilizando apenas a camada inferior, que foi utilizado posteriormente no processo de impressão da placa.

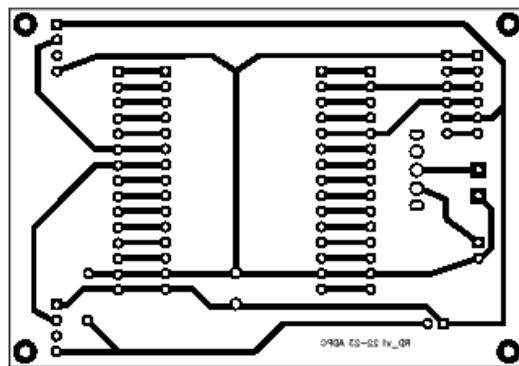


Figura 5.8: Layout da PCB

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). 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 demonstra pela Figura 5.9.

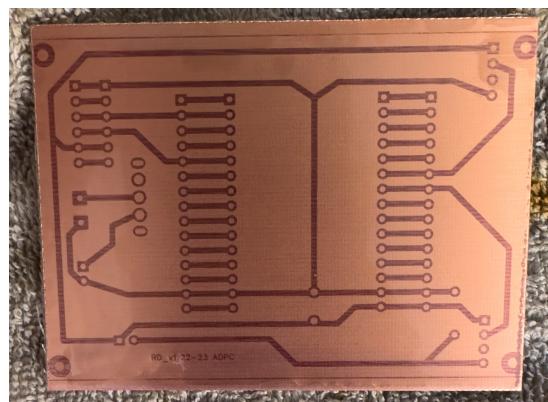


Figura 5.9: Revelação das pistas

Após a etapa de revelação, o cobre que não estava protegido pelo verniz positivo foi removido através de um banho de corrosão com percloreto de ferro. Essa ação garantiu a formação das pistas desejadas, mantendo-as isoladas e protegidas. 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 5.10.

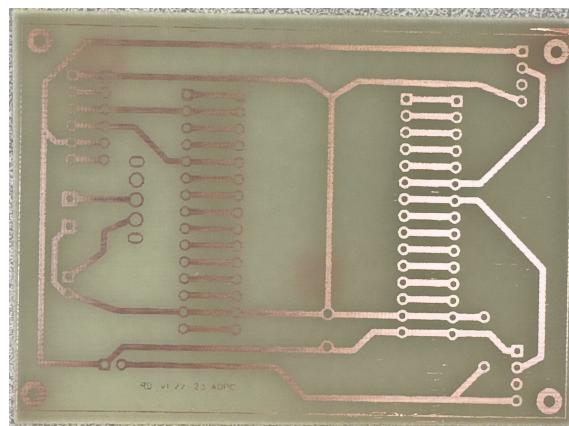


Figura 5.10: Impressão final

Por fim, utilizando uma broca específica para PCB, foram realizados os furos necessários na placa, que serão posteriormente utilizados para a instalação dos componentes eletrónicos (Figura 5.11).

O último passo envolveu a soldagem dos componentes eletrónicos na PCB e a sua montagem na plataforma didática (Figura 5.12), resultando no alcance dos objetivos previstos para o projeto.



Figura 5.11: Furação da PCB

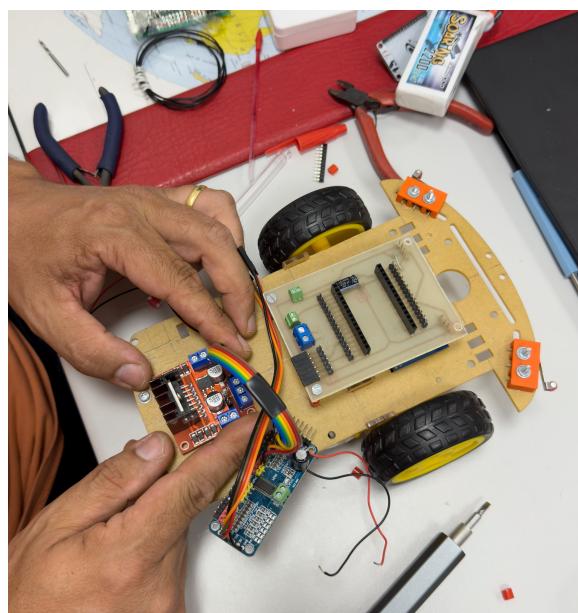


Figura 5.12: Montagem da PCB



# Validação e Testes

Com o intuito de adquirir conhecimento sobre as tecnologias envolvidas e validar as implementações desenvolvidas em pequena escala, foram realizados vários testes práticos, análise de um comportamento que originava um desvio antes da paragem do robô (Secção 6.1), implementação de um diferencial por *software* (Secção 6.2) e custos do protótipo e custos produção (Secção 6.3).

## 6.1 Desvio Antes da Paragem

Verificou-se que no 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.

## 6.2 Diferencial por Software

Quando o robô se movia em linha reta, notou-se que o mesmo começava a desviar e a realizar uma curva acentuada para a direita. Esse fenômeno é conhecido como desvio de trajetória e pode ser causado por múltiplos motivos, 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 cada *encoder* durante 5 segundos. Para minimizar possíveis interferências de inclinações ou deformações na superfície, esses testes foram realizados com o robô

suspensos no ar, sem contacto com o solo. Observou-se que os motores só iniciam o movimento com um *duty cycle* igual ou superior a 50%. Os resultados encontram-se apresentados na Tabela 6.1 e o respetivo gráfico é apresentado na Figura 6.1.

Tabela 6.1: Resultados dos testes práticos sem compensação

Duty cycle	Teste 1		Teste 2		Teste 3		Média		
	Esq.	Dir.	Esq.	Dir.	Esq.	Dir.	Esq.	Dir.	Erro [%]
50	586	555	585	561	583	557	584,67	557,67	4,61
60	665	640	668	642	666	640	666,33	640,67	3,84
70	725	697	723	699	720	696	722,67	697,33	3,51
80	736	744	760	743	762	746	761,67	744,33	1,97
90	791	782	793	784	790	783	791,33	783,00	1,05
100	860	852	856	853	849	851	855,00	852,00	0,35

A tabela apresenta os resultados de um teste de velocidade realizado em três experimentos (Teste 1, Teste 2 e Teste 3) para cada uma das configurações de velocidade dos motores. Os valores registrados são para o lado esquerdo (Esq.) e lado direito (Dir.) do teste.

A média para cada configuração de velocidade dos motores é calculada somando os valores do lado esquerdo e do lado direito e dividindo por dois. A média geral é obtida somando todas as médias das configurações de velocidade dos motores e dividindo pelo número de configurações.

A coluna "Erro" representa a diferença entre as médias do lado esquerdo e do lado direito, expressa em percentagem. O valor da diferença é calculado subtraindo a média do lado direito da média do lado esquerdo, dividindo pela média do lado direito e multiplicando por 100.

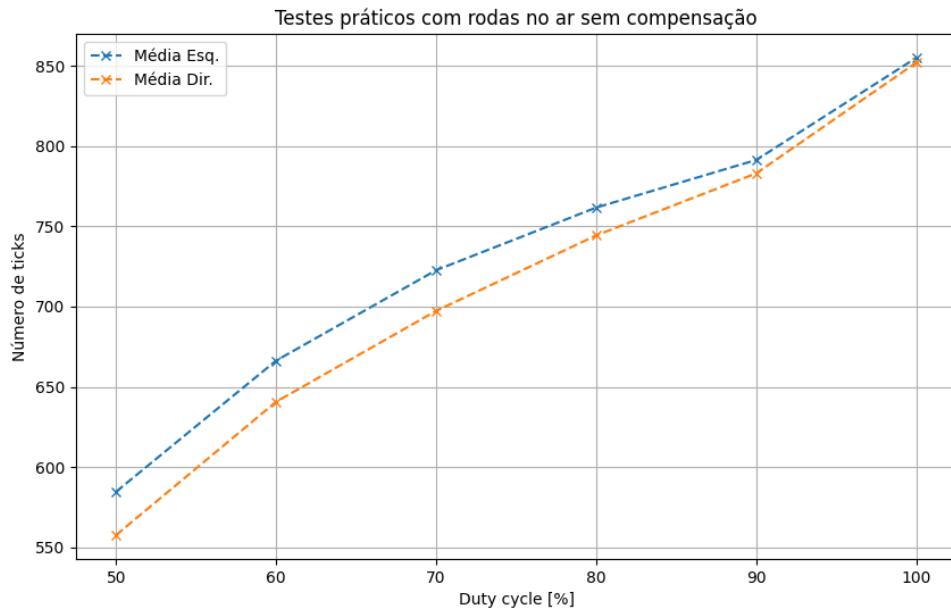


Figura 6.1: Testes práticos com rodas no ar sem compensação

Conforme era esperado, os motores apresentaram uma diferença no número de ticks. Essa diferença é cada vez mais significativa 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 um diferencial por *software*, consistindo num sistema de compensação de direção, o qual ajusta a velocidade de cada um dos motores conforme o movimento desejado.

O sistema implementado funciona da seguinte forma: 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 é realizado o ajuste da velocidade dos motores, são calculadas as diferenças de *ticks* entre a amostra atual e a anterior, permitindo obter os *ticks* contabilizados 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. O algoritmo desenvolvido encontra-se descrito 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()←
11        ;
12    while ((currLeftCounter < leftTarget || currRightCounter←
13        < rightTarget))
14    {
15        unsigned long currTime = millis();
16        if (currTime >= timeout)
17        {
18            currLeftCounter = _motors[MOTOR_LEFT].getCounter←
19                ();
20            currRightCounter = _motors[MOTOR_RIGHT].←
21                getCounter();
22            int diffLeft = currLeftCounter - dataLine[←
23                indxDataLine].ticksLeft;
24            int diffRight = currRightCounter - dataLine[←
25                indxDataLine].ticksRight;
26            timeout = currTime + _PERIOD;
27            int maxDiff = max(diffLeft, diffRight);
28            int minDiff = min(diffLeft, diffRight);
29            float ratio = 0.0f;
30            float motorDif;
31            float rightSpeed, leftSpeed;
32            if (minDiff > 0)
33            {
34                ratio = (float)(maxDiff) / (float)(minDiff);
35                float ratioAux = (ratio - 1.0) / 2.0;
36                float ratioAdd = 1.0 + ratioAux;
37                float ratioSub = 1.0 - ratioAux;
38            }
39        }
40    }
41 }
```

```

30         if (diffLeft > diffRight)
31     {
32         rightSpeed = _motors[MOTOR_RIGHT].getPWM() * ratioAdd;
33         leftSpeed = _motors[MOTOR_LEFT].getPWM() * ratioSub;
34     }
35     else
36     {
37         rightSpeed = _motors[MOTOR_RIGHT].getPWM() * ratioSub;
38         leftSpeed = _motors[MOTOR_LEFT].getPWM() * ratioAdd;
39     }
40     float motorDif_right = rightSpeed < _MIN_PWM ? _MIN_PWM - rightSpeed : (rightSpeed > _MAX_PWM ? _MAX_PWM - rightSpeed : 0);
41     float motorDif_left = leftSpeed < _MIN_PWM ? _MIN_PWM - leftSpeed : (leftSpeed > _MAX_PWM ? _MAX_PWM - leftSpeed : 0);
42     rightSpeed = rightSpeed + motorDif_left < _MIN_PWM ? _MIN_PWM : (rightSpeed + motorDif_left > _MAX_PWM ? _MAX_PWM : rightSpeed + motorDif_left);
43     leftSpeed = leftSpeed + motorDif_right < _MIN_PWM ? _MIN_PWM : (leftSpeed + motorDif_right > _MAX_PWM ? _MAX_PWM : leftSpeed + motorDif_right);
44     _motors[MOTOR_RIGHT].setPWM(rightSpeed);
45     _motors[MOTOR_LEFT].setPWM(leftSpeed);
46     ++indxDataLine;
47     dataLine[indxDataLine].pwmLeft = _motors[MOTOR_LEFT].getPWM();
48     dataLine[indxDataLine].pwmRight = _motors[MOTOR_RIGHT].getPWM();
49     dataLine[indxDataLine].ticksLeft = currLeftCounter;
50     dataLine[indxDataLine].ticksRight = currRightCounter;
51     dataLine[indxDataLine].ratio = ratio;
52   }
53 }
54 }
55 }
```

De seguida, o mesmo teste prático foi repetido, com o robô com as rodas no ar, durante 5 segundos, mas desta vez com o algoritmo de compensação implementado

no sistema. Os resultados encontram-se apresentados na Tabela 6.2 e na Figura 6.2.

Tabela 6.2: Resultados dos testes práticos com compensação

Duty cycle	Teste 1		Teste 2		Teste 3		Média		
	Esq.	Dir.	Esq.	Dir.	Esq.	Dir.	Esq.	Dir.	Erro [%]
50	658	657	673	670	669	670	666,66	665,66	0,15
60	678	677	678	678	678	678	678,00	677,66	0,05
70	704	702	698	698	703	702	701,67	700,67	0,14
80	719	720	717	716	716	715	717,67	717,00	0,09
90	731	729	730	729	730	730	730,33	729,33	0,14
100	727	728	725	724	727	727	726,33	726,33	0,14

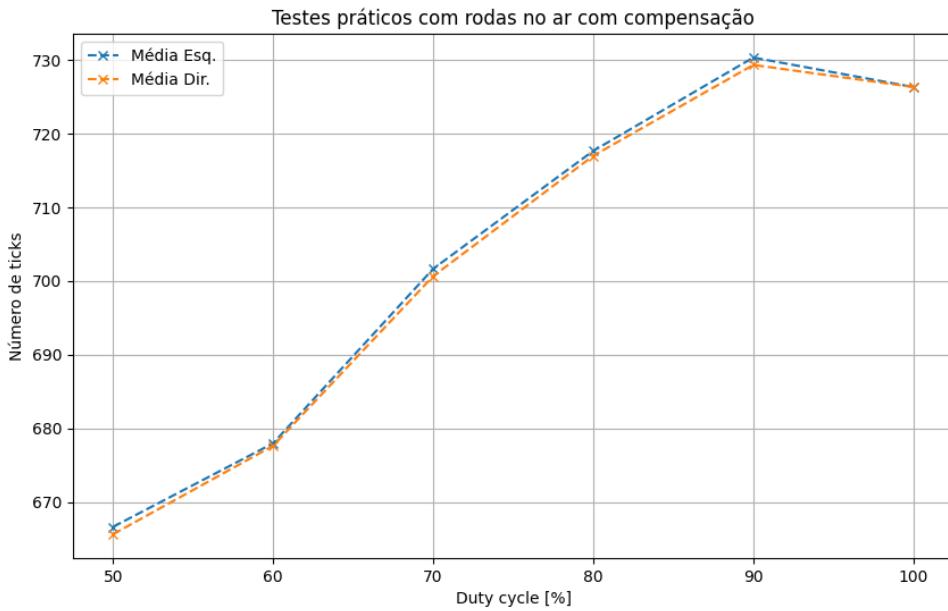


Figura 6.2: Testes práticos com rodas no ar com compensação

É de notar que a diferença entre médias de *ticks* diminuiu significativamente após a implementação e inclusão do algoritmo de compensação de velocidade. No entanto, há uma redução no número de *ticks* para a velocidade máxima. Esta redução ocorre devido à necessidade de abrandar o motor com maior rotação, pois o controlo por PWM satura nos 100%. Assim, de modo a equilibrar o movimento o motor com menor rotação é colocado na velocidade máxima (100% no PWM),

enquanto o outro motor vai reduzindo o PWM, reduzindo o número de *ticks* por intervalo de tempo.

No repositório associado ao projeto [15], na diretoria 04\_Teste, encontram-se disponibilizados um conjunto de videos representativos da execução dos movimentos que foram enviados ao robô. Estes videos permitem verificar a compensação por *software* em execução.

## 6.3 Custos do Protótipo e Custo de Produção

De forma a calcular e comparar o preço da plataforma desenvolvida com a plataforma da LEGO® MINDSTORMS® [2] utilizada nas aulas de Fundamentos de Sistemas Operativos [16], foi concebido uma BOM de forma a calcular os custos do protótipo criado, bem como de uma eventual produção de 1000 unidades. Respetivamente na Tabela 6.3 e a Tabela 6.4.

Tabela 6.3: BOM protótipo

Produto	Preço [€]	Quantidade
Kit para carro	14,90	1
Motor	3,44	2
Bateria	8,05	1
Encoder	8,14	2
Controlador de Motor	2,15	1
SONAR	2,25	1
Sensor de Toque	3,46	2
Microcontrolador	15,35	1
Expansor	13,40	1
PCB	0.8	1
<b>Total</b>	<b>86,98</b>	

Tabela 6.4: BOM produção

<b>Produto</b>	<b>Preço [€]</b>	<b>Quantidade</b>
Kit para carro	11,92	1000
Motor	2,75	2000
Bateria	6,44	1000
Encoder	6,51	2000
Controlador de Motor	1,72	1000
SONAR	1,80	1000
Sensor de Toque	2,77	2000
Microcontrolador	12,28	1000
Expansor	10,72	1000
PCB	0,22	1000
<b>Total</b>	<b>69 167,50</b>	
<b>Total por unidade</b>	<b>69.17</b>	

Após a análise destes custos, é de notar que o preço de um robô didático com as mesmas características do desenvolvido para o projeto é bastante mais barato que o antigo, e já descontinuado, LEGO® MINDSTORMS® [2] que custava cerca de 500 euros. É de notar também que produzir 1000 unidades da plataforma proposta ficaria cerca de 20% mais barato. Assim, conclui-se que o robô didático aqui proposto é uma opção viável e menos onerosa que um robô comercial, que poderá, por exemplo, vir a substituir os robôs utilizados na Unidade Curricular de Fundamentos de Sistemas Operativos [16].

# Conclusões e Trabalho Futuro

Durante a realização deste projeto, foram estabelecidos objetivos que abrangiam desde a pesquisa à 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/telemóvel. Também foi desenvolvida uma PCB para facilitar a montagem do sistema.

Os resultados obtidos com a implementação e validação do sistema atingiram os objetivos propostos. A plataforma demonstrou ser capaz de realizar movimentos no espaço 2D, além de permitir o controlo remoto através de comandos enviados pelo computador/telemóvel. O controlo por *software* implementado garante também que o robô cumpre movimentos retilíneos com um erro máximo de 0.15%. Os trabalhos futuros do projeto são mais ambiciosos e darão maior potencialidade ao projeto.

Propõe-se como próximas etapas de desenvolvimento:

- A criação de uma aplicação dedicada 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;
- A criação de uma biblioteca em Java para controlar o robô: A criação desta biblioteca permitirá que outros programadores integrem facilmente a funcionalidade do robô nos seus próprios projetos e aplicações;
- A 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

## Capítulo 7. Conclusões e Trabalho Futuro

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, 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.

---

# Bibliografia

- [1] Repositório GitHub. Projeto-robot-didatico. <https://github.com/AlexFigas/Projeto-Robot-Didatico>, 07 2023.
- [2] LEGO®. LEGO® MINDSTORMS® EV3. <https://www.lego.com/en-us/product/lego-mindstorms-ev3-31313>. [Online: Acedido em junho 2023].
- [3] Jorge Pais. Aulas de computação física, 2021.
- [4] Carolina Neves. Projeto final de curso - Robô, Novembro 2020.
- [5] Mauser.pt. Kit para carro robot 2wd c/ motores e rodas. [https://mauser.pt/catalog/product\\_info.php?products\\_id=096-7641](https://mauser.pt/catalog/product_info.php?products_id=096-7641). [Online: Acedido em junho 2023].
- [6] Mauser.pt. Motor com roda para carros robóticos - joy-it. [https://mauser.pt/catalog/product\\_info.php?products\\_id=096-7645](https://mauser.pt/catalog/product_info.php?products_id=096-7645). [Online: Acedido em junho 2023].
- [7] 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-xt60-plug.html>. [Online: Acedido em junho 2023].
- [8] Mauser.pt. Módulo sensor de velocidade por infravermelhos (lm393) para arduino - whadda wpse347. [https://mauser.pt/catalog/product\\_info.php?products\\_id=096-3881](https://mauser.pt/catalog/product_info.php?products_id=096-3881). [Online: Acedido em junho 2023].
- [9] Mauser.pt. L298n dc motor driver module. [https://mauser.pt/catalog/product\\_info.php?products\\_id=096-6807](https://mauser.pt/catalog/product_info.php?products_id=096-6807). [Online: Acedido em junho 2023].
- [10] Botnroll.com. 16-canais 12-bit pwm/servo driver - i2c interface - pca9685. <https://www.botnroll.com/pt/controladores/3681-16-channel-12-bit-pwm-servo-driver-i2c-interface-pca9685.html>. [Online: Acedido em junho 2023].

- [11] Mauser.pt. Sensor ultrassónico (hc-sr04) compatível com arduino. [https://mauser.pt/catalog/product\\_info.php?products\\_id=096-6220](https://mauser.pt/catalog/product_info.php?products_id=096-6220). [Online: Acedido em junho 2023].
- [12] 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](https://mauser.pt/catalog/product_info.php?cPath=324_1401_707&products_id=010-1429). [Online: Acedido em junho 2023].
- [13] Mauser.pt. Esp32 wroom-32 devkit v1. [https://mauser.pt/catalog/product\\_info.php?products\\_id=096-7620](https://mauser.pt/catalog/product_info.php?products_id=096-7620). [Online: Acedido em junho 2023].
- [14] EasyEDA.com. <https://easyeda.com/>. [Online: Acedido em junho 2023].
- [15] Repositório do Projeto. Prj\_27\_48577\_48579. [https://iselpt-my.sharepoint.com/:f/g/personal/a48577\\_alunos\\_isel\\_pt/EmQk8Mx4q9dJv-LgF1nNAkIBWHD15P8yUi05I-vGnrZ5Iw?e=xFTV1D](https://iselpt-my.sharepoint.com/:f/g/personal/a48577_alunos_isel_pt/EmQk8Mx4q9dJv-LgF1nNAkIBWHD15P8yUi05I-vGnrZ5Iw?e=xFTV1D), 07 2023.
- [16] Jorge Pais. Aulas de fundamentos de sistemas operativos, 2022.

# Esquema elétrico da PCB do 2º protótipo

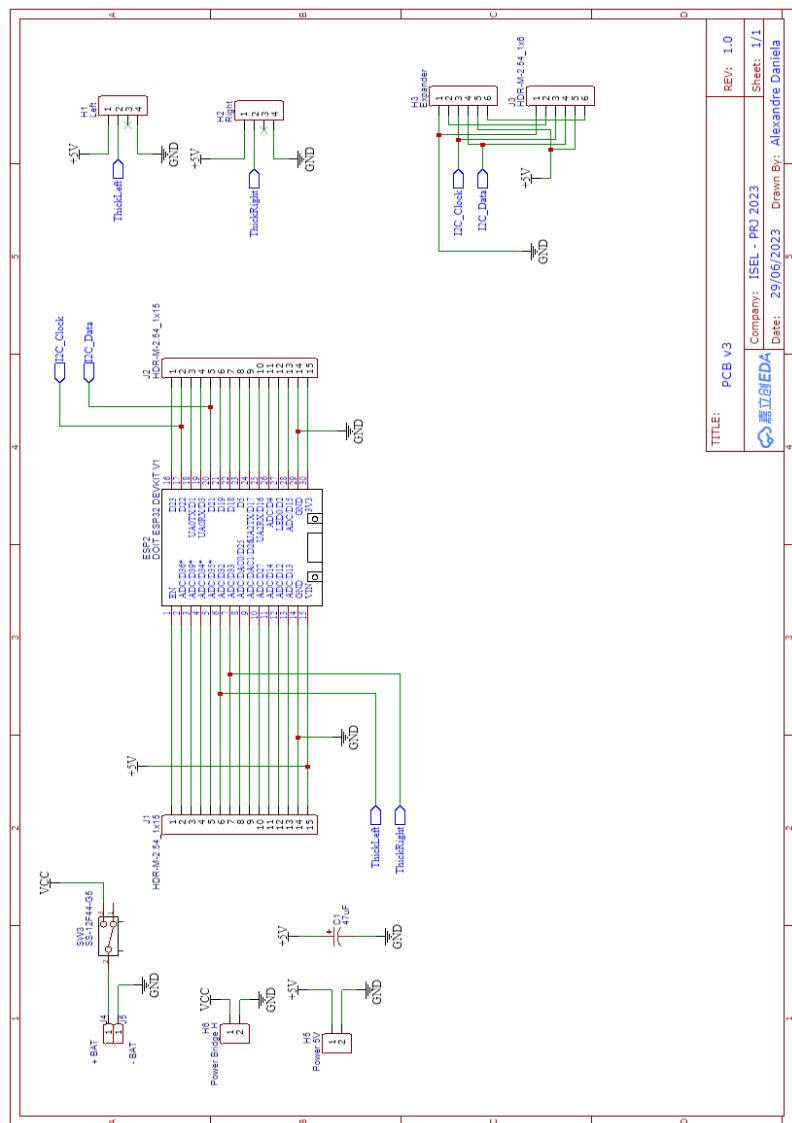


Figura A.1: Esquema elétrico da PCB do 2º protótipo

Apêndice A. Esquema elétrico da PCB do 2º protótipo

# Diagrama de Classes UML

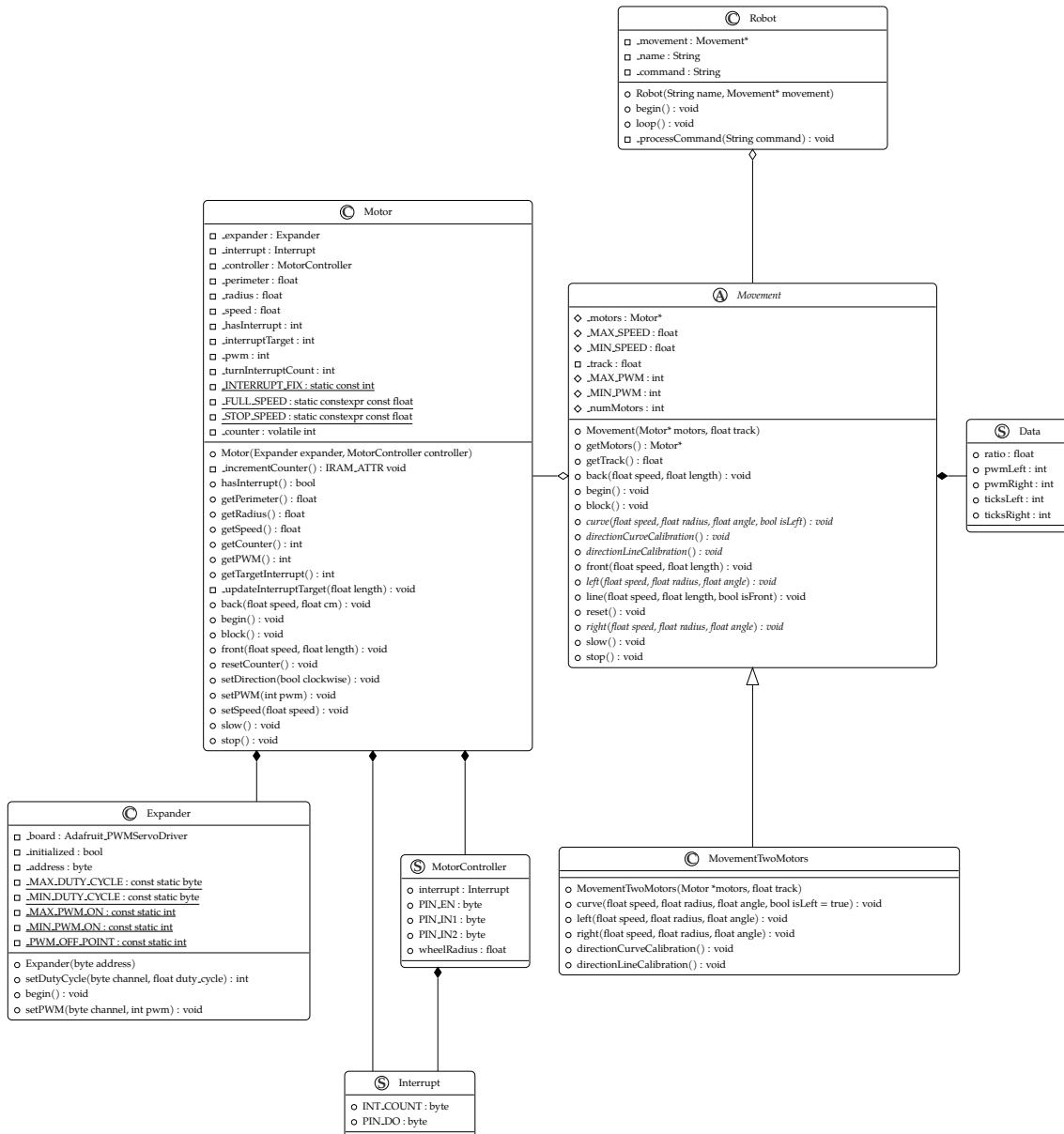


Figura B.1: Diagrama de classes UML completo

## Apêndice B. Diagrama de Classes UML