



## PROVA DE APTIDÃO PROFISSIONAL

TÍTULO DA PROVA:	ROBÔ DISTRIBUIDOR DE CARGA POR COR		
NOME DO ALUNO:	JOÃO MIGUEL GAMEIRO OLIVEIRA	NÚMERO:	4334
CURSO:	GESTÃO DE EQUIPAMENTOS INFORMÁTICOS	TRIÉNIO:	21.24
ORIENTADOR DE PROJETO:	PAULO SILVA		
DATA DO RELATÓRIO:	27/03/24		

# **Prova de Aptidão Profissional**

**João Miguel Gameiro Oliveira**

**Curso Profissional de Técnico de  
GESTÃO DE EQUIPAMENTOS INFORMÁTICOS  
2021 - 2024**



**ROBÔ DISTRIBUIDOR**



## Agradecimentos

Gostaria de começar por agradecer à Escola Profissional de Ourém por disponibilizar as instalações e os equipamentos necessários para a execução deste projeto. Sem o seu apoio, este projeto não teria sido possível.

Ao nosso orientador de PAP e professor, Paulo Silva, o meu mais sincero agradecimento pelo apoio, orientação, motivação e dedicação durante todo o desenvolvimento do meu projeto. A sua ajuda foi preciosa e fundamental para o meu sucesso.

Aos meus colegas de turma agradeço o apoio constante e a amizade que me proporcionaram durante este percurso. A vossa presença foi um grande incentivo. Ao professor Charly Silva e à professora Fátima Lucas agradeço a disponibilidade para a revisão do meu relatório e pelas suas valiosas sugestões.



# ROBÔ DISTRIBUIDOR





## ÍNDICE

A. Índice de imagens .....	V
B. Índice de tabelas .....	VIII
C. Lista de abreviaturas .....	IX
1. Introdução .....	1
1.1 Nota prévia .....	1
1.2 Objetivos a atingir .....	1
1.3 Breve Descrição .....	1
1.3.1 Logótipo .....	2
2. Memória descritiva e justificativa do projeto.....	3
2.1 Enquadramento e motivação do problema .....	3
2.1.1 Motivação para o projeto .....	3
2.1.2 Levantamento das necessidades .....	3
2.2 Especificações do projeto realizado .....	3
2.2.1 Requisitos do projeto .....	4
2.2.1.1 Descrição do design do protótipo .....	4
2.2.1.2 Material usado na estrutura .....	8
2.2.1.2.1 Software usado na impressão 3D .....	9
2.3 Seleção da plataforma controladora.....	10
2.3.1.1 Arduíno Uno .....	10
2.3.2 Aplicação de desenvolvimento escolhida .....	11
2.3.2.1 Arduíno IDE .....	11
2.3.2.2 Código no Arduíno .....	12
2.4 Explicação do código .....	13
2.4.1 Código seguidor de linha usando PID .....	13
2.4.1.1 O que é o PID e o seu funcionamento .....	13
2.4.1.1.1 Cálculo de erro usando PID.....	16
2.4.2 Leitura e classificação de cores.....	18
2.4.3 Escolha dos trajetos .....	19





2.5	Componentes Utilizados no Projeto.....	21
2.5.1	Arduíno Uno .....	21
2.5.2	Ponte H L298N .....	22
2.5.3	Sensor de cor RGB TCS3200 .....	23
2.5.4	Módulo sensor infravermelho TCRT5000 .....	24
2.5.5	Servo Motor SG90 9g Tower Pro.....	25
2.5.6	Motores TT 3V ~ 12V DC.....	26
2.5.7	Placa de Expansão para Arduíno Uno R3 .....	27
2.5.8	Bateria de Lítio Recarregável .....	28
2.5.9	Fios DuPont .....	28
2.6	Desenvolvimento do Projeto.....	29
2.6.1	Etapas do desenvolvimento do projeto .....	29
2.6.1.1	Evolução Do Projeto.....	29
2.6.1.1.1	Fase Inicial.....	29
2.6.1.1.2	Fase Intermédia .....	39
2.6.1.1.2.1	Construção do Projeto.....	39
2.6.1.1.2.2	Programação do Projeto.....	41
2.6.1.1.3	Fase Final .....	45
2.7	Orçamento.....	46
2.7.1	Orçamento dos componentes utilizados .....	46
2.7.2	Horas de mão-de-obra .....	47
2.7.3	Custos totais do projeto.....	49
3.	Conclusão e Reflexão do Projeto .....	50
3.1	Análise de Planeamentos .....	50
3.1.1	Planeamento Inicial.....	50
3.1.2	Planeamento final .....	50
3.1.3	Conclusão sobre o planeamento .....	50
3.2	Conclusão do trabalho de PAP realizado.....	51



3.3	Sugestões/Melhorias Futuras.....	52
4.	Netgrafia .....	53
5.	Anexos.....	54
5.1	Repositório dos anexos do projeto: .....	54
5.2	Esboços e Modelos 3D.....	54
5.3	Documentos do projeto .....	55
5.4	Datasheet dos componentes utilizados .....	55



## A. ÍNDICE DE IMAGENS

Figura 1 - Logótipo.....	2
Figura 2 - Whadda Wpb100.....	4
Figura 3 -Logótipo Fusion 360 .....	4
Figura 4 - Logótipo Autodesk .....	4
Figura 5 - Chassi modelo 3D .....	5
Figura 6 - Basculante modelo 3D.....	5
Figura 7 - Suporte do basculante .....	6
Figura 8 – Veio.....	6
Figura 9 - Suporte do motor modelo 3D .....	6
Figura 10 - Proteção do motor modelo 3D.....	6
Figura 11 - Suporte dos módulos sensores infravermelhos modelo 3D .....	7
Figura 12 - Proteção do suporte dos módulos sensores infravermelhos modelo 3D.....	7
Figura 13 - Projeto com suportes e componentes modelos .....	8
Figura 14 - Impressões 3D .....	8
Figura 15 - Logótipo UltiMaker.....	9
Figura 16 - UltiMaker Cura software .....	9
Figura 17 - Software UltiMaker Cura.....	9
Figura 18 - Datasheet de Arduíno Uno.....	10
Figura 19 - Arduíno IDE.....	11
Figura 20 - Código Qr , Código no Arduino ( <a href="https://bit.ly/493ZbwE">Https://bit.ly/493ZbwE</a> ).....	12
Figura 21 - Diagrama de cálculos do PID .....	14
Figura 22 - Excerto de código, variáveis do PID .....	15
Figura 23 - Excerto de código, funções PID .....	16
Figura 24 - Excerto de código, convertor binário PID.....	16
Figura 25 - Excerto de código, determinação de erro de leituras PID .....	16
Figura 26 - Excerto de código, velocidade e erro PID.....	<b>Erro! Marcador não definido.</b>
Figura 27 - Excerto de código, leitura de cor.....	18
Figura 28 - Excerto de código, escolha de trajetos .....	19
Figura 29 - Excerto de código, função da cor vermelha.....	19
Figura 30 - Ilustração da numeração de interseções .....	20
Figura 31 - Controlador Ponte H L298N .....	22
Figura 32 - Sensor de Cor RGB TCS3200.....	23
Figura 33 - Módulo Sensor Infravermelhos TCRT5000.....	24



Figura 34 - Servo Motor SG90 9g Tower Pro .....	25
Figura 35 - Motor TT 3V~12V .....	26
Figura 36 - Placa de Expansão para Arduino Uno R3 .....	27
Figura 37 - Bateria de Lítio Recarregável.....	28
Figura 38 - Fios DuPont .....	28
Figura 39 - Suporte de módulos sensores infravermelhos defeituoso .....	30
Figura 40 - Suporte de módulos sensores infravermelhos corrigido .....	30
Figura 41 - Suporte de basculante danificado.....	31
Figura 42 - Sistema basculante defeituoso .....	31
Figura 43 - Sistema basculante melhorado .....	32
Figura 44 - Sistema basculante defeituoso .....	32
Figura 45 - Módulo Sensor Infravermelhos, 8 sensores, teste.....	33
Figura 46 - Módulo Sensor Infravermelho Tcrt 5000, teste .....	33
Figura 47 - Código teste do Controlador Ponte H .....	34
Figura 48 - Teste com Controlador Ponte H e motor TT .....	35
Figura 49 - Circuito de teste com Controlador Ponte H e motor TT .....	35
Figura 50 - - Testes com Módulo Sensor Infravermelhos .....	36
Figura 51 - Código teste do Módulo Sensor Infravermelhos.....	36
Figura 52 - Circuito teste com Módulo Sensor de Infravermelhos .....	37
Figura 53 - Código teste do Módulo Sensor de Cor RGB.....	37
Figura 54 - Teste do Módulo Sensor de Cor RGB .....	38
Figura 55 - Circuito teste com Módulo Sensor de Cor RGB.....	38
Figura 56 - Motores TT e suportes .....	39
Figura 57 - Placas controladoras afixadas no chassis .....	39
Figura 58 - Afixação dos Módulos Sensores Infravermelhos no suporte.....	40
Figura 59 – Processo de montagem final do sistema basculante .....	40
Figura 60 - Demonstração da organização e preparação de cabos .....	41
Figura 61 - Excerto de código, Variáveis Declaradas.....	42
Figura 62 - Excerto de código, Função mission() .....	43
Figura 63 - Excerto de código, Função readSens() .....	44
Figura 64 - Circuito geral do Projeto .....	45
Figura 65 - Projeto Final .....	45
Figura 66 - Reforma do Aspetto Físico do Robô .....	52
Figura 67 - Código Qr , Repositório ( <a href="https://bit.ly/3ITxSL2">https://bit.ly/3ITxSL2</a> )......	54



Figura 68 - Código Qr , Esboços e Modelos 3D ( <a href="https://bit.ly/3PCcGgl">https://bit.ly/3PCcGgl</a> ).....	54
Figura 69 - Código Qr , Documentos do projeto ( <a href="https://bit.ly/3xd5mBl">https://bit.ly/3xd5mBl</a> ).....	55
Figura 70 - Código Qr , Datasheet ( <a href="https://bit.ly/3xcLsGI">https://bit.ly/3xcLsGI</a> ).....	55



## B. ÍNDICE DE TABELAS

Tabela 1 - Orçamento dos componentes utilizados.....	46
Tabela 2 - Horas de mão de obra .....	Erro! Marcador não definido.
Tabela 3 - Custos totais do projeto .....	Erro! Marcador não definido.



### C. LISTA DE ABREVIATURAS

- PAP – *Prova de Aptidão Profissional*
- C/C++ - *Linguagem de programação de alto nível*
- PID - *Proporcional-Integral-Derivativo*
- 3D – *Tridimensional*
- RGB - *Vermelho (Red), o Verde (Green) e o Azul (Blue)*



## 1. INTRODUÇÃO

A Prova de Aptidão Profissional (PAP) é um projeto prático de final de curso, com a correspondente apresentação e defesa perante um júri, sendo obrigatória para a obtenção do diploma de qualificação profissional no final do curso. Concentra-se em temas e problemas relevantes para os contextos de trabalho e é realizada com a orientação e acompanhamento de um ou mais professores.

A PAP implica a planificação e execução de um projeto prático com base nas competências mais significativas do curso, neste caso, do curso Profissional de Técnico de Gestão de Equipamentos Informáticos.

### 1.1 NOTA PRÉVIA

A minha Prova de Aptidão Profissional consiste num projeto de criação de um Robô distribuidor de carga desenhado para ambientes fechados e industrializados.

O motivo pelo qual escolhi este tema de projeto prende-se com o meu gosto por componentes de programação e eletrónica. A área de desenho e modelação 3D é também uma componente que me fascina, e assim aproveitei para desenvolver um projeto que abrangesse estas três áreas.

### 1.2 OBJETIVOS A ATINGIR

O principal objetivo deste projeto é criar um distribuidor de embalagens separadas pela sua cor, utilizando para isso um robô seguidor de linha com um Arduíno UNO, usando 6 sensores de linha infravermelhos analógicos. O robô será capaz de identificar e classificar as cores: vermelho, verde e azul enquanto se move ao longo de uma trajetória pré-definida, deixando cada caixa no sítio designado pelas cores definidas.

### 1.3 BREVE DESCRIÇÃO

O projeto consiste, neste momento, em ter um sensor RGB TCS3200, um Whadda Wpb100 (Cópia de Arduíno Uno Rv3), Placa de expansão para a placa programadora, cabo DuPont, um Micro servo SG90 9g, 2 Motores TT 3V ~ 12V DC, Controlador Ponte H L298N, 6 Módulos de sensores infravermelhos TCRT5000. Todos estes elementos são essenciais para o bom funcionamento do robô distribuidor.

Durante cerca de um mês, realizei um trabalho de investigação sobre os componentes ideais para usar no projeto. Após muito tempo utilizado nessa pesquisa e no estudo dos componentes, consegui escolhê-los com precisão para o correto funcionamento do robô distribuidor e para a fase de montagem.

### **1.3.1 LOGÓTIPO**

Um logótipo é uma disposição visual de elementos gráficos, combinados com cores e uma variedade de designs, usado para identificar e representar uma marca ou uma empresa específica.

Neste caso, o logotipo em questão incorpora três cores principais: azul, vermelho e verde. Estas cores foram escolhidas para refletir as tonalidades detetadas pelo robô enquanto ele traça o seu caminho.



# **ROBÔ DISTRIBUIDOR**

*Figura 1 - Logótipo*



## 2. MEMÓRIA DESCRIPTIVA E JUSTIFICATIVA DO PROJETO

Neste capítulo irei explicar o tema da Prova de Aptidão Profissional, que não só incluirá a apresentação das várias soluções estudadas, mas também uma descrição detalhada do projeto em questão. Além disso, serão demonstrados todos os passos realizados ao longo do trabalho, realçando os principais desafios enfrentados, as decisões tomadas e as etapas percorridas até à sua conclusão.

### 2.1 ENQUADRAMENTO E MOTIVAÇÃO DO PROBLEMA

Segue-se a análise da escolha do projeto, na qual serão abordados os motivos que influenciaram essa decisão, bem como os fatores que me levaram a escolher este tema específico. Serão explorados detalhes sobre as razões subjacentes à seleção do projeto, destacando a sua relevância e o potencial contributo para a área de estudo.

#### 2.1.1 MOTIVAÇÃO PARA O PROJETO

Motivado pelo meu interesse na área de eletrónica, programação, robótica, modelação e Impressão 3D, bem como pelo desejo de aprimorar as minhas competências em áreas menos exploradas no Curso de Gestão de Equipamentos Informáticos, decidi abraçar um projeto desafiante e inovador. Com esse propósito, concentrei-me em estudar e desenvolver uma solução que integrasse as áreas mencionadas.

A ideia de explorar o ambiente fabril e a sua capacidade de armazenamento surgiu como ponto de partida. Após uma pesquisa aprofundada, comecei a conceber métodos práticos para otimizar o armazenamento de caixas, visando reduzir a dependência da mão-de-obra humana nessas tarefas.

#### 2.1.2 LEVANTAMENTO DAS NECESSIDADES

Os produtos usados neste projeto podem ser encontrados em várias lojas de eletrónica, embora alguns tenham custos um pouco elevados.

Apesar do tempo de entrega ser mais demorado, todos os produtos foram comprados na China para me permitir economizar o custo do projeto. Atualmente as entregas já estão mais controladas e consegui seguir as encomendas com a previsão da data de entrega, o que é muito importante que estejam disponíveis nas datas previstas.

## 2.2 ESPECIFICAÇÕES DO PROJETO REALIZADO

Neste ponto vou apresentar os requisitos que achei necessários como mínimos para a realização de um projeto acessível e funcional.

## **2.2.1 REQUISITOS DO PROJETO**

Desde o início do projeto, estabeleci como requisito mínimo a criação de um código capaz de seguir uma linha preta de forma precisa e eficiente, independentemente do ambiente e das condições de luminosidade.

Para atender a esse requisito, optei por desenvolver o código de programação em C++ para o hardware Whadda Wpb100 (Cópia de Arduíno Uno R3).

Essa escolha foi fundamental para garantir um desempenho ótimo e uma leitura da linha preta com baixa margem de erro.

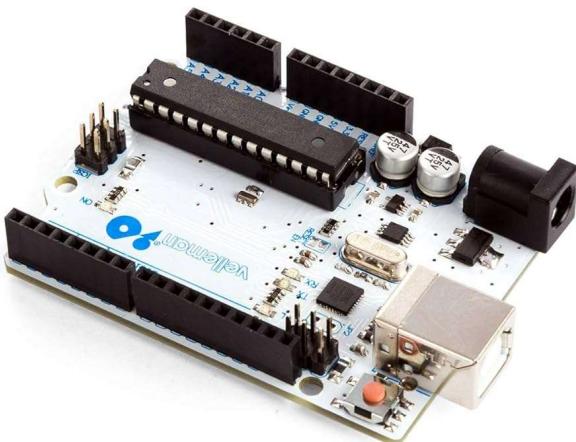


Figura 2 - Whadda Wpb100

### **2.2.1.1 DESCRIÇÃO DO DESIGN DO PROTÓTIPO**

Durante dois meses dediquei-me ao estudo e design do protótipo para garantir o ótimo funcionamento do robô, abrangendo desde o aspetto visual até ao desempenho e movimentação. Utilizei o software Autodesk Fusion 360 para criar um projeto em 3D detalhado e funcional.



Fusion 360



*Figura 4 - Logótipo Autodesk*

Comecei por projetar um chassi robusto para o robô, utilizando uma espessura de 3mm para garantir resistência e durabilidade. A sua largura é de 150mm e o seu comprimento de 218mm. Desenvolvi suportes estratégicos para fixar o Arduino Uno e Controlador Ponte H L298N no chassi, garantindo uma montagem estável e segura. Além disso, projetei aberturas precisas para a passagem de cabos, facilitando uma conexão fácil e organizada.

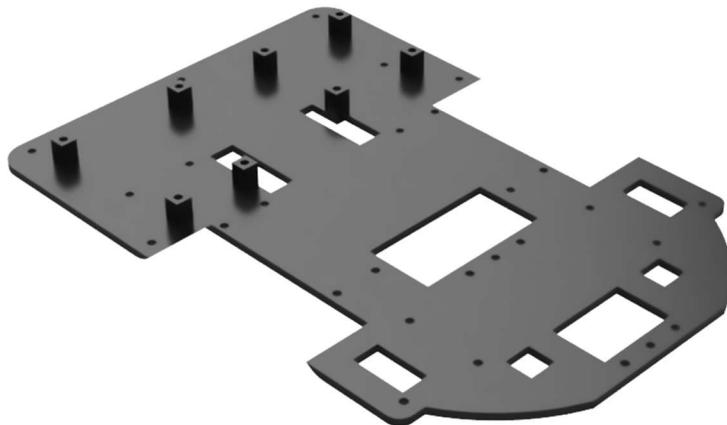


Figura 5 - Chassi modelo 3D

Em seguida dei início ao estudo do mecanismo para o despejo da carga e a leitura da cor da carga, utilizando um micro servo num ponto estratégico. Comecei por desenhar a basculante de descarga, onde a caixa se encaixa e onde será feita a leitura da cor.

Neste basculante projetei os suportes essenciais para a fixação do Micro Servo SG90 9g, que desempenha um papel fundamental no funcionamento do sistema, bem como o suporte para o Sensor RGB TCS3200. Também incluí aberturas para a saída dos cabos, garantindo uma organização adequada e facilitando a conexão dos componentes. Os seguintes esboços estão representados em milímetros.

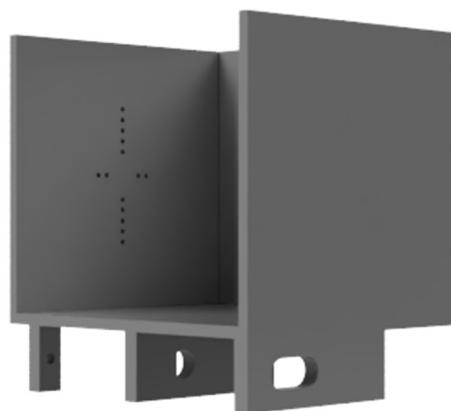


Figura 6 - Basculante modelo 3D



Depois desenhei o suporte para a peça basculante para que possa ser fixada firmemente no chassi do robô. Desenhei uma fenda para o encaixe do veio. O veio é uma peça importante, sem ele a caixa ficaria desnivelada devido à instabilidade da fixação do Micro servo SG90 9g.



Figura 7 - Suporte do basculante modelo 3D



Figura 8 – Veio modelo 3D

Depois desta tarefa foi elaborado o suporte para os dois motores, levando em consideração a sua fixação ao chassi do robô por meio de parafusos M3. O suporte incluiu a criação de aberturas estratégicas no suporte, permitindo uma fixação robusta e fácil dos motores.

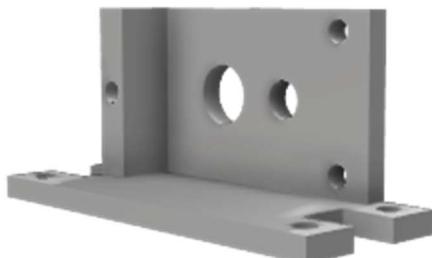


Figura 9 - Suporte do motor modelo 3D

Para completar o suporte dos motores, foi elaborado uma proteção para os motores, com o objetivo de evitar danos nos motores e assegurar o correto funcionamento do mesmo.

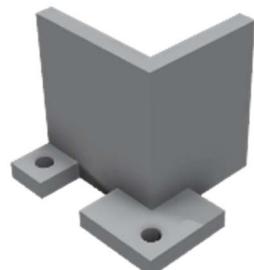


Figura 10 - Proteção do motor modelo 3D



A seguir foi projetado o suporte para os 6 módulos sensores infravermelhos, fixando-os com parafusos M3. O suporte possui aberturas para a eventual passagem de cabos, se necessário. Para garantir uma base resistente, foram adicionadas curvas cónicas em ambos os lados, proporcionando estabilidade ao suporte.



Figura 11 - Suporte dos módulos sensores infravermelhos modelo 3D

Em seguida, foi projetada uma peça para proteger os 6 Módulos Sensores Infravermelhos e protegê-los da influência da luz ambiente, o que contribuirá para o ótimo funcionamento dos Módulos Sensores Infravermelhos. Esta peça foi desenhada para encaixar firmemente no suporte dos módulos e é fixada ao suporte com outra peça posterior, utilizando parafusos M2.



Figura 12 - Proteção do suporte dos módulos sensores infravermelhos modelo 3D

Após a conclusão do desenho das peças do robô, estas foram integradas ao chassi juntamente com os componentes projetados, conferindo uma aparência mais realista e definida ao conjunto do robô.

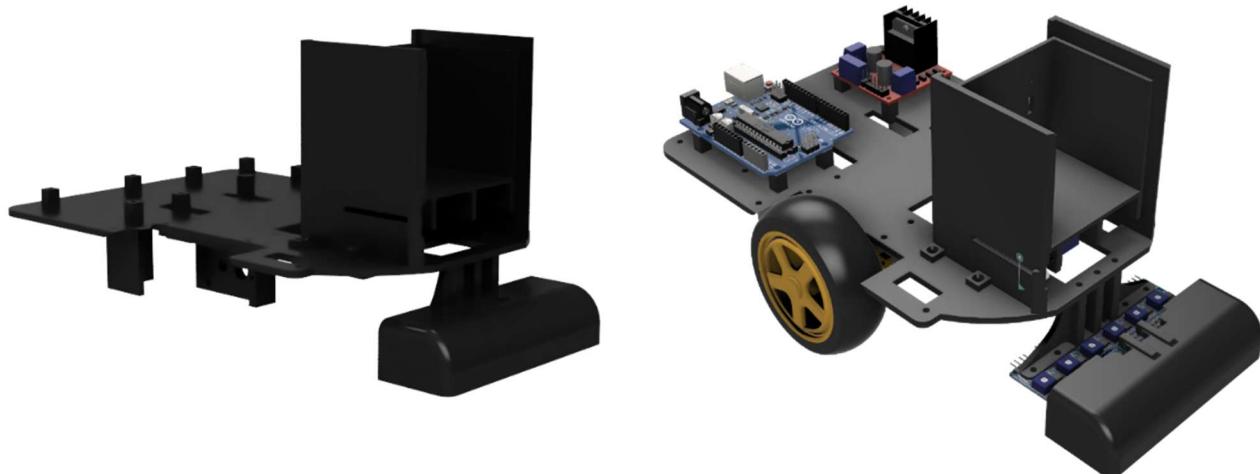


Figura 13 - Projeto com suportes e componentes modelos

#### 2.2.1.2 MATERIAL USADO NA ESTRUTURA

A estrutura do robô foi toda desenhada e projetada para ser impressa em 3D utilizando filamento preto (PLA). Quanto ao material PLA, foi selecionado devido às suas propriedades de resistência, durabilidade e facilidade de impressão, sendo capaz de enfrentar as exigências físicas da operação do robô.

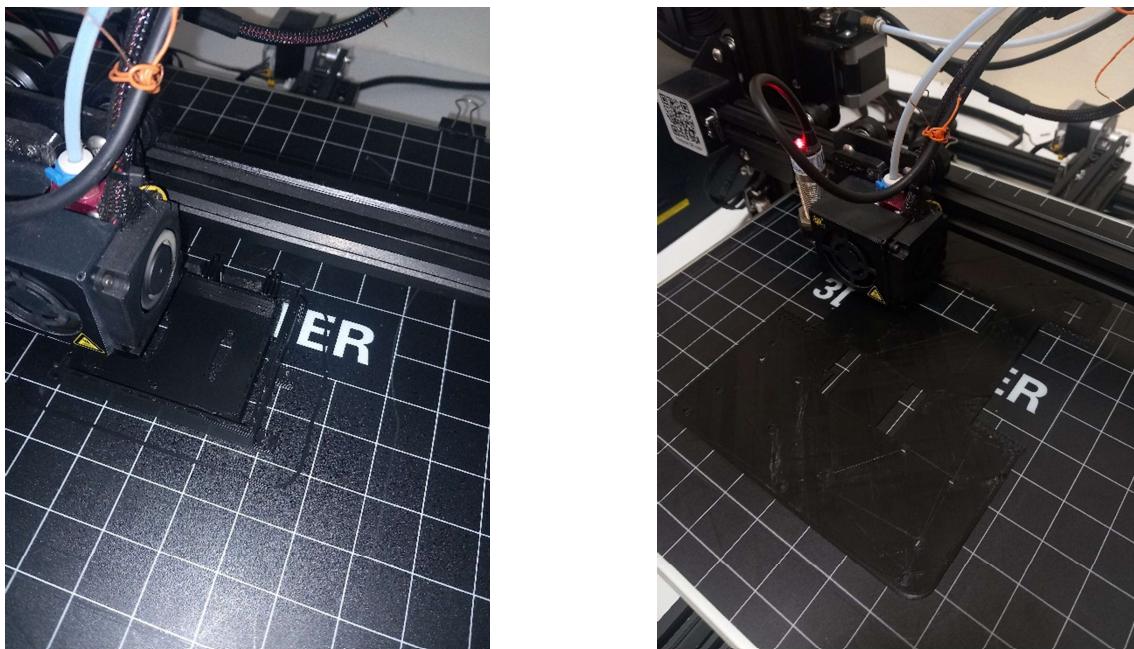


Figura 14 - Impressões 3D

### 2.2.1.2.1 SOFTWARE USADO NA IMPRESSÃO 3D

O software usado na impressão 3D foi recomendada pelo professor orientador da PAP. O software chama-se UltiMaker Cura.

O Cura é um software de código aberto amplamente utilizado para preparar modelos 3D para impressão em impressoras 3D. Desenvolvido pela Ultimaker, é conhecido como um "fatiador", que é essencialmente um programa que divide um modelo 3D em camadas finas (fatias) e gera instruções para a impressora 3D seguir durante o processo de impressão.



Figura 15 - Logótipo UltiMaker



Figura 16 - UltiMaker Cura software

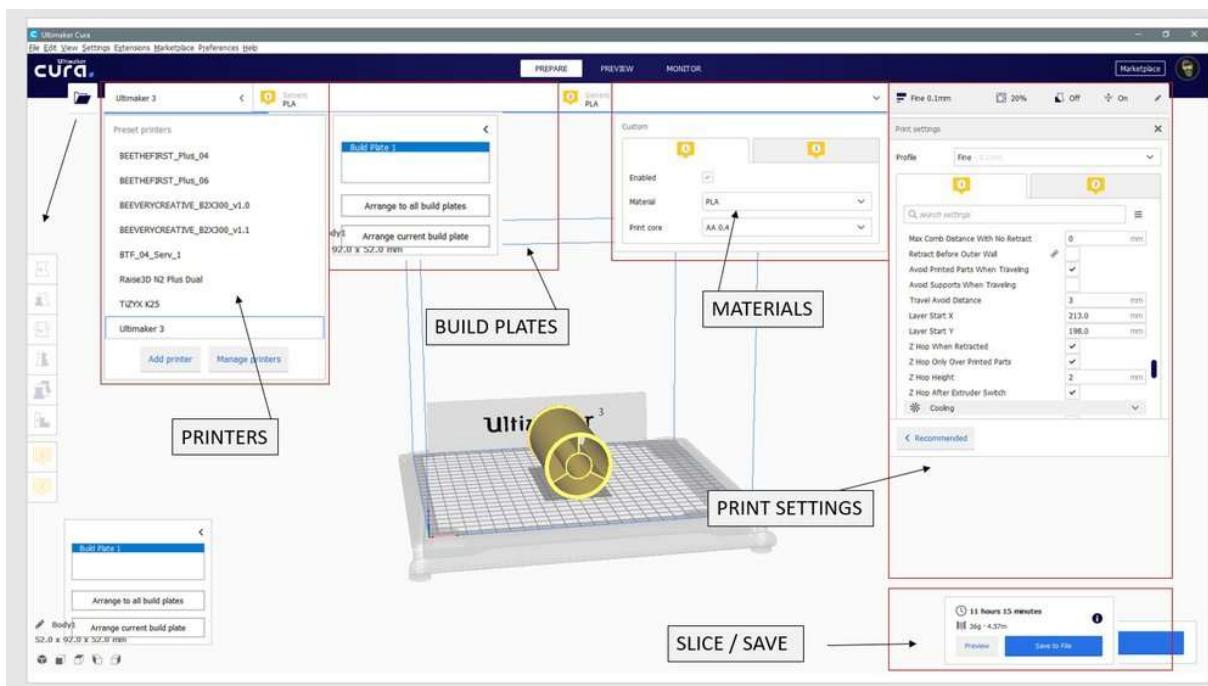


Figura 17 - Software UltiMaker Cura

## 2.3 SELEÇÃO DA PLATAFORMA CONTROLADORA

Neste ponto será exposto a escolha da plataforma controladora de entre as possíveis e a escolha final da solução mais adequada para o projeto.

### 2.3.1.1 ARDUÍNO UNO

Arduíno é uma plataforma open-source de prototipagem eletrónica com hardware e software flexíveis e fáceis de usar, destinado a qualquer pessoa interessada em criar objetos ou ambientes interativos. O Arduíno contém um microcontrolador - o ATmega328 de 32kb, um conector USB, pinos de entrada e saída, pinos de alimentação, botão de reset, Conversor Serial USB e LEDs TX/RX, Conector de alimentação, LED de alimentação e LED interno.

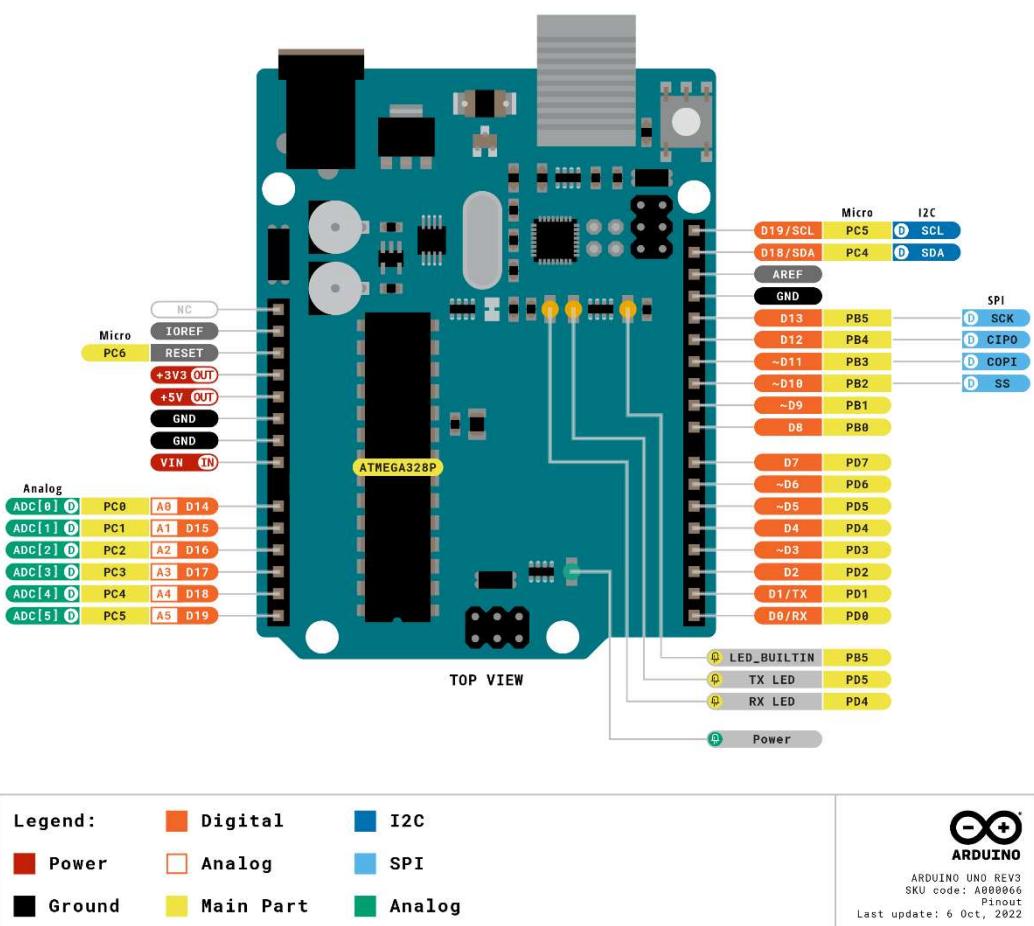


Figura 18 - Datasheet de Arduíno Uno

### 2.3.2 APLICAÇÃO DE DESENVOLVIMENTO ESCOLHIDA

A aplicação de desenvolvimento escolhida para este projeto foi o Arduíno IDE. Elaborei várias pesquisas e comparações para ver qual seria a mais adequada ao meu projeto.

Escolhi o Arduíno IDE pois é uma plataforma com a qual eu já estou habituado a trabalhar e à sua utilização, e adequa-se bastante às minhas necessidades.

É uma aplicação open-source bastante simples de utilizar, de fácil interface e com a vantagem de haver inúmeras dicas e tutoriais na internet.

#### 2.3.2.1 ARDUÍNO IDE

O Arduíno IDE é uma aplicação multiplataforma escrita em Java. Foi concebido para simplificar o desenvolvimento tanto a programadores como a pessoas não familiarizadas com o desenvolvimento de software.

O software inclui um editor de código com vários recursos, sendo capaz de compilar e carregar programas para a placa com um clique. Tem a capacidade de programar em C/C++, o que permite criar com facilidade muitas operações de entrada e saída, tendo de definir apenas duas funções no pedido para fazer um programa funcional.



Figura 19 - Arduíno IDE



### 2.3.2.2 CÓDIGO NO ARDUÍNO

Neste capítulo, apresentamos o código utilizado no projeto, desenvolvido em C++. Devido à extensão do código fonte, optei por disponibilizar apenas uma página com um código QR. Ao escanear o código QR com um dispositivo, é direcionado para o documento digital contendo a versão completa do código do projeto.



Figura 20 - Código Qr, Código no Arduino ([Https://bit.ly/493ZbwE](https://bit.ly/493ZbwE))



## 2.4 EXPLICAÇÃO DO CÓDIGO

### 2.4.1 CÓDIGO SEGUIDOR DE LINHA USANDO PID

Neste ponto irei explicar o código que possibilita o robô seguir uma linha e também como funciona a tecnologia PID utilizada nesse código.

#### 2.4.1.1 O QUE É O PID E O SEU FUNCIONAMENTO

O controlo Proporcional-Integral-Derivativo (PID) é o algoritmo de controlo mais comum usado em indústria e tem sido universalmente aceite. O desempenho robusto numa ampla gama de condições operacionais e, em parte, à sua funcionalidade simples, permite que os engenheiros os operem de maneira simples. Como o nome sugere, um PID.

O algoritmo consiste em três coeficientes básicos: proporcional, integral e derivativo. Estes ganhos são variados para obter uma resposta ideal do sistema.

**P** - O termo P, na tecnologia de controlo PID, representa a parte proporcional do controlo, é um coeficiente que determina a magnitude do ajuste que será aplicado com base no erro atual do sistema. Em termos simples, o P é responsável por corrigir o erro atual de forma proporcional, ou seja, quanto maior o erro, maior será o ajuste realizado. Por exemplo, se o erro entre a posição desejada e a posição atual do sistema é grande, o ajuste proporcional (P) será maior, resultando numa correção mais significativa no sistema. Para ajustar o P usei:

$$\text{Ajuste} = P * \text{erro}$$

**I** - O termo I, na tecnologia de controlo PID, é a parte Integral do controlo. Ele acumula os erros ao longo do tempo e compensa os fatores externos que afetam o sistema. Em suma, o termo I ajuda a corrigir erros persistentes e a manter o sistema próximo do ponto de referência desejado, mesmo em condições variáveis. Por exemplo, se um robô seguidor de linha enfrenta variações na velocidade devido a mudanças no terreno, como subidas ou descidas, o termo I ajusta gradualmente a correção para compensar essas variações e manter o robô na trajetória desejada. Neste projeto não foi usado o "I" porque o robô vai funcionar numa superfície plana e nivelada.

**D** – O termo D serve para melhorar a sua capacidade de convergir suavemente para o valor desejado, reduzindo oscilações e minimizando o tempo de resposta. Ele calcula a diferença entre o erro atual e o erro

anterior. Quando o erro está a diminuir, o termo D reduzirá a correção aplicada pelo termo P, evitando assim um possível excesso além do ponto desejado. Por outro lado, quando o erro está a aumentar, o termo D aumentará a correção do termo P para ajudar a corrigir a trajetória do sistema. Em resumo, o termo D proporciona uma correção proporcional à taxa de mudança do erro, contribuindo para um controlo mais suave e preciso. Para determinar o ajuste do D usei:

$$\text{Ajuste} = P * \text{error} + D * (\text{error} - \text{UltimoErro})$$

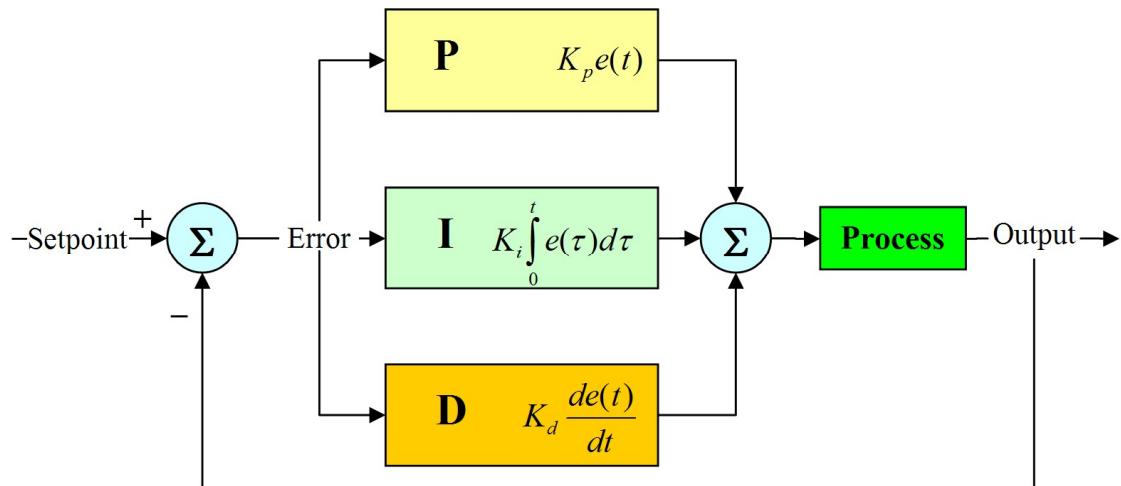


Figura 21 - Diagrama de cálculos do PID



```
21 const int numPins = 6;      // Número de pinos analógicos (A0 a A5)
22 int sensorValues[numPins]; // Array para armazenar leituras analógicas
23
24 int sensorVal[6] = { 0, 0, 0, 0, 0, 0 }; // Array para armazenar valores binários dos sensores de
25 int error, last_error, MV, pid_l, pid_r, D, D1, D2, D3, I, I1, I2, I3, P, Pd, bitSensor;
26 int Max_MV;
27 unsigned char Kp = 12.5; // Kp é um coeficiente que determina a magnitude da correção proporcional
28 ||| ||| ||| ||| //com base no erro entre a posição desejada e a posição real.
29 unsigned char Kd = 0; // Kd é um coeficiente que determina como a correção do controlador PID é ir
30 unsigned char Ts = 1;
31 unsigned char maxPwm = 200; // maxPwm define a velocidade máxima dos motores
32 unsigned char intersection = 0; // Variável para armazenar o número de interseções
33
```

Figura 22 - Excerto de código, variáveis do PID

Como representado no código acima, temos as variáveis chamadas de “Kp”, “Kd”, “Ts” e “maxPwm”. A variável “Kp” é o coeficiente proporcional no controlador PID, determina a influência da correção proporcional (P) na saída do controlador. Ajustar o “KP” afeta a rapidez com que o controlador responde ao erro. Um valor maior de “KP” resulta numa resposta mais rápida, enquanto um valor menor resulta numa resposta mais suave. Ajustar bem o “KP” é essencial para um controlo preciso e estável.

A variável “Kd” no controlador PID serve para ajustar a influência da taxa de mudança do erro na correção aplicada pelo controlador. Em outras palavras, ele determina o quanto o controlador reage à velocidade com que o erro está mudando.

Em resumo, o “Kp” ajusta a magnitude da correção de acordo com o tamanho do erro atual, enquanto o “Kd” ajusta a rapidez com que essa correção é aplicada, considerando a taxa de mudança do erro ao longo do tempo. Ambos os coeficientes trabalham em conjunto para garantir um controlo preciso e estável do sistema.

A variável "Ts" representa o coeficiente que define o intervalo de amostragem em sistemas de controlo em tempo discreto. Este intervalo determina com que frequência o controlo PID é atualizado. Um período de amostragem mais curto permite atualizações mais frequentes, enquanto um período mais longo resulta em atualizações menos frequentes.

A variável "maxPWM" define a velocidade máxima dos motores do robô, controlando o quanto o sinal PWM pode aumentar a potência entregue aos motores.

#### 2.4.1.1.1 CÁLCULO DE ERRO USANDO PID

```

85 void robotRun() {
86     // Função para controlar o movimento do robô com base nos sensores de linha
87     readSens(); // Ler os sensores de linha

```

Figura 23 - Excerto de código, funções PID

Para calcular o erro de leitura, é chamada a função “readSens()” para ler os valores dos sensores de linha e armazená-los no array “sensorVal[]”. Podemos ver a chamada da função no código acima.

```

88     bitSensor = ((sensorVal[0] * 1) + (sensorVal[1] * 2) + (sensorVal[2] * 4) +
89     (sensorVal[3] * 8) + (sensorVal[4] * 16) + (sensorVal[5] * 32)); // Converter os valores dos sensores para um número binário

```

Figura 24 - Excerto de código, convertor binário PID

Em seguida os valores dos sensores são convertidos num número binário representado pela variável “bitSensor”. Cada sensor contribui com um bit ao número binário, onde um valor alto (HIGH) corresponde a um bit 1 e um valor baixo (LOW) corresponde a um bit 0. A fórmula  $((sensorVal[0] * 1) + (sensorVal[1] * 2) + \dots)$  realiza essa conversão.

```

89 switch (bitSensor) {
90     // Determinar o erro com base na posição dos sensores
91     case 0b110000: error = -4; break;
92     case 0b010000: error = -3; break;
93     case 0b011000: error = -2; break;
94     case 0b001000: error = -1; break;
95     case 0b001100: error = 0; break;
96     case 0b000100: error = 1; break;
97     case 0b000110: error = 2; break;
98     case 0b000010: error = 3; break;
99     case 0b000011: error = 4; break;
100    case 0b000001: error = 5; break;
101 }

```

Figura 25 - Excerto de código, determinação de erro de leituras PID

No código descrito acima, o bloco de switch mapeia o valor binário bitSensor para um valor de erro correspondente. Cada caso no switch representa um padrão de saída específico dos sensores e o seu valor de erro associado.

Por exemplo, se bitSensor for igual a 0b110000, o robô é inclinado para a esquerda e mais sensores no lado direito são ativados, de modo que o erro é definido como -4.



Portanto, o código calcula o erro com base na posição dos sensores de linha e usa esse valor para ajustar o movimento do robô, garantindo que ele permaneça centrado na linha.

```
Max_MV = Kp * 5; // Calcular o máximo de MV
P = Kp * error; // Calcular P (proporcional)
D1 = Kd * 8;
D2 = D1 / Ts;
D3 = error - last_error;
D = D2 * D3;

last_error = error;
MV = P + D;

// Controlar a velocidade dos motores com base no erro calculado
if (MV >= -Max_MV && MV <= Max_MV) {
    pid_l = maxPwm + MV;
    pid_r = maxPwm - MV;
    if (pid_l < 0) pid_l = 0; // Garantir que o valor não ultrapasse os limites
    if (pid_l > 200 ) pid_l = 200 ;
    if (pid_r < 0) pid_r = 0;
    if (pid_r > 200 ) pid_r = 200 ;
    forward(pid_r, pid_l); // Mover para frente
} else if (MV < -Max_MV) {
    turnLeft(190 , 190); // Virar à esquerda
} else if (MV > Max_MV) {
    turnRight(190, 190); // Virar à direita
} else {
    forward(pid_r, pid_l); // Continuar em linha reta
}
```

Figura 26 - Excerto de código, velocidade e erro PID

Este pedaço do código controla o movimento do robô com base no valor calculado do controlo PID. Se o valor de controlo (MV) estiver dentro dos limites permitidos (-Max\_MV a Max\_MV), os valores de velocidade dos motores esquerdo (pid\_l) e direito (pid\_r) são ajustados para a frente, garantindo que ficam dentro dos limites permitidos (0 a 200).



## 2.4.2 LEITURA E CLASSIFICAÇÃO DE CORES

```
131 int readColor() {  
132     // Função para ler a cor detectada pelo sensor de c  
133     // Leitura da cor vermelha  
134     digitalWrite(S2, LOW);  
135     digitalWrite(S3, LOW);  
136     frequency = pulseIn(sensorOut, LOW);  
137     int R = frequency;  
138     Serial.print("R= ");  
139     Serial.print(frequency);  
140     Serial.print(" ");  
141     delay(50);  
142     // Leitura da cor verde  
143     digitalWrite(S2, HIGH);  
144     digitalWrite(S3, HIGH);  
145     frequency = pulseIn(sensorOut, LOW);  
146     int G = frequency;  
147     Serial.print("G= ");  
148     Serial.print(frequency);  
149     Serial.print(" ");  
150     delay(50);  
151     // Leitura da cor azul  
152     digitalWrite(S2, LOW);  
153     digitalWrite(S3, HIGH);  
154     frequency = pulseIn(sensorOut, LOW);  
155     int B = frequency;  
156     Serial.print("B= ");  
157     Serial.print(frequency);  
158     Serial.println(" ");  
159     delay(50);  
160     // Determinar a cor predominante com base nas leitu  
161     if (R < 45 && R > 25 && G < 125 && G > 100) {  
162         Serial.println(" Red !\n");  
163         color = 1; // Vermelho  
164     }  
165     if (R < 145 && R > 125 && G < 115 && G > 95) {  
166         Serial.println(" Green !\n");  
167         color = 2; // Verde  
168     }  
169     if (G < 95 && G > 70 && B < 45 && B > 25) {  
170         Serial.println(" Blue !\n");  
171         color = 3; // Azul  
172     }  
173     return color; // Retornar a cor detectada  
174 }  
175 }
```

Figura 27 - Excerto de código, leitura de cor

Para a leitura e classificação das cores usando o Sensor de Cor RGB TCS32000, o código é uma função que utiliza um sensor de cor para ler as intensidades das cores vermelha, verde e azul. Em seguida, com base nessas leituras, determina a cor predominante e retorna-a como um valor numérico.

A função começa por configurar o sensor para ler o vermelho, faz a leitura da frequência correspondente usando `pulseIn()` e armazena o valor na variável `R`. O mesmo procedimento é repetido para as cores verde e azul, armazenando os valores em `G` e `B`, respetivamente.

Após as leituras, o código analisa os valores de `R`, `G` e `B` para determinar a cor predominante. Se esses valores estiverem dentro de intervalos específicos para vermelho, verde ou azul, a função imprime o nome da cor predominante na porta serial e atribui um valor numérico correspondente à variável `color`. Essa variável é então retornada como o resultado da função. O robô realiza a leitura da cor da carga somente quando os módulos sensores infravermelhos detetam a linha de partida na pista.



## **2.4.3 ESCOLHA DOS TRAJETOS**

A determinação dos trajetos acontece após o robô detetar a linha de partida na pista. Nesse momento, é feita a leitura da cor correspondente à carga posta no suporte de descarga, sendo essa cor que irá definir o trajeto para o destino adequado.

```
177 void mission() {
178     // Função para coordenar as operações do robô
179     robotRun(); // Executar o controle do robô
180     boxServo.write(35); // Mover o servo motor
181     // Verificar se o robô está em uma interseção
182     if (sensorVal[5] == 1 && sensorVal[0] == 1) {
183         stopRun(); // Parar o movimento do robô
184         color = readColor(); // Ler a cor detectada
185
186         // Executar ações com base na cor detectada
187         switch (color) {
188             case 1: // Vermelho: Virar à esquerda
189                 red();
190                 break;
191             case 2: // Verde: Seguir em frente
192                 green();
193                 break;
194             case 3: // Azul: Virar à direita
195                 blue();
196                 break;
197             case 0: // Nenhuma cor detectada: Parar
198                 stopRun();
199                 break;
200         }
201     }
202 }
```

*Figura 28 - Excerto de código, escolha de trajetos*

Após detetar a linha de partida na pista usando a seguinte if (`SensorVal[5] == 1 && SensorVal[0] == 1`), o código chama a função `readColor()` para obter a cor da carga que foi explicada no tópico anterior. Depois, com base na cor detetada, o robô executa ações específicas, chamando as funções correspondentes para cada cor.

```
206 void red() {
207     intersection++;
208     if (intersection == 1) {
209         turnBack(100, 100); // Virar de volta
210     } else if (intersection == 2) {
211         turnLeft(100, 100); // Virar à esquerda
212     } else if (intersection == 3) {
213         boxServo.write(90); // Mover o servo motor
214         stopRun(); // Parar
215         delay(500); // Aguardar
216         turnBack(100, 100); // Virar de volta
217     } else if (intersection == 4) {
218         turnRight(100, 100); // Virar à direita
219         delay(100); // Aguardar
220     } else if (intersection == 5) {
221         resetFunc(); // Reiniciar o Arduino
222     }
223 }
```

*Figura 29 - Excerto de código, função da cor vermelha*



Vamos imaginar que a função `readColor()` detetou o vermelho, então vai ser chamada a função `red()`. Dentro desta função, sempre que todos os módulos de sensores infravermelhos detetam uma intercetação, ou seja, quando estão numa superfície preta, o robô conta as interseções que já passou. Cada intercetação dispara um movimento diferente para seguir o caminho definido pela cor detetada.

### 1º Interseção

Na primeira interseção, que é a linha de partida da pista, o robô vira-se de volta para a pista para se deslocar até à próxima interseção.

### 2º Interseção

Na segunda interseção, o robô vira na direção determinada pela cor detetada, que neste caso é a vermelha. Assim, o robô vira para a esquerda e segue o percurso nessa direção.

### 3º Interseção

Na terceira interseção, o robô alcança o seu destino. Ele larga a sua carga nessa interseção e depois vira-se de volta para a pista, preparando-se para se deslocar até à próxima interseção.

### 4º Interseção

Na quarta interseção, o robô vira na direção da linha de partida. Para o vermelho, o robô vira para a direita, seguindo o caminho de volta em direção à linha de partida.

### 5º Interseção

No quinto cruzamento, o robô atingiu a linha de partida, reiniciando assim o código para se preparar para receber outra carga.

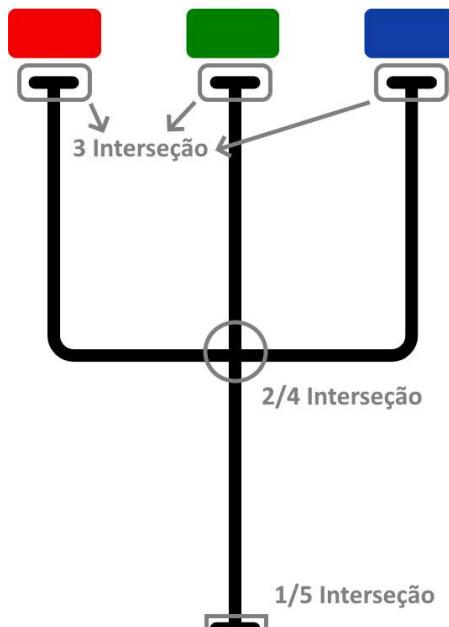


Figura 30 - Ilustração da numeração de interseções



## 2.5 COMPONENTES UTILIZADOS NO PROJETO

Neste ponto falarei sobre os componentes utilizados no projeto. Mais detalhes no anexo encontrado no ponto *5.4 Datasheet dos componentes utilizados*.

### 2.5.1 ARDUÍNO UNO

Foi utilizado o Arduíno Uno como já foi mencionado no ponto *2.3.1.1 Arduíno IDE*.

## 2.5.2 PONTE H L298N

O módulo ponte H L298N é um componente eletrónico usado para controlar motores DC (corrente contínua). Atua como uma ponte H, um circuito eletrónico que permite controlar a direção e a velocidade de um motor. Este módulo é comum em projetos de robótica e automação, pois oferece uma forma eficaz de controlar motores de diferentes tipos e tamanhos. Pode ser controlado por sinais de entrada, como sinais PWM (Pulse Width Modulation), e é capaz de fornecer correntes relativamente altas para os motores.

### Especificações:

- Tensão da Operação: 4~35V;
- Chip: ST L298N;
- Controlo de 2 motores DC ou 1 motor de passo;
- Corrente da Operação máxima: 2A por canal ou 4A max;
- Tensão lógica: 5V;
- Corrente lógica: 0~36mA;
- Limites de Temperatura: -20 a +135°C;
- Potência Máxima: 25W;
- Dimensões: 43 x 43 x 27mm;
- Peso: 30g.

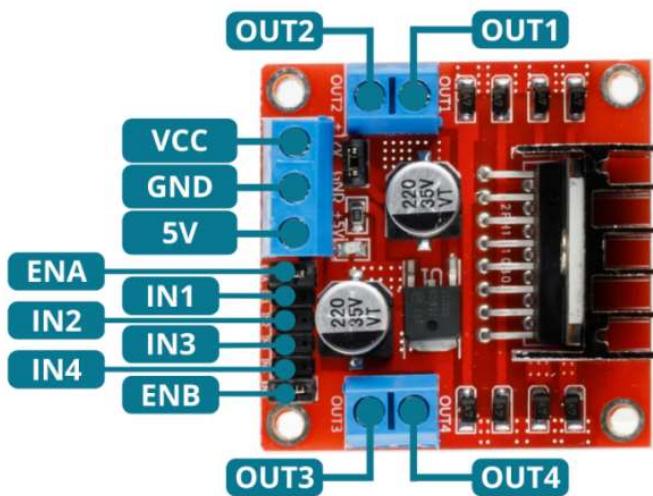


Figura 31 - Controlador Ponte H L298N

### 2.5.3 SENSOR DE COR RGB TCS3200

O sensor de cores utiliza o chip TCS3200 para detetar os níveis de luzes RGB dos objetos através do envio de dados para um microcontrolador, como o Arduino, Raspberry, Pic, entre outros. Esse chip tem 64 foto dióodos, sendo com 16 filtros para o vermelho, 16 para a verde, 16 para a azul e 16 sem filtro, permitindo a criação de sistemas de deteção de cores eficientes. Também possui quatro LEDs brancos para iluminação e oito pinos de conexão.

#### Características

- Tensão de Alimentação: 2.7V a 5.5V;
- Interface: Digital TTL;
- Conversão de alta-resolução da intensidade de luz para frequência;
- Cor programável e gama alargada de frequência de saída (full-scale);
- Comunica diretamente ao microcontrolador.



Figura 32 - Sensor de Cor RGB TCS3200



#### 2.5.4 MÓDULO SENSOR INFRAVERMELHO TCRT5000

O Módulo Sensor Infravermelhos TCRT5000 possui integrado um sensor infravermelho (emissor) e uma foto transístor (recetor). Foi especialmente desenvolvido para bloquear outras faixas de luz que não seja a do próprio emissor, evitando que as iluminações do ambiente venham causar alguma interferência.

É muito utilizado em projetos com Arduíno como robôs seguidores de linha, identificação de objetos reflexivos, etc.

Este Módulo Sensor conta ainda com um potenciômetro para ajuste da sensibilidade do sensor, o que facilita bastante a construção do protótipo, ganhando-se tempo, já que não há a necessidade de implementar este ajuste no software do microcontrolador.

#### Especificações:

- Modelo: TCRT5000;
- Distância de deteção: 1mm ~ 8 mm;
- Tensão de trabalho: 3.3V-5V;
- Saída Digital: TTL;
- Dimensões: 32mm x 14 mm;
- Utiliza comparador de tensão LM393.

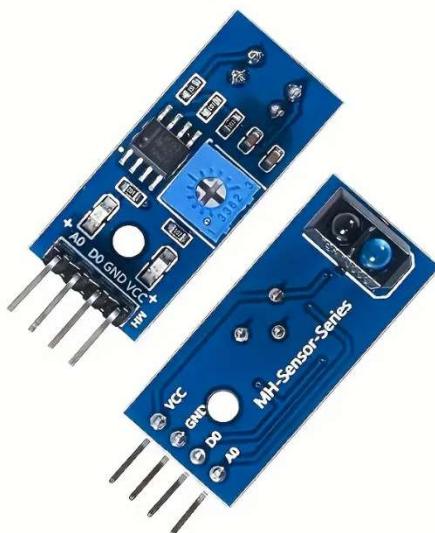


Figura 33 - Módulo Sensor Infravermelhos TCRT5000



### 2.5.5 SERVO MOTOR SG90 9G TOWER PRO

O Servo Motor SG90 é um módulo que apresenta movimentos proporcionais aos comandos indicados, controla o girar e a posição, diferente da maioria dos motores. Possui um ângulo de rotação de 180 graus e acompanha um cabo de 3 pinos referente à alimentação/controlo e diversos acessórios (3 tipos de braços + parafusos).

#### Especificações:

- Servo motor SG90;
- Tensão da Operação: 3,0 - 7,2V;
- Ângulo de rotação: 180 graus;
- Velocidade: 0,12 seg/60Graus (4,8V) sem carga;
- Torque: 1,2 kg.cm (4,8V) e 1,6 kg.cm (6,0V);
- Temperatura da Operação: -30C ~ +60C;
- Tamanho do cabo: 245mm;
- Dimensões: 32 x 30 x 12mm;
- Peso: 9g.



Figura 34 - Servo Motor SG90 9g Tower Pro



## 2.5.6 MOTORES TT 3V ~ 12V DC

Um motor TT é um tipo de motor DC que tem uma caixa de engrenagens acoplada. A caixa de engrenagens reduz a velocidade do motor e aumenta a sua força. Um motor TT é normalmente usado em aplicações como acionamento de rodas, hélices, ventiladores, entre outros.

### Especificações:

Tensão nominal: 3~12V

Corrente contínua sem carga: 150mA +/- 10%

Velocidade mínima de operação (3V): 90+/- 10% RPM

Velocidade mínima de operação (6V): 200+/- 10% RPM

Torque: 0.15Nm ~0.60Nm

Torque de bloqueio (6V): 0.8kg.cm

Relação de transmissão: 1:48

Dimensões do corpo: 70 x 22 x 18mm

Peso: 30.6g



Figura 35 - Motor TT 3V~12V

## 2.5.7 PLACA DE EXPANSÃO PARA ARDUÍNO UNO R3

A placa de expansão de sensores "Arduino Sensor Shield V5.0" é um dispositivo que amplia os pinos de terra (ground) e VCC, permitindo mais ligações de componentes eletrônicos. Além disso, a placa disponibiliza espaços designados especificamente para acomodar alguns sensores ou placas adicionais.

### Especificações:

- 14 pinos digitais, incluindo 6 pinos de função PWM
- 8 pinos analógicos (A6 e A7 são projetados para a placa de controlo BlueBird)
- LED integrado, botão de reset e porta externa de alimentação
- LED vermelho -- Indicador de energia
- LED amarelo -- Conectado ao pino D13
- Botão RESET
- Porta de alimentação externa de 5V

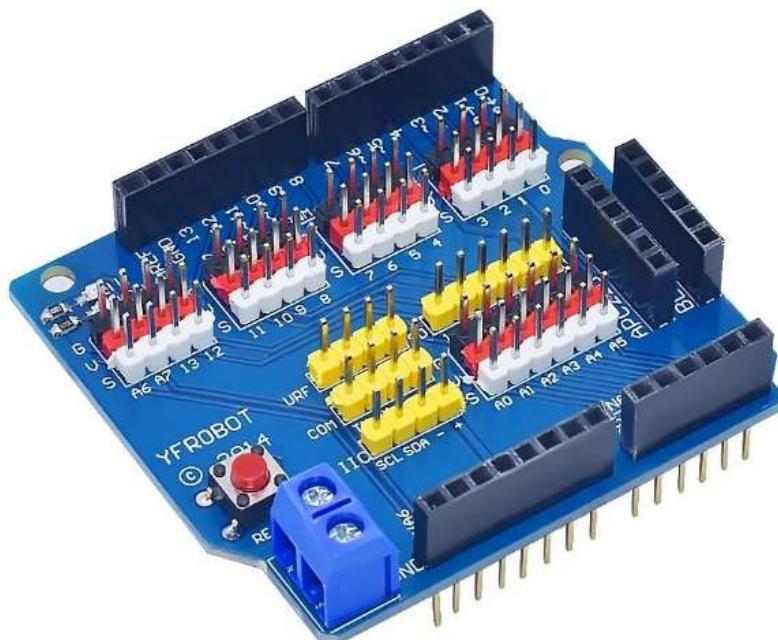


Figura 36 - Placa de Expansão para Arduíno Uno R3



### 2.5.8 BATERIA DE LÍTIO RECARREGÁVEL

Bateria recarregável, carregada via USB

#### Especificações:

Tipo: Bateria do lítio

Capacidade: 12800mAh

Tensão nominal: 9V

Dimensão: 48x26x16mm

Peso: ≈ 20g



Figura 37 - Bateria de Lítio Recarregável

### 2.5.9 Fios DuPONT

Os cabos de ligação DuPont utilizados no projeto.

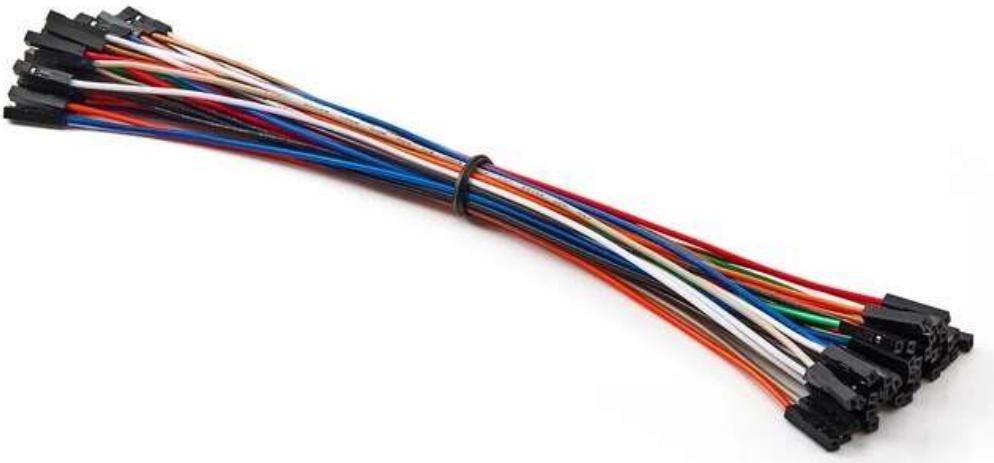


Figura 38 - Fios DuPont



## 2.6 DESENVOLVIMENTO DO PROJETO

Para realizar este projeto realizaram-se vários testes para saber se os códigos submetidos estavam a realizar a tarefa solicitada. Para a conclusão deste projeto foi necessário tomar várias atitudes sobre o que fazer e a ordem de execução.

### 2.6.1 ETAPAS DO DESENVOLVIMENTO DO PROJETO

Decidi dividir o projeto em três fases: Fase Inicial, Fase Intermedia e Fase Final. Na Fase Inicial, desenhei o projeto em 3D, o que consiste em fazer com que o equipamento (Sensores + Motores + Arduíno) fizesse o mínimo dos requisitos solicitados. Na Fase Intermedia, desenvolvi a programação e a construção do projeto. Finalmente, na Fase Final, garanti o correto funcionamento e realizei a instalação do projeto.

#### 2.6.1.1 EVOLUÇÃO DO PROJETO

Neste ponto são referidas as fases em que o projeto foi desenvolvido, os testes dos vários componentes que participam no funcionamento do projeto, tanto a nível de programação como a nível do hardware usado, os ensaios dos conjuntos, a correção de eventuais irregularidades identificadas no protótipo final.

##### 2.6.1.1.1 FASE INICIAL

No processo de desenho do projeto do robô, enfrentei um desafio devido à minha falta de experiência e conhecimento em desenho 3D no Fusion 360. Isso levou a atrasos no desenho, com várias tentativas e muitos erros. No entanto, cada erro foi uma oportunidade de aprendizagem, o que me permitiu adquirir conhecimentos e melhorar as minhas habilidades ao longo do processo. Alguns problemas tais como:

- Durante o desenho do projeto do robô, percebi que tinha dificuldade em perceber as medidas virtuais e as medidas reais. Isso resultou na criação de uma primeira base com falta de espaço e organização, além de medidas assimétricas. Depois de muitos dias de tentativas e de erros finalmente foi feita a base perfeita para o robô.
- Ao desenhar os suportes para os componentes, deparei-me com problemas de resistência insuficiente. Os suportes tinham pouca espessura e um aspeto visual pouco atrativo. Durante a instalação, as bases não conseguiram suportar a fixação ao chassi e a carga e acabaram por partir devido à falta de espessura e à falta pontos de reforço adequados.

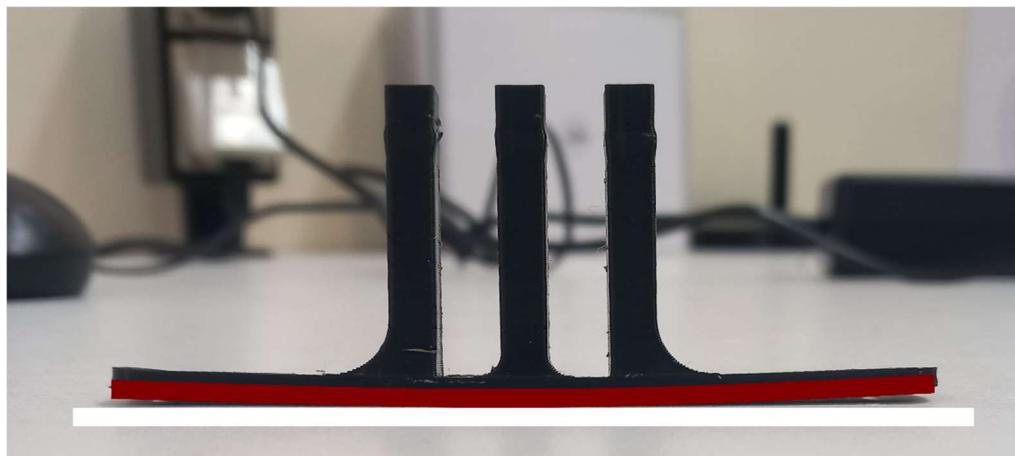


Figura 39 - Suporte de módulos sensores infravermelhos defeituoso

Aqui podemos ver um dos primeiros suportes impressos e desenhados para módulos de sensores infravermelhos. Este suporte demonstra pouca resistência, podemos ver na figura a cima , em que a base empenou-se , devido a fixação dos sensores , resultando em uma posição desigual dos módulos sensores.



Figura 40 - Suporte de módulos sensores infravermelhos corrigido

Após o erro de design inicial, foi desenvolvido um novo suporte para este módulo. Este suporte apresenta curvas cônicas na base, proporcionando estabilidade e garantindo que os sensores permaneçam nivelados entre si. A eficácia do novo suporte pode ser observada na figura acima.



No sistema basculante, também enfrentamos problemas. No círculo vermelho destacado na imagem abaixo, pode-se observar a falta de resistência do suporte basculante, que se partiu devido à fixação do servo motor com parafusos. Para resolver este problema, uma coluna com espessura aumentada foi adicionada para reforçar a resistência na área problemática.

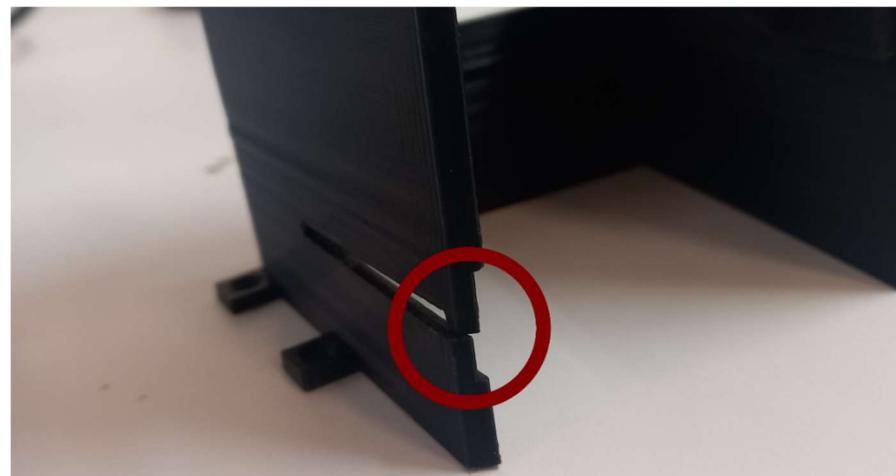


Figura 41 - Suporte de basculante danificado

Na figura abaixo, podemos observar um desnível significativo entre o basculante e o suporte, resultante da fraca força de fixação do servo motor ao suporte basculante. Para solucionar esse problema, foi desenhada uma fenda tanto no suporte basculante quanto no basculante para a colocar um veio, com o propósito de proporcionar maior estabilidade e firmeza no basculante.

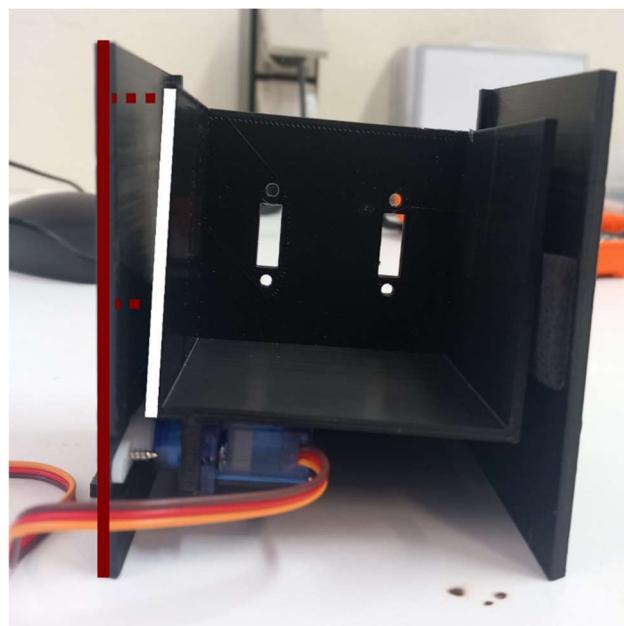


Figura 42 - Sistema basculante defeituoso



Figura 43 - Sistema basculante melhorado

Na fase inicial do projeto também encontrei algumas dificuldades na montagem do projeto, tais como:

- Primeiramente, tive alguns problemas na programação do Arduíno com o Controlador de Ponte H, pois os motores ligados ao controlador estavam a funcionar de forma inesperada, tanto na sua velocidade como no seu sentido e direção. Depois de algumas horas, verificou-se que o código estava correto e que era apenas uma má conexão de cabos, além de uma potência excessiva entregue ao controlador de Ponte H.
- Em seguida tive muitos problemas com os Módulos Sensores Infravermelhos. Neste projeto já foram usados três Módulos Sensores Infravermelhos diferentes, dois Módulos Sensores encontravam-se com muitas falhas de leitura e a distâncias de leitura muito reduzidas, devolvendo assim valores incorretos o que ia condicionar o bom funcionamento do Robô.

Vou falar dos problemas identificados de cada Modulo Sensor Infravermelhos usados:

- **Módulo Sensor Infravermelhos, 8 Sensores:**

Neste módulo identificou-se que os valores lidos pelos sensores extremos direitos e extremos esquerdos eram muito instáveis e apresentavam uma grande margem de erro. Após várias tentativas de correção, não foi encontrada nenhuma solução para o problema. Também se verificou que a distância entre os sensores era demasiado curta, resultando que os quatro sensores detetassem a linha quando o ideal seriam apenas dois sensores. Os restantes sensores destinam-se a detetar interseções e linhas de paragem.

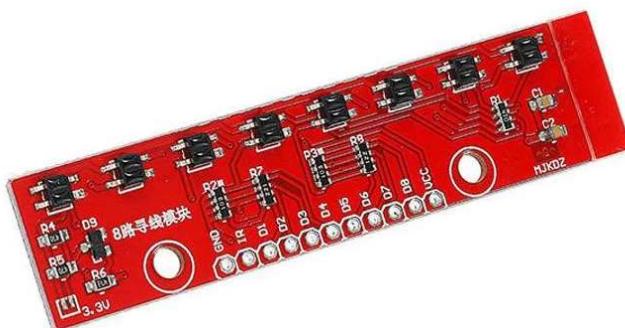


Figura 45 - Módulo Sensor Infravermelhos, 8 sensores, teste

- **Módulo Sensor Infravermelho Tcrt 5000**

Neste módulo sensor percebeu-se que os valores obtidos pelo sensor Tcrt 5000 eram apenas em sinal digital. Devido à falta de portas digitais disponíveis, este módulo sensor não foi utilizado, uma vez que só estavam disponíveis portas analógicas na placa Arduino Uno.



Figura 46 - Módulo Sensor Infravermelho Tcrt 5000, teste

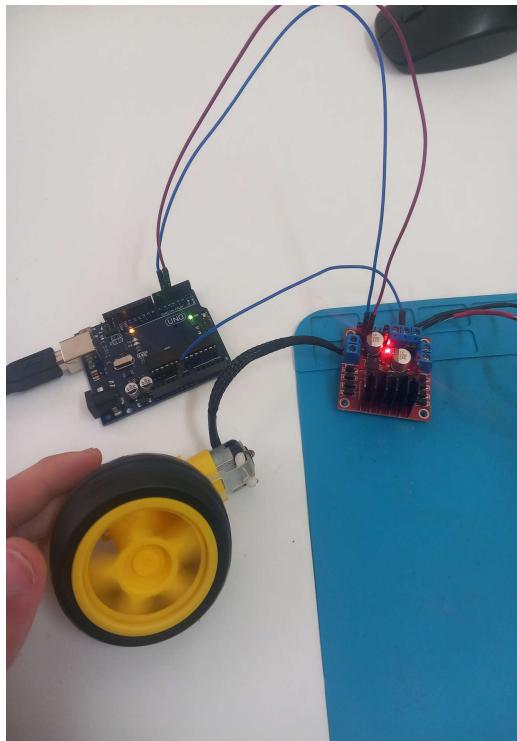


Quando recebi o Arduíno Uno R3 pela primeira vez, conectei-o ao Controlador Ponte H e fiz todas as conexões necessárias sem problemas. Depois conectei os Motores TT ao Controlador Ponte H e realizei alguns testes durante o processo de montagem.

The screenshot shows the Arduino IDE interface with the title bar "sketch\_mar5a | Arduino IDE 2.2.1". The toolbar includes icons for save, upload, and refresh. The board selector dropdown shows "Arduino Uno". The left sidebar has icons for file, folder, book, and search. The main code editor window displays the following C++ code:

```
sketch_mar5a.ino
3 //motor A
4 int IN1 = 2 ;
5 int IN2 = 3 ;
6
7 //Inicializa Pinos
8 void setup(){
9     pinMode(IN1,OUTPUT);
10    pinMode(IN2,OUTPUT);
11    pinMode(IN3,OUTPUT);
12    pinMode(IN4,OUTPUT);
13 }
14
15 void loop(){
16     /*Inicio dos Estados do motor A*/
17
18     //Sentido Horario
19     digitalWrite(IN1,HIGH);
20     digitalWrite(IN2,LOW);
21     delay(5000);
22
23     //trava Motor
24     digitalWrite(IN1,HIGH);
25     digitalWrite(IN2,HIGH);
26     delay(5000);
27
28     //Sentido Anti-Horario
29     digitalWrite(IN1,LOW);
30     digitalWrite(IN2,HIGH);
31     delay(5000);
32
33     //travaMotor
34     digitalWrite(IN1,HIGH);
35     digitalWrite(IN2,HIGH);
36     delay(5000);
37
38     /*Fim dos Estados do motor A*/
```

Figura 47 - Código teste do Controlador Ponte H



*Figura 48 - Teste com Controlador Ponte H e motor TT*

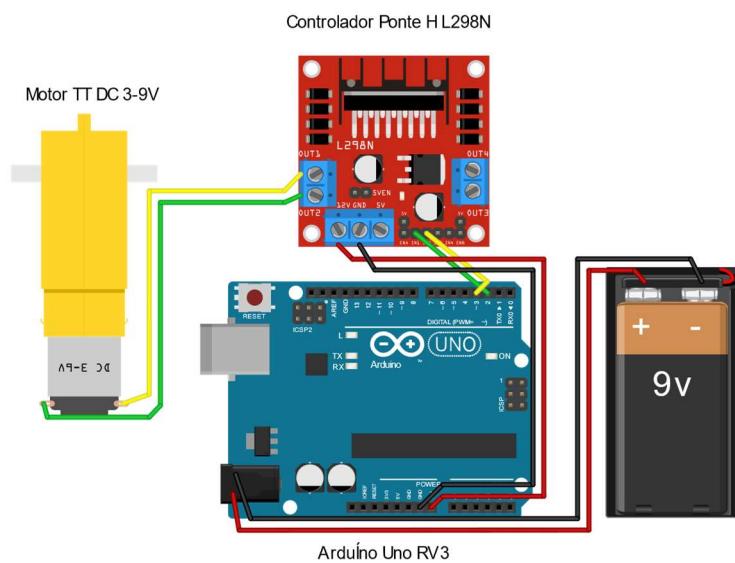


Figura 49 - Circuito de teste com Controlador Ponte H e motor TT

Em seguida, com os motores ligados ao Controlador Ponte H e ao Arduíno Uno, comecei a estudar os Módulos Sensores Infravermelhos. O meu objetivo era perceber e estudar a variação dos valores de distância e ângulos para obter as leituras mais precisas possíveis. Também calculei e defini a meta de valores entre as superfícies branca e preta. Estabeleci que valores de 0 a 50 seriam identificados como superfície branca, enquanto valores acima de 50 seriam considerados superfície preta.

```

2
3 void setup() {
4     pinMode(A0, INPUT);
5     Serial.begin(9600);
6 }
7
8 void loop() {
9     int Branco;
10    int Preto;
11    int valorAnalogico = analogRead(A0);
12
13    if (valorAnalogico <= 50) {
14        Branco = valorAnalogico;
15        Serial.print(Branco);
16        Serial.println("---Branco");
17    } else {
18    }
19
20    if (valorAnalogico >= 50) {
21        Preto = valorAnalogico;
22        Serial.print(Preto);
23        Serial.println("---Preto");
24    } else {
25    }
26
27    delay(300);
28 }
```

12:22:03.750 -> 662---Preto 12:22:04.050 -> 640---Preto 12:22:04.364 -> 640---Preto 12:22:04.634 -> 649---Preto 12:22:04.931 -> 672---Preto 12:22:05.261 -> 648---Preto 12:22:05.560 -> 234---Preto 12:22:05.832 -> 36---Branco 12:22:06.162 -> 26---Branco 12:22:06.429 -> 27---Branco 12:22:06.742 -> 27---Branco 12:22:07.027 -> 25---Branco 12:22:07.359 -> 27---Branco 12:22:07.623 -> 27---Branco 12:22:07.955 -> 28---Branco 12:22:08.219 -> 28---Branco
--

Figura 51 - Código teste do Módulo Sensor Infravermelhos

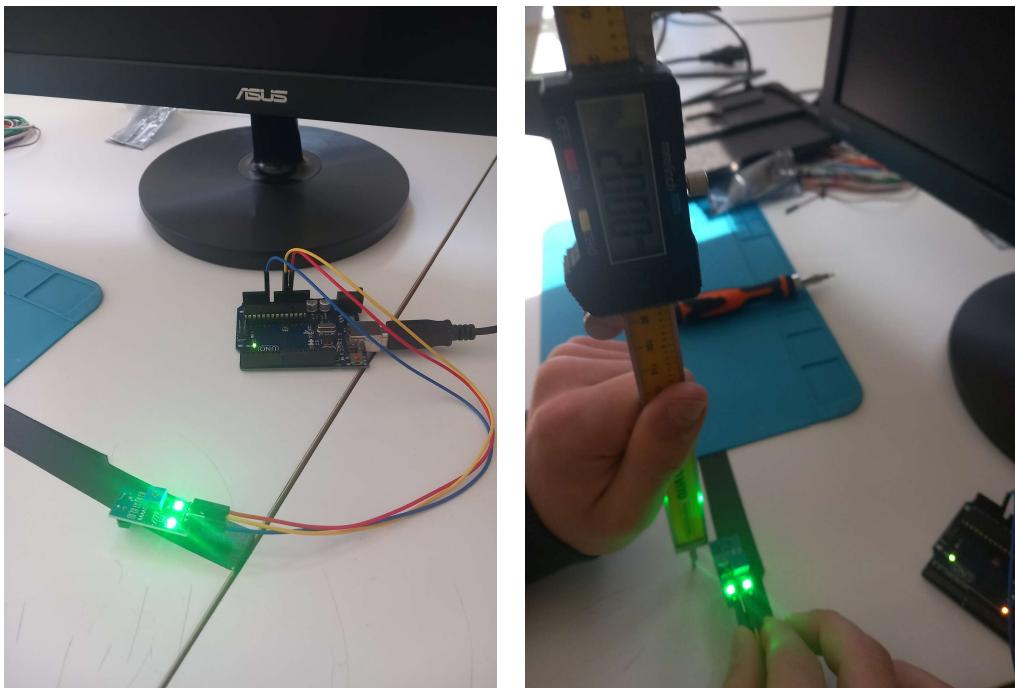


Figura 50 - Testes com Módulo Sensor Infravermelhos

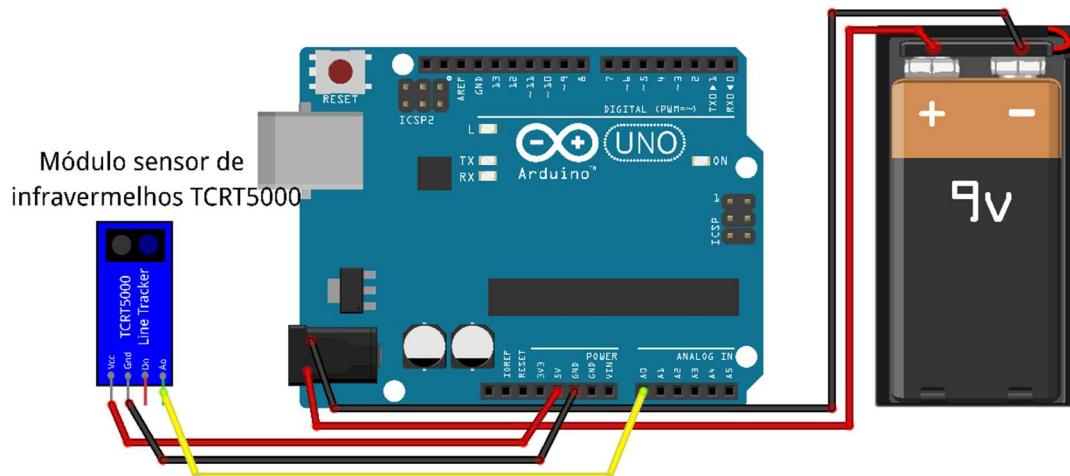


Figura 52 - Circuito teste com Módulo Sensor de Infravermelhos

Após os testes com os Módulos Sensores Infravermelhos, procedi ao teste do Sensor de Cor RGB TCS3200.

Este foi conectado diretamente ao Arduíno Uno sem quaisquer dificuldades. Posteriormente, foi impressa uma folha com as cores projetadas a serem usadas no projeto, como o vermelho, azul e verde. Utilizei essa folha para perceber a receção de valores, estudá-los e apontá-los para fazer comparações que identificassem as cores.

```
#include <tcs3200.h>
int red, green, blue, white;
tcs3200 tcs(4, 5, 6, 7, 8); // (S0, S1, S2, S3, output pin) //
void setup() {
    Serial.begin(9600);
}
void loop() {
    //red = tcs.colorRead('r', 0);    //scaling can also be put to
    //red = tcs.colorRead('r', 20);
    //red = tcs.colorRead('r', 100);

    red = tcs.colorRead('r');    //reads color value for red
    Serial.print("R= ");
    Serial.print(red);
    Serial.print("  ");

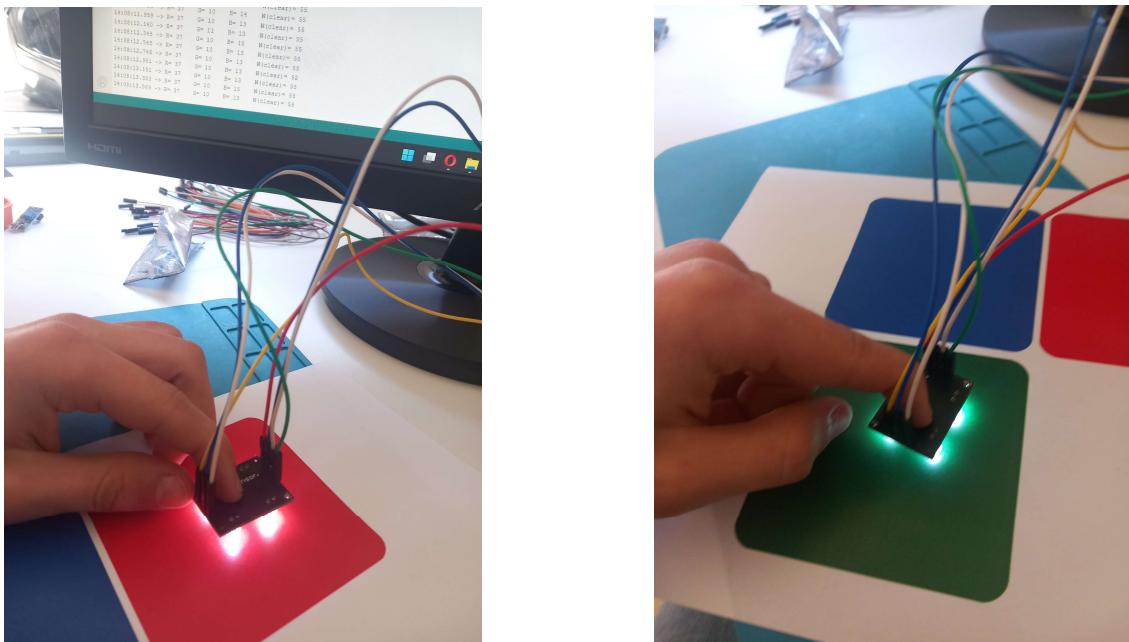
    green = tcs.colorRead('g');    //reads color value for green
    Serial.print("G= ");
    Serial.print(green);
    Serial.print("  ");

    blue = tcs.colorRead('b');    //reads color value for blue
    Serial.print("B= ");
    Serial.print(blue);
    Serial.print("  ");

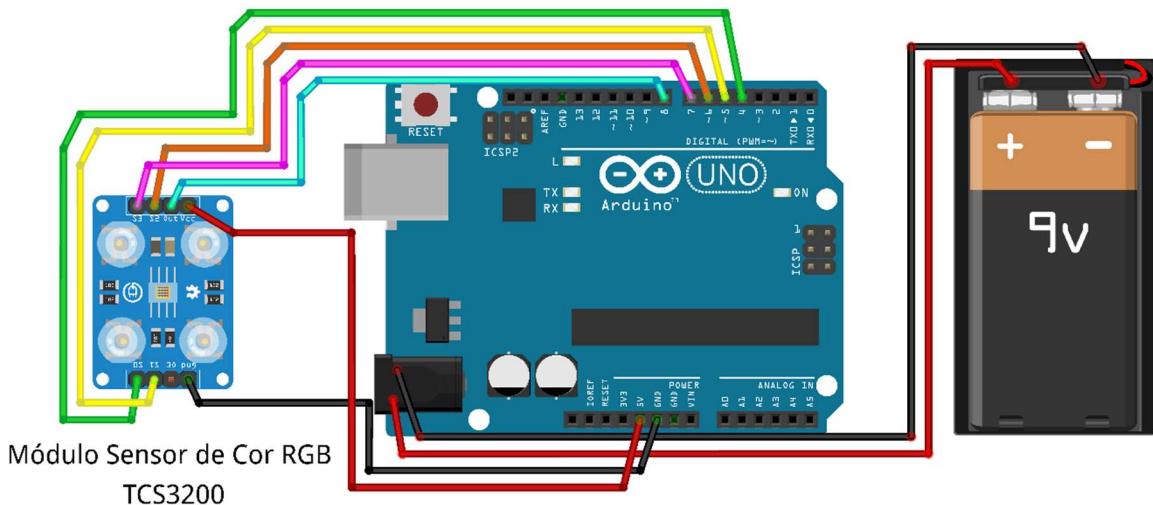
    white = tcs.colorRead('c');    //reads color value for white
    Serial.print("W(clear)= ");
    Serial.print(white);
    Serial.print("  ");

    Serial.println();
    delay(200);
}
```

Figura 53 - Código teste do Módulo Sensor de Cor RGB



*Figura 54 - Teste do Módulo Sensor de Cor RGB*



*Figura 55 - Circuito teste com Módulo Sensor de Cor RGB*

Após concluir todos estes testes, considero concluída a primeira fase do projeto. Esta fase consistiu em colocar em prática as funções primárias, nomeadamente:

- Desenhar o robô e os seus suportes para os componentes e funcionalidades.
  - Estudar e compreender as funções de cada componente e a sua programação.
  - Estabelecer a comunicação entre os componentes ao Arduíno.

### 2.6.1.1.2 FASE INTERMÉDIA

#### 2.6.1.1.2.1 CONSTRUÇÃO DO PROJETO

Na Fase Intermédia do projeto, iniciei a construção do robô com as peças projetadas e impressas em 3D. Durante o processo de construção, não ocorreu nenhum problema, pois todas as peças foram projetadas para se encaixarem perfeitamente, sem qualquer dificuldade. Importante realçar que o projeto foi concebido de modo que as peças fossem fixadas com parafusos M3 de 6mm e 8mm de comprimento.

Primeiramente, comecei a anexar os dois suportes dos motores TT ao chassi do robô e, posteriormente, instalando os motores nos mesmos suportes.

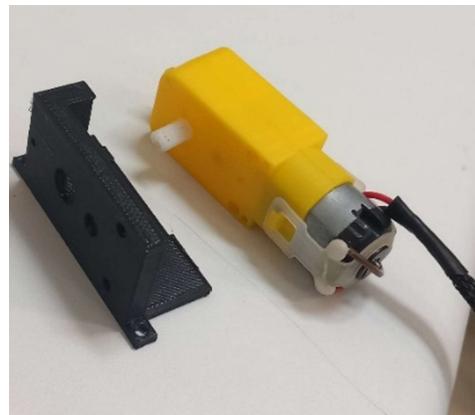
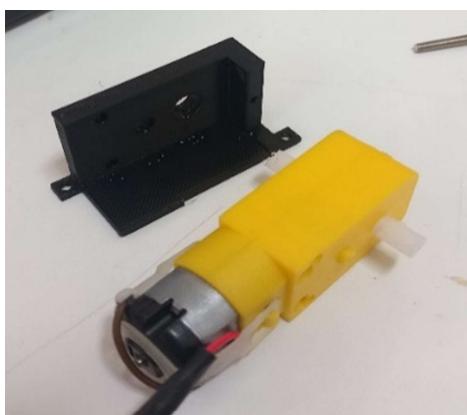


Figura 56 - Motores TT e suportes

Após a instalação dos motores no chassi, procedi à instalação das placas de controlo, incluindo o Arduíno Uno e o Controlador de Ponte H. Não foram encontrados quaisquer problemas durante essa fase de instalação.

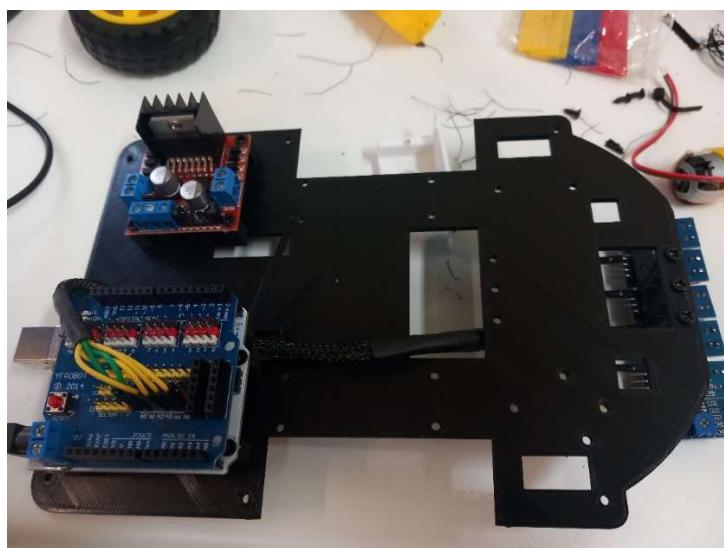


Figura 57 - Placas controladoras afixadas no chassi



Durante o processo de fixação das placas de controlo, também foi instalado um suporte para acomodar os Módulos Sensores Infravermelhos, aproveitando também para instalar os 6 Módulos Sensores Infravermelhos TCRT5000.



Figura 58 - Afixação dos Módulos Sensores Infravermelhos no suporte

Após a instalação dos Módulos Sensor Infravermelhos, o sistema de descarga foi anexado ao chassi do robô sem encontrar quaisquer dificuldades. Aproveitando essa etapa, também foram instalados o Servo Motor SG90 e o Sensor de Cor RGB TCS3200 nos suportes designados. Além disso, foi aplicado um tecido na área de carga. Esta medida foi tomada devido ao facto de que as caixas usadas para testar o robô serem bastante leves, tornando a superfície escorregadia. Isso poderia resultar na queda da carga sempre que o robô executasse movimentos bruscos. Essas adaptações garantem uma manipulação mais segura e eficiente da carga, prevenindo potenciais contratemplos durante a operação do robô.

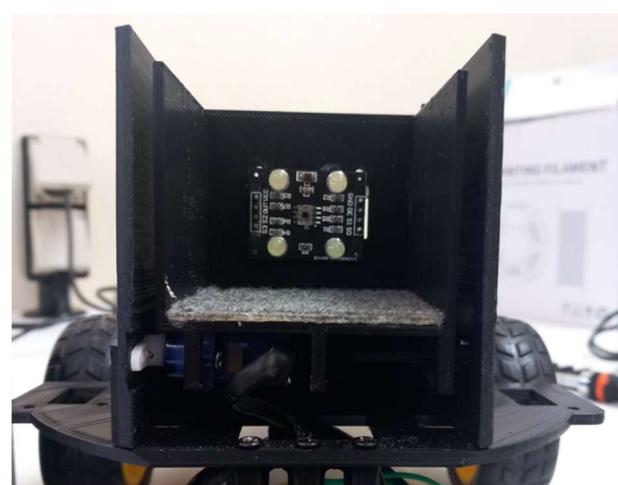
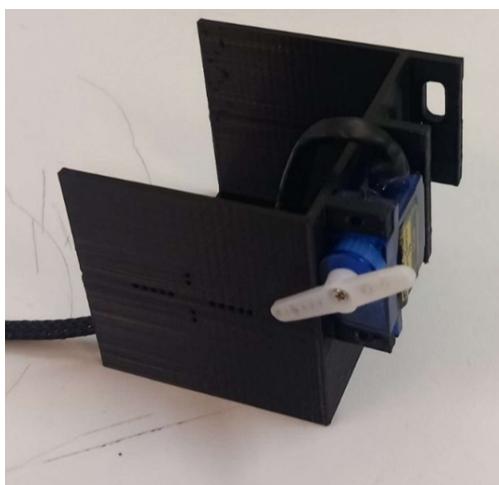


Figura 59 – Processo de montagem final do sistema basculante



Todas as peças foram assim fixadas ao chassi do robô com sucesso e conforme o planeado. O cabeamento também foi cuidadosamente planeado e executado. Como mencionado anteriormente, optou-se por utilizar cabos DuPont coloridos. De forma a não afetar o aspeto visual do robô, proteger os cabos contra danos e manter a organização do cabeamento para facilitar o trabalho, foi utilizada uma manga trançada preta ao longo do percurso dos cabos, sendo as suas extremidades finalizadas com mangas térmicas pretas para evitar que a manga se desfizesse e para garantir que permanecesse fixa ao cabo.

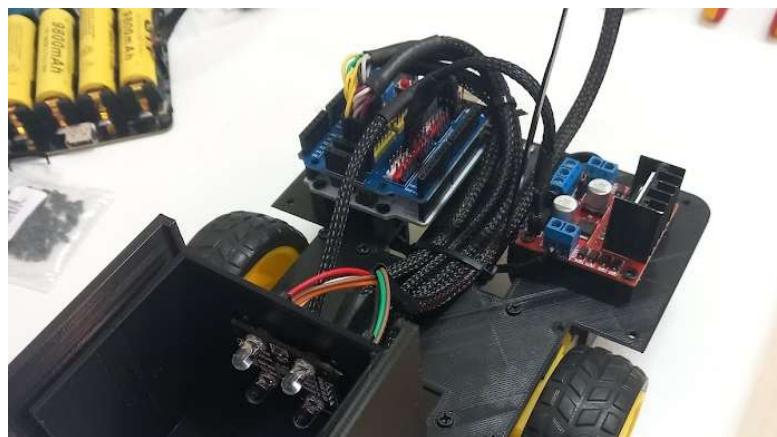


Figura 60 - Demonstração da organização e preparação de cabos

#### 2.6.1.1.2.2 PROGRAMAÇÃO DO PROJETO

A programação do projeto foi desafiadora e levou mais tempo do que o esperado devido à minha pouca experiência em programação em C++. Após muitos dias de tentativas, erros e desafios enfrentados, consegui concluir a programação com sucesso, embora ainda haja espaço para melhorias, naturalmente.

Primeiramente, declarei as variáveis necessárias e adequadas para os tipos de valores recebidos ou emitidos, adicionando também as bibliotecas essenciais, como para o Micro Servo SG90. Além disso, configurei os pinos e defini-os como saídas ou entradas (OUTPUT, INPUT) conforme necessário.



```
PAP_FINAL.ino
1 #include <servo.h> // Biblioteca que permite o funcionamento do servo motor
2
3 Servo boxServo;
4
5 // Sensor de Cor TCS3200
6 #define S0 2
7 #define S1 3
8 #define S2 8
9 #define S3 9
10 #define sensorOut 10
11 int frequency = 0; // Variável para armazenar a frequência
12 int color = 0; // Variável para armazenar a cor detectada
13
14 // Motor L298N
15 int mRight = 5; // Pino de controle do motor direito
16 int dirRight = 4; // Pino de direção do motor direito
17 int mLeft = 6; // Pino de controle do motor esquerdo
18 int dirLeft = 7; // Pino de direção do motor esquerdo
19
20 // Sensor de Linha
21 const int numPins = 6; // Número de pinos analógicos (A0 a A5)
22 int sensorValues[numPins]; // Array para armazenar leituras analógicas
23
24 int sensorVal[6] = { 0, 0, 0, 0, 0, 0 }; // Array para armazenar valores binários dos sensores de linha
25 int error, last_error, MV, pid_l, pid_r, D, D1, D2, D3, I, I1, I2, I3, P, Pd, bitSensor;
26 int Max_MV;
27 unsigned char Kp = 7.5; // Kp é um coeficiente que determina a magnitude da correção proporcional aplicada num sistema de controlo,
28 ||||| //com base no erro entre a posição desejada e a posição real.
29 unsigned char Kd = 0; // Kd é um coeficiente que determina como a correção do controlador PID é influenciada pela taxa de variação do erro.
30 unsigned char Ts = 1;
31 unsigned char maxPwm = 170; // maxPwm define a velocidade máxima dos motores
32 unsigned char intersection = 0; // Variável para armazenar o número de intersecções
33
34 void setup() {
35     Serial.begin(9600); // Inicialização da comunicação serial
36     // Configuração dos pinos do motor L298N
37     pinMode(mRight, OUTPUT);
38     pinMode(dirRight, OUTPUT);
39     pinMode(mLeft, OUTPUT);
40     pinMode(dirLeft, OUTPUT);
41     // Configuração dos pinos do sensor de linha
42     pinMode(A0, INPUT);
43     pinMode(A1, INPUT);
44     pinMode(A2, INPUT);
45     pinMode(A3, INPUT);
46     pinMode(A4, INPUT);
47     pinMode(A5, INPUT);
48     // Configuração dos pinos do sensor de cor
49     pinMode(S0, OUTPUT);
50     pinMode(S1, OUTPUT);
51     pinMode(S2, OUTPUT);
52     pinMode(S3, OUTPUT);
53     pinMode(sensorOut, INPUT);
54     digitalWrite(S0, HIGH);
55     digitalWrite(S1, LOW);
56     boxServo.attach(11); // Anexar o servo motor
57     boxServo.write(35); // Movimento inicial do servo motor
58 }
```

*Figura 61 - Excerto de código, Variáveis Declaradas*



Em seguida, criei as funções "void", onde cada função executará uma tarefa específica do robô. Essas funções ajudaram a tornar a programação do robô mais organizada e dinâmica. Pensei em funções, como:

- **mission()** – Esta função, chamada de "mission", em português missão, é a função principal do robô. Ela é responsável por instruir o robô a seguir a linha, ler a cor da carga, detetar as linhas de início e fim, e as interseções. Além disso, esta função também é responsável por definir a rota com base na cor detetada.

```
176 void mission() {  
177     // Função para coordenar as operações do robô  
178     robotRun();           // Executar o controlo do robô  
179     boxServo.write(35);   // Mover o servo motor  
180     // Verificar se o robô está em uma interseção  
181     if (sensorVal[5] == 1 && sensorVal[0] == 1) {  
182         stopRun();        // Parar o movimento do robô  
183         color = readColor(); // Ler a cor detectada  
184     }  
185     // Executar ações com base na cor detectada  
186     switch (color) {  
187         case 1: // Vermelho: Virar à esquerda  
188             red();  
189             break;  
190         case 2: // Verde: Seguir em frente  
191             green();  
192             break;  
193         case 3: // Azul: Virar à direita  
194             blue();  
195             break;  
196         case 0: // Nenhuma cor detectada: Parar  
197             stopRun();  
198             break;  
199     }  
200 }  
201 }  
~~~
```

Figura 62 - Excerto de código, Função mission()

- **robotRun()** – A função é a que faz permitir que o robô siga a linha de forma correta e acertada, calculando velocidades e margem de erros que depois definem a velocidade do robô a seguir a linha.  
Mais detalhes no tópico 2.2.1 Código seguidor de linha usando PID.

- **readSens()** – Como o nome sugere, esta função vai ler os 6 módulos de sensor infravermelho e vai armazenar os valores obtidos numa matriz declarada. Para simplificar o código, os valores obtidos diretamente dos módulos de sensor infravermelho estavam na faixa de 0 a 1023. Para facilitar a programação, esses valores foram convertidos em binário (0, 1) usando comparações com um valor de referência de 125.

```

66 void readSens() {
67     // Função para ler os valores dos sensores de linha
68     for (int analogPin = A0; analogPin <= A5; analogPin++) {
69         int sensorValue = analogRead(analogPin); // Leitura do valor analógico do sensor
70         Serial.print(sensorValue); // Imprimir o valor lido
71         Serial.print('\t');

72         sensorValues[analogPin - A0] = sensorValue; // Armazenar o valor lido no array
73         if (sensorValue >= 125) {
74             sensorVal[analogPin - A0] = 1; // Converter o valor para binário (0 ou 1)
75         } else {
76             sensorVal[analogPin - A0] = 0;
77         }
78         Serial.print(sensorVal[analogPin - A0]); // Imprimir o valor binário
79         Serial.print('\t');
80     }
81     Serial.println();
82 }
83 }
```

Figura 63 - Excerto de código, Função readSens()

- **readColor()** - Esta função é responsável por ler a cor da carga, armazenando os valores obtidos e realizando comparações para classificar as cores. No final, as cores são representadas por números, onde o número 1 corresponde ao vermelho, o número 2 à cor verde e o número 3 ao azul. Sem esta função, o funcionamento do robô seria impossível. Mais detalhes no tópico 2.2.2 *Leitura e classificação de cores*.
- **red(), green(), blue()** - Estas funções são responsáveis por definir a ação do robô com base na cor classificada pela função readColor(). Por exemplo, se a cor for vermelha, a função red() instruirá o robô a virar à esquerda na interseção. Cada função representa uma direção diferente. Mais detalhes no tópico 2.2.3 *Escolha dos trajetos*.
- **Forward(), straight(), turnRight(), turnLeft(), turnBack(), stopRun()** – Estas funções são responsáveis pelos movimentos do robô. Essas funções coordenam os motores do robô para realizar as ações desejadas, como avançar, seguir em linha reta, virar à direita, virar à esquerda, dar meia-volta e parar o movimento. Elas desempenham um papel fundamental na execução das tarefas do robô de acordo com a lógica do programa.



### 2.6.1.1.3 FASE FINAL

Na fase final deste projeto, instalei e conectei todos os componentes que foram testados e estudados durante a Fase Intermediária ao Arduino Uno. Foi uma tarefa bastante fácil, pois já havia planeado onde os cabos DuPont seriam conectados na placa durante a programação. A conexão dos cabos foi pensada de forma que o projeto fosse organizado e fácil de compreender nas ligações feitas.

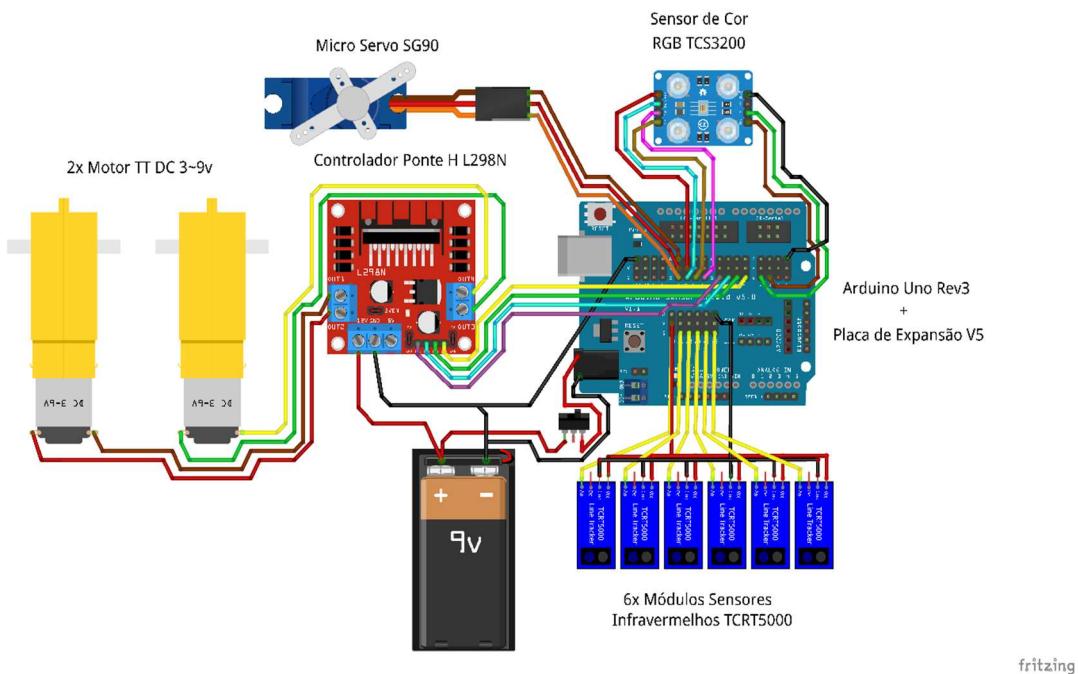


Figura 64 - Circuito geral do Projeto

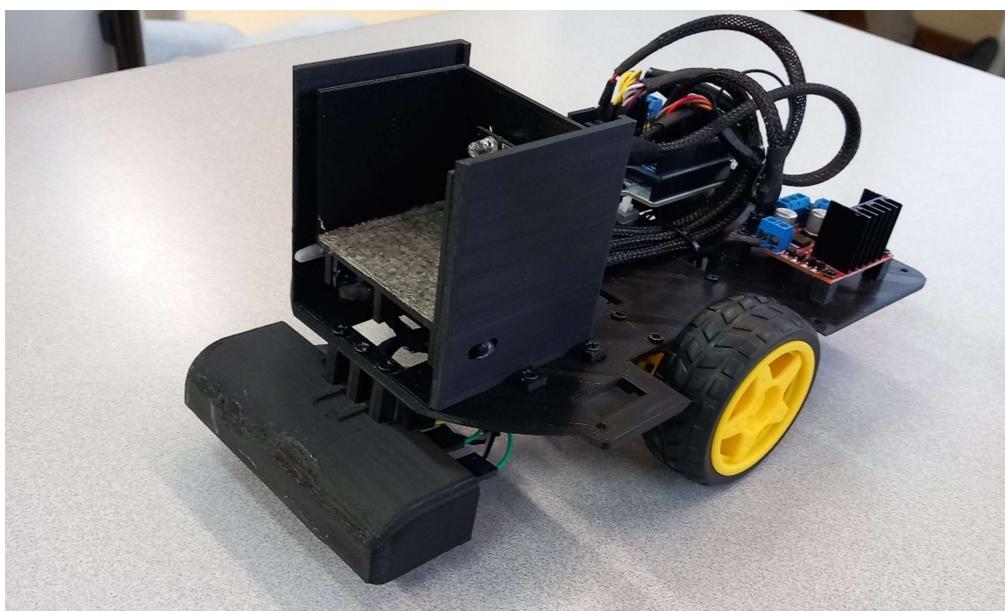


Figura 65 - Projeto Final



## 2.7 ORÇAMENTO

Nos tópicos seguintes serão demonstradas as tabelas dos custos de todos os componentes e matérias utilizados para o desenvolvimento do projeto. O tempo utilizado para a construção e o tempo para a codificação também serão aqui contados.

### 2.7.1 ORÇAMENTO DOS COMPONENTES UTILIZADOS

Componentes	Unit.	Preço Unit.	Preço Total
<b>Development Module Compatible with Arduino Uno w/ ATmega328 - WHADDA WPB100</b>	<b>1</b>	<b>23,20 €</b>	<b>23,20 €</b>
<b>Sensor Shield V5 Para ARDUINO UNO R3 - COLORIDO</b>	<b>1</b>	<b>1,49 €</b>	<b>1,49 €</b>
<b>Motor micro servo 4.8V..6V DC SG90 - 180º</b>	<b>1</b>	<b>4,63 €</b>	<b>4,63 €</b>
<b>Sensor de Cor RGB TCS230/TCS3200</b>	<b>1</b>	<b>2,94 €</b>	<b>2,94 €</b>
<b>L298N Dual H-Bridge Motor Driver</b>	<b>1</b>	<b>1,96 €</b>	<b>1,96 €</b>
<b>Módulo de sensor reflexivo infravermelho TCRT5000 – 10 Unid.</b>	<b>1</b>	<b>4,93 €</b>	<b>4,93 €</b>
<b>Motor de engrenagem DC TT – 2 Unid.</b>	<b>1</b>	<b>3,94 €</b>	<b>3,94 €</b>
<b>Bateria de Lítio Recarregáveis Micro USB, 9 V, 12800mAh</b>	<b>1</b>	<b>8,71 €</b>	<b>8,71 €</b>
<b>Manga trançada preta, 4 mm, 10 M</b>	<b>1</b>	<b>5,17 €</b>	<b>5,17 €</b>
<b>Fios DuPont 30 cm – 120 Unid.</b>	<b>1</b>	<b>14,33 €</b>	<b>14,33 €</b>
<b>Filamento PLA 1000 g Preto</b>	<b>1</b>	<b>21,34 €</b>	<b>21,34€</b>
<b>Fita Isolante Preto 18mm</b>	<b>2</b>	<b>2,7 €</b>	<b>1,35 €</b>
<b>Parafusos Aço Carbono M3 8mm, 6mm</b>	<b>1</b>	<b>3,45 €</b>	<b>3,45 €</b>
<b>Valor total dos componentes do projeto:</b>			<b>97,44 €</b>

Tabela 1 - Orçamento dos componentes utilizados





## 2.7.2 HORAS DE MÃO-DE-OBRA

Esta tabela mostra o tempo gasto em cada tarefa da construção do robô, desde o início até ao final. Abrangendo todo os tipos de tarefas realizadas no projeto. Análise detalhada das tarefas na página seguinte.

Horas de Mão-de-Obra	
Tarefas	Número de horas
Pesquisa	16 H
Programação	75 H
Testes	28 H
Desenho 3D	350 H
Impressão 3D	34 H
Construção da Robô	45 H
Relatório/Apresentação	85 H
<b>Total de Horas</b>	<b>633 H</b>
<b>Total em Valor</b>	<b>633 H x 10 = 6 330 €</b>

Tabela 2 - Horas de mão de obra



- Pesquisa (16 Horas):
  - A pesquisa foi fundamental para o projeto. Nela, estudei e analisei projetos similares, com foco em seus componentes e funcionalidades. Li as fichas técnicas (Datasheet) de cada componente para avaliar sua adequação ao nosso projeto.
- Programação (75 Horas):
  - A programação foi a etapa demorada, com 75 horas, aqui criei o código que controla o comportamento do robô, definindo suas funções e movimentos. Foi um desafio por causa da complexidade do projeto e da minha pouca experiência com a linguagem C++.
- Testes (28 Horas):
  - A etapa de testes e ajustes foi importante para garantir o bom funcionamento do robô. Durante essa fase, realizei diversos testes em diferentes situações, com isto, identificar falhas e realizar ajustes para melhorar o comportamento do robô.
- Desenho 3D (350 Horas):
  - Nesta etapa, foram desenhadas e pensadas a aparência, estrutura e disposição de todos os componentes. Investi 350 horas nessa tarefa devido à sua complexidade e à minha inexperiência na área designada. Mais detalhes no tópico 2.6.1.1.1. *Fase Inicial*.
- Impressão 3D (34 Horas):
  - Nesta fase, as peças desenhadas foram fisicamente produzidas através da impressão 3D. Investi um total de 34 horas nesse processo, que foi afetado tanto pela velocidade de impressão de cada peça e também pelos erros que ocorreram durante o processo de impressão 3D.
- Construção do projeto (45 Horas):
  - Nesta fase o projeto ganhou forma, aqui foram afixadas todas as peças e desenhadas e impressas. Também como a instalação dos componentes eletrónicos e as suas ligações e a placa controladora. Mais detalhes no tópico 2.6.1.1.2.1. *Construção do projeto*.
- Relatório/Apresentação (85 Horas):
  - Aqui foram realizados o relatório e a apresentação do projeto. Este tempo foi contabilizado pelo Microsoft Word e Microsoft PowerPoint cerca 85 horas no total. Este tempo encontrasse devido a complexidade do projeto.

No total, foram dedicadas 633 horas ao projeto. O custo total das horas foi calculado considerando um valor de 10 euros por hora, resultado de uma multiplicação do número total de horas pelo valor por hora.

### 2.7.3 CUSTOS TOTAIS DO PROJETO

Nesta tabela estão apresentados os custos totais do projeto, sendo que o total foi calculado somando os custos totais dos equipamentos juntamente com o custo da mão de obra.

Custos totais do projeto	
Componentes	Valor
Custo de Equipamentos	97,44 €
Custo de Mão-de-Obra	6 330 €
<b>Total:</b>	<b>6 427,44 €</b>

Tabela 3 - Custos totais do projeto



### 3. CONCLUSÃO E REFLEXÃO DO PROJETO

Neste ponto será apresentada a conclusão e reflexão do projeto.

#### 3.1 ANÁLISE DE PLANEAMENTOS

É aqui feita uma análise do planeamento proposto face ao planeamento seguido, bem como uma conclusão comparativa de ambas.

##### 3.1.1 PLANEAMENTO INICIAL

Neste ponto, está apresentado o planeamento inicial, elaborado a partir de um gráfico de Gantt.



##### 3.1.2 PLANEAMENTO FINAL

O projeto começou no início do curso e foi evoluindo gradualmente, acompanhando as aprendizagens e necessidades. Isso permitiu cumprir todos os prazos, inclusive que foi estendido, aproveitando o tempo extra para melhorias.

##### 3.1.3 CONCLUSÃO SOBRE O PLANEAMENTO

Ao aderir à pontualidade dos prazos e distribuir o trabalho ao longo do tempo, em vez de deixá-lo acumular até o último momento, torna-se viável realizar as tarefas de forma mais serena. Dessa maneira, é possível identificar e aprimorar aspectos que possam estar mais frágeis ou não estejam funcionando tão eficazmente.



### 3.2 CONCLUSÃO DO TRABALHO DE PAP REALIZADO

Com a finalização deste projeto, pude constatar a quantidade de trabalho, estudo e conhecimentos necessários para sua concretização. Ao longo desta jornada, deparei-me com diversos desafios que me proporcionaram uma aprendizagem significativa, tanto dentro da minha área do curso quanto em áreas diferentes.

Um dos maiores desafios foi a área de design, na qual não tive formação específica. No entanto, com dedicação e estudo, consegui desenvolver as habilidades necessárias para criar um design visualmente atraente e profissional para o projeto.

Embora árduo, o caminho percorrido proporcionou-me crescimento profissional e intelectual. A cada obstáculo superado, a sensação de conquista e aprendizagem se intensificavam cada vez mais. O resultado, um projeto completo e bem-sucedido, é a prova do meu empenho e dedicação.

Sinto-me realizado com o resultado. Todos os objetivos traçados foram atingidos, e o projeto superou as minhas expectativas. Acredito que esta experiência me tornou um profissional mais completo e capacitado para enfrentar os desafios do mercado de trabalho.

### 3.3 SUGESTÕES/MELHORIAS FUTURAS

Embora o projeto esteja num estado avançado de desenvolvimento e atenta aos requisitos iniciais, ainda existem algumas áreas que podem ser aprimoradas no futuro. As seguintes melhorias visam aumentar a funcionalidade, a usabilidade e a confiabilidade do projeto.

- Desenvolvimento de uma aplicação móvel:
  - Permitirá ao usuário aceder ao estado do robô em tempo real.
  - Possibilitará o controlo remoto do robô, aumentando a flexibilidade.
  - Enviará notificações sobre eventos importantes do robô (bateria fraca, tarefa concluída, etc..).
- Ecrã informativo na estrutura do robô:
  - O usuário poderá visualizar o estado atual do robô (ligado, a funcionar, em pausa, etc...).
  - Serão exibidas mensagens de erro e alertas para auxiliar na resolução de problemas.
  - O usuário poderá configurar parâmetros e preferências diretamente no robô.
- Substituição da bateria por um modelo de maior capacidade energética:
  - Aumentará o tempo de funcionamento do robô sem necessidade de recarga frequente.
  - Permitirá que o robô execute tarefas mais longas sem interrupções.
- Implementação de sensores ultrassónicos para deteção de obstáculos:
  - Evitará colisões com objetos, protegendo o robô e o ambiente.
  - Melhorará a navegação autónoma em espaços complexos.
- Reforma do Aspetto Físico do Robô
  - Adicionar uma estrutura ao robô para fornecer mais segurança ao equipamento.

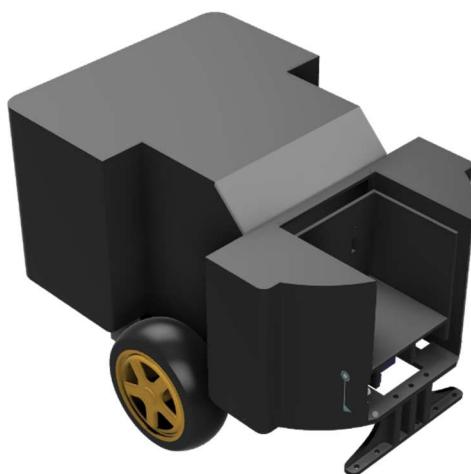


Figura 66 - Reforma do Aspetto Físico do Robô



#### 4. NETGRAFIA

Para a elaboração deste projeto houve muita pesquisa na internet e apontamentos tirados ao longo dos 3 anos de curso, o que foi bastante útil.

<https://forum.arduino.cc/>

<https://chat.openai.com/>

<https://forums.autodesk.com/>

<https://robotresearchlab.com/2019/02/16/pid-line-follower-tuning/>

<https://robotresearchlab.com/2019/02/12/how-to-program-a-line-following-robot/>

<https://www.aurothschild.net/posts/PID-line-follower/>

<https://www.hackster.io/anova9347/line-follower-robot-with-pid-controller-cdedbd>

<https://www.instructables.com/Line-Follower-Basic-Using-Arduino-Nano/>



## 5. ANEXOS

Neste capítulo, encontram-se os anexos do presente projeto. Cada código QR fornecerá um link direto para o documento digital correspondente.

### 5.1 REPOSITÓRIO DOS ANEXOS DO PROJETO:



Figura 67 - Código Qr, Repositório (<https://bit.ly/3ITxSL2>)

### 5.2 ESBOÇOS E MODELOS 3D



Figura 68 - Código Qr, Esboços e Modelos 3D (<https://bit.ly/3PCcGgl>)



### 5.3 DOCUMENTOS DO PROJETO



Figura 69 - Código Qr, Documentos do projeto (<https://bit.ly/3xd5mBl>)

### 5.4 DATASHEET DOS COMPONENTES UTILIZADOS



Figura 70 - Código Qr, Datasheet (<https://bit.ly/3xcLsGJ>)

DATA 27/03/2024

ASSINATURA DO ALUNO : \_\_\_\_\_

ASSINATURA DO ORIENTADOR DE PAP : \_\_\_\_\_

ASSINATURA DO DIRETOR PEDAGÓGICO : \_\_\_\_\_