Universidade de Évora - Escola de Ciências e Tecnologia

Mestrado em Engenharia Mecatrónica

Trabalho de Projeto

Desenvolvimento de um Veiculo Autoguiado

Miguel Alexandre Machado Lança Silva

Orientador(es) | Fernando Manuel Janeiro

João Manuel Figueiredo

Évora 2025

Universidade de Évora - Escola de Ciências e Tecnologia

Mestrado em Engenharia Mecatrónica

Trabalho de Projeto

Desenvolvimento de um Veiculo Autoguiado

Miguel Alexandre Machado Lança Silva

Orientador(es) | Fernando Manuel Janeiro

João Manuel Figueiredo

Évora 2025

O trabalho de projeto foi objeto de apreciação e discussão pública pelo seguinte júri nomeado pelo Diretor da Escola de Ciências e Tecnologia:

Presidente	|	Mouhaydine Tlemcani (Universidade de Évora)
Vogais	|	Fernando Manuel Janeiro (Universidade de Évora) (Orientador) Oumaima Mesbahi (Universidade de Évora) (Arguente)

Évora 2025

Resumo

Desenvolvimento de um Veículo Autoguiado

Os veículos guiados autonomamente (AGV) têm se tornado uma parte essencial da automação industrial, sendo amplamente utilizados em diversos setores devido à sua capacidade de realizar tarefas de forma eficiente e precisa. O presente relatório explora a implementação e os benefícios dos AGVs em ambientes de produção, destacando a importância dessa tecnologia na otimização de processos. Neste trabalho é explorado os diferentes tipos de AGV´s, e as suas diferentes aplicações. É também exploradas as diferentes capacidades da visão artificial usada nas diferentes indústrias e finalmente como foi aplicada a este projeto.

Este projeto consiste no desenvolvimento de um veículo com a base num microcontrolador Raspberry Pi 3A+ que tem como objetivo seguir uma linha no chão a partir de várias funções pré-determinadas, tendo em consideração que se trata de um sistema fechado em que todo o processamento e tomada de decisão ocorre localmente. Além do Raspberry Pi foi também usada uma câmera para a aquisição de imagens, uma placa de controlo de motores para controlar o movimento dos motores, os respetivos motores e finalmente uma

bateria recarregável para alimentar todo o sistema.

Abstract

Development of an Automated Guided Vehicle

Autonomous guided vehicles (AGVs) have become an essential part of industrial automation, being widely used in various sectors due to their ability to perform tasks efficiently and accurately. This report explores the implementation and benefits of AGVs in production environments, highlighting the importance of this technology in process optimization. This work explores the different types of AGVs and their different applications. It also explores the different capabilities of machine vision used in different industries and finally how it was applied to this project.

This project consists of developing a vehicle based on a Raspberry Pi 3A+ microcontroller that aims to follow a line on the ground from various predetermined functions, taking into account that it is a closed system in which all processing and decision-making takes place locally. In addition to the Raspberry Pi, a camera was also used to acquire images, a motor control board to control the movement of the motors, the respective motors and finally a rechargeable battery to power the entire system.

List of Figures

Contents

Chapter 1

Introdução

Desde há muito tempo que o Homem procura ver as suas tarefas realizadas de forma automatizada. Este desejo consubstancia-se na busca de mecanismos que promovam um trabalho semelhante àquele que o ser-humano desenvolve, buscando ainda sistemas que proporcionem um desempenho acima das suas capacidades com vista a aumentar a produtividade assim como a qualidade dos processos.

Esta procura pelo desenvolvimento destes mecanismos conduziram àquilo que hoje é tida como uma área científica interdisciplinar que apelidamos de robótica. Assim, é possível desenvolver um largo número de sistemas autónomos. Debrucemos a nossa atenção aquando da robótica no contexto industrial, que como consequência das limitações existentes relativamente a questões de gestão e organização, veio responder à necessidade de melhorar o rendimento destes procedimentos. Deste modo, surgem os AGV´s(Automated Guided Vehicles), que numa resposta aos maiores volumes de produção, consistem em diversos sistemas com poder de processamento, que irão permitir

a tomada de decisões ponderadas e racionais baseadas em matrizes definidas.

Com isto, pretende-se no relatório em causa mostrar o desenvolvimento de um AGV preparado por um sistema de visão artificial, esclarecendo o seu processo de criação, com a finalidade de vir a cumprir o seu potencial num ambiente real. Isto passa pela elaboração de um esqueleto físico que suporte todo o hardware do AGV numa única estrutura, de forma a assegurar a integridade física dos vários componentes. Toda esta armação é modelada através do software SOLIDWORKS, onde todas as partes funcionais, excetuando as ligações aparafusas e asseguradas por ferragens em metal, irão ser produzidas por deposição de material fundido (Fused Deposition Modeling – FDM), a partir de um termoplástico, a fim de acautelar uma estrutura individualizada e menos pesada, permitindo aliviar o esforço mecânico nos motores elétricos responsáveis pela deslocação do AGV.

A montante, não devemos descurar um outro objetivo deste relatório, que será o de revelar o progresso científico nesta área, mais do que não seja pelo desenvolvimento de um veículo autoguiado conforme o modelo de POC(Proof Of Concept), dando ênfase à criação de um sistema que recorre a serviços Open-Source. Desta forma, fica assegurada a livre exposição dos vários componentes utilizados, facilitando e promovendo um futuro sem limitações.

Destarte, e num espírito de preservar este sistema, pretende o presente AGV cumprir com a garantia de um trajeto autónomo, baseado num processo de aquisição de imagem em tempo real, recorrendo a um modelo de visão artificial, a juntar a um algoritmo desenvolvido em Python.

Devemos exigir que todos estes elementos contribuam para a produção de um sistema preciso e rigoroso, e isto no seguimento dos trajetos definidos

pelo utilizador num contexto real.

Chapter 2

AGV´s

2.1 O que são?

AGV´s, sigla pela qual são conhecidos os Automated Guided Vehicles ou Veículos Guiados Automaticamente representam os sistemas de transporte autónomo usados principalmente em ambientes industriais e logísticos. A sua função assenta na responsabilidade de mover materiais e produtos dentro de fábricas, armazéns ou centros de distribuição sem a intervenção direta do Homem.

2.2 História

A história dos AGV´s remonta ao início dos anos 50 do século passado, quando em 1953 foi usado, pela primeira vez, num armazém de mercearias nos EUA um veículo guiado automaticamente. Providos de para-choques mecânicos de segurança, eram guiados por uma guia (passo a redundância) no chão ou era utilizado um sensor ótico. Munidos de vantagens como a redução dos custos ou o aumento da eficiência, os AGV´s foram alvo de uma procura imensa, atendendo que estávamos nos anos após a 2ª Grande Guerra e a Indústria estava a voltar ao ativo, impulsionando o desenvolvimento de tecnologias mais avançadas.

Mais tarde, no final dos anos 80, ocorreu uma grande recessão que atingiu uma parte substancial do mundo da Indústria, na medida em que se interrompeu o desenvolvimento dos AGV´s pois os mesmos eram bastante dispendiosos e tornou-se necessário reduzir custos. Outro fator preponderante foi a introdução da designada Lean Production (produção eficiente) na indústria automóvel japonesa, que permitia reduzir os custos operacionais e aumentar a qualidade.

No fim da década de 90 observou-se um progresso nas formas de navegação com a introdução da navegação por laser, tornando os AGV´s mais flexíveis e de confiança. Assim se têm mantido até aos dias de hoje, não obstante ao início se tratar de sistemas dispendiosos, rígidos ou pouco confiáveis e de difícil instalação e manutenção. Atualmente temos os AGV´s como uma evidência de fiabilidade e eficiência, sendo estes sistemas utilizados numa pluralidade de indústrias.

Porém, um sistema AGV representa um investimento significativo para uma empresa. Desta forma, estamos perante um sistema que é de uso quase exclusivo para grandes empresas.

2.3 A sua aplicação

Quanto ao âmbito de aplicação dos AGV´s, estes são usados em uma variedade de indústrias, como são exemplo:

- **Automóvel**, nomeadamente para mover peças e componentes entre as linhas de produção como o reboque de peças do armazém até à linha de produção. Também importa a função de auxiliar de montagem, isto aquando do AGV carregar o produto por entre as diversas estações ao longo do processo, onde os operadores vão acrescendo valor a este produto. Tudo se resume à capacidade de transferência entre as várias áreas do processo produtivo;

- **Logística**, no que concerne à aplicação dos AGV´s em armazéns e centros de distribuição. Isto passa pela tarefa de organização de armazéns, mais no que respeita ao transporte de matérias-primas ou de um produto acabado entre diferentes setores, aos processos de industrialização, aos pontos de expedição, entre outros. Uma responsabilidade dos AGV´s neste tipo de indústria será ainda o transporte de cargas específicas, cujo transporte não se consegue realizar através de equipamentos comuns como as empilhadoras pois estamos diante de produtos de maior peso ou volume ou então requerem maior cuidado pela sua fragilidade;

- **Farmacêutica**, em especial no transporte de componentes e medicamentos entre as diferentes secções de produção na fábrica ou mesmo na cadeia de distribuição de medicamentos, isto em relação à sua eti-

quetação, o que se revela fulcral no controlo de qualidade;

- **Alimentar**, onde os AGV´s têm como missão principal o transporte das matérias-primas até às máquinas onde são processadas e posteriormente enquanto produto final até aos pontos de expedição.

2.4 Vantagens do uso de AGV´s

De uma forma sucinta, podemos assumir que para além dos hospitais, são os armazéns e as fábricas os locais onde os AGV´s têm assumido um papel preponderante onde têm vindo a ser retirados os mais variados benefícios. Assim, podemos separá-los em seis esferas:

- **Aumento da segurança**, com o objetivo de evitar acidentes que possam ocorrer com operadores humanos quando usam veículos manuais. Um exemplo será o de garantir a segurança dos trabalhadores na aproximação destes veículos, isto é, pela circulação destes AGV´s nos caminhos previstos e designados previamente pelos utilizadores;

- **Redução dos custos operacionais**, isto aquando da possibilidade de se observar de forma contínua um trabalho 24/7 (24 horas por dia, 7 dias por semana). A vantagem está em que não há necessidade de pausas para refeições, baixas por doença ou a liberdade de gozar férias, o que permite um aumento da produtividade;

- **Redução dos custos com a mão-de-obra**, no ponto de vista em que não será necessária tanta força de trabalho humana, permitindo reduzir os gastos com salários, por exemplo, enquanto se otimiza o fluxo de materiais;

- **Controlo do inventário**, dado que os veículos são permanentemente monitorizados e controlados por um sistema informático que armazena todo a informação do inventário;

- **Diminuição dos danos nos produtos e nas instalações**, uma vez que pelos sensores e caminhos desenhados pelos utilizadores vão impedir os AGV´s de colidirem contra outros equipamentos ou estruturas, eliminando erros humanos no transporte de materiais;

- **Flexibilidade**, pois estes veículos adaptam-se a diferentes layouts e ambientes de trabalho, sendo de fácil instalação, fácil modificação e têm a possibilidade de ser integrados com outros equipamentos.

Assim, podemos inferir que os AGV´s contribuem para o aumento da produtividade, permitem uma maior eficiência do tempo e ainda possibilitam a redução de custos, pelo que se depreende que fornecem uma enorme vantagem competitiva às empresas.

2.5 Desafios

Embora os AGV´s ofereçam muitos benefícios, também enfrentam alguns desafios que precisam de ser considerados antes da sua implementação. São exemplo:

- **Custo inicial**, onde no processo de aquisição e instalação o investimento inicial em AGV´s pode ser elevado, em especial para pequenas empresas. Isto inclui o custo de compra do veículo, a instalação da infraestrutura de navegação (como fitas magnéticas, sensores, etc.) e a integração dos sistemas de software, onde por vezes é necessário personalizar os veículos para tarefas bastante específicas;

- **Manutenção**, na medida em que os AGV´s exigem uma manutenção regular e por vezes preventiva, dependendo de especialistas para a sua reparação ou atualizações, onde qualquer problema técnico pode atrasar os processos produtivos;

- **Necessidade de integração**, visto que a implementação de AGV´s implica à priori a compatibilidade dos softwares com outros sistemas, onde se não existir uma incorporação de ambos pode resultar em problemas de sincronização e gestão de operações. Não esquecer também a dependência dos AGV´s de uma estrutura tecnológica sem falhas de rede, onde estas podem fazer com que haja interrupções nas operações;

- **Interação humana**, em razão de em muitas situações estes veículos atuarem melhor num ambiente com o mínimo de interação humana, isto com vista a evitar acidentes ou interrupções. Todavia, quando

perante situações complexas, o discurso pode ser diferente. É assim, pois perante uma imprevisibilidade, quando comparamos os AGV´s a um operador humano, estes veículos tendem a ser menos ágeis ou mesmo menos eficientes, enquanto o Homem tem a capacidade de ajustar rapidamente a rota ou executar em espaços mais apertados, por exemplo.

Posto isto, e ainda que se reconheça algumas limitações aos AGV´s como o custo, a flexibilidade ou a adaptação a ambientes complexos, tudo fatores que necessitam de uma avaliação antes de uma futura implementação, estes veículos são uma excelente solução para muitos ambientes industriais e logísticos. É a análise cuidadosa destes aspetos que permite às empresas tomar decisões mais ou menos equilibradas aquando da adoção deste mecanismo.

Chapter 3

Visão Artificial

3.1 Introdução

A visão por computador é um domínio da inteligência artificial que utiliza a aprendizagem automática e as redes neuronais para ensinar computadores e sistemas a obter informações significativas a partir de imagens digitais, vídeos e outras entradas visuais e a fazer recomendações ou tomar medidas quando detetam defeitos ou problemas. Se a Inteligência Artificial permite aos computadores pensar, a visão por computador permite-lhes ver, observar e compreender. A visão computacional funciona de forma muito semelhante à visão humana, exceto que os humanos têm uma vantagem inicial. A visão humana tem a vantagem de dispor de vidas de contexto para treinar a distinção entre objetos, a distância a que se encontram, se estão em movimento ou se há algo de errado com uma imagem. A visão por computador treina as máquinas para desempenharem estas funções, mas tem de o fazer em muito menos tempo com câmaras, dados e algoritmos em vez de retinas, ner-

vos óticos e um córtex visual. Como um sistema treinado para inspecionar produtos ou observar um ativo de produção pode analisar milhares de produtos ou processos por minuto, detetando defeitos ou problemas impercetíveis, pode rapidamente ultrapassar as capacidades humanas.

3.2 História da visão artificial

Na década de 1960, os investigadores começaram a desenvolver algoritmos para processar e analisar dados visuais, mas a tecnologia era limitada pelo poder computacional. Na década de 1970, os investigadores tinham desenvolvido algoritmos mais sofisticados para o processamento de imagens e reconhecimento de padrões. Um dos principais avanços foi o desenvolvimento da transformada de Hough, que permitiu a deteção de linhas e outras formas geométricas nas imagens. Nas décadas de 1980 e 1990, os investigadores concentraram-se no desenvolvimento de algoritmos de aprendizagem automática para a visão computacional. Estes algoritmos permitiram aos computadores aprender com os dados e melhorar a precisão ao longo do tempo. O algoritmo de deteção de rostos Viola-Jones, desenvolvido em 2001, foi um dos avanços mais significativos desta época[1]. Nas décadas de 2000 e 2010, os algoritmos de aprendizagem profunda revolucionaram a visão computacional, permitindo que os computadores aprendessem representações hierárquicas de dados visuais. O desenvolvimento das redes neurais convolucionais (CNN) e de outros algoritmos de aprendizagem profunda permitiu que os computadores reconhecessem objetos, seguissem movimentos e realizassem outras tarefas complexas com maior precisão do que nunca. Atualmente, a visão computacional é um campo em rápido crescimento com aplicações em veículos autónomos, imagiologia médica e robótica, entre muitas outras áreas. Com os avanços da inteligência artificial, espera-se que a visão por computador continue a crescer e a transformar a forma como interagimos com os dados visuais.

3.3 Como funciona a visão artificial

A visão por computador permite que os computadores compreendam e interpretem a informação visual do mundo, tal como o sistema visual humano. Envolve a utilização de algoritmos e modelos para processar e analisar imagens e vídeos digitais.

Aquisição de imagens: O processo começa com a aquisição de imagens ou vídeos digitais. Estas imagens podem ser obtidas de várias fontes, como câmaras, drones, satélites ou arquivos digitais.

Pré-processamento: Uma vez adquiridas as imagens, estas são frequentemente submetidas a um pré-processamento para melhorar a sua qualidade e prepará-las para análise. As etapas comuns de pré-processamento incluem:

- Redução de ruído: Remoção ou redução de ruído indesejado na imagem.
- Melhoria da imagem: Ajustar o brilho, o contraste e a nitidez.
- Normalização: Garantir condições de iluminação consistentes.
- Redimensionamento e corte: Tornar as imagens consistentes em termos de tamanho e foco nas regiões de interesse.

Extração de caraterísticas: Os algoritmos de visão por computador identificam e extraem características significativas das imagens. Estas características podem ser arestas, cantos, texturas, cores, formas ou padrões mais complexos. A extração de características é crucial para compreender o conteúdo de uma imagem.

Representação de caraterísticas: As características extraídas são transformadas em um formato adequado para processamento posterior. Esta etapa envolve a criação de vetores de características ou outras representações que codificam a informação relevante da imagem.

Aprendizagem automática e aprendizagem profunda: Muitas tarefas de visão computacional envolvem modelos de aprendizagem automática e de aprendizagem profunda. Esses modelos são treinados em conjuntos de dados rotulados para aprender padrões e relacionamentos nos dados. Os tipos comuns de modelos incluem:

- Redes Neuronais Convolucionais (CNNs): Altamente eficazes na classificação de imagens, deteção de objectos e segmentação.
- Redes neurais recorrentes (RNNs): Usadas para tarefas que envolvem dados sequenciais em vídeos.
- Transformadores: Aplicados a tarefas que requerem mecanismos de atenção, como a legendagem de imagens.

Processamento específico da tarefa:

- Deteção de objectos: Envolve a identificação e localização de objectos numa imagem ou vídeo. Utiliza frequentemente técnicas de proposta de regiões e modelos baseados em CNN.
- Classificação de imagens: Atribui um rótulo ou categoria a uma imagem com base no seu conteúdo. As CNNs são normalmente utilizadas para esta tarefa.

- Segmentação de imagens: Divide uma imagem em regiões ou segmentos significativos. A segmentação semântica atribui uma etiqueta de classe a cada pixel, enquanto a segmentação de instâncias distingue instâncias de objectos individuais.

- Reconhecimento de rostos: Identifica e verifica indivíduos com base em caraterísticas faciais utilizando técnicas como redes neurais profundas.

- Seguimento de movimentos: Envolve o seguimento do movimento de objectos ou caraterísticas através de fotogramas numa sequência de vídeo.

Pós-processamento: Depois de gerar resultados, podem ser aplicados passos de pós-processamento para refinar ou filtrar o resultado. Isto pode incluir a remoção de regiões pequenas e irrelevantes, a suavização de limites ou a aplicação de restrições adicionais.

Visualização e Interpretação: O passo final envolve a visualização e interpretação dos resultados. Isto pode incluir a sobreposição de objectos detectados na imagem original, a criação de mapas de calor para destacar áreas de interesse ou o fornecimento de descrições textuais do conteúdo.

Circuito de feedback: E finalmente no caso deste projeto, os sistemas de visão computacional podem fornecer feedback a sistemas externos ou mecanismos de controlo. Por exemplo, em veículos autónomos, os sistemas de visão computacional podem ajudar a tomar decisões sobre a direção, a travagem e a aceleração. Os algoritmos e técnicas específicos utilizados podem variar consoante a tarefa e a complexidade dos dados visuais que estão a ser analisados.

3.4 Diferentes aplicações

A visão computacional tem uma vasta gama de aplicações em vários sectores, tirando partido da sua capacidade de interpretar e compreender dados visuais. Vamos discutir algumas aplicações específicas do sector da visão por computador:

Cuidados de saúde:

- Diagnóstico por imagem: Análise automatizada de raios X, tomografias computadorizadas e ressonâncias magnéticas para detetar e diagnosticar condições como fraturas, tumores e distúrbios neurológicos.
- Patologia: Assistência a patologistas na análise de amostras de tecido para deteção e classificação de cancro.
- Dermatologia: Identificação de problemas de pele e sinais para deteção precoce de cancro da pele.
- Assistência cirúrgica: A visão por computador ajuda em cirurgias minimamente invasivas, fornecendo aos cirurgiões informações visuais em tempo real, aumentando a precisão e reduzindo os riscos.
- Monitorização remota de doentes: Monitorização e análise de sinais vitais, movimento e comportamento de pacientes nas suas casas ou hospitais, utilizando câmaras e sensores.

Automóvel:

- Veículos autónomos: A visão computacional desempenha um papel crucial nos veículos autónomos, ajudando em tarefas como a deteção

de objectos, a deteção de faixas e o seguimento de peões para uma navegação segura.

- Sistemas avançados de assistência ao condutor (ADAS): As funcionalidades dos ADAS, como o controlo de cruzeiro adaptativo, a assistência à manutenção na faixa de rodagem e os sistemas de prevenção de colisões, dependem da visão por computador para a análise de dados em tempo real.

- Reconhecimento de placas de veículos (LPR): Os sistemas automatizados de reconhecimento de matrículas são utilizados para gestão de estacionamento, cobrança de portagens e aplicação da lei.

Retalho e comércio eletrónico:

- Pesquisa visual: Permite que os clientes pesquisem produtos utilizando imagens em vez de consultas de texto, melhorando a experiência de compra.

- Gestão de inventário: Acompanhamento e gestão automatizados do stock, reduzindo os casos de falta de stock e otimizando os processos da cadeia de fornecimento.

- Lojas sem caixas: A visão por computador é utilizada para seguir os artigos selecionados pelos clientes e cobrá-los automaticamente, eliminando a necessidade das tradicionais filas de caixa.

Fabrico:

- Controlo de qualidade: Deteção de defeitos nos processos de fabrico, como a identificação de falhas em produtos nas linhas de montagem.

- Automação robótica: Orientação de robôs em tarefas como recolha e colocação de itens, montagem e inspeção de qualidade.

- Manutenção preditiva: Utilização da visão por computador para monitorizar máquinas e prever quando é necessária manutenção, reduzindo o tempo de inatividade e os custos de manutenção.

Agricultura:

- Agricultura de precisão: A visão computorizada e os drones são utilizados para analisar a saúde das culturas, detetar pragas e doenças e otimizar a irrigação e a utilização de fertilizantes, conduzindo a um aumento do rendimento das culturas e à redução do impacto ambiental.

- Monitorização do gado: Acompanhamento da saúde e do comportamento do gado para deteção precoce de doenças e gestão eficiente.

- Controlo de ervas daninhas e pragas: Identificar e selecionar ervas daninhas e pragas, reduzindo a necessidade de intervenções químicas.

Estes são apenas alguns exemplos de como a visão computacional está a transformar várias indústrias. A capacidade de analisar e compreender dados visuais está a conduzir a uma maior eficiência, segurança e inovação em todos os sectores.

Apesar destas todas utilidades a visão artificial também enfrenta alguns desafios tais como:

- **Qualidade e quantidade dos dados:** Muitos algoritmos de visão computacional, especialmente os modelos de aprendizagem profunda,

requerem conjuntos de dados grandes e diversificados para treino. A obtenção desses conjuntos de dados pode ser um desafio para domínios específicos ou eventos raros.

- **Processamento em tempo real:** Algumas aplicações, como os veículos autónomos e a robótica, exigem um processamento em tempo real, o que torna essencial o desenvolvimento de algoritmos eficientes e de hardware capaz de suportar a carga computacional.

- **Privacidade e segurança:** Em termos de privacidade os sistemas de visão computacional podem inadvertidamente invadir a privacidade dos indivíduos, por exemplo, através de vigilância não autorizada ou reconhecimento facial sem consentimento. E em termos de segurança, a proteção dos sistemas de visão por computador contra ataques, violações de dados ou acesso não autorizado é crucial, especialmente em aplicações críticas como veículos autónomos e sistemas de segurança.[1]

3.5 OpenCV

Para o projeto atual a escolha tanto de software como de hardware foi motivada por dois parâmetros iniciais, relação qualidade preço e capacidade de resposta rápida. Depois de uma pesquisa de mercado em relação às hipóteses a considerar em termos de software, uma realçou-se entre as demais, por ser open source, grátis de utilizar e ter uma interface em python, uma linguagem de programação familiar. Mais concretamente OpenCV (Open Source Computer Vision Library) foi feita para proporcionar a todos uma infraestrutura de desenvolvimento de visão artificial comum e acessível. É uma biblioteca com mais de 2500 algoritmos otimizados que incluem tanto algoritmos mais antigos como algoritmos ponta de gama. Estes variam em utilização desde detetar e reconhecer caras, identificar objetos, classificar ações de humanos em vídeos, seguir movimentos tanto de humanos como de objetos, entre outras possibilidades. OpenCV tem uma comunidade de desenvolvedores com mais de quarenta e sete mil pessoas, e com mais de dezoito milhões de downloads por todo o mundo, e é imensamente usada em empresas, grupos de pesquisa e governos[2].

3.6 Análise de imagem

3.6.1 Introdução

Um software de análise de imagem tem características bastante diferentes dos já conhecidos softwares de manipulação de imagem como o Adobe Photoshop ou Gimp. Ambos os softwares têm a capacidade de ler imagens numa variedade de formatos e aplicar uma manipulação em termos de luminosidade, contraste , cores e suavidade de imagem. No entanto, enquanto um software de manipulação de imagem tem como objetivo melhorar a aparência de uma imagem, um software de análise de imagem tem como objetivo retirar e quantificar características específicas da imagem.

A análise mais comum é em relação a aspectos dimensionais, tais como comprimento, área e perímetro. Mas também pode ser em termos quantitativos, tais como o número de uma quantidade de objetos numa imagem. Para ambas estas abordagens, o normal é o selecionamento prévio de uma área de interesse à qual a operação de análise irá ser feita. Esta área de interesse pode ser calculada automaticamente, ou de uma forma manual a inserir uma região de pixeis pré-definida.

Ao nível mais baixo os computadores operam com 1's e 0's, e isto não é diferente quando trabalham com imagens. Os computadores conseguem representar uma imagem como uma matriz de 1's e 0's, quando um computador mostra uma imagem, está a mostrar uma grelha de pixeis, cada um preenchido com uma cor. Ao ponto de exemplificar vamos considerar uma forma simples tal como este triângulo preto abaixo: Esta imagem é simples-

mente uma matriz de pixeis armazenados no computador :

0	0	0	1	0	0	0
0	0	1	1	1	0	0
0	1	1	1	1	1	0
1	1	1	1	1	1	1

Figure 3.1: Matriz Pixeis Apenas com 1´s e 0's

Isto é bastante simples para imagem com apenas duas cores, preto e branco. Mas se a imagem tiver por exemplo tons de cinzento é necessário outro nível de complexidade para representar o nível de luminosidade de cada pixel no ecrã. Existem demasiadas cores para atribuir cada uma a um número que seja intuitivo, então foram desenvolvidos sistemas para dividir os componentes individuais de cada cor e atribuir um número a cada componente. Temos por exemplo o modelo mais conhecido, o RGB que divide cada cor em percentagem de "red"(vermelho), "green"(verde) e "blue"(azul) respetivamente, que combinados criam a cor desejada. Neste sistema cada cor é chamada de canal, e para uma imagem de oito bits cada canal tem um valor entre 0 e 255 (o que é o valor máximo para um número binário de 8 bits).

Para o exemplo do triângulo na imagem anterior os valores de RGB para esta específica tonalidade de verde são 154, 205 e 50 respetivamente. Mas vale a pena notar que se este verde fosse apenas a cor verde o canal ver-

Figure 3.2: Triângulo verde

melho e azul seria zero(0,255,0). Nesta representação nota-se que temos bastante vermelho mas pouco azul misturado com o verde para criar esta tonalidade de verde. Cada pixel verde é representado por 145,205,50 e cada pixel branco é representado por 0,0,0. Isto significa que esta imagem de 28 pixeis é representada por três matrizes de 7x4. Podemos imaginar estas 3 matrizes combinadas para representar esta imagem.

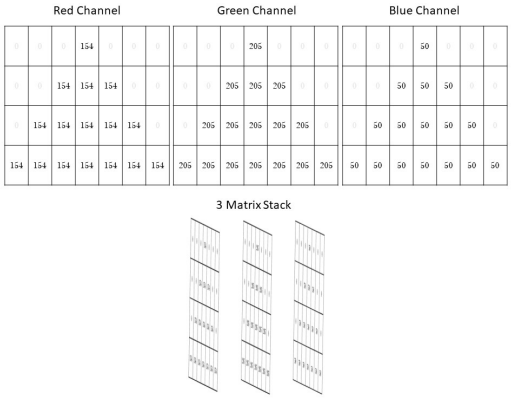


Figure 3.3: Matrizes diferentes cores [2]

Com esta representação conseguimos ter uma ideia de quantos números e matrizes que será preciso para representar uma imagem de alta qualidade, por exemplo uma imagem com a resolução 4K (que contêm 4000 pixeis). É de notar que o sistema RGB não é o único a atribuir cores a imagens. Temos por exemplo também o sistema CYMK que usa quatro números para representar cada cor. Este sistema usa quatro cores "cyan"(ciano) , magenta, "yellow"(amarelo) e "black"(preto). Este sistema é benéfico em situações em que precisamos de usar um sistema de cor subtrativa (onde ao adicionar pigmentos de cor vamos escurecendo a imagem) por exemplo em impressões. Mas no âmbito deste projeto iremos considerar apenas o sistema RGB.

3.6.2 Isolar objetos a partir de thresholding

Thresholding é o processo de segmentação de uma imagem com base nos valores numéricos atribuídos a cada pixel onde na maior parte das imagens temos uma diferença de luminosidade entre o nosso objeto que queremos estudar e o resto da imagem. Thresholding é o processo que nos permite identificar o perímetro do objeto em estudo ao definir um valor de luminosidade limite que nos permite separar pixeis claros e pixeis escuros.

3.6.3 cv2.ctvColor()

Este método é usado para converter uma imagem de um espaço de cor para outro. Uma utilização comum é a conversão de uma imagem de RGB (Red, Green, Blue) para tons de cinzento que fornece uma melhor análise de fronteiras.

3.6.4 cv2.threshold

Neste método para cada pixel, é aplicado o mesmo valor de limiar. Se o valor do pixel for menor que o limiar, é definido como 0, caso contrário é definido como um valor máximo. A função cv.threshold é utilizada para aplicar a limiarização. O primeiro argumento é a imagem de origem, que deve ser uma imagem em tons de cinzento. O segundo argumento é o valor de limiar que é utilizado para classificar os valores de pixel. O terceiro argumento é o valor máximo que é atribuído aos valores de pixéis que excedem o limiar. O OpenCV fornece diferentes tipos de limiarização que são dados pelo quarto parâmetro da função. E neste projeto foi usado o parâmetro [**cv.THRESH˙BINARY˙INV**]. Este tipo de limiarização é representada graficamente como sendo o terceiro gráfico(sendo este apenas um inverso do segundo), em comparação ao primeiro que apresenta o valor do pixel e o valor definido de threshold como mostrado na figura 3.4.

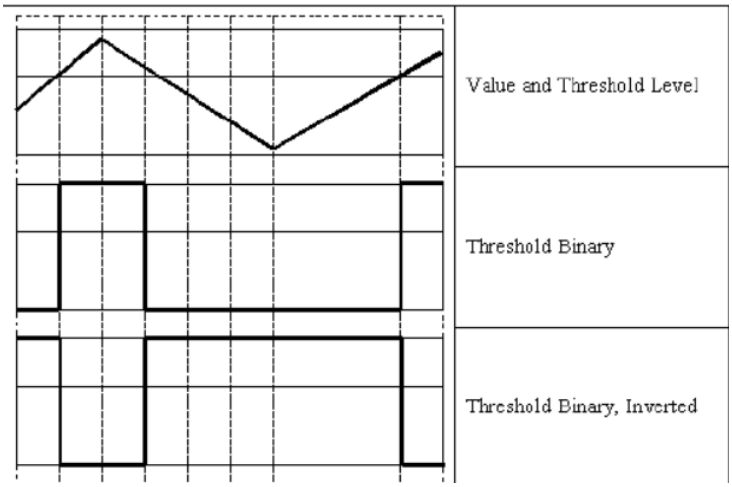


Figure 3.4: Gráfico Threshold [2]

Os valores finais calculados são de acordo com a fórmula na figura 3.5.

$$\texttt{dst}(x,y) = \begin{cases} 0 & \text{if } \texttt{src}(x,y) > \texttt{thresh} \\ \texttt{maxval} & \text{otherwise} \end{cases}$$

Figure 3.5: Formula threshold[2]

3.6.5 cv2.erode e cv2.dilate

O método cv2.erode() é utilizado para efetuar a erosão na imagem. A ideia básica da erosão é exatamente como a erosão do solo, só que corrói os limites do objeto em primeiro plano (tenta sempre manter o primeiro plano a branco). Normalmente, é efetuada em imagens binárias. Necessita de duas entradas: uma é a nossa imagem original e a segunda é designada por elemento estruturante ou kernel, que decide a natureza da operação. Um pixel na imagem original (1 ou 0) só será considerado 1 se todos os pixeis sob o núcleo forem 1, caso contrário é apagado (transformado em zero). No nosso caso o kernel é cinco, ou seja, a operação foi efetuada a partir de um kernel com uma matriz de 5x5 para uma erosão mais eficaz. E posteriormente é aplicado o método cv2.dilate que consiste em aplicar um kernel "B" à imagem inicial, este kernel "B" tem um ponto de ancoragem definido, sendo normalmente o centro do kernel. À medida que o kernel "B" e substituímos o pixel da imagem na posição do ponto de ancoragem por esse valor máximo. Como pode deduzir, esta operação de maximização faz com que as regiões brilhantes de uma imagem "cresçam" (daí o nome dilatação)[2].

3.6.6 cv2.Canny

Um dos métodos mais importantes neste projeto é o método de deteção de linhas (fronteiras). O método Canny foi concebido para ser um detetor de bordos ótimo (de acordo com critérios específicos, existem outros detetores que também afirmam ser ótimos em relação a critérios ligeiramente diferentes). Toma como entrada uma imagem em escala de cinzentos ou a preto e branco e produz como saída uma imagem que mostra as posições das descontinuidades de intensidade detetadas. O filtro Canny é um detetor de linhas (fronteiras) em várias fases. Utiliza um filtro baseado na derivada de uma Gaussiana para calcular a intensidade dos gradientes, reduzindo o efeito do ruído presente na imagem. Depois, as potenciais arestas são reduzidas a curvas de apenas um pixel de largura, removendo os pixeis não máximos da magnitude do gradiente. Por fim, os pixeis de borda são mantidos ou removidos utilizando o limiar de histerese na magnitude do gradiente. O Canny tem três parâmetros ajustáveis: a largura da Gaussiana (quanto mais ruidosa for a imagem, maior será a largura) e o limiar baixo e alto para a limiarização por histerese [3].

3.6.7 cv2.houghlinesP

Este método é utilizado para detetar linhas retas numa imagem e preferencialmente é aplicado antes um método de deteção de fronteiras(no nosso caso o método Canny). Como sabemos, uma linha no espaço de imagem pode ser expressa com duas variáveis. Por exemplo:

- No sistema de coordenadas cartesianas: Parâmetros: (m,b)

- No sistema de coordenadas polares: Parâmetros: (r, θ)

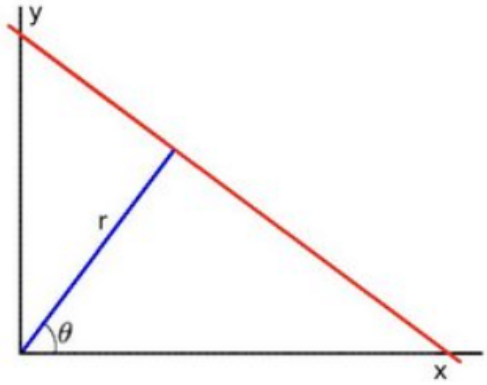


Figure 3.6: Coordenadas Polares [4]

Para as Transformadas de Hough, expressaremos as linhas no sistema polar. Assim, a equação de uma reta pode ser escrita como:

$$y = \left(-\frac{\cos\theta}{\sin\theta}\right) x + \left(\frac{r}{\sin\theta}\right)$$

Simplificando a equação, ficamos com: $r = x\cos\theta + y\sin\theta$.

Em geral, para cada ponto (x_0, y_0), podemos definir a família de retas que passa por esse ponto como:

$$r_\theta = x_0 \cdot \cos\theta + y_0 \cdot \sin\theta$$

Ou seja, cada par (r_θ, θ) representa cada reta que passa por (x_0, y_0).

Se para um dado ponto (x_0, y_0) traçarmos a família de retas que o atravessa, obtemos uma sinusoide. Por exemplo, para $x_0 = 8$ e $y_0 = 6$ obtemos o seguinte gráfico inserido na figura 3.7 (num plano $\theta - r$).

Consideramos apenas pontos em que $r > 0$ e $0 < \theta < 2\pi$. Podemos fazer a mesma operação acima para todos os pontos de uma imagem. Se as

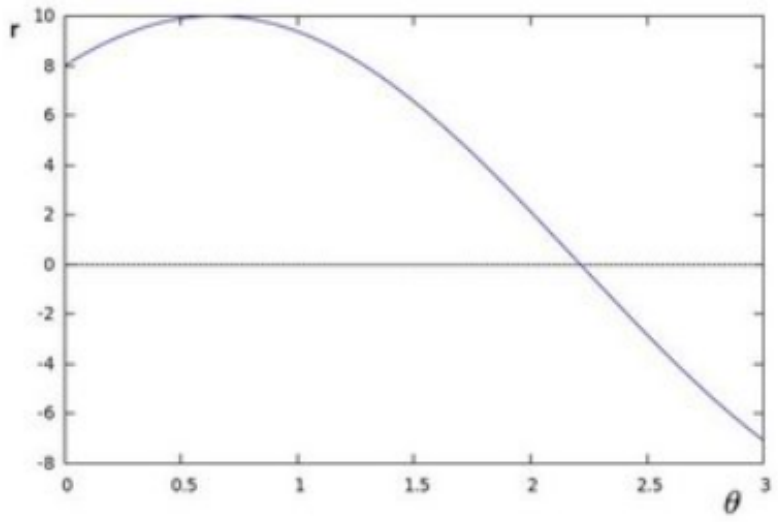


Figure 3.7: Gráfico Valores $x_0 = 8$ e $y_0 = 6$ [4]

curvas de dois pontos diferentes se intersectarem no plano $\theta - r$, isso significa que ambos os pontos pertencem a uma mesma reta. Por exemplo, seguindo o exemplo anterior e desenhando o gráfico para mais dois pontos: $x_1 = 4$, $y_1 = 9$ e $x_2 = 12$, $y_2 = 3$, obtemos o gráfico representado na figura 3.8:

Os três gráficos intersectam-se num único ponto $(0,925; 9,6)$, estas coordenadas são os parâmetros (θ, r) ou a linha em que (x_0, y_0), (x_1, y_1) e (x_2, y_2) se encontram.

Em resumo, uma reta pode ser detetada encontrando o número de intersecções entre curvas. Quanto mais curvas se intersectarem, mais pontos terá a reta representada por essa intersecção. Em geral, podemos definir um limiar do número mínimo de intersecções necessárias para detetar uma linha. É isto que a Transformada de Linha de Hough faz. Ela regista a intersecção entre as curvas de cada ponto da imagem. Se o número de in-

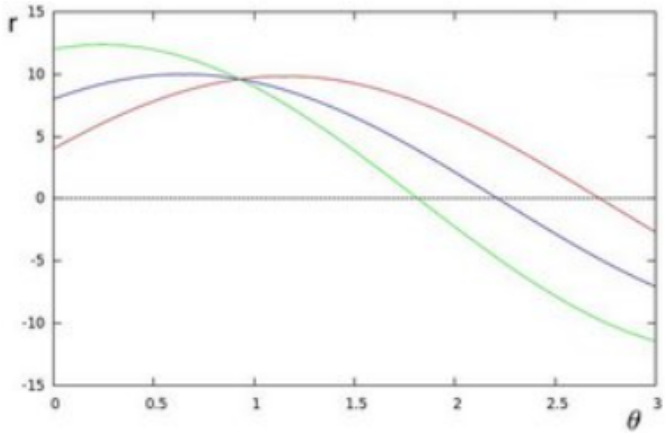


Figure 3.8: Gráfico para os valores $x_1 = 4$, $y_1 = 9$ e $x_2 = 12$, $y_2 = 3$[4]

tersecções for superior a um determinado limiar, declara-o como uma linha com os parâmetros (θ, r_θ) do ponto de intersecção[4].

Dentro da biblioteca temos acesso a dois tipos de transformadas de Hough:

- **A transformada de Hough padrão:** Consiste em praticamente tudo o que acabámos de explicar na secção anterior e dá como resultado um vetor de pares (θ, r_θ)
- **A transformada probabilística de Hough:** Uma implementação mais eficiente da Transformada de Hough e dá como saída os extremos das linhas detetadas (x_0, y_0, x_1, y_1)

Chapter 4

Desenvolvimento de Hardware

4.1 Introdução

Neste capítulo serão discutidos todos os pontos relativamente às partes físicas do projeto, desde os componentes escolhidos, a necessidade de projetar novos componentes e também a tecnologia de manufatura que foi utilizada.

Sendo que no âmbito deste projeto foram escolhidos os seguintes componentes:

1. Raspberry Pi 3 Modelo B
2. Ponte H
3. Câmara raspberry pi
4. Motores DC
5. Bateria Portátil

4.2 Conceções sistema

O sistema do veículo está ligado de acordo com a figura 4.1.

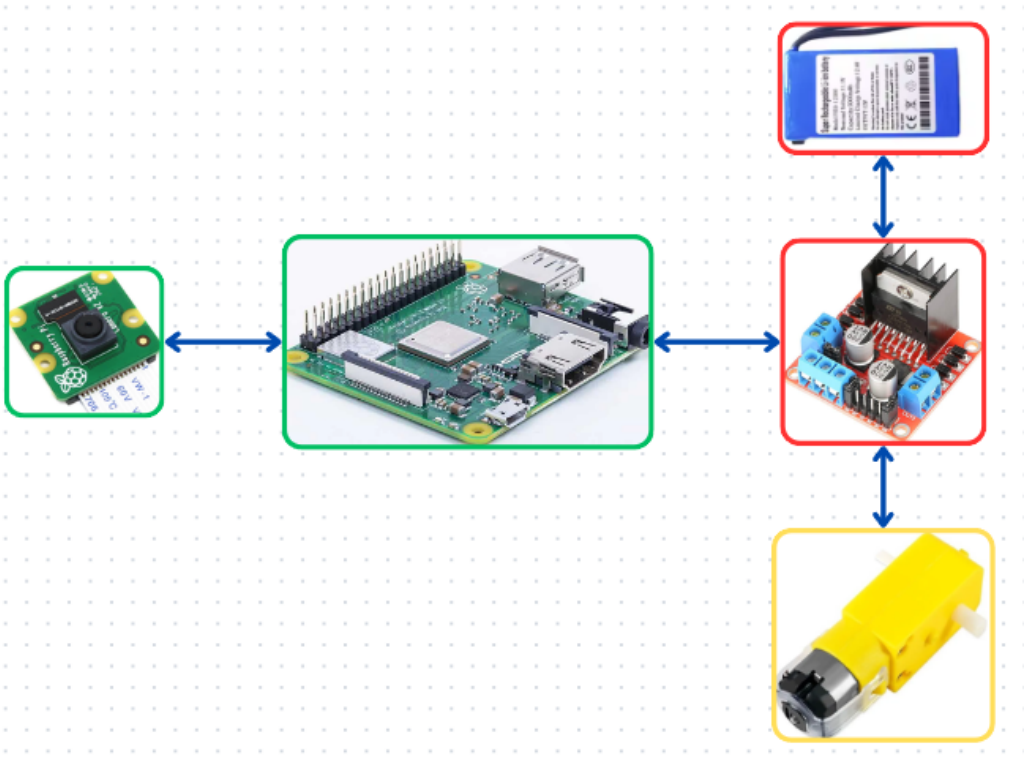


Figure 4.1: Diagrama Ligações sistema

Neste sistema temos a comunicação entre o raspberry pi e a câmera através de um cabo de fita, tornando possível a aquisição dos frames para análise. Existe também uma ligação entre o raspberry pi e a ponte H enviando sinais para o controlo dos motores, mas também recebe alimentação da mesma. Finalmente a ponte H é alimentada por uma bateria de 12 V recarregável, estando também ligada aos dois motores, controlando assim

ambos.

4.3 Raspberry Pi 3

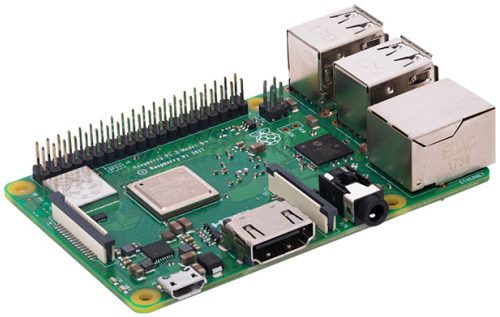


Figure 4.2: Raspberry Pi 3

4.3.1 Definição

O Raspberry Pi 3 é um pequeno computador de placa única desenvolvido pela Fundação Raspberry Pi. É um computador do tamanho de um cartão de crédito que pode ser ligado a um monitor ou a uma televisão, ou até como no caso deste projeto transmitir o seu interface para um computador ligado na mesma rede sendo assim possivel funcionar completamente apenas com ligação direta a uma fonte de tensão tornando possível a utilização para vários projetos e aplicações. O modelo utilizado neste projeto foi o Raspberry Pi 3 Modelo B com as seguintes especificações:

- Processador: 1.2 GHz quad-core ARM Cortex-A53

- RAM: 1GB
- Conectividade: Wi-Fi, Bluetooth, Ethernet
- Sistema operativo: Pode executar vários sistemas operativos, incluindo diferentes versões do Linux e até do Windows 10 IoT Core.
- Portas: HDMI, USB, pinos GPIO, tomada de áudio, porta de câmara CSI, porta de visualização DSI

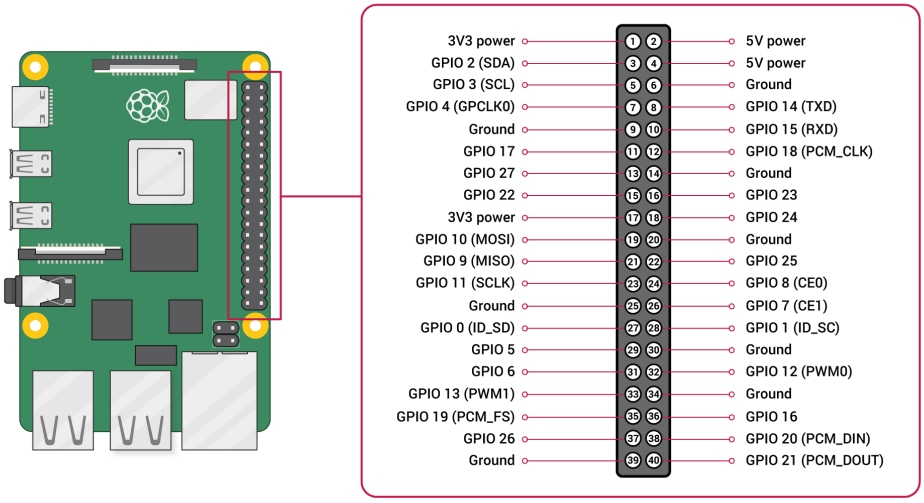


Figure 4.3: Portas GPIO

Hoje em dia o Raspberry Pi tem bastantes usos em diversas áreas como por exemplo na educação, onde é amplamente utilizado em escolas e instituições educacionais para ensinar programação e ciência da computação. Em projetos de bricolage em que muitos entusiastas e curiosos utilizam o Raspberry Pi 3 para vários projetos, como automação doméstica, centros mul-

timédia, consolas de jogos retro e muito mais. É também bastante popular na área de prototipagem de novas ideias de hardware e software devido ao seu baixo custo e versatilidade, e até pode ser usado como um pequeno servidor para alojar páginas na web, executar aplicações simples ou como um dispositivo de armazenamento ligado à rede. Um ponto também bastante apelativo para a escolha deste sistema foi a existência de uma comunidade grande e ativa de entusiastas, onde são partilhados projetos, tutoriais e resolução de problemas comuns.

Para este projeto foi inicialmente considerado usar o microcontrolador ESP32, um sistema com ainda menos dimensões físicas mas consequentemente menos capacidade de processamento, e depois de bastante investigação no estado da arte do sistema cheguei à conclusão que não seria possível usar o sistema ESP32 porque não tinha capacidades de capturar uma imagem e fazer internamente o processamento da mesma, apenas a capacidade de enviar a imagem para outro sistema para fazer o processamento exteriormente, o que ia contra o objetivo do projeto de criar um sistema independente.

Foi assim escolhido o sistema Raspberry Pi 3 pela facilidade e versatilidade de ligar a vários inputs e outputs de sensores mas também pela facilidade de usar a linguagem de programação aprendida anteriormente, Python 3, e o módulo de análise de imagem OpenCV.

4.4 Ponte H

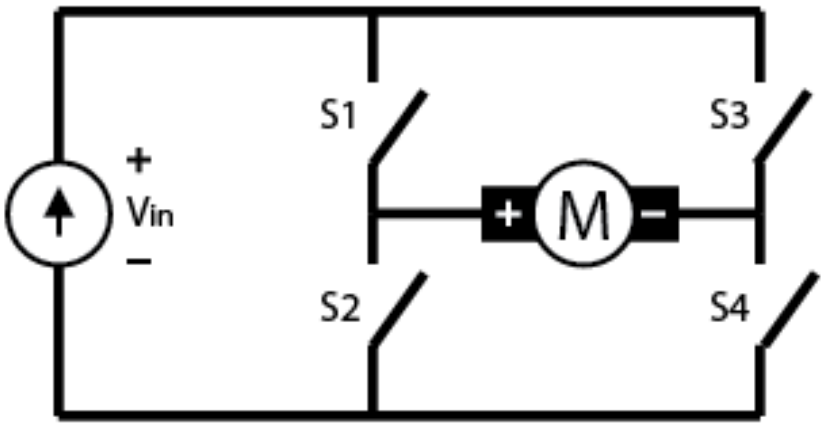


Figure 4.4: Cirtuito Ponte H

Uma ponte H é constituída por quatro interruptores que controlam o fluxo de corrente para uma carga. Na figura 4.3, a carga é o motor que liga os dois conjuntos de interruptores. Utilizando uma fonte de corrente, é possível conduzir a corrente em duas direções fechando dois interruptores. Se os interruptores 1 e 4 estiverem fechados, a corrente fluirá da esquerda para a direita como na figura 4.4: Se fechar o interruptor 1 e o interruptor 4, a corrente fluirá da fonte, através do interruptor 1, e depois através da carga, depois através do interruptor 4, e depois de novo para a fonte. Neste projeto usamos o controlador de motor L298N devido a ser uma forma económica de controlar dois motores DC. Pode controlar a velocidade e a direção de dois motores DC, incluindo steppers bipolares como os do tipo NEMA 17.

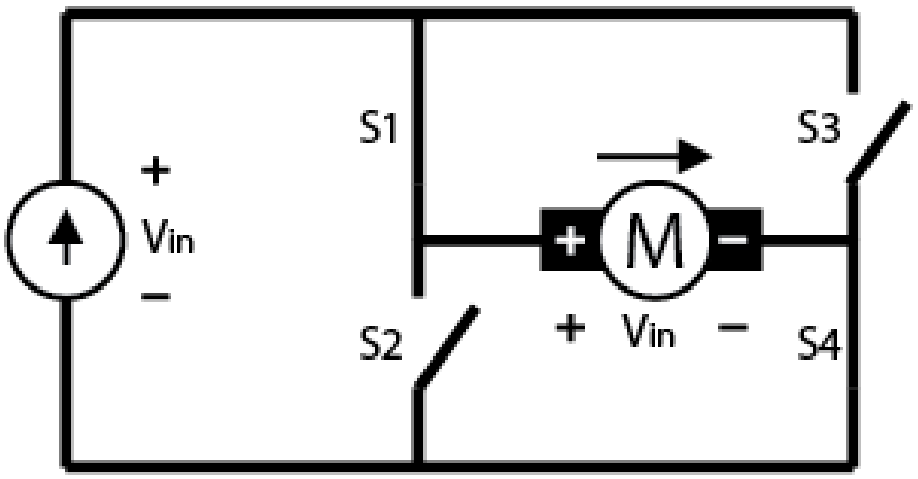


Figure 4.5: Ponte H configurada para ter o interrutor 1 e o interrutor 4 fechados.

O driver de motor L298N tem dois canais, canal A e canal B.O motor no canal A é ligado entre os pins OUT1 e OUT2. A velocidade do motor é controlada pelo sinal PWM no pino Enable A , enquanto a direção é controlada pelo nível dos pinos In1 e In2 . Para um desempenho ótimo deste tipo de motor, o controlador de motor L298N é idealmente emparelhado com uma fonte de alimentação de 5 V, mas neste caso pode tambem ser alimentado por 12 V

O módulo de acionamento do motor L298N contém dois blocos de terminais de parafuso, um para um motor e outro para um pino de terra. Além disso, o módulo tem um pino de 5 V que pode ser uma entrada ou uma saída.

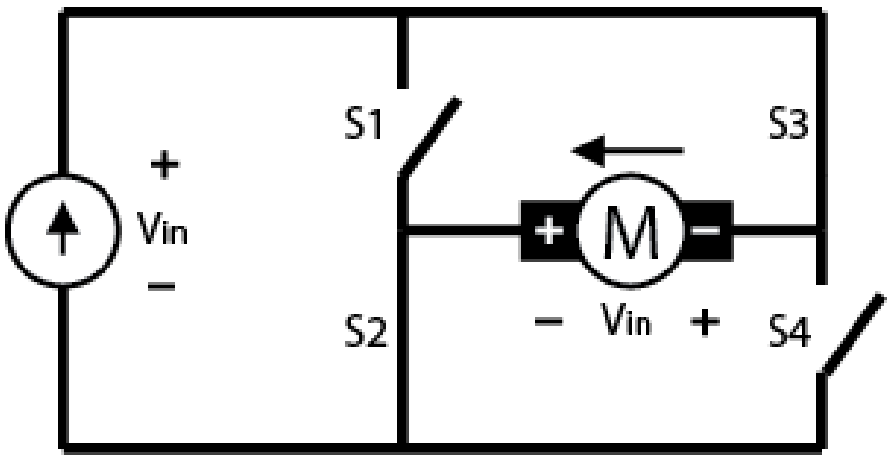


Figure 4.6: Um circuito de ponte H com S2 e S3 fechados.

O sinal PWM do motor do L298N controlará a velocidade do motor. Os engenheiros conceberam o circuito para funcionar sem problemas com vários componentes electrónicos e mecânicos, incluindo sensores, interruptores e detetores.

O driver de motor L298N é um design de ponte H dupla com baixo aquecimento e interferência. Este dispositivo suporta até 2 A de corrente e tem uma potência nominal de 25 W. Este módulo pode acionar dois motores DC, um motor de passo de 2 fases e um motor de passo de 4 fases.

4.5 Câmera Raspberry Pi

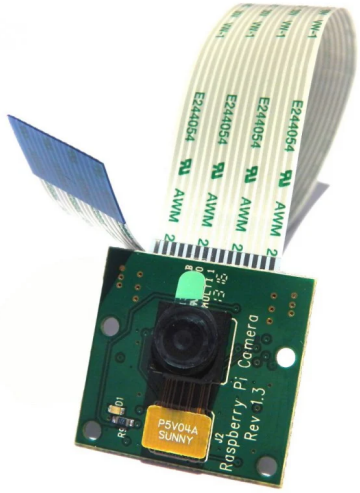


Figure 4.7: Camera Raspbery Pi

A placa de câmara Raspberry Pi liga-se diretamente ao conetor CSI do Raspberry Pi. É capaz de fornecer uma imagem de resolução cristalina de 5MP, ou gravação de vídeo HD 1080p a 30fps. Concebida e fabricada à medida pela Raspberry Pi Foundation no Reino Unido, a Raspberry Pi Camera Board possui um sensor Omnivision 5647 de 5MP (2592x1944 pixels) num módulo de focagem fixa. O módulo liga-se ao Raspberry Pi, através de um cabo de fita de 15 pinos, à interface MIPI Camera Serial Interface (CSI) dedicada de 15 pinos, que foi concebida especialmente para a interface com câmaras. O barramento CSI é capaz de taxas de dados extremamente elevadas e transporta exclusivamente dados de pixéis para o processador BCM2835. A placa tem as dimensões de 25 mm x 20 mm x 9 mm, e pesa pouco mais de 3 g, o que a torna perfeita para aplicações móveis ou

outras em que o tamanho e o peso são importantes. O sensor em si tem uma resolução nativa de 5 megapixéis e tem uma lente de focagem fixa integrada. Em termos de imagens fixas, a câmara tem capacidade para imagens estáticas de 2592 x 1944 pixéis e também suporta gravação de vídeo de 1080p a 30 fps, 720p a 60 fps e 640x480p 60/90. A câmara é compatível com a versão mais recente do Raspbian, o sistema operativo que foi usado neste projeto.

4.6 Motores DC

Figure 4.8: Conjunto motor DC e RODA

Foi adquirido o seguinte conjunto de motor e roda que constitui em dois motores TT 130 e duas rodas de 66 mm. Este motor é alimentado com uma tensão de 3 V a 6 V DC e uma corrente de aproximadamente 180 mA. O motor tem uma relação 48:1, uma velocidade de rotação de cerca de 80 rpm, um binário de 0,5 kg*cm. Este motor foi escolhido pelo baixo preço e pela compatibilidade com os sistemas anteriores, sendo no nosso caso necessário sinais binários para controlar a direção dos motores e sinais analógicos para

controlar a velocidade.

4.7 Desenho em 3D

Uma das partes mais importantes na prototipagem é o design 3D.

O design 3D utiliza software de design assistido por computador para modelação e documentação técnica. É utilizado para modelar e visualizar produtos e elementos do mundo real, e para estudar e otimizar a sua conceção antes de entrar no processo de fabrico. O que tradicionalmente era feito com recurso a desenho manual pode agora ser substituído por software CAD ligado sem problemas a tecnologias emergentes que permitem aos designers e engenheiros iterar mais rapidamente nos projetos, reduzir os custos renunciando aos custos de prototipagem e ao desperdício de materiais e melhorar a qualidade geral do produto.

Neste projeto é imperativo a utilização desta tecnologia para manter os custos de produção baixos, visto que é necessário este design para proceder a impressão 3D de várias peças.

Para este design foi escolhido o software Solidworks, que é utilizado por milhões de projetistas e engenheiros em centenas de milhares de empresas. É um dos softwares de projeto e engenharia mais populares do mercado. Conhecido pela sua gama de características e elevada funcionalidade, sendo utilizado em várias profissões e indústrias em todo o mundo.

O Solidworks utiliza o design paramétrico, razão pela qual é uma ferramenta tão eficaz para designers e engenheiros. Isto significa que o designer pode ver como as alterações irão afetar os componentes vizinhos, ou mesmo a solução global. Por exemplo, se o tamanho de um único componente for aumentado, isso afetará a junta ou o orifício ao qual está ligado. Isto permite

aos projetistas detetar e corrigir problemas de forma rápida e fácil. Sendo assim o software ideal para este projeto.

O objetivo deste design é desenvolver uma plataforma que suporte e ligue todos os componentes mecânicos deste projeto. O primeiro passo deste design foi fazer o levantamento das medidas físicas recorrendo a um paquímetro digital, tais como as furações da ponte H, dos motores e da roda giratória padrão. Além destas medidas, também foram tiradas as medidas externas da bateria portátil para desenhar a base do veículo de forma a agrupar todos estes componentes.

Com esta informação e sabendo que o método de produção futura será de impressão 3D as peças foram desenhadas com o objetivo de terem as características principais apenas num plano, ficando assim mais resistentes quando impressas. Neste contexto foram desenhadas as seguintes peças:

- **Base:** A primeira peça e a mais importante foi a base onde todos os outros componentes serão ligados através de furações e respetivos parafusos (Anexo C.1).

- **Suporte motores:**Com as medidas exteriores dos motores selecionados e com as medidas das furações já feitas nos motores foi desenhado um suporte de motor que conecta o motor à estrutura da base a utilizar dois parafusos M4 e as respetivas porcas para cada motor (Anexo C.2).

- **Suporte câmara:** Para conectar a câmara à estrutura foi desenhado um suporte que levanta o ponto de vista em relação a estrutura do veiculo. Com um ponto de vista mais elevado conseguimos assim uma melhor imagem da fita diretamente à frente do carro, o que facilita

a análise da mesma. A ligação entre este suporte e a câmara é feita por um parafuso, ou seja o ângulo da câmara também é facilmente ajustável(Anexo C4).

- **Suportes bateria:** Foram também projetados uns suportes simples onde a bateria assenta na base(Anexo C3).

- **Espaçadores:** Finalmente foram também projetados uns simples espaçadores para a ligação da ponte H à base de modo aos pinos da parte inferior da placa de ponte H não ficarem em contanto direto com a base.

A figura 4.8 mostra uma renderização do veículo incluindo as peças desenhadas e as peças padrão.

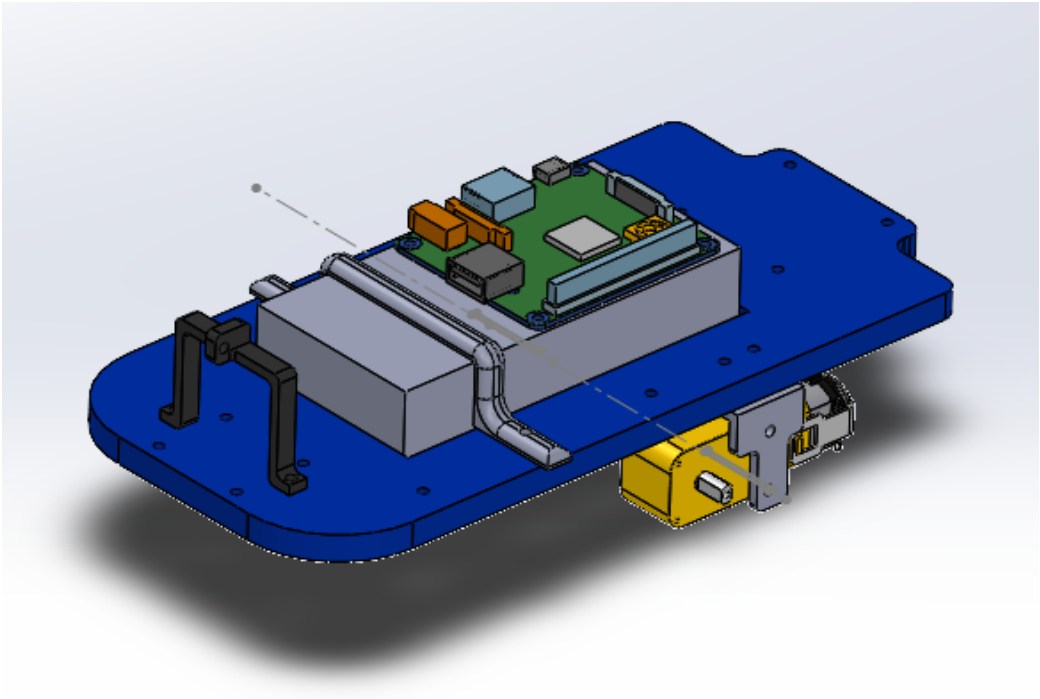


Figure 4.9: Renderização do veículo

4.8 Impressão peças em 3D

Para a fabricação das peças anteriormente referidas foi escolhido o método de impressão em 3D, também conhecido por técnica aditiva de fabrico.

Em resumo o método de impressão em 3D é o mais utilizado numa situação de prototipagem devido ao seu baixo custo de produção, onde em média 1kg de material(PLA) custa entre 15 a 20 euros. Para termos em perspetiva, todas as peças utilizadas neste projeto rondam os 100 gramas ou seja no máximo 2 euros em custo de material.

Este processo começa geralmente com um modelo criado em softwares de design(CAD) como o Solidworks que foi o utilizado neste projeto, uma vez que temos a peça completamente desenhada com as dimensões desejadas exportamos o ficheiro no formato STL.O nome STL é um acrónimo que significa estereolitografia , uma tecnologia de impressão 3D popular. Também pode ser referido como Standard Triangle Language ou Standard Tessellation Language, neste formato cada ficheiro é composto por uma série de triângulos ligados que descrevem a geometria da superfície de um modelo ou objeto 3D. Quanto mais complexo for o desenho, maior será o número de triângulos utilizados e maior será a resolução. Já com um ficheiro em formato STL apenas falta passar essa informação para uma série de passos ou G-Code, a linguagem que a impressora 3D recebe para criar um objeto. Esta tradução de STL para G-Code pode ser feita a partir de vários softwares, sendo que o que foi utilizado é o Cura, considerado pela comunidade o melhor software deste tipo tanto em termos de preço, visto que é grátis, em termos de curva de aprendizagem, sendo bastante intuitivo e finalmente em qualidade de pro-

duto final. Nestes softwares é possível alterar diversos parâmetros relativos à impressão, desde temperatura de fundição do material (para diferentes materiais existem diferentes temperaturas de fundição adequadas), velocidade, enchimento da peça, altura de camadas e muitos mais, todos responsáveis pela qualidade final da peça. Finalmente estes parâmetros todos são exportados para um cartão SD e inseridos na impressora onde é apenas necessário selecionar o ficheiro relevante e a produção começa sozinha, camada a camada até termos a peça final. Para este projeto foi utilizada a impressora Creality Ender 3.

Figure 4.10: Impressora 3D Creality Ender 3

Em relação ao material foi usado um rolo de plástico PETG genérico com uma temperatura de fusão de aproximadamente 230 graus celsius.

E com todos os componentes de Hardware ficamos com um veiculo como aparece na figura 4.11.

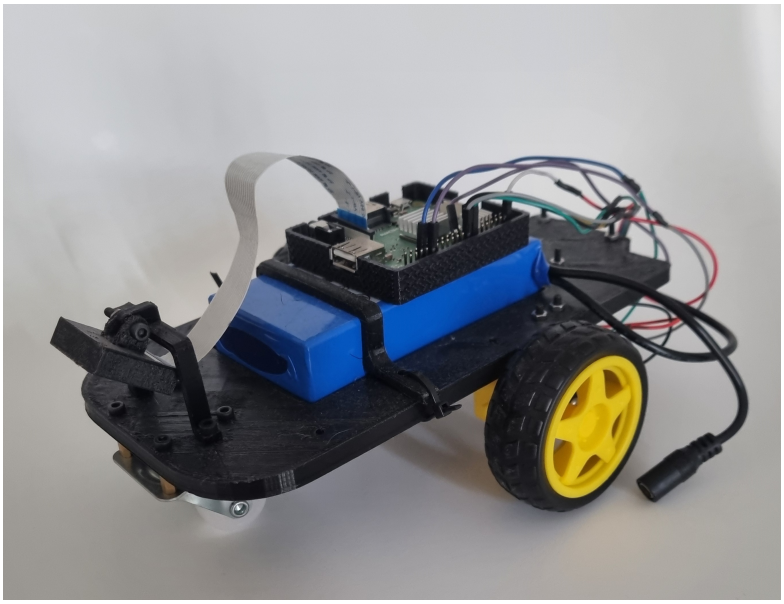


Figure 4.11: Veiculo Completo

Chapter 5

Código

Nesta secção será descrito o código efetuado em Python. O objetivo geral deste projeto foi de garantir que um veículo guiado autonomamente conseguia seguir uma linha preta no chão com a maior velocidade possível. Isto num sistema fechado, ou seja o sistema captura as imagens e toma as decisões sem ajuda de processamento externo. Para começar foram importadas todas as bibliotecas necessárias, tais como: time , math , cv2(opencv) , numpy, gpio e finalmente o método responsável pela comunicação com os motores e o controlo de direção. É efetuada fora do ciclo principal a inicialização da câmara, onde criamos uma instância da classe Picamera e de seguida os atributos necessários, tais como a resolução da câmara e o framerate desejado. De seguida desenvolveu-se a função **Virar** que é a responsável por comparar o valor retirado do loop principal(valor este das coordenadas do centro da linha) e comparar com os valores de threshold adequados, e depois passar essa decisão de virar ou à esquerda , à direita, ou continuar em frente para o modulo secundário **motores**. No ciclo principal temos como objetivo obter

a informação relativa à posição da fita no chão relativamente ao veículo, para depois com esta informação tomar a decisão na função **Virar** descrita anteriormente. Para obter esta informação é aplicada a seguinte análise a todas as frames da câmara. Primeiro obtém-se o frame atual, que depois tem de ser convertido para um formato de matriz utilizando a biblioteca **numpy**. Agora com a imagem já em formato de matriz, esta vai sofrer o seguinte processamento:

1. **Redimensionamento:** Já com a imagem num tipo de matriz é feito um redimensionamento no eixo y da imagem passando de 480 pixeis de altura para 130. Em resultado deste redimensionamento temos imediatamente um aumento em performance considerável visto que temos apenas de processar agora 27% da imagem inicial.

2. **cv2.flip:** Visto que a câmara está fisicamente montada ao contrário este primeiro método serve para rodar a imagem em torno do eixo y, ajudando a simplificar o troubleshooting visual.

3. **cv2.Color:**Neste processamento é aplicado a imagem um filtro de cinzento simplificando ainda mais a imagem visto que agora só temos um nível de matriz em relação ao formato anterior (RGB).

4. **cv2.threshold:**É agora aplicado um método de binarização, ou seja todos os pixeis irão passar a ter apenas dois valores ou um ou zero, sendo esta decisão feita comparando o tom de cinzento ao valor de threshold predefinido, e depois de muitos testes em relação à luminosidade ambiente para a binarização mais eficiente concluiu-se que o valor mais adequado era o 45.

5. **cv2.erode:** Como anteriormente descrito este modulo tem o objetivo de retirar pixeis soltos na imagem que posteriormente poderiam baixar a eficácia dos módulos finais de deteção de fonteiras e de deteção de linhas.

6. **cv2.dilate:** Agora com todos os pixeis "falsos" retiradas o método de erosão também erode o objeto principal, no nosso caso a fita no chão, e com esta dilação retornamos o objeto principal ao tamanho original.

Finalmente com todo o pré-processamento completado passamos aos métodos principais onde são retirados os valores importantes do frame em análise. O primeiro método a ser aplicado é o método de deteção de linhas Canny. Este método como já explicado tem como argumentos a imagem já processada e dois valores de limiar, onde é obtido a melhor deteção de linhas com os valores do primeiro threshold a 50 e o segundo de 200. Com com um método de deteção de linhas efetuado é possível aplicar aplicar a transformada probabilística de Hough que tem como argumentos a imagem anterior, uma lista onde vão ser armazenados os valores dos pontos das linhas detetadas da seguinte forma:$(x_{inicial}, y_{inicial}, x_{final}, y_{final})$. Além destes argumentos é atribuído também um limite mínimo de interseções para ser considerada uma linha, neste caso o melhor valor é 10, o numero mínimo de pontos para ser considerada uma linha de modo a não haver falsos positivos que interfiram com os cálculos seguintes, sendo também o valor 10 para este argumento, e finalmente é também definido o espaçamento entre dois pontos para serem considerados da mesma linha, sendo atribuído para este argumento o valor de 100. Depois destes métodos a análise da imagem está completa e temos agora uma lista de linhas, tendo para cada linha a seguinte informação já

descrita :($x_{inicial}, y_{inicial}, x_{final}, y_{final}$). Com estes valores é feito o seguinte processamento:

1. Primeiro é feita uma comparação dos valores de $x_{inicial}$ e x_{final} de todas as linhas e garantido que o valor absoluto entre os dois valores é inferior a 20, para serem descartadas todas a linhas possivelmente horizontais que não acrescem valor nenhum ao processamento, e pelo contrário iriam negativamente influenciar a decisão final.

2. De seguida é analisada cada linha e retirada o valor de $x_{inicial}$ e anexado a uma nova lista exclusivamente todos os x iniciais de todas as linhas.

3. A esta lista é aplicada uma ordenação para os valores ficarem organizados de forma crescente.

4. Finalmente é retirado o primeiro valor da lista (o menor valor) e o último valor da lista (o maior valor). Com estes dois valores assumimos que cada um destes valor é referente a cada uma das fronteiras da linha, e fazendo a média destes valores ficamos com um valor final do centroide da linha.

Temos assim completa a análise de um frame com o output de um valor ao qual é aplicado a função **Virar**.

Completando, para esta análise ser refletida fisicamente temos o módulo **motors** onde são definidas as funções que representam as direções ou apenas seguir em frente. Neste módulo o controlo dos motores é feito a partir de quatro pins que são definidos como saídas digitais. Estes sinais estão ligados à placa de ponte H que, dependendo da combinação dos sinais controla o movimento dos motores. É de notar que não há regulação direta da velocidade

dos motores, é apenas controlado durante quanto tempo é que os motores rodam para qualquer que seja o lado predefinido. E assim, ao controlar o tempo de ativação e a direção dos motores temos o controlo de velocidade e direção implementado.

Chapter 6

Testes e Conclusão

6.1 Testes

Para testar as capacidades do projeto, vai ser analisada a capacidade de percorrer dois percursos de teste diferentes. O primeiro percurso será apenas uma linha reta com o objetivo de conseguir analisar a capacidade de simplesmente seguir a linha e também conseguir retirar um valor da velocidade média máxima do veículo que é atingida teoricamente em linha reta. O segundo percurso será um percurso com a forma oval analisando a capacidade do veiculo de efetuar curvas.

Os parâmetros finais a serem julgados serão a velocidade com a qual o veículo percorre o percurso e se foi preciso reajustar o veículo, ou seja houve algum movimento não desejado o que causou o veiculo a deixar de ver a linha do chão, sendo este movimento possivelmente causado por uma alteração de luminosidade causando uma má interpretação da linha ou um movimento demasiado brusco dos motores tirando a linha do campo de visão do veículo.

Com o objetivo de obter o melhor resultado possível, ou seja a maior velocidade sem haver necessidade de reajustar o veículo, temos então os seguintes parâmetros importantes que afetam diretamente os resultados:

- **Threshhold luminosidade:** Visto que os testes são feitos num ambiente com uma luminosidade não constante qualquer variação significativa afeta os valores lidos pela câmera e por consequência os valores analisados. Na altura dos testes o valor de threshhold com os resultados de deteção de linhas mais constante é o de 45, sendo então este igual em todos os testes.

- **Tensão fornecida aos motores:** Como já foi descrito os motores utilizados têm uma tensão funcinal variável desde os 3 V a 6 V. Este valor é também um parâmetro que foi necessário encontrar o melhor equilíbrio entre velocidade enquanto era garantido que a linha ficava dentro do campo de visão (visto que com uma tensão elevada cada processamento resultava num movimento excessivo retirando a linha do campo de visão).

- **Tempo de ativação dos motores:** Finalmente , e juntamente com o parâmetro anterior este parâmetro é responsável pelo ângulo de rotação por frame analisado se a decisão for rodar, ou o tempo de deslocação em frente quando a decisão feita é seguir em frente.

Foram então feitos primeiro dez testes por percurso com o objetivo de encontrar os melhores parâmetros para ambos os percursos, resumidamente atingir a velocidade média o mais alta no percurso em reta, garantido ainda que percorre o percurso oval sem incidentes.

De seguida apresentam-se duas tabelas com os resultados dos testes para os dois percursos, notando que foi inicialmente feito os testes no percurso 1 dando assim um intervalo de possíveis valores dos parâmetros inferior para o segundo percurso. É também de notar que na tabela com o título **"Foi necessário reajustar"** refere á necessidade de voltar a reposicionar o veículo em cima da linha quando a reação dos motores era excessiva. E finalmente o tempo percorrido referente a cada teste não contém o tempo necessário para reajustar o veículo, tendo assim tempos inferiores e velocidades médias maiores em alguns casos que foi preciso reajustar o veículo.

Temos então a primeira tabela referente ao primeiro percurso na figura 6.1.

Percurso um						
Teste nº	Threshold Luminosidade	Tensão fornecida aos motores (% de 0-12V)	Tempo ativação motores (seg)	Tempo percorrido (seg)	Foi necessário reajustar (Sim/ Não)	Velocidade media (m/s)
1	45	40	0.2	9	Sim	0,15
2	45	35	0.2	10	Sim	0,13
3	45	35	0.15	12	Sim	0,11
4	45	30	0.15	13	Sim	0,10
5	45	27	0.15	15	Sim	0,09
6	45	25	0.13	16	Sim	0,08
7	45	25	0.1	18	Sim	0,07
8	45	24	0.07	19	Não	0,07
9	45	24	0.06	21	Não	0,06
10	45	24	0.05	24	Não	0,06

Figure 6.1: Tabela percurso 1

Recordando que este percurso é referente a uma reta com 134 cm, medida esta para utilizada para calcular a velocidade média para cada teste. Nesta primeira tabela temos primeiros testes com um tempo percorrido bastante

baixo devido ao elevado movimento efetuado por frame analisado, mas ocorrendo uma decisão de virar ou a esquerda ou direta resultava num movimento mais elevado do que o necessário levando constantemente a que o veículo perdesse a linha do campo de visão sendo necessário o reajuste. Tendo acontecido este fenómeno até chegarmos a valores a rondar os 25% de tensão de alimentação dos motores e tempo de ativação de 0,27 seg de tempo em que os motores estão ativos efetuando a rotação, sendo que a partir destes valores já não foi necessário reajustar o veículo.

Com estes valores em consideração foi iniciado o segundo percurso oval mais complexo em que os resultados obtidos foram os representados na figura 6.2.

Percurso dois						
Teste nº	Threshold Luminosidade	Tensão fornecida aos motores (% de 0-12V)	Tempo ativação motores (seg)	Tempo percorrido (seg)	Foi necessário reajustar (Sim/ Não)	Velocidade media (m/s)
1	45	22	0.02	41	Não	0,04
2	45	23	0.03	36	Não	0,05
3	45	24	0.04	31	Não	0,06
4	45	24	0.05	25	Não	0,07
5	45	25	0.06	25	Sim	0,07
6	45	25	0.07	23	Sim	0,08
7	45	25	0.08	23	Sim	0,08
8	45	24	0.09	22	Sim	0,08
9	45	24	0.10	21	Sim	0,09
10	45	24	0.11	21	Sim	0,09

Figure 6.2: Tabela percurso 2

No segundo percurso a abordagem foi a contrária, ir aumentando a potência fornecida aos motores e ir aumentando o tempo de ativação dos motores entre frames. Foram então concluídos os melhores parâmetros para ambos

os percursos oferecendo assim a velocidade máxima sem perder a precisão necessária para ser autónomo. Sendo estes valores finais de 24% de tensão fornecida aos motores, ficando cerca de 3 V, e um tempo de ativação dos motores de 0,05 seg.

6.2 Conclusão

Em conclusão foi alcançado o objetivo de um veículo guiado autonomamente capaz de seguir uma linha no chão através de um processamento e controlo completamente feito a partir do próprio.

Neste projeto foi bastante interessante aprofundar os conhecimentos e as praticamente infinitas possibilidades da visão artificial, e mais do que isso a facilidade de acesso a bibliotecas e ferramentas gratuitas que ficam atrás em termos de capacidade relativamente ás ferramentas usadas pela indústria. Outra nota também bastante importante é a possibilidade de ajustes e possíveis melhoramentos a este projeto numa fase em que custos e tempo não são um entrave, como por exemplo:

- **Estação de carregamento autónoma:** Usando um sistema de carregamento sem fios, e uma entrada analógica no raspberry pi com a informação da tensão atual da bateria, poderia ser implementada uma função para o veiculo se dirigir a uma estação de carregamento situada no fim de uma linha para recarregar autonomamente.

- **Motores:** Seria também possível aplicar em vez de dois motores implementar quatro melhorando a capacidade de direção.

- **Segurança:** Em termos de segurança também seria possível aplicar sensores para medir a distância até um possível obstáculo, interrompendo o ciclo quando necessário. Além disso também seria possível aplicar sensores de temperatura aos motores, à bateria e ponte H monitorizando contra algum incidente em que ocorresse alguma avaria e através de

um relé cortar a alimentação ao veículo.

- **Luzes:** Seria possível ligar uma luz led à bateria, iluminando a linha à frente do veículo para termos um valor de luminosidade mais constante, facilitando a implementação em ambientes com condições diferentes.

Bibliography

[1] The Sama Team (Samasource Impact Sourcing). *Computer Vision: History How it Works*. Acesso em: 2023/11/17. 2021. URL: https://www.sama.com/blog/computer-vision-history-how-it-works.

[2] Open CV. *Open CV*. Acesso em: 2024/10/10. URL: https://opencv.org/about/.

[3] Medium (DataMount). *What Canny Edge Detection algorithm is all about?* Acesso em: 2023/08/01. URL: https://medium.com/@datamount/what-canny-edge-detection-algorithm-is-all-about-103d94553d21.

[4] Learning Open CV (Bradski and Kaehler). *Hough Line Transform*. Acesso em: 2023/12/17. 2017. URL: https://docs.opencv.org/3.4/d9/db0/tutorial_hough_lines.html.

[5] Autodesk (Nishantk06). *Install OpenCV on Raspberry Pi in Less Than 10 Minutes*. Acesso em: 2023/12/17. URL: https://www.instructables.com/Install-OpenCV-on-Raspberry-Pi-in-Less-Than-10-Min/.

[6] Electronics Hub (Ravi Teja). *Raspberry Pi L298N Interface Tutorial — Control a DC Motor with L298N and Raspberry Pi*. Acesso em: 2023/07/01. 2024. URL: https://www.electronicshub.org/raspberry-

pi-l298n-interface-tutorial-control-dc-motor-l298n-raspberry-pi/.

[7] Medium (Constantin Toporov). *Line following robot with OpenCV and contour-based approach*. Acesso em: 2023/11/01. 2018. URL: https://const-toporov.medium.com/line-following-robot-with-opencv-and-contour-based-approach-417b90f2c298.

[8] Shree K. (First Principals of Computer Vision) Nayar. *Hough Transform — Boundary Detection. YouTube*. Acesso em: 2023/11/17. 2021. URL: https://www.youtube.com/watch?v=XRBc_xkZREg.

[9] PyImageSearch (Adrian Rosebrock). *Accessing the Raspberry Pi Camera with OpenCV and Python*. Acesso em: 2023/07/30. 2015. URL: https://pyimagesearch.com/2015/03/30/accessing-the-raspberry-pi-camera-with-opencv-and-python/.

[10] Big Vision (Krutika Bapat). *Hough Transform with OpenCV (C++/Python)*. Acesso em: 2023/10/10. 2019. URL: https://learnopencv.com/hough-transform-with-opencv-c-python/.

Appendix A

Setup do Raspberry Pi

No desenvolvimento do software o primeiro passo foi instalar o sistema operativo raspbian no raspberry pi, o que foi feito seguindo os seguintes passos: O primeiro passo foi fazer o download do software "Win32 Disk Imager", sendo este um software que nos permite criar facilmente imagens ISO de arranque. É um software de código aberto e foi desenvolvido por gruemaster e tuxinator2009. Podemos utilizar este utilitário para gravar os seus ficheiros ISO em CDs, DVDs e cartões SD/CF. O segundo software a instalar é o "SD Card Formatter", que foi utilizado porque o simples facto de apagar ou eliminar imagens nos seus cartões de memória não elimina totalmente os dados restantes dos cartões. A formatação é uma forma mais completa de limpar ficheiros antigos do cartão e pode reduzir o risco de corrupção de dados. E finalmente a versão mais recente do ficheiro iso do sistema operativo do Raspberry, o Raspbian OS. O segundo passo foi inserir um cartão micro sd com um minimo de 8 GB de memória num leitor de cartões SD e ligado ao computador. Uma vez ligado ao computador temos de saber em que drive é

que o cartão está ligado, e com esta informação usar o software "SD card formatter" para formatar a drive respetiva ao cartão. Com o cartão formatado abrimos o software "win32 Disk Imager" e selecionamos ambos o ficheiro . img referente ao sistema operativo Raspbian, selecionar a drive do cartão SD e clicar em escrever para escrever a imagem no cartão SD.

A.1 Connectar por VNC

No âmbito deste projeto temos de ter a possibilidade de nos ligar e programar no raspberry pi de uma forma wireless, e para ser possível foi usado o software VNC. O VNC (virtual network computing) é um software cliente-servidor que pode ser utilizado para apresentar e controlar o conteúdo do ecrã do sistema de destino (servidor) noutro sistema (cliente). Para este efeito, as entradas do teclado, os movimentos do rato e os cliques no lado do cliente são transferidos diretamente para o computador remoto. Assim, se instalar e ativar o VNC num Raspberry Pi, podemos facilmente instalar, configurar e gerir o minicomputador a partir de outro PC. O VNC é baseado no protocolo de rede multiplataforma Remote Framebuffer Protocol (RFB), que transmite o conteúdo como bitmaps e usa a porta TCP 5900. Sendo tambem que o software VNC foi concebido para ser o mais eficiente possível em termos de recursos. Quanto mais recursos forem necessários para outras aplicações em execução no Raspberry Pi, mais relevante se torna este ponto. Em princípio, no entanto, não há restrições quanto à escolha do modelo Raspberry Pi: pode configurar e executar um servidor VNC tanto num Pi de primeira geração como na versão mais recente. Para começar com a instalação temos primeiro de garantir que temos instalado a versão mais recente do VNC no raspberry pi, e usamos os comandos no terminal mostrados na figura A.1:

Desta forma o Raspbian atualiza tanto a aplicação do servidor como a do cliente. Depois de atualizar o software, pode iniciar o servidor. Pode fazê-lo através da interface de utilizador basta abrir o menu principal (ícone da framboesa) e depois selecionar “definições” e “configuração do Raspberry-

```
sudo apt-get update
sudo apt-get install realvnc-vnc-server
sudp apt-get install realvnc-vnc-viewer
```

Figure A.1: Comandos Terminal para instalar o VNC

Pi". Mude para o separador "interfaces" e defina a entrada VNC como "activada". Depois deste procedimento temos apenas de reiniciar o Raspberry e abrir a aplicação do VNC para retirar a informação do IP do raspberry. Com o VNC instalado no raspberry falta só instalar um software no pc de controlo que nos premita ligar ao raspberry, por exemplo o software "realVNCviewr" onde inserimos o IP do raspberry e temos assim controlo do Raspberry(é de notar que ambos os sistemas tem de estar ligados á mesma rede Wifi).

A.2 Connectar câmera ao raspberry pi

Primeiro temos de ligar o "ribbon cable" da câmera ao adaptador respetivo no raspberry pi. Depois de termos a câmara conectada temos de habilitar o módulo abrindo o terminal e executar o seguinte comando:

```
$ sudo raspi-config
```

Figure A.2: Comando para abrir a configuração da câmara

O comando abre a seguinte janela:

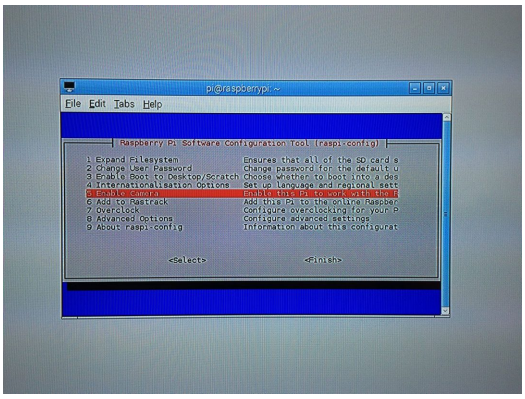


Figure A.3: Janela para habilitar câmara

Utilizando as teclas de seta para se deslocar para a Opção 5: Ativar a câmara, premir a tecla Enter para ativar a câmara e, em seguida, clicar no botão Concluir e premir novamente Enter. Por último, terá de reiniciar o Raspberry Pi para que a configuração tenha efeito[9]. Já com a câmara a funcionar temos de seguida instalar o modulo "picamera" com o seguinte comando no terminal:

```
$ pip install "picamera[array]"
```

Figure A.4: Comando instalar Picamera

Temos de especificar "picamera[array]" e não apenas "picamera", devido ao facto que enquanto o modulo estandardizado "picamera" proporciona métodos de interface com a câmara nos precisamos do sub-modulo opcional "array" para podermos utilizar o OpenCv. Visto que ao utilizar comandos Python, o OpenCV representa as imagens como matrizes NumPy, e o submódulo de matrizes permite-nos obter matrizes NumPy a partir do módulo da câmara do Raspberry Pi[9].

A.3 Setup OpenCV

Já com o os módulos da câmera instalados falta apenas instalar o modulo mais importante, o modulo de OpenCV. Visto que estamos a instalar num raspberry pi a correr rapian Os a instalação é mais complexa do que fosse num computador windows, sendo necessário seguir os seguintes passos: Garantir que o dispositivo tem conexão á internet; Garantir que a versão de Python instalada é superior á 2.7 ; Instalar uma série de dependências necessárias para o funcionamento do OpenCV, que é possível por correr os seguintes comandos[5]:

```
sudo apt-get install libhdf5-dev libhdf5-serial-dev libhdf5-103
sudo apt-get install libqtgui4 libqtwebkit4 libqt4-test
sudo apt-get install libatlas-base-dev
sudo apt-get install libjasper-dev
```

Depois de todas estas dependências instaladas falta instalar a última versão do pip (pip é um sistema de gestão de pacotes padrão de facto usado para instalar e gerir pacotes de software escritos em Python) a correr os seguintes comandos[5]:

```
wget  https://bootstrap.pypa.io/get-pip.py
sudo python3 get-pip.py
```

Agora com todos estes passos concluídos falta apenas o comando final para instalar finalmente o OpenCV que é o seguinte:

```
sudo pip3 install opencv-contrib-python==3.4.6.27
```

E depois da instalação estar completa já temos a biblioteca disponível para desenvolver o código[5].

Appendix B

Código em python

B.1 Motores

```python
import RPi.GPIO as gpio
import time
gpio.setmode(gpio.BOARD)
gpio.setup(32, gpio.OUT)
gpio.setup(33, gpio.OUT)
E1=gpio.PWM(32,100)
E2=gpio.PWM(33,100)
E1.start(24)
E2.start(24)

def init():
    gpio.setmode(gpio.BOARD)
    gpio.setup(13, gpio.OUT)
    gpio.setup(15, gpio.OUT)
    gpio.setup(16, gpio.OUT)
    gpio.setup(18, gpio.OUT)
    gpio.setup(32, gpio.OUT)
    gpio.setup(33, gpio.OUT)

def forward():
    init()
    gpio.output(13, False)
    gpio.output(15, True)
    gpio.output(16, True)
    gpio.output(18, False)
    print(" A ir em frente")
    time.sleep(0.05)
    gpio.cleanup()

#função de marcha a tras que acabou por nao ser utilizada
'''def reverse():
    init()
    gpio.output(13, True)
    gpio.output(15, False)
    gpio.output(16, False)
    gpio.output(18, True)
    time.sleep(sec)
    gpio.cleanup()'''

def right_turn():
    init()
    gpio.output(13, False)
    gpio.output(15, False)
    gpio.output(16, True)
    gpio.output(18, False)
    print(" A virar á direita")
    time.sleep(0.05)
    gpio.output(13, False)
    gpio.output(15, True)
    gpio.output(16, True)
    gpio.output(18, False)
    time.sleep(0.05)

    gpio.cleanup()

def left_turn():
    init()
    gpio.output(13, False)
    gpio.output(15, True)
    gpio.output(16, False)
    gpio.output(18, False)
    print(" A virar á esquerda")
    time.sleep(0.05)
    gpio.output(13, False)
```

```
    gpio.output(15, True)
    gpio.output(16, True)
    gpio.output(18, False)
    time.sleep(0.05)

    gpio.cleanup()

# Secção usada para testes
'''seconds = 3
time.sleep(seconds)
print("forward")
forward()
time.sleep(seconds-2)
print("right")
right_turn()
time.sleep(seconds-2)
time.sleep(seconds)
print("forward")
forward()
time.sleep(seconds-2)
print("right")
left_turn()
time.sleep(seconds-2)
gpio.cleanup()'''
```

B.2 Código Principal

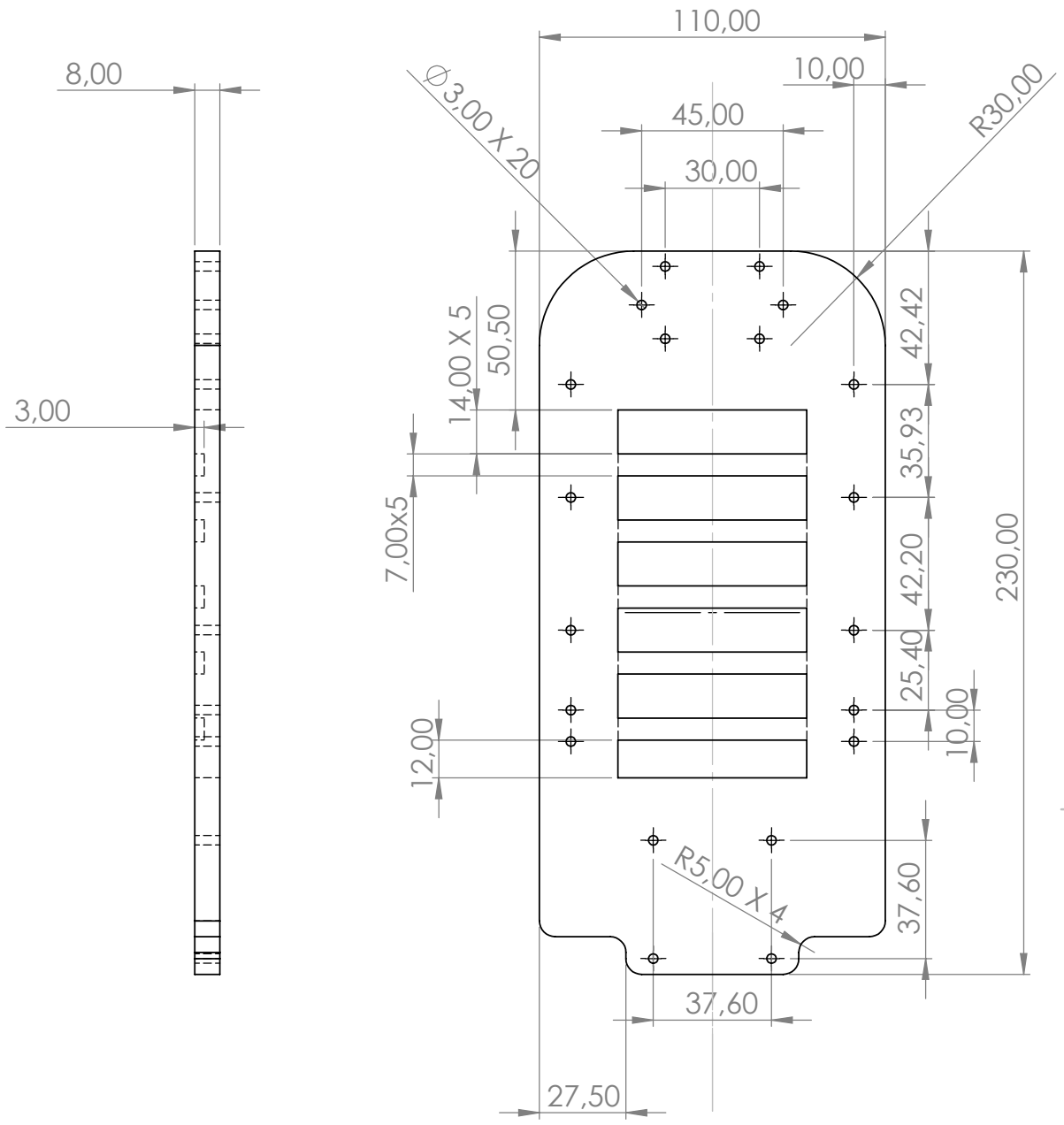
```python

from picamera.array import PiRGBArray
from picamera import PiCamera
import time
import math
import cv2
import numpy as np
import RPi.GPIO as gpio
import motors
gpio.cleanup()
#Inicialização da camera
camera = PiCamera()
camera.resolution = (640, 480)
camera.framerate = 32
rawCapture = PiRGBArray(camera, size=(640, 480))
# Tempo de pausa para a camera inicializar
time.sleep(1)


def Virar(ang):
    if ang<280:
        print("Virar direita")
        motors.right_turn()
    elif ang>320:
        print("virar esquerda")
        motors.left_turn()
    else :
        print("Continuar em Frente")
        motors.forward()

# capturar frames da camera
for frame in camera.capture_continuous(rawCapture, format="bgr", use_video_port=True):

    img = frame.array
    img = img[170:300,1:640]
    img = cv2.flip(img, 1)
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    img = cv2.threshold(gray,45,255, cv2.THRESH_BINARY_INV)[1]
    img = cv2.erode(img, None, iterations=5)
    img = cv2.dilate(img, None, iterations=5)
    cv2.imshow("img", img)
    edges = cv2.Canny(img, 45, 200,apertureSize=3 )
    cv2.imshow("edges", edges)
    lines = cv2.HoughLinesP(edges, 1, np.pi/180, 4, None, 20,10)
    ListadeLinhas = []
    teste = 0
    avgrho = 0
    ValoresDePosicao = []
    try:
        #print(lines)
        for line in lines:
            #Retirar os valores de x e y das linhas encontradas
            x1, y1, x2, y2 = line[0]
            if abs(x1-x2)< 35 :
                #Este passo tem como objetivo eliminar possiveis falsos positivos de linhas
horizontais
                ListadeLinhas.append(line[0])
                cv2.line(gray, (x1, y1), (x2, y2), (255, 0, 0), 3)
                cv2.imshow("linhas", gray)
            else:
                pass
        for i in range(len(ListadeLinhas)):
            a,b,c,d = ListadeLinhas[i]
            #Vamos agora retirar apenas o valor de x1 de cada linhas
```

```
            ValoresDePosicao.append(a)
        ValoresDePosicao.sort()
        #Com os valores ordenados conseguimos fazer a media do primeiro e ultimo valor tendo assim um valor de posição do centro da linha
        avg = (ValoresDePosicao[0] + ValoresDePosicao[-1:]) / 2
        print(avg)
        Virar(avg)


    except Exception as e:
        #Passo para quando não houver linhas detetadas continuar o ciclo do programa
        print(e)
        pass


    key = cv2.waitKey(1) & 0xFF

    rawCapture.truncate(0)

    if key == ord("q"):
        break

gpio.cleanup()

```

Appendix C

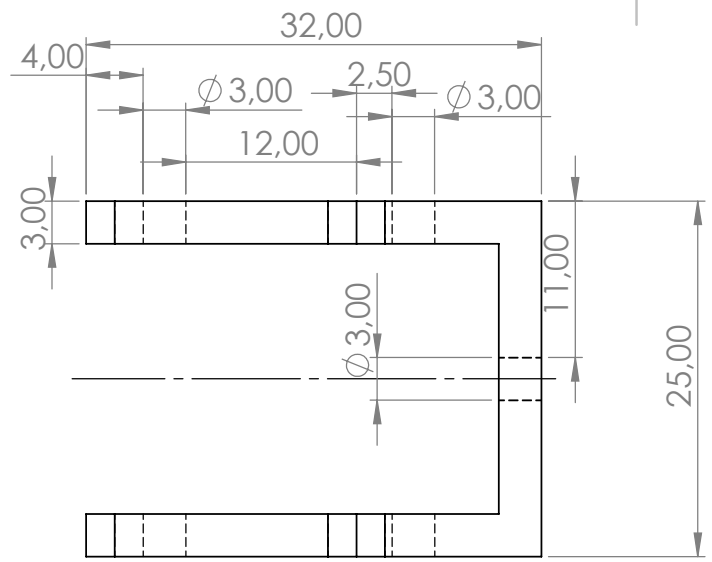
Desenhos 3D

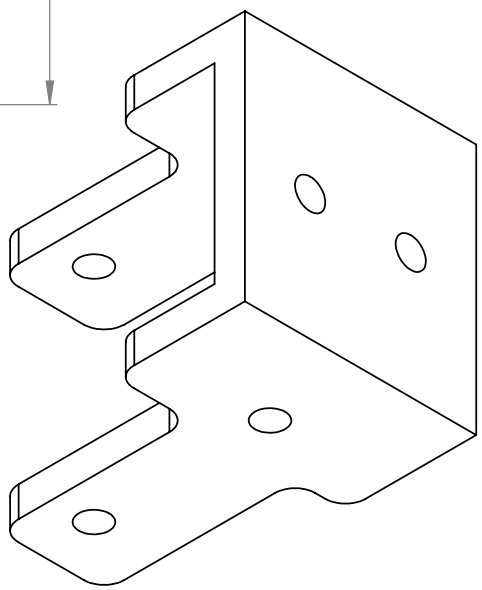
C.1 Base

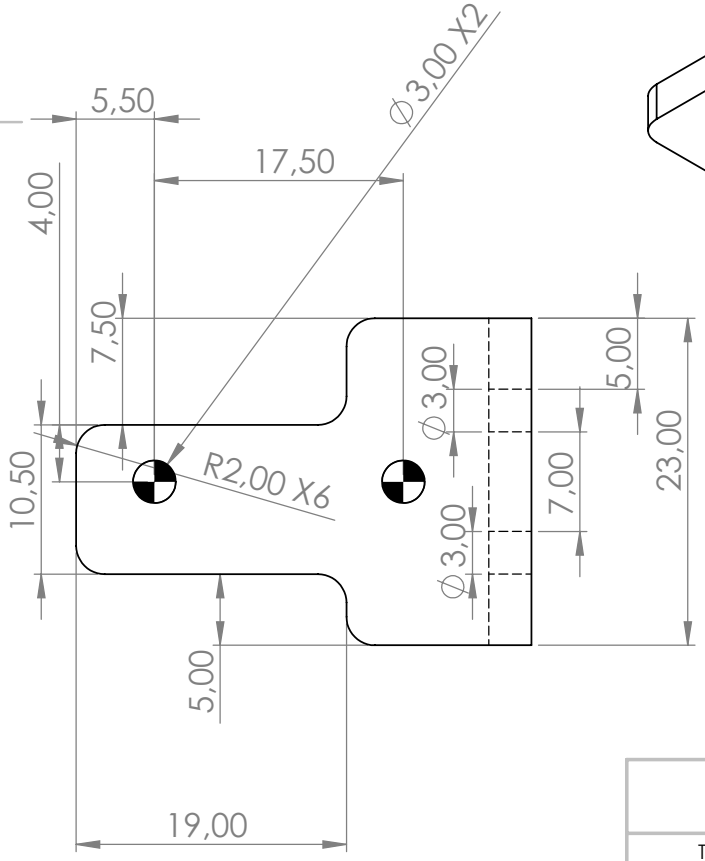


Universidade de Évora		
TITLE: Base		
DWG NO. 1	DRAW BY: Miguel Silva	A4
SCALE:1:2	SHEET 1 OF 1	

C.2 Suporte do Motor

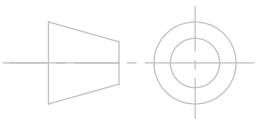






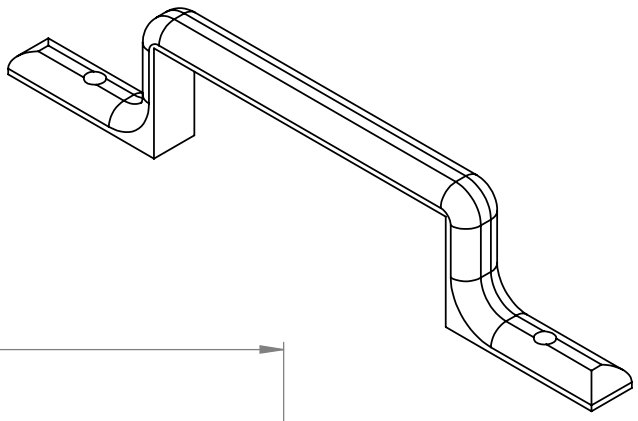
Universidade de Évora

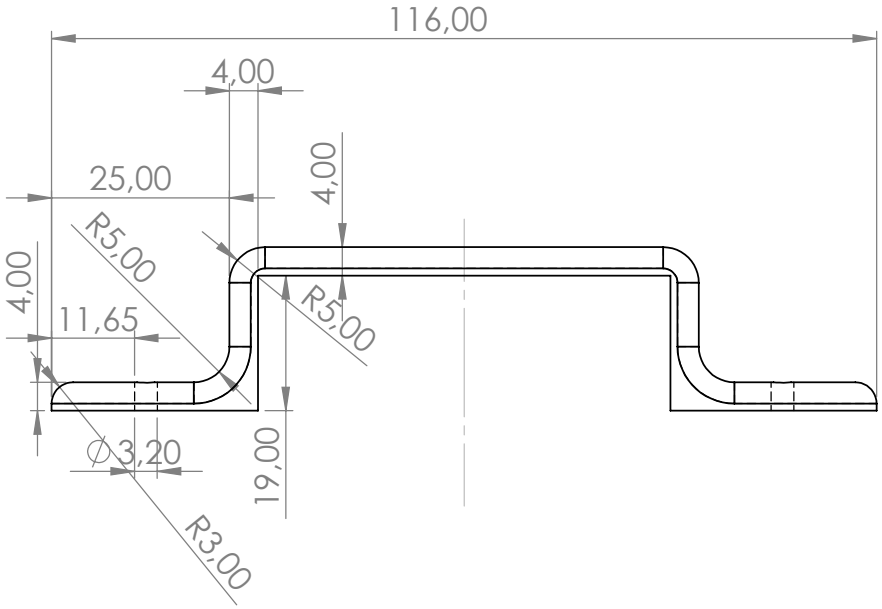
TITLE:
Suporte Motor

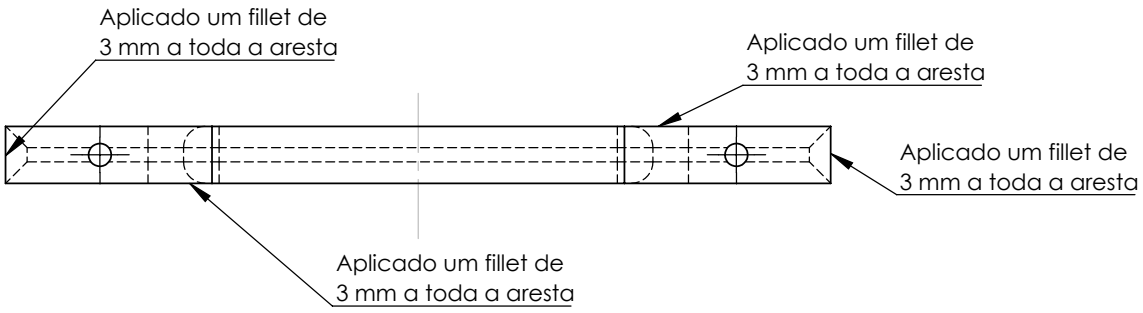


DWG NO.	DRAW BY:	
2	Miguel Silva	A4
SCALE:2:1	SHEET 1 OF 1	

C.3 Suporte da Bateria

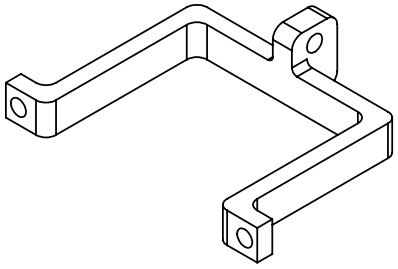


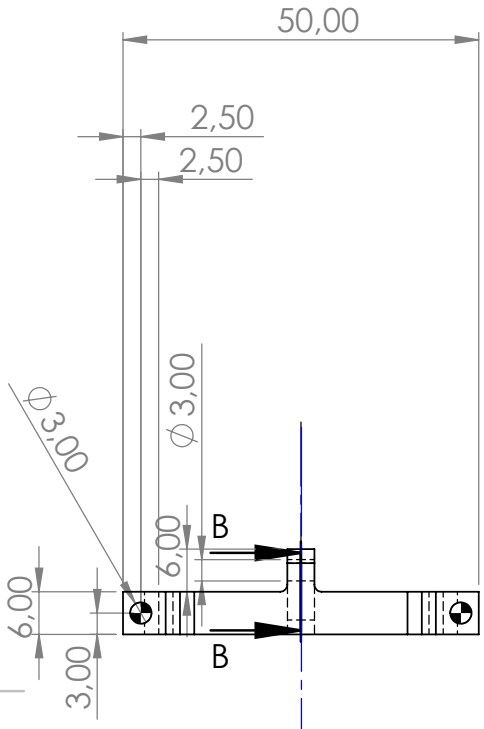


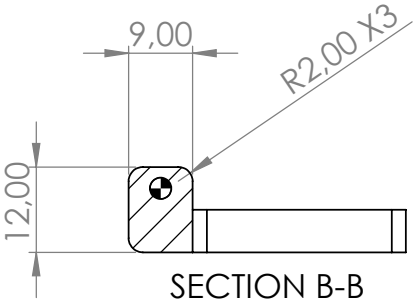


Universidade de Évora		
TITLE: suporte bateria		
DWG NO. 4	DRAW BY: Miguel Silva	A4
SCALE:1:1	SHEET 1 OF 1	

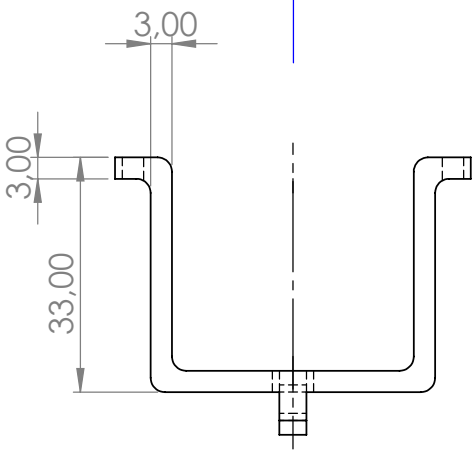
C.4 Suporte da Câmara







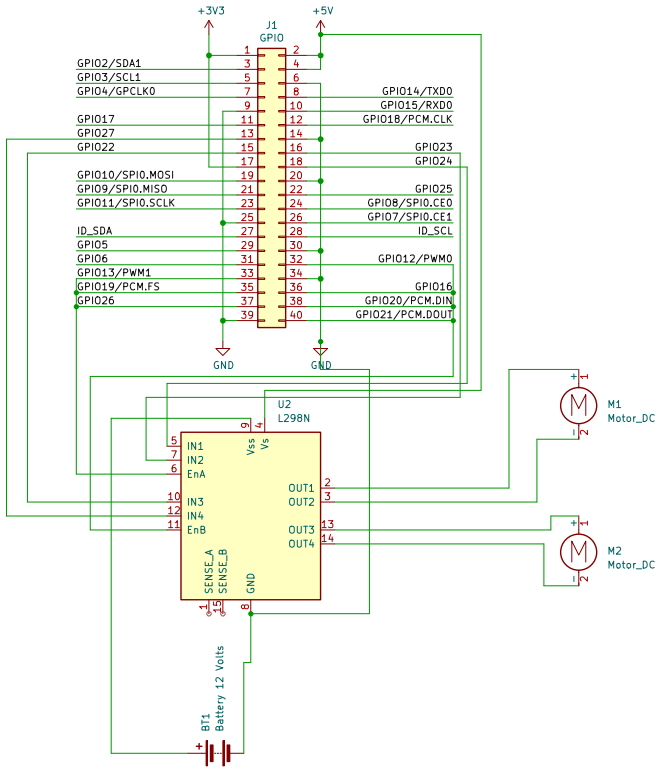
SECTION B-B



Universidade de Évora		
TITLE: Suporte Camera		
DWG NO. 3	DRAW BY: Miguel Silva	A4
SCALE:1:1	SHEET 1 OF 1	

Appendix D

Esquema Elétrico



Universidade de Évora		
Sheet: /		
File: esquema.kicad_sch		
Title: Esquema eletrico veículo		
Size: A4	Date: 15 nov 2012	**Rev:**
KiCad E.D.A. 8.0.6		Id: 1/1