Politécnico
de Viseu

Escola Superior
de Tecnologia
e Gestão de Viseu

Desenvolvimento de plataforma autónoma para a Agricultura

Pedro Fernando Marques Gonçalves

Dissertação

Mestrado em Engenharia Eletrotécnica - Energia e Automação Industrial

Trabalho efetuado sob a orientação de
Professor Doutor Vasco Eduardo Graça Santos
Professor Doutor Daniel Filipe Albuquerque

Março de 2023

Politécnico
de Viseu

Escola Superior
de Tecnologia
e Gestão de Viseu

Desenvolvimento de plataforma autónoma para a Agricultura

Pedro Fernando Marques Gonçalves

Dissertação

Mestrado em Eng. Eletrotécnica - Energia e Automação Industrial

Trabalho efetuado sob a orientação de

Professor Doutor Vasco Eduardo Graça Santos
Professor Doutor Daniel Filipe Albuquerque

Março de 2023

Dedicatória

Aos meus pais Carlos Gonçalves e Maria Jacinta Gonçalves, por nunca me terem deixado desistir e pelo incentivo e apoio sempre para lutar pelos meus objetivos.

"O fracasso é apenas a oportunidade de recomeçar de novo, desta vez de forma mais inteligente" - **Henry Ford**

RESUMO

Nas últimas décadas, o crescimento populacional tem aumentado, provocando o uso exaustivo dos recursos do planeta em todos os setores. Um dos setores mais afetados é a agricultura, que teve de se adaptar às mudanças dos tempos e, consequentemente, necessita de produzir uma maior quantidade de alimentos para suprir o aumento populacional. Atualmente, a agricultura adquiriu processos baseados na indústria, com o intuito de se tornar mais inteligente e, desta forma, conseguir ser mais eficiente e produtiva. Uma das razões que permitem esta melhoria é a introdução de tecnologia.

A dissertação proposta tem como objetivo o desenvolvimento de um sistema de navegação autónomo para a plantação de árvores, com a perspetiva futura de poder ser utilizado para outras tarefas, mediante a alteração da ferramenta acoplada ao robô. Para a realização desta ideia, foi necessário obter uma estrutura que permitisse ter condições de movimentação e fosse versátil a mudanças estruturais necessárias.

O método utilizado em definitivo para a realização da estratégia de navegação foi a visão computacional, com a utilização das bibliotecas *OpenCV* e *TensorFlow*, na linguagem *Python.* Ao utilizar a estratégia de navegação baseada na visão computacional, temos dois modos de funcionamento: um para a noite e outro para o dia. As estratégias consistem na utilização de um ponto de referência para o robô se guiar. Temos então uma estratégia para funcionar de dia e que consiste na utilização da biblioteca *TensorFlow* para detetar objetos. Através da deteção do objeto consegue-se utilizar o mesmo como referência para o robô se guiar. Neste caso durante os testes realizados utilizou-se uma pessoa, mas pode ser utilizado uma cadeira, um carro ou outro objeto, visto que a biblioteca permite identificar um conjunto de objetos. Para a estratégia noturna o funcionamento consiste na utilização da biblioteca *OpenCV*. Esta estratégia consiste na utilização de um projetor *LED* infravermelho como referência e o robô deteta o ponto com maior brilho na imagem captada pela câmara. Consequentemente em função da deteção, o robô executa a sua movimentação em direção a esse ponto.

Concluindo, este projeto teve bastantes evoluções, como qualquer projeto, permitindo detetar necessidades de melhoria e de possíveis abordagens diferentes a serem realizadas para um futuro trabalho. Com base nos resultados obtidos, foi possível cumprir, de forma mínima, a estratégia de navegação autónoma baseada na visão computacional. Os resultados obtidos permitiram visualizar que foi possível obter uma execução razoável em função das condições de funcionamento existentes. Contudo existe a necessidade de constante evolução do projeto, de forma a tentar melhorar a execução de movimentação do robô, procurando o rigor na sua movimentação.

ABSTRACT

In recent decades, the world's population growth has been increasing, leading to the exhaustive use of resources in all sectors. One of the most affected sectors is agriculture, which has had to adapt to the changing times and consequently needs to produce a larger quantity of food to meet the growing population's demands. Currently, agriculture has adopted industry-based processes in order to become smarter and, thus, more efficient and productive. One of the key factors enabling this improvement is the introduction of technology.

The proposed dissertation aims to develop an autonomous navigation system for tree planting, with the future prospect of being adaptable for other tasks by modifying the tool attached to the robot. To accomplish this idea, it was necessary to obtain a structure that would allow for adequate mobility and be versatile enough to accommodate necessary structural changes.

The chosen method for implementing the navigation strategy was computer vision, utilizing the OpenCV and TensorFlow libraries in the Python language. By employing a computer vision-based navigation strategy, there are two operating modes: one for nighttime and one for daytime. The strategies involve using a reference point for the robot to navigate. For the daytime strategy, the TensorFlow library is utilized for object detection. By detecting the object, it serves as a reference point for the robot to navigate towards. During the conducted tests, a person was used as the reference object, but it could be a chair, a car, or any other identifiable object since the library can detect a variety of objects. For the nighttime strategy, the operation involves using the OpenCV library and an infrared LED projector as a reference. The robot detects the brightest point in the captured image, and accordingly, moves towards that point.

In conclusion, this project has undergone significant advancements, as is common with any project, allowing for the identification of areas for improvement and potential alternative approaches for future work. Based on the obtained results, it was possible to achieve, to a minimum extent, the computer vision-based autonomous navigation strategy. The results demonstrated that reasonable execution could be achieved considering the existing operating conditions. However, there is a constant need for project evolution to enhance the robot's movement execution and strive for precision.

PALAVRAS CHAVE

Robô
Navegação autónoma
Agricultura

KEY WORDS

Robot
Autonomous navigation
Agriculture

AGRADECIMENTOS

Gostaria de expressar meus sinceros agradecimentos a todos que contribuíram para o sucesso desta dissertação. Primeiramente, quero agradecer aos meus orientadores, Vasco Eduardo Graça Santos e Daniel Filipe Albuquerque por todo o seu apoio, orientação e paciência ao longo deste percurso. Sem a sua orientação, este trabalho não teria sido possível. Queria agradecer também ao engenheiro Nelson Rafael Rodrigues dos Santos pelo apoio dado no desenvolvimento da parte mecânica da plataforma robótica.

Também quero agradecer aos meus pais Carlos Alberto de Matos Gonçalves e Maria Jacinta Marques Gonçalves e ao meu irmão Francisco Marques Gonçalves, pelo seu amor incondicional, encorajamento e suporte durante toda a minha jornada acadêmica. Vocês foram a minha maior inspiração e força motriz para perseguir os meus sonhos.

Gostaria de agradecer aos meus amigos, que me acompanharam nesta trajetória. O apoio de vocês foi fundamental para me manter motivado e determinado a enfrentar os desafios que surgiram ao longo do caminho. Sou muito grato pela amizade e companheirismo que compartilhamos.

Por fim, gostaria de agradecer a todos os que de alguma forma contribuíram para este trabalho, incluindo a minha instituição de ensino e todos os professores que me ensinaram e inspiraram durante o curso. Este é um momento muito especial para mim e não poderia ter alcançado sem a ajuda e apoio de cada um de vocês.

Muito obrigado!

ÍNDICE GERAL

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

ABREVIATURAS E SIGLAS

3D	Três dimensões
AGV	*Automated Guided Vehicle*
APP	Aplicação
ARM	*Acorn RISC Machine*
CAN	*Controller Area Network protocol*
GNU GPL	*GNU Não é Unix General Public License*
GPIO	*General Purpose Input/Output*
GPS	*Global Position System*
GTK+	*GIMP ToolKit*
I&D	Investigação & Desenvolvimento
ICT	*Information and Communication Technology*
IDE	*Integrated Development Environment*
IMU	*Inertial Measurement Unit*
LED	*Light Emitting Diode*
LIDAR	*Light Detection and Ranging*
OpenCV	*Open Source Computer Vision*
RTK-GPS	*Real Time Kinematic – Global Position System*
TDOA	*Time Difference Of Arrival*
TOF	*Time Of Flight*
USB	*Universal Serial Bus*
UWB	*Ultra Wind Band*
WIFI	*Wireless Fidelity*

1. Introdução

1.1 Contexto e Motivação

O crescimento em cerca de 2 mil milhões de pessoas nos próximos 30 anos, dos atuais 8 mil milhões para 9,7 mil milhões em 2050 e que poderá atingir os 10,9 mil milhões no final do século é um indicador e um alerta para o futuro pois o consumo aumentará de forma linear em quase todos os setores. Este crescimento exponencial da população mundial e as alterações climáticas conduzem, em termos de sustentabilidade, a um novo paradigma, nomeadamente no que respeita à alimentação das populações, pelo que a agricultura, entre outras atividades (pesca, etc) são setores que irão sofrer uma maior pressão, uma vez que terão de fazer face às necessidades alimentares de todos.

No que refere à agricultura o aumento da área arável não é solução pois a disponibilidade de terrenos agrícolas é cada vez menor [1], pelo que é fundamental aumentar a produtividade e introduzir novos produtos agrícolas, animais e florestais, para se aumentar a produção de bens de consumo. Só assim é possível fazer face à crescente procura de alimentos, para que a sociedade seja, em termos alimentares, mais equitativa e se possa manter a civilização, tal como a conhecemos no mundo ocidental. Na agricultura, a crescente utilização de equipamentos, integral ou parcialmente suportados por sensores e tecnologias de comunicação (*Information and Communication Technology* – ICT), tem vindo a impor-se, permitindo uma maior racionalidade das atividades agrícolas e aumentos de produtividade e sustentabilidade.

A Índia é um país onde a agricultura tem bastante importância, mas tem alguns problemas que tem vindo a enfrentar, como o aumento de custo de matérias-primas, a falta de mão-de-obra qualificada, a falta de recursos hídricos e a falta de monitorização das plantações são alguns pontos importantes em que a implementação de tecnologias de automação podem maximizar o desenvolvimento da sua agricultura [2].

Muito resumidamente a Figura 1, ilustra a influência que as revoluções industriais tiveram nas atividades agrícolas. E como se pode ver o uso de mão-de-obra intensiva tem vindo a ser substituído por processos de gestão industrial [3].

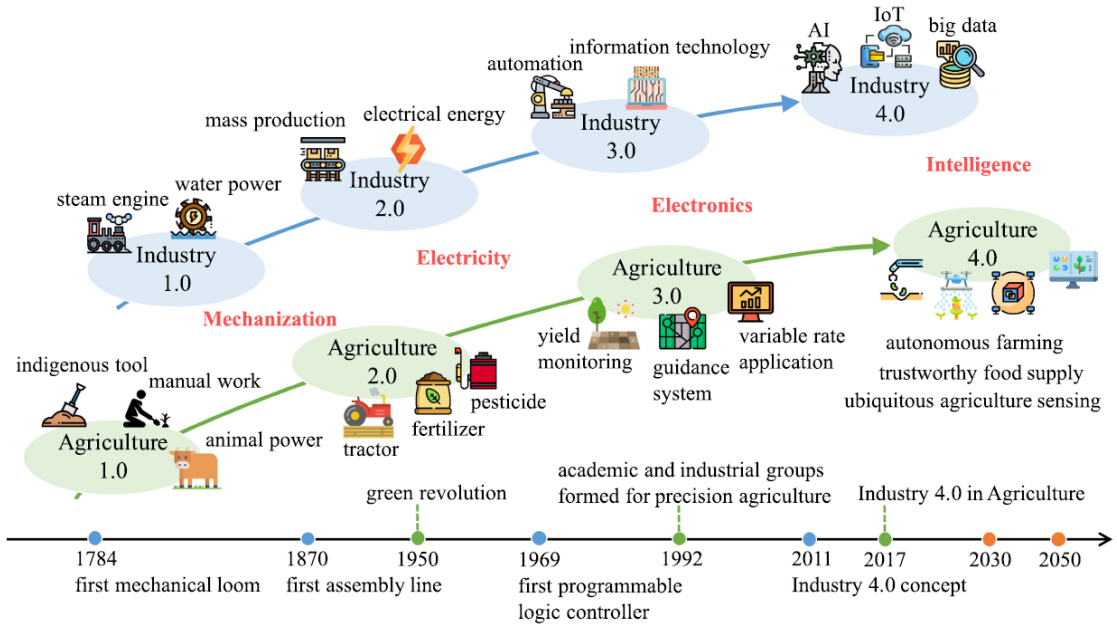


Figura 1 – Evolução das revoluções tecnológicas na Indústria e Agricultura. Fonte: [3].

Nos últimos 20 anos, um grande número de robôs têm sido desenvolvido com o propósito de ajudar nas tarefas diárias agrícolas. O seu *design* e autonomia tem vindo a evoluir cada vez mais. Um dos principais fatores que tem levado ao desenvolvimento das mais variadas plataformas robóticas e a navegação autónoma [4]. A navegação é uma característica importante no desenvolvimento do robô para a agricultura exterior justamente devido a uma grande área de ação, enquanto em agricultura interior as áreas de movimentação são mais reduzidas o que permite mais alternativas na escolha da tecnologia de navegação. Apesar dos grandes avanços realizados na área da automação robótica autónoma, a quantidade de projetos postos em funcionamento e comercializados na agricultura ainda é escassa. Isto em grande parte deve-se, a falta de investimento, visto que muitos agricultores ainda não estão preparados para realizar um investimento tão elevado. Outra questão importante é a segurança e a legislação, pois em alguns países, é exigido o uso de equipamentos de segurança durante o manuseamento de maquinaria pesada. E em relação a utilização de robôs autónomos ainda não existe nenhuma abordagem em relação as medidas de segurança, assim como questões éticas por parte dos governos o que diminui a adesão deste tipo de tecnologia. Contudo, espera-se que seja possível à medida que a tecnologia avance e os agricultores se familiarizem com os benefícios da mesma se consiga um maior investimento por parte dos agricultores e governos e desta forma haja um aumento na adesão a utilização de robôs autónomos criando competitividade empresarial.

A motivação para a realização deste projeto teve por base o desenvolvimento de um veículo autónomo de baixo custo, com vista a tornar mais atrativa a utilização de veículos robotizados na agricultura. A evolução da tecnologia é uma constante e a agricultura também precisa de evoluir, tornando-se mais eficiente e sustentável.

Garantindo por um lado, a qualidade dos produtos agrícolas e diminuição do impacto negativo nos solos com o uso de pesticidas e herbicidas, uma melhor gestão da água e por outro mitigar a falta de mão-de-obra deste setor principalmente em tarefas repetitivas. Com esta melhor gestão na agricultura também diminui o impacto na fauna. Basicamente adotar uma abordagem mais inteligente que já se tem vindo a aplicar, mas que com a robotização pode melhorar em muitos aspetos.

1.2 Objetivos

Este projeto tem como objetivo desenvolver uma plataforma agrícola robotizada de baixo custo, capaz de se movimentar autonomamente mediante um trajeto pré-definido.

A estrutura será projetada para ser robusta e flexível, permitindo a realização de várias tarefas em ambientes agrícolas e terá de ser economicamente acessível.

Para a navegação do robô, deverá ser desenvolvido um sistema de visão que utiliza uma câmara e bibliotecas públicas em linguagem *Python* com o objetivo de criar a melhor estratégia de navegação possível, considerando as condições de condução que a estrutura permitirá.

A principal função do robô estará em se movimentar em linha reta com o menor desvio possível para garantir que a plantação das árvores seja realizada com a distância adequada entre elas. Isso é crucial para o sucesso da plantação e otimização do uso de terra.

Em conclusão esta dissertação terá como objetivo o desenvolvimento de uma solução possível para um robô autónomo que seja economicamente acessível e eficiente em automatizar tarefas agrícolas, contribuindo para a melhoria da produtividade e redução de custos na agricultura.

1.3 Organização da Dissertação

Esta dissertação está estruturada em 6 capítulos onde se apresenta todo o desenvolvimento realizado neste projeto.

No capítulo 1 apresenta o contexto, a motivação, os objetivos e a organização da dissertação.

O capítulo 2 expõe a revisão bibliográfica da utilização e evolução do desenvolvimento de robôs autónomos na agricultura e a apresentação de tecnologias utilizadas para o desenvolvimento de sistemas de navegação.

O capítulo 3, menciona os materiais utilizados assim como as técnicas utilizadas no desenvolvimento do projeto.

O capítulo 4, apresenta a implementação do sistema de navegação desenvolvido.

O capítulo 5 apresenta os resultados obtidos e a discussão dos mesmos.

Por fim, o capítulo 6 relata as conclusões de uma análise ao trabalho realizado, assim como também as propostas de evolução futuras para o desenvolvimento do mesmo.

2. Revisão Bibliográfica

A automação tem marcado a sua posição na agricultura onde se destaca uma grande evolução, mas também um longo caminho a percorrer, nomeadamente nos métodos de sistemas de navegação. A atividade agrícola requer precisão de movimentos, proporcionando uma dificuldade na navegação autónoma, especialmente em ambiente exterior com grande área de trabalho e imprevisibilidade de inúmeros fatores ambientais e mesmo de obstáculos e orografia. O espaço de trabalho na agricultura é caracterizado por ser não estruturado quanto as suas características e impossível de controlar, ou seja, não é um ambiente estático, está em constante mudança. Ao longo da revisão bibliográfica iremos encontrar diferentes abordagens de sistemas de navegação idealizados e aplicados na agricultura quer em ambientes externos como internos na agricultura, assim quer diferentes abordagens no desenvolvimento dos seus sistemas de navegação noutras aplicações.

2.1 Plataformas robóticas desenvolvidas para a agricultura em espaços exteriores

Existem algumas plataformas robóticas desenvolvidas, industrialmente e outras em fase de I&D (Investigação e Desenvolvimento) para serem utilizadas na agricultura com funções diversificadas. Existem alguns fatores importantes a verificar na revisão bibliográfica referente a estas plataformas. A estrutura que deve ser simples, eficaz e adaptável ao ambiente em que trabalha, assim como nas tarefas que lhe forem propostas a realizar. A razão deve-se a variação topográfica e de cultivo agrícola, pois colher fruta é diferente de colher batatas. A escolha dos materiais e sensores, tem de ter o propósito de serem eficazes, adaptáveis e confiáveis num ambiente externo. Um aspeto importante para não resultar em avarias e maus funcionamentos aquando da sua utilização em diferentes tipos de ambientes [5]. Uma dificuldade que não ajuda ao desenvolvimento deste tipo de plataformas é o fator económico, pois muitos destes componentes que as integram são dispendiosos. O sistema de navegação é onde reside um dos maiores desafios a ser ultrapassado no desenvolvimento de uma plataforma robótica, como também o sistema de deteção, quer seja obstáculos ou matéria de cultivo. O desafio deve-se a escolha da tecnologia pois existem uma variedade de opções, mas consoante as necessidades do ambiente de trabalho é importante avaliar muito bem as vantagens e desvantagens de cada uma de forma a desenvolver um sistema de navegação eficiente e eficaz.

Um exemplo de constante evolução na agricultura é o desenvolvimento de adubos para acelerar o crescimento e produtos químicos para combater pragas infestantes. Mas nem tudo é positivo, pois o elevado uso destes produtos também prejudica o solo e a qualidade dos alimentos assim como a fauna. O uso intensivo gera problemas não só para o solo como também na economia dos agricultores.

Desenvolvendo um sistema, robotizado capaz de rentabilizar a aplicação destes produtos de uma forma mais eficiente e sustentável, faz com que se reduzam os gastos contribuindo-se para uma melhor utilização dos solos, minimizando a sua contaminação e saturação obtendo produtos agrícolas com melhor qualidade, isto é mais saudáveis (com menos contaminantes).

Outro exemplo de aplicação de robótica na agricultura é a sua utilização na colheita de tomate, morangos entre outros produtos agrícolas em que uma aplicação robótica fosse rentável por se tratar de um trabalho repetitivo e como sabemos trabalhadores são cada vez menos para este tipo de trabalho. Teremos então já projetos desenvolvidos em tarefas agrícolas como colheita, pulverização de químicos, controlo de ervas, monitoramento da plantação entre outras.

2.1.1 Tecnologias incorporadas em tratores usados em grandes campos Agrícolas

Existem atualmente tratores utilizados em grandes campos agrícolas, que incorporam sistemas de automação que elevam a produção agrícola de modo a esta ser mais precisa, com o intuito de retirar a maior eficiência possível na execução de tarefas realizadas.

A condução dos tratores e a tecnologia de sensores de direção já existe e tem evoluído ao longo do tempo. Este tipo de tecnologia possibilita um posicionamento com exatidão nas culturas de cultivo com pouca necessidade de controlo da direção por parte do operador. Reduz a fadiga, libertando a sua atenção para a monitorização de operações/manobras de alfaias acopladas por exemplo.

A tecnologia RTK-GPS (*Real Time Kinematic – Global Position System*) também já vem sendo aplicada nos tratores pela sua principal característica de possibilitar o histórico de rotas de movimentação do trator com rigor, reduzindo assim a possibilidade de passar duas vezes na mesma área onde já se semeou, pulverizou, fertilizou ou colheu. A Figura 2 é uma ilustração das comunicações que se realizam para o funcionamento desta tecnologia e um exemplo de como numa grande plantação onde é necessário ter mais do que um trator a operar consegue-se evitar como já foi dito passagem duas vezes na mesma área de terreno, evitando assim perda de tempo e poupança nos custos [6].

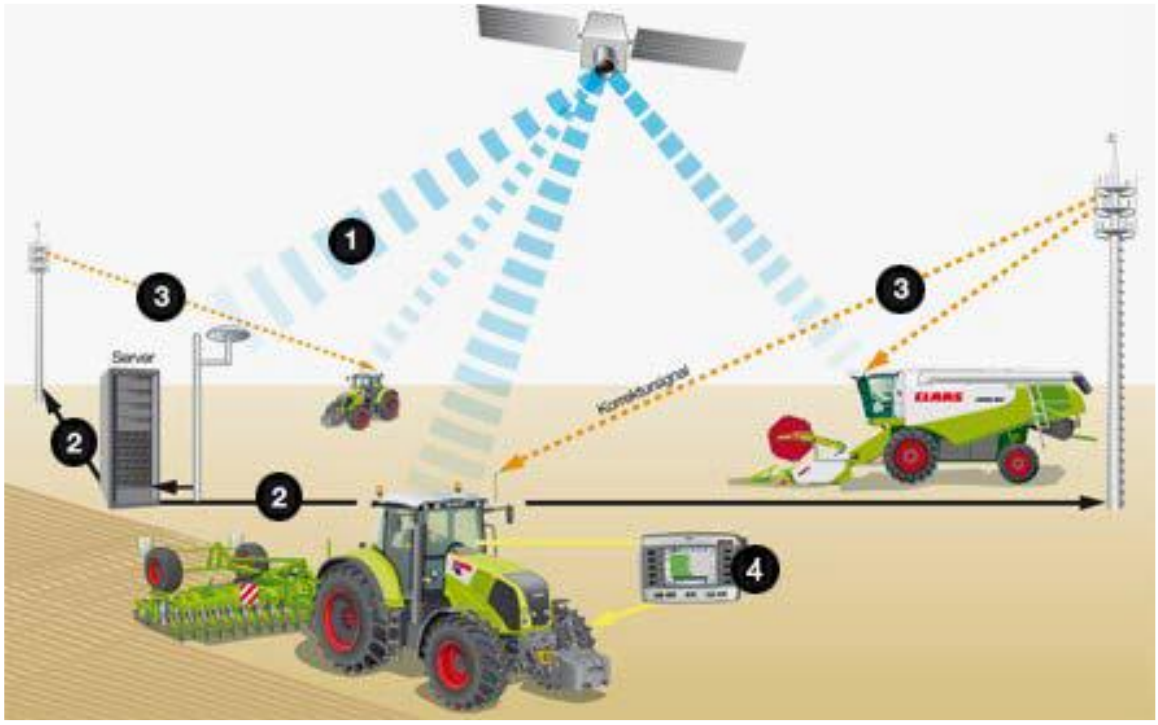


Figura 2 – Ilustração das comunicações realizadas pelo sistema RTK-GPS. Fonte: [6].

Marcas como a *John Deere*, *New Holland's*, *Deutz-Fahr* entre outras têm evoluído, desenvolvendo os seus próprios softwares ou então comercializam produtos de outras empresas que desenvolvam este tipo de sistemas de navegação. A comunicação máquina com máquina envolve uma conexão direta via rádio sendo utilizada com o intuito de melhorar a logística e eficiência do trabalho com numerosos tratores em funcionamento. Esta comunicação permite que se partilhe mapas da área de terreno que já foi percorrida e linhas de navegação para a realização de cultivo, colheita ou fertilização [7].

a) b)

Figura 3 - a) Visão ao carregar; b) Visão do ponto cego do trator. Fonte: [8].

Na Figura 3, ilustram exemplos de utilização das câmaras nos tratores ou máquinas agrícolas. São situações diferentes, a imagem **3.a)** ilustra um momento de colheita e de visualização do atrelado para desta forma o operador poder ver e realizar um carregamento total do mesmo e saber quando este está cheio, já na imagem **3.b)** podemos ver um ponto cego do trator, que alinhado com a tecnologia incorporada na transmissão do trator se uma pessoa se encontrar naquele ponto o trator impede o operador de realizar movimento prevenindo um acidente.

2.1.2 Plataformas robóticas comercialmente disponíveis e em fase de teste

Nas diversas plataformas robóticas desenvolvidas, verifica-se que o sistema de navegação RTK-GPS é o mais utilizado, devido à sua fidedignidade na determinação da localização. Este sistema equipa a maioria das plataformas robotizadas utilizadas, para determinar a localização e definição de rotas em trabalhos a realizar em grandes campos agrícolas. Este sistema é muito utilizado também para realizar a cartografia da plantação. Na área da deteção de obstáculos é onde existe maior diversidade na escolha de sensores, nomeadamente Visão, LIDAR (*Light Detection and Ranging*), Sonar, Radar entre outros, estes são bastantes versáteis e podem ser utilizados para diversas tarefas. Por exemplo, com a visão computacional também é possível realizar a navegação do robô/plataforma numa situação de navegar, por exemplo em vinhas. O robô através da visão consegue analisar os obstáculos e plantas e neste caso específico ao detetar as videiras este consegue se guiar pela carreira formada pelas mesmas. A visão é também muito popular para utilização específica de identificação de objetos, neste caso específico de plantas, em que esta aplicação tem maior utilização em tarefas de colheita.

Existem configurações com associação entre sensores tendo como objetivo de o robô conseguir desenvolver tarefas eficazmente em diversos tipos de ambientes. Pois, um dos principais objetivos é conseguir ter uma plataforma robótica flexível capaz de desempenhar múltiplas tarefas e que apresente um baixo custo de aquisição, pois o que impede a ampla disseminação de plataformas robóticas na agricultura é o preço elevado associado a estes sistemas. Outra das razões é o facto de as plataformas existentes não serem flexíveis para a execução de inúmeras tarefas e usarem diferentes alfaias.

2.1.2.1 Exemplos de robôs autónomos para a realização de colheita

Na Figura 4, encontra-se uma máquina de colheita de cereais equipada com RTK-GPS, juntamente com sensores IMU (*Inertial Measurement Unit*), com o intuito de recolher informação sobre a máquina quanto ao seu posicionamento, sendo o seu controlo feito através de CAN (*Controller Area Network protocol*), permitindo um controlo mais suave. Esta máquina ainda permite o seu funcionamento em dois modos, o manual e o automático. O manual será quando um operador fica responsável pelo seu controlo e o automático será quando estará a funcionar autonomamente controlada por um computador.

Figura 4 – Ceifeira equipada para condução autónoma. Fonte: [9].

Em colheita de horticultura, fruta ou vegetais é necessário proceder a uma avaliação no momento. Por exemplo no caso da maçã no momento da colheita é necessário perceber quais maçãs se encontram em condições de serem colhidas. No caso das hortaliças ou dos vegetais passa-se o mesmo procedimento e como se pode entender em culturas de grande cultivo é necessária numerosa mão-de-obra. A investigação e desenvolvimento de plataformas robóticas para a colheita de fruta existe desde 1980, porém ainda existe bastante investigação de forma a encontrar uma plataforma economicamente apetecível.

Figura 5 – Plataforma robótica a colher maçãs. Fonte: [10].

A Figura 5 é um exemplo de uma plataforma desenvolvida para a colheita de maçã. Este tipo de trabalho agrícola para ser robotizado requer um maior desenvolvimento, especialmente no desenvolvimento da estrutura da plataforma, pois necessita de um instrumento para realizar a colheita. No exemplo demonstrado na figura é apresentada uma plataforma com um braço robótico para colher a maçã e uma zona de descarga para colocar as maçãs colhidas. É uma plataforma robótica de lagartas que contém baterias e sensores. Como sistema de navegação autónomo é utilizado o GPS (*Global Position System*), tendo uma velocidade típica muito baixa [10].

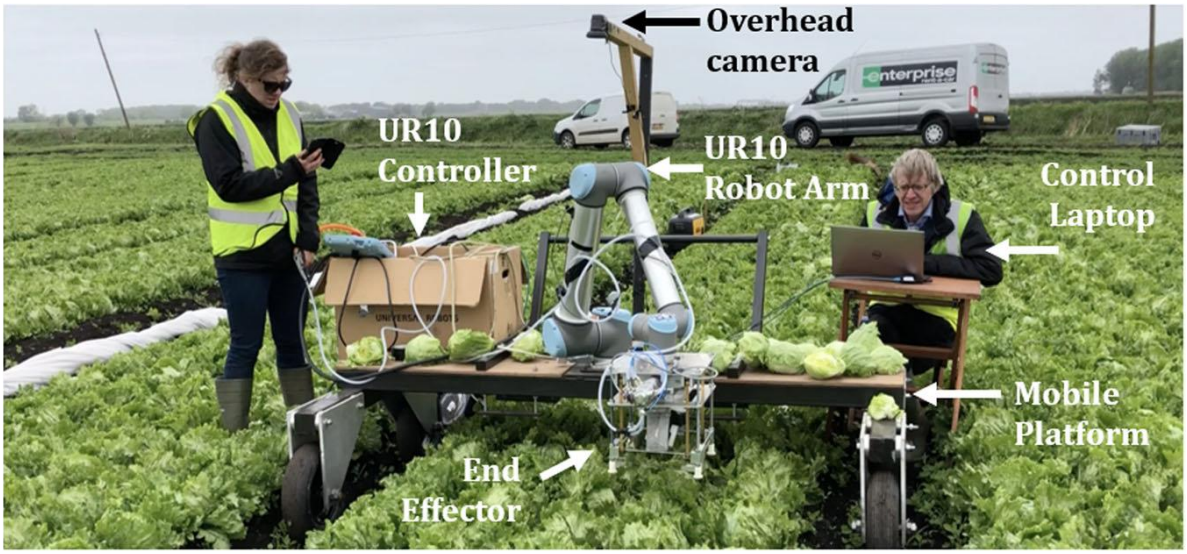


Figura 6 – Plataforma robótica a realizar testes. Fonte: [11].

Por último também se pode visualizar na Figura 6 um robô para a colheita de hortaliças em fase de teste. Neste contexto da colheita estamo-nos a focar em campos abertos, por isso mesmo é feita referência à colheita de frutas ou de couves e até de tomates. As plataformas agrícolas desenvolvidas para espaços fechados têm outras condições e realizam outro tipo de colheitas, normalmente exploradas neste tipo de espaços.

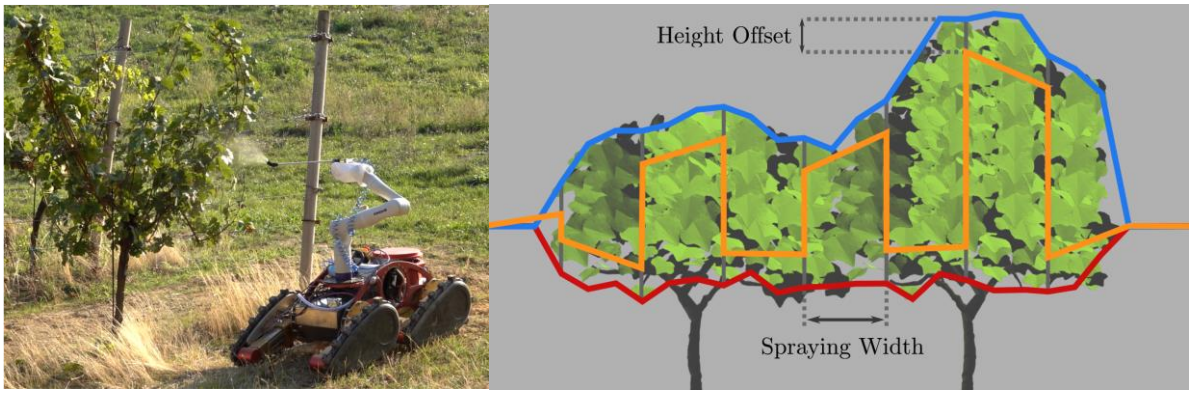
2.1.2.2 Exemplos de robôs autónomos para a realização de pulverização e controlo de ervas daninhas

A pulverização é um procedimento bastante importante na agricultura, pois a realização de uma boa colheita ou a perda da mesma pode muitas vezes depender da aplicação de produtos pesticidas e ou fitoquímicos. Esta é então uma tarefa comum realizada na agricultura que contudo contém as suas desvantagens, pois apesar deste tipo de produtos serem eficientes deixam resíduos no solo que reduzem a sua fertilidade e a biodiversidade [12].

Figura 7 – Protótipo de robô para aplicação de pulverização em tempo real. Fonte: [13].

Na Figura 7, demonstra um trator modificado para ser autónomo com o acoplamento de um sistema de pulverização, este também desenvolvido para ser inteligente. Como sistema de navegação trata-se de mais um caso em que se utiliza o sistema RTK-GPS para a realização da navegação do robô, assim como também a recolha de informação dos locais já pulverizados com o intuito de evitar a repetição, como já foi referido no caso da colheita. Como tecnologias incorporadas temos o laser visando detetar objetos que possam aparecer no caminho.

O sistema inteligente de pulverização dispõe da visão como meio de avaliar em tempo real a situação. Esta é processada realizando uma análise em tempo real para ordenar a pulverização no local correto. Este é composto por 12 bicos pulverizadores em que cada um possui uma válvula solenoide, para possibilitar que cada uma possa ser acionada individualmente [13]. No caso dos pesticidas aplicam-se somente na plantação, caso fosse a aplicação de herbicidas o foco seria somente nas ervas. Assim a aplicação do herbicida é realizada de forma mais precisa e focada, retirando benefícios económicos e ambientais com este método de aplicação autónomo e inteligente.



a) b)

Figura 8 – a) Robô a pulverizar videiras; b) Movimentação do pulverizador. Fonte:[14].

O tratamento das vinhas é importante para que a produção seja rentável, pois previnem-se perdas contra doenças e pestes. Um dos métodos de tratamento é a utilização de químicos aplicados através da pulverização. Existem estudos e avaliações realizadas com o intuito de obter conhecimento sobre a influência da quantidade de químicos para realizar um controlo eficaz de pragas e doenças, minimizando assim a sua utilização e realizando esta de forma mais inteligente [15]. Na Figura 8, podemos uma plataforma robótica autónoma a realizar pulverização em vinhas. Equipada estruturalmente com um braço robótico e uma plataforma de movimentação todo-o-terreno com 4 lagartas controladas individualmente, na navegação temos LIDAR, câmara estereoscópica e GPS [16].

A incorporação do braço robótico equipado com um pulverizador permite uma maior flexibilidade e precisão relativamente ao momento de pulverização com a movimentação da plataforma. Pela movimentação que o braço robótico permite que se elabore algoritmos com o propósito de o braço robótico atuar de forma precisa sobre a área de real interesse a pulverizar. Na Figura 8, temos a linha laranja que demarca a movimentação do braço robótico a pulverizar dentro da área limitada pelas linhas azul e vermelha.

2.1.2.3 Outras plataformas robóticas com aplicação em recolha de informação fenotípica

A informação fenotípica é atualmente muito importante para a agricultura, esta consiste na atividade de aquisição de informação sobre as características observáveis da planta, como o seu desenvolvimento, a sua aparência. Basicamente reúne informação das plantas relativamente a reação das mesmas com o meio ambiente envolvente. Esta informação é fundamental para a evolução, permitindo a seleção das espécies com as características pretendidas, ajudando assim a melhorar a produção agrícola, a prevenir doenças e a identificar as plantas que melhor se adaptam as alterações climatéricas.

Os robôs autónomos implementados para a execução desta tarefa são importantes, pois trata-se de uma tarefa repetitiva e até aborrecida, podendo se tornar difícil devido às condições ambientais a que se está sujeito para a sua realização. O robô não se queixa e pode trabalhar sobre condições climatéricas desconfortáveis para o ser o humano como calor e frio. Este traz como vantagens o facto de poder processar uma maior quantidade de aquisições de dados, tendo capacidade de recolher mais precisamente devido aos sensores equipados. Resumidamente, a sua utilização para aquisição de informação fenotípica leva a que a segurança e qualidade dos dados adquiridos seja de maior confiança e eficiência em comparação com um trabalhador.

Figura 9 – Plataforma robótica com sensor 3D LIDAR. Fonte: [17].

Na Figura 9, apresentamos uma plataforma agrícola experimental com o propósito de realizar reconhecimento 3D das plantas através do sensor LIDAR e recolha de informação fenotípica. Trata-se de um robô equipado com sensor RTK-GPS e sensor IMU para poder realizar a sua orientação de forma autónoma. Outra das capacidades do sensor LIDAR consiste em permitir que se consiga realizar uma avaliação ao terreno detetando buracos e irregularidades do mesmo [17].

2.2 Automação e robótica em ambientes interiores-estufas

Os desenvolvimentos de plataformas robóticas autónomas têm-se expandido bastante na agricultura como já foi mencionado anteriormente com alguns exemplos de plataformas existentes para um ambiente exterior. No caso da agricultura em estufas, como se trata de um ambiente mais controlado, permite que exista mais alternativas disponíveis no desenvolvimento de sistemas de navegação. Uma das razões prende-se com o facto de as áreas de cultivo serem menor escala e não estarem tão sujeitas a fatores ambientais não controláveis, permitindo o desenvolvimento de sistemas mais estandardizados.

Existem várias tarefas árduas que são necessárias realizar numa estufa, as quais exigem o recurso a mão-de-obra para as executar. Como vem sendo dito, automatizar é uma inevitabilidade necessária de mudar esse paradigma, visto que trabalhadores neste setor são cada vez mais escassos. Então existem tarefas a serem realizadas por robôs como, por exemplo, a pulverização, a colheita, a recolha de informação das condições da plantação, entre outras [18].

A pulverização de químicos é um trabalho bastante perigoso, pois é executado num local fechado com humidade, elevada temperatura e pouca circulação de ar, e por isso mesmo a automatização vem diminuir o risco de saúde dos trabalhadores agrícolas [19].

Como soluções para sistemas de navegação para este tipo de situação em espaços fechados temos sistemas de rádio frequência, taqueometria, UWB (*Ultra Wind Band*), magnéticos e de acústicos, tendo todos precisões a rondar o centímetro. Como cada sistema tem os seus prós e contras será necessário um estudo de modo a identificar o que melhor se enquadra com a situação.

O ideal seria ter uma plataforma robótica capaz realizar várias tarefas, sendo somente necessário mudar de ferramenta/alfaia, mas ainda não foi encontrada uma solução economicamente atraente. Graças à medição em tempo real das grandezas monitorizadas, o operador agrícola poderá ser avisado de uma eventual falha ou erro no sistema, permitindo que este tome as medidas necessárias sem que a tarefa realizada seja prejudicada.

2.2.1 Sistemas de navegação desenvolvidos para estufas

Existem sistemas de navegação desenvolvidos e inspirados nos AGV (*Automated Guided Vehicle*) da indústria, em que estes se guiam por uma linha colocada no chão das fábricas. Na implementação de robôs em estufas baseados no sistema AGV, existem métodos como por exemplo colocar tubos de inox enterrados com o mesmo intuito que as linhas nas fábricas, neste caso utilizam sensores indutivos para detetar o tubo. Das desvantagens ao utilizar este tipo de métodos de navegação, temos o custo que acarreta na aquisição, a sua complexidade de construção e a manutenção necessária.

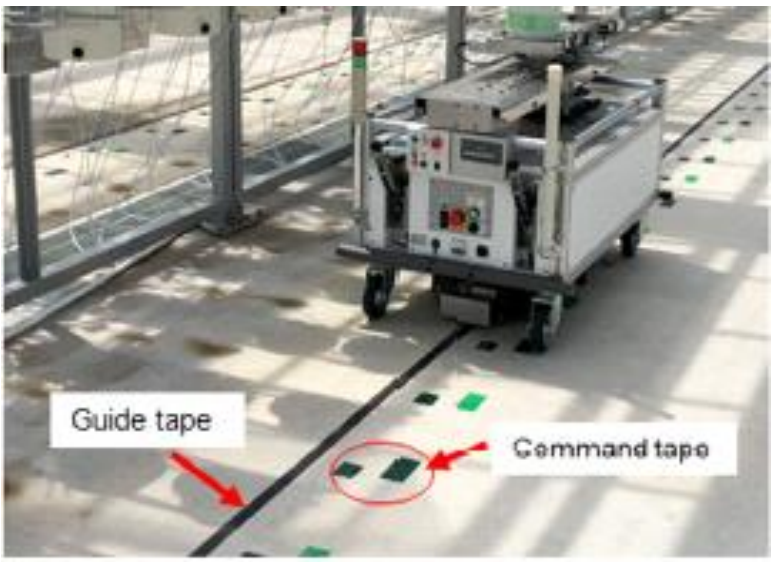


Figura 10 – Linhas de comando e guia de navegação. Fonte: [20].

A Figura 10, ilustra um exemplo de aplicação em estufas, da tecnologia AGV, no transporte de vasos de plantas [20].

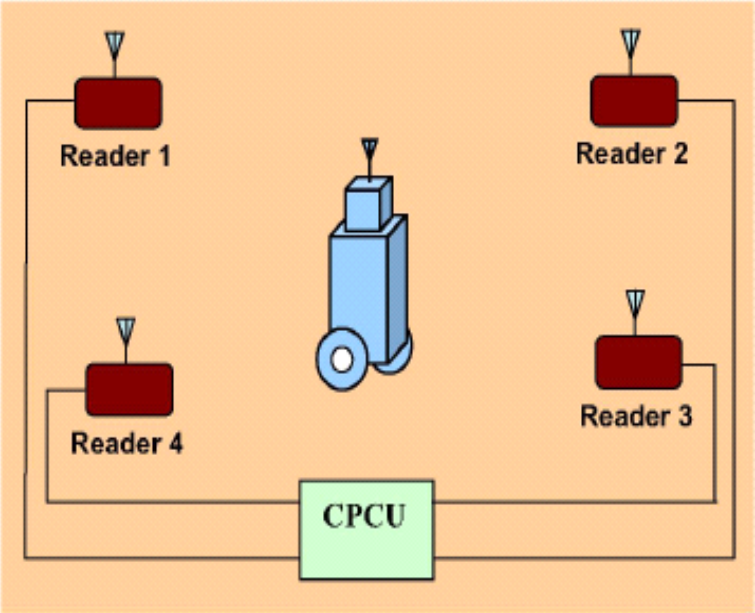


Figura 11 – Sistema de localização UWB. Fonte: [21].

Podemos ver na Figura 11, uma ilustração simples de como o sistema de navegação UWB pode ser outra tecnologia a utilizar em estufas. Este sistema tem uma precisão elevada e funciona através da colocação de recetores na área de movimentação do robô, isto é a área interna da estufa. Estes recetores estarão todos ligados a uma base que processa os sinais recebidos pelo emissor que será o robô em movimento, e no fim irá enviar sinais para este saber em que posição se encontra.

A comunicação entre os *readers* e o robô consiste em envio de pulsos de rádio UWB.[21]. O algoritmo mais implementado neste tipo de situação é o TDOA (*Time Difference Of Arrival*) pois é o que consegue obter maior precisão em tempo real, contudo existem outros algoritmos disponíveis [22].

Como vantagens este sistema apresenta um consumo reduzido de energia e permite que o sinal emitido transponha facilmente obstáculos, isto é, não interfere com outros sinais de rádio e é resistente a ruído e interferências de que poderá ser alvo. No entanto existem alguns problemas nesta tecnologia como por exemplo o facto de causar interferência noutros meios de comunicação, nomeadamente dispositivos *wireless*. A hipótese de poder interferir com outros sistemas é prejudicial, pois as estufas automatizadas utilizam equipamentos de comunicação para o controlo de outros equipamentos.

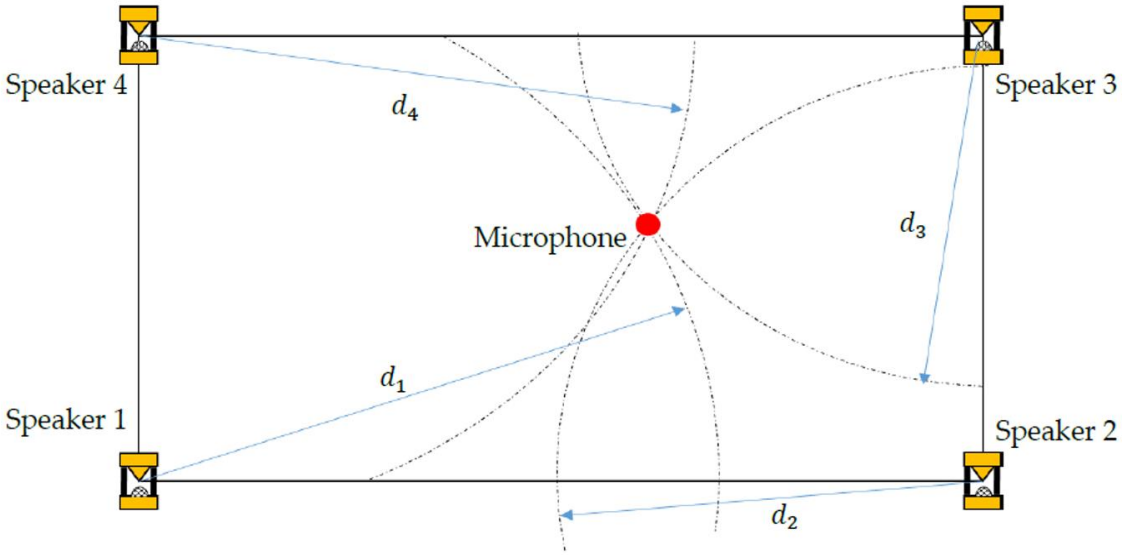


Figura 12 – Sistema de navegação através do som. Fonte: [23].

O sistema de navegação através do som apresenta bastantes vantagens de utilização relativamente aos outros sistemas já mencionados. Este funciona através do som e tem bastante semelhança com o sistema já mencionado UWB, apresenta também precisão e é atrativo economicamente. A sua implementação consiste na colocação de um microfone no robô e no posicionamento de 4 colunas de som. Estas devem estar estrategicamente colocadas com o intuito de formar a área pretendida para a movimentação do robô, a Figura 12 ilustra uma possível configuração [23]. O microfone instalado no robô recebe o som emitido pelas colunas, que com o método de processamento correto consegue indicar onde se encontra dentro da área delimitada sendo capaz de ter um erro inferior a um centímetro. Na agricultura em espaços exteriores esta tecnologia de navegação não é viável devido ao facto de estar limitada em termos de área de cobertura.

2.3 Principais Tecnologias utilizadas nas diversas plataformas para a navegação exterior

A agricultura tem vindo a beneficiar significativamente com o desenvolvimento da tecnologia de navegação, disponibilizando atualmente uma grande variedade de tecnologias com aplicação na agricultura. Neste projeto, que consiste na conceção de um robô autónomo com aplicação na agricultura, foi necessário realizar um estudo sobre as tecnologias existentes. É importante notar que cada tecnologia apresenta vantagens e desvantagens e, portanto, a junção delas permite a realização de um robô autónomo mais eficiente, pois é possível tirar o melhor de cada uma. Por exemplo, a visão pode ser usada para a identificação de objetos, o LIDAR para deteção de obstáculos e o RTK-GPS para a determinação da localização em tempo real e a criação de rotas para percorrer.

Assim, nos subcapítulos seguintes, serão apresentadas as tecnologias mais utilizadas para implementação em condução autónoma na agricultura.

2.3.1 Visão

Os sistemas de visão são os mais utilizados e antigos, devido principalmente ao seu custo benefício, daí serem a primeira opção a ser pensada para realizar navegação autónoma quando se tem um baixo orçamento. Na agricultura temos várias tarefas de aplicação em que a visão tem vindo a ser aplicada para melhorar a sua eficiência de execução, como por exemplo a colheita, pulverização, cultivo e eliminação de ervas daninhas [24]. No caso de eliminar as ervas de uma cultura de milho, é um cenário onde se pode utilizar mais do que um sistema de visão com diferentes funções. Em questão temos um sistema de visão para se guiar pelas carreiras de milho, por exemplo, e outro para detetar as ervas e dar ordem para a eliminação das mesmas, tendo o poder de processamento para distinguir o milho das ervas [25].

Figura 13 – Visualização das linhas desenvolvidas pela visão computacional. Fonte: [26].

Na Figura 13 podemos visualizar a marcação das linhas de forma a determinar o centro entre as carreiras de milho. Estas são desenvolvidas por um programa para o robô conseguir realizar a movimentação e ter assim uma referência para se guiar durante a sua movimentação. A visão é um exemplo claro de como tecnologia simples e primordial referentemente a evolução e desenvolvimento de robôs autónomos na agricultura. Atualmente os tratores mais equipados já incorporam sistemas de visão, tanto para a ajuda ao operador na execução de determinadas atividades, como assegurar a segurança dos trabalhadores monitorando zonas mortas para o operador [8].

A visão permite-nos obter informação em parâmetros de cor, posição, tamanho, forma e textura, apesar de em muitos parâmetros poderá não sempre ser fiável, contudo, permite que seja uma boa ferramenta de trabalho na agricultura de precisão [27].

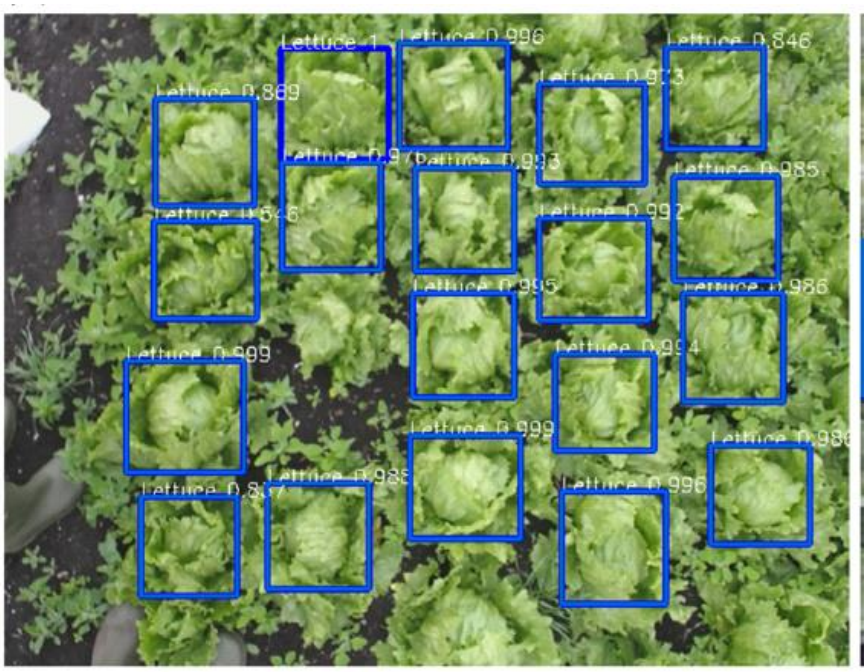


Figura 14 – Ilustração da deteção das alfaces pela visão computacional. Fonte: [11].

Podemos afirmar que a visão computacional em tarefas como colheita e pulverização tem bastante desenvolvimento em métodos e algoritmos. A Figura 14, exemplifica um teste realizado validando a visão quanto à forma capaz de detetar e classificar as alfaces a colher. Este sistema desenvolvido permite classificar as alfaces em quatro aspetos distintos: em crescimento, pronta para ser colhida, infetada e quando não tem alface [11]. Apesar da fiabilidade que a visão nos entrega existem alguns fatores ambientais entre outros, que dificultam a sua utilização como a luminosidade que influência o correto funcionamento.

2.3.2 LIDAR

A tecnologia *LIDAR* tem vantagens significativas na aplicação em agricultura, pois não é afetada pelas mudanças de luminosidade do ambiente. No entanto o seu alto custo tem sido uma barreira para a sua adoção generalizada. A redução de custos ao longo dos anos aumentou o interesse e permitiu mais avanços em sua utilização em ambiente agrícola [28]. O *LIDAR* funciona com base no princípio TOF (*Time Of Flight*), que envolve o envio de pulsos de luz de um laser e a medição do tempo decorrido desde a sua emissão até a deteção da sua reflexão. Essas medições permitem a criação de uma representação em 3D do ambiente. Os dispositivos mecânicos complexos de vidros que oferecem uma visão de 360º podem ter custos exorbitantes, o que torna o desenvolvimento de sensores *LIDAR* de baixo custo uma necessidade, como os *solid-state LIDAR* e *infrared LIDAR* [29].

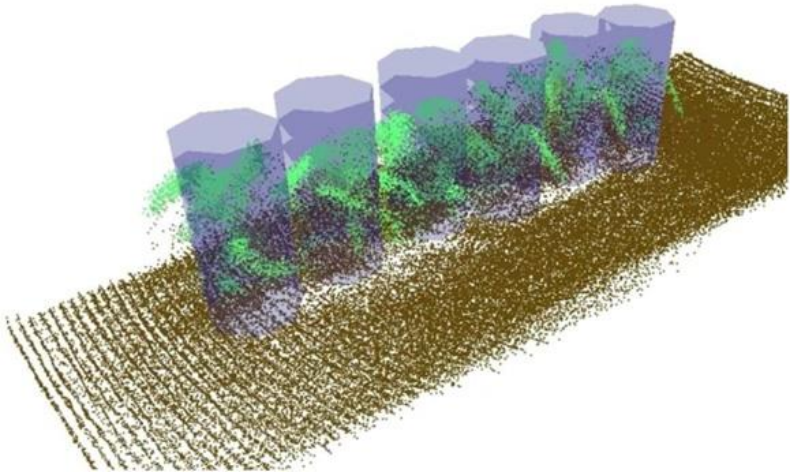


Figura 15 – Ilustração do milho em 3D, através de informação recolhida pelo sensor LIDAR. Fonte: [17].

Este sensor é utilizado para medir distâncias, mapear áreas e detetar e evitar obstáculos. Graças a sua elevada precisão, essa tecnologia torna-se bastante utilizada na agricultura, especialmente para sistemas de orientação em vinhas permitindo a criação de mapas precisos das plantações [30]. A Figura 15 ilustra uma leitura realizada pelo LIDAR, que gerou uma imagem artificial de uma carreira de milho. Esse é um exemplo de deteção e mapeamento de plantas para auxiliar a navegação de robôs agrícolas [17]. Observa-se que, no estudo desta tecnologia, tem substituído, em certa medida, a utilização da visão computacional, devido às suas características que eliminam muitos problemas, sendo o principal a sensibilidade à luminosidade, que não afeta em nada o seu funcionamento.

2.3.3 RTK-GPS

Ao longo dos anos a agricultura tem evoluído consideravelmente graças ao controlo realizado nas tarefas agrícolas com informação geográfica. O método RTK-GPS é muito utilizado atualmente para a navegação e cartografia das máquinas agrícolas. Devido à sua precisão, que pode chegar a centímetros, enquanto o GPS convencional trabalha na ordem dos metros, a maioria dos robôs autónomos desenvolvidos para a agricultura exterior utilizam a tecnologia RTK-GPS [31].

Atualmente, o método é usado de forma extensiva na agricultura, com as máquinas agrícolas a virem já com o equipamento e *software*. A *John Deere* tem o seu próprio sistema de navegação deste tipo, que é totalmente restritivo em termos de funcionamento dos módulos e *software*. Um recetor RTK-GPS da *John Deere* só funciona na rede de produtos da marca, sendo restrito na comunicação com outros aparelhos que não sejam da marca [32].

Esta técnica tem como base os princípios DGNSS, em que temos uma *base station* como referência que envia as correções da posição para o *rover*. O *rover* no caso da Figura 16 é um carro, mas dependendo da situação pode ser um robô, mota, bicicleta até pessoa.

Para que realize um bom funcionamento, a distância entre o *rover* e a *base station* tem de estar entre os 10 e 20km. Caso contrário, a precisão diminui à medida que a distância entre eles aumenta [33].

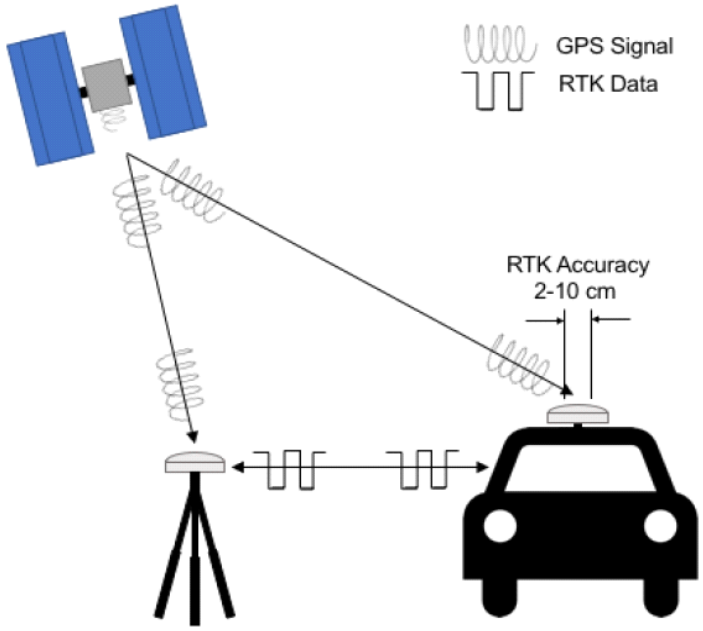


Figura 16 – Sistema RTK-GPS. Fonte: [34].

Na Figura 16, é apresentada uma pequena ilustração do método RTK-GPS. Este método requer dois recetores, um que funciona como *base station* e outro como *rover*. O primeiro é um ponto fixo cujas coordenadas são conhecidas com exatidão, enquanto o segundo é móvel [34].

Em constante comunicação com pelo menos cinco satélites, o funcionamento consiste na mediação do ângulo da fase portadora. Dessa forma, a distância entre um recetor e os satélites é medida através de um conjunto desconhecido de ondas inteiras, designando como *integer ambiguity*, juntamente com o comprimento da onda fracionária, que é dado pelo ângulo de fase. O sistema precisa de resolver a ambiguidade existente das medições entre o recetor e os satélites para se poder obter uma medição precisa do local por meio de otimização [35].

Ao utilizar este método, é importante ter em atenção o ambiente em que os recetores se encontram, o sinal pode ter dificuldades em penetrar em alguns materiais, o que pode afetar significativamente a precisão do RTK-GPS.

É importante mencionar que os recetores podem ser de frequência única ou dupla. Um dos fatores a ser considerado é a atmosfera, que gera um atraso de propagação do sinal. Portanto, os recetores de dupla frequência são capazes de eliminar o efeito da atmosfera, enquanto os de frequência única são incapazes de o fazer [36].

3. Materiais e Métodos

3.1 Materiais

3.1.1Estrutura do robô e motorização

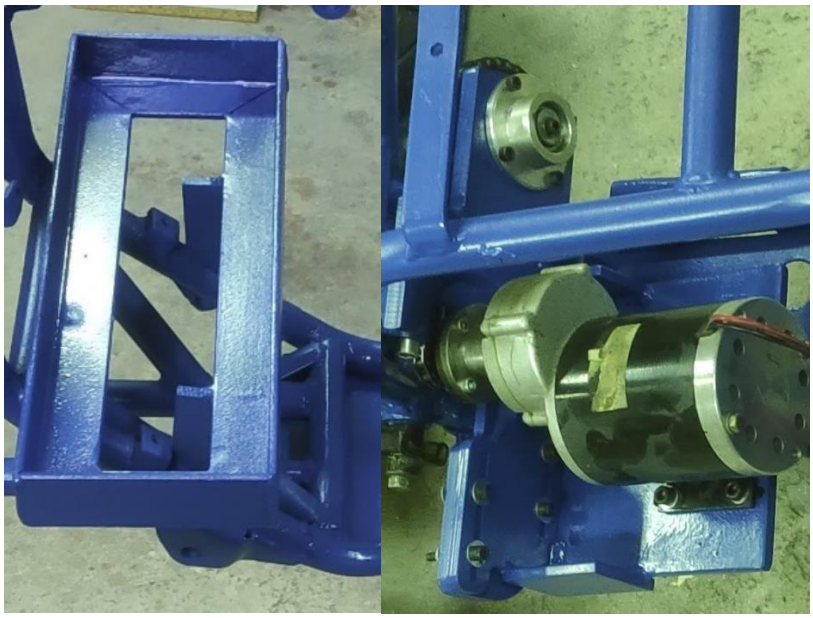
Este robô deve ser capaz de se mover autonomamente com uma estrutura versátil e de acomodar facilmente ferramentas para diversas tarefas, e estar preparado para melhorias futuras. A plataforma robótica final utilizada encontra-se na Figura 17.

Figura 17 – Plataforma robótica utilizada em desenvolvimento da dissertação

Para o desenvolvimento do projeto foi utilizada uma minimoto-quatro, em que foi totalmente desmontada ficando somente com a estrutura metálica e as rodas. Inicialmente tinha uma estrutura padrão de uma moto a combustão e efetuaram-se alterações mecânicas para que esta deixasse de trabalhar a combustão e passasse a eletricidade. Foi então realizado um desenho mecânico por uma pessoa especializada para a montagem e criação das peças que foram necessárias aplicar, e que se vão apresentar de seguida. A Tabela 1 apresenta algumas das características técnicas da estrutura do robô.

Tabela 1 – Especificações técnicas da estrutura desenvolvida.

Especificação do Robô	
Comprimento	150 cm
Largura	89 cm
Altura	80 cm
Peso aproximado	50 kg



a) b)

Figura 18 – a) Suporte das Baterias; b) Suporte do motor com redutor e montado

Na Figura 18, podemos visualizar duas imagens a ***a)***, que representa o suporte idealizado para duas baterias e a ***b)***, que mostra o suporte feito para a montagem do motor juntamente com o mesmo.

Figura 19 – Relação de 3:1 implementada.

Mesmo já existindo uma redução interna no motor de 6.67:1, houve necessidade de implementar uma redução de 3:1 aproximadamente, tendo assim no final das contas uma redução de 20:1 com o objetivo de reduzir a velocidade e em contrapartida aumentar o binário com o intuito de evitar o escorregamento por parte das rodas de tração. A redução implementada encontra-se na Figura 19 consiste na utilização de duas rodas dentadas de diferentes diâmetros. A de menor pertence ao eixo de transmissão do motor elétrico e a outra ao veio das rodas traseiras interligados por uma corrente.

Esta alteração teve o propósito de tentar implementar o maior binário possível na tração das rodas e assim evitar o escorregamento, permitindo que se consiga uma movimentação mais controlada, principalmente no momento de mudança de direção.

Tabela 2 - Especificações técnicas do motor.

Motor	
Tensão	24 V
Potência nominal	600 W
Relação de redução	6.67:1
Velocidade nominal	3200 rpm
Peso	5.3 Kg

Na Tabela 2 encontra-se as características técnicas do motor utilizado para fazer movimentar o robô. Trata-se, portanto, de um motor elétrico de 24V, com uma potência de 600W, tendo assim potência suficiente para fazer conseguir movimentar a plataforma e todo o seu peso.

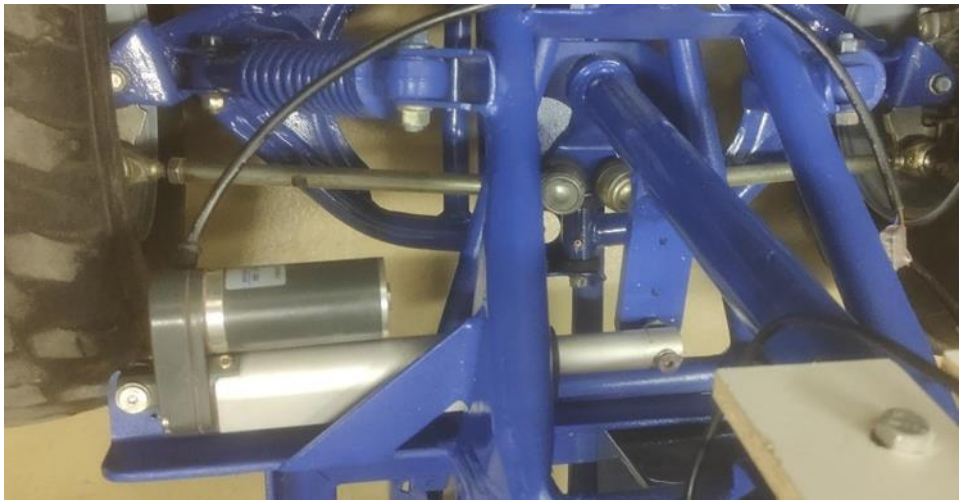


Figura 20 – Sistema de direção implementado com suporte e motor utilizado.

O volante foi eliminado para se conseguir aplicar um motor linear que controla a direção eletricamente. Tratasse de um motor de 12V com capacidade de força de 750N o que permite que seja capaz de fazer rodar as rodas na pior situação que é quando o robô se encontra parado. Na Figura 20 demonstra como se encontra a colocação do motor e o suporte para o mesmo. Tendo este um comprimento de movimentação de 55mm, e encontrando-se no seu máximo é a posição do robô com as rodas estão totalmente viradas para a esquerda. No meio do percurso, ou seja, com 25mm as rodas da frente estão alinhadas com as rodas traseiras e por último quando este faz o total recuo e encontra-se na posição zero, tem as rodas totalmente viradas para a direita.

a) b)

Figura 21 – Relés de 24V e 12V utilizados para o controlo dos motores.

Para controlar os motores do robô, foram utilizados dois relés para o motor de tração e outros dois para o motor que controla a direção do robô. Os relés têm como objetivo ligar/desligar os motores seguindo as ordens de comando emitidas pelo *Raspberry Pi*. A necessidade de ter dois relés para cada motor deve-se ao fato ser preciso mudar a direção de funcionamento. Em termos mais simples, com os dois relés em controlo do motor, é possível que o robô rode no sentido horário e anti-horário. O mesmo acontece com o motor da direção, a utilização dos dois permite movimentar as rodas para esquerda e para a direita. Na Figura 21 apresenta-se os relés utilizados, em que ***a)*** representa os relés de 24V para o motor de tração e ***b)*** de 12V para o motor da direção.

3.1.2 *Raspberry Pi* e câmara utilizada

O *Raspberry Pi* é considerado um computador de reduzidas dimensões e de baixo custo, permitindo a realização de projetos escolares e industriais bastante interessantes, graças ao facto de ser acessível e versátil na sua capacidade de computação. Tornando-se assim bastante popular no mundo da tecnologia para o desenvolvimento de projetos.

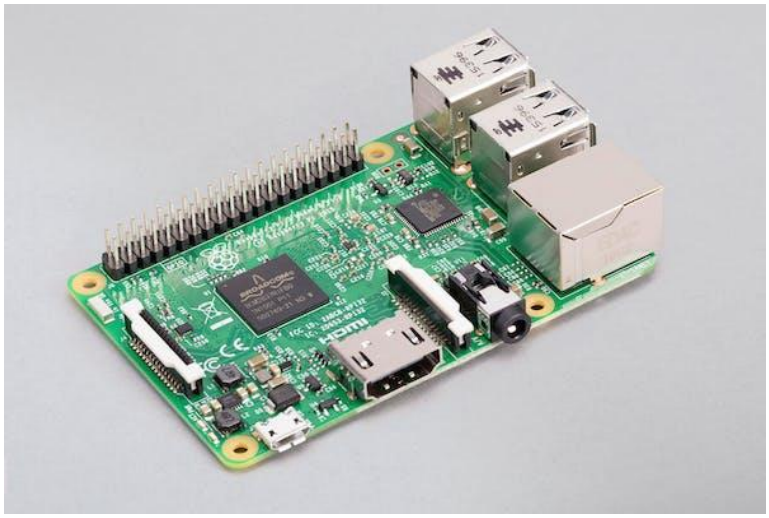


Figura 22 – *Raspberry Pi 3 B+* utilizado para o controlo do robô.

Uma das razões pelo facto de se utilizar em projetos de robótica é a sua capacidade de interface de entradas e saídas GPIO (*General Purpose Inpu/Output*), que permite ligar a sensores, motores e outros dispositivos eletrónicos e realize o seu controlo. A sua versatilidade no suporte a várias linguagens de programação permite aos criadores de projetos terem uma boa diversidade na sua escolha.

Tabela 3 – Especificações técnicas do *Raspberry Pi 3 B+*.

SoC	Broadcom BCM2837BO quad-core A53 (ARMv8) 64-bit @ 1.4GHz
GPU	Broadcom Videocore-IV
RAM	1GB LPDDR2 SDRAM
Networking	Gigabit Ethernet (via USB channel), 2.4GHz and 5GHz 802.11b/g/n/ac Wi-Fi
Bluetooth	Bluetooth 4.2, Bluetooth Low Energy (BLE)
Storage	Micro-SD
GPIO	40-pin GPIO header, populated
Ports	HDMI, 3.5mm analogue audio-video jack, 4xUSB 2.0, Ethernet, Camera Serial Interface (CSI), Display Serial Interface (DSI)
Dimensions	82mm x 56mm x 19.5mm, 50g

Além do mais, também no que se refere à comunicação de dados este permite a ligação de várias maneiras, tais como *WIFI*, *Ethernet*, *USB* e *Bluetooth.* São portas de conectividade que permitem ao *Raspberry Pi* conectar-se a outros dispositivos físicos e remotos. Isto tudo faz com que o *Raspberry Pi* seja bastante utilizado e tenha uma grande comunidade que desenvolve projetos, ajudando com ideias e dúvidas sobre principalmente problemas que possam aparecer durante o desenvolvimento de um projeto. Na Tabela 3 encontra-se as principais características técnicas. O *Raspberry Pi 3 B+* utilizado neste projeto encontra-se na Figura 22, foi utilizado devido as suas capacidades já mencionadas e além disso também muito devido ao seu preço.

Figura 23 – *Raspberry Pi* câmara de visão noturna.

A Figura 23 representa a câmara utilizada, que é exclusivamente concebida para ser utilizada com o *Raspberry Pi*. Esta permite a gravação de vídeos e captura de fotografias, tanto em ambiente noturno como ambiente diurno. As suas características foram aproveitadas ao máximo. Além disso, é bastante intuitiva para integrar com o *Raspberry Pi* e realizar o seu controlo. A facilidade em adquirir uma torna-a numa excelente opção para este projeto aliado às suas características.

3.1.3 Optocoupler

O *optocoupler* é constituído por um foto-díodo e um foto-transístor e trata-se de um componente eletrónico que transfere sinais entre dois circuitos isolados como se pode ver na Figura 24.

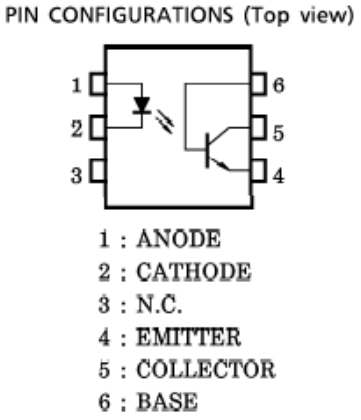


Figura 24 – *Optocoupler*.

O componente ao receber a corrente do sinal fonte e ao passar pelo foto-díodo emite uma luz infravermelha proporcional a intensidade da corrente recebida e o foto-transístor ao receber essa luz ativa a sua condução. Este componente permite isolar a parte de baixa tensão com a parte de maior potência. Prevenindo que o sistema de controlo neste caso seja afetado caso aconteça alguma falha de funcionamento na parte de maior potência.

Neste projeto o que acontece é que os sinais emitidos pelo *Raspberry Pi* não estão ligados diretamente com os componentes elétricos utilizados. O *Raspberry Pi* está assim protegido contra qualquer defeito que possa acontecer no circuito do motor do robô.

3.1.4 Componentes de orientação

Figura 25 – GPS *G-MOUSE*.

Num sistema autónomo de navegação é sempre necessário saber em que posição se encontra e para isso utilizou-se um GPS. A bússola e um sensor de *hall* servirão como meio de informação para direcionar e saber o posicionamento de direção do robô. O GPS fornece as coordenadas geográficas e através destas pode-se realizar cálculos com o intuito de perceber a distância do robô até as coordenadas finais do trajeto. Já sem falar, se pode guardar o histórico de movimentação, que na agricultura é bastante útil este tipo de informação para melhorar a sua eficiência.

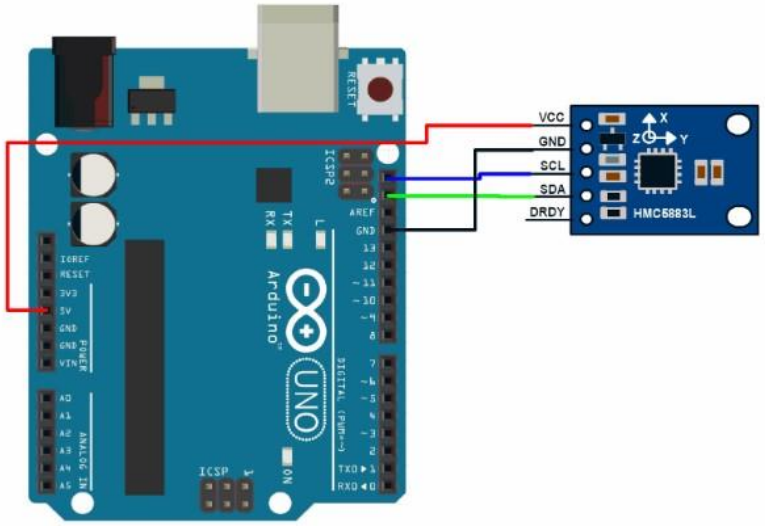


Figura 26 – *Arduino UNO* e bússola digital HMC5883L.

A bússola dá-nos informação sobre o direcionamento da frente do robô em relação ao norte magnético. Com esta informação conseguimos perceber para onde o robô está direcionado e trabalhar com a mesma para realizar a estratégia de condução autónoma do robô.

Este sensor funciona com a leitura de campos magnéticos e durante a sua aplicação foi percetível que ao ser colocado próximo das baterias ou até mesmo da estrutura de ferro causaria erros grosseiros nos valores lidos.

A solução encontrada para este problema foi a colocação do mesmo a uma distância relativa das interferências identificadas.

Conseguiu-se evitar essas interferências, porém devido a distância entre o sensor e o *Raspberry Pi*, houve necessidade de utilizar um *Arduino Uno* que comunica com este através de comunicação *Serial*.

Assim deste modo envia os valores lidos para o *Raspberry Pi* e conseguimos ter o sensor a ler valores o mais próximo do correto.

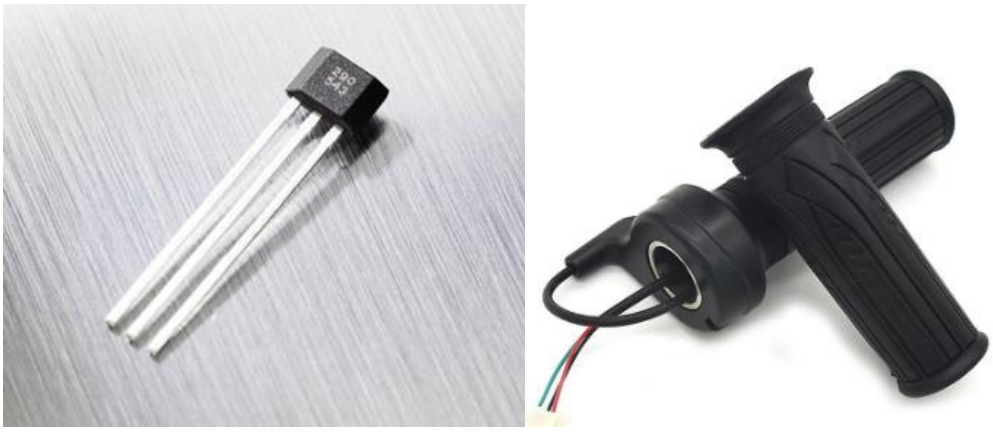


Figura 27 – Sensor de *hall* utilizado para identificar a direção das rodas.

O posicionamento das rodas é uma informação crucial para alinhar as rodas em frente e sabendo em que posição estas se encontram. O sensor utilizado para esse propósito é mostrado na Figura 27 e é um sensor analógico.

No entanto, como o *Raspberry Pi* não é capaz de adquirir valores analógicos, foi necessário utilizar o ADS1115 para converter o sinal analógico em digital, permitindo que este leia os valores de posição em que estão direcionadas as rodas.

Figura 28 – Ilustração onde se encontra posicionado o sensor de hall no robô.

O funcionamento consiste na leitura dos valores emitidos pelo sensor para determinar a posição das rodas. Para compreender os valores correspondentes à posição das rodas, foi necessário posicionar as rodas totalmente viradas para esquerda por exemplo, e observar os valores recebidos pelo *Raspberry PI*. Após esse procedimento, foi possível identificar os valores correspondentes às posições das rodas quando estão alinhadas e viradas totalmente para a esquerda e direita. É importante considerar que este sensor possui uma margem de erro nas leituras, sendo assim, foi estabelecido um intervalo de valores para cada posição.

3.2 Métodos

3.2.1 Raspberry Pi OS

O sistema operativo *Raspberry Pi OS* utilizado é baseado no *Debian*, otimizado para o *Raspberry Pi* e é o recomendado. Contém cerca de 35000 pacotes pré compilados e fáceis de instalar, são especificamente configurados para o desempenho otimizado do hardware ARM11. Este sistema operativo do *Raspberry Pi* está em constante evolução, tendo como principais

O sistema operativo *Buster* traz consigo algumas ferramentas que ficam a nossa disposição para utilizar. O *Geany* que é um editor de texto, já vem incorporado com o sistema operativo.

Este editor de texto foi desenvolvido em GTK+, possui funções básicas para um ambiente de desenvolvimento integrado (IDE) e é licenciado sob GNU GPL versão 2, tendo sido desenvolvido com o intuito de fornecer um IDE leve, rápido e com poucas dependências. A Figura 29 demonstra como é o ambiente de trabalho do Geany.

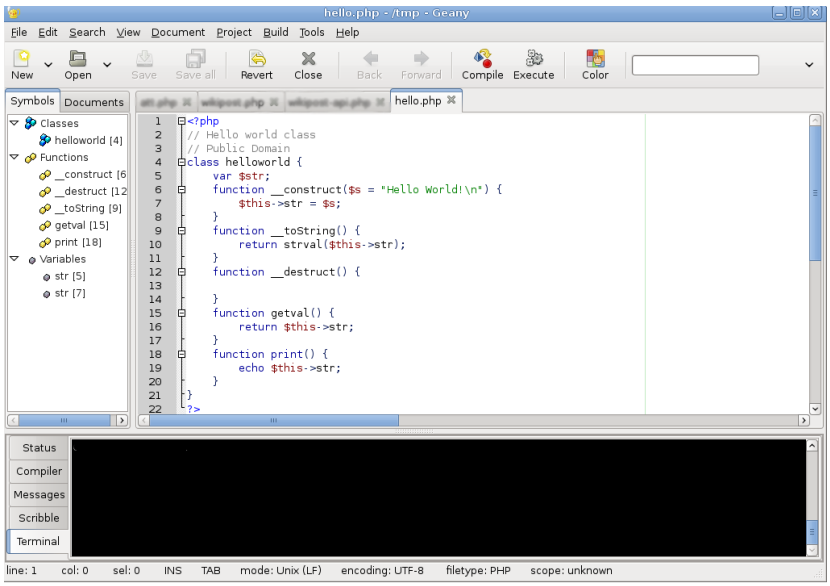


Figura 29 – Ambiente de trabalho do *software Geany.*

3.2.2 Linguagem *Python*

Python é uma linguagem computacional muito usada para a construção de *websites* e *software* de automação e análise de dados. Tratando-se de uma linguagem versátil, tendo sido desenvolvida com o intuito de ser fácil e entusiasmante para quem começa. É bastante popular, tornando-se das mais utilizadas neste momento. Foi lançada em 1991 e o seu nome é uma homenagem ao grupo de comédia *Monty Python*.

Graças a sua capacidade de suporte de módulos e *packages*, facilita muito no desenvolvimento de código através da reutilização.7

O que torna esta linguagem especial é o facto de ser comunitária e por isso mesmo várias pessoas independentes estão em constante construção de funcionalidades e bibliotecas, o que permite uma grande diversidade. E a simples sintaxe e a habilidade de usar menos linhas de código em comparação com outras linguagens. Além do mais tem a capacidade de ser compatível com plataformas como *Windows, Mac, Linux, Raspberry Pi* e outras. Porém pelo facto de ser bastante flexível e dinâmica a sua manutenção pode não ser fácil de executar, pois cresce em tamanho e torna-se mais complexa, tornando cada vez mais difícil a capacidade de encontrar e corrigir erros, e por isso mesmo, os utilizadores precisam de experiência para elaborar código de modo a facilitar a sua manutenção [37].

3.2.3 Blue Dot APP

A *Blue Dot* é uma aplicação que permite a comunicação com o *Raspberry Pi* via *Bluetooth* e está disponível para utilização livre. Esta aplicação, permite alterar o design gráfico de acordo com as necessidades do projeto, como adicionar botões com diferentes formas, tais como quadrados, retângulos ou círculos. É muito intuitiva para a programação do controlo do robô e simplifica a comunicação, exigindo apenas a ligação do telemóvel com o *Raspberry Pi* via *Bluetooth*.

Em algumas situações, pode ser necessário ao utilizador realizar controlo da movimentação do robô manualmente. Um exemplo, seria enquanto se carrega e se descarrega o robô para transportar em grandes distâncias entre campos de cultivo. Outra das situações é a colocação e alinhamento do mesmo para iniciar o trabalho ou em caso de acidente, reposicionar o robô para que este volte a trabalhar. Para isso, é utilizada a aplicação *Blue Dot*, que através de uma comunicação fácil e rápida permite guiar o robô como se estivesse a ser controlado por um joystick. A Figura 30 demonstra um exemplo de como a aplicação funciona. O círculo azul ao ser pressionado nos extremos, faz o robô mover-se nas respetivas direções.

Figura 30 – Exemplo de utilização da aplicação *Blue Do*t. Fonte: [38].

3.2.4 Biblioteca *OpenCV*

A biblioteca *OpenCV* (*Open Source Computer Vision*), foi desenvolvida pela *Intel* nos anos 2000 e está aberta ao público para livre utilização. Foi construída para providenciar uma infraestrutura no desenvolvimento de aplicações de visão computacional. Tem mais de 2500 algoritmos otimizados que podem ser utilizados para identificar objetos, vigiar movimentos de objetos e pessoas entre outras funcionalidades que se possa pensar referentemente a manipulação de imagem. É uma biblioteca que tem mais de 47000 pessoas que fazem parte da comunidade e ultrapassa os 18 milhões de *downloads* já realizados, é bastante utilizada em indústria e investigações académicas [39]. Além da visão computacional também se aplica em

Machine Learning, processamento de imagem em tempo real, sem faltar a inteligência artificial.

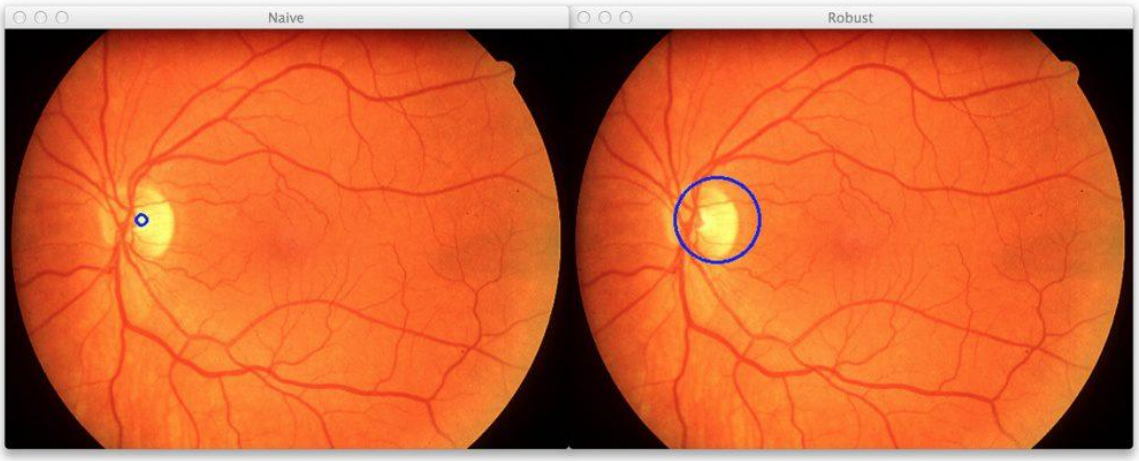


Figura 31 – Exemplo da utilização da biblioteca *OpenCV*. Fonte: [40].

Neste projeto recorreu-se a esta biblioteca principalmente para a manipulação e controlo da imagem com as ferramentas que esta nos permite utilizar para a realizar a deteção da zona com maior brilho. A Figura 31 é uma demonstração da deteção do sítio com maior brilho da imagem.

3.2.5 Biblioteca *TensorFlow*

Criado pela *Google* a *TensorFlow* foi lançada em 2015 e trata-se de uma biblioteca disponível para quem quiser utilizar na computação numérica e *Machine Learning* em larga escala para a construção de aplicações e que facilitam no processo de aquisição de dados, treino de modelos e antecipação de acontecimentos. Utiliza *Python* como *front-end* e executa com eficiência em C++.

O *TensorFlow* é uma biblioteca muito utilizada e bastante popular tendo como utilizadores o *Facebook* para uso de reconhecimento facial e a *Apple* com o reconhecimento de voz para a *Siri*. Estes são alguns exemplos da popularidade desta biblioteca.

O principal benefício é a abstração, pois em vez de lidar com pormenores de implementação de algoritmos o utilizador pode se focar no panorama geral da lógica da aplicação. Dentro das técnicas disponíveis a utilizar, a deteção de objetos quer seja em vídeo ou imagem vai ser utilizada neste projeto.

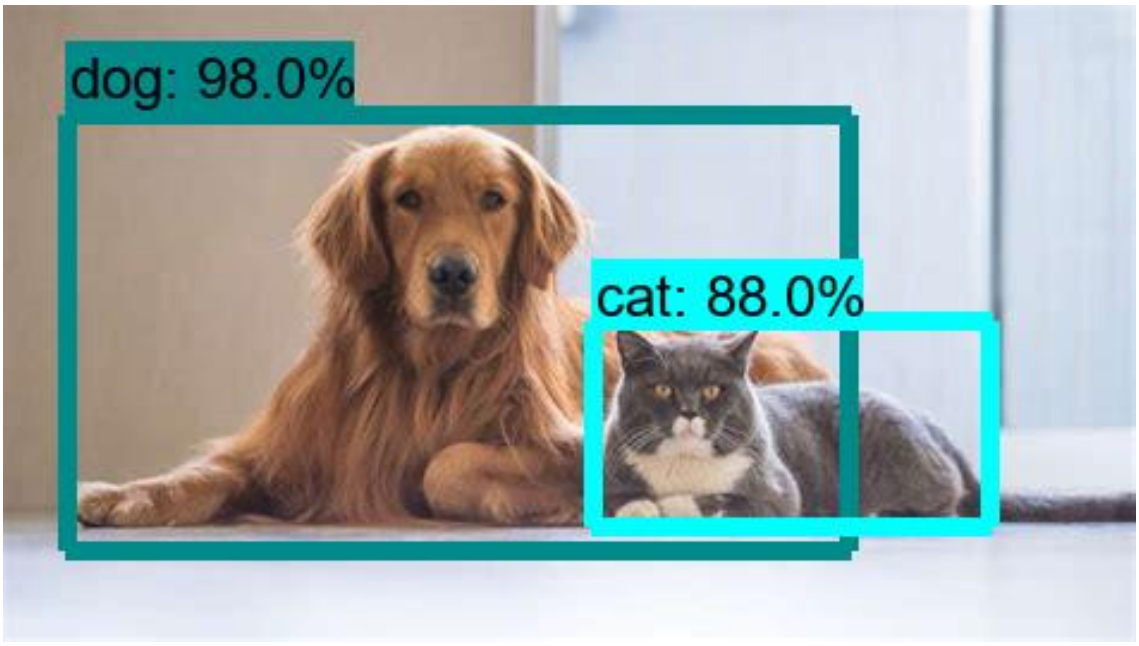


Figura 32 – Exemplo da utilização da biblioteca *TensorFlow*. Fonte: [41].

A deteção na presença de uma imagem ou vídeo encontra e identifica qualquer objeto conhecido de um conjunto de classe que é treinado para detetar. Caso seja necessário também se pode treinar para detetar objetos que não façam parte da classe de objetos já treinados. Esta técnica resumidamente funciona em três passos.

O primeiro passo consiste em gerar quadrados na imagem, e o segundo passo reside em prever em cada quadrado se é válido ou não, e por fim, os quadrados válidos formam um grande quadrado identificando o objeto, como é caso da Figura 32 que demonstra a deteção e identificação com quadrados à volta do cão e do gato, isto na mesma imagem [41].

Já a percentagem apresentada aquando deteta um objeto representa a confiança da deteção, portanto se aparece 88% no canto do quadrado que envolve o objeto identificado, significa que a percentagem de confiança, neste caso apresentado na Figura 32 de ser um gato é de 88%.

4. Implementação

Na primeira abordagem para a elaboração do sistema de condução autónoma, a estratégia implementada foi com a utilização do GPS e da bússola digital. Esta estratégia consiste em definir um ponto de coordenadas geográficas referente à conclusão do movimento, e através da leitura constante das coordenadas realizada pelo GPS, juntamente com os valores da bússola, orientar a movimentação do robô até esse ponto inicialmente pré-definido.

No entanto, durante os testes realizados, verificou-se que essa abordagem tinha muitas limitações, principalmente devido aos erros de leitura que tanto o GPS como a bússola apresentam. Por isso, teve de ser adotada uma outra abordagem que entregasse uma movimentação mais precisa, e decidiu-se utilizar a visão computacional.

Foi necessário adquirir uma câmara, neste caso com capacidade de visão noturna, e também um projetor de LEDs infravermelhos para servir de referência ao robô para orientação. No entanto, esta estratégia também revelou limitações no seu funcionamento tornando-a inviável para a utilização durante o dia.

Perante essa situação, houve outra vez necessidade de elaborar uma nova estratégia de funcionamento, e é aqui que entra o *TensorFlow*. Optou-se pela deteção de objetos para atender às necessidades de funcionamento durante situações de luminosidade.

Em conclusão, o sistema de condução autónoma funcional elaborado para o robô consistirá em duas estratégias de navegação.

Uma estratégia para o funcionamento em condições noturnas, onde o projetor de LED infravermelho servirá de referência para a orientação, e por último, temos a estratégia de deteção de objetos para funcionamento durante o dia.

4.1 Implementação das ligações

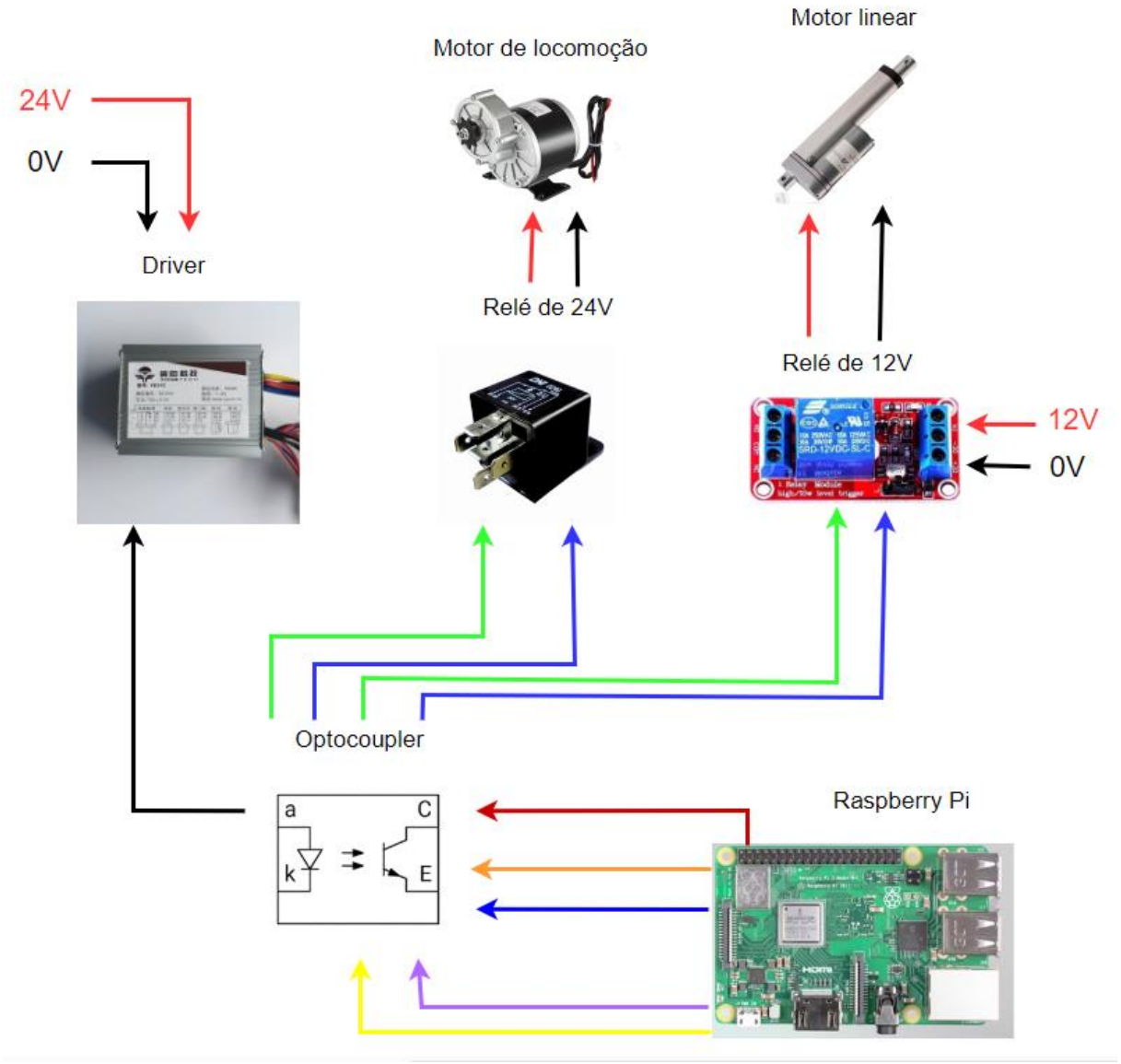


Figura 33 – Ligações para a realização do controlo dos motores.

Na Figura 33 apresenta-se as interligações entre os componentes, neste caso utilizados para controlar a movimentação e direção do robô. No esquema apresentado os relés são duplicados pois iremos ter um para acionar o robô movimentar-se em frente e o outro para se movimentar para trás, isto para os relés de 24V. Além dos relés temos ainda um *driver* que tem como função principal controlar a velocidade que quisermos que o motor de locomoção execute. Já os módulos de relé de 12V têm o intuito de controlar o motor linear, ou seja, realiza o direcionamento das rodas para a esquerda ou direita. O cérebro que emite os sinais de controlo como se pode ver na Figura 33 é o *Raspberry Pi*, e emite os sinais para os *optocouples* e estes emitem para os componentes respetivos. Neste caso só está apresentado na figura um *optocoupler* a receber todos os sinais, mas o que acontece realmente é que temos um *optocoupler* por sinal de controlo necessário.

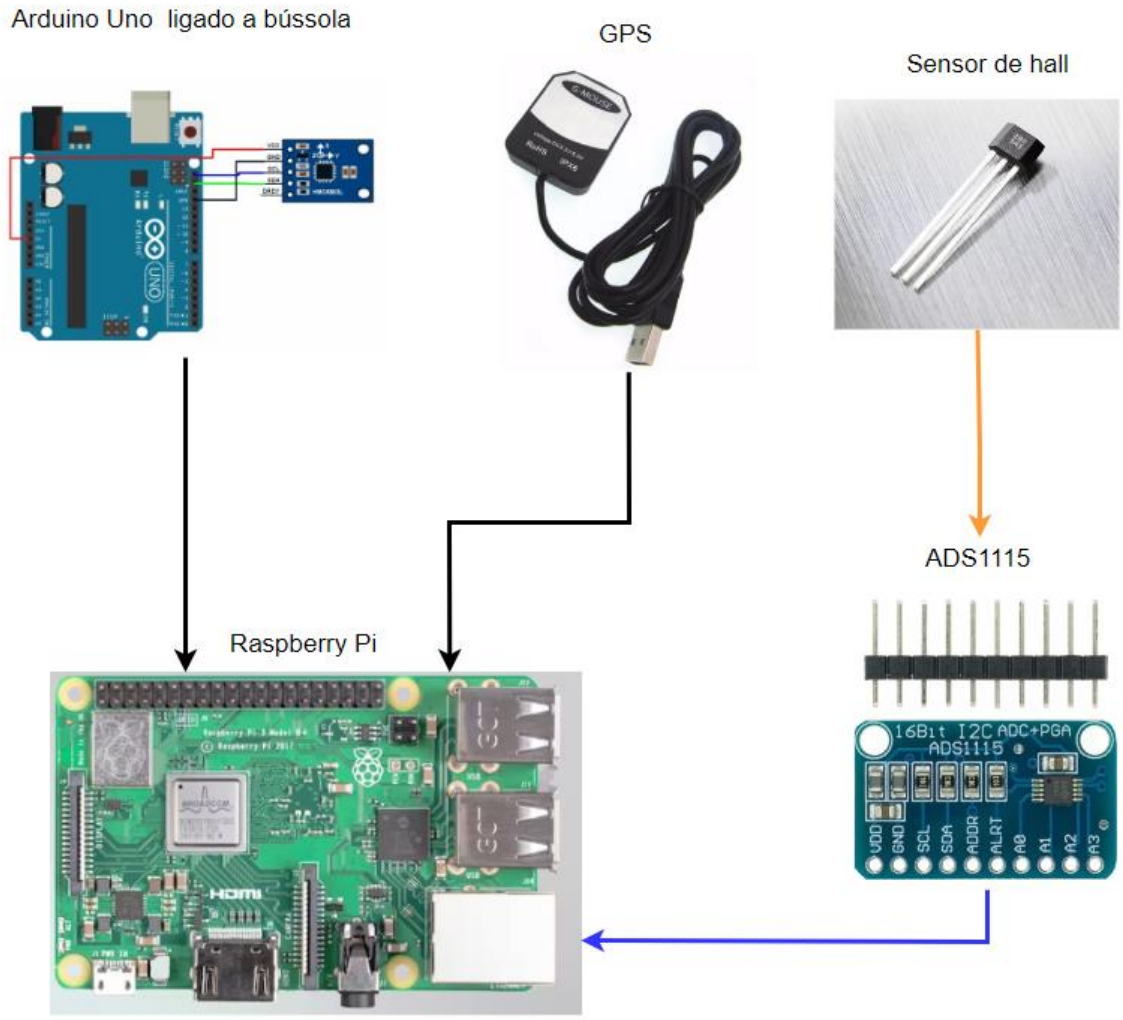


Figura 34 – Ligações para a implementação da primeira abordagem na implementação da navegação autónoma.

Na Figura 34 encontra-se o esquema de ligações realizado para executar a primeira estratégia de navegação autónoma na realização deste projeto. O que podemos ver na imagem mais uma vez é o *Raspberry Pi* sendo o cérebro, mas neste caso esta a receber informação.

Através de comunicação *Serial* o *Raspberry Pi* recebe a informação do GPS e da bússola digital. No caso da bússola digital está conectada ao *Arduino Uno*, em que este recebe a informação gerada pela mesma e a transmite para o *Raspberry Pi*.

Por último, temos então o sensor de *hall* que tem como função indicar-nos a posição em que se encontram as rodas. Como este sensor é analógico houve necessidade de converter o sinal para digital e esse é o papel do componente ADS1115.

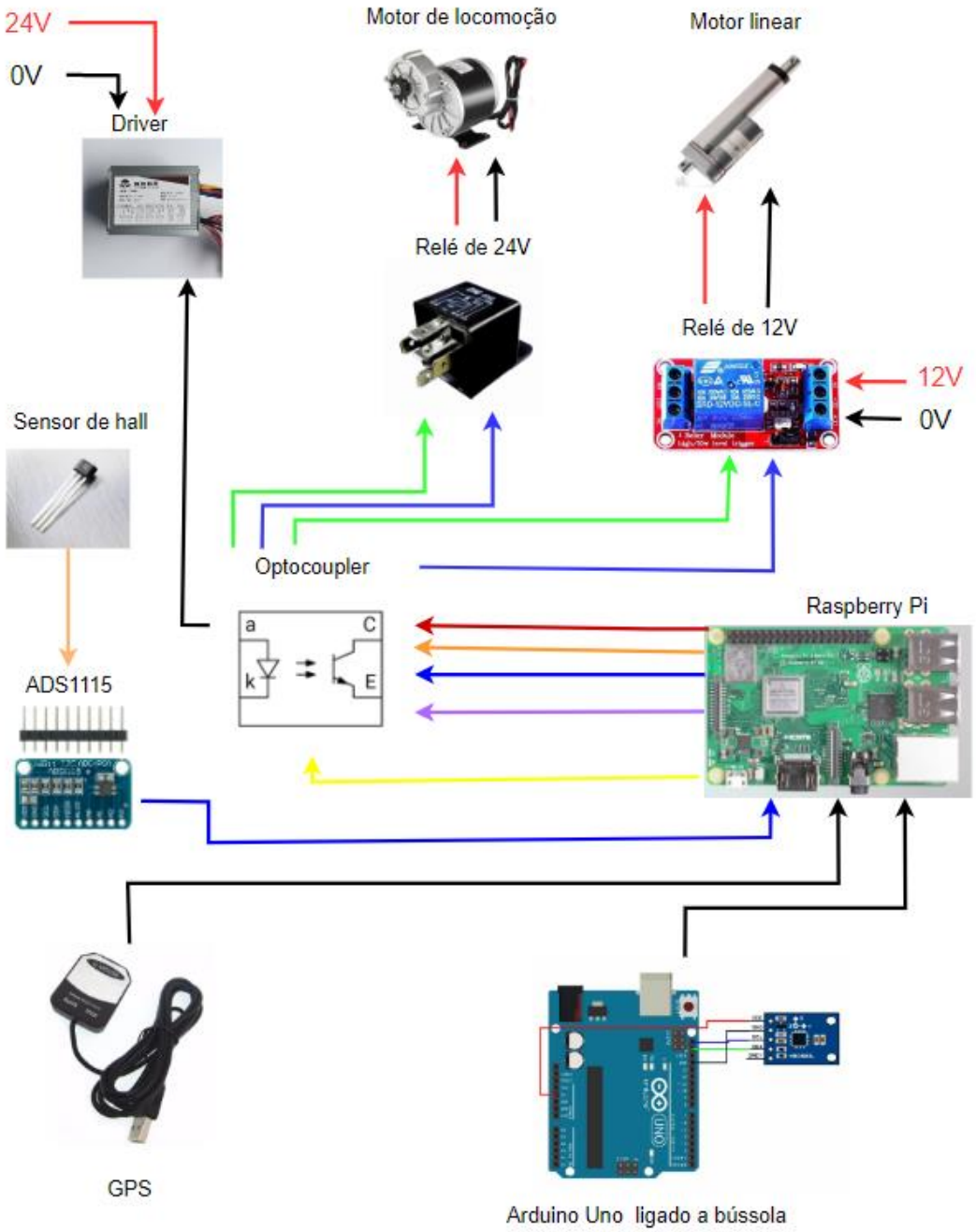


Figura 35 – Ligações para a implementação da primeira estratégia.

Na Figura 35 temos então a interligação de todos os componentes que foram utilizados para a primeira abordagem de estratégia de navegação autónoma.

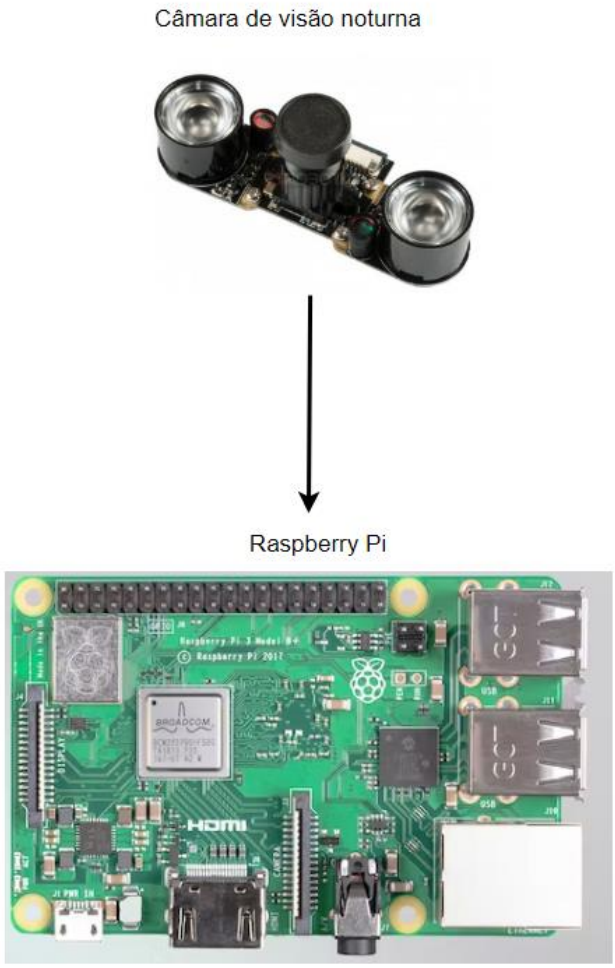


Figura 36 – Ligação para a implementação da estratégia de condução autónoma através da visão computacional.

Por último, apresenta-se na Figura 36, a ligação realizada para aplicação das duas estratégias de navegação desenvolvidas por último.

A informação da posição das rodas continua a ser importante, e sendo assim é necessário manter a ligação conforme se encontra na Figura 34.

Conectou-se então uma câmara *night vision* ao Raspberry Pi que servirá para executar as estratégias elaboradas baseadas em visão computacional.

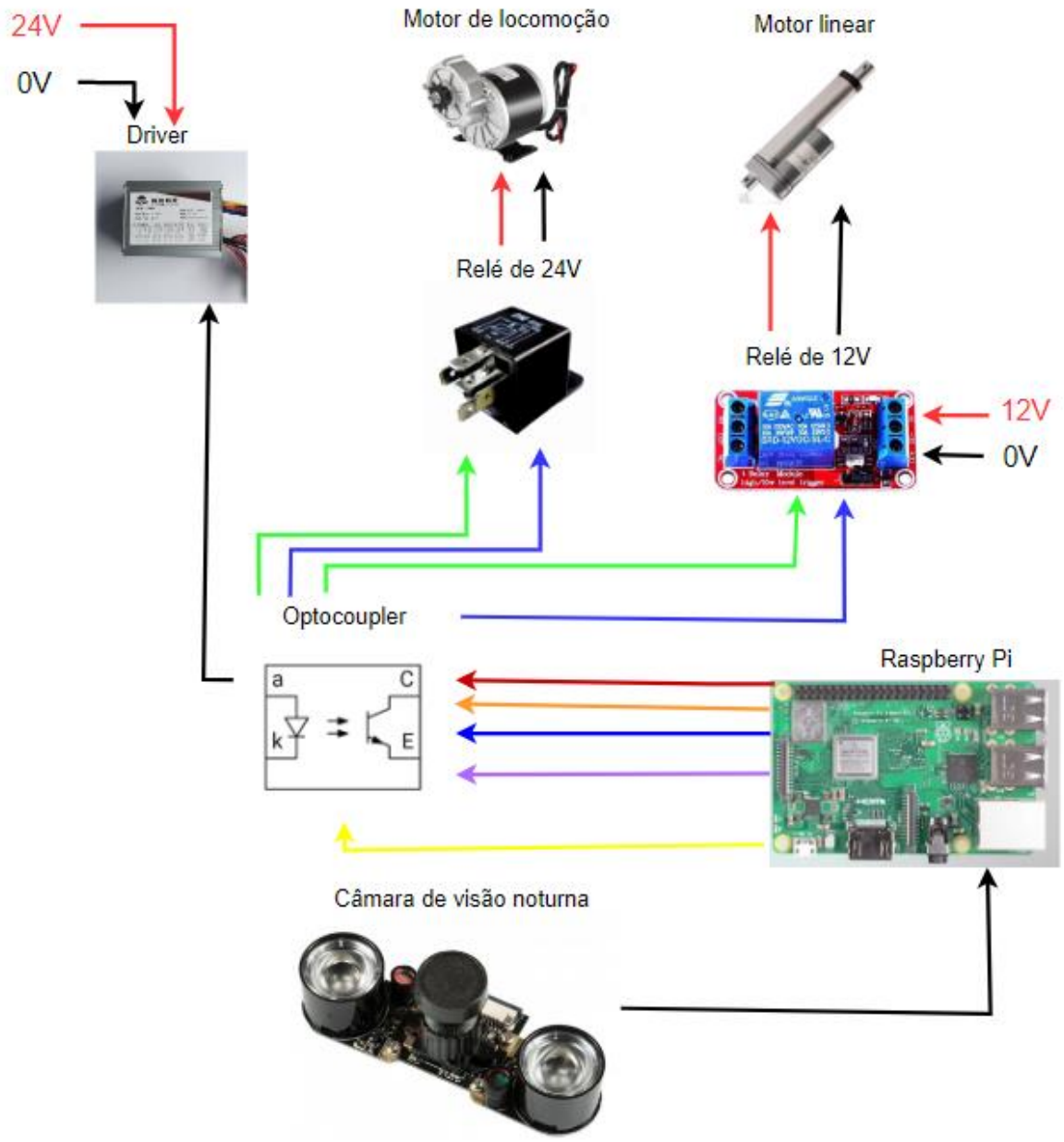


Figura 37 – Ligações para a implementação das estratégias baseadas na visão computacional.

Por último, apresenta-se na Figura 37 a interligação de todos os componentes do sistema de navegação autónoma baseado em visão computacional.

4.2 Estratégia para a primeira abordagem do sistema de navegação

O projeto inicial para implementar um sistema de navegação autónomo no robô baseia-se no uso de GPS e bússola digital. Para ser autónomo, o robô deve ser capaz de processar as informações que recebe e optar por um caminho, mas precisa de lhe ser concedido um objetivo para ter funcionalidade. Desta forma, a abordagem consiste em fornecer coordenadas geográficas como destino e, com a leitura dos dados fornecidos pelo GPS e bússola digital, o robô ser capaz de processar esses valores e executar os comandos necessários para seguir a trajetória até ao destino.

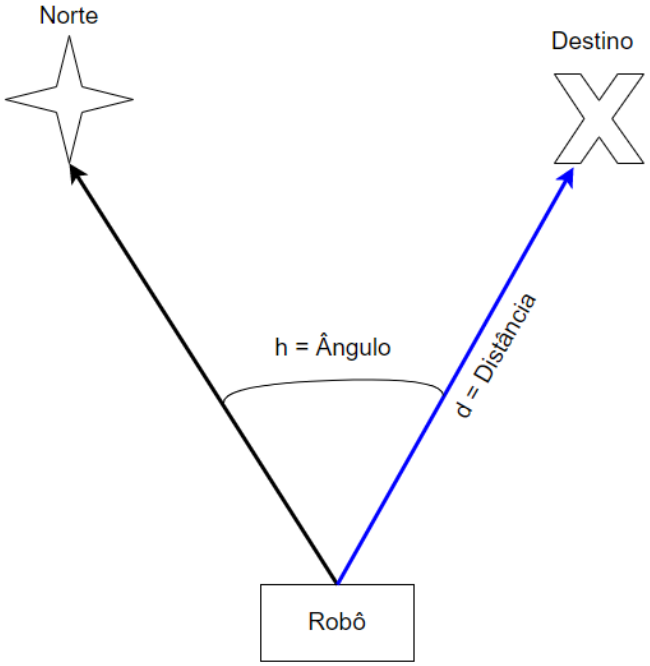


Figura 38 – Ilustração do funcionamento da estratégia planeada para a navegação autónoma.

A Figura 38 apresenta o esquema de funcionamento deste método para a implementação da condução autónoma. É preciso realizar cálculos de variáveis que estão destacadas na Figura 38. As variáveis são a distância e o ângulo, entre o robô e o ponto de coordenadas definido como destino final. Usando as coordenadas atuais do robô e do destino, é possível calcular tanto a distância quanto o ângulo.

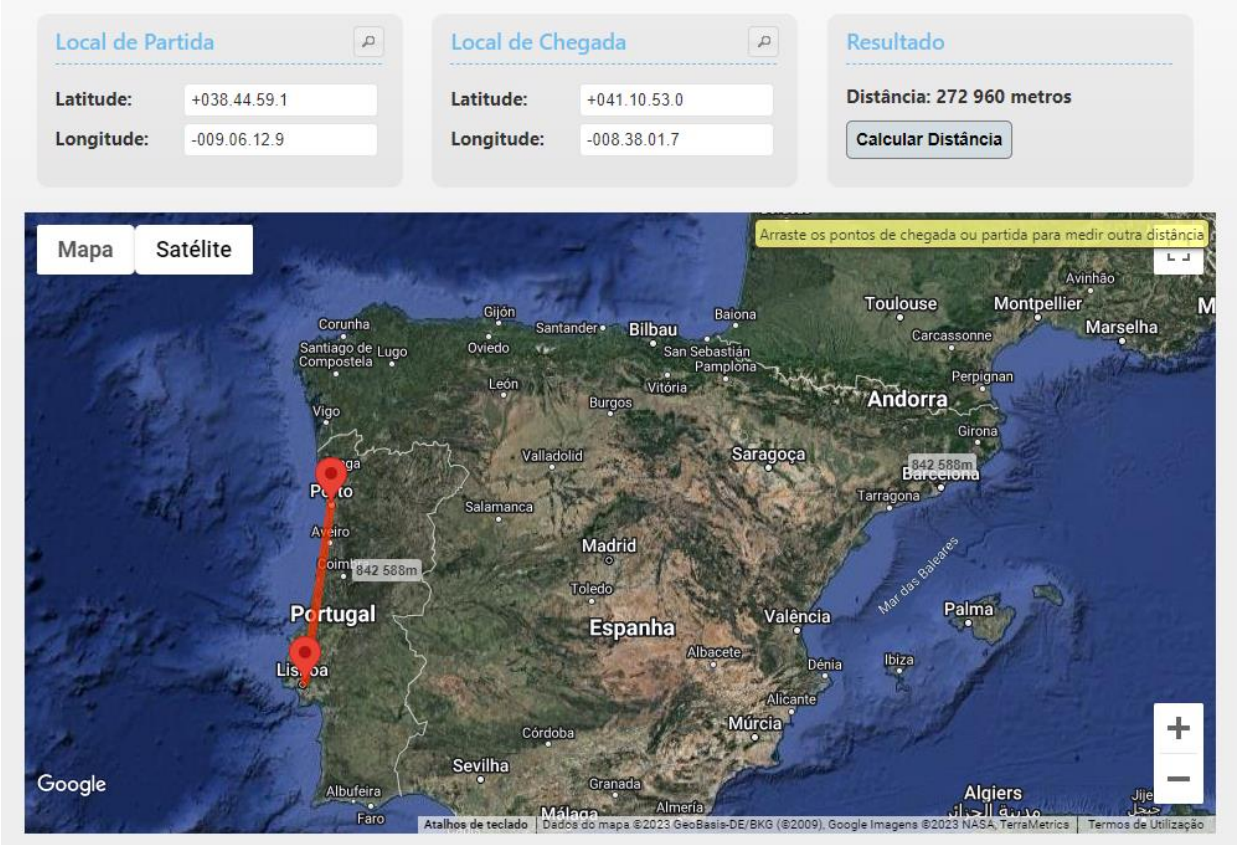


Figura 39 – Ilustração do cálculo da distância entre Porto e Lisboa. Fonte: [42].

Para a realização do cálculo da distância é utilizado a equação *haversine* que é bastante utilizada em aplicações de localização geográfica, pois esta considera a curvatura da terra no cálculo da distância entre dois pontos. Podemos visualizar na Figura 39 um exemplo do resultado do cálculo da distância em específico entre Porto e Lisboa. A equação (a) foi utilizada para o cálculo da distância e a equação (b) para o ângulo.

$$d = 2 * sin^{-1}(\sqrt{sin^2\left(\frac{\varphi 2-\varphi 1}{2}\right) + cos(\varphi 1) * cos(\varphi 2) * sin^2(\frac{\lambda 2-\lambda 1}{2})}) \qquad [a]$$

Em que:

d = Distância entre dois pontos

r = Raio da terra

λ1, φ1 = longitude e latitude da localização do robô

λ2, φ2 = longitude e latitude da localização do ponto final de destino

$$h = \text{atan2}(\sin(\lambda 2 - \lambda 1)\cos(\theta 2), \cos(\theta 1)\sin\theta 2 - \sin\theta 1\cos(\theta 2)\cos(\lambda 2 - \lambda 1))$$

[b]

h = Ângulo calculado

$\lambda 1$, $\varphi 1$ = longitude e latitude da localização do robô

$\lambda 2$, $\varphi 2$ = longitude e latitude da localização do ponto final de destino

$\theta 1$, $\theta 2$ – diferenças da longitude

O funcionamento do robô será baseado nas variáveis calculadas e nos valores obtidos pela bússola e GPS. A distância irá determinar se o robô chegou ao seu destino ou se precisa percorrer mais, enquanto o ângulo irá determinar a orientação que o robô tem de tomar em função dos valores da bússola. Se o valor do ângulo for igual ao da bússola ou se tiver dentro do intervalo definido, o robô está no rumo correto e pode seguir em frente.

Caso contrário tem de efetuar mudança de direção até se alinhar no sentido correto. O valor da bússola tem de ser um valor igual ou aproximado ao ângulo calculado para o robô estar direcionado corretamente. O código desenvolvido para aplicação desta estratégia encontra-se no Anexo 1 e as decisões do robô são baseadas nas variáveis apresentadas.

A correção da rota do robô é feita através do direcionamento das rodas para a esquerda ou direita. Durante o momento que o robô aciona o motor linear para direcionar as rodas este mantem em movimentação, ou seja, o motor de locomoção e o linear estão a funcionar ao mesmo tempo.

Porém a velocidade do motor de locomoção neste momento é menor para dar tempo suficiente para o robô corrigir a sua posição.

4.3 Estratégia de condução diurna

O código responsável pelo método de navegação durante o dia está disponível no Anexo 2. Este foi projetado com base na utilização da biblioteca *TensorFlow*, o que requer a instalação e a preparação adequada para a sua utilização. Foi utilizado o ficheiro COCO que contém modelos pré-treinados.

De modo a facilitar a movimentação durante os testes, optou-se por usar uma pessoa como objeto. A estratégia de orientação consiste na deteção do objeto e no alinhamento do robô em relação ao objeto. Isso define se o robô precisa virar à direita ou à esquerda ou se pode seguir em frente.

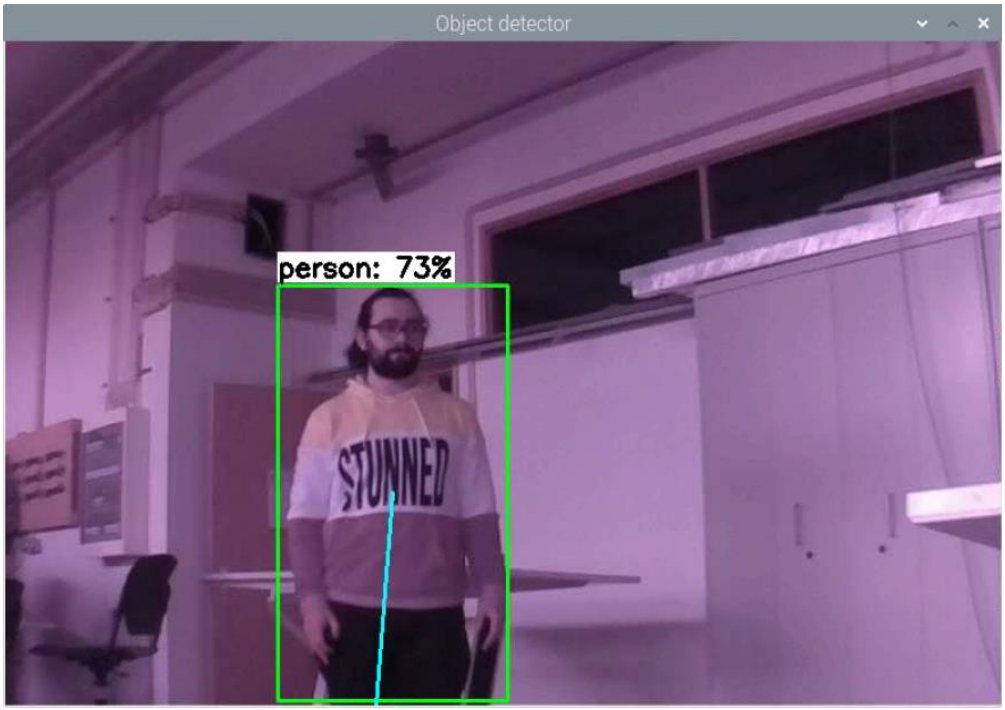


Figura 40 – Funcionamento da deteção com a pessoa alinhado com o robô.

A Figura 40 apresenta o momento em que o robô está alinhado com a pessoa. A caixa verde ao redor da pessoa representa o sistema de deteção, e a linha azul representa o alinhamento do objeto detetado com o robô. Toda a imagem tem uma resolução e se considerarmos que a imagem representada na Figura 40, tem uma resolução de 780x520, é através deste conhecimento que conseguimos determinar o meio da imagem. Sabemos então que a imagem tem 780 pixéis na horizontal e 520 pixéis na vertical, logo o meio da imagem é a na posição 390, considerando uma tolerância de 100 o que acontece para o robô estar a considerar que esta alinhado com a pessoa é que o valor tem de ser maior que 290 e menor que 490. Este esta a ser representado na imagem pela linha azul que se move conforme neste caso a pessoa se move.

Figura 41 – Funcionamento da deteção com a pessoa mais a esquerda do robô.

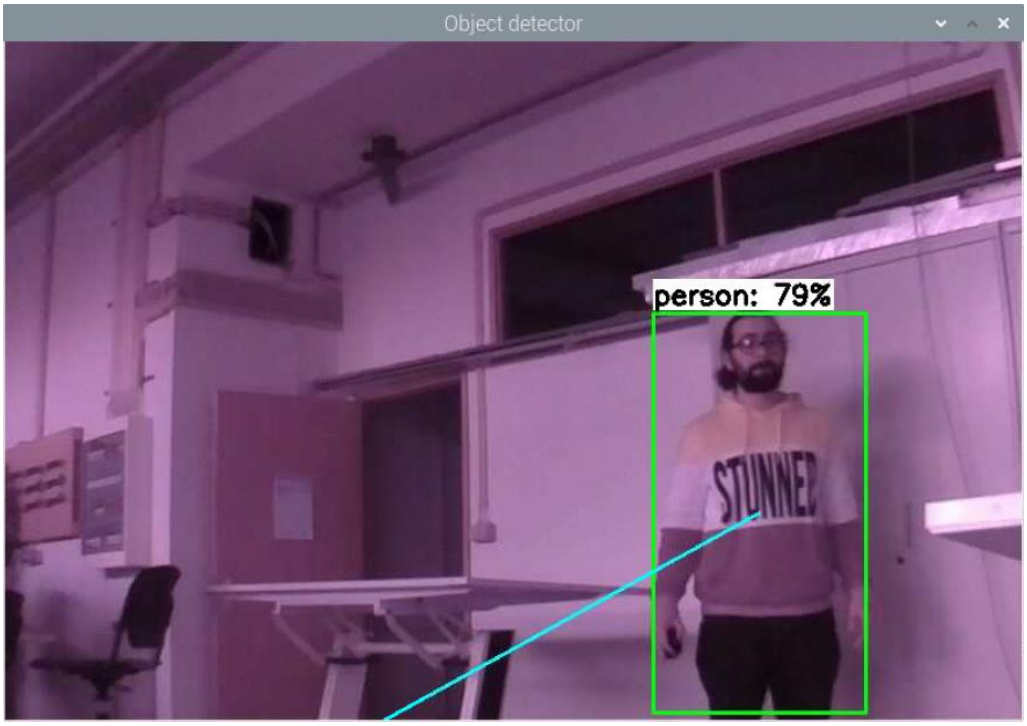


Figura 42 – Funcionamento da deteçao com a pessoa mais a direita do robô.

A Figura 41 e Figura 42, representam os momentos em que o alinhamento não existe, e o robô precisa tomar uma decisão de viragem. Assim, na Figura 41, robô precisa virar à esquerda para se alinhar com o objeto, e na Figura 42, ele precisa virar à direita.

No código desenvolvido o que esta a acontecer é que os valores na Figura 42 são inferiores a 290 e na Figura 43superiores a 490 e através desta comparação o robô tem que tomar as decisão de virar as rodas para a esquerda ou direita.

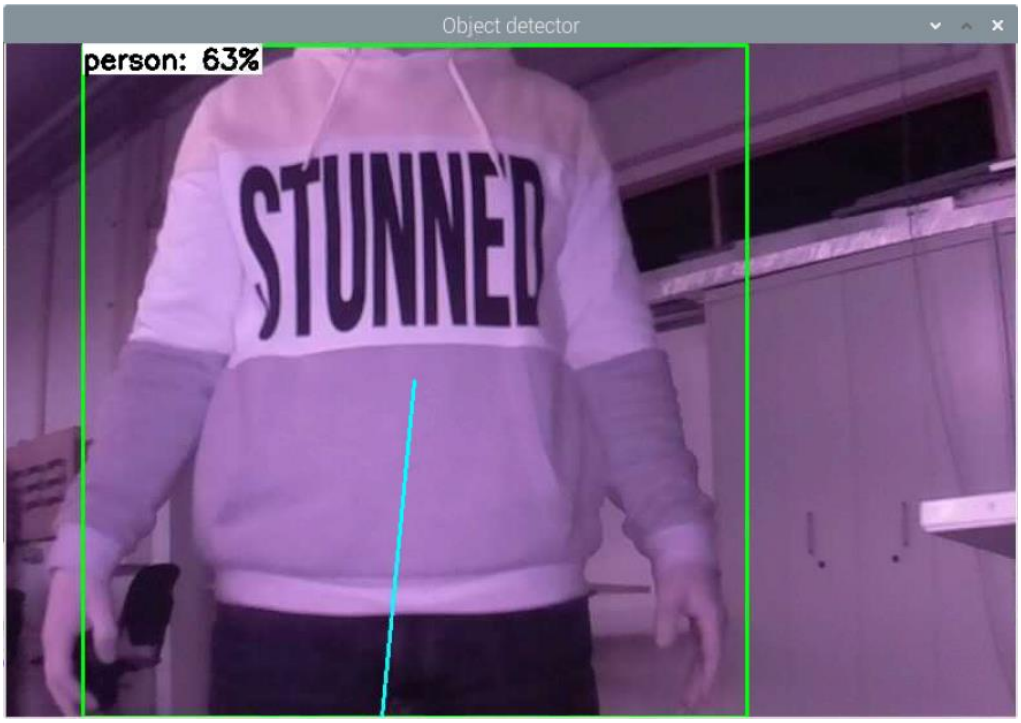


Figura 43 – Funcionamento da deteção com a pessoa perto do robô, ou seja, em situação de paragem.

Por último, temos o momento de paragem do robô, que ocorre quando atinge uma distância mínima do objeto. Quando a distância entre o fundo da imagem e o fundo do quadrado verde representado na figura é menor que 30, o robô fica a saber que chegou perto do objeto e precisa de parar. E enquanto esse valor for superior a 30 este mantém-se em movimento até chegar ao ponto destino que é chegar perto do objeto. Este valor é definido graças como já explicado anteriormente graças ao conhecimento da resolução da imagem, logo é percetível que a contagem começa de baixo para cima.

Na Figura 44, encontra-se o fluxograma do código desenvolvido em linguagem *Python* com a utilização da biblioteca *TensorFlow* para a deteção de objetos. Este demonstra a interligação do funcionamento entre o processamento que é realizado quando o objeto é detetado e a sua conexão com o restante código de programação, principalmente com as decisões de ativação e desativação dos motores.

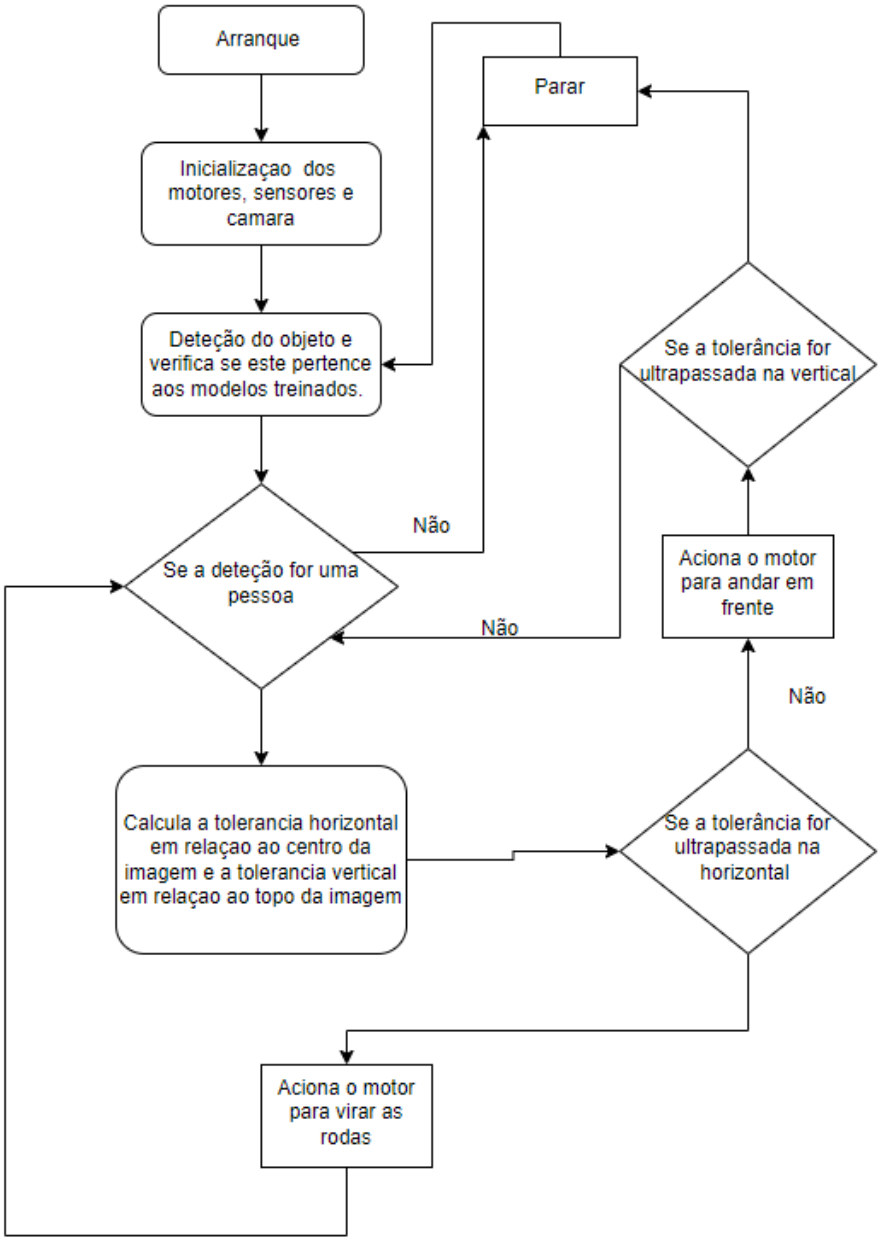


Figura 44 – Fluxograma do funcionamento do código desenvolvido para a estratégia de navegação autónoma.

4.4 Estratégia de condução noturna

Neste algoritmo, o método de navegação funciona de maneira semelhante ao que foi explicado anteriormente na estratégia de condução diurna. No Anexo 2 também se encontra o código elaborado para a execução desta estratégia. Como pode ser visto nas imagens apresentadas a única diferença é o ponto de referência. Para desenvolver este algoritmo, foi utilizada a biblioteca *OpenCV*, e, como tal, foi necessário instalá-la para utilizar todos os seus métodos disponíveis para processar imagens e vídeos.

Neste caso, é utilizado um projetor de LEDs infravermelhos, e o código desenvolvido tem como objetivo detetar o ponto com maior brilho gerado pelo mesmo.

Para detetar este ponto foi necessário realizar um processamento de imagem para reduzir ao máximo o brilho refletido em objetos. Entre outros processos feitos igualmente necessários para um bom funcionamento. O processamento de imagem para a redução de brilho visa tornar o código o mais preciso possível na deteção do projetor.

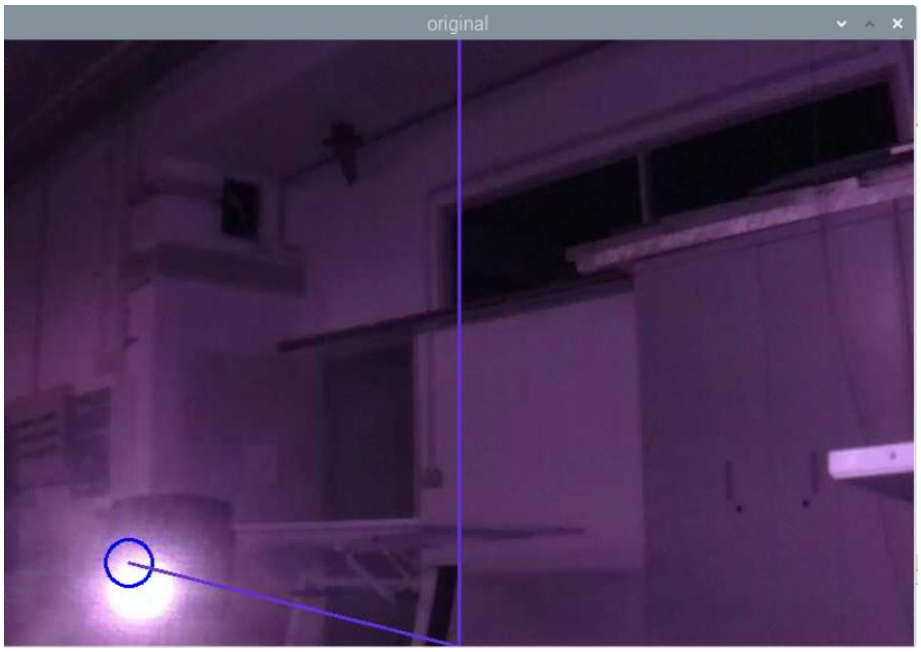


Figura 45 – Funcionamento da deteção do projetor mais a esquerda do robô.

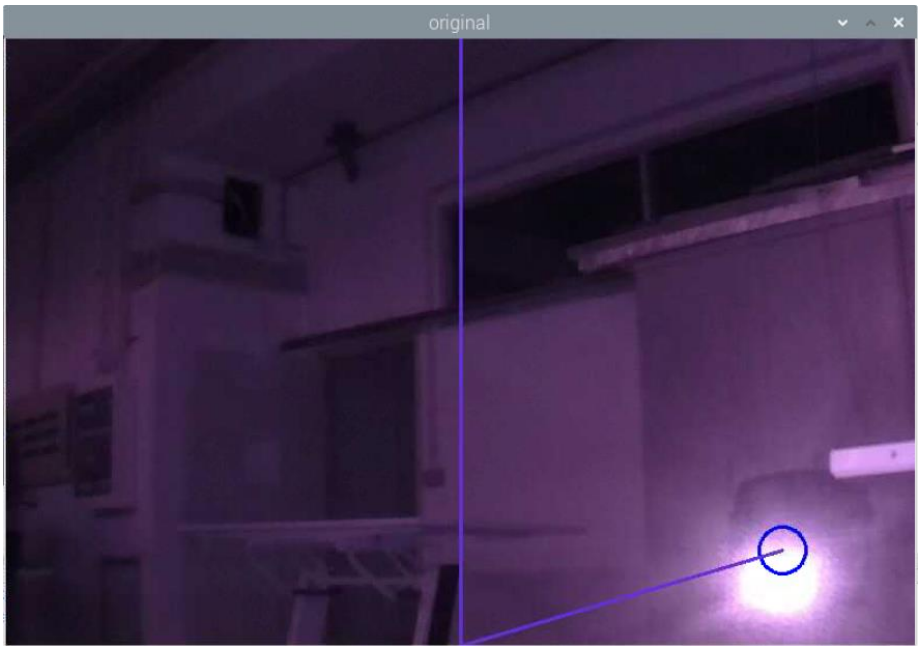


Figura 46 – Funcionamento da deteção do projetor mais a direita do robô.

Na Figura 45 e Figura 46, representam os momentos em que o robô tem de decidir virar a as rodas para a esquerda ou direita, a fim de se alinhar com o projetor.

Figura 47 – Funcionamento da deteção do projetor alinhado com o robô.

Figura 48 - Funcionamento da deteção do projetor alinhado e próximo do mesmo.

Na Figura 47, pode-se ver o robô alinhado com o projetor e a uma distância que permite o robô acionar o motor para se movimentar em frente.

Na Figura 48, demonstra ainda o momento em que o robô se aproxima do projetor, ou seja, chega ao seu destino e através do mesmo processo de avaliação utilizado na estratégia diurna faz com que o robô pare.

5. Resultados

5.1 Resultado da estratégia baseada na utilização do GPS e bússola digital

O teste realizado ao robô com esta estratégia de navegação foi realizado num espaço exterior, onde o número de interferências no GPS é menor em comparação a um espaço fechado. Isto porque num espaço fechado podemos ter interferências como bloqueio dos sinais de satélite por parte do edifício, eletromagnética por parte de dispositivos eletrónicos e entre outras mais que podem prejudicar a qualidade do sinal.

Em vista disso, optou-se por realizar os testes num parque de estacionamento com o benefício de ter espaço suficiente para o robô realizar as manobras. Para movê-lo para a área externa, utilizou-se a aplicação *Blue Dot*, controlando-o manualmente até o local pretendido para iniciar os testes de condução autónoma.

Ao executar o código de navegação autónoma, percebeu-se logo no primeiro teste que os movimentos eram constantes durante um curto espaço de tempo, e que em seguida o robô ficava totalmente desorientado a realizar movimentos. Virando por exemplo a direção totalmente para a direita e a começar a se mover, basicamente realizava círculos. Essas desorientações nos movimentos são explicadas pelos valores de longitude e latitude, e ângulo.

Tabela 4 - Resultados obtidos do robô no mesmo sítio do GPS e bússola.

Nº de medições	**Latitude (º)**	**Longitude (º)**	**Ângulo (º)**
1º	40.64299217	-7.91902283	241.579148
2º	40.64299283	-7.91902283	241.578960
3º	40.64299583	-7.91902300	241.578084
4º	40.64303450	-7.91902100	241.567291
5º	40.64302950	-7.91902433	241.568325
6º	40.64302800	-7.91903483	241.567516
7º	40.64300967	-7.91902933	241.573391
8º	40.64298083	-7.91906083	241.577907
9º	40.64305950	-7.91901350	241.561046
10º	40.64301233	-7.91904250	241.571081

O GPS e a bússola digital usados têm um erro de leitura nos valores durante o seu funcionamento, que foi detetado durante os testes realizados. Essa imprecisão na leitura causou indeterminação da posição e má orientação do robô.

O valor obtido do ângulo neste teste não apresenta variação considerável, porque se realizou com o robô parado. É durante o movimento do robô, que se consegue perceber a variação existente nos valores obtidos do ângulo, não permitindo desta forma uma determinação da direção correta em relação ao ponto de destino pré-definido.

No caso dos valores apresentados na Tabela 4, demonstram claramente o erro existente nos valores obtidos por parte do GPS. O erro consegue ser percetível ao calcularmos a distância entre o primeiro ponto e o último, obtendo um resultado de três metros entre os dois pontos. tendo em consideração que o robô se encontra parado na obtenção dos valores apresentados e que foram somente realizadas dez medições num curto espaço de tempo.

Claramente durante os testes realizados, o robô estava constantemente em alteração de direção, realizando movimentos sem sentido e fugindo completamente da orientação para o ponto destino pré-definido. Já que não se conseguia ter uma posição minimamente exata, especialmente em movimento.

O erro é expressivo o suficiente para perceber que esta estratégia não é viável para a condução autónoma utilizando este tipo de qualidade de componentes. Em resumo, esta estratégia de navegação autónoma não é eficaz devido à imprecisão dos componentes, GPS e bússola digital utilizados.

5.2 Resultados da navegação diurna

Para realizar este teste prático, foi necessário criar condições adequadas. Com condições adequadas quer se referir principalmente a realizar os testes num local aberto e com espaço suficiente para o robô se poder movimentar livremente. Umas das funções desenvolvidas para o robô foi a utilização de uma aplicação para controlá-lo manualmente por meio do telemóvel. Essa função foi útil para mover o robô e o posicionar adequadamente.

Um dos fatores que mais importa e que influencia muito a visão computacional é a luminosidade. A estratégia de navegação utilizada baseia-se na deteção de uma pessoa que serve como referência para guiar o robô. E os testes realizados demonstraram que a deteção da pessoa é garantida em várias condições de luminosidade a uma distância de até dez metros. Porém existem algumas dificuldades que o robô pode apresentar quando a luz do sol é bastante intensa e incide diretamente na câmara. Quero com isto dizer que o robô pode ter maior necessidade em executar movimentos para se alinhar com a pessoa, pois a deteção pode falhar em alguns momentos.

Figura 49 - Teste realizado ao desvio existente durante a movimentação do robô alinhado com o objeto.

A Figura 49 ilustra o teste realizado para avaliar o desvio na movimentação do robô em numa distância de aproximadamente dez metros. O robô está alinhado com a linha vermelha mostrada na Figura 49. O objetivo deste teste é verificar o desvio do robô ao percorrer os dez metros em linha reta.

Tabela 5 - Resultados obtidos dos desvios medidos.

	Nº de testes realizados									
	1º (cm)	**2º** (cm)	**3º** (cm)	**4º** (cm)	**5º** (cm)	**6º** (cm)	**7º** (cm)	**8º** (cm)	**9º** (cm)	**10º** (cm)
1 (m)	0	0	0	0	0	0	0	0	0	0
2 (m)	0	5	10	16	13	0	0	0	19	6
3 (m)	14	15	20	26	24	8	9	6	30	17
4 (m)	25	21	27	33	29	14	16	16	35	29
5 (m)	34	28	36	28	20	20	26	20	25	35
6 (m)	33	29	38	5	0	18	29	21	0	25
7 (m)	20	17	27	18	31	8	27	12	29	0
8 (m)	0	0	0	39	55	6	15	6	50	38
9 (m)	28	13	24	49	68	15	17	24	40	60
10 (m)	36	0	40	32	40	0	40	38	0	50

Na Tabela 5 apresentam-se então os resultados obtidos do desvio medido e destaca-se o primeiro valor que é sempre zero. O robô tem um comprimento de 150 cm, e no arranque do teste o mesmo é posicionado e alinhado com a pessoa de modo que este percorra em linha reta. Como o marcador é colocado na parte de trás do robô e as medições são feitas de metro em metro, acontece que no primeiro metro o robô está sem desvio nenhum. Mas como se pode ver com dois metros percorridos já começa a revelar algum desvio.

Alguns dos testes revelaram valores aceitáveis, isto se deve muito às condições de luminosidade em que foram realizados. Como foram realizados testes em diferentes zonas, a luminosidade afetava a câmara de maneiras diferentes. Pode-se então concluir que os testes que indicam piores resultados significam que a luminosidade afetou o seu funcionamento na deteção correta do objeto.

Na Figura 49, o meio utilizado para perceber o desvio foi a colocação de uma garrafa de água na parte de trás do robô, que ia pingando durante a sua movimentação. Para não estar sempre à espera que o chão secasse houve necessidade de realizar os testes em três zonas diferentes. Acredito que pela influência da luminosidade na qualidade de captação da imagem, por parte da câmara, fez com que uns testes ficassem bons e outros não no desvio encontrado. Contudo outro fator que se pode perceber nos testes realizados é que apesar de não se obter um desvio risório a movimentação do robô durante os dez metros é realizada em zigzags, que se acredita que a razão seja muito devido ao estado em que se encontram os pneus como se pode verificar nos resultados obtidos durante ambiente noturno. Devido à direção mecânica não estar corretamente alinhada e o facto de um pneu estar desgastado em relação ao outro.

Resumindo perante os testes realizados conclui-se que o robô tem capacidade de realizar um percurso autonomamente de pelo menos dez metros de distância com um desvio aceitável, considerando os valores apresentados na Tabela 5. Contudo consegue-se verificar que o sistema desenvolvido tem lacunas e que para ser preciso necessita de melhorias. Para um sistema implementado de baixo custo conseguiu-se atingir um resultado capaz de pelo menos realizar uma movimentação num espaço de dez metros aceitável para as condições de funcionamento existentes.

5.3 Resultados da estratégia de navegação noturna

Inicialmente, esta estratégia foi concebida para ser utilizada durante do dia. No entanto, os testes realizados durante o seu desenvolvimento revelaram que seria inviável a sua utilização em ambiente luminoso.

A razão deve-se ao facto da deteção do ponto com maior brilho poder ser imprecisa, uma vez que a luz reflete em vários objetos, tornando-os também pontos de brilho. Confundindo desta forma o robô e desviando-o do objetivo que seria o projetor.

Assim tornou-se impossível a execução de uma movimentação autónoma minimamente precisa durante uma determinada distância. Essa é a principal razão pela qual foi necessário adotar outra estratégia para a utilização do robô durante o dia e deixar esta estratégia para utilizar em ambiente noturno.

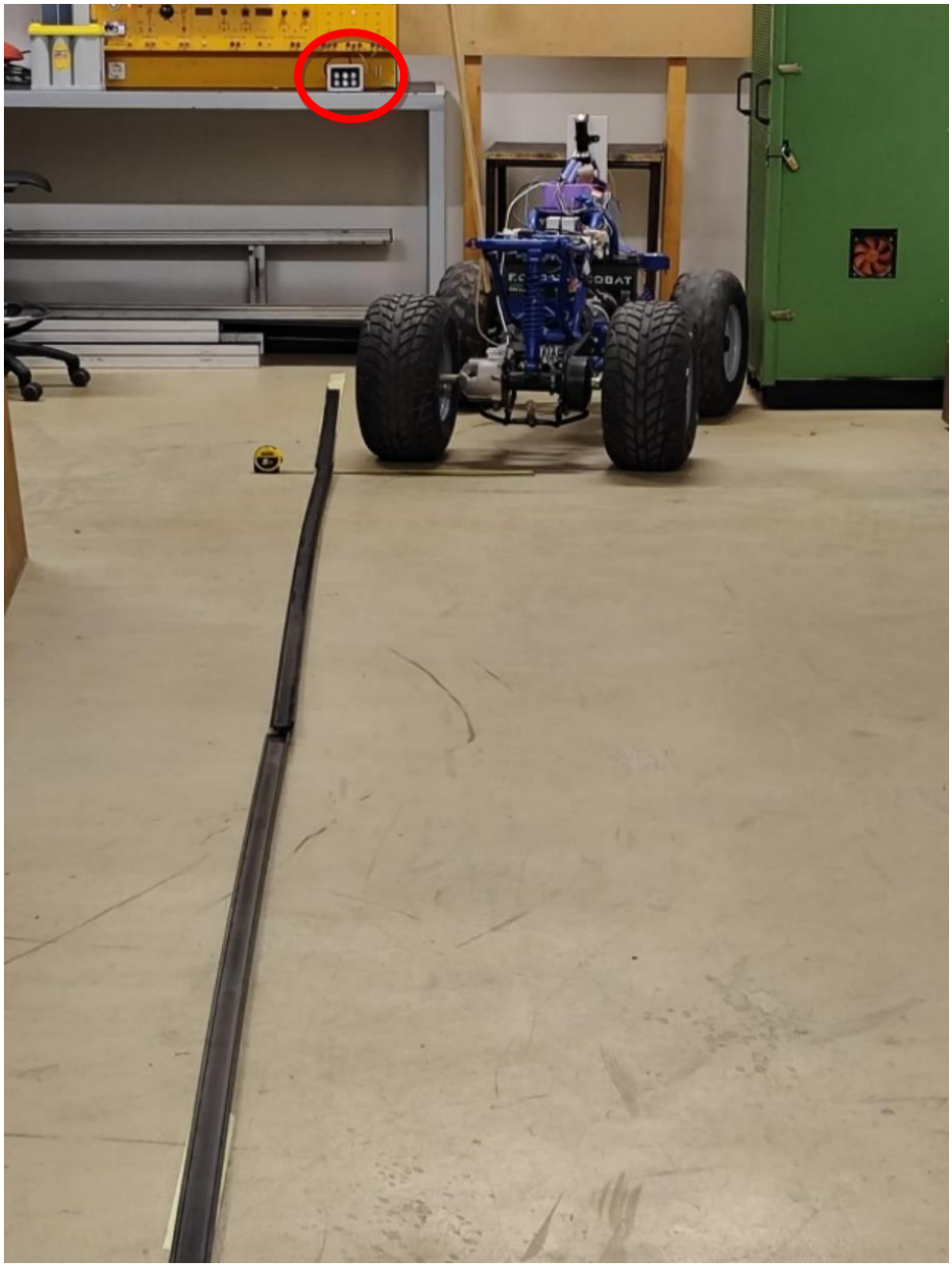


Figura 50 – Imagem da posição final de um dos testes realizados em ambiente noturno.

Na Figura 50, é uma apresenta o robô no final de um teste realizado em que a linha a preto que se pode ver server de referência para perceber o desvio encontrado durante os testes realizados, que se apresentam na Tabela 6.

Na figura pode se ver um círculo a vermelho que é onde se encontra o projetor de *leds* infravermelhos, não se consegue visualizar nenhuma luz emitida pelo mesmo, porque só é percetível pela câmara, como se pode ver nas figuras apresentadas anteriormente.

Tabela 6 – Medidas do desvio retiradas na realização do teste noturno.

Nº de teste realizados	**Valor das medidas (cm)**
1º	54
2º	23
3º	12
4º	22
5º	32
6º	40
7º	30
8º	52
9º	38
10º	42

Durante o teste realizado à noite, verificou-se que existe um desvio que é causado constantemente. As medições que se encontram na Tabela 6 foram realizadas somente no fim do percurso.

O motivo pelo qual de se ter realizado a medida só no fim do percurso, deve-se ao facto, de se verificar que o desvio durante a movimentação aumentava progressivamente. A distância percorrida durante a realização destes testes foi de mais ou menos 8 metros. Na Figura 50 encontra-se o robô em posição final de um teste realizado.

A Figura 51 mostra como o pneu da direita está mais gasto que o da esquerda e acredita-se que seja um dos motivos causador do desvio por parte do robô. Acoplado a este facto também se pode verificar que o alinhamento das rodas mecanicamente não esta condições e pode ser causador de desvio.

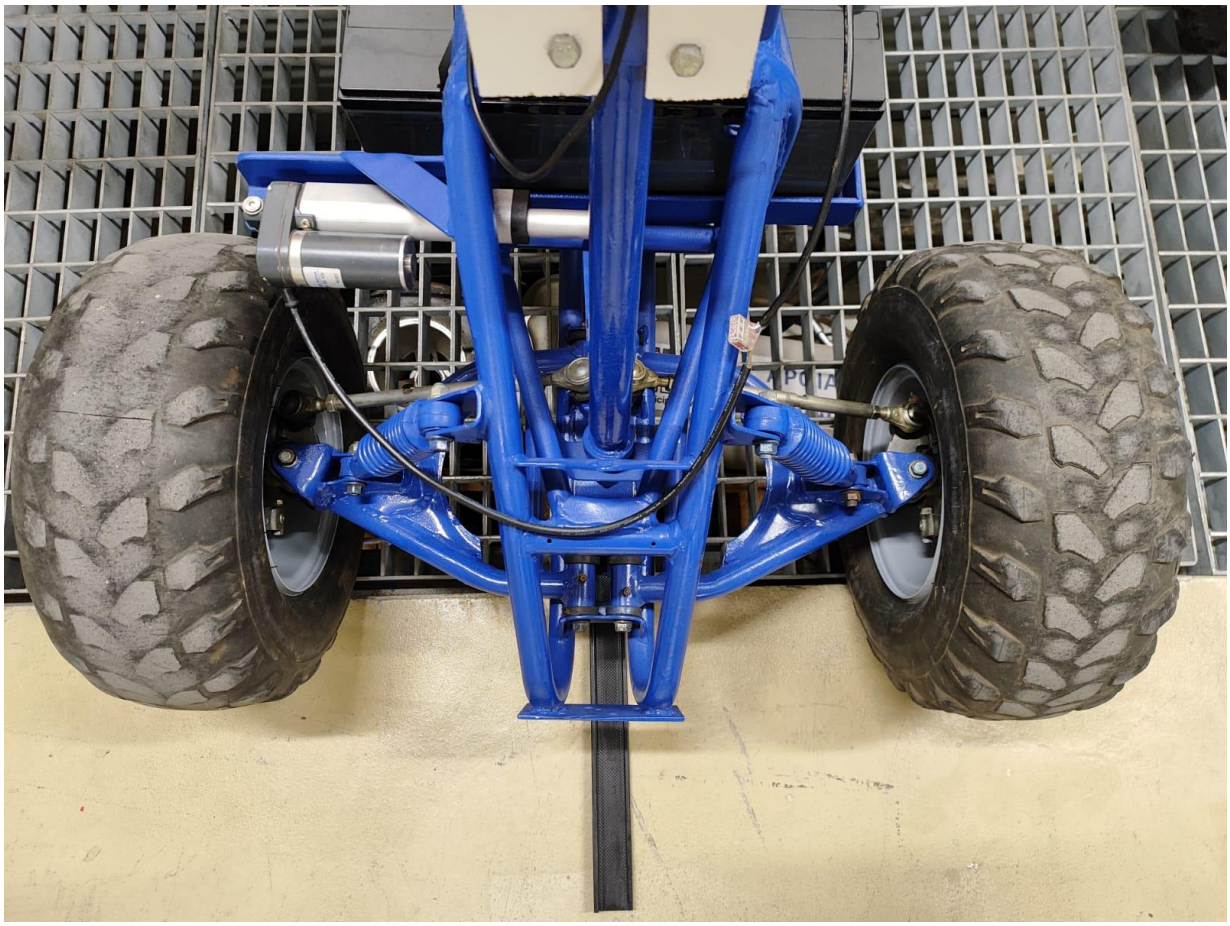


Figura 51 – Imagem reveladora da situação dos pneus do robô.

No que respeita à estratégia de navegação noturna, verifica-se que o funcionamento é correto quanto à deteção do brilho emitido pelo projetor. Durante os testes revelou-se ainda que, quando o robô está alinhado com o projetor, este segue em frente sem necessidade de acertar a sua posição de direção muitas vezes durante a sua movimentação.

Em comparação com a estratégia utilizada durante o dia, onde se verificava a correção da rota constantemente. Isto deve-se provavelmente ao facto de o projetor ser um objeto que não se mexe, enquanto uma pessoa tem sempre tendência de se mover, nem que seja só ligeiramente. Num sistema destes dependendo da calibração realizada pode ser o suficiente para o robô ter de acertar a sua movimentação.

6. Conclusões e melhorias futuras

O principal objetivo desta dissertação visa desenvolver uma estratégia para um sistema de navegação autónomo de baixo custo e o mais preciso possível. Para isso, foi realizado um estudo sobre os projetos já desenvolvidos para perceber as tecnologias mais utilizadas na condução autónoma com aplicação na agricultura. Concluiu-se que a visão computacional, *LIDAR* e RTK-GPS são as mais utilizadas, sendo que os projetos que combinam várias tecnologias obtêm melhor rendimento em tarefas autónomas, como a identificação com visão computacional, deteção de obstáculos com *LIDAR* e localização com RTK-GPS.

Durante o projeto, foram implementadas duas estratégias de navegação autónoma. A primeira, baseada em GPS e bússola digital, revelou-se imprecisa e impraticável para funcionar em condições reais. Foi então decidido utilizar a visão computacional como principal sistema de navegação autónoma. A primeira implementação com a visão computacional utilizou, como referência um projetor LED infravermelho. Também não obteve sucesso durante o dia devido à quantidade de pontos de brilho detetados pela visão computacional, causados pelo reflexo nos objetos. A solução final foi então implementar a deteção de objetos através da biblioteca *TensorFlow* durante o dia. E utilizar a biblioteca *OpenCV* para a estratégia de navegação noturna com o projetor de LED infravermelho.

Ao longo do projeto, foram enfrentados vários obstáculos, mas o objetivo de obter um sistema de navegação autónomo de baixo custo foi alcançado dentro do possível. Testes realizados mostraram que o desvio do robô era inferior a 50cm em uma distância percorrida de 10 metros durante o dia e noite. De noite foi percetível a elevada precisão ao detetar o objeto e sem necessidades de acertos de posição tendo em conta que o robô está alinhado com o mesmo.

No entanto, algumas falhas foram identificadas, como a lentidão e limitações do sistema de direção do robô que é débil pela necessidade de espaço para realizar uma movimentação em 180º. Além disso, o desvio criado pelas questões mecânicas do robô provou-se que afetam a sua precisão. No geral, este projeto mostrou que é possível desenvolver robôs autónomos, porém ainda é muito difícil atingir uma plataforma totalmente autónoma e preparada para a agricultura com um orçamento reduzido.

Em resumo, esta dissertação teve um contributo para a agricultura, sendo alcançado por meio de testes e implementação de diferentes estratégias de baixo custo. Foram identificados desafios ao longo do projeto, mas demonstrou-se que é possível alcançar resultados com uma capacidade financeira limitada.

7. Referências

[1] K. E. Giller et al., «The future of farming: Who will produce our food?», doi: 10.1007/s12571-021-01184-6, Acedido: março de 2022.

[2] B. Ravinder e P. Reddy, «An Advanced Agriculture IoT Technology with Wireless Application», doi: 10.1109/ICAC3N53548.2021.9725711, Acedido: março de 2022.

[3] Y. Liu, X. Ma, L. Shu, G. P. Hangfdcke, e A. M. Abu-Mahfouz, «From Industry 4.0 to Agriculture 4.0: Current Status, Enabling Technologies, and Research Challenges», doi: 10.1109/TII.2020.3003910, Acedido: março de 2022.

[4] L. Emmi, «A hybrid representation of the environment to improve autonomous navigation of mobile robots in agriculture», https://doi.org/10.1007/s11119-020-09773-9, Acedido: março de 2022.

[5] N. Shalal et al., «A REVIEW OF AUTONOMOUS NAVIGATION SYSTEMS IN AGRICULTURAL ENVIRONMENTS», https://research.usq.edu.au/item/q2456/a-review-of-autonomous-navigation-systems-in-agricultural-environments, Acedido: abril de 2022

[6] C. P. Baillie et al., «A review of the state of the art in agricultural automation. Part III: Agricultural machinery navigation system», doi: 10.13031/aim.201801591, Acedido: abril de 2022.

[7] J. A. Thomasson et al., «A review of the state of the art in agricultural automation. Part II: On-farm agricultural communications and connectivity»; doi: 10.13031/aim.201801590, Acedido: abril de 2022.

[8] C. P. Baillie et al., «A review of the state of the art in agricultural automation. Part I: Sensing technologies for optimization of machine operation and farm inputs»; doi: 10.13031/aim.201801589, Acedido: abril de 2022.

[9] Z. Zhang et al., «Development of a Robot Combine Harvester for Wheat and Paddy Harvesting», doi: 10.3182/20130327-3-JP-3017.00013, Acedido: maio de 2022.

[10] Z. De-An, «Design and control of an apple harvesting robot», doi:10.1016/j.biosystemseng.2011.07.005, Acedido: maio de 2022.

[11] S. Birrell et al., «A field-tested robotic harvesting system for iceberg lettuce», doi: 10.1002/rob.21888, Acedido: maio de 2022.

[12] A. R. Baltazar et al., «Smarter Robotic Sprayer System for Precision Agriculture», doi: 10.3390/electronics10172061, Acedido: maio de 2022.

[13] M. Pérez-Ruiz et al., «Highlights and preliminary results for autonomous crop protection», doi: 10.1016/j.compag.2014.11.010, Acedido: junho de 2022.

[14] I. Vatavuk, G. Vasiljević, e Z. Kovačić, «Task Space Model Predictive Control for Vineyard Spraying with a Mobile Manipulator», doi: 10.3390/agriculture12030381, Acedido: junho de 2022.

[15] O. Ranta et al., «Quality Analysis of Some Spray Parameters When Performing Treatments in Vineyards in Order to Reduce Environment Pollution», doi: 10.3390/su13147780, Acedido: junho de 2022.

[16] J. Goricanec et al., «Heterogeneous autonomous robotic system in viticulture and mariculture - project overview», doi: 10.23919/ConTEL52528.2021.9495969, Acedido: junho de 2022.

[17] U. Weiss e P. Biber, «Plant detection and mapping for agricultural robots using a 3D LIDAR sensor», doi: 10.1016/j.robot.2011.02.011, Acedido: julho de 2022.

[18] J. Roldán et al., «Heterogeneous Multi-Robot System for Mapping Environmental Variables of Greenhouses», doi: 10.3390/s16071018, Acedido: julho de 2022.

[19] A. Rafiq, «Construction and development of an automatic sprayer for greenhouse», https://cigrjournal.org/index.php/Ejounral/article/view/2606, Acedido: julho de 2022.

[20] T. Tanabata et al., «Development of a plant conveyance system using an AGV and a self-designed plant-handling device: A case study of DIY plant phenotyping», doi: 10.1270/jsbbs.21070, Acedido: julho de 2022.

[21] S. Krishnan et al., «A UWB based Localization System for Indoor Robot Navigation», doi: 10.1109/ICUWB.2007.4380919, Acedido: julho de 2022.

[22] P. Dabove et al., «Indoor positioning using Ultra-wide band (UWB) technologies: Positioning accuracies and sensors' performances», doi: 10.1109/PLANS.2018.8373379, Acedido: julho de 2022.

[23] Z. Huang et al., «A sound-based positioning system with centimeter accuracy for mobile robots in a greenhouse using frequency shift compensation», doi: 10.1016/j.compag.2021.106235, Acedido: agosto de 2022.

[24] J. L. Xue e T. E. Grift, «Agricultural Robot Turning in the Headland of Corn Fields», doi: 10.4028/www.scientific.net/AMM.63-64.780, Acedido: agosto de 2022.

[25] Astrand e Baerveldt, «An Agricultural Mobile Robot with Vision-Based Perception for Mechanical Weed Control», https://link.springer.com/article/10.1023/A:1015674004201, Acedido: agosto de 2022.

[26] S. Bonadies e S. A. Gadsden, «An overview of autonomous crop row navigation strategies for unmanned ground vehicles», doi: 10.1016/j.eaef.2018.09.001, Acedido: agosto de 2022.

[27] Y. Song et al., «Technology Application of Smart Spray in Agriculture: A Review», doi: 10.1080/10798587.2015.1015781, Acedido: agosto de 2022.

[28] S. A. Hiremath et al., «Laser range finder model for autonomous navigation of a robot in a maize field using a particle filter», doi: 10.1016/j.compag.2013.10.005, Acedido: setembro de 2022.

[29] S. Campbell et al., «Sensor Technology in Autonomous Vehicles : A review», doi: 10.1109/ISSC.2018.8585340, Acedido: setembro de 2022.

[30] A. Bechar e C. Vigneault, «Agricultural robots for field operations: Concepts and components», doi: 10.1016/j.biosystemseng.2016.06.014, Acedido: setembro de 2022.

[31] X. Han et al., «Development of a low-cost GPS/INS integrated system for tractor automatic navigation», http://www.ijabe.org/index.php/ijabe/article/view/3070, Acedido: outubro de 2022.

[32] J. A. Thomasson et al., «Autonomous Technologies in Agricultural Equipment: A Review of the State of the Art», https://elibrary.asabe.org/data/pdf/6/913c0119/913C0119.pdf, Acedido: outubro de 2022.

[33] B. Matias et al., «High-accuracy low-cost RTK-GPS for an unmannned surface

vehicle», doi: 10.1109/OCEANS-Genova.2015.7271673, Acedido: outubro de 2022.

[34] J. Jackson, «Real-time Kinematic Positioning: Background, Assessment and Forecasting», https://www.proquest.com/openview, Acedido: outubro de 2022.

[35] S. K. Upadhyaya, G. S. Pettygrove, J. W. Oliveira, e B. R. Jahn, «AN INTRODUCTION - GLOBAL POSITIONING SYSTEM», Acedido: novembro de 2022.

[36] H. Fazeli, et al., «EVALUATING THE POTENTIAL OF RTK-UAV FOR AUTOMATIC POINT CLOUD GENERATION IN 3D RAPID MAPPING», doi: 10.5194/isprsarchives-XLI-B6-221-2016, Acedido: novembro de 2022.

[37] «What is Python? | Teradata». https://www.teradata.com/Glossary/What-is-Python, Acedido: janeiro de 2023.

[38] «Blue Dot — bluedot 2.0.0 Documentation». https://bluedot.readthedocs.io/en/latest/index.html, Acedido: março de 2023.

[39] «About», OpenCV. https://opencv.org/about/, Acedido: janeiro de 2023.

[40] A. Rosebrock, «Finding the Brightest Spot in an Image using OpenCV», PyImageSearch, 29 de setembro de 2014. https://pyimagesearch.com/2014/09/29/finding-brightest-spot-image-using-python-opencv/, Acedido: janeiro de 2023.

[41] G. L. Team, «Real-Time Object Detection Using TensorFlow», Great Learning Blog: Free Resources what Matters to shape your Career!, 24 de dezembro de 2021. https://www.mygreatlearning.com/blog/object-detection-using-tensorflow/, Acedido: janeiro de 2023

[42] F. P. de Columbofilia, «Cálculo de Distâncias». http://nacionais2022.fpcolumbofilia.pt/Distancia, Acedido: janeiro de 2023).

ANEXO 1

```
import RPi.GPIO as GPIO
import time
import serial
import math
import haversine as hs
import Adafruit_ADS1x15
from haversine import Unit
from math import atan2, degrees

global waypoint_la
waypoint_la = 40.5587319862429
global waypoint_lo
waypoint_lo = -8.123523160372107

global ser
ser = serial.Serial ("/dev/ttyACM1")
global lat_in_degrees
global long_in_degrees
lat_in_degrees = 0
long_in_degrees = 0

global ber
ber = serial.Serial ("/dev/ttyACM0")

PWM_pin = 32
GPIO.setwarnings(False)
GPIO.setmode(GPIO.BOARD)
GPIO.setup(PWM_pin,GPIO.OUT)
global pi_pwm
pi_pwm = GPIO.PWM(PWM_pin,1000)
pi_pwm.start(0)

global foward
global back
global right
global left
right = 40
```

```
left = 36
foward = 37
back = 35
GPIO.setup(foward, GPIO.OUT)
GPIO.setup(back, GPIO.OUT)
GPIO.setup(right, GPIO.OUT)
GPIO.setup(left, GPIO.OUT)

GPIO.output(right, False)
GPIO.output(left, False)

global p
p = 0

NMEA_buff = 0

def go_forward():

    GPIO.output(foward, GPIO.HIGH)
    GPIO.output(back, GPIO.LOW)
    pi_pwm.ChangeDutyCycle(40)

def go_back():

    GPIO.output(foward, GPIO.LOW)
    GPIO.output(back, GPIO.HIGH)
    pi_pwm.ChangeDutyCycle(40)

def go_left():

    GPIO.output(foward, GPIO.HIGH)
    GPIO.output(right, GPIO.LOW)
    GPIO.output(left, GPIO.HIGH)
    pi_pwm.ChangeDutyCycle(30)

def go_right():

    GPIO.output(foward, GPIO.HIGH)
    GPIO.output(right, GPIO.HIGH)
    GPIO.output(left, GPIO.LOW)
    pi_pwm.ChangeDutyCycle(30)
```

```
def stop():

    GPIO.output(right, GPIO.LOW)
    GPIO.output(left, GPIO.LOW)
    pi_pwm.ChangeDutyCycle(0)

def average_volante():

    adc = Adafruit_ADS1x15.ADS1115()
    adc = Adafruit_ADS1x15.ADS1115(address=0x48, busnum=1)
    GAIN = 2/3
    value = adc.read_adc(3, gain=GAIN)
    return value

def GPS_Info():
    global NMEA_buff
    global lat_in_degrees
    global long_in_degrees

    gpgga_info = "$GPGGA,"  #Open port with baud rate
    GPGGA_buffer = 0
    NMEA_buff = 0


    while True:
        received_data = (str)(ser.readline())                #read NMEA string received
        GPGGA_data_available = received_data.find(gpgga_info)   #check for NMEA GPGGA
string
        if (GPGGA_data_available>0):
            GPGGA_buffer = received_data.split("$GPGGA,",1)[1]   #store data coming after
"$GPGGA," string
            NMEA_buff = (GPGGA_buffer.split(','))                #store comma separated data in
buffer
            nmea_time = []
            nmea_latitude = []
            nmea_longitude = []
            nmea_time = NMEA_buff[0]                #extract time from GPGGA string
            nmea_latitude = NMEA_buff[1]             #extract latitude from GPGGA string
            nmea_longitude = NMEA_buff[3]            #extract longitude from GPGGA string
```

```
        lat = float(nmea_latitude)              #convert string into float for calculation
        longi = float(nmea_longitude)           #convertr string into float for calculation

        lat_in_degrees = convert_to_degrees(lat)    #get latitude in degree decimal format
        long_in_degrees = convert_to_degrees(longi) #get longitude in degree decimal format

    return (lat_in_degrees,long_in_degrees)


def convert_to_degrees(raw_value):
    decimal_value = raw_value/100.00
    degrees = int(decimal_value)
    mm_mmmm = (decimal_value - int(decimal_value))/0.6
    position = degrees + mm_mmmm
    position = "%.8f" %(position)
    return position

def distance_GPS(latitude, longitude, WP_la, WP_lo):
    global dist_calc
    loc1 = (float(latitude), float(longitude))
    loc2 = (WP_la, -WP_lo)

    dist_calc = hs.haversine(loc1,loc2,unit=Unit.METERS)

    return (dist_calc)


def calculate_initial_compass_bearing(pointA_lat, pointA_long, pointB_lat, pointB_long):

    lat1 = math.radians(float(pointA_lat))
    lat2 = math.radians(pointB_lat)

    diffLong = math.radians(pointB_long - (-float(pointA_long)))

    x = math.sin(diffLong) * math.cos(lat2)
    y = math.cos(lat1) * math.sin(lat2) - (math.sin(lat1) * math.cos(lat2) * math.cos(diffLong))

    initial_bearing = math.atan2(x, y)

    initial_bearing = math.degrees(initial_bearing)
    compass_bearing = (initial_bearing + 360) % 360
```

```
    return compass_bearing

if __name__ == "__main__":
    while True:
        p = average_volante()
        lat_in_degrees, long_in_degrees = GPS_Info()
        a = calculate_initial_compass_bearing(lat_in_degrees, long_in_degrees, waypoint_la, waypoint_lo )
        d = distance_GPS(lat_in_degrees, long_in_degrees, waypoint_la, waypoint_lo)
        ser_bytes = ber.readline()
        b = float(ser_bytes[0:len(ser_bytes)-2].decode("utf-8"))
        if(lat_in_degrees != 0 and long_in_degrees != 0):
            if(d > 0):
                if(a-b > 6 and a-b < 45):
                    go_right()
                elif(a-b > -6 and a-b < -45):
                    go_left()
                elif(a-b < 180 and a-b > 46):
                    if(p > 7750):
                        go_right()
                        time.sleep(1)
                        go_forward()
                    elif(p < 20680):
                        go_left()
                        time.sleep(1)
                        go_back()
                elif(a-b < -180 and a-b < -46):
                    if(p < 20680):
                        go_left()
                        time.sleep(1)
                        go_forward()
                    elif(p > 7750):
                        go_right()
                        time.sleep(1)
                        go_back()
            elif(d <= 0):
                stop()
```

ANEXO 2

```
# Import packages
import os
import argparse
import cv2
import cv2 as cv
import numpy as np
import sys
import time
from threading import Thread
import importlib.util
import math
import numpy
import Adafruit_ADS1x15
from bluedot import BlueDot
from signal import pause
import utilGPIO as ut
ut.inite_gpio()


global x_deviation, ymax, tolerance, tolerance_x, ymin
tolerance=100
x_deviation=0
ymax=0

# Define VideoStream class to handle streaming of video from webcam in separate processing thread
# Source - Adrian Rosebrock, PyImageSearch: https://www.pyimagesearch.com/2015/12/28/increasing-raspberry-pi-fps-with-python-and-opencv/
class VideoStream:
    """Camera object that controls video streaming from the Picamera"""
    def __init__(self,resolution=(340,180),framerate=30):
        # Initialize the PiCamera and the camera image stream
        self.stream = cv2.VideoCapture('/dev/video0')#cv2.CAP_DSHOW)
        ret = self.stream.set(cv2.CAP_PROP_FOURCC, cv2.VideoWriter_fourcc(*'MJPG'))
        ret = self.stream.set(3,resolution[0])
        ret = self.stream.set(4,resolution[1])
```

```
        # Read first frame from the stream
        (self.grabbed, self.frame) = self.stream.read()

         # Variable to control when the camera is stopped
        self.stopped = False

    def start(self):
         # Start the thread that reads frames from the video stream
        Thread(target=self.update,args=()).start()
        return self

    def update(self):
        # Keep looping indefinitely until the thread is stopped
        while True:
            # If the camera is stopped, stop the thread
            if self.stopped:
                # Close camera resources
                self.stream.release()
                return

            # Otherwise, grab the next frame from the stream
            (self.grabbed, self.frame) = self.stream.read()

    def read(self):
         # Return the most recent frame
        return self.frame

    def stop(self):
         # Indicate that the camera and thread should be stopped
        self.stopped = True


def get_delay(deviation):

    deviation=abs(deviation)

    if(deviation>=40):
        d=4
    elif(deviation>=35 and deviation<40):
        d=3
```

```
    elif(deviation>=20 and deviation<35):
        d=2
    else:
        d=1
    return d


# Define and parse input arguments
parser = argparse.ArgumentParser()
parser.add_argument('--modeldir', help='Folder the .tflite file is located in',
                    required=True)
parser.add_argument('--graph', help='Name of the .tflite file, if different than detect.tflite',
                    default='detect.tflite')
parser.add_argument('--labels', help='Name of the labelmap file, if different than labelmap.txt',
                    default='labelmap.txt')
parser.add_argument('--threshold', help='Minimum confidence threshold for displaying detected objects',
                    default=0.5)
parser.add_argument('--resolution', help='Desired webcam resolution in WxH. If the webcam does not support the resolution entered, errors may occur.',
                    default='780x520')
parser.add_argument('--edgetpu', help='Use Coral Edge TPU Accelerator to speed up detection',
                    action='store_true')

args = parser.parse_args()

MODEL_NAME = args.modeldir
GRAPH_NAME = args.graph
LABELMAP_NAME = args.labels
min_conf_threshold = float(args.threshold)
resW, resH = args.resolution.split('x')
imW, imH = int(resW), int(resH)
use_TPU = args.edgetpu

# Import TensorFlow libraries
# If tflite_runtime is installed, import interpreter from tflite_runtime, else import from regular tensorflow
# If using Coral Edge TPU, import the load_delegate library
pkg = importlib.util.find_spec('tflite_runtime')
```

```
if pkg:
    from tflite_runtime.interpreter import Interpreter
    if use_TPU:
        from tflite_runtime.interpreter import load_delegate
else:
    from tensorflow.lite.python.interpreter import Interpreter
    if use_TPU:
        from tensorflow.lite.python.interpreter import load_delegate

# If using Edge TPU, assign filename for Edge TPU model
if use_TPU:
    # If user has specified the name of the .tflite file, use that name, otherwise use default 'edgetpu.tflite'
    if (GRAPH_NAME == 'detect.tflite'):
        GRAPH_NAME = 'edgetpu.tflite'

# Get path to current working directory
CWD_PATH = os.getcwd()

# Path to .tflite file, which contains the model that is used for object detection
PATH_TO_CKPT = os.path.join(CWD_PATH,MODEL_NAME,GRAPH_NAME)

# Path to label map file
PATH_TO_LABELS = os.path.join(CWD_PATH,MODEL_NAME,LABELMAP_NAME)

# Load the label map
with open(PATH_TO_LABELS, 'r') as f:
    labels = [line.strip() for line in f.readlines()]

def average_volante():
    adc = Adafruit_ADS1x15.ADS1115()
    adc = Adafruit_ADS1x15.ADS1115(address=0x48, busnum=1)
    GAIN = 2/3
    value = adc.read_adc(3, gain=GAIN)
    return value

def countdown(t):

    while t:
        mins, secs = divmod(t, 60)
        timer = '{:02d}:{:02d}'.format(mins, secs)
```

```
        #print(timer, end="\r")
        time.sleep(1)
        t -= 1


def following_certo():
        global tolerance_x
        # Have to do a weird fix for label map if using the COCO "starter model" from
        # First label is '???', which has to be removed.
        if labels[0] == '???':
                del(labels[0])

        # Load the Tensorflow Lite model.
        # If using Edge TPU, use special load_delegate argument
        if use_TPU:
                interpreter = Interpreter(model_path=PATH_TO_CKPT,

experimental_delegates=[load_delegate('libedgetpu.so.1.0')])
                print(PATH_TO_CKPT)
        else:
                interpreter = Interpreter(model_path=PATH_TO_CKPT)

        interpreter.allocate_tensors()

        # Get model details
        input_details = interpreter.get_input_details()
        output_details = interpreter.get_output_details()
        height = input_details[0]['shape'][1]
        width = input_details[0]['shape'][2]

        floating_model = (input_details[0]['dtype'] == np.float32)

        input_mean = 127.5
        input_std = 127.5

        # Check output layer name to determine if this model was created with TF2 or TF1,
        # because outputs are ordered differently for TF2 and TF1 models
        outname = output_details[0]['name']

        if ('StatefulPartitionedCall' in outname): # This is a TF2 model
                boxes_idx, classes_idx, scores_idx = 1, 3, 0
```

```
        else: # This is a TF1 model
                boxes_idx, classes_idx, scores_idx = 0, 1, 2

        # Initialize frame rate calculation
        frame_rate_calc = 1
        freq = cv2.getTickFrequency()

        # Initialize video stream
        videostream = VideoStream(resolution=(imW,imH),framerate=30).start()
        time.sleep(1)
        tx = 0
        t = 0
        y_min = 0

        #for frame1 in camera.capture_continuous(rawCapture,
format="bgr",use_video_port=True):
        while True:
                tx = []
                t = []
                y_min = []
                # Start timer (for calculating frame rate)
                t1 = cv2.getTickCount()

                # Grab frame from video stream
                frame1 = videostream.read()
                if frame1 is None:
                        break
                # Acquire frame and resize to expected shape [1xHxWx3]
                frame = frame1.copy()
                frame_rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
                frame_resized = cv2.resize(frame_rgb, (width, height))
                input_data = np.expand_dims(frame_resized, axis=0)

                # Normalize pixel values if using a floating model (i.e. if model is non-quantized)
                if floating_model:
                        input_data = (np.float32(input_data) - input_mean) / input_std

                # Perform the actual detection by running the model with the image as input
                interpreter.set_tensor(input_details[0]['index'],input_data)
                interpreter.invoke()
```

```
        # Retrieve detection results
        boxes = interpreter.get_tensor(output_details[boxes_idx]['index'])[0] # Bounding box coordinates of detected objects
        classes = interpreter.get_tensor(output_details[classes_idx]['index'])[0] # Class index of detected objects
        scores = interpreter.get_tensor(output_details[scores_idx]['index'])[0] # Confidence of detected objects

        pv = average_volante()

        # Loop over all detections and draw detection box if confidence is above minimum threshold
        for i in range(len(scores)):
            if ((scores[i] > min_conf_threshold) and (scores[i] <= 1.0) and labels[int(classes[i])] == 'person'):
                global tolerance_x

                ymin = int(max(1,(boxes[i][0] * imH)))
                xmin = int(max(1,(boxes[i][1] * imW)))
                ymax = int(min(imH,(boxes[i][2] * imH)))
                xmax = int(min(imW,(boxes[i][3] * imW)))

                x_diff=xmax-xmin
                y_diff=ymax-ymin

                obj_x_center=int(xmin+(x_diff/2))
                obj_x_center=round(obj_x_center,3)
                obj_y_center=int(ymin+(y_diff/2))
                obj_y_center=round(obj_y_center,3)

                tolerance_x = obj_x_center-(int(imW/2)-100)
                tolerance_y = int(imH)-ymax

                cv2.rectangle(frame, (xmin,ymin), (xmax,ymax), (10, 255, 0), 2)

                # Draw label
                object_name = labels[int(classes[i])] # Look up object name from "labels" array using class index
                label = '%s: %d%%' % (object_name, int(scores[i]*100)) # Example: 'person: 72%'
```

```
                    labelSize, baseLine = cv2.getTextSize(label, cv2.FONT_HERSHEY_SIMPLEX, 0.7, 2) # Get font size
                    label_ymin = max(ymin, labelSize[1] + 10) # Make sure not to draw label too close to top of window
                    cv2.rectangle(frame, (xmin, label_ymin-labelSize[1]-10), (xmin+labelSize[0], label_ymin+baseLine-10), (255, 255, 255), cv2.FILLED) # Draw white box to put label text in
                    cv2.putText(frame, label, (xmin, label_ymin-7), cv2.FONT_HERSHEY_SIMPLEX, 0.7, (0, 0, 0), 2) # Draw label text

                    print(tolerance_x)
                    print(ymin)
                    print("pv", pv)
                    if(tolerance_x < tolerance and tolerance_x > -tolerance):
                        print("estou direito")
                        if(ymin < 30):
                            ut.Stop()
                            countdown(2)
                            ut.SRight()
                            countdown(10)
                            ut.Forward()
                            countdown(4)
                            ut.Stop()
                            ut.SLeft()
                            countdown(10)
                            ut.Stop()
                            ut.Back()
                            countdown(6)
                            ut.Stop()
                            if pv >= 12100 and pv <= 12900:
                                ut.Stop()

                            elif pv < 12100:
                                countdown(1)
                                ut.SLeft()

                            elif pv > 12900:
                                countdown(1)
                                ut.SRight()

                        else:
```

```
                                ut.Stop()
                                time.sleep(0.5)
                                ut.Forward()

                                if pv >= 12000 and pv <= 14000:
                                        print("ajuste em frente")
                                        ut.Stop()
                                        time.sleep(0.5)
                                        ut.Forward()

                                elif pv < 12100:
                                        print("ajuste a esquerda")
                                        ut.Stop()
                                        countdown(1)
                                        ut.Left()

                                elif pv > 12900:
                                        print("ajuste a direita")
                                        ut.Stop()
                                        countdown(1)
                                        ut.Right()

                        elif(tolerance_x >= tolerance):
                                ut.Right()
                                time.sleep(0.5)
                                ut.Stop()

                        elif(tolerance_x <= -1*tolerance):
                                ut.Left()
                                time.sleep(0.5)
                                ut.Stop()
                        cv2.line(frame, (int(imW/2)-100,imH), (obj_x_center, obj_y_center),
(255,255,0), 2)
                # Draw framerate in corner of frame
                #cv2.putText(frame,'FPS:
{0:.2f}'.format(frame_rate_calc),(30,50),cv2.FONT_HERSHEY_SIMPLEX,1,(255,255,0),2,c
v2.LINE_AA)
                cv2.imshow('Object detector', frame)
                # Calculate framerate
                t2 = cv2.getTickCount()
                time1 = (t2-t1)/freq
```

```
            frame_rate_calc= 1/time1

            # Press 'q' to quit
            if cv2.waitKey(1) == ord('q'):
                break
        # Clean up
        cv2.destroyAllWindows()
        videostream.stop()

def precision_angle():
        global average
        average = 0
        videostream = VideoStream(resolution=(imW,imH),framerate=30).start()
        while True:

            img = videostream.read()
            if img is None:
                break

            color = (222,49,99)
            thickness=2
            height = imH
            width = imW

            start_point=(int(width/2),0)
            end_point = (int(width/2),height)
            image = cv2.line(img, start_point, end_point, color, thickness)
            gamma = 0.09
            # build a lookup table mapping the pixel values [0, 255] to
            # their adjusted gamma values
            invGamma = 1.0/gamma
            table = np.array([((i / 255.0) ** invGamma) * 255
                for i in np.arange(0, 256)]).astype("uint8")
            # apply gamma correction using the lookup table
            darkch = cv2.LUT(img, table)

            #try to reduce the glare to focus just in the project light
            # convert to gray
            gray = cv2.cvtColor(darkch, cv2.COLOR_BGR2GRAY)

            # threshold grayscale image to extract glare
```

```
mask = cv2.threshold(gray, 220, 255, cv2.THRESH_BINARY)[1]

# Optionally add some morphology close and open, if desired
kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (7,7))
mask = cv2.morphologyEx(mask, cv2.MORPH_CLOSE, kernel, iterations=1)

# use mask with input to do inpainting
result = cv2.inpaint(darkch, mask, 21, cv2.INPAINT_TELEA)
#-----------------------------------------------------------------
gray = cv2.cvtColor(result, cv2.COLOR_BGR2GRAY)
(minVal, maxVal, minLoc, maxLoc) = cv2.minMaxLoc(gray)
gray = cv2.circle(img, maxLoc,(20), (255, 0, 0), 2)

ser = maxLoc
a = np.array([int(width/2),0])
b = np.array([int(width/2),height])
c = np.array([ser[0],ser[1]])

image = cv2.line(img, end_point, maxLoc, color, thickness)

ba = a - b
bc = c - b
pv = average_volante()
if(ser[0] > int(width/2)):
        cosine_angle = np.dot(ba, bc) / (np.linalg.norm(ba) * np.linalg.norm(bc))
        angle = np.arccos(cosine_angle)
        angulo = np.degrees(angle)
        accumulator = 0
        for _ in range(10):
                reading = angulo
                accumulator += reading
        average = accumulator / 30
        #yield average
elif(ser[0] < int(width/2)):
        cosine_angle = np.dot(ba, bc) / (np.linalg.norm(ba) * np.linalg.norm(bc))
        angle = np.arccos(cosine_angle)
        angulo = -np.degrees(angle)
        accumulator = 0
        for _ in range(10):
                reading = angulo
                accumulator += reading
```

```
                average = accumulator / 30

            if average < -15:
                ut.Left()
                time.sleep(0.5)
                ut.Stop()
            elif average > 15:
                ut.Right()
                time.sleep(0.5)
                ut.Stop()
            elif average >= -15 and average < 15:
                ut.Stop()
                time.sleep(0.5)
                ut.Forward()

                if pv >= 12100 and pv <= 12900:
                    ut.Stop()
                    time.sleep(0.5)
                    ut.Forward()

                elif pv < 12100:
                    ut.Stop()
                    time.sleep(0.5)
                    ut.Left()

                elif pv > 12900:
                    ut.Stop()
                    time.sleep(0.5)
                    ut.Right()

            cv2.imshow("original", img)

                #This breaks on 'q' key
            if cv2.waitKey(1) & 0xFF == ord('q'):
                break
        videostream.stop()
        cv2.destroyAllWindows()

if __name__ == '__main__':
    #following_certo()
    #precision_angle()
```