UNIVERSIDADE
BEIRA INTERIOR

Sistema de navegação autónoma baseada em GPS para rover robótico multitarefa para aplicações agrícolas e aplicação web de realidade aumentada para apoio à supervisão

Gil Ayres Menezes

Dissertação para obtenção do Grau de Mestre em
Engenharia Eletromecânica
(2º ciclo de estudos)

Orientador: Prof. Doutor Pedro Dinis Gaspar

Outubro de 2021

Agradecimentos

Tem sido um grande prazer integrar a comunidade académica da Universidade da Beira Interior. Foi um período muito enriquecedor em que tive a oportunidade de me desenvolver de forma pessoal, profissional e científica. Por isso, sou muito grato a toda comunidade académica em especial aos professores, coordenadores, funcionários e alunos do mestrado em Engenharia Eletromecânica.

Minha gratidão ao professor e orientador Pedro Dinis Gaspar por me ter concedido a oportunidade de participar desse projeto, e também, por todo apoio, zelo e dedicação para que este trabalho pudesse ser desenvolvido da melhor forma possível.

Esta investigação faz parte das atividades do projeto PrunusBot – Sistema robótico aéreo autónomo de pulverização controlada e previsão de produção frutícola, Operação nºPDR2020-101-03158(líder), Consórcio nº340, Iniciativa nº 140, promovido pelo PDR2020 e cofinanciado pelo FEADER e União Europeia no âmbito do Programa Portugal 2020.

Agradeço também a minha mãe Edione Calazans e ao meu padrasto Neilton Calazans pelo apoio incondicional as minha aventuras académicas e por todo incentivo concedido para que eu pudesse estar a realizar esse trabalho. Não tenho palavras para dizer o quanto aprendi com esse maravilhoso casal.

A minha esposa Pollyanna Marsi por todo apoio e compreensão, por todas as contribuições, orientações e críticas extremamente úteis sobre o conteúdo dessa dissertação de mestrado. Sei o quanto sou feliz por encontrar na mesma pessoa, alguém cujo conselho, companheirismo e senso de humor eu tanto admiro, e que também é o amor da minha vida.

Resumo

Por consequência do aumento da população mundial e a crescente procura por alimentos, há uma grande necessidade de aumentar a produtividade da agricultura. Entretanto, o problema é agravado com a migração populacional das áreas rurais paras as cidades, o que causa uma diminuição da mão-de-obra no setor agrícola. Neste intuito, dentro do conceito de agricultura 4.0, a introdução de robôs autónomos nas atividades agrícolas poderá enfrentar esses problemas, apoiando a falta de mão-de-obra crescente e promovendo o aumento da produtividade agrícola. Algumas soluções de robôs autónomos aplicados a agricultura são expostas nessa dissertação, contudo, ainda será necessário um esforço maior da comunidade científica para desenvolver soluções de robôs com maior automação e que viabilizem sua utilização com a diminuição dos custos de produção.

Esse trabalho expõe o algoritmo utilizado para realizar a navegação autónoma, baseada em GPS, do Rover Robótico para Aplicações Agrícolas (R2A2) multitarefa destinado a realizar a pulverização de herbicida. Com isso, são expostos os métodos clássicos de programação de sistemas de locomoção autónoma, bem como são expostos todos os sensores e atuadores necessário para a realização dessa tarefa. Também foi desenvolvida uma aplicação web de realidade aumentada para auxiliar na supervisão do veículo autónomo. Foi desenvolvido um código em C/C++ para a movimentação autónoma da plataforma robótica utilizando como microprocessador principal um Arduino Mega2560 e a aplicação web de realidade aumentada baseada em posicionamento que foi desenvolvida utilizando a bibliotecas AR.JS e o Framework A-FRAME compilados em um código HTML. A aplicação foi testada num pomar de pessegueiros tendo apresentado uma média de acertos aproximada de 94% o que revela a precisão da solução tecnológica desenvolvida. São expostos também, os resultados e conclusões do algoritmo de movimentação autónoma e da aplicação web.

Palavras-chave

Sistema de Movimentação Autónoma, Robótica, Robô terrestre, Agricultura 4.0, Algoritmo de movimentação autónoma, Computação Física, Desenvolvimento web, Realidade Aumentada.

Abstract

As a result of the increase in the world population and the growing demand for food, there is a great need to increase the productivity of agriculture. However, the problem is aggravated by the migration of population from rural areas to cities, which causes a decrease in the labor force in the agricultural sector. Within the concept of agriculture 4.0, the introduction of autonomous robots in agricultural plantations will be able to face these problems, solving the growing lack of labor and promoting an increase in agricultural productivity. Some autonomous robot solutions applied to agriculture are exposed in this dissertation, however, it will still be necessary a greater effort from the scientific community to develop robot solutions with a higher level of automaticity and that enable their use with a reduction in production costs.

This work exposes the algorithm used to perform the autonomous GPS-based navigation of the Robotic Rover for Agricultural Applications (R2A2) multitasking for carrying out herbicide spraying. Thus, the classic methods of programming autonomous locomotion systems are exposed, as well as all the sensors and actuators necessary to perform this task are exposed. An augmented reality web application was also developed to assist in the supervision of the autonomous vehicle. For this, a code was made in C/C++ for the autonomous movement of the robotic platform using an Arduino Mega2560 as the main microprocessor and the augmented reality web application based on positioning was developed using the AR.JS libraries and the A-FRAME Framework compiled in an HTML code. The application was tested in a peach orchard presenting an average hit score around 94% revealing the precision of the technological solution developed. The results and conclusions of the autonomous movement algorithm and the web application are also exposed.

Keywords

Autonomous Movement System, Robotics, Land Robot, Agriculture 4.0, Autonomous Movement Algorithm, Physical Computing, web development, Augmented Reality,

Índice

Lista de Figuras

Lista de Tabelas

Nomenclatura

Geral:

a	Aceleração, [m·s^{-2}];
B	Matriz de controlo;
d	Distância, [m];
h	Ângulo de direção, [rad];
H	Matriz de transformação
g	Aceleração da gravidade, {9,81 m·s^{-2}};
f	Função degrau de ativação;
F	Matriz que denota a dinâmica do sistema;
k	Intervalo de tempo;
P_k	Matriz de covariância prevista;
Pot	Potência elétrica, [W];
Q	Covariância de ruido;
R	Raio da Terra, [m];
t	Tempo, [s];
R_k	Matriz de covariância do ruído de medição;
T	Temperatura, [K ou ºC];
U	Diferença de potencial, [V];
$\hat{x}_k$	Estado estimado do sistema;
w	Vetor de pesos;
x	Vetor de entradas;
y	Vetor de saída;
z_k	Vetor de medição;

Índices inferiores:

GND	Neutro;
in	Entrada;
out	Saída;
RX	Pino de receção UART;
TX	Pino de transmissão UART;

Vcc	Pino de alimentação;

Simbologia grega:

θ	Latitude, [rad];
λ	Longitude, [rad];

Acrónimos:

2D	Bidimensional;
3D	Tridimensional;
ADC	*Analog-Digital Converter*;
CSS	Linguagem Cascading Style Sheets;
DC	*Direct current*;
EKF	*Extended Kalman Filter* - Filtro de Kalman Estendido;
GPS	*Global Positioning System*;
HTML	*Linguagem Hyper Text Markup Language*;
IDE	Ambiente de desenvolvimento integrado;
IP	*Internet Protocol*;
LIDAR	*Light Detection And Ranging*;
MEMS	*Micro-ElectroMechanical Machines*;
PI	Proporcional-Integral;
PID	Proporcional-Integral-Derivativo;
RA	Realidade Aumentada;
RTK	*Real Time Kinematics*;
SDA	*Serial Data*;
SLAM	Localização e mapeamento simultâneos;
SCL	*Serial Clock*;
TCP	*Transmission Control Protocol*;
UBI	Universidade da Beira Interior;

1. Introdução

Os sistemas robóticos autónomos móveis são utilizados em diferentes ambientes. Estes robôs desempenham um papel significativo em muitas aplicações agrícolas, o que reduz o trabalho humano e, consequentemente, aumenta a segurança das operações (Shalal, et al., 2013). Por exemplo, em atividades agrícolas como plantação, pulverização, fertilização, cultivo, colheita, desbaste, remoção de ervas daninhas e inspeção, a necessidade de sistemas de navegação autónomo fez com que, nos últimos anos, muitas investigações envolvessem veículos mais inteligentes e adaptáveis (Shalal, et al., 2013).

Projetar robôs móveis para serem utilizados em ambientes externos é uma atividade muito desafiadora. A navegação em ambiente agrícola apresenta variadas dificuldades, devido às mudanças climáticas e variação da natureza do terreno e da vegetação. Por conta disso, é requerido a criação de um sistema de controlo e deteção eficiente. Além disso, a autonomia do robô é obtida por meio da deteção do ambiente, através de sensores, e do emprego de algoritmos apropriados para realização de cada tarefa (Shalal, et al., 2013).

Veículos terrestres e aéreos não tripulados são utilizados na agricultura para automatizar uma série de operações que exigem muita mão-de-obra, exigem recurso e consomem tempo (Guzueva, et al., 2020). Entretanto, os principais entraves à aplicação efetiva da robótica neste setor são a dificuldade de implementação de automatismo, pois, as tarefas necessárias a desempenhar incluem uma série de complexidades causadas, a priori, pela variedade de condições extrínsecas, como condições ambientais e obstáculos no terreno. A estas condições intrínsecas têm que ser adicionadas outras, como sejam o crescimento irregular das culturas e a heterogeneidade de formatos, cor e tamanhos.

1.1. Enquadramento

A população mundial está a aumentar constantemente e a necessidade de alimentos e produtos agrícolas tem uma taxa de crescimento estimada de 70% até 2050 (Ruane & Sonnino, 2011). Por outro lado, a mão de obra qualificada no mundo rural está a diminuir por conta de outros tipos de indústrias que atraem a força de trabalho que antes pertencia a agricultura (Hassall, 2010).

Decerto, com o aumento da população mundial, é previsto que em 2050 o número de habitantes do planeta seja superior a 10 mil milhões. Por consequência, o crescimento na produção agrícola

terá que ser continuado. Portanto, isso combinado com uma força de trabalho rural cada vez menor, conduz à necessidade de aumentar a eficiência no campo, que consequentemente leva ao aumento do nível de automação em campo (Hassall, 2010).

A fim de aumentar a produtividade, foram realizas mudanças nos tipos de máquinas usadas nas explorações agrícolas mais modernas. Tratores *drive-by-wire* que utilizam GPS e recebem, pela internet, dados e um mapeamento preciso, já foram capazes de aumentar a produtividade agrícola. Porém, futuramente será necessário maior aumento da produtividade. Estes podem ser conseguidos com aplicação da automação e robótica na agricultura, juntamente com a diminuição dos custos de produção (Mousazadeh, 2013).

1.2. O problema em estudo e a sua relevância

Os sistemas robóticos automatizados têm vindo a transformar inúmeras indústrias e impactando vários setores da economia, pois possibilitam aumento da produtividade, inclusive em áreas caracterizadas pela baixa produtividade, como é o caso da agroindústria (Duckett, et al., 2018). O crescimento da população global, mudanças climáticas e a migração pressionam os produtores de alimentos para que aumentem a produtividade. Adicionalmente, os empregos nessa atividade exigem um grande esforço e um trabalho bastante repetitivo, além de ser operados em ambientes adversos. Por conta dessas circunstâncias, a aplicação de robôs autónomos pode transformar o setor agroalimentar. A implementação dessas tecnologias pode adicionar, em 13 anos, cerca de 67 bilhões de euros na economia do Reino Unido além de atrair trabalhadores mais qualificado para o setor (Maier, 2017). Têm sido anunciados grandes investimentos para apoiar a inovação e a robótica na indústria agroalimentar, na continuação de outros investimentos para financiar o programas de tecnologias aplicadas a agricultura (Duckett, et al., 2018).

Além de proporcionar benefícios económicos pelo aumento da produtividade, a utilização de sistemas robóticos autónomos propicia benefícios sociais e ambientais. No caso dos solos, a utilizações de grandes máquinas agrícolas causam danos como compactação e erosão. A compactação do solo pode ocasionar a diminuição da produtividade nas lavouras e, consequentemente, causa o aumento da utilização de recursos naturais e fertilizantes. As perdas económicas ocasionadas no Pais de Gales por conta da compactação dos solos e erosão foi estimado em cerca de £ 1,2 bilhões (Graves, et al., 2015). Assim sendo, a compactação do solo pode ser minimizada pela utilização de pequenos robôs móveis autónomos, ao invés de grandes máquinas agrícolas. Com isso, existe a tendência que os robôs leves substituam os tratores tradicionais de elevada massa. O que permite a redução gradual da compactação do solo e, também, a re-aeração que aumenta a qualidade do solo.

A agricultura representa 70% do consumo global de água doce. Em contraponto, 4 bilhões de pessoas vivem em regiões globais com escassez de água (Mekonnen & Hoekstra, 2016). Além

disso, o clima é, de certa forma, imprevisível e pode ocasionar longos períodos de seca ou enchentes nas regiões de produção agrícola, o que pode prejudicar toda a plantação durante muito tempo. Com isso, robôs autónomos estão a ser implementados para auxiliar os produtores rurais a medir, mapear e otimizar a água utilizada para irrigação.

A utilização de máquinas mais inteligentes, visando a redução de utilização e o melhor direcionamento de recursos naturais na agricultura, fez surgir uma nova gama de equipamentos agrícolas flexíveis e com base em pequenos robôs inteligentes que otimizam a produção agrícola. Também existe um potencial considerável para utilização de sistemas autónomos na agricultura, pois, eles são capazes de trabalhar a noite, trabalham em solos húmidos e etc (Naik, et al., 2016).

Os dados sensoriais captados por estes robôs podem fornecer informações sobre o solo, sementes, pecuária, safras, custos, uso de água e fertilizante. Além disso, tecnologias de IoT (*Internet of Things*) com custo reduzido e análises avançadas já estão a apoiar agricultores a analisarem dados sobre o clima, temperatura e humidade. Com isso, os robôs autónomos podem fornecer informações para tomada de decisão dos produtores visando a otimização do rendimento, melhorar o planeamento da utilização dos recursos disponíveis (Duckett, et al., 2018).

1.3. Objetivos e contribuição da dissertação

Essa dissertação tem como objetivo providenciar um sistema de locomoção autónoma baseado em GPS a uma plataforma robótica terrestre destinada a utilização em pomares de pessegueiros, na sequência do projeto PrunusBOT. Entretanto, esta plataforma robótica poderá ser empregue em outras culturas como as de maça, pera ou laranja, quando utilizados outros algoritmos de deteção de frutas. Este trabalho também visa dotar o robô autónomo com outros sensores e visão de máquina a fim de tornar a navegação mais fiável.

O trabalho de investigação é resultado de umas das atividades do projeto PrunusBOT, que propõe o desenvolvimento de sistemas robóticos destinados à inovação tecnológica na fruticultura, especificamente de pomares de prunóideas da região da Beira Interior. Esta plataforma tem como proposta a sustentabilidade e visa causar o mínimo de impacto ao meio ambiente.

O robô autónomo terá que desempenhar duas funções primordiais na cultura de pêssegos. A primeira é a aplicação precisa de herbicida para combater as infestantes, de forma que o impacto ambiental seja minimizado, reduzindo a utilização de defensivos agrícolas. Dessa forma, o uso direcionado e otimizado deste produto implicará numa maior rentabilidade para o produtor rural. E a segunda, é retirar frutos caídos do chão, com intuito de reaproveitá-los para alimentação de ruminantes. Adicionalmente, o robô recolherá imagens para ser tratadas por algoritmo de previsão de estimativa de produção e também para gerar informações sobre a saúde da plantação. Será integrado outro algoritmo baseado em web de Realidade Aumentada que imprimirá na tela

avisos que irão auxiliar os supervisores das operações para a melhor tomada de decisão para ações inerentes ao processo de cultivo.

1.4. Visão geral e organização da dissertação

Este trabalho está organizado em quatro capítulos onde se disserta sobre a aplicação da robótica autónoma na agricultura. Posteriormente, serão apresentados os equipamentos que serão utilizados para formação do sistema de locomoção autónoma, bem como as suas características e esquemas teóricos. E finalmente, será apresentado o algoritmo utilizado e seus resultados, concluindo com a viabilidade de sua utilização.

Sendo assim, o Capítulo 1 contém o enquadramento e os objetivos desse trabalho.

O Capítulo 2 discursa sobre o estado da arte da utilização de robôs autónomos na agricultura, bem como as características e especificações técnicas dos seus sistemas.

O Capítulo 3 descreve todas as técnicas utilizadas para compor o sistema de navegação autónoma do robô e a aplicação de realidade aumentada.

O Capítulo 4 descreve como foi realizada a codificação do algoritmo utilizado para o controlo do sistema e o código de desenvolvimento da aplicação web de realidade aumentada.

E finalmente, o Capítulo 5 apresenta uma análise conclusiva do trabalho que foi desenvolvido, além de mostrar ideias e orientações a fim de auxiliar na produção de trabalhos futuros.

2. Estado da Arte

Na década de 20, começou-se a estudar a orientação automática de veículos e, desde então, foram realizadas uma série de inovações. Entretanto, veículos agrários totalmente autónomos ainda são raros de ser encontrados em estado operacional (Yaghoubi, et al., 2013). Isto é, desde a década de 50 são estudados protótipos de tratores sem motoristas usando sistema de orientação de cabo líder (Li, et al., 2009).

Na década de 1980, com o incremento da computação nas máquinas e a possibilidade de combinar sensores de imagens, surgiram sistemas de orientação baseados em visão. Por outro lado, começaram a ser desenvolvidos sistemas de colheita robótica. Um destes foi realizado com sucesso na Universidade da Flórida, pois demonstrou a capacidade de um robô remover frutas da árvore. Entretanto, não demonstrou a aplicação prática da colheita robótica, pois para conseguir um número aceitável de frutas por unidade de tempo, a máquina precisaria ter mais braços (Harrell, et al., 1990).

Os potenciais benefícios de veículos agrícolas automatizados incluem melhoria na produtividade, e na precisão das atividades. Somado a isso, o rápido avanço da eletrónica e da computação renovou o interesse no desenvolvimento dos sistemas de orientação de veículos. Com isso, vários tipos de tecnologias de orientação foram desenvolvidos, por exemplo, orientação mecânica, ótica, por onda de rádio e por ultrassom (Shalal, et al., 2013). Com isso, o sistema de navegação autónomo para veículos agrícolas é considerado uma alternativa promissora para combater a diminuição da força de trabalho no campo, além de solucionar o problema da necessidade de aumentar a eficiência na produção agrícola, e garantir uma operação mais segura (Reid, et al., 2000).

As descrições anteriores apresentaram diferentes aspetos relacionados a navegação autónoma de veículos e robôs móveis em ambientes agrícolas. Os estudos de revisão sobre automação abordam temas como sensorização, perceção, raciocínio lógico, aprendizagem máquina, comunicação de dados, planeamento e execução de tarefas. Neste sentido, é verificada a convergência da investigação em sensores de navegação, métodos computacionais, planeamento de navegação e controladores de direção. Portanto, os sistemas de navegação autónoma para robôs agrícolas móveis consistem em sensores de navegação, métodos computacionais e estratégia de controlo (Shalal, et al., 2013) , como pode se encontra discriminado na Fig. 1.

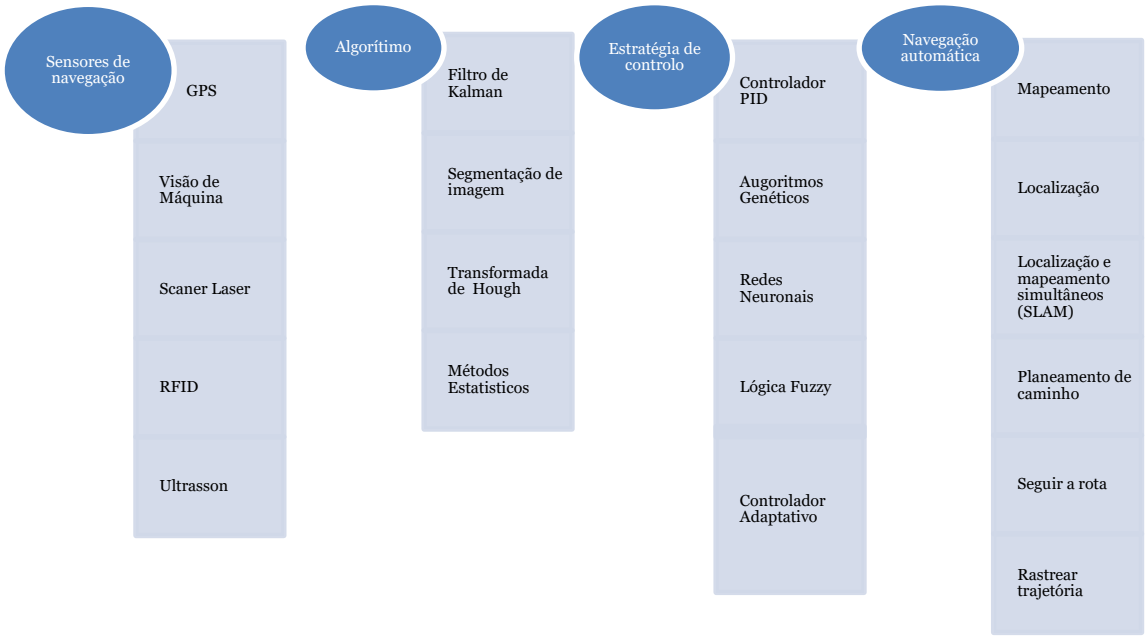

Fig. 1. Elementos básicos do sistema de navegação autónomo para robôs agrícolas móveis (Mousazadeh, 2013)-Adaptado.

2.1.1. Autonomous crop treatment Vehicle

Esse veículo consiste numa plataforma robótica para utilização na horticultura. Este robô realiza a sensorização de camadas mais superficiais do solo com finalidade de obter informações sobre as suas características. A plataforma utiliza as plantas para corrigir a posição da plataforma através dos sensores de odometria, sensor laser, sonar e visão máquina. O veículo autónomo fornece informações de orientação, além de acompanhar os padrões das fileiras e permite distinguir entre os tipos de plantas. Consiste numa boa ferramenta para o combate às ervas daninhas, como apresentado na Fig. 2 (Hague, et al., 2000).

Fig. 2. O veículo autónomo experimental Silsoe (Hague, et al., 2000).

2.1.2. Autonomous Christmas tree weede

Este veículo propõe-se realizar o controlo mecânico de ervas daninha em plantações de árvore destinadas a ornamentação de Natal. Inicialmente, o robô foi projetado para que fosse capaz de roçar um círculo de 80 cm de diâmetro em volta do tronco das árvores.

O robô foi criado a partir da adaptação de um cortador de relva comercial, tendo sido adicionados atuadores lineares, controlados por computador, para comandar os pedais de embraiagem, travão, aceleração e também a direção do veículo. Foi incluído também, um braço, na lateral do veículo, capaz de realizar a retirada das ervas daninhas. Ademais, foram adicionados os sensores de scanner a laser, inclinómetro, sensores de toque e ultrassónico que auxiliam na navegação da plataforma robótica, além do sensor principal de localização, que realiza um posicionamento cinemático em tempo-real por sistema de posicionamento global (GPS RTK - *Global Positioning System Real Time Kinematic*). Todos esses sensores foram ligados a um processador industrial embarcado e toda comunicação era realizada via rede Ethernet. O veículo encontra-se exposto na Fig. 3.

A posição de cada árvore era inserida e a área de trabalho eram entradas do controlador. Já o *software*, gera informações que serão posteriormente utilizadas para a definição da rota do veículo. Adicionalmente, a posição do braço é definida em função da localização da árvore enquanto o veículo estiver a movimentar-se, visando realizar um círculo de diâmetro predefinido em torno delas (Have, et al., 2005).

Fig. 3. Autonomous christman tree weed (Have, et al., 2005).

2.1.3. Autonomous platform and Information System

O robô autónomo é constituído por uma estrutura composta por um chassi e quatro conjuntos formados por uma roda e um sistema eletromecânico de controlo de movimentação. As

dimensões do chassi são 1,2 m x 1 m, e foi concebido para ser utilizado em terrenos limpos e culturas agrícolas de baixo porte. Nos conjuntos de propulsão, é incluso um motor elétrico e a eletrónica de controlo de cada módulo. Para a realização de manobras, foi instalado um mecanismo de dois graus de liberdades movido por um motor fixado no topo da estrutura. A eletrónica desses componentes permite o controlo de binário e velocidade, enquanto o ângulo de direção é controlado por servo-motores e um encoder que está instalado no veio. A Fig. 4 mostra a plataforma robótica.

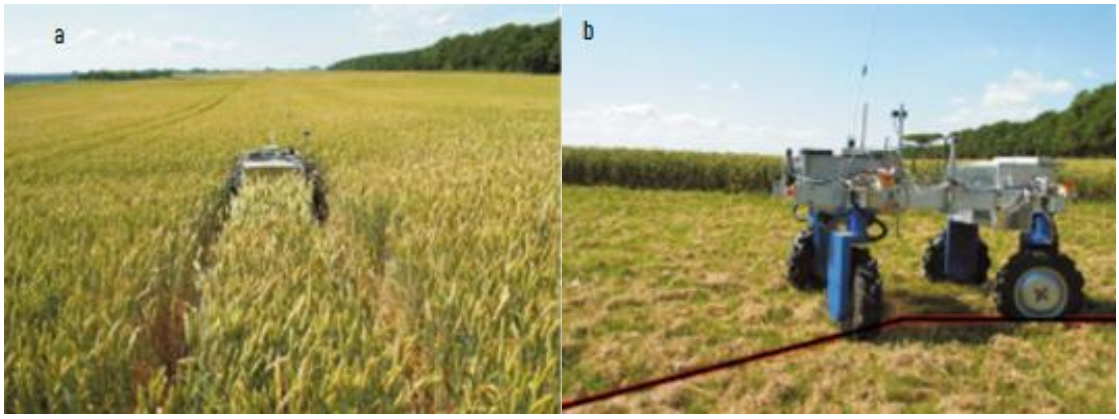

Fig. 4. Plataforma robótica: (a) Movimentação do veículo ao longo da plantação; (b) O robô segue um caminho predefinido (Bak & Jakobsen, 2004).

O robô possui um *firmware* embarcado responsável pelo controlo em tempo real do sistema. Utiliza como base o sistema operativo Linux e foi programado com auxílio da ferramenta computacional MathWorks real-time workshop. Nesta é possível programar algoritmos de controlo em linguagem C personalizáveis, tendo como base modelos de simulação. Esta solução permite uma maior produtividade, pois as linhas de código são criadas de forma automática e posteriormente são executadas na plataforma. A comunicação wireless utiliza protocolo TCP/IP. Os sensores GPS, giroscópio e bussola magnética são conectados ao computador de bordo com auxílio de cabos RS232. O sistema eletrónico do conjunto de movimentação é ligado ao computador com auxílio da interface de comunicação CAN 2.0 (Bak & Jakobsen, 2004).

2.1.4. Autonomous Rice Transplanting

Este robô foi projetado visando transformar um transplantador de arroz do fabricante japonesa ISEKI & CO para operar de forma autónoma. Atuadores foram aplicados para mover a direção, o acelerador, a embraiagem e o travão, enquanto o ângulo de viragem das rodas era obtido por um

motor DC de 80 W e o posicionamento era verificado através de um encoder absoluto. O veículo autónomo adaptado pode ser observado na Fig. 5.

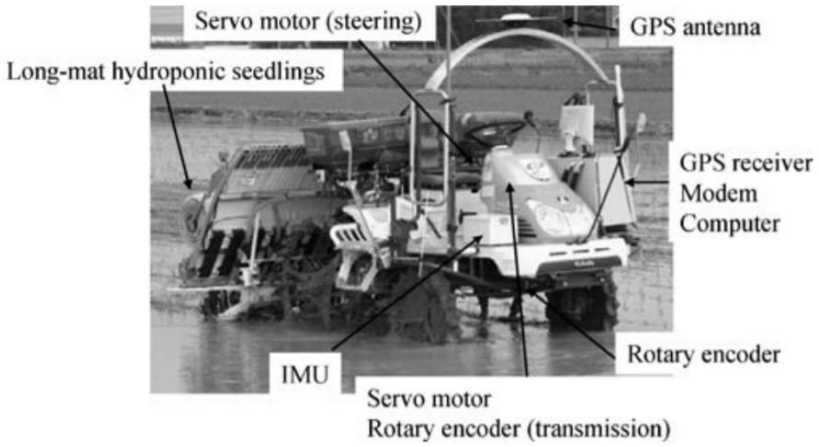

Fig. 5. Transplantador de arroz modificado (Nagasaka, et al., 2009).

O sistema de navegação utiliza um GPS RTK em conjunto com um giroscópio de fibra ótica que estão conectados ao sistema de controlo. Esse sistema possui um processador ULV Celeron 400-MHz a operar num sistema operativo PC-DOS 2000 e possui interface CAN. Foi programado em linguagem C um algoritmo de controlo, em que o computador recebe os dados de posicionamento do veículo, calcula os parâmetros de controlo e os envia para um microcontrolador que irá comandar os atuadores (Nagasaka, et al., 2009).

2.2. Sensores de navegação de robôs móveis

Nos últimos anos houve um rápido aumento no uso de sensores nos veículos empregues na agricultura. Os sensores de navegação fornecem informações precisas sobre o atual estado do robô e sobre o ambiente no qual ele se encontra a desempenhar as suas atividades. Alguns sensores de orientação fornecem informações sobre o posicionamento absoluto enquanto outros fornecem o posicionamento relativo (Lerke & Schwieger, 2021).

O ambiente agrícola oferece circunstâncias bastantes diferentes na qual um robô projetado para operar em laboratório não irá encontrar. Com isso, os sensores podem ser prejudicados por conta do contacto com humidade, poeira, nevoeiro ou radiação solar. Porém, o uso de sensores é fundamental para essa aplicação, pois, eles detetam as inúmeras mudanças no ambiente para que

os atuadores do robô transformem os sinais recebidos em respostas instantâneas que o fazem executar as tarefas.

2.2.1. Navegação baseada em GPS (*Global Positioning System)*

Desde o início da década de 90, o GPS tem sido amplamente utilizado como sensor de orientação global. Por se tornar uma tecnologia disponível comercialmente para aplicação na agricultura, os sistemas de orientação baseados em GPS puderam ser utilizados para várias aplicações no campo, como exemplo, cultivo, sementeira, plantação e lavra (Li, et al., 2009).

Atualmente, muitos fabricantes de tratores oferecem o Real-Time Kinematic (RTK) que é um sistema de direção automática baseado em GPS. Neste, a informação da posição, fornecido pelo RTK GPS, pode ser usada por ambos os sistemas de orientação e outras aplicações como mapeamento de plantação, controlo de tráfego e controlo da lavoura. Como o GPS não depende das características da cultura, eles não são afetados pela densidade de ervas daninhas, sobras, plantas ausentes ou outras condições que podem prejudicar o desempenho da máquina que usam outros tipos de sistema de orientação. Outra vantagem do GPS é que ele pode ser facilmente programado para seguir em linha reta quanto em curvas (Slaughter, et al., 2008).

Entretanto, os sistemas de orientação baseados em GPS apresentam três problemas característicos. O primeiro tipo de problema é devido aos terrenos, pois, em áreas protegidas contra micro-ondas, o sistema não funciona. Além disso, em terrenos ingrimes, ou com incidências de árvores, o sistema deixa de apresentar precisão centimétrica. A segunda limitação mais comum é devido ao tempo de processamento de dados necessário para determinar os locais que podem prejudicar a utilização do robô em atividades que exigem velocidades mais altas, visto que, a demora do processamento irá interferir na correta movimentação do veículo. E o terceiro é o custo de implementação para os sistemas no campo para os sistemas mais precisos (Li, et al., 2009).

Contudo, com o desenvolvimento de novas tecnologias, esses problemas têm sido cada vez mais incomuns e essas limitações tecnológicas estão sendo superadas. E o aumento da implementação vem diminuindo os preços de aquisição desse tipo de sistema. Com isso, os sistemas de orientação baseado em GPS vem se tornando cada vez mais uma boa escolha para ser utilizado em sistemas de orientação para veículos agrícolas autónomos.

2.2.2. Navegação baseada em visão computacional

A navegação baseada em visão computacional é amplamente utilizada nos robôs móveis autónomos por conta do custo benefício dos seus sensores e também, por causa da capacidade para gerar informações que podem ser utilizadas pelos sistemas de controlo para gerar sinais de

direção para os robôs agrícolas. Os sistemas baseados em visão computacional estão a ser popularizados para aplicações de campo como localização, mapeamento, navegação autónoma de acompanhamento de rotas, monitorização e prevenção de obstáculos.

Ao longo do tempo, a investigação na temática tem explorado diversos tipos de sensores de visão máquina para auxiliar o veículo realizar uma rota entre as fileiras das plantações. Por exemplo, detetar a posição e a orientação das linhas de cultivo e detetar as bordas ao longo das plantações, como pode ser visto na Fig. 6 (Kanagasingham, et al., 2019). Esse tipo de sistema concentra-se principalmente no desenvolvimento de diferentes técnicas de segmentação de imagens para extrair as informações de orientação para aplicações em linhas de cultivo.

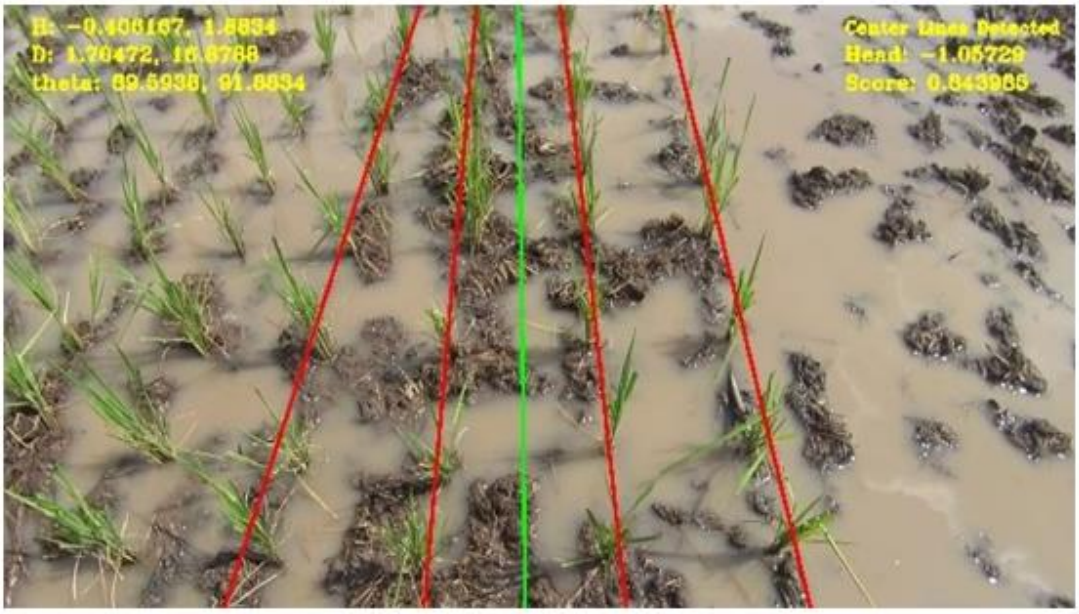

Fig. 6. Resultado da navegação baseada em visão computacional mostrando as linhas de cultivo (Kanagasingham, et al., 2019).

A fim de alcançar operações completamente autónomas, é necessário resolver o problema de guiar o robô com segurança e de forma estável entre as fileiras de cultivo, sem danificar a plantação. As lavouras, de uma maneira geral, como a grande maioria dos ambientes agrícolas, representam um grande desafio para a navegação de robôs. O solo, que poderá encontrar-se repleto de lama e ser irregular, o crescimento variável das plantas, a mudança nas cores da planta durante diferentes estágios do crescimento, e as condições climáticas adversas (como chuva e radiação solar) podem vir a prejudicar a captação de informações pelos sensores. Entretanto, algumas técnicas de plantação de arroz em linha reta e com intervalos iguais entre plantas podem amenizar esses problemas para a utilização de sensores de visão computacional (Kanagasingham, et al., 2019).

O algoritmo mostrado na Fig. 7 resume o algoritmo usado para determinar as linhas de cultivo. Um dos principais problemas do método de navegação baseada em visão computacional consiste na identificação das ervas daninhas, para que não se confunda com as culturas cultivadas. Numerosos métodos foram testados para extrair a vegetação intrusa do fundo da imagem. Um método que se baseia em árvores de decisão HSV (tonalidade, saturação e valor) destinado a extrair plantas verdes da imagem de culturas com outras cores características, mostrou-se com desempenho superior em relação a outros métodos populares, como sejam o método de Otsu, índice verde em excesso (ExG), e método de extração do índice de cor da vegetação (CIVE) (Yang, et al., 2015) .

Fig. 7. Resultado da navegação baseada em visão computacional mostrando as linhas de cultivo (Kanagasingham, et al., 2019)-Adaptado.

Benson et al. (2003) propuseram um sistema de orientação baseado em visão máquina para colheita de grãos que utiliza uma única câmara monocromática. O Algoritmo de orientação foi baseado na posição lateral da borda de corte da cultura e foi capaz de localizar linhas de cultura com precisão. Podem ser observadas na Fig. 8 as diferenças visuais limitadas entre as partes cortadas, no lado direito e a parte não cortada do lado esquerdo.

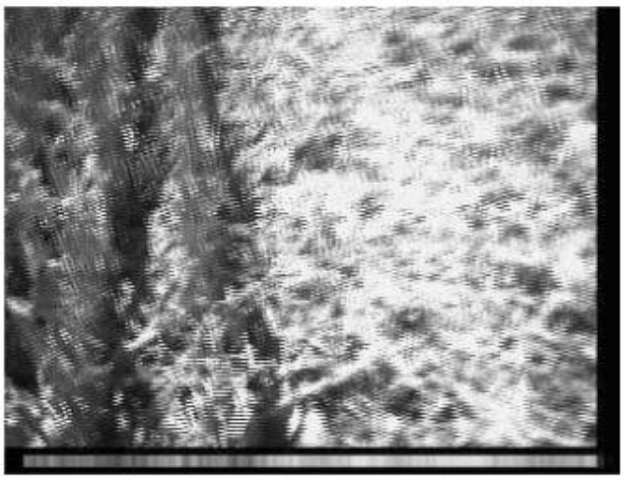

Fig. 8. Imagem monocromática típica de orientação baseado na posição lateral da borda de corte da cultura (Benson, et al., 2003).

2.2.3. Navegação baseado em scanner a laser

O scanner a laser é um dos dispositivos mais populares em aplicações externas, e, ultimamente, a investigação tem contribuído bastante para o desenvolvimento dos sistemas de navegação baseados em scanner a laser. Nesse tipo de sistema, a distância relativa entre os objetos ao redor do robô é calculada de forma relativa, medindo o tempo de deteção relativos aos pulsos de laser. Com isso, é possível fornecer dados fiáveis para a deteção de objetos o que permite que o veículo autónomo opere de forma mais confiável em diferentes condições climáticas e de iluminações ambiente.

Alguns estudos relataram que a utilização sensores a laser para deteção de fileira de cultivo, e assim, determinar a melhor direção para a orientação do veículo. São utilizados inúmeros métodos para a deteção de linhas e fusão de dados do sensor. A fusão de dados do sensor é o processo de combinar observações de vários sensores para fornecer uma descrição robusta e completa do ambiente. Assim sendo, o sensor a laser é usado como sensor de navegação e geralmente combinado com outros sensores utilizando o algoritmo de fusão de dados adequados (Durrant-Whyte & Henderson, 2016).

2.2.4. Navegação baseada em scanner a laser e visão

Um sistema automático de orientação de linhas de cultivo para tratores, foi investigado, visando utilizar sensor a laser capaz de detetar a altura e as posições das linhas de cultura. Isso tornou-se possível pois foi montado no trator, um computador e um espelho giratório, que deteta a posição da linha da cultura ou as linhas marcadas na superfície do solo por uma sementeira. Com isso, um alto grau de linearidade foi obtido entre as posições do objeto e os valores reais mensurados pelo sensor (Satow, et al., 2004).

O sensor laser 3D LIDAR baseado em MEMS (*Micro-ElectroMechanical Machines*) pode ser utilizado para deteção e segmentação de plantas para realizar localização, mapeamento e navegação em ambientes agrícolas. Com isso, as plantas podem ser detetadas de forma individual. A vantagem dessa técnica é que, primeiramente, o robô segue uma linha colheita sem prejudicar nenhuma planta. Segundo, as plantas podem ser usadas como marcos naturais e com isso podem ser mais bem mapeadas. Na Fig. 9 é possível observar o sensor estudado em uma linha de cultivo de milho e a deteção das plantas (Weiss & Peter Biber, 2011).

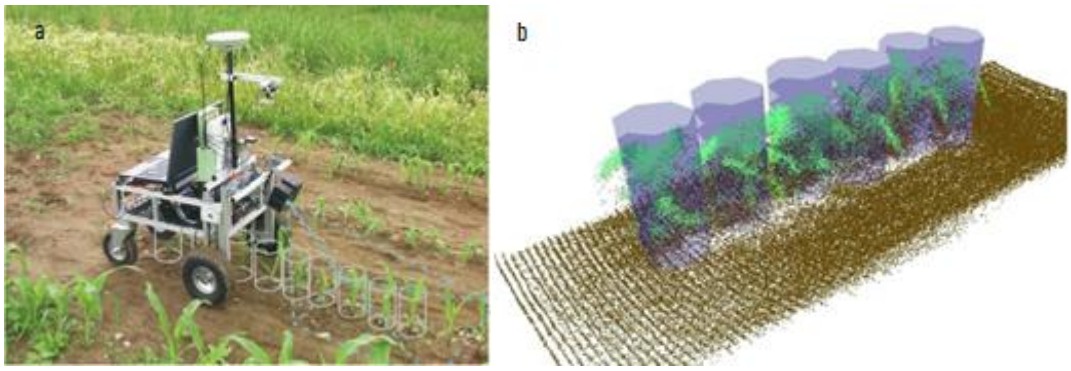

Fig. 9. (a) Sensor laser 3D LIDAR baseado em MEMS. (b) Imagem formada com auxílio de sensor a laser (Weiss & Peter Biber, 2011).

2.3. Métodos computacionais

Um robô autónomo móvel requer um posicionamento preciso para se deslocar em diversos tipos de ambiente. A determinação exata da posição e orientação do robô é consequência das leituras dos sensores. Entretanto, cada sensor é responsável por determinar, com precisão limitada, um ou dois parâmetros ambientais (Hoang, et al., 2012).

Cada uma das principais tecnologias utilizadas para o controlo de robôs autónomos aplicados a agricultura exige esforços significativos de pesquisa e desenvolvimento, uma vez que vibração, poeira e outros problemas associados a implementação de um sistema de controlo robótico, que pode conter diversas soluções sensores para a realização de diversos tipos de atividades. Entretanto, essas soluções necessitam de estar corretamente integradas para que funcionem como um sistema completo, transmitindo informações bem-sucedidas entre subsistemas utilizados (Slaughter, et al., 2008).

Com essa finalidade, algoritmos e métodos robustos são absolutamente necessários para extrair os recursos necessários do ambiente para conduzir, de forma autónoma, os veículos. Ainda mais, diferentes técnicas computacionais são usadas para lidar com a integração dos dados dos diversos sensores visando o fornecimento de informações para a navegação autónoma de veículos agrícolas. Logo, a escolha e a melhoria dos métodos são muito importantes, pois, esses métodos computacionais possuem a finalidade principal de detetar e processar dados que serão utilizados pelo sistema de orientação autónomo.

2.3.1. Filtro de Kalman

O filtro de Kalman fornece um método matemático capaz de realizar a fusão de dados de vários sensores em tempo real. Com isso, o método é capaz de reunir inúmeras medições, ao longo do

tempo. As combinações dessas medidas são utilizadas para estimar o estado do sistema em cada instante. Este método pode estimar tanto o estado anterior e o presente quanto o estado futuro (Shalal, et al., 2013).

A versão padrão do filtro de Kalman é projetada para aplicações onde o processo é descrito por equações diferenciais estocásticas lineares. Entretanto, na maioria dos casos os sistemas robóticos têm características não lineares, por isso algumas melhorias nesse sistema foram desenvolvidas e o EKF (*Extended Kalman Filter* - Filtro de Kalman Estendido) foi utilizado para resolver o problema de sistemas não lineares (Shalal, et al., 2013).

Várias aplicações foram desenvolvidas com a aplicação de Filtros de Kalman, as quais incluem-se, navegação, localização e rastreamento de objetos. Com isso, o filtro de Kalman é um algoritmo que estima o estado de um processo controlado por tempo na forma discreta. Esse método é famoso na teoria de sistemas dinâmico-estocásticos, e são úteis para melhorar as estimativas, sendo uma das técnicas mais úteis para implementação em sistemas lineares (ALATISE & HANCKE, 2020).

As Equações (1) e (2) servem para implementação de Filtro de Kalman (Drongelen, 2018).

$$\hat{x}_k = F_k \hat{x}_{k-1} + B_k u_k \tag{1}$$

$$\hat{P}_k = F_k \hat{P}_{k-1} F_k^T + Q_k \tag{2}$$

Sendo o vetor $\hat{x}_k$ o estado estimado do sistema x_k. P_k é a matriz de covariância prevista. F é a matriz que denota a dinâmica do sistema. B é a matriz de controlo e Q é a covariância de ruido. A equação do filtro de Kalman é utilizada para criar estimativas através da adição de uma unidade externa para correção. Essa técnica envolve outra etapa para atualizar as estimativas, que é descrita pelas Equações (3) a (5).

$$\hat{x}'_k = \hat{x}_k + K'(z_k - H_k \hat{x}_k) \tag{3}$$

$$P'_k = P_k - k' H_k P_k \tag{4}$$

$$K' = P_k H_k^T \hat{x}_k (H_k P_k H_k^T + R_k)^{-1} \tag{5}$$

Das equações acima, z_k é vetor de medição, ou seja, são as leituras dos sensores. H é a matriz de transformação, R é a matriz de covariância do ruído de medição e k é o intervalo de tempo. Com isso é possível descrever a quantidade de atualizações necessárias em cada estimativa recursiva.

Uma plataforma de robô móvel com alguns sensores foi projetada utilizando um Filtro de Kalman estendido. Como informação de entrada é utilizado um sistema odométrico, um sensor de bússola, um sensor a laser e uma câmara unidirecional. E como saída é mostrada a posição e a orientação do robô. De tal forma que a estimativa dos valores de saída são os mais próximos do real quando as informações de todos os sensores são fundidas usando o filtro de Kalman estendido (Hoang, et al., 2012).

2.3.2. Transformada de Hough

A transformada de Hough é uma técnica de extração de recursos usada em análise de imagem, visão computacional e processamento digital de imagem. E tem como objetivo o isolamento de recursos de forma específica numa imagem. A transformada de Hough clássica propõe a identificação de linhas na imagem gerada, entretanto, posteriormente foi aperfeiçoada para identificar curvas como círculos e elipses, como pode ser observado na Fig. 10 (Hamner, et al., 2010).

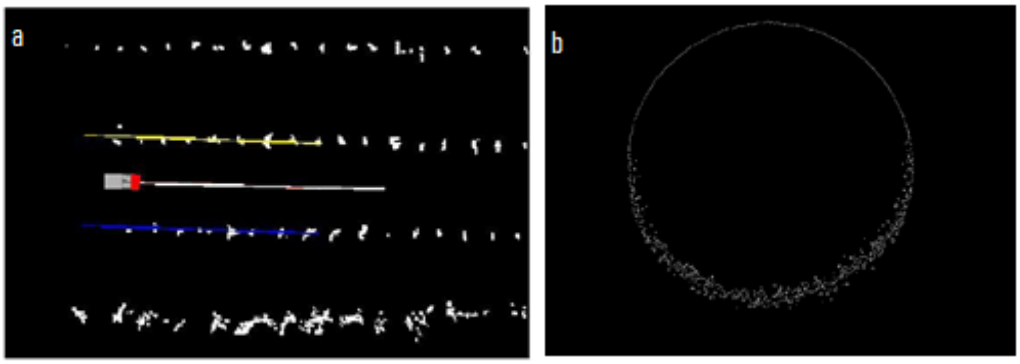

Fig. 10. (a) Transformada de Hough para deteção de linhas de árvores (Hamner, et al., 2010) (b) Visão de linha circular detetada através da transformada de transformada de Hough (Anon., 2021).

A transformada de Hough foi utilizada em diversos estudos para o reconhecimento de linhas de cultivo ou árvores, com auxílio de sensores de visão e scanner a laser. Através do algoritmo de Hough, a qualidade da estimativa da posição da linha pode ser avaliada pela comparação com as

linhas encontradas pelo algoritmo com as linhas posicionadas sobre as linhas de cultivo, através da inspeção visual.

Outra aplicação da transformada de Hough é a construção de caminhos após a deteção das filas de árvores, ou seja, o caminho é localizado entre duas filas de árvores. Com isso, esse método mostra-se bastante robusto para reconhecimento dos ambientes, o que permite guiar máquinas agrícolas. Entretanto, as linhas podem estar incompletas e as plantas podem possuir diferentes tamanhos ao longo da plantação. Outro problema são as ervas daninhas que podem perturbar a aparência de uma estrutura de linha de vegetação, o que pode prejudicar a deteção. Esses fatores tornam a tarefa de deteção de linha muito mais difícil (Åstrand & Baerveldt, 2005).

2.3.3. Segmentação de imagem

Na visão computacional, a segmentação de imagem refere-se ao processo de particionar uma imagem digital em várias regiões ou conjunto de pixéis. Em geral, a aplicação dessa técnica não resulta numa imagem, mas num conjunto de região ou objetos, na qual, a precisão, na fase de segmentação, costuma determinar o sucesso ou falha do procedimento de análise de imagem por computador (Li & Qiu, 2021).

A saber, a segmentação de imagem pode se dividir em duas estratégias. A primeira, é denominada descontinuidade, em que a partição da imagem é efetuada baseada nas alterações abruptas de intensidade, por exemplo, em determinações de contornos. Ou por similaridade, de tal forma que a partição da imagem é realizada com base na similaridade entre pixéis, de acordo com determinados critérios, como exemplo, binarização, crescimento, divisão e junção de regiões.

Em ambientes agrícolas, as técnicas de segmentação de imagem são usadas para separar objetos em diferentes classes (plantação, árvores, terreno, ervas daninhas e etc) para extrair informações e, com isso, os robôs podem definir as suas rotas. Neste caso, a técnica é normalmente utilizada para que os robôs autónomos móveis consigam localizar os limites (linhas ou curvas) em imagem e, por intermédio delas, calcular sua melhor rota (Shalal, et al., 2013).

Vários métodos de segmentação de imagens têm sido propostos na literatura. Um deles investigou a aplicação de técnicas de processamento de imagens em visão computacional para dispositivos móveis agrícolas projetado para solucionar problema de trajetória de navegação. Para isso o robô contava com auxílio de algoritmos de segmentação de imagem somado a um algoritmo de perceção multicamada. Além disso, foi utilizado o algoritmo de retropropagação personalizado juntamente com métodos estatísticos como seja o método Heurístico, para que com isso, fosse possível a deteção de cores (Lulio, et al., 2012). Um exemplo pode ser observado na Fig. 11.

Fig. 11. Resultado do processo de segmentação de imagem (Lulio, et al., 2012).

2.4. Estratégias de controlo de Navegação

O projeto do controlador utilizado em veículos autónomos agrícolas é um desafio demasiado difícil, visto que os robôs móveis agrícolas operam em vários tipos de terreno que podem sofrer modificações e tornando-se imprevisíveis. Por isso, ao projetar o sistema de controlo, o tipo de terreno para o robô móvel deve ser identificado. Geralmente são classificados como: irregulares, nivelados, escalados ascendente, escalados descendentes (Saudabayev, et al., 2015).

Para navegação autónoma, os controladores devem ser capazes de fornecer comandos de direção necessários em respostas as inúmeras variações no estado do robô, velocidade, condições do solo e outros fatores que afetam o movimento. Diferentes estratégias de controlo como Controlador-Proporcional-Integral-Derivativo (PID), Rede Neuronais Artificiais, Algoritmos Genéticos e Lógica Fuzzy, foram identificadas na literatura (Singh, et al., 2008). Um modelo esquemático de um controlador inteligente utilizado em robôs autónomos é mostrado na Fig. 12.

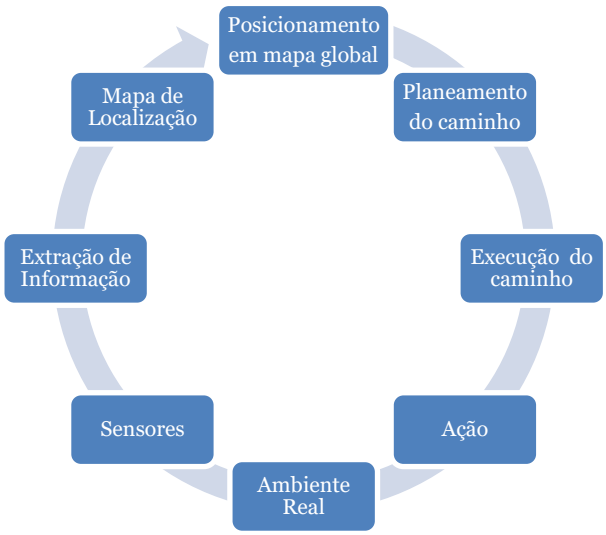

Fig. 12. Esquema de um controlador inteligente (Singh, et al., 2008) - Adaptado

Entretanto, a questão de como será possível ajustar um controlador para ser utilizado em qualquer aplicação de forma rápida e eficiente mantém-se relevante. A resposta é demasiada complexa e ocupa constantemente a comunidade científica dedicada à investigação em automação e robótica (Singh, et al., 2008).

2.4.1. Controlador PID (Proporcional-Integral-Derivativo)

Os primeiros controladores a serem de desenvolvidos foram os controladores PID. Devido ao rápido desenvolvimento da teoria de controlo, foi suposto que os controladores PID seriam progressivamente substituídos por controladores mais avançados. Todavia, isso não ocorreu, principalmente, devido a sua simples estrutura e fácil aplicação prática. Por muitos anos, foram preferidos os controladores PID pela sua grande capacidade de eliminar o erro de controlo, por conta do uso do integrador, pela sua capacidade de melhorar o desempenho usando a tendência da variável controlada por meios de canais derivativos e por muitos outros benefícios (Shamsuzzoha, 2018).

A aplicabilidade comercial dos controladores PID é confirmada por pesquisas que afirmam que 90% de todos os controladores instalados em *loops* de controlo industrial são controladores PIDs. Entretanto, 20% deles estão sintonizados corretamente, e 30% de todas as aplicações de PID, o método de síntese é escolhido de forma errada. Outros problemas são causados por erro nos atuadores não linear (30%) e, os 20% restantes representam uma opção inadequada no período de amostragem ou filtragem de sinal (Shamsuzzoha, 2018).

O controlador PID tem sido utilizado em diversos estudos para sistemas de orientação de robôs móveis em ambientes agrícolas. Uma abordagem para o problema de navegação autónoma em ambientes agrícolas usou controlador Proporcional-Integral (PI) para o controlo da velocidade e direção do robô. Nesse estudo, o controlador PI foi suficiente para controlo da velocidade, pois era hábil para compensar as perturbações proveniente do ângulo de direção, ao menos em estado estacionário. O esquema do controlador pode ser observado na Fig. 13.

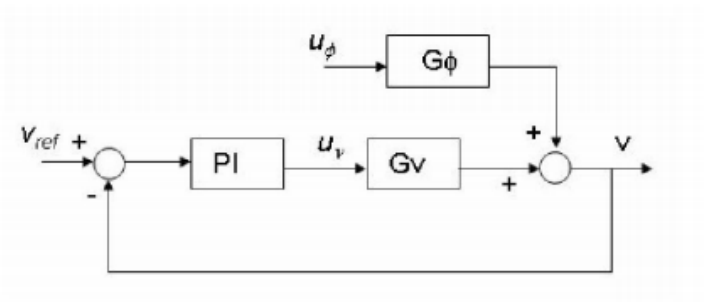

Fig. 13. Diagrama de bloco do sistema linearizado, considerando a tensão de armadura do motor de propulsão DC como entrada e velocidade linear do robô como saída (Ortiz & Olivares, 2006).

2.4.2. Redes Neuronais Artificiais

As Redes Neuronais Artificiais são sistemas formados por unidades, nomeados nós, de processamento simples que são capazes de aplicar funções matemáticas a dados que foram recebidos na entrada, simulando um neurónio biológico. Neste caso, o neurónio é formulado a partir de um vetor x de entradas $x_1, x_2 \dots x_n$ e de uma saída y. Acopladas as entradas dos neurónios, estão os pesos que são constituídos por um vetor w com valores $w_1, w_2 \dots w_n$. Como resultado, a saída y é ativada após aplicação de uma função de ativação *f*, que, em geral, recebe combinações lineares do sinal de entrada x_i (Braga & Ludemir, 2007).

$$In = \sum_{i=1}^{n} w_i x_i \tag{6}$$

E, a saída é descrita como:

$$y = f(in) \tag{7}$$

Podendo $f(x)$ ser representada por uma função degrau ou uma função sigmóide, como por exemplo uma função logarítmica sigmóide:

$$f(x) = \frac{1}{1+e^{-\alpha x}} \tag{8}$$

Com o desenvolvimento da aprendizagem máquina, as Redes Neuronais Artificiais vêm sendo cada vez mais utilizadas. Essa rede Neuronal é treinada para fornecer à máquina o comportamento desejado. Entretanto, o treino consome demasiados recursos computacionais, visto que, numerosos conjuntos de dados são requeridos para treinar a rede. A programação de uma rede neuronal é feita baseada no funcionamento do cérebro humano. Em que cada neurónio recebe uma entrada, processa e comunica o sinal para outros neurónios. Logo, as redes neuronais artificiais consistem em uma ou mais camadas com neurónios de entrada, internos e de saída, como pode ser observado na Fig. 14.

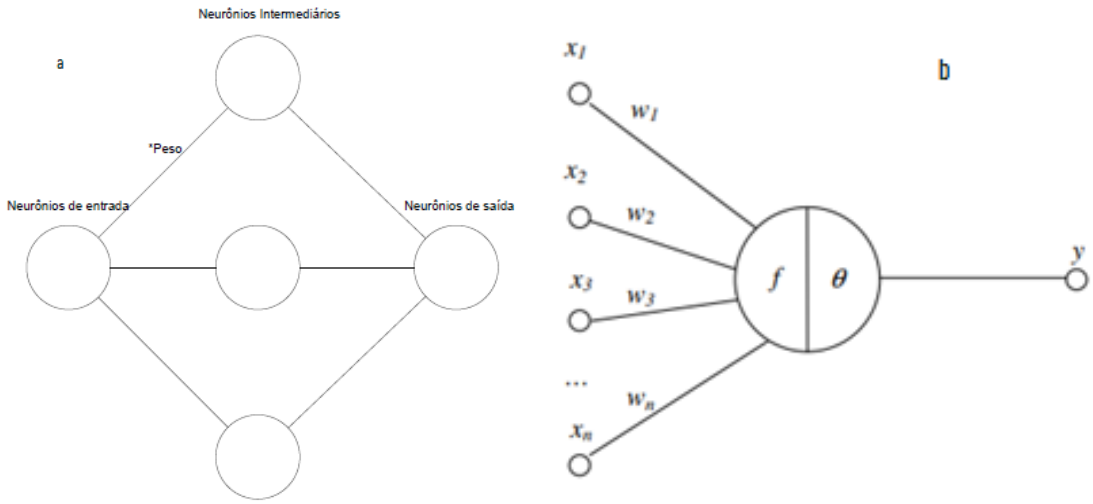

Fig. 14. (a) Esquema de uma Rede Neuronal Artificial e (b) de um neurónio artificial (Silva, 2010).

São encontrados vários tipos de redes neuronais na literatura (Kim, et al., 2021), como também, possuem diferentes classes e são utilizadas em inúmeras aplicações, como exemplo, reconhecimento de voz ou classificação de imagem. As Redes Neuronais Artificiais que passam a saída de uma camada para outra sem formar um ciclo são denominadas Feedforward. Já as que contêm ciclos direcionados onde há feedback são nomeadas Recorrentes. Entretanto, se as redes Neuronais possuírem várias camadas ocultas, permitindo que uma tarefa de aprendizagem máquina mais complexa seja implementada, são chamadas de Deep Neural Networks.

Investigações utilizaram algoritmos de Redes Neuronais Artificiais para controlar o movimento de robôs autónomos móveis aplicados na agricultura (Zhu, et al., 2005). Um projeto de veículo que utiliza redes neuronais foi proposto visando obter relações entre a direção e o comportamento de tratores em terrenos inclinados. Sempre que um veículo agrícola com rodas se move em terreno inclinado, distúrbios externos, como a atração gravitacional, que puxa o robô para baixo, causa forte não linearidade na dinâmica do veículo. Logo, um modelo de veículo baseado em Redes Neuronais Artificiais foi bem-sucedido para relacionar as entradas e saídas para o controlo de um veículo em terrenos inclinados. Sendo que, a interconexão ajustada dos pesos nos Neurônios do modelo inclui todos os elementos efetivos como deslizamento devido a gravidade, condições irregulares, declives e outras influencias externas (Zhu, et al., 2005).

Um outro modelo de veículo autónomo, utilizado em aplicações rurais, baseado em Redes Neuronais Artificiais foi proposto com finalidade de estimar o comportamento de veículos em terrenos inclinados. Entretanto, foi utilizado um método que combinou com Algoritmos Genéticos e Algoritmos de retro propagação a fim de realizar o treino da Rede Neuronal. Com isso, um controlador de direção baseado em redes Neuronais Artificiais foi desenvolvido para

encontrar a direção ideal para diferentes inclinações de terreno. Foram realizados testes de movimentação autónoma ao longo de caminhos em forma de retângulo e verificou-se que o trator desempenhou o circuito com precisão (ASHRAF, et al., 2010).

2.4.3. Algoritmos genéticos

Os Algoritmos genéticos são um subconjunto da Inteligência Artificial e podem ser usados para solucionar problemas de automação em diversos sistemas (RAISCH, 2021). A ideia principal desta técnica utiliza uma estratégia de solução de problemas baseado no mecanismo de evolução biológica. Com isso, ao analisar um problema específico, a entrada de dados é um conjunto de soluções (podem ser chamadas de função de fitness) que serão posteriormente avaliadas e otimizadas. Assim, o algoritmo genético cria uma população de soluções e aplica operadores genéticos, como mutação e cruzamentos a fim de evoluir as soluções e encontrar as melhores. Logo um algoritmo genético é uma estrutura para solução de problemas, não uma solução em si, portanto, é necessário adaptá-lo para diferentes problemas. Na Fig. 15, pode ser visto um fluxograma de funcionamento de um Algoritmo Genético (Mousazadeh, 2013).

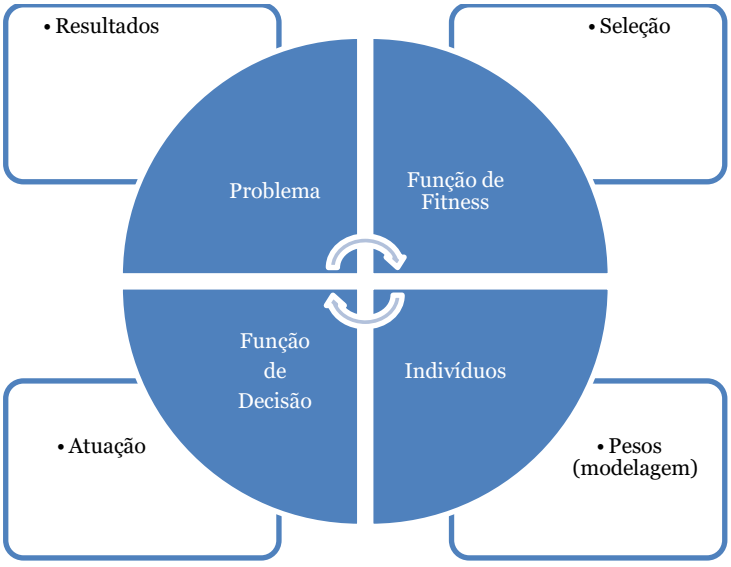

Fig. 15. Fluxograma do funcionamento de um Algoritmo Genético.

Algoritmos genéticos foram aplicados a navegação autónomas de veículos agrícolas para encontrar a melhor rota ao guiar um trator para cobrir completamente um campo determinado e evitar todos obstáculos. Nesse caso, o problema de planeamento de caminho baseado em algoritmos genéticos foi personalizado por meio de cinco componentes principais para

representar o campo, os obstáculos, a grande área percorrida, o veículo automatizado e o caminho. Os resultados da simulação demostram a viabilidade da utilização do método e, entretanto, é necessário a realização de melhorias no código visando sua aplicação em equipamentos agrícolas no campo em uma situação real (Ryerson & Zhang, 2007).

Além disso, em outro estudo, foi avaliada a utilização de um algoritmo genético para o planeamento cinodinâmico (classe de problemas em que os limites de velocidade, aceleração, força, torque e restrições cinemáticas devem ser satisfeitos) em veículos agrícolas autónomos inseridos em ambientes que possuem obstáculos e fronteiras complicadas. Foram realizados ajustes e o algoritmo genético foi executado cerca de 2000 vezes, tendo a função custo otimizada e o tempo de execução como fator crítico para aplicação, pois as decisões de navegação necessitam ser tomadas rapidamente em ambiente de mudanças dinâmicas como seja o ambiente agrícola (FERENTINOS, et al., 2002).

2.4.4. Controlador Fuzzy Proporcional Integral Derivativo (PID)

O controlador Fuzzy, controlador que utiliza Lógica Fuzzy ou Nebulosa, é adequado para realizar o comando de um robô móvel autónomo, pois, é capaz de fazer interferências mesmo sob incerteza. Esse sistema é usado para produzir as entradas de controlo com entrada para vários sensores. Os sinais do sensor são alimentados ao sistema de lógica nebulosa e a saída fornece comando de controlo do motor. Ao contrário de outros métodos, em geral, é fácil a determinação da ação que, no tempo futuro, o controlador Fuzzy PID irá executar. Isso porque, esses controladores geralmente possuem estruturas analíticas e é codificada de acordo com uma regra lógica simples, da forma de "se um conjunto de condições for satisfeito, então, um conjunto de consequências são inferidas" (Singh, et al., 2008).

O controlador Fuzzy consiste em quatro partes principais. Entretanto, antes da informação ser fornecida ao método de processamento, ela necessita ser pré-processada. Nessa etapa, as informações aferidas são condicionadas antes de entrar no controlador. Pode-se tomar como exemplo a filtragem de ruído.

I. Fuzzificação - É a primeira etapa dentro do Fuzzy PID, onde os dados de entrada são convertidos em graus de pertinência através da utilização de funções de pertinência.

II. Regras de base – As regras de base são formadas por normas no formato *if-then* que são derivadas da experiência de um especialista.

III. Motor de inferência – Com finalidade de tirar conclusões sobre o mecanismo de inferência da base de regra que emprega um algoritmo que pode produzir uma saída de forma se-então.

IV. Defuzzyficação- Os conjuntos fuzzy resultantes são convertidos em um número que pode ser enviado para o processo como um sinal de controlo. Esta conversão de valores nebulosos em valores nítidos é chamada de defuzzificação.

Por conseguinte, ocorre o pós processamento. Nesse caso, a saída é definida em um universo padrão e deve ser convertida para uma unidade do sistema de medidas. Essa conversão, do valor defuzzyficado para unidades, contendo algum fator de escala, é denominada pós processamento. Essa etapa contém um ganho de saída que pode ser ajustado, e também, pode conter um integrador (Khan & Rapal, 2006).

Um sistema baseado em controlador Neuro-Fuzzy foi proposto para navegação reativa de um robô móvel utilizando controlo baseado em comportamento. Esse algoritmo utilizou amostragem discreta baseada no treino ótimo de Redes Neuronais Artificiais. Os sistemas Neuros-Fuzzy tem sido bastante aplicados em robôs por conta da sua capacidade de simular a experiência humana e utilizar o conhecimento adquirido para desenvolver as estratégias de navegação autónoma. Os resultados das simulações mostraram que a técnica utilizada se mostrou eficiente em todos os tipos de ambientes e obstáculos (Joshi & Zaveri, 2011).

Um controlo de rastreamento utilizando um controlador Fuzzy PID de auto ajuste foi proposto para um trator. Nesse caso, o controlo de direção é realizado nas rodas dianteiras e a tração é operada nas rodas traseiras. Os resultados mostraram uma boa eficiência na utilização do controlador Fuzzy PID em comparação com os demais PIDs, o que garantiu uma resposta rápida do volante e um pequeno erro de distanciamento e ângulo de direção (Upaphai, et al., 2019).

2.5. Realidade Aumentada aplicada a robótica

A Realidade Aumentada tem vindo a ser popularizada na última década e vem sendo aplicada em inúmeras pesquisas com diferentes áreas do conhecimento. Em geral, sua utilização visa melhorar o feedback visual das informações do sistema, e para isso, são criadas diversas aplicações. Alem disso, com computadores mais rápidos, câmaras de melhor qualidade e novos algoritmos, fazem com o que haja motivação para expandir as áreas de aplicação (Zhou & Li, 2021).

Não apenas a Indústria 4.0 desencadeou o uso da Realidade Aumentada em sistemas físicos conectados e na comunicação homem-máquina. Também robôs estão sendo aplicados desde a indústria até nas áreas como robótica de reabilitação, robótica social e robótica móvel. Com isso, a Realidade Aumentada auxilia a robótica como um novo meio de troca de informações com sistemas autónomos o que leva ao aumento da eficiência da interação homem-robô (Makhataeva & Varol, 2020).

Na literatura científica no campo da agricultura, alguns autores têm apresentado trabalhos utilizando a Realidade Aumentada. Um sistema de orientação, utilizando GPS e tecnologia de

Realidade Aumentada, permitiu ao operador de tratores ver o trajeto real através de óculos monitores que tratavam as imagens e as mostravam em zonas de diferentes cores. O sistema de orientação é composto por uma aplicação programada em linguagem Python e executado num sistema de trator onde o operador pode estar dentro ou fora da máquina. A aplicação adquire informações de alguns sensores, processa essas informações e mostra um vídeo nos óculos monitores, como pode ser visto na Fig. 16**Erro! A origem da referência não foi encontrada.** (Fernández, et al., 2010).

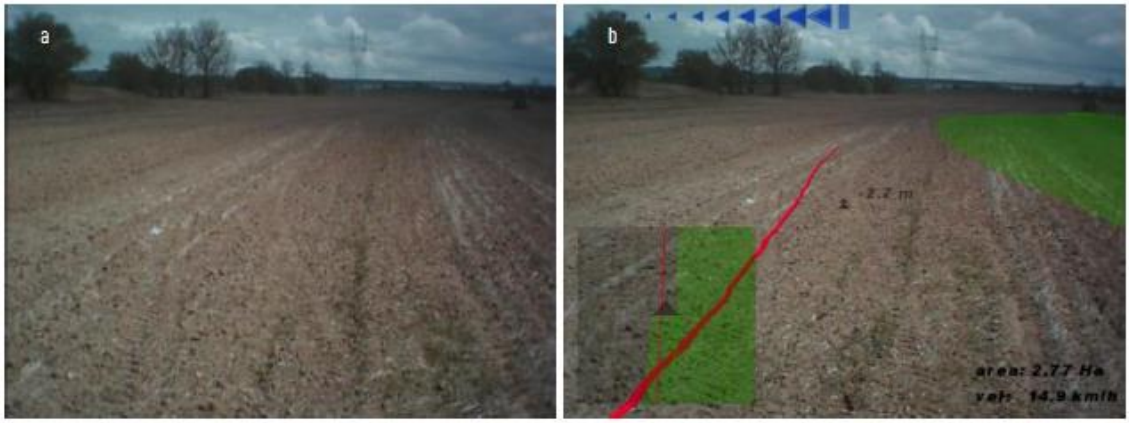

Fig. 16. Imagem mostrada para o operador (a) sem o sistema de orientação e (b) com o sistema de orientação de Realidade Aumentada (Fernández, et al., 2010).

Por conta do crescente interesse em agricultura de precisão, tecnologia da informação e navegação autónoma, foi proposto um sistema de Realidade Aumentada a fim de auxiliar na tomada de decisão para o controlo de ervas daninhas. Foi criado um algoritmo de reconhecimento de imagens para identificar e quantificar ervas daninhas por espécie e outro software para quantificar a dosagem de herbicida de acordo com a densidade das ervas daninhas (VIDAL & VIDAL, 2010).

2.6. Nota conclusiva

Atualmente, os sistemas móveis automatizados estão sendo cada vez mais utilizados em ambientes agrícolas. Ao utilizar sensores como GPS, ultrassons, visão máquina, entre outros, tornam a navegação mais confiável. Entretanto a utilização de apenas um sensor pode prejudicar a navegação. Em contraponto, a utilização de sensores redundantes pode aumentar a fiabilidade do sistema.

A ideia básica por trás da fusão de sensores é que, combinando diversos dados de sensorização é possível obter navegação por mapeamento e estimativa de posição mais precisa do robô. Todavia, o uso de muitos sensores tornará o projeto mais oneroso e complexo. Pode-se concluir que tanto o GPS quanto a visão máquina podem ser fundidos com outras tecnologias, como o laser, para o desenvolvimento de sistemas móveis agrícolas.

Conclui-se também que o uso de tecnologias de Realidade Aumentada aplicada como auxílio a navegação autónoma é considerada uma mais-valia para a supervisão das operações de campo, fornecendo ao supervisor das operações agrícolas inúmeras informações que irá auxilia na gestão de mapas, criação de rotas, alerta de perigo, entre outros.

3. Materiais e Métodos

3.1. Materiais

3.1.1. Robô terrestre Agrícola Multitarefa

Desenvolvido no projeto PrunusBot (PDR2020-101-031358), o Rover Robótico para Aplicações Agrícolas (R2A2), é um robô terrestre agrícola multitarefa que tem por finalidade realizar de forma autónoma a pulverização de herbicida de forma particularizada em pomares de pessegueiro. O sistema robótico possui, basicamente, duas funções executadas em épocas distintas. A primeira, consiste na realização de pulverização de precisão de herbicida ao iniciar o ano, pois é o momento em que as ervas daninhas ainda estão pequenas. E a segunda, ao final da safra, fará a recolha dos pêssegos caídos no chão do pomar (Veiros & Gaspar, 2020). Durante o crescimento da cultura, encontra-se munido de câmaras e de algoritmos de inteligência artificial que através do processamento das imagens irão realizar a deteção de frutos, para proceder à sua contagem e desse modo providenciar uma previsão mais assertiva da produção, e também a sua classificação, para possibilitar a identificação precoce de doenças nos frutos e desse modo permitir ao fruticultor tomar as medidas necessárias atempadamente.

Com isso, o impacte ambiental tende a ser minimizado, visto que, a quantidade de herbicida utilizada no controlo das ervas daninhas será reduzida por conta da pulverização de precisão. Como também, a recolha dos pêssegos caídos evita a proliferação de insetos e bactérias que crescem nos frutos caídos e que hibernarão até a cultura do próximo ano. A plataforma também visa reduzir os custos de mão de obra associados as atividades de remoção manual de pêssegos.

A plataforma robótica foi construída de forma que tivesse capacidade para se mover de forma autónoma no pomar e realizar tarefas junto ao tronco das árvores, entretanto, sem passar entre duas delas, deslocando-se, apenas, na entrelinha das árvores. O robô foi projetado para superar inclinações do terreno de até 20º e mover-se sob coberturas maiores que 600 mm. Na Tabela 1 apresentam-se valores referentes as demais especificações do robô.

Tabela 1. Especificações técnicas do Rover Robótico para Aplicações Agrícolas (R2A2).

Especificação do robô	Valor
Peso aproximado	90 kg
Carga útil	15 kg
Velocidade Máxima	1,4 m/s
Aceleração	1 m/s²
Comprimento	1200 mm
Largura	1050 mm
Altura	500 mm

A modelagem do protótipo foi desenvolvida com software CAD Solidworks. Com isso, foi realizado o planeamento de toda a construção e montagem do robô. A ferramenta também possibilitou a extração dos desenhos técnicos que auxiliaram na construção do sistema, inclusive todos os desenhos das peças que foram fabricadas com auxílio das máquinas CNC. Na Fig. 17 encontra-se a vista explodida do veículo realizada pelo software CAD utilizado (Veiros, 2019).

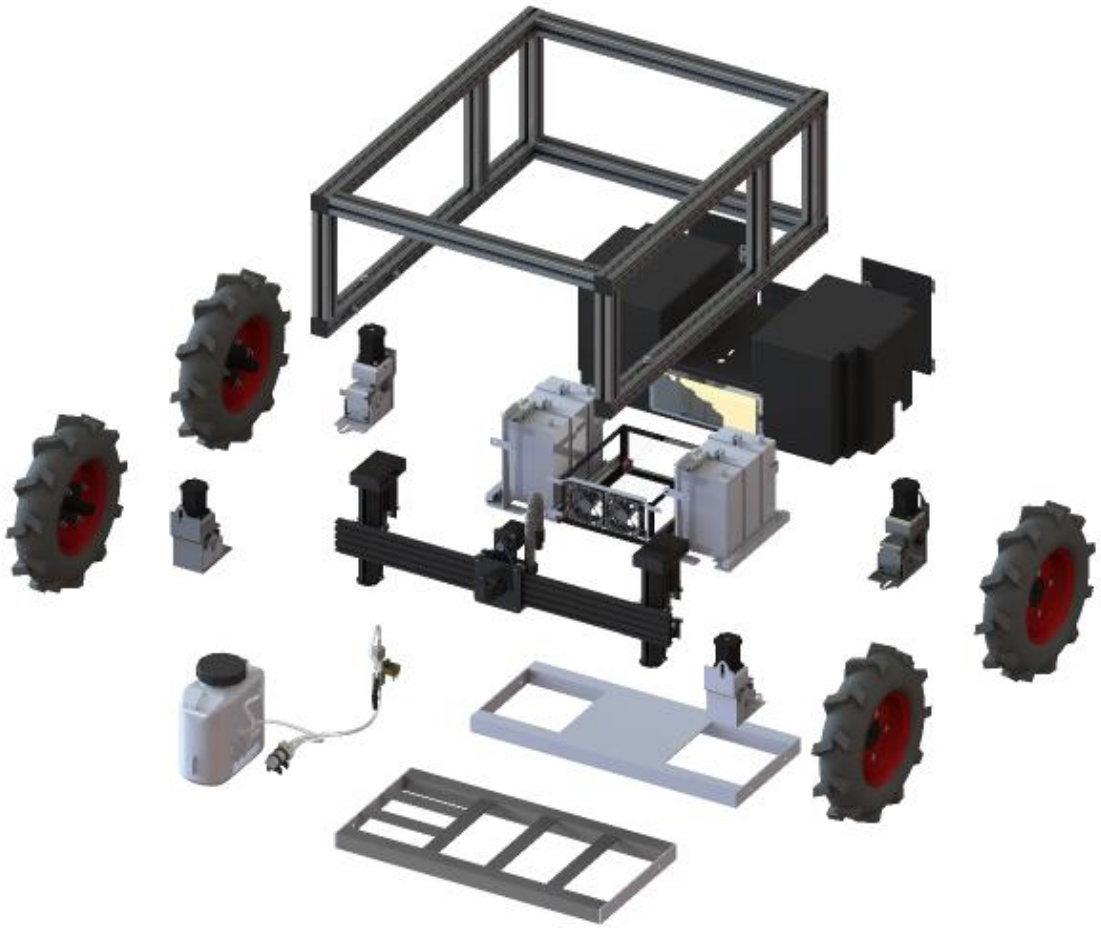

Fig. 17. Vista explodida do robô.

A componente estrutural do robô é formada utilizando perfil de alumínio com ranhuras em forma de T-slot 45x45 mm da marca Boch Rexroth, o que cria uma estrutura muito resistente e leve. Essa solução foi selecionada visto que é de fácil montagem, e também, permite a fácil fixação de diferentes peças que possam ser adicionadas após a conclusão do projeto. Além do fácil reposicionamento e ajustabilidade da estrutura.

A plataforma robótica possui um braço cartesiano destinado à realização de pulverização. Esse braço robótico possui 5 eixos e direciona o bico de pulverização e a garra que realizará a recolha dos pêssegos que estiverem caídos ao chão. Nesse caso, será possível realizar tarefas em extensão de 1200 mm, pois, os sistemas acionados podem trabalhar fora da estrutura do robô para realização de atividades próximo ao caule das culturas.

Já na Fig. 18, é mostrado a perspetiva isométrica do robô, ou seja, o desenho 3D final da plataforma robótica, já montado e com todas a componentes que lhe auxiliarão a desempenhar suas tarefas. O modelo 3D foi uma mais-valia pois possibilitou a otimização dos recursos construtivos, melhoria de componentes e restringiu possíveis erros de montagem. Com isso, foi possível realizar uma lista de materiais mais precisa que auxilia na gestão do projeto de montagem evitando que fosse interrompido por alguma falta de componente.

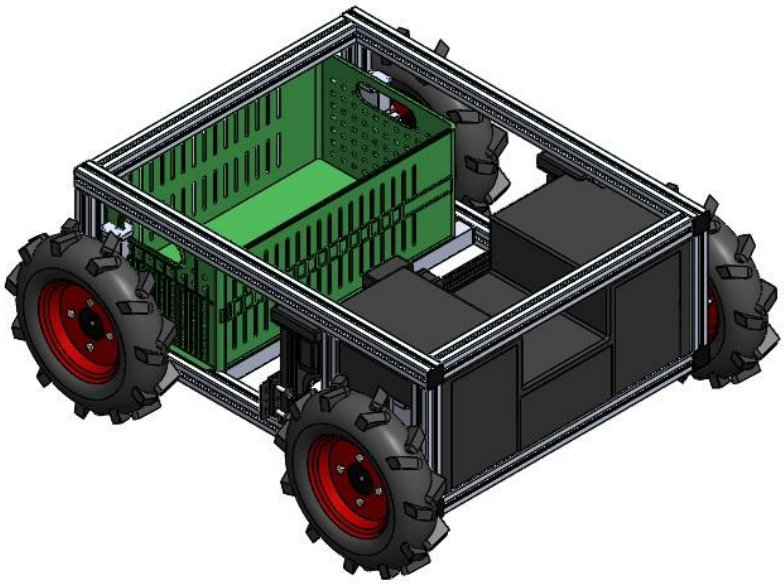

Fig. 18. Perspetiva isométrica do robô.

Após a montagem, foram realizados alguns testes em campo em que foi avaliada a sua manobrabilidade e deslocamento, que o mesmo executou sem maiores problemas. Entretanto, foi

verificado algumas dificuldades ao realizar transporte de carga, como pedras e ramos caídos, pois o robô não conseguiu manter uma velocidade constante dada a derrapagem das rodas traseiras. O sistema de pulverização também foi testado e não apresentou problemas. Na Fig. 19 é possível observar o veículo em operação no campo.

Fig. 19. Plataforma robótica em operação no campo.

3.1.2. Sistema de Locomoção

Estão montados no robô, 4 motores de passo NEMA 23 com caixa redutora que fornece um binário de 1 Nm, por motor. Com isso, o veículo possui tração nas quatro rodas. Os motores possuem um driver controlador embutido, que é controlada por meio de um microcontrolador ESP32. A alimentação é provida por duas baterias LIPO de 10000 mAh. As baterias foram posicionadas na frente do robô com finalidade de contrapesar e equilibrar o robô, pois os sistemas de pulverização e a caixa coletora de pêssegos encontram-se na parte traseira. Na Fig. 20 pode ser visto o motor e a bateria utilizada na plataforma robótica.

Fig. 20. (a) Motor de passo NEMA 23, (b) Baterias LIPO de 10000 mAh.

Para realizar as inúmeras tarefas inerentes à plataforma de modo independente, possui duas câmaras, nas quais uma tem a função de detetar ervas daninhas e pêssegos e a outra auxilia na navegação combinada com o GPS e um LIDAR. As imagens captadas por essas câmaras serão utilizadas para criação de uma aplicação de realidade aumentada baseada em WEB que mostrará em tempo real imagens virtuais para auxiliar o supervisor das operações na melhor tomada de decisão inerente aos processos a serem realizados (Simões, et al., 2021).

3.1.3. Arduino MEGA

O Arduino é considerado uma plataforma de hardware *open source*, projetado, tendo como base, o microcontrolador Atmel AVR que utiliza programação semelhante ao C/C++. Com auxílio dessa plataforma, é possível criar projetos interativos, com auxílio da computação física, ou seja, unindo o *software* diretamente com o *hardware*. Com isso, é possível controlar componentes eletrónicos, como sensores e atuadores, tornando possível a construção de sistemas que interagem com o ambiente que estão inseridos (Arduino, 2018).

A placa Arduino é composta por um microcontrolador, memória RAM, memória Flash (armazenamento secundário e clock, além de outras funcionalidades. Essa plataforma pode ser alimentada utilizando cabos USB ou através de uma fonte de 9 V, podendo ser controlados por um computador auxiliar ou desconectados para trabalhar de maneira autónoma, caso que será abordado nesta dissertação. Na Fig. 21 é apresentado um Arduino MEGA 2560, que será utilizado nesta dissertação (MONK, 2013).

Fig. 21. Arduino MEGA 2560 (Anon., 2021).

O Arduino Mega2560 é uma placa microcontrolada baseada no processador ATmega2560. É composta por 54 pinos de entrada ou saída digital, entretanto apenas 15 podem ser utilizados como saída PWM e 16 como entradas analógicas. Possui também 4 UARTs que são portas série de hardware, um oscilador de cristal de 16 MHz, uma conexão USB, um conector de alimentação,

um conector ICSP, além de um botão de *reset* (Anon., 2021). Na Tabela 2, estão expostas as especificações técnicas do Arduino Mega 2560.

Tabela 2. Especificações técnicas da placa Arduino Mega 2560 (Anon., 2021).

Parâmetro	Valor
Tensão operacional	5 V
Tensão de entrada (recomendado)	7-12 V
Tensão de entrada (limite)	6-20 V
Pinos de E/S digitais	54 (dos quais 15 fornecem saída PWM)
Pinos de entrada analógica	16
Corrente DC por pino de E/S	20 mA
Corrente DC para pino de 3,3 V	50 mA
Memória flash	256 KB, dos quais 8 KB usados pelo bootloader
SRAM	8 kB
EEPROM	4 kB
Velocidade do relógio	16 MHz
LED_BUILTIN	13
Comprimento	101,52 mm
Largura	53,3 mm
Peso	37 g

3.1.4. Módulo WIFI ESP8266 ESP-01

O módulo WIFI ESP8266 ESP-01 é um dispositivo de IoT que possui um microprocessador ARM de 32 bits que possui suporte a rede WI-FI e, também, possui memória flash integrada. Por conta disso, o módulo pode ser programado de forma independente, podendo trabalhar integrado ou não com o Arduino. Quando utilizado junto ao Arduino, agrega a conexão sem fio à placa de desenvolvimento, o que aumenta sua gama de utilização, pois torna-se possível a integração do Arduíno com smartphones, tablets ou qualquer outra aplicação WEB. Na Tabela 3, está explicitado as especificações do módulo WIFI ESP8266 ESP-01.

Tabela 3. Especificações do módulo WIFI ESP8266 ESP-01 (ESPRESSIF SMART, 2013).

Parâmetro	Valor
Tensão operacional	3,3 Vdc
Tensão de entrada (rec.)	7-12 V
Suporte de redes	802.11 b/g/n
Alcance	90 m
Comunicação:	*Serial* (TX/RX)
Suporta comunicação	TCP e UDP
Conectores	GPIO, I2C, SPI, UART, ADC, PWM e Sensor int. temperatura.
Modo de segurança	OPEN/WEP/WPA_PSK/WPA2_PSK/WPA_WPA2_PSK
Dimensões	25 x 14 x 1 mm
Peso	7 g

O dispositivo foi projetado com o objetivo de alcançar o menor consumo de energia com uma combinação de técnicas que operam em 3 modos, denominado modo ativo, modo sono e modo de suspensão profunda. Essas técnicas avançadas de gestão de energia servem para controlar a

alternância entre os modos ativos e o modo de hibernação. Sendo assim, o módulo consome menos que 10 μA em modo de espera e menos 1,0 mW para ficar conectado ao ponto de acesso (ESPRESSIF SMART, 2013). Além disso, quando o modo de hibernação está ativo, o relógio em tempo real permanece ativo. Dessa forma, o módulo pode ser programado para despertar à medida que forem detetadas certas condições. Sendo assim, o dispositivo pode ser programado para ligar apenas quando a comunicação WI-FI for necessário. Na Fig. 22 é possível observar o ESP8266 ESP-01 e o modelo esquemático dos seus pinos.

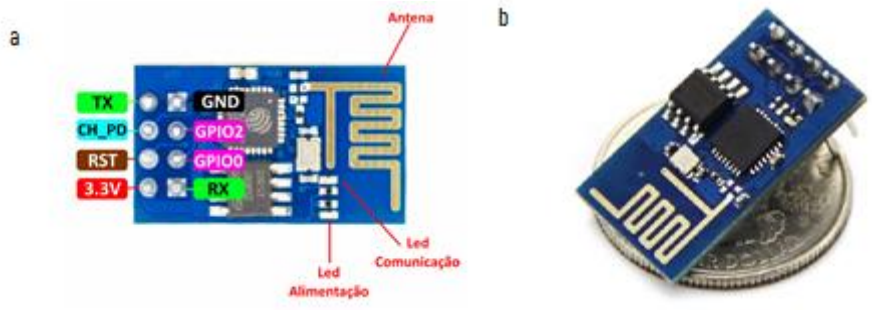

Fig. 22. (a) Modelo esquemático dos pinos e (b) Comparação do módulo ESP8266 ESP-01 com uma moeda (Almeida & Rodrigues, 2016).

3.1.5. Módulo GPS NEO-6M

O módulo GPS NEO-6M é um recetor de GPS autónomo compatível com Arduino e que utiliza o motor de posicionamento de alto desempenho u-box 6. Esses recetores são bastante flexíveis e económicos, além de oferecerem várias opções de conectividade. Essa placa é ideal para dispositivos móveis que utilizam bateria, os quais, em geral, possui restrições de espaço. Na Fig. 23 é exposto o módulo GPS NEO-6M explicitando os pinos.

Fig. 23. Pinos do módulo GPS NEO-6M (Anon., 2021).

O módulo possui quatro pinos de saída que se encontram descritos na Tabela 4. O módulo e a interface de comunicação são alimentados através desses quatro pinos.

Tabela 4. Especificações do módulo WIFI ESP8266 ESP-01 (u-blox, 2011).

Pino	Descrição
Vcc	Pino de potência positiva
RX	PIN de receção UART
TX	Pino de transmissão UART
GND	Neutro

O dispositivo possui a funcionalidade de gerar e armazenar automaticamente dados orbitais precisos de satélite que podem ser utilizados para futuras correções de posição do GPS. Esses dados podem ser obtidos sem necessidade de conexão com internet e funciona unicamente com dados de efemeridades de satélites que são transmitidos anteriormente e obtidos pelo recetor de GPS. Está integrada, também, a solução U-blox ADR que combina dados do GPS com outros sensores digitais usando um Filtro de Kalman a fim de melhorar a precisão da posição durante os períodos de ausência ou degradação do sinal de GPS (u-blox, 2011).

3.1.6. Módulo Bússola Eletrónica HMC5883L

O módulo Bússola HMC5883L permite a determinação da direção através da medida do campo magnético terrestre. A comunicação com o processador é baseada no protocolo I2C e a placa está equipada com sensores magnéticos resistivos HLMC118X de alta resolução e um conversor analógico-digital (Analog-Digital Converter – ADC) de 12-bit que garantem uma precisão na direção de 1º a 2º. Estão embutidas na bússola eletrónica as ferramentas de condicionamento de sinal como amplificador, driver cinta de desmagnetização e cancelamento de compensação. Quando utilizado com uma fonte de alimentação, ele utiliza uma tensão reguladora MIC5205-2,5, o que, permite a utilização da placa junto a qualquer fonte de alimentação de 3,3 Vcc a 6 Vcc. A Fig. 24 mostra o módulo bússola eletrónica HMC5883L de três eixos (Honeywell, 2013).

Fig. 24. Módulo bussola eletrónica HMC5883L (ElectronicWings, 2021).

As relações dos eixos X, Y e Z com a posição física da placa pode ser observada através da indicação impressa na placa. Entretanto, este módulo apresenta um valor, offset, somado a ela, e por conta disso, é imprescindível realizar um processo de calibração, em que são aferidas as leituras de modo que o módulo é rodado e, posteriormente, são realizadas novamente as medidas a fim de calcular o offset de cada eixo tendo estas como base. Após a obtenção da medida da intensidade do campo magnético nos eixos referenciais, as relações trigonométricas permitem a obtenção do ângulo de direção em relação ao norte magnético da Terra.

3.1.7. Módulo ultrassónico HY-SRF05

O módulo HY-SRF05 é um sensor ultrassónico frequentemente utilizado em robótica para a detetar possíveis obstáculos, e também, pode ser usado para mensurar distância. Esse módulo é a evolução do HY-SRF04, tendo sido melhorado o alcance para 4 m, que na versão anterior era de 3 m. Na Tabela 5 estão descritas algumas características do sensor ultrassónico HY-SRF05 (DataSheet4U, 2008).

Tabela 5. Especificações do módulo HY-SRF05 (REES52, 2021).

Parâmetro	Valor
Tensão de operação	5 V
Ângulo de deteção	15 a 35 graus
Faixa de deteção	2cm - 4m
Frequência	40 kHz

O módulo ultrassónico envia um sinal, gatilho, e recebe um pulso de eco proporcional à distância. Se a largura do pulso for medida em µs, então, ao dividir por 58 obtém-se a distância em cm e se dividida por 148 será fornecida a distância em polegadas. O SRF05 pode ser disparado tão rápido quanto a cada 50 ms antes do próximo gatilho, mesmo que seja detetado um objeto próximo e o pulso de eco se torne mais curto. Na Fig. 25 é possível observar o sensor ultrassónico HY-SRF05 (REES52, 2021).

Fig. 25. Módulo ultrassónico módulo HY-SRF05 (Solectro, 2021).

3.1.8. ESP32

O microcontrolador ESP32 é desenvolvido pela empresa Espressif Systems e é uma série de baixo custo e baixo consumo de energia num chip com wi-fi integrado e Bluetooth de modo duplo. Essa série pode possuir um microprocessador Tensilica Xtensa LX6 com a opção single-core e dual-core, microprocessador Xtensa LX7 dual-core ou um microprocessador RISC-V single-core. O ESP32 inclui interruptores de antenas integrado, amplificadores de recessão de baixo ruido, amplificadores de potência, RF balun, filtros e módulos de gerenciamento de energia (ESP32net, 2017). A Fig. 26 mostra uma Placa ESP32 Wroom-32.

Fig. 26. Placa ESP32 Wroom-32.

3.2. Métodos

3.2.1. Ambiente de desenvolvimento integrado - Arduino IDE

O software Arduino IDE possui código aberto e é uma ferramenta que facilita a escrita de códigos e a sua transferência para o Arduino. Adicionalmente, possui uma interface de edição bastante intuitiva e ágil, principalmente pelas novas ferramentas de preenchimento automático, navegação de código e um depurador em tempo real. Um programa desenvolvido para ser embutido no Arduino é denominado Sketch e pode ser visto na Fig. 27 (Arduino, 2021).

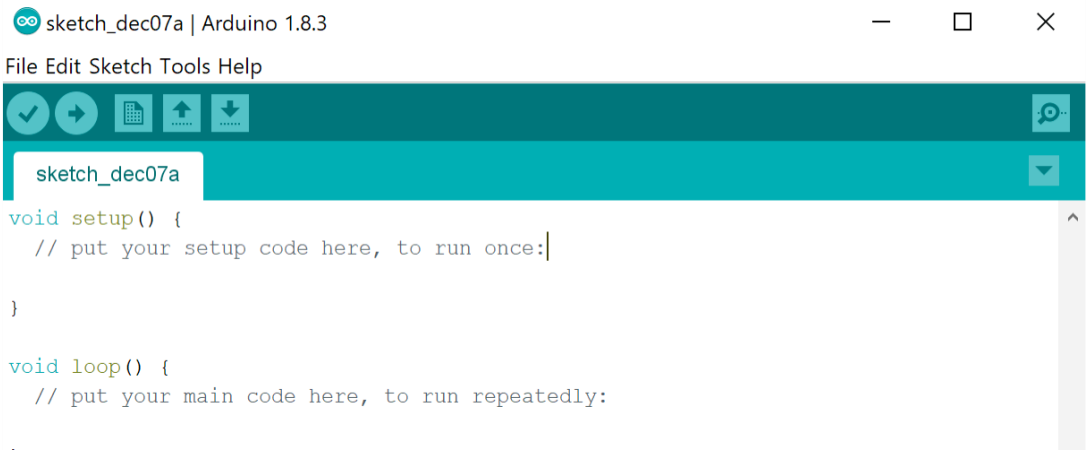

Fig. 27. Sketch no Arduino IDE (Microsoft, 2021).

A linguagem de Programação do Arduino IDE é chamada de Wiring e sua interface gráfica é desenvolvida em Java. Após o desenvolvimento do código-fonte, o compilador avalia sua sintaxe e torna o sketch acessível para utilização no Arduino após ser transferido via cabo USB. Em geral, a estrutura básica de um sketch desenvolvido em Wiring é formado por duas funções principais.

A primeira é a `Void setup()`, que é utilizada com finalidade de iniciar as configurações do programa do Arduino. De tal forma que são definidas as configurações iniciais com os pinos que serão usados e de que forma eles serão utilizados (input ou output). Todos os comandos inseridos nessa função são executados apenas uma vez ao iniciar o programa. Já a segunda é a função `Void loop()`, que se comporta como um ciclo de repetição, sendo executada todas as vezes enquanto o Arduino estiver ligado. De modo que, após a execução de comando da última linha, o sketch será lido novamente, e assim acontece ininterruptamente até que o Arduino seja desligado ou seja feito o reset.

A forma mais comum de sintaxe para escrita dos comandos do Arduino funciona da seguinte forma. O ponto e virgula (;) é indispensável para indicar o final de uma linha de comando; as chavetas ({ }) representam o bloco de comando, sendo bastante utilizadas nas funções e outras estruturas de comandos; Os símbolos (/ /) ou (/* */) indicam comentários numa linha ou em mais de uma linha respetivamente, ou seja, são os trechos dos códigos que não serão executados pelo compilador, sendo todavia, são demasiado importante para a organização programador ou de outra pessoa que tenta compreender o código executado.

3.2.2. Linguagem Hyper Text Markup Language (HTML)

HTML é a linguagem de marcação padrão para o desenvolvimento de páginas na web. O termo significa Hyper Text Markup Language e seu código descreve a estrutura de uma página web, contendo os elementos HTML que informa ao navegador o modo de exibição do conteúdo e elementos que identificam que tipo de conteúdo é exibido, por exemplo, título, parágrafos, links e etc. Na Fig. 28 é exposto um exemplo de código HTML simplificado.

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>My First Heading</h1>
<p>My first paragraph.</p>

</body>
</html>
```

Fig. 28. Código HTML simplificado (W3, 2021).

Uma página da web é um elemento de front-end, que interage diretamente com o utilizador, que regra geral, é desenvolvida usando um código HTML e posteriormente é conectada a um navegador web que é uma interface HTTP. Esse código tem uma estrutura composta por tag que é representada por um código entre os sinais maior que e menor que <>. A tag `<!DOCTYPE html>` indica que este documento é um código HTML5, a tag `<html>` define que o elemento é a raiz de uma página HTML, enquanto a tag `<body>` e `<p>` definem o corpo do documento, ou seja, a parte visível da página e os parágrafos respetivamente (Park, et al., 2018).

A versão mais atual do HTML é denominada HTML5 e é considerada por muitos desenvolvedores e empresas como um padrão de desenvolvimento para web podendo ser utilizado em aplicações nativas da web ou hibridas pois, nessa versão, essas diferenças foram significativamente reduzidas. Logo, uma aplicação desenvolvida em sistema local possui estrutura semelhantes às aplicações nativas da web como fornecimento de informações sobre tratamento de erros, suporte offline, armazenamento de dados, conectividade de rede, multimédia, desenhos de animação e controlo de formulários avançados. Um recurso muito importante do HTML5 é o armazenamento do lado do utilizador, podendo ser armazenado até 5 MB de dados pelo cliente da aplicação web (Litayem, et al., 2015).

3.2.3. Linguagem Cascading Style Sheets (CSS)

CSS significa Cascading Style Sheets e é uma linguagem usada para definir o estilo de uma página web, ou seja, descreve a forma com o que os elementos HTML serão mostrados na tela. Nesse sentido, o CSS reduz o esforço computacional pois pode controlar várias páginas de uma só vez. O CSS é usado para a definição de estilos para as páginas da web, incluindo o design, layout e variações de exibições para diferentes dispositivos e tamanhos de tela. Sendo assim, com a adição do CSS, a formatação de estilo foi retirada do código HTML, o que foi um grande avanço para computação baseada em web.

Em geral não há nenhuma desvantagem em usar CSS como folha de estilo, entretanto, existem alguns problemas persistentes de inconsistência do navegador. Porém, esses erros podem ser evitados ao utilizar um navegador adequado. O CSS é uma ferramenta de design robusta e poderosa, e não possui elementos de imagens, entretanto, todos os ficheiros .img são usados como fundo. Ao separar o CSS da estrutura, é possível criar páginas bastante sofisticadas como pode ser observada na Fig. 29 (Robbins, 2012).

Fig. 29. Página web com designe baseado em CSS (CSS ZEN GARDEN, 2021) .

Diferentemente do HTML que é uma linguagem de estrutural, o CSS complementa-o, pois, é uma linguagem estilística. Além disso, o CSS também oferece recursos simples que fornecem ao desenvolvedor um maior controlo sobre a aparência da página da web. A internet tornou-se mais competitiva, visto que é possível criar anúncios de publicidade muito mais atrativos e com recursos mais agradáveis.

3.2.4. Linguagem JavaScript

JavaScript é uma linguagem de programação que permite aos desenvolvedores interagir com os navegadores web. O JavaScript é uma linguagem de script em que o código é executado a cada bite e, praticamente, o seu objetivo consiste na modificação do comportamento de outra aplicação que normalmente foi programada em outra linguagem, que é executada e em tempo real. No caso mais comum, JavaScript pode modificar códigos feitos em HTML e estilos em CSS. Esta linguagem tornou-se muito importante para o desenvolvimento da web e atualmente é a bastante utilizada para criação de aplicações que interagem com a Web (Theisen, 2019).

A linguagem JavaScript possui os aspetos comuns à grande maioria das linguagens de programação, como exemplo, as funções e ciclos. Entretanto, esta linguagem apresenta uma série de diferenças significativas nos seus objectivos e no *namespace* global. As instruções em JavaScript são compostas de valores, operadores, expressões, palavras-chaves e comentários.

Um código desenvolvido em JavaScript deve ser executado para uma página da web, logo, o seu código é executado carregando uma página da web no navegador. Contudo, devido à necessidade

de utilização do JavaScript para aceder a todos os aspetos de uma página na internet, a sua construção principal é basicamente um grande conjunto de recursos denominado objetivo global. A união de todas as variantes definidas no objetivo global é denominada *namespace* global.

Como a linguagem JavaScript está sempre a evoluir, é necessário que os navegadores estejam sempre atualizados, pois um navegador mais antigo pode nãos ser compatível com os recursos dessa linguagem. Em geral, os desenvolvedores da linguagem têm o cuidado de não realizarem alterações que possam desabilitar programas já existentes. Desse modo, novos navegadores podem executar programas antigos. Esta condição é importante, visto que, os navegadores web não são as únicas plataformas onde o JavaScript é usado, pois ele pode ser utilizado, como script e linguagem de consulta, em bancos de dados como MongoDB e CouchDB. Adicionalmente, diversas plataformas para programação de desktop e servidor, como o Node.js, fornece um ambiente para programar JavaScript fora do navegador (Haverbeke, 2018).

3.2.5. Ambiente de programação colaborativo - Glitch

Glitch é um ambiente de programação colaborativo que reside no navegador e implanta o código automaticamente de acordo com o que é digitado. A plataforma Glitch promove aos desenvolvedores um modo simples para a criação instantâneas de aplicações na web. Nesse ambiente de desenvolvimento, vários utilizadores podem colaborar em conjunto visando desenvolver o mesmo projeto ao mesmo tempo e todo podem observar as alterações em tempo real enquanto o estão a codificar. Na Fig. 30 é possível observar o ambiente de desenvolvimento do Glitch e ao lado direito é mostrado o resultado em tempo real do que foi programado (Glitch, 2021).

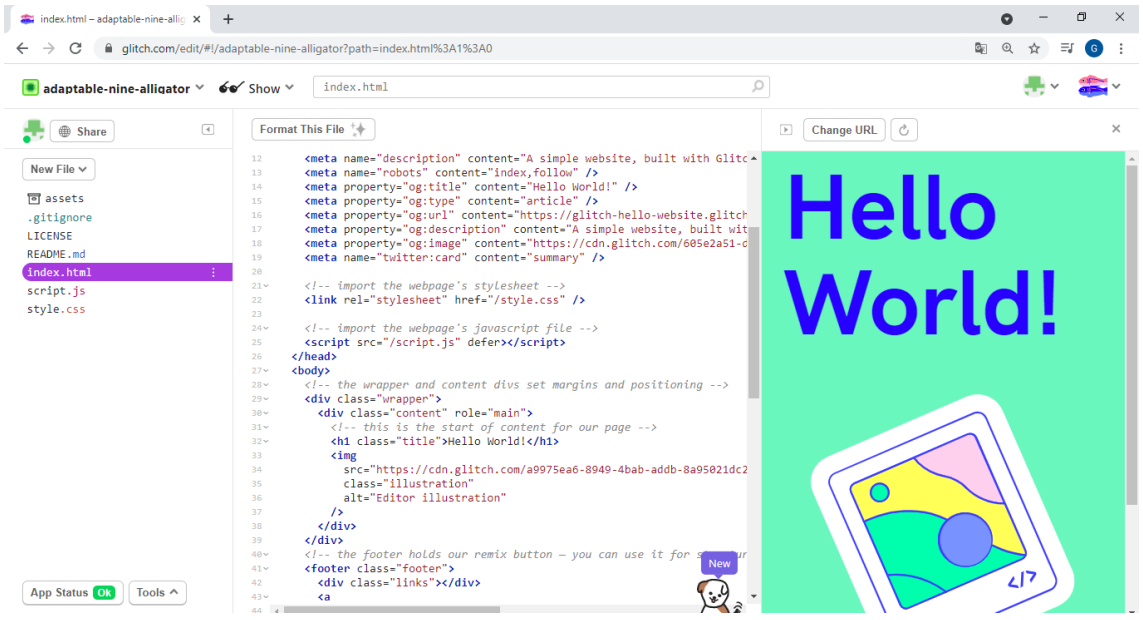

Fig. 30. Ambiente de desenvolvimento da plataforma Glitch.

Na plataforma Glitch é possível desenvolver aplicações a partir das estruturas de JavaScript mais populares como as bibliotecas *React* e o framework *Angular*, que criam interfaces de utilizadores na web e auxiliam na execução de aplicações de página única. Além disso, permite a integração com ferramentas de HTML e CSS. O Glitch também possui integração com plataformas como o GitHub, permitindo a importação e exportação entre ambas as ferramentas o que permite a colaboração com outros desenvolvedores de todo mundo, que poderão realizar teste na aplicação.

3.2.6. Biblioteca AR.js

A biblioteca AR.js é utilizada para criar aplicações de realidade aumentada (RA) baseadas em web. Essa ferramenta utiliza recursos como rastreamento de imagem, realidade aumentada baseada em localização e rastreamento de Marcador. Sendo também compatível com vários tipos dispositivos como smartphones, monitores de desktop e tablets. Deste modo, o desenvolvimento de realidade aumentada na web pode ser otimizado pela utilização de tecnologias como JavaScript, HTML e CSS e lançar as novas versões de forma instantânea, corrigir bugs e lançar novos recursos em tempo real. Na Fig. 31 mostra uma aplicação desenvolvida com AR.js baseada em localização por GPS (AR.js, 2021).

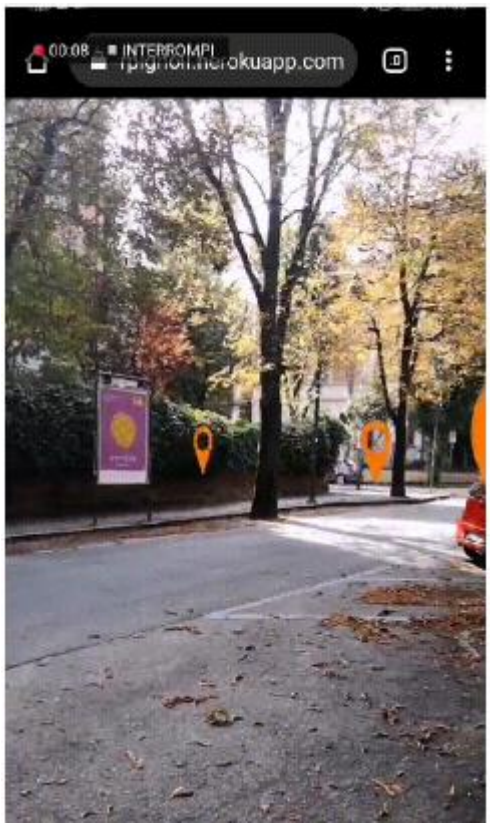

Fig. 31. Realidade aumentada baseada em localização (GPS) desenvolvida com auxílio do AR.js para web (Carpignoli, 2021) - Adaptado.

Através do AR.js é possível desenvolver três tipos de aplicações de realidade aumentada. O primeiro é o rastreamento de imagem, que consiste quando a câmara encontra uma imagem 2D e é gerado um conteúdo em cima da imagem ou, próximo a ela. O conteúdo pode ser uma imagem 2D, um GIF, um modelo 3D (que pode ser animado) ou um vídeo 2D. O segundo tipo de realidade aumentada é com base na localização. Este tipo de RA usa locais no mundo real para apresentar conteúdos de realidade aumentada, podendo ver diferentes tipos e formas de conteúdo ao se mover. E finalmente, o rastreamento de marcadores, que funciona de modo semelhante ao rastreamento de imagem, sendo os marcadores mais estáveis e mais limitados em forma, cores e tamanhos.

3.2.7. Framework A-FRAME

O framework A-Frame é uma estrutura da web baseada em html que auxilia na construção de experiência de realidade aumentada. Essa ferramenta foi concebida visando a facilitar o desenvolvimento de realidade aumentada e realidade virtual na web. Entretanto, o framework possui um núcleo bem estruturado e funciona como código aberto. Ele é compatível com a maioria dos óculos de realidade virtual, telemóveis e monitores de computador (A-FRAME, 2021)

O A-Frame é uma estrutura three.js (biblioteca que permite criar gráficos 3D em linguagem Javascript) que fornece aos desenvolvedores uma estrutura de componente de entidade declarativa, combinável e reutilizável que dão acesso a ferramentas para criação de realidade aumentada e realidade virtual. Com isso, as atualizações dos objetos 3D são realizadas na memória gerando pouco lixo e com pouca sobrecarga, o que permite criar aplicações mais interativas e de grande escala que atualizam suavemente a 90 fps.

3.3. Nota conclusiva

Em suma, o R2D2 contará com uma estrutura leve e robusta que é ideal para aplicação com precisão de herbicida e também para a apanha dos pêssegos caídos no chão. Com isso, o robô foi anteriormente projetado de forma que consiga operar próximo ao tronco das árvores sem precisar passar entre dois pessegueiros consecutivos.

Para o sistema de locomoção, o robô contará com quatro motores de passo NEMA 23 e duas baterias LIPO de 10000 mAh. O motor já possui driver integrado e sua rotação para realização das manobras será controlado por uma placa ESP32 que será conectado a placa Arduino MEGA, onde será processada toda lógica para a movimentação da plataforma. A navegação será feita baseada no posicionamento de GPS e o sensor utilizado é o GPS NEO-6M que será conectado ao Arduino. O robô também está provido com uma bussola eletrónica para o orientar no arranque

das movimentações e um módulo WIFI que ajudará no envio de informações para o supervisor das operações.

Toda programação do algoritmo de navegação autónomo do robô foi desenvolvida no Arduino IDE em linguagem C/C++ utilizando os princípios de Computação Física. Com isso é possível desenvolver um código que integre todos os sensores e com Arduino de forma a permitir que o robô opere de forma autónoma. Visando a melhoria da supervisão do processo, é utilizado uma plataforma web que interage com o supervisor ao gerar imagens em realidade aumentada que exibirá informações sobre a trajetória da plataforma. Essa aplicação web foi desenvolvida utilizando as linguagens HTML, CSS e JavaScript integrada com as bibliotecas AR.js e o framework A-Frame.

4. Análise de Discussão de Resultados

4.1. Introdução

O desenvolvimento do código de navegação tomou como base a aquisição de dados de longitude e latitude do local onde o robô se encontra e dos locais para onde o veículo se deve dirigir, também chamados de *waypoints*. Entretanto, para conduzir o robô até o próximo *waypoint*, é necessário calcular alguns parâmetros matemáticos.

A localização de destino é fornecida no algoritmo como ângulos de latitude e longitude. Após a fixação da localização atual, o algoritmo irá calcular a distância entre a localização atual e de destino. Com as coordenadas atuais e de destino, o código fornecerá o ângulo de rumo em relação ao Norte geográfico da Terra. E também calculará o angulo de direção atual do robô através da bússola eletrónica. A diferença entre ângulo de rumo dado pela bússola e o ângulo calculado pelo algoritmo será comparada para que as ações de movimentação do veículo sejam tomadas.

Após a decisão do ângulo de rumo, a plataforma robótica será conduzida até o local de destino pelo acionamento dos motores. E simultaneamente, é necessário que o módulo GPS atualize as informações do posicionamento atual do robô. Ou seja, à medida que o robô se aproxima das coordenadas de destino, o GPS atualizará a localização e a distância diminuirá. Nesse caso, assim que o robô atingir uma distância de 0.5 m do alvo indicado é considerado que ele atingiu o *waypoint* desejado e passará a operar em direção ao próximo *waypoint*.

4.2. Algoritmo de movimentação autónoma baseado em GPS

4.2.1. Cálculo da distância

Para que ocorra a movimentação faz se necessário o cálculo da distância entre a localização atual e a posição alvo. Nesse intuito, foram utilizadas as equações de Haversine, as quais, calculam a distância entre dois pontos na superfície, aproximadamente esférica, da Terra usando equações trigonométricas, como pode ser visto na Fig. 32. Para isso, é desconsiderado o fato que a Terra é

ligeiramente elíptica. Os pontos utilizados na fórmula matemática são a latitude e a longitude dos pontos (em radianos), conforme Equação (9) a Equação (12) (Ikasari, et al., 2021).

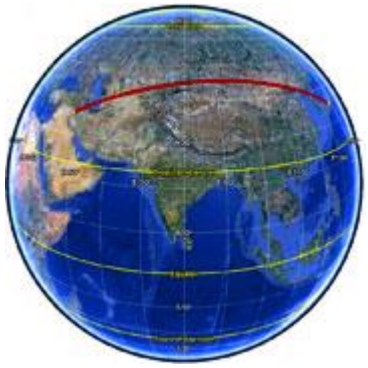

Fig. 32. Distância de Bagdá a Osaka.

$$Haversine\left(\frac{d}{R}\right) = Haversine(\theta_1 - \theta_2) + cos(\theta_1)\,cos(\theta_2)Haversine(\lambda_2 - \lambda_1) \quad (9)$$

$$Haversine(\theta) = sin^2\left(\frac{\theta}{2}\right) = \frac{1-cos\,\theta}{2} \quad (10)$$

Em que:

θ_1= Latitude do ponto 1 (localização do veículo)

θ_2= Latitude do ponto 2 (localização de destino)

λ_1= Longitude do ponto 1 (localização do veículo)

λ_2= Longitude do ponto 2 (localização de destino)

R = Raio da terra (6378140 m).

d = Distância entre dois pontos.

d_{normal} = Distancia normal entre doi pontos

Logo:

$$d = R \times 2 \times sin^{-1}\left(\sqrt{sin^2\left(\frac{\theta_2-\theta_1}{2}\right) + cos(\theta_1)\,cos(\theta_2)\,sin^2\left(\frac{\lambda_2-\lambda_1}{2}\right)}\right) \quad (11)$$

Bem como a equação de Haversine, existe a formulação da distância normal que permite calcular a distância ente qualquer dois pontos na Terra. A distância normal é expressa pela Equação (12):

$$d_{normal} = R \times \sqrt{2} \times \sqrt{1 - (cos(\theta_1)\, cos(\theta_2)\, cos(\lambda_2 - \lambda_1) - sin\,\theta_1\ sin\,\theta_2)} \quad (12)$$

Desta forma, a fórmula de Haversine calcula a distância entre quaisquer dois pontos na superfície da Terra considerando que o planeta possui uma superfície esférica. Já a fórmula da distância normal, faz aproximações e considera que a superfície da Terra é plana, ou seja, não considera a curvatura esférica da Terra. Para pequenas distâncias, a diferença entre essas duas expressões é demasiado pequena em comparação com os resultados reais. Contudo, com o aumento da distância, as diferenças entre essas duas equações aumentam e a fórmula da distância normal fornece resultados imprecisos. Na Tabela 6 é realizada a comparação da distância entre dois pontos calculada através da equação de Haversine e pela distância normal.

Tabela 6. Comparação entre as fórmulas de Haversine e da Distância Normal (Paradkar & Sciortino, 2016).

Coordenadas do pondo de origem (latitude, longitude)	Coordenadas do pondo de destino (latitude, longitude)	Distância Normal [km]	Distância por Haversine [km]	Distância por Google [km]
39.28357, -120.51933	39.61731, -120.53251	37,1	37,1	37,1
42.69078, -119.08012	44.33957, -120.28325	207,4	207,4	207,4
39.93950, -115.60020	43.05685, -119.69549	486	486,1	486,3
42.59784, -91.21056	46.90922, -118.81658	2208.4	2219,6	2220,3
62.78254, -108.78868	60.14424, 135.18732	5207.1	5318,2	5319,7
55.413, -95.780	57.031, 63.46856	7168.3	7372,9	7375,04

Através do método de Haversine é possível obter resultados mais precisos, por isso esta formulação foi utilizada nesse projeto para cálculo de distância entre quaisquer duas coordenadas na superfície da Terra. Entretanto, é necessário utilizar os valores das coordenadas em radiano, ou seja, multiplicando o valor em grau por $\frac{\pi}{180}$. Após a conversão, a latitude e a longitude para radianos, os valores são inseridos na fórmula de Haversin para obtenção da distância final.

No algoritmo, a distância calculada é continuamente verificada, sendo imposta uma condição para que quando a distância final for menor ou igual que 0,5 m do *waypoint*, é assumido que o robô encontrou o seu alvo. Decerto, essa fórmula é muito precisa e muito útil para programação de algoritmos de navegação.

4.2.2. Definição do rumo do robô

O Ângulo de direção é definido como o ângulo entre o ponto em que o robô está localizado e o próximo *waypoint*. Isto é, o ângulo entre a linha norte-sul da Terra ou meridiano e a linha que liga os dois pontos na superfície terrestre. Com isso, esse método é usado para que o robô se desloque na direção correta. A equação (13), em que h é o ângulo de direção, calcula o rumo inicial entre dois pontos, pois o rumo irá variar conforme o veículo percorre uma grande distância, ou seja, o rumo final será diferente do inicial de vários graus de acordo com a latitude e a distância do alvo (Movable Type Scripts, 2021).

$$h = atan2(sin(\lambda_2 - \lambda_1)\, cos(\theta_2)\,, cos(\theta_1)\, sin\,\theta_2 - sin\,\theta_1\, cos(\theta_2)\, cos(\lambda_2 - \lambda_1)\,) \quad (13)$$

$$h_{final} = h - h_{atual} \quad (14)$$

O ângulo do rumo é calculado em radianos e em relação ao norte geográfico. Após o cálculo do rumo, o seu valor é comparado com o ângulo atual do robô, h_{atual}, mensurado através da bússola eletrónica. Portanto, a diferença entre os dois ângulos exprime o rumo final, h_{final} , no qual a plataforma autónoma deverá seguir para alcançar o seu alvo, como pode ser visto na Equação (14). Esse processo é reiterado sempre que o GPS recebe a localização instantânea do veículo. Logo após o cálculo do rumo final, é preciso que a plataforma decida a direção que tomará para o norte, sul, leste ou oeste. O ângulo de azimute é definido como o ângulo que uma direção faz com outra direção de referência (Vicente, 1997), medido no sentido anti-horário, que neste algoritmo, é a direção do Norte Magnético, N. O valor deste ângulo foi utilizado para a tomada de decisão de qual direção a plataforma robótica irá seguir para atingir o seu destino final, como pode ser observado na Fig. 33.

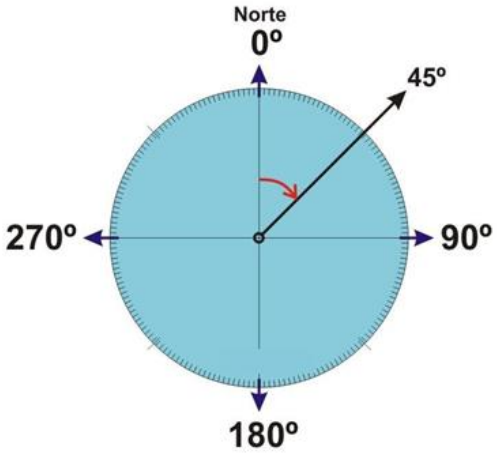

Fig. 33. Exemplo de ângulo de Azimute (Sodelli, 2021).

Entretanto, devido à imprecisão da bússola eletrónica, é utilizado um grande intervalo dentre os ângulos de azimute para a tomada de decisão. De modo que, se o ângulo estiver entre 225° e 315°, o robô deve se locomover em direção ao oeste, ou seja, para esquerda. Se estiver entre 45° e 135°, a plataforma mover-se-á para a direita, em direção ao leste. Da mesma forma, na medida que o ângulo de direção estiver entre 315° e 45°, o robô mover-se-á para frente em direção ao norte. A Fig. 34 mostra um esquema da movimentação do robô.

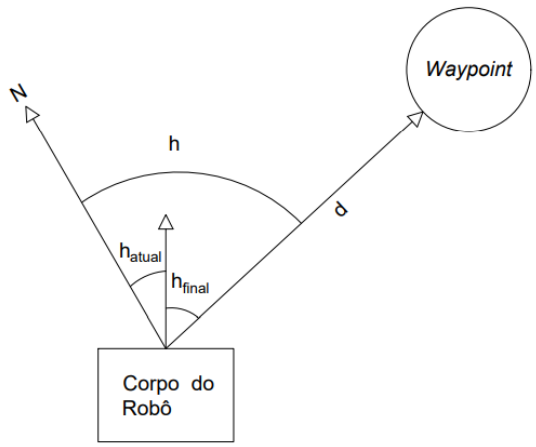

Fig. 34. Esquema do vetor navegação do robô.

4.2.3. Fluxograma de comando

É proposto um algoritmo para a navegação autónoma da plataforma robótica como apresentado no fluxograma da Fig. 35. A priori, após o arranque, os pinos de comando dos motores, do módulo GPS, da bússola eletrónica e do sensor ultrassónico são ativados de acordo com o modo que foram programados. Nesta etapa, o GPS localiza a posição atual do robô pela primeira vez e, posteriormente, são armazenadas duas variáveis como latitude e longitude.

As coordenadas de destinos serão definidas no algoritmo. Com elas, é possível calcular a distância necessária a ser percorrida entre os dois pontos dada pela Equação (11). Adicionalmente, o rumo atual do rover robótico é aferido usando a bússola eletrónica e o ângulo de direção é calculado usando a Equação (13). Com a finalidade de direcionar a plataforma no rumo correto, o rumo final é calculado com auxílio da Equação (14). Após a realização dessas etapas, o robô começa a mover-se em direção do primeiro *waypoint*, e permanecerá a mover-se até que a distância para o seu alvo seja menor que 0,5 metros.

Devido à dificuldade do equipamento de redução do rumo final a zero, que indicaria o exato rumo para atingir o ponto de destino, foi definido uma faixa de tolerância de ± 5°. Portanto, qualquer valor dentro do intervalo entre -5 a 5°, o veículo seguirá rumo ao seu alvo.

Se a condição de distância mínima de 0,5 m for satisfeita, significa que o robô atingiu seu local de destino. Logo após, o contador é incrementado e o processo é repetido para o robô seguir ao próximo *waypoint*. Entretanto, se o sensor de ultrassónico detetar um obstáculo a menos de 0,5 m, o robô para imediatamente até a remoção do obstáculo. Essa paragem de emergência foi projetada visto que o risco de colisão com as filas de plantas é demasiado elevado, já que a largura do veículo é de 1050 mm e dependendo do tamanho do obstáculo, poderá ocorrer alguma colisão com a cultura ao tentar contornar o obstáculo. Não no âmbito desta dissertação encontra-se a ser desenvolvido um sistema de navegação para esta plataforma robótica destinado a navegação de proximidade, i.e., vocacionado para contornar obstáculos e providenciar uma solução de navegação mais precisa baseada em visão computacional.

Porém, como salvaguarda, foi necessário criar um alerta para indicar ao supervisor da operação que existe um obstáculo que impede a passagem do rover, podendo ser necessário que esse vá ao encontro do veículo para desobstruir o caminho. Nesse aspeto, a realidade aumentada torna-se muito útil para gerar imagens virtuais de alertas que auxiliam o supervisor da operação. Logo, é mais fácil notar a ocorrência de alguma obstrução entre as filas de árvores que impeça a passagem do robô.

Foi utilizado o Google Maps para adquirir as coordenadas dos *waypoints* a fim de definir a rota na qual o robô segue. Para isso, é possível marcar alguns pontos de destino no Google Maps e, posteriormente, copiar as coordenadas clicando com o botão direito do rato e clicando na latitude e longitude apresentada. Ao realizar esses passos, as coordenadas serão automaticamente copiadas e podem ser coladas na matriz das coordenadas no algoritmo.

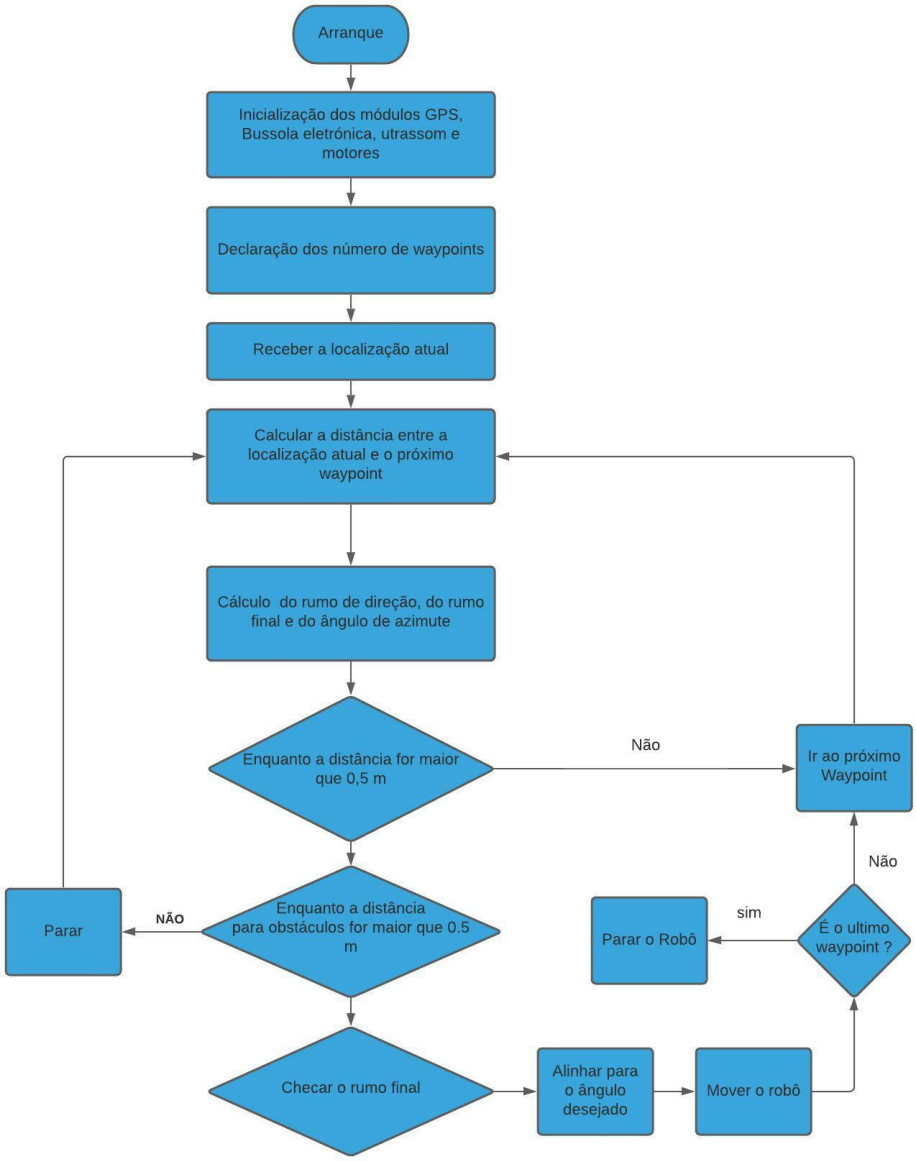

Fig. 35. Fluxograma de operações.

4.3. Acionamento dos Motores

Os 4 motores de passo NEMA 23 com caixa redutora estão ligados ao ESP-WROOM-32, como pode ser visto na Fig. 36. Os pinos 13, 12 e 11 do Arduino Mega são ligados nos pinos 17, 18 e 20 (GPIO 14, 12 e 13) do ESP-WROOM-32. Os três pinos do Arduino são definidos como saída e os três pinos do ESP são programados como entrada. O *pinout* do ESP utilizado é mostrado na Fig. 37.

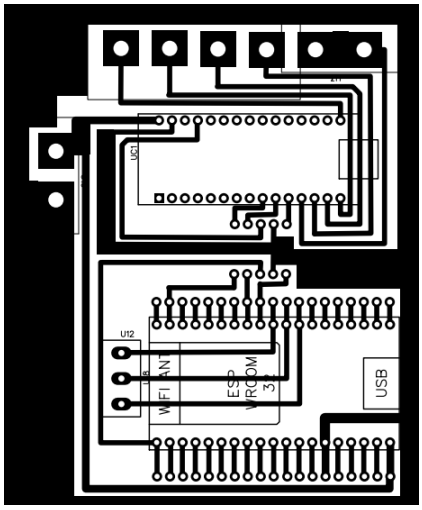

Fig. 36. Negativo do ESP WROOM 32 com os pinos utilizados para movimentação dos motores.

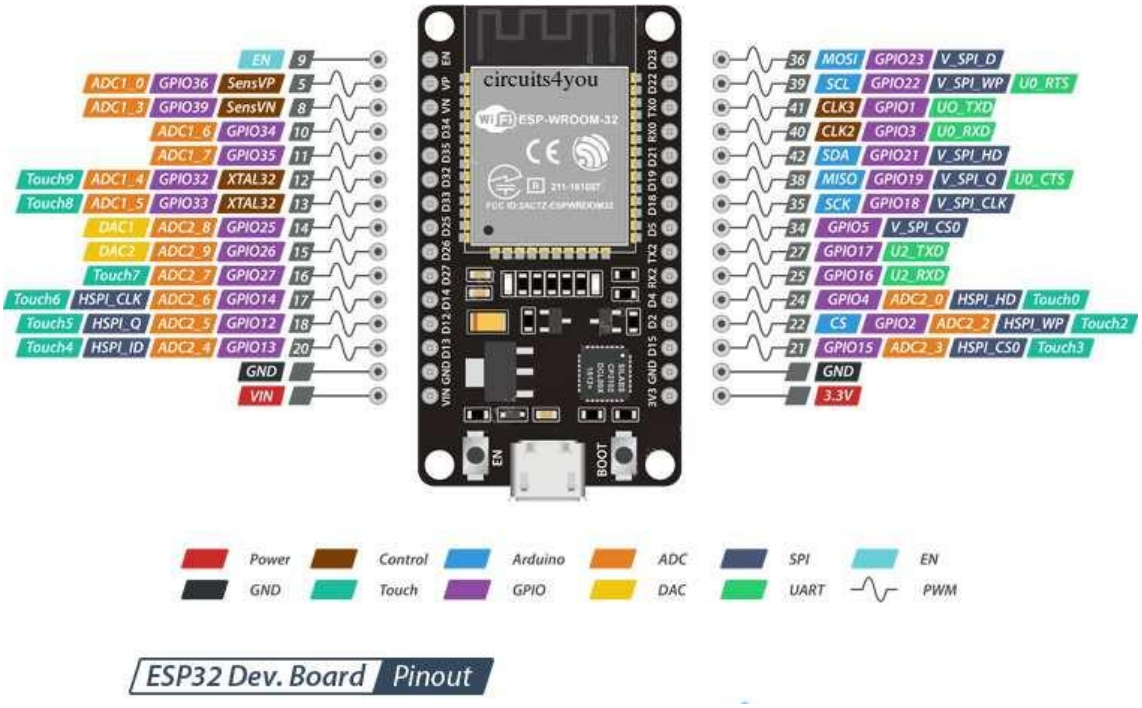

Fig. 37. *Pinout* do ESP-WROOM-32.

No Algoritmo de navegação autónoma baseado em GPS foram definidas cinco funções do tipo Void que representam a movimentação do robô. São elas `FOWARD ()`, `RIGHT ()`, `LEFT ()`, `REVERSE ()` e `STOP ()`. Cada função aciona os pinos 11, 12, 13 do Arduino para que quando a função `FOWARD ()` for acionada, os pinos são ligados da seguinte maneira `(HIGH, LOW, HIGH)` e o robô movimentar-se-á para frente. Da mesma forma, A função `REVERSE ()` aciona os pinos da forma (LOW, HIGH, LOW) e faz o veículo mover-se para trás. As funções `RIGHT ()` e `LEFT`

`()` acionam as saídas de forma `(HIGH, LOW, LOW)` e `(LOW, HIGH, HIGH)`, que fazem o rover mover-se para a direita e para esquerda, respetivamente. Finalmente, a função `STOP ()` aciona os pinos de forma (LOW, LOW, LOW) e faz o robô parar. Os pinos Vcc e GND do módulo GPS NEO 7M

4.3.1. Circuito de controlo

Foi ligado uma fonte de alimentação de 9 V à entrada de alimentação do Arduino Mega. A *protoboard* foi alimentada ligando a entrada positiva no pino 5 V e o neutro no pino GND do Arduino Mega. Os pinos Vcc e GND do módulo GPS foram conectados na linha de alimentação da *protoboard* e os pinos rx e tx foram conectados aos pinos 4 e 3 do Arduino Mega, respetivamente. Os pinos Vcc e GND do módulo bussola eletrónico foram conectados a linha de alimentação da *protoboard* e os pinos SDA e SCL foram conectados aos pinos 20 e 21 do Arduino Mega. E, finalmente, os pinos Vcc e GND do módulo ultrassónico foram conectados a linha de alimentação da *protoboard* e os pinos Trig e Echo foram ligados aos pinos 9 e 10 do Arduino Mega. Na Fig. 38 é possível observar as ligações no circuito de controlo do robô R2A2.

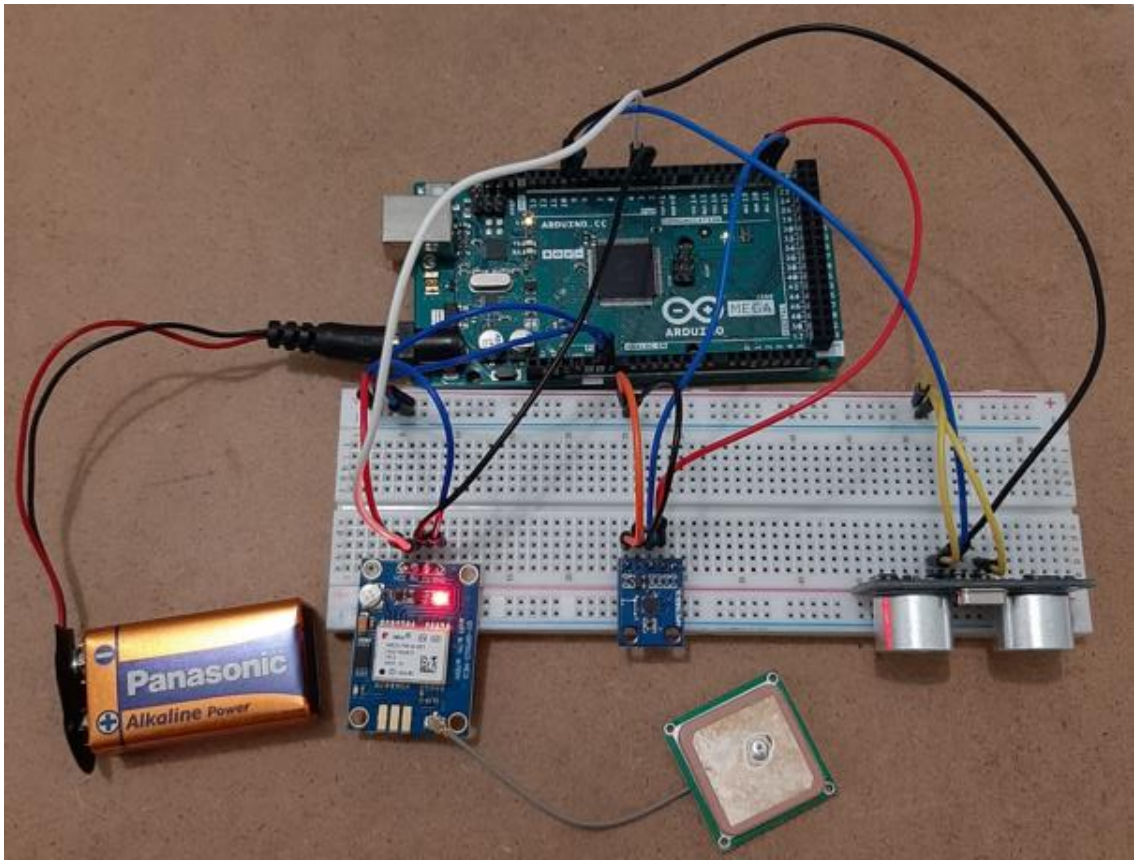

Fig. 38. Circuito de controlo do robô R2A2.

4.4. Aplicação de Realidade Aumentada baseada web

4.4.1. Realidade Aumentada com base em localização na web

A biblioteca AR.js possibilita a utilização da realidade aumentada baseada em localização na web. Com o auxílio dessa ferramenta foi possível desenvolver uma aplicação web que possibilita uma experiência de realidade aumentada para o supervisor das atividades quando a plataforma robótica autónoma estiver a ser utilizada. Dessa forma, a aplicação foi desenvolvida de modo que a imagem aumentada pode ser gerada com a utilização da localização geográficas de pontos específicos que são representados pelas coordenadas geográficas dos *waypoints* que serão utilizados no algoritmo de movimentação do robô.

A aplicação foi desenvolvida com AR.js e AR baseada em localização, de modo que as coordenadas geográficas foram adicionadas de forma estática usando HTML. Dentro da *tag* `<head>` foi importado o framework Aframe e a biblioteca AR.js (que contém o código baseado em localização). Nesse caso, o componente `arjs,` que está incluído na *tag* `<a-scene>,` inicializa o AR.js. O campo `sourceType` foi definido como `webcam` para que a página possa requisitar a câmara utilizada pelo sistema. O componente `videoTexture` foi definido como verdadeiro. Esta condição é essencial numa aplicação de realidade aumentada baseada em localização externa, pois permite que o conteúdo aumentado possa ser visualizado usando uma estrutura de three.js para o *feed* da câmara, que é combinado com o conteúdo aumentado.

O componente `gps-projected-camera,` que pertence à *tag* `<a-camera>`, converte de modo automático as informações de latitude e longitude em coordenadas mundiais 3D, permitindo o uso de coordenadas geográficas no ambiente de desenvolvimento. O `gps-projected-camera` facilita o uso de dados geográficos arbitrário fornecido por um servidor, pois utiliza a projeção Spherical Mercator para representar as coordenadas do conteúdo aumentado. As unidades esféricas de Mercator são comumente usadas para representar dados de mapeamento e são mais precisa quanto mais distante estiver o objeto dos pólos terrestres. Dentro do intervalo de 85°03'04.0636" norte a 85°03'04.0636" sul, é possível utilizar as unidades de Mercator aproximadamente iguais ao metro (Arc GIS Desktop, 2020).

A entidade `gps-projected-entity-place` permite posicionar a imagem aumentada a partir da latitude e longitude, ou seja, qualquer entidade A-Frame pode ser posicionada numa latitude ou longitude. É muito importante que todas as coordenadas geográficas utilizadas sejam de ambientes externos, onde o sinal de GPS é bom e o sistema pode se movimentar com pouca restrição. Logo, ao adicionar o local é decidido previamente qual informação será exibida quando a plataforma robótica se aproxima do local.

4.4.2. Estrutura da aplicação web

Para o desenvolvimento da aplicação web foi utilizada a plataforma Glitch como ambiente de desenvolvimento. Foram criados os arquivos css, JavaScript e HTML. Dessa forma o site pode ser aberto em tablets, smartphone e computadores, tendo como pré-requisito uma câmara integrada ao sistema e um chip de GPS. Esses componentes deverão ser incorporados no robô para uso semelhante a um computador de bordo, onde o supervisor irá visualizar informações do status da tarefa a qual o robô está submetido. Na Fig. 39 é possível observar a estrutura de arquivos criada no Glitch que possibilitou o desenvolvimento da aplicação.

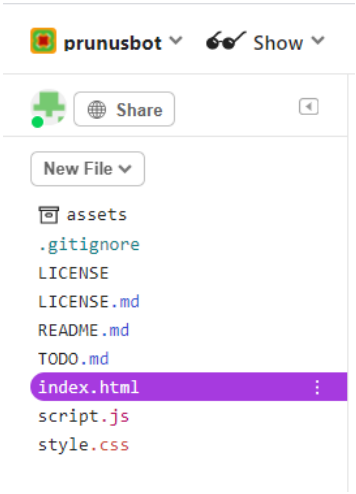

Fig. 39. Estrutura de arquivos para aplicação web no Glitch.

Nesta aplicação os arquivos `script.js` e `style.css` são utilizados apenas pelo back-end (estrutura computacional sem acesso ao utilizador) e para a configuração da câmara. Entretanto, pouco influem na geração das imagens aumentadas. Por conta disso, não serão exaustivamente aprofundados nesta dissertação. Entretanto, futuramente podem ser utilizados para tornar a aplicação ainda mais funcional com introdução automática das coordenadas com GEOJSON e outros APIs do Google Maps.

No `index.html` foram incrementadas a *tag* `<a-image>`, dentro da *tag* `<a-scene>`, em que foi alterada para gerar a imagem aumentada numa coordenada específica. Com essa *tag*, é possível alterar a forma da imagem, tamanho, rotação, entre outros parâmetros. Essas imagens são utilizadas para informar o supervisor da operação, gerando a direção na qual o robô deve prosseguir, a latitude e longitude. Na Fig. 40 a Fig. 44 é mostrado o resultado da página web com as respetivas imagens aumentadas baseadas nas coordenadas geográficas. Correspondem respetivamente ao resultado da página web com o incremento da realidade aumentada em cada ponto de destino: Ponto de destino 1 – virar à direita (Fig. 40); ponto de destino 2 – seguir em frente (Fig. 41); ponto de destino 3 – virar à esquerda (Fig. 42); ponto de destino 4 – parar (Fig. 43); e sinal de alerta para quando o robô sair da rota (Fig. 44).

Fig. 40. Resultado da página web com realidade aumentada. Ponto de destino 1 – virar à direita.

Fig. 41. Resultado da página web com realidade aumentada. Ponto de destino 2 –em frente.

Fig. 42. Resultado da página web com realidade aumentada. Ponto de destino 3 – virar à esquerda.

Fig. 43. Resultado da página web com realidade aumentada. Ponto de destino 4 – parar.

Fig. 44. Resultado da página web com realidade aumentada. Sinal de alerta para quando o robô sair da rota.

A página web solicita a utilização da câmara do sistema. Quando a permissão é concedida, a página utiliza o GPS do aparelho para quando o robô atingir as coordenadas dos *waypoints,* a página irá gerar sobreposta à imagem da câmara, as informações que irão auxiliar no bom funcionamento da plataforma robótica autónoma. Por exemplo, ao informar a direção no qual o robô deve seguir, se por um acaso, o robô tomar uma direção diferente o supervisor pode intervir e parar o robô, para que ele volte ao seu rumo correto.

A aplicação foi testada numa simulação realizada no Parque Polis na Guarda, onde foram definidos quatro pontos de destino. Utilizando um Smartphone SAMSUNG Galaxy A60, ao passar pelas coordenadas de cada ponto, o site gerou as imagens aumentadas mostrada nas Fig. 40 a Fig. 44. Na Fig. 45 é mostrado um fluxograma do circuito em que o código HTML foi testado, e na Fig. 46 **Erro! A origem da referência não foi encontrada.** é exposta uma imagem de satélite obtida no Google Earth indicando o local de cada *waypoint*.

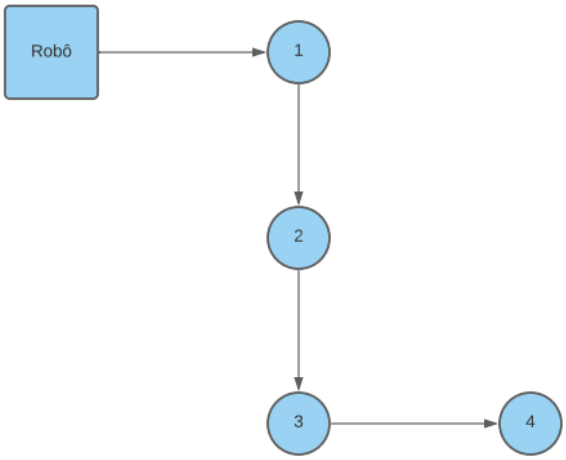

Fig. 45. Fluxograma do circuito de teste da aplicação web.

Fig. 46. Imagem de satélite do circuito de teste da aplicação Web no Google Earth.

Na Fig. 44, sempre que o robô sair da rota planeada, a imagem aumentada com o sinal de STOP será gerada para indicar ao supervisor que algo errado ocorreu na operação do robô, e que o mesmo necessita parar. Dessa forma será mais um item de segurança que possibilitará uma navegação mais fiável com a rota previamente planeada. Para a implementação desse sinal de alerta, foi necessário escolher algumas coordenadas fora da rota prevista, e se o robô estiver a passar por estes pontos a imagem aumentada é gerada.

A aplicação mostrou-se estável em todos os pontos testados e cumpriu com o que foi determinado, gerando imagem aumentada na forma de texto e de imagem nas coordenadas corretas. Entretanto, algumas vezes a página apresentava um *bug* e gerava imagens sobrepostas numa coordenada, facto que pode ser explicado por alguma flutuação na comunicação com os satélites de GPS, ocasionando perda de precisão e incompreensão das mensagens de texto.

Há ainda a possibilidade de espelhar no visor do aparelho que estiver sendo utilizado no rover terrestre numa sala de controlo com a finalidade de promover a supervisão remota do processo. Nesse sentido, existem diversas aplicações que podem cumprir com essa tarefa. Com por exemplo o ScreenCast, que possibilita o espelho do visor de Smartphones e tablets para uma TV ou computador, com acesso à mesma rede wi-fi, como pode ser visualizado na Fig. 47.

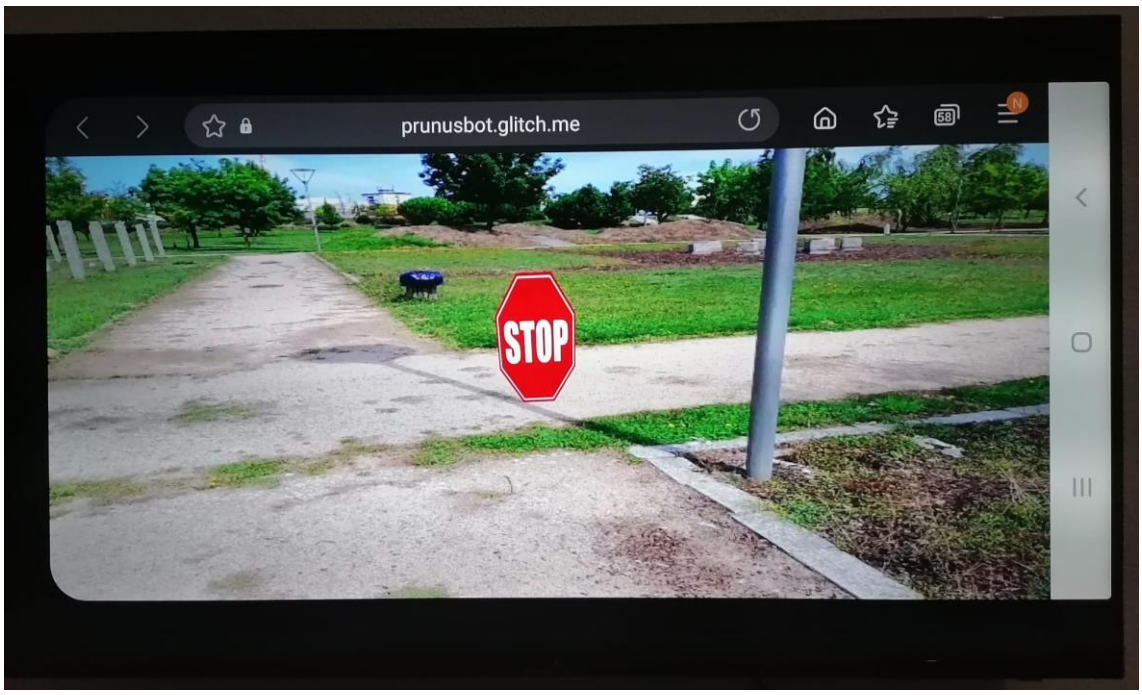

Fig. 47.Aplicação web de realidade aumentada espelhada em SmartTV.

4.4.3. Teste de campo da aplicação web

Foram realizados três testes da aplicação web num pomar de pessegueiros onde a plataforma robótica autónoma foi projetada para atuar. Este pomar localiza-se em Orjais, freguesia portuguesa pertencente ao município da Covilhã. A finalidade dos testes foi verificar se a aplicação gera a imagem correta no local correto. Desta forma, foram definidos pontos de destinos com distância de 5, 15 e 20 m. Ao percorrer com o telemóvel SAMSUNG Galaxy A60 no local, foi verificado se a imagem gerada corresponde a informação verdadeira sobre a próxima movimentação do robô. As distâncias entre um ponto e outro foram aferidas no pomar de pessegueiros com um medidor digital a laser da marca Parkside.

As coordenadas foram obtidas pela página Google Maps. Para isso, é possível marcar alguns pontos de destino no Google Maps e, logo após copiar as coordenadas ao clicar com o botão direito do rato e clicar na informação de latitude e longitude apresentada. As distâncias entre os pontos para o código foram medidas através da mesma plataforma, clicando com o botão direito e, em seguida, escolhendo a opção medir distância. Os pontos foram obtidos entre as filas 3 e 4 do pomar de pessegueiros (ver Fig. 48) e foram adicionados aos códigos de teste da plataforma. Quando o telemóvel se encontra no local, informa a aplicação da direção que o robô terá que seguir. Essa informação é gerada na forma de imagem que é aumentada sobre a imagem adquirida através da câmara do aparelho utilizado como computador de bordo do robô.

Foram desenvolvidos três códigos HTML distintos que contêm as coordenadas dos pontos com distância de 5, 10 e 15 m respetivamente. As medições foram realizadas em triplicado, ou seja, foram realizadas três medições e ao passar pelo local indicado foi avaliado se a imagem apresentada corresponde à imagem correta ou não. Foi definido que se a imagem aumentada for apresentada mais de uma vez no mesmo ponto, não apresentarem divergência de sentido entre elas, sendo considerado que a aplicação obteve êxito no local indicado.

Primeiramente foi realizada a medição para o caso com os *waypoints* distantes 5 m. Os pontos foram obtidos de forma intercalada para tentar diminuir o surgimento de imagens duplicadas. O percurso foi percorrido 3 vezes e a cada 5 m, foi verificado se a aplicação gerava a imagem correta. A Fig. 48 mostra a disposição dos pontos de destino utilizado nas três medições do teste com os pontos com 5 m de distância.

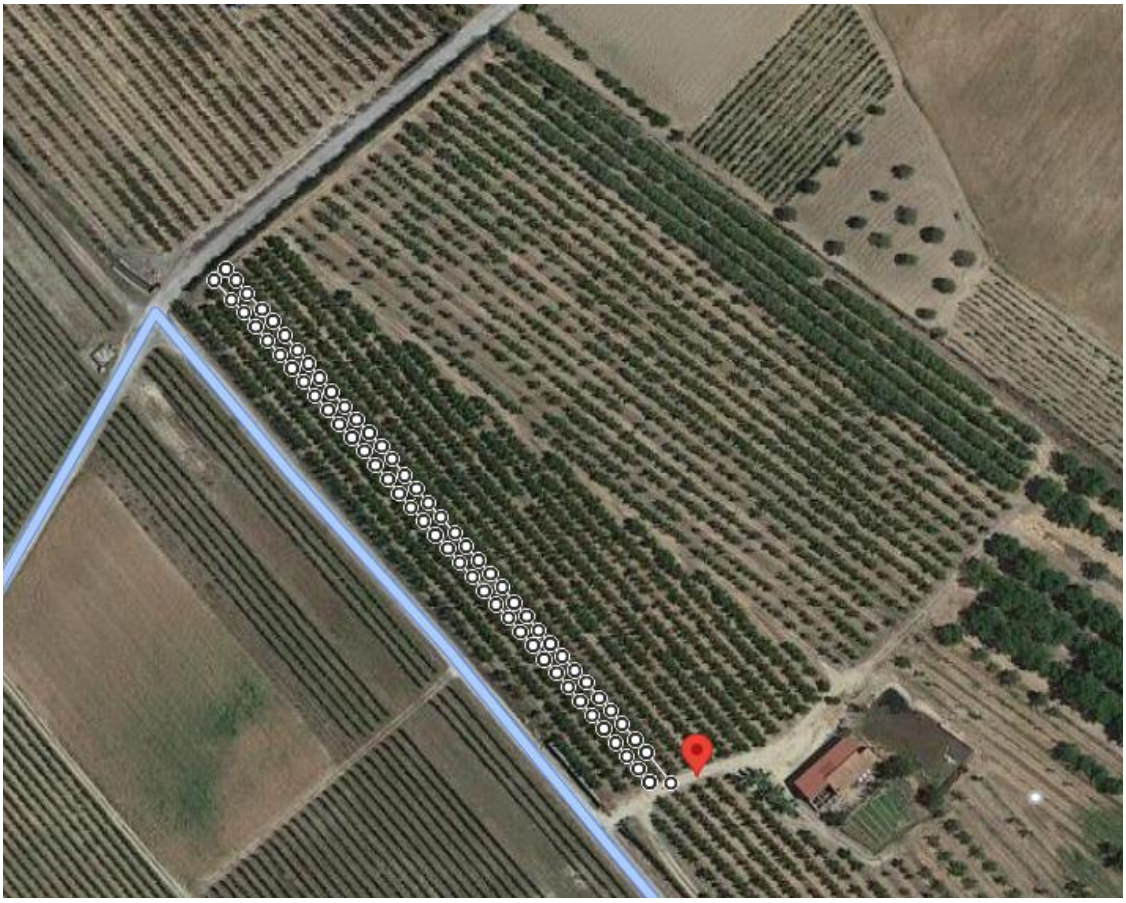

Fig. 48. Imagem de satélite dos *waypoints* utilizados com distância de 5 m.

Na Tabela 7 é exposto o resultado do teste da aplicação web de realidade aumentada para os *waypoints* distante 5 m entre si. Sempre que a aplicação gerou a imagem aumentada correta, no ponto correto, foi marcado com "OK". Entretanto, quando a aplicação não gerava nenhuma imagem ou gerava duas ou mais imagens com indicações conflituosas foi assinalado com "x".

Tabela 7. Resultado do teste de campo da aplicação web com os *waypoints* distante 5 m.

Waypoint	Latitude	Longitude	Distância (m)	Teste 1	Teste 2	Teste 3
1	40.33965753778872	-7.38026585024289	0	ok	ok	ok
2	40.33969372584911	-7.380300411330721	5	ok	ok	ok
3	40.339728275271995	-7.380336483954758	10	ok	ok	ok
4	40.33976363026514	-7.380373612431518	15	ok	ok	ok
5	40.33979719142242	-7.380410577446708	20	ok	ok	ok
6	40.339829381999216	-7.380451193238333	25	ok	ok	ok
7	40.33986753104998	-7.380485052876364	30	x	ok	ok
8	40.33990016061162	-7.38052727556545	35	ok	ok	ok
9	40.3399360193602	-7.380562836938312	40	ok	ok	ok
10	40.339966927445985	-7.38060470322929	45	ok	ok	ok
11	40.34000187621993	-7.380644639501126	50	ok	ok	ok
12	40.34003490454951	-7.380682487289 28	55	ok	ok	ok
13	40.340070875499286	-7.380717059145009	60	ok	ok	x
14	40.34010219538138	-7.380759130621731	65	ok	ok	ok
15	40.34013741099216	-7.38079656552882	70	ok	ok	ok
16	40.3401691501391	-7.380835573577234	75	ok	ok	ok
17	40.3402046996840 94	-7.380875941316188	80	ok	ok	ok
18	40.34023758144 55	-7.380913537983846	85	ok	ok	ok
19	40.34027318765206	-7.380950372616844	90	ok	ok	ok
20	40.34030625963677	-7.380994526335815	95	ok	ok	ok
21	40.34034057132091	-7.381031113709347	100	ok	ok	ok
22	40.34037316613725	-7.381071152370817	105	ok	ok	ok
23	40.34040829113311	-7.381107481563077	110	ok	ok	ok
24	40.3404435304759 16	-7.381145846227841	115	ok	ok	ok
25	40.340477093313645	-7.38118256855622	120	x	ok	ok
26	40.34050913388767	-7.381223154636067	125	ok	ok	ok
27	40.340544517394314	-7.381261304323196	130	ok	ok	ok
28	40.34057827729557	-7.381299019444897	135	ok	ok	ok
29	40.340612907361674	-7.381337380975831	140	x	ok	ok
30	40.34064555994822	-7.381375247020448	145	ok	ok	x
31	40.340679561125796	-7.381417249308805	150	ok	ok	ok
32	40.34071424263445	-7.381453794677718	155	ok	x	ok
33	40.340748623354 86	-7.381493021740581	160	ok	x	ok
34	40.340780510402936	-7.381531456764051	165	ok	x	ok
35	40.34081608857092	-7.381570805802528	170	ok	ok	x
36	40.340848486686724	-7.38161058192852	175	ok	ok	x
37	40.34089888669842	-7.381664012566414	182	ok	x	ok
38	40.34091933066487	-7.38163249661028	186	ok	ok	ok
39	40.34089343991293	-7.381592977471207	193	x	ok	x
40	40.34086328504868	-7.381557438202421	198	x	x	x
41	40.340824441474844	-7.38150915844105	203	x	ok	x
42	40.34079499202108	-7.381477038537739	208	ok	ok	ok
43	40.34076060943795	-7.381437567989424	213	ok	ok	ok
44	40.34072463699968	-7.381401416387059	218	ok	ok	ok
45	40.34067158566168	-7.381340295940148	223	ok	ok	ok
46	40.34063639307608	-7.381299114249637	228	ok	ok	ok
47	40.34062700071348	7.381289621267384	233	ok	ok	ok
48	40.340587327530585	-7.381242935467512	238	ok	ok	x
49	40.340555875822005	-7.381185723868563	243	ok	ok	x
50	40.34052367639858	-7.381166647764429	248	x	ok	ok
51	40.34048714907672	-7.38112691647554	253	ok	x	x
52	40.34045910573369	-7.381093087164952	258	ok	ok	ok
53	40.34042079688891	-7.38105325839385	263	ok	ok	ok
54	40.34038203224056	-7.381018601998095	268	ok	ok	ok
55	40.34034995205814	-7.380977746986486	273	ok	ok	ok
56	40.34031652262102	-7.380936280860677	278	x	ok	ok
57	40.34027701508381	-7.380893219884426	283	ok	ok	ok
58	40.34024625110254	-7.380855791560783	288	ok	ok	ok
59	40.340210491068866	-7.380816225106745	293	x	ok	ok
60	40.34017535714954	-7.380778878851063	298	ok	ok	ok
61	40.34014499907118	-7.380741894991249	303	ok	ok	x
62	40.34010872590754	-7.380702490448946	308	ok	x	x
63	40.34007692217463	-7.380658868460447	313	ok	ok	x
64	40.34003748883386	-7.380624322857009	318	ok	ok	ok
65	40.34000523001805	-7.380587962399465	323	ok	ok	ok
66	40.33997362384372	-7.380546496272612	328	x	ok	ok
67	40.339942017654586	-7.380508219847827	333	ok	x	ok
68	40.33990651058896	-7.380468437130434	338	x	ok	x
69	40.33987178908061	-7.380424264270636	343	ok	ok	ok
70	40.33984232839285	-7.380396656233261	348	ok	ok	ok
71	40.33980865901967	-7.38035800498093	353	ok	ok	ok
72	40.339774989629674	-7.380313832121137	358	ok	x	ok
73	40.339741320222856	-7.380271039663206	363	ok	x	ok
74	40.33966614592976	-7.380194084909855	368	ok	ok	x

A percentagem de acerto dos três testes é apresentada na Fig. 49. Para a distância entre os *waypoints* de 5 m, foi obtido no primeiro teste uma percentagem de acerto de 85,2%, de 86,5% no segundo teste e de 79,7% no terceiro teste. Esta variação ocorre devido à sensibilidade do sensor do aparelho utilizado nos testes, pois os erros de obtenção da posição atual e da posição do *waypoint* são variáveis e ambos podem chegar a mais de 10 m de raio. Assim, o GPS pode gerar várias imagens ao mesmo tempo, sempre que o sensor estiver dentro do raio do erro de medição de dois ou mais pontos. Dessa forma, como as distâncias entre os pontos são de 5 m, pode ocorrer a geração de várias imagens num local, inclusive imagens aumentadas com informações conflituosas, o que explica o maior número de erros. Assim, com a distância de 5 m, a aplicação web apresentou uma percentagem média de acertos de 83,8%, o que configura o menor percentual médio de acertos aferido nesse trabalho.

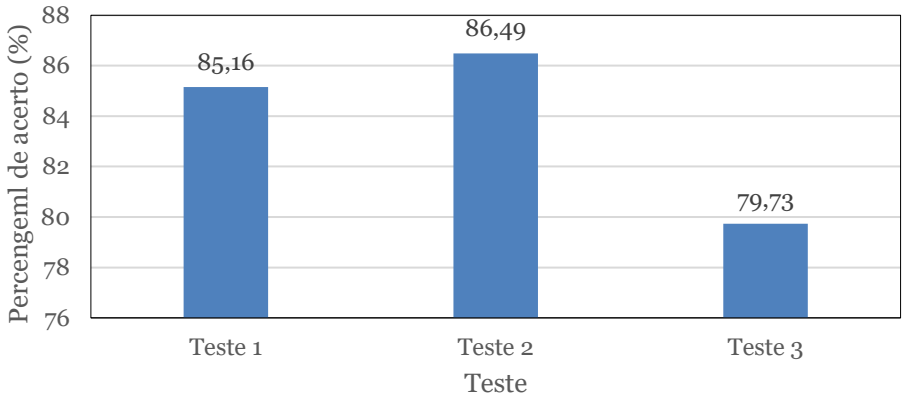

Fig. 49. Percentagem de acerto da aplicação web testada com os *waypoints* com distância de 5 m.

De maneira análoga à anterior, foi realizada a medição para o caso com os *waypoints* distantes 10 m. Os pontos também foram adquiridos de forma intercalada para tentar minimizar o surgimento de imagens múltiplas. O percurso foi realizado 3 vezes e a cada 10 metros verificou se a aplicação gerava a imagem correta. A Fig. 50 mostra a disposição dos pontos de destino utilizado nas três medições do teste com os pontos com 10 metros de distância.

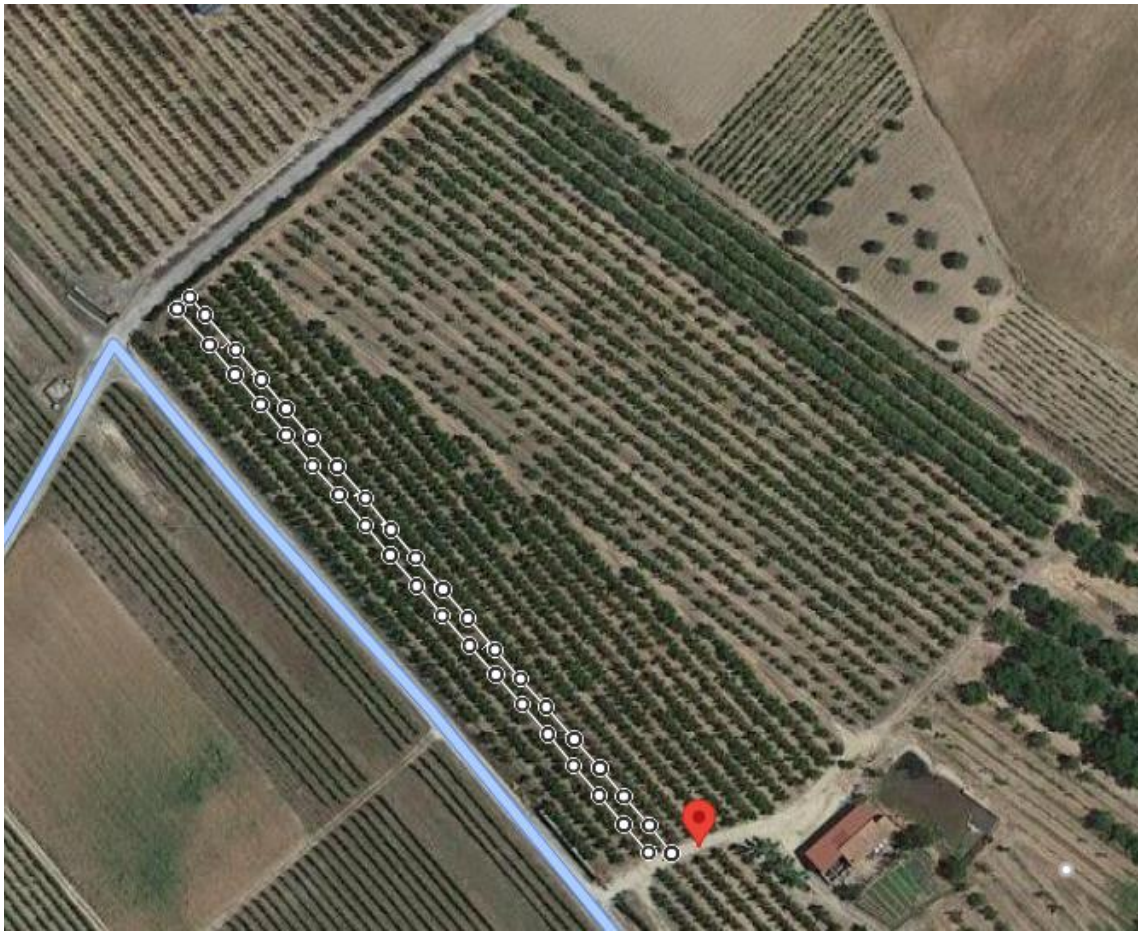

Fig. 50. Imagem de satélite dos *waypoints* utilizados com distância de 10 m.

Do mesmo modo, na Tabela 8 é exposto o resultado do teste da aplicação web de realidade aumentada para os *waypoints* distante 10 m entre si. Foi adotado o mesmo procedimento e sempre que a aplicação concebeu a imagem aumentada correta, no ponto correto, foi indicado "ok". Contudo, quando não era gerada nenhuma imagem ou gerava duas ou mais imagens aumentadas com indicações conflituosas foi assinalado com "x".

Tabela 8. Resultado do teste de campo da aplicação web com os *waypoints* distante 10 m.

Waypoint	Latitude	Longitude	Distância (m)	Teste 1	Teste 2	Teste 3
1	40.33965753778872	-7.38026585024289	0	ok	ok	ok
2	40.339728275271995	-7.380336483954758	10	ok	ok	ok
3	40.33979719142242	-7.380410577446708	20	ok	ok	ok
4	40.33986753104998	-7.38048505287636	30	ok	ok	ok
5	40.3399360193602	-7.380562836938312	40	ok	ok	ok
6	40.34000187621993	-7.380644639501126	50	ok	ok	ok
7	40.3400708754992 86	-7.380717059145009	60	ok	ok	ok
8	40.34013741099216	-7.38079656552882	70	ok	ok	ok
9	40.340204699684094	-7.380875941316188	80	ok	ok	ok
10	40.34027318765206	-7.380950372616844	90	ok	ok	ok
11	40.34034057132091	-7.381031113709347	100	ok	ok	x
12	40.34040829113311	-7.381107481563077	110	ok	ok	ok
13	40.340477093313645	-7.38118256855622	120	ok	ok	ok
14	40.340544517394314	-7.381261304323196	130	ok	x	ok
15	40.340612907361674	-7.381337380975831	140	ok	ok	ok
16	40.340679561125796	-7.381417249308805	150	ok	ok	ok
17	40.340748623354 86	-7.381493021740581	160	ok	ok	ok
18	40.34081608857092	-7.381570805802528	170	x	ok	x
19	40.340898886698 42	-7.381664012566414	182	ok	ok	ok
20	40.34091933066487	-7.38163249661028	186	ok	ok	ok
21	40.34086328504868	-7.381557438202421	198	x	ok	ok
22	40.34079499202108	-7.381477038537739	208	ok	x	ok
23	40.34072463699968	-7.381401416387059	218	ok	ok	ok
24	40.34063639307608	-7.381299114249637	228	ok	ok	x
25	40.340587327530585	-7.381242935467512	238	ok	ok	ok
26	40.34045910573369	-7.38109308716495	258	ok	x	ok
27	40.34052367639858	-7.381166647764429	248	ok	ok	ok
28	40.34038203224056	-7.381018601998095	268	ok	x	ok
29	40.34031652262102	-7.38093628086067	278	x	ok	ok
30	40.34024625110254	-7.380855791560783	288	ok	ok	ok
31	40.34017535714954	-7.380778878851063	298	ok	ok	ok
32	40.34010872590754	-7.38070249044894	308	ok	ok	ok
33	40.340037488833 86	-7.38062432285700	318	ok	ok	ok
34	40.33997362384372	-7.380546496272612	328	ok	ok	ok
35	40.33990651058896	-7.380468437130434	338	ok	ok	x
36	40.33984232839285	-7.380396656233261	348	ok	ok	ok
37	40.339774989629674	-7.380313832121137	358	ok	x	ok
38	40.33966614592976	-7.38019408490985	368	ok	ok	ok

Para a distância entre os *waypoints* de 10 m, foi obtido no primeiro teste uma percentagem de acerto de 92,1%, de 86,8% no segundo teste e de 89,5% no terceiro teste. Com a distância de 10 m, a aplicação apresenta uma percentagem média de acertos de 89,5%, o que configura a percentagem de acerto média aferida maior que no caso anterior. De maneira análoga a anterior, com os *waypoints* distante 10 m, ainda sim o erro de aquisição da posição dos pontos de destino pode ser maior que a distância entre eles, podendo serem geradas imagens aumentadas conflituosas. Entretanto, e apesar de estas serem em menor número que no caso anterior, pois a distância é maior e o erro é variável, num número menor de casos ainda ocorre o aparecimento de imagens conflituosas. Principalmente na aproximação a um ponto de mudança de sentido. A Fig. 51 mostra a percentagem de acerto dos três testes com distância de 10 m entre os pontos de destinos.

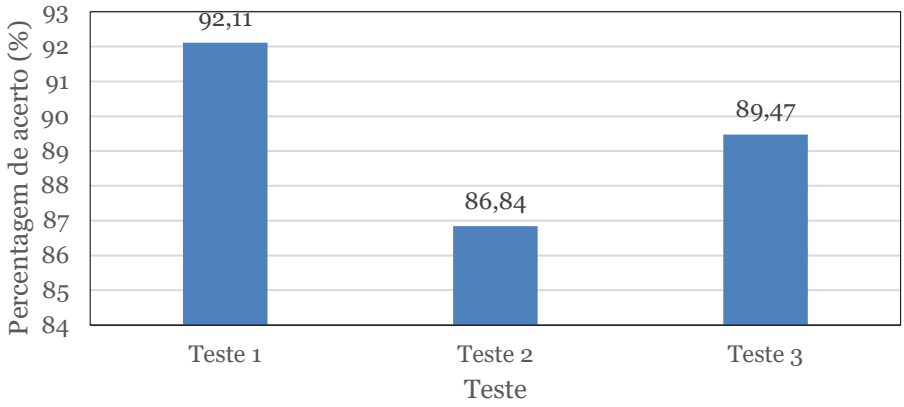

Fig. 51. Percentagem de acerto da aplicação web testada com os *waypoints* com distância de 10 m.

E finalmente, foram realizadas as medições com os *waypoints* distantes 15 m entre si. A Fig. 52 mostra a disposição dos pontos de destino utilizado nas três medições do teste com os pontos com 15 m de distância. Na Tabela 9 são expostos os resultados do teste da aplicação web de realidade aumentada para estes pontos.

Fig. 52. Imagem de satélite dos *waypoints* utilizados com distância de 15 m.

Tabela 9. Resultado do teste de campo da aplicação web com os *waypoints* distante 15 m.

Waypoint	Latitude	Longitude	Distância (m)	Teste 1	Teste 2	Teste 3
1	40.33965753778872	-7.3802658502428 97	0	ok	ok	ok
2	40.33976363026514	-7.3803736124315185	15	ok	ok	ok
3	40.33986753104998	-7.3804850528763 64	30	ok	ok	ok
4	40.339966927445985	-7.3806047032292 94	45	ok	ok	ok
5	40.340070875499286	-7.380717059145009	60	ok	ok	ok
6	40.3401691501391	-7.380835573577234	75	ok	ok	ok
7	40.34027318765206	-7.380950372616844	90	ok	ok	ok
8	40.34037316613725	-7.381071152370817	105	ok	ok	ok
9	40.340477093313645	-7.38118256855622	120	ok	ok	ok
10	40.34057827729557	-7.3812990194448975	135	ok	ok	ok
11	40.340679561125796	-7.381417249308805	150	ok	x	ok
12	40.3407805104029 36	-7.381531456764051	165	x	ok	ok
13	40.34089888669842	-7.381664012566414	182	ok	ok	ok
14	40.34091933066487	-7.38163249661028	186	ok	ok	ok
15	40.340824441474844	-7.38150915844105	203	x	ok	ok
16	40.34072463699968	-7.381401416387059	218	ok	ok	x
17	40.34062700071348	7.381289621267384	233	ok	ok	ok
18	40.34052367639858	-7.381166647764429	248	ok	ok	ok
19	40.34042079688891	-7.38105325839385	263	ok	ok	ok
20	40.34031652262102	-7.380936280860677	278	ok	x	ok
21	40.340210491068866	-7.380816225106745	293	ok	ok	ok
22	40.34010872590754	-7.380702490448946	308	ok	ok	ok
23	40.34000523001805	-7.380587962399465	323	ok	ok	ok
24	40.33990651058896	-7.3804684371304345	338	ok	ok	ok
25	40.33980865901967	-7.380358004980935	353	ok	ok	ok
26	40.33966614592976	-7.3801940849098555	368	ok	ok	ok

No caso da distância entre os *waypoints* de 15 m, foram obtidos no primeiro e no segundo teste uma percentagem de acerto de 92,3% e de 96,2% no terceiro teste. Com a distância de 15 m a aplicação apresentou uma percentagem média de acertos de 93,6%, o que se configura como a percentagem de acerto média mais elevada aferida entre todos os casos. Desse modo, a menor percentagem de erro é justificada pelo aumento da distância entre os *waypoints*. Todavia, se a distância entre os *waypoints* aumentar em demasiado, acarretará na diminuição de pontos de destinos e, consequentemente, será gerado um número menor de imagens aumentadas o que tornará a aplicação subutilizada. A Fig. 53 mostra a percentagem de acerto dos três testes com distância de 15 m entre os pontos de destinos.

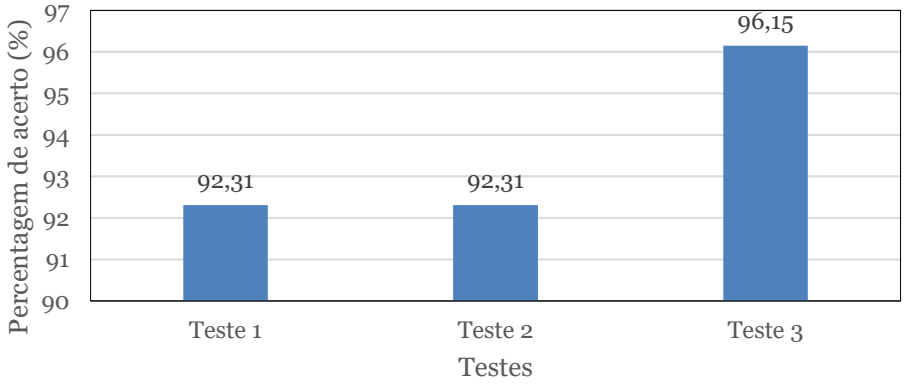

Fig. 53. Gráfico do Percentual de acerto da aplicação web testada com os *waypoints* com distância de 15 m.

Logo, a melhor disposição para a utilização da aplicação web de realidade aumentada é quando os pontos de destinos têm, entre si, uma distância de 15 m. Esta distância é superior à maior parte dos erros de aquisição da posição dos pontos de destinos, o que faz a aplicação web apresente um menor número de erros de geração de imagens aumentadas conflituosas. Adicionalmente, esta distância proporciona uma quantidade adequada de pontos de destino, que leva a aplicação web a apresentar imagens aumentadas em quantidade suficiente para alertar o supervisor da operação. Desta forma, a aplicação mostrou-se estável gerando o mínimo de imagens duplicadas, com informações conflituantes ou a ausência de imagens nos pontos desejados. Nas Fig. 54 a Fig. 57 são mostradas algumas imagens aumentadas geradas durante os testes nos três casos estudados, que foram geradas no local correto e com a informação correta.

Fig. 54. Resultado da página web com o incremento da realidade aumentada com 10 m de distância entre os *waypoints*: Virar à direita.

Fig. 55. Resultado da página web com o incremento da realidade aumentada com 5 m de distância entre os *waypoints*: virar à esquerda.

Fig. 56. Resultado da página web com o incremento da realidade aumentada com 15 m de distância entre os *waypoints*: seguir em frente.

Fig. 57. Resultado da página web com o incremento da realidade aumentada com 10 m de distância entre os *waypoints*: parar.

Entretanto, de notar alguns erros que ocorreram com maior frequência, como seja a geração de imagens duplicadas, sendo apresentado um exemplo na Fig. 58. Neste caso, o erro não interfere no uso da aplicação, pois ainda indica o caminho correto para o movimento do robô. Por isso, ao detetar esse erro, ainda assim foi considerado que a aplicação obteve êxito. No entanto, quando a aplicação mostrava imagens conflituosas, como exposto na Fig. 59, foi considerado como falha da aplicação.

Fig. 58. Imagem aumentada com informação correta e duplicada.

Fig. 59. Erro de imagem aumentada com informação conflituosa.

4.5. Nota conclusiva

Conclui-se pelo conteúdo exposto ao longo do capítulo 4, que o Rover Robótico para Aplicações Agrícolas (R2A2) pode mover-se dentro de um circuito previamente determinado. Para isso, foi implementado um algoritmo em C/C++ baseado em GPS para que o R2A2 pudesse movimentar de maneira autónoma. Foram utilizadas as equações de Haversine para cálculo da distância entre duas coordenadas e foi obtido o azimute do rumo entre as duas coordenadas. Logo após a determinação do rumo, o seu valor foi comparado com o rumo obtido pela bússola para que fosse possível tomar a decisão da direção a seguir.

Foi detalhado também o processo de programação da aplicação web baseada em realidade aumentada para auxílio do supervisor da operação da plataforma robótica autónoma. Dessa forma, foi implementado um código em HTML, juntamente com as bibliotecas AR.JS e o *Framework* A-FRAME, para construir um ambiente de realidade aumentada que pode ser utilizado em qualquer smartphone, tablet ou portátil com acesso a internet e tenha um GPS integrado. O aparelho tem que estar localizado no robô, sendo utilizado de maneira semelhante a um computador de bordo. Foi realizado um teste de campo da aplicação web de realidade aumentada e foi obtido uma percentagem média de acertos de 93,6%, o que revela a precisão da solução tecnológica desenvolvida.

5. Conclusões

5.1. Conclusões gerais

O principal objetivo deste trabalho consiste no desenvolvimento de um algoritmo de navegação autónoma baseado em GPS para o Rover Robótico para Aplicações Agrícolas (R2A2) multitarefa. O segundo objetivo consiste na implementação de uma aplicação web de realidade aumentada para apoio a supervisão das operações do robô. Com isso, foi possível que a plataforma robótica realizasse um circuito de forma autónoma utilizando as coordenadas geográficas planeadas pelo supervisor da operação, que também poderá visualizar de forma segura a operação num aparelho com acesso à internet instalado no robô.

Foi implementado um algoritmo que calcula a distância entre a localização do robô e o próximo ponto de destino, através das coordenadas geográficas. Posteriormente, o azimute da direção entre os dois pontos é calculado e comparado com o azimute da direção atual do robô, aferido pela bússola eletrónica. Após a comparação do rumo final, é tomada a decisão de seguir em frente, virar à direita, virar à esquerda ou parar. Em cada local, as coordenadas são atualizadas, e todos os parâmetros são recalculados continuamente. Devido à atualização contínua do local, o robô move-se na direção correta e para quando atinge o último *waypoint*. O algoritmo foi desenvolvido para que pudessem ser fornecidas várias coordenadas para que o robô realize o circuito desejado e atinja o ponto final.

Para isso, foi necessário desenvolver um circuito utilizando como microcontrolador principal um Arduino Mega2560, juntamente com um módulo GPS NEO-6M e um módulo Bússola HMC5883L. Esses três componentes são imprescindíveis para que o robô opere de forma satisfatória de acordo com a temática do projeto. Foi adicionado também um módulo WIFI ESP8266 ESP-01 para dar a possibilidade de enviar as coordenadas de forma remota para o Arduino e um módulo sensor ultrassónico HY-SRF05 para evitar a colisão com obstáculos nos pomares.

A aplicação Web baseada em realidade aumentada mostrou-se bastante útil ao projeto. De maneira geral, a imagem aumentada é apresentada de forma nítida, estável e eficaz. Posteriormente, o ecrã do aparelho utilizado como computador de bordo do rover autónomo pode ser espelhado para uma sala de controlo onde o supervisor pode verificar o processo de forma remota e ver as informações apresentadas na forma de imagem aumentada. Neste caso foi utilizada a aplicação gratuita ScreenCast.

No teste realizado no campo de operação para a qual a plataforma robótica foi projetada atuar, os resultados foram bastantes relevantes. A aplicação web obteve a melhor percentagem média de acerto, igual a 93,6%, no teste em que os *waypoints* estão distanciados de 15 m. A aplicação também apresentou bons resultados nos testes de 5 m e de 10 m, com percentagens médias de acertos igual a 89,5% e 79,7%, respetivamente. As imagens aumentadas eram nítidas e de fácil compreensão, o que faz da aplicação web de realidade aumentada uma mais-valia na supervisão da operação do robô R2A2.

5.1.1. Adversidades encontradas

Na realização desse projeto foram encontradas diversas dificuldades desde a programação do algoritmo, à utilização correta dos componentes eletrónicos, devido à complexidade do projeto e pela forte componente prática. Ao longo da implementação do algoritmo de movimentação autónoma foram surgindo alguns problemas com as bibliotecas utilizadas, em que ocasionava diversos erros de compilação, até que fosse encontrado a biblioteca ideal para cada componente do circuito eletrónico.

A sensibilidade do sensor GPS por diversas vezes prejudicava a operação do circuito robótico, visto que, diversas vezes o sensor não recebia a sua localização com precisão. Trata-se de um grave problema pois o campo de trabalho, entre as filas de culturas é bastante restrito, não sendo aceitável que o robô, em operação, tenha um erro de medição maior que a distância entre duas filas consecutivas de arvores. Contudo, essa perda de sensibilidade justifica a futura troca do sensor por um sensor GPS de maior exatidão e precisão.

Como a aplicação web de realidade aumentada também é baseada em GPS, ocorreram problemas precisão similares, o que levou à geração de imagens virtuais alguns metros antes ou depois do ponto de destino especificado. Entretanto, na maioria das vezes, o resultado ocorreu no local escolhido. Outro erro que se verificou foi a sobreposição de imagens aumentadas, ou seja, numa determinada localização, a aplicação gerava a imagem aumentada do local, entretanto, sobreposta com as imagens virtuais de outros locais anteriores ou posteriores. Com isso o texto, impresso no ecrã do aparelho utilizado, tornava-se ilegível, o que pode tornar a aplicação, por vezes, pouco funcional.

5.2. Sugestões de trabalhos futuros

Ainda assim, todo o sistema pode ser otimizado por desenvolvimento e aplicação dos seguintes trabalhos:

Implementar um controlador PID completo para navegação. Esta estratégia de controlo tornaria a navegação mais suave e eficaz.

Integrar o algoritmo de navegação autónoma baseado em GPS com um algoritmo de navegação baseada em visão, para que o sistema baseado em GPS se torne redundante e secundário, visando a melhoria da fiabilidade do sistema de locomoção do rover.

Implementar filtros de Kalman ou filtro de média móvel para melhorar as leituras GPS. Este resultado tornaria o robô mais fiável e evitaria que o mesmo vagasse por fora do percurso programado.

Unificar a aplicação web integrando um sistema de envio das coordenadas geográficas, visualização da localização utilizando o API do Google Maps e a visualização das imagens em realidade aumentada.

Incluir o código em JavaScript para tornar a página mais responsiva e eficiente. Pode-se também utilizar JSON para realizar os envios dos dados das coordenadas geográficas com a finalidade de melhorar o protocolo de comunicação série.

Utilizar banco de dados SQL ou MongoDB para receber as coordenadas e programar um código em JavaScript para aceder a essas informações ao invés de usar as coordenadas código HTML. Essa modificação teria a finalidade de tornar o código mais simples de ser atualizado.

6. Referências Bibliográficas

A-FRAME, 2021. *Introduction*. [Online]
Available at: https://aframe.io/docs/1.2.0/introduction/

ALATISE, M. B. & HANCKE, G. P., 2020. A Review on Challenges of Autonomous Mobile Robot and Sensor Fusion Methods. *IEEE Access,* Volume 8, pp. 39830-39846.

Almeida, R. C. & Rodrigues, E. L. L., 2016. *Proposta de Sistema de Segurança Interativo Baseado emConceitos de Internet of Things,* São Carlos: Escola de Engenharia de São Carlos.

Anon., 2021. *Arduino*. [Online]
Available at: https://store.arduino.cc/arduino-mega-2560-rev3

Anon., 2021. *Stackexchange*. [Online]
Available at: https://dsp.stackexchange.com/questions/38020/how-can-i-isolate-a-hough-circle-that-i-perceive-to-be-the-best-fit-but-is-rank

AR.js, 2021. *AR.JS Documentation*. [Online]
Available at: https://ar-js-org.github.io/AR.js-Docs/

Arc GIS Desktop, 2020. *ArcMAP*. [Online]
Available at: https://desktop.arcgis.com/en/arcmap/latest/map/projections/mercator.htm#GUID-D44CEA15-C448-449E-9766-9717D589330E
[Acedido em 14 Setembro 2021].

Arduino, 2018. *Getting Started with Arduino MEGA2560*. [Online]
Available at: https://www.arduino.cc/en/Guide/ArduinoMega2560
[Acedido em 23 setembro 2021].

Arduino, 2021. *Arduino IDE*. [Online]
Available at: https://www.arduino.cc/en/software

ASHRAF, M. A., TAKEDA, J.-i. & TORISU, R., 2010. Neural Network Based Steering Controller for Vehicle Navigation on Sloping Land.. *Engineering in Agriculture, Environment and food EAEF 3(3),* pp. 100-104.

Åstrand, B. & Baerveldt, A., 2005. vision based row-following system for agricultural field machinery. *Mechatronics,* Volume 15, pp. 251-269.

Bak, T. & Jakobsen, H., 2004. Detection, Agricultural Robotic Platform with Four Wheel Steering for Weed. *Automation and Emerging Technologies,* Volume 87(2), pp. 125-136.

Benson, E., Reid, J. & Zhang, Q., 2003. Machine Vision-based Guidance System for Agricultural Grain Harvestersusing Cut-edge Detection. *Biosystems Engineering,* Volume 86(4), p. 389–398.

Braga, A. P. C. & Ludemir, T., 2007. *Redes Neurais Artificiais,Teoria e Aplicações.*. Rio de Janeiro : LTC - Livros Técnicos e Cietíficos.

Carpignoli, N., 2021. *Augmented Reality on the Web with Model Viewer*. [Online] Available at: https://nicolcarpignoli.medium.com/

CSS ZEN GARDEN, 2021. *CSS ZEN GARDEN*. [Online] Available at: http://www.csszengarden.com/217/

DataSheet4U, 2008. *SRF05 - Ultra-Sonic Ranger*. [Online] Available at: https://www.madnesselectronics.com/datasheets/sensores/srf05.pdf [Acedido em 21 setembro 2021].

Drongelen, W. V., 2018. *Signal Processing for Neuroscientists*. 2ed ed. New York, NY: ELSEVIER.

Duckett, T., Pearson, S., Blackmore, S. & Grieve, B., 2018. Agricultural Robotics: The Future of Robotic Agriculture. *UK-RAS White papers,* pp. 2398-4414.

Durrant-Whyte, H. & Henderson, T., 2016. *Multisensor Data Fusion. In: Siciliano B., Khatib O. (eds)*. s.l.:Springer Handbook of Robotics. Springer Handbooks. Springer, Cham..

ElectronicWings, 2021. *Magnetometer HMC5883L Interfacing With Arduino UNO*. [Online] Available at: https://www.electronicwings.com/arduino/magnetometer-hmc5883l-interfacing-with-arduino-uno

ESP32net, 2017. *The internet of things with ESP32*. [Online] Available at: http://esp32.net/ [Acedido em 21 setembro 2021].

ESPRESSIF SMART, 2013. ESPRESSIF SMART CONNECTIVITY PLATFORM: ESP8266. *ESP8266 802.11bgn Smart Device,* 12 10.

FERENTINOS, K., ARVANITIS, K. & SIGRIMIS, N., 2002. Heuristic optimization methods for motion planning of autonomous agricultural vehicles. *Journal of Global Optimization,* Volume 23, pp. 155-170.

Fernández, J. S., Gil, J. G. & Cirilo, L. d. P., 2010. Design and Implementation of a GPS Guidance System for Agricultural Tractors Using Augmented Reality Technology.. *Sensors,* Issue 10, pp. 10435-10447.

Glitch, 2021. *Glitch About us.* [Online] Available at: https://glitch.com/about

Graves, A. et al., 2015. The total costs of soil degradation in England and Wales. *Ecological Economics,,* Volume 119, p. 399–413.

Guzueva, E. R. et al., 2020. The impact of automation of agriculture on the digital economy. *IOP Conference Series: Earth and Environmental Science,* Issue 022047, p. 421.

Hague, T., Marchant, J. & Tillett, N., 2000. Ground based sensing systems for autonomousagricultural vehicles. *Computers and Electronics in Agriculture,* pp. 11-28.

Hamner, B., Singh, S. & Bergerman, M., 2010. Improving orchard efficiency with autonomous utility vehicles. *ASABE Annual International Meeting,,* Issue 1009415.

Harrell, R. C., Adsit, P. D. & Pool, T. A. H. R., 1990. THE FLORIDA ROBOTIC GROVE-LAB. *ASAE Transactions,* 2(3), p. 391—399.

Hassall, J., 2010. *Future trends in precision agriculture: a look into the Farming Scholars,* s.l.: s.n.

Have, H. et al., 2005. Autonomous Weeder for Christmas Trees – Basic Development and Tests. *Pesticides Research,* Volume 95.

Haverbeke, M., 2018. *Eloquent JavaScript*. 3rd edition ed. s.l.:Marijn Haverbeke.

Hoang, T. T. et al., 2012. Multi-Sensor Perceptual System for Mobile Robot and Sensor Fusion-based Localization. *International Conference on Control, Automation and Information Sciences (ICCAIS),* pp. 259-264.

Honeywell, 2013. 3-Axis Digital Compass ICHMC5883L. *Advanced Information.*

Ikasari, D., Widiastuti & Andika, R. h., 2021. *Determine the Shortest Path Problem Using Haversine Algorithm, A Case Study of SMA Zoning in Depok,* s.l.: IEEE Xplore.

Joshi, M. M. & Zaveri, M. A., 2011. Reactive Navigation of Autonomous Mobile Robot Using NeuroFuzzy System. *International Journal of Robotics and Automation (IJRA),* 2(3), pp. 128-145.

Kanagasingham, S., Ekpanyapong, M. & Chaihan, R., 2019. Integrating machine vision-based row guidance with GPS and compass-based routing to achieve autonomous navigation for a rice field weeding robot. *Precision Agriculture*, p. 831–855.

Khan, A. A. & Rapal, N., 2006. Fuzzy PID Controller: Design, Tuning and Comparison with Conventional PID Controller. *IEEE International Conference on Engineering of Intelligent Systems,* pp. 1-6.

Kim, D. et al., 2021. Review of machine learning methods in soft robotics.. *PLoS ONE* .

Lerke, O. & Schwieger, V., 2021. Analysis of a kinematic real-time robotic total station network for robot control.. *Journal of Applied Geodesy,* 15(3), pp. 169-188.

Li, M., Imou, K., Wakabayashi, K. & Yokoyama, S., 2009. Review of research on agricultural vehicle autonomous guidance. *Internacional Journal of Agricultural and Biological Engineering,* 2(3), pp. 1-16.

Litayem, N., Dhupia, B. & Rubab, S., 2015. Review of Cross-Platforms for Mobile Learning Application Development. *International Journal of Advanced Computer Science and Applications,* 6(1), pp. 31-39.

Li, X. & Qiu, Q., 2021. *Autonomous Navigation for Orchard Mobile Robots: A Rough Review.* China, s.n., pp. 552-557.

Lulio, L. C., T. M. L. & Porto, A. J. V., 2012. Cognitive-merged Statistical Pattern Recognition Method for Image Processing in Mobile Robot Navigation. *Brazilian Robotics Symposium and Latin American Robotics Symposium IEEE,* pp. 279-283.

Maier, J., 2017. *Made smarter review. Department for Business, Energy and Industrial Strategy..* [Online]
Available at: http://hdl.voced.edu.au/10707/444094.

Makhataeva, Z. & Varol, H. A., 2020. Augmented Reality for Robotics: A Review. *Robotcs,* Volume 9(2), 21.

Mekonnen, M. M. & Hoekstra, A. Y., 2016. Four billion people facing severe water scarcity. *Science Advances,* 2(2).

Microsoft, 2021. *Support Microsoft.* [Online] Available at: https://support.microsoft.com/pt-pt/topic/c%C3%B3digo-de-placa-de-carregamento-e-arduino-ide-a9723765-1314-49e0-a69b-bb5c3e1f628d

MONK, S., 2013. *30 Arduino Projects for the Evil Genius.* second edition ed. EUROPE: MCGRAW-HILL EDUCATION.

Mousazadeh, H., 2013. A technical review on navigation systems of agricultural. *Journal of Terramechanics,* pp. 211-232.

Movable Type Scripts, 2021. *Calculate distance, bearing and more between Latitude/Longitude points.* [Online] Available at: https://www.movable-type.co.uk/scripts/latlong.html [Acedido em 26 Julho 2021].

Nagasaka, Y. et al., 2009. An Autonomous Rice Transplanter Guided by Global Positioning System and Inertial Measurement Unit. *Journal of Field,* Volume 26, p. 537–548.

Naik, N. S., Shete, V. V. & Danve, S. R., 2016. Precision Agriculture Robot for Seeding Function. *International Conference on Inventive Computation Technologies (ICICT),* pp. 1-3.

Ortiz, J. M. & Olivares, M., 2006. A Vision Based Navigation System for an Agricultural Field Robot. *IEEE 3rd Latin American Robotics Symposium,* pp. 106-114.

Paradkar, A. D. & Sciortino, A., 2016. *GPS GUIDED AUTONOMOUS ROBOT,* Long Beach: California State University.

Park, J.-S.et al., 2018. Smart Contract-Based Review System for an IoT Data Marketplace. *Sensors,* 18(10), pp. 1424-8220.

RAISCH, S., 2021. ARTIFICIAL INTELLIGENCE AND MANAGEMENT:THE AUTOMATION–AUGMENTATION PARADOX. *Academy of Management Review,* 46(1), pp. 192-210.

REES52, 2021. *HY-SRF05 Precision Ultrasonic Sensor*. s.l.:s.n.

Reid, J. F., Zhang, Q., Noguchi, N. & Dickson, M., 2000. Agricultural automatic guidance research in North America. *Computers and Electronics in Agriculture,* Volume 25, pp. 155-167.

Robbins, J. N., 2012. *Learning Web Design: A Beginner's Guide to HTML, CSS, JavaScript, and Web Graphics.* 4ª ed. s.l.:Simon St. Laurent.

Ruane, J. & Sonnino, A., 2011. Agricultural biotechnologies in developing countries and their possible contribution to food security. *Journal of Biotechnology,* 156(4), pp. 356-363.

Ryerson, A. & Zhang, Q., 2007. Vehicle Path Planning for Complete Field Coverage Using Genetic Algorithms. *Agricultural Engineering International: the CIGR Ejournal.,* Volume IX.

Satow, T. et al., 2004. Development of laser crop row sensor for automatic guidance system of implements. *Proceedings of the Automation Technology for Off-road Equipment (ATOE),* pp. 131-139..

Saudabayev, A., Kungozhin, F., Nurseitov, D. & Varol, H. A., 2015. Locomotion strategy selection for a hybrid mobile robot using time of flight depth sensor. *J. Sensors,* 03, Volume 2015, pp. 1-14.

Shalal, N., Low, T., Mccarthy, C. & Hancock, N., 2013. A review of autonomous navigation systems in agricultural. *In Innovative Agricultural Technologies for a Sustainable Future; Society for Engineering in Agriculture (SEAg),* p. pp 1–16.

Shamsuzzoha, M., 2018. *PID Control for Industrial Processes.* s.l.:Intechopen.

Silva, A. P., 2010. *Análise do Risco IOperacional em Bancos Baseada em Redes Neurais Artificiais e Descoberta do Conhecimento em Base de Dados. ,* Recife : Universidade Federal de Pernambuco.

Simões, M. et al., 2021. MANUTENÇÃO DO SOLO EM POMARES DE PESSEGUEIRO. *AGROTEC,* Volume 39.

Singh, M., Parhi, D., Bhowmik, S. & Kashyap, S., 2008. Intelligent Controller for Mobile Robot: Fuzzy Logic Approach. *Proceedings of the 12th International Conference of International Association for Computer Methods and Advances in Geomechanics (IACMAG).*

Slaughter, D. C., Giles, D. K. & Downery, D., 2008. Autonomous robotic weed control systems: A review.. *Computers and Electronics in Agriculture,* Volume 6 1, p. 63–78.

Sodelli, F., 2021. *Métrica.* [Online]
Available at: https://metrica.zendesk.com/hc/pt-br/articles/360023247651-Como-digitar-%C3%A2ngulos-no-M%C3%A9trica-TOPO-Azimute-Rumo-e-%C3%82ngulo-Interno-?mobile_site=true
[Acedido em 29 Julho 2021].

Solectro, 2021. *Solectro Shop.* [Online]
Available at: https://solectroshop.com/pt/sensores-de-distancia-e-proximidade-vao-e-ultrassonicos/1196-hc-sr05-sensor-de-distancia-de-ultrassom.html?gclid=CjwKCAjw4qCKBhAVEiwAkTYsPN1yTV-ryWcJD6Jc8MAJ1h6JcbtDGPMxVGd1ulWR-sEG4djWG4uylRoCCaUQAvD_BwE
[Acedido em 20 Setembro 2021].

Theisen, K. J., 2019. Programming languages in chemistry: a review of HTML5/JavaScript. *Journal of Cheminformatics,* Issue 11:11.

u-blox, 2011. NEO-6 - Data Sheet.

Upaphai, W., Bunyawanichakul, P. & Janthong, M., 2019. Design of Self-tuning Fuzzy PID Controllers for Position Tracking Control of Autonomous Agricultural Tractor. *Pertanika J. Sci. & Technol,* 27(1), pp. 263 - 280.

Veiros, A. F. R. & Gaspar, P. D., 2020. *Sistema robótico terrestre para apoio a atividades de manutenção de solo em pomares de prunóideas,* Covilhã: Universidade da Beira Interior.

Vicente, M. A. F., 1997. *Métodos de Determinação do Azimute por Observações Astronómicas,* Coimbra: Universidade de Coimbra.

VIDAL, N. & VIDAL, R., 2010. AUGMENTED REALITY SYSTEMS FOR WEED ECONOMIC THRESHOLDS APPLICATIONS. *SCIELO,* pp. ISSN 0100-8358.

W3, 2021. *HTML Introduction.* [Online] Available at: https://www.w3schools.com/html/html_intro.asp

Weiss, U. & Peter Biber, 2011. Plant detection and mapping for agricultural robots using a 3D LIDAR sensor. *Robotics and Autonomous Systems,* Volume 59, pp. 265-273.

Yaghoubi, S. et al., 2013. Autonomous Robots for Agricultural Tasks and Farm Assignment and Future Trends in Agro Robots. *International Journal of Mechanical & Mechatronics Engineering IJMME-IJENS,* 13(3).

Yang, W. et al., 2015. Greenness identifcation based on HSV decision tree. *Information Processing in Agriculture*, Volume (3/4), pp. 149-160.

Zhou, C. & Li, S., 2021. Application of children Artificial Intelligence science popularization books based on Augmented Reality technology. pp. 22-26.

Zhu, Z. et al., 2005. Neural network for estimating vehicle behaviour on sloping terrain. *Biosystems Engineering,* Volume 91, pp. 403-411.

Anexos

Anexo A: Algoritmo de navegação autónoma baseada em GPS

```
#include <TinyGPS++.h>
#include <TinyGPS.h>
#include <SoftwareSerial.h>
#include <Wire.h>
#include <HMC5883L.h>

//--Mapeamento de Hardware--
#define sensor1 6
#define sensor2 7
#define in1 11
#define in2 12
#define in3 13


static const int RXPin = 4, TXPin = 3;
static const uint32_t GPSBaud = 4800;

// The TinyGPS++ object
TinyGPSPlus gps;
//bool gpsdump(TinyGPSPlus &gps);

// The serial connection to the GPS device
SoftwareSerial ss(RXPin, TXPin);

//bussola

HMC5883L compass;

#define waypoints 5   // número de waypoints
int waycont=1;
float lat_array[]={28.34140014 , 28.3415508 , 28.3415508 , 28.3414192 ,
28.3413600 };
float  lon_array[]={-82.27369689  ,  -82.2736816  ,  -82.2741317  ,  -
82.2741470 , -82.2739028 };
float flat;
float flon;
float x2lat;
float x2lon;

int HMC6352Address = 0x42;
int slaveAddress;
byte headingData[2];
int i, headingValue;
int headingcompass;
```

```
 //Sensor ultrassônico  HC-SR05
*/
const int triPin = 9;
const int echPin = 10;
long tm;
int disInCm, disInInch;

//--Protótipo das funções auxiliáres

void distance();
void robo_frente();
void robo_tras();
void robo_direita();
void robo_esquerda();
void robo_parado();
//void Posicao_atual()

void setup()
{
  Serial.begin(9600);                                        // Starting
Serial Terminal
  pinMode(triPin, OUTPUT);
  pinMode(echPin, INPUT);

  Wire.begin();
  Serial.begin(115200);
  pinMode(in1,OUTPUT);
  pinMode(in2,OUTPUT);
  pinMode(in3,OUTPUT);

  Serial.begin(9600);

  // Initialize Initialize HMC5883L
  Serial.println("Initialize HMC5883L");
  while (!compass.begin())
  {
    Serial.println("Could  not  find  a  valid  HMC5883L  sensor,  check
wiring!");
    delay(500);
  }

  // Set measurement range
  compass.setRange(HMC5883L_RANGE_1_3GA);

  // Set measurement mode
  compass.setMeasurementMode(HMC5883L_CONTINOUS);

  // Set data rate
  compass.setDataRate(HMC5883L_DATARATE_30HZ);

  // Set number of samples averaged
  compass.setSamples(HMC5883L_SAMPLES_8);

  // Set calibration offset. See HMC5883L_calibration.ino
  compass.setOffset(0, 0);


   Serial.begin(115200);
  ss.begin(GPSBaud);
```

```
  Serial.println(F("FullExample.ino"));
  Serial.println(F("An extensive example of many interesting TinyGPS++
features"));
  Serial.print(F("Testing      TinyGPS++      library      v.      "));
Serial.println(TinyGPSPlus::libraryVersion());
  Serial.println(F("by Mikal Hart"));
  Serial.println();
  Serial.println(F("Sats HDOP  Latitude   Longitude   Fix  Date      Time
Date Alt     Course Speed Card  Distance Course Card  Chars Sentences
Checksum"));
  Serial.println(F("          (deg)     (deg)      Age                 Age
(m)    --- from GPS ----  ---- to Covilhã  ----  RX    RX       Fail"));
  Serial.println(F("----------------------------------------------------
-------------------------------------------------------------------
----------------"));


}

 //Atualiza a posição de gps a cada vez que é chamada
  void Posicao_atual()
{
  Serial.print(F("Location: "));
  if (gps.location.isValid())
  {
    Serial.print(gps.location.lat(), 6);
    Serial.print(F(","));
    Serial.print(gps.location.lng(), 6);
    float flat=gps.location.lat();
    float flon=gps.location.lng();
  }
  else
  {
    Serial.print(F("INVALID"));
  }
}

bool feedgps()
{
  while (ss.available())
  {
    if (gps.encode(ss.read()))
      return true;
  }
  return false;
}

float calc_distance(float flat1, float flon1, float flat2, float flon2)
{
  float delta_lat;
  float delta_lon;
  float dist;
  float a;
  float c;
  int R = 6378140;

 //aplicando as equações de Haversine
  delta_lat = radians((flat2)-(flat1));
  delta_lon = radians((flon2)-(flon1));
  flat1 = radians(flat1);
```

```
  flat2 = radians(flat2);
  a         =         (sin(delta_lat*0.5)*sin(delta_lat*0.5))         +
cos(flat1)*cos(flat2)*(sin(delta_lon*0.5)*sin(delta_lon*0.5));
  c = 2*atan2(sqrt(a),sqrt(1-a));
  dist = c*R;

  return dist;
}

//Função que calcula a direção através das coordenadas geográficas
// Obtido de https://www.sunearthtools.com/pt/tools/distance.php#top

float get_direction(float lat1, float lon1, float lat2, float lon2)
{
  float delta_fi =  log( tan( lat2 / (2 + 3.1415/ 4 )) / tan( lat1/( 2
+ 3,1415 / 4)) );
  float delta_lon2 = abs(lon1-lon2);
  float calc_heading = atan2(delta_lon2, delta_fi);
  calc_heading += 2 * PI;
  return calc_heading;
}

float direcao_bussola()
{
  Vector norm = compass.readNormalize();

  // Calculate heading
  float heading = atan2(norm.YAxis, norm.XAxis);

  // Set declination angle on your location and fix heading
  // You can find your declination on: http://magnetic-declination.com/
  // (+) Positive or (-) for negative
  // For Bytom / Poland declination angle is 4'26E (positive)
  // Formula: (deg + (min / 60.0)) / (180 / M_PI);
  float declinationAngle = (-1.0 - (13.0 / 60.0)) / (180 / M_PI);
  heading += declinationAngle;

  // Correct for heading < 0deg and heading > 360deg
  if (heading < 0)
  {
    heading += 2 * PI;
  }

  if (heading > 2 * PI)
  {
    heading -= 2 * PI;
  }

  // Converter para graus
  float headingDegrees = heading * 180/M_PI;

  // Saida
  Serial.print(" Heading = ");
  Serial.print(heading);
  Serial.print(" Degress = ");
  Serial.print(headingDegrees);
  Serial.println();

  return heading;
```

```
}


void loop()
{

      // bool newdata = false;
     // unsigned long start = millis();
     // while (millis() - start < 250)            //obeter novos dados
a cada 1/4 de segundos
     // {
       // if (feedgps())
        //  newdata = true;
    //  }
    //if (newdata)
    //  {
       // Serial.println("Acquired Data");
       // Serial.println("-------------");
       // Posicao_atual();
       // Serial.println("-------------");
       //Serial.println();

     // }
Serial.print(" robo para tras ");
 for ( int i = 0; i <= 4;)
   {
      while   (calc_distance(gps.location.lat(),   gps.location.lng(),
lat_array[i], lon_array[i]) < 0.5)
      {
        Posicao_atual();
        float       dir       =       get_direction(gps.location.lat(),
gps.location.lng(), lat_array[i], lon_array[i]) - direcao_bussola();
        //if(dir <0){ dir +=  180;}
        //if(dir >PI){dir -= 180;}
        if(dir>=-180){
          if(dir<=0){
            Right()    // virar  a direita
          }
        }
        if((dir >=225)&(dir <= 315))
   Left();      //virar a esquerda

   if((dir >=45)&(dir <= 135))
   Right();//virar a direita

   if((dir >=315)&(dir <= 45))
     Forward(); // seguir em frente
   delay(500);

    digitalWrite(triPin, LOW);
    delayMicroseconds(2);
    digitalWrite(triPin, HIGH);
    delayMicroseconds(10);
    digitalWrite(triPin, LOW);
    tm = pulseIn(echPin, HIGH);
    disInCm  = tm*0.034/2;                                  // Applying
distance formula in centimeters considering sound speed
    disInInch = tm*0.0133/2;                                // Applying
distance formula in centimeters considering sound speed
    Serial.print(disInInch);
```

```
    Serial.print("in, ");
    Serial.print(disInCm);
    Serial.print("cm");
    Serial.println();
    delay(1000);

     if (disInCm > 50)
      {
           Stop();
      }

   }
   if(calc_distance(gps.location.lat(),            gps.location.lng(),
lat_array[i], lon_array[i]) < 0.5){
          Stop();
          delay(1000);
          i++;
          }


}}
//-- Desenvolvimento das funções auxiliares
 void Forward()
   {
  // 1 0 1 - move robot forward
    digitalWrite(in,HIGH);
    digitalWrite(in,LOW);
    digitalWrite(in,HIGH);
    Serial.print(" robo para tras ");
   }

     void Back()
  {
  // 0 1 0 - move the robot backward
    digitalWrite(in1,LOW);
    digitalWrite(in2,HIGH);
    digitalWrite(in3,LOW);
   }
     void Right()
  {
  // 1 0 0 - move the robot right side
    digitalWrite(in1,HIGH);
    digitalWrite(in2,LOW);
    digitalWrite(in3,LOW);
  }
     void Left()
  {
  // 0 1 1 - move the robot left side
    digitalWrite(in1,LOW);
    digitalWrite(in2,HIGH);
    digitalWrite(in3,HIGH);
  }
  void Stop()
 {
  // 0 0 0 - para o robo
    digitalWrite(in1,LOW);
    digitalWrite(in2,LOW);
    digitalWrite(in3,LOW);
    Serial.print("Robô Parado ");
  }
```

Anexo B: Algoritmo de integração do Arduino MEGA2560 com o ESP-WROOM-32

```
int OUT_ONE = 23;
int OUT_TWO = 19;
int OUT_THREE = 18;
int CH1 = 5;
int CH2 = 17;
int CH3 = 16;
//variáveis do arduino
int ard0 = 12;
int ard1= 13;
int ard2= 14;

void setup() {
  pinMode(CH1,INPUT);
  pinMode(CH2,INPUT);
  pinMode(CH3,INPUT);
  pinMode(OUT_ONE,OUTPUT);
  pinMode(OUT_TWO,OUTPUT);
  pinMode(OUT_THREE,OUTPUT);

  pinMode(ard0,INPUT);
  pinMode(ard1,INPUT);
  pinMode(ard2,INPUT);

}

void loop() {
  if (pulseIn(CH3, HIGH) <= 1000){
  } // No Mode
  else if (pulseIn(CH3, HIGH) >= 1200 && pulseIn(CH3, HIGH) <= 1600){
       if(pulseIn(CH1, HIGH) >= 1600){
       digitalWrite(OUT_ONE,LOW);
       digitalWrite(OUT_TWO,LOW);
       digitalWrite(OUT_THREE,HIGH);
    }//FOWARD
    else if(pulseIn(CH1, HIGH) <= 1300){
       digitalWrite(OUT_ONE,LOW);
       digitalWrite(OUT_TWO,HIGH);
       digitalWrite(OUT_THREE,HIGH);
    }//REVERSE
    else if(pulseIn(CH2, HIGH) <= 1300){
       digitalWrite(OUT_ONE,HIGH);
       digitalWrite(OUT_TWO,LOW);
       digitalWrite(OUT_THREE,HIGH);
    }//RIGHT
    else if(pulseIn(CH2, HIGH) >= 1600){
       digitalWrite(OUT_ONE,HIGH);
       digitalWrite(OUT_TWO,HIGH);
       digitalWrite(OUT_THREE,HIGH);
    }//LEFT
    else {
       digitalWrite(OUT_ONE,LOW);

       digitalWrite(OUT_TWO,LOW);
       digitalWrite(OUT_THREE,LOW);
    }//STOP
  } // Manual Mode
  else if (pulseIn(CH3, HIGH) >= 1900){
```

```
    if
(digitalRead(ard0)==HIGH&&digitalRead(ard1)==LOW&&digitalRead(ard2)==H
IGH){
      digitalWrite(OUT_ONE,LOW);
       digitalWrite(OUT_TWO,LOW);
       digitalWrite(OUT_THREE,HIGH);
      }//FOWARD

     else                                                              if
(digitalRead(ard0)==LOW&&digitalRead(ard1)==HIGH&&digitalRead(ard2)==L
OW){
      digitalWrite(OUT_ONE,LOW);
       digitalWrite(OUT_TWO,HIGH);
       digitalWrite(OUT_THREE,HIGH);
      }//REVERSE

       else                                                            if
(digitalRead(ard0)==HIGH&&digitalRead(ard1)==LOW&&digitalRead(ard2)==L
OW){
       digitalWrite(OUT_ONE,HIGH);
       digitalWrite(OUT_TWO,LOW);
       digitalWrite(OUT_THREE,HIGH);
      }//RIGHT

      else                                                             if
(digitalRead(ard0)==LOW&&digitalRead(ard1)==HIGH&&digitalRead(ard2)==H
IGH){
       digitalWrite(OUT_ONE,HIGH);
       digitalWrite(OUT_TWO,HIGH);
       digitalWrite(OUT_THREE,HIGH);
      }//LEFT

       else                                                            if
(digitalRead(ard0)==LOW&&digitalRead(ard1)==LOW&&digitalRead(ard2)==LO
W){
        digitalWrite(OUT_ONE,LOW);
        digitalWrite(OUT_TWO,LOW);
        digitalWrite(OUT_THREE,LOW);
      }//STOP


  } // GPS Mode

}
```

Anexo C: Código HTML da aplicação de realidade aumentada

```
<html>
    <head>
        <title>PrunusBOT project</title>
        <script
src="https://aframe.io/releases/1.0.4/aframe.min.js"></script>
        <script                      src="https://raw.githack.com/AR-js-
org/AR.js/master/aframe/build/aframe-ar-nft.js"></script>
        <!-- Look-at component. We don't need this now, but we will
later. -->
        <script                  src="https://unpkg.com/aframe-look-at-
component@0.8.0/dist/aframe-look-at-component.min.js"></script>
    </head>
    <body>
    <a-scene
        vr-mode-ui="enabled: false"
        arjs='sourceType: webcam; videoTexture: true; debugUIEnabled:
false;'
        renderer='antialias: true; alpha: true'>
            <a-camera gps-projected-camera rotation-reader></a-camera>

            <a-assets>
              <img  id="stop"src="https://cdn.glitch.com/aee37fec-0163-
4ae7-aa0e-68f3fbc44c96%2FImagem5.png?v=1631722257472">
            </a-assets>

<a-text
          value="Waypoint 1
                 latitude: 40.54849561010509
                 longitude:-7.2437013735141
                 Virar a Direita
                 >  >  >  >  >  >  >  >  >  >
                 "
          look-at="[gps-camera]"
          scale="3 3 3"
          gps-entity-place="latitude: 40.54849561010509  ; longitude: -
7.2437013735141 ;"
          align-items: center
        ></a-text>
<a-text
          value="Waypoint 2
                 latitude: 40.548233144540674
                 longitude:-7.243770811013698
                 Seguir em Frente
                 ^  ^  ^  ^  ^  ^  ^  ^  ^
                 "
          look-at="[gps-camera]"
          scale="3 3 3"
          gps-entity-place="latitude: 40.548233144540674  ; longitude:
-7.243770811013698 ;"
          align-items: center
        ></a-text>

<a-text
          value="Waypoint 3
                 latitude: 40.54796496857769
                 longitude:-7.24383536769663
                 Virar a esquerda
                 <  <  <  <  <  <  <  <  <  <
```

```
                  "
          look-at="[gps-camera]"
          scale="3 3 3"
          gps-entity-place="latitude: 40.54796496857769   ; longitude:
-7.24383536769663 ;"
          align-items: center
        ></a-text>

<a-text
          value="Waypoint 4
                 latitude: 40.54791098718087,
                 longitude:-7.2434921052597
                 Parar
                 X  X  X  X  X  X  X  X  X  X
                 "
          look-at="[gps-camera]"
          scale="3 3 3"
          gps-entity-place="latitude: 40.54791098718087  ; longitude: -
7.2434921052597 ;"
          align-items: center
        ></a-text>

      <a-text
          value="Waypoint 4
                 latitude: 40.55201481592982
                 longitude:-7.243350764280298
                 Parar
                 X  X  X  X  X  X  X  X  X  X
                 "
          look-at="[gps-camera]"
          scale="3 3 3"
          gps-entity-place="latitude: 40.55201481592982  ; longitude: -
7.243350764280298 ;"
          align-items: center
        ></a-text>

<a-image   gps-projected-entity-place='latitude:40.547858008675206   ;
longitude: -7.243149153946062 '
              width="5"
              height="5"
              rotation="0 0 0"
              src="#stop"></a-image>
        </a-scene>
    </body>
</html>
```

Anexo D: Código HTML da aplicação de realidade aumentada para o teste com *waypoints* distante 5 m

```
<html>
    <head>
        <title>PrunusBOT project</title>
        <script
src="https://aframe.io/releases/1.0.4/aframe.min.js"></script>
        <script                      src="https://raw.githack.com/AR-js-
org/AR.js/master/aframe/build/aframe-ar-nft.js"></script>
        <!-- Look-at component. We don't need this now, but we will
later. -->
        <script                  src="https://unpkg.com/aframe-look-at-
component@0.8.0/dist/aframe-look-at-component.min.js"></script>
    </head>
    <body>
    <a-scene
        vr-mode-ui="enabled: false"
        arjs='sourceType: webcam; videoTexture: true; debugUIEnabled:
false;'
        renderer='antialias: true; alpha: true'>
            <a-camera gps-projected-camera rotation-reader></a-camera>

            <a-assets>
              <img  id="way1"src="https://cdn.glitch.com/8925f5fb-4a7c-
400b-a7b8-928fe6e127ec%2Fesq.png?v=1633099272499">
              <img  id="stop"src="https://cdn.glitch.com/aee37fec-0163-
4ae7-aa0e-68f3fbc44c96%2Fstop.png?v=1631657495077">

              <img     id="frente"src="https://cdn.glitch.com/8925f5fb-
4a7c-400b-a7b8-928fe6e127ec%2Fkisspng-computer-icons-opposite-arrows-
symbol-right-arrow-
5ac72e63741d83.94298962152300297 94756.png?v=1633001287631">
              <img id="right"src="https://cdn.glitch.com/8925f5fb-4a7c-
400b-a7b8-928fe6e127ec%2Fdir.png?v=1633099272498">
            </a-assets>

     <!-- Waypoint distantes 5 m. -->


      <!-- Waypoint 1. -->
      <a-image gps-projected-entity-place='latitude:40.33965753778872 ;
longitude:  -7.380265850242897 ',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#way1"></a-image>

       <!-- Waypoint 2. -->
      <a-image   gps-projected-entity-place='latitude:40.33969372584911
; longitude:  -7.380300411330721 ',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

       <!-- Waypoint 3. -->
      <a-image  gps-projected-entity-place='latitude:40.339728275271995
; longitude:  -7.380336483954758 ',
               width="10"
```

```
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

      <!-- Waypoint 4. -->
      <a-image   gps-projected-entity-place='latitude:40.33976363026514
; longitude:   -7.3803736124315185 ',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

      <!-- Waypoint 5. -->
      <a-image   gps-projected-entity-place='latitude:40.33979719142242
; longitude:   -7.380410577446708 ',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

      <!-- Waypoint 6. -->
      <a-image  gps-projected-entity-place='latitude:40.339829381999216
; longitude:    -7.380451193238333',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

      <!-- Waypoint 7. -->
      <a-image   gps-projected-entity-place='latitude:40.33986753104998
; longitude:    -7.380485052876364',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

      <!-- Waypoint 8. -->
      <a-image   gps-projected-entity-place='latitude:40.33990016061162
; longitude:     -7.38052727556545',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

      <!-- Waypoint 9. -->
      <a-image    gps-projected-entity-place='latitude:40.3399360193602
; longitude:    -7.380562836938312',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

      <!-- Waypoint 10. -->
      <a-image  gps-projected-entity-place='latitude:40.339966927445985
; longitude:     -7.380604703229294',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>
```

```
      <!-- Waypoint 11. -->
      <a-image   gps-projected-entity-place='latitude:40.34000187621993
; longitude:    -7.380644639501126',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

      <!-- Waypoint 12. -->
      <a-image   gps-projected-entity-place='latitude:40.34003490454951
; longitude:     -7.380682487289281',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

      <!-- Waypoint 13. -->
      <a-image  gps-projected-entity-place='latitude:40.340070875499286
; longitude:     -7.380717059145009',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

       <!-- Waypoint 14. -->
      <a-image   gps-projected-entity-place='latitude:40.34010219538138
; longitude:      -7.380759130621731',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

      <!-- Waypoint 15. -->
      <a-image   gps-projected-entity-place='latitude:40.34013741099216
; longitude:      -7.38079656552882',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

      <!-- Waypoint 16. -->
      <a-image    gps-projected-entity-place='latitude:40.3401691501391
; longitude:       -7.380835573577234',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

      <!-- Waypoint 17. -->
      <a-image  gps-projected-entity-place='latitude:40.340204699684094
; longitude:      -7.380875941316188',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

      <!-- Waypoint 18. -->
      <a-image    gps-projected-entity-place='latitude:40.3402375814455
; longitude:      -7.380913537983846',
               width="10"
```

```
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

      <!-- Waypoint 19. -->
      <a-image   gps-projected-entity-place='latitude:40.34027318765206
; longitude:      -7.380950372616844',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

      <!-- Waypoint 20. -->
      <a-image   gps-projected-entity-place='latitude:40.34030625963677
; longitude:       -7.380994526335815',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

      <!-- Waypoint 21. -->
      <a-image   gps-projected-entity-place='latitude:40.34034057132091
; longitude:       -7.381031113709347',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

      <!-- Waypoint 22. -->
      <a-image   gps-projected-entity-place='latitude:40.34037316613725
; longitude:       -7.381071152370817',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

      <!-- Waypoint 23. -->
      <a-image   gps-projected-entity-place='latitude:40.34040829113311
; longitude:       -7.381107481563077',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

      <!-- Waypoint 24. -->
      <a-image  gps-projected-entity-place='latitude:40.340443530475916
; longitude:       -7.381145846227841',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

      <!-- Waypoint 25. -->
      <a-image  gps-projected-entity-place='latitude:40.340477093313645
; longitude:       -7.38118256855622',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>
```

```
      <!-- Waypoint 26. -->
      <a-image   gps-projected-entity-place='latitude:40.34050913388767
; longitude:       -7.381223154636067',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

      <!-- Waypoint 27. -->
      <a-image  gps-projected-entity-place='latitude:40.340544517394314
; longitude:       -7.381261304323196',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

      <!-- Waypoint 28. -->
      <a-image   gps-projected-entity-place='latitude:40.34057827729557
; longitude:       -7.3812990194448975',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

      <!-- Waypoint 29. -->
      <a-image  gps-projected-entity-place='latitude:40.340612907361674
; longitude:       -7.381337380975831',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

      <!-- Waypoint 30. -->
      <a-image   gps-projected-entity-place='latitude:40.34064555994822
; longitude:       -7.381375247020448',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

      <!-- Waypoint 31. -->
      <a-image  gps-projected-entity-place='latitude:40.340679561125796
; longitude:       -7.381417249308805',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

      <!-- Waypoint 32. -->
      <a-image   gps-projected-entity-place='latitude:40.34071424263445
; longitude:       -7.381453794677718',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

      <!-- Waypoint 33. -->
      <a-image   gps-projected-entity-place='latitude:40.34074862335486
; longitude:       -7.381493021740581',
               width="10"
```

```
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

      <!-- Waypoint 34. -->
      <a-image  gps-projected-entity-place='latitude:40.340780510402936
; longitude:       -7.381531456764051',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

      <!-- Waypoint 35. -->
      <a-image   gps-projected-entity-place='latitude:40.34081608857092
; longitude:       -7.381570805802528',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

      <!-- Waypoint 36. -->
      <a-image  gps-projected-entity-place='latitude:40.340848486686724
; longitude:       -7.38161058192852',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

       <!-- Waypoint 37. -->
      <a-image   gps-projected-entity-place='latitude:40.34089888669842
; longitude:       -7.381664012566414',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#right"></a-image>

      <!-- Waypoint 38. -->
      <a-image   gps-projected-entity-place='latitude:40.34091933066487
; longitude:       -7.38163249661028',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#right"></a-image>


      <!-- Waypoint 39. -->
      <a-image   gps-projected-entity-place='latitude:40.34089343991293
; longitude:       -7.3815929774712075',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

      <!-- Waypoint 40. -->
      <a-image  gps-projected-entity-place='latitude:40.34086328504868;
longitude:       -7.381557438202421',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>
```

```
      <!-- Waypoint 41. -->
      <a-image  gps-projected-entity-place='latitude:40.340824441474844
; longitude:       -7.38150915844105',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

      <!-- Waypoint 42. -->
      <a-image   gps-projected-entity-place='latitude:40.34079499202108
; longitude:       -7.381477038537739',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

      <!-- Waypoint 43. -->
      <a-image   gps-projected-entity-place='latitude:40.34076060943795
; longitude:       -7.381437567989424',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

      <!-- Waypoint 44. -->
      <a-image   gps-projected-entity-place='latitude:40.34072463699968
; longitude:        -7.381401416387059',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

      <!-- Waypoint 45. -->
      <a-image   gps-projected-entity-place='latitude:40.34067158566168
; longitude:         -7.381340295940148',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

      <!-- Waypoint 46. -->
      <a-image   gps-projected-entity-place='latitude:40.34063639307608
; longitude:       -7.381299114249637',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

      <!-- Waypoint 47. -->
      <a-image   gps-projected-entity-place='latitude:40.34062700071348
; longitude:        7.381289621267384',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

      <!-- Waypoint 48. -->
      <a-image  gps-projected-entity-place='latitude:40.340587327530585
; longitude:         -7.381242935467512',
```

```
                width="10"
                height="10"
                rotation="0 0 0"
                src="#frente"></a-image>

      <!-- Waypoint 49. -->
      <a-image  gps-projected-entity-place='latitude:40.340555875822005
; longitude:        -7.381185723868563',
                width="10"
                height="10"
                rotation="0 0 0"
                src="#frente"></a-image>

      <!-- Waypoint 50. -->
      <a-image   gps-projected-entity-place='latitude:40.34052367639858
; longitude:        -7.381166647764429',
                width="10"
                height="10"
                rotation="0 0 0"
                src="#frente"></a-image>

      <!-- Waypoint 51. -->
      <a-image   gps-projected-entity-place='latitude:40.34048714907672
; longitude:         -7.38112691647554',
                width="10"
                height="10"
                rotation="0 0 0"
                src="#frente"></a-image>

       <!-- Waypoint 52. -->
      <a-image   gps-projected-entity-place='latitude:40.34045910573369
; longitude:          -7.381093087164952',
                width="10"
                height="10"
                rotation="0 0 0"
                src="#frente"></a-image>

      <!-- Waypoint 53. -->
      <a-image   gps-projected-entity-place='latitude:40.34042079688891
; longitude:         -7.38105325839385',
                width="10"
                height="10"
                rotation="0 0 0"
                src="#frente"></a-image>

      <!-- Waypoint 54. -->
      <a-image   gps-projected-entity-place='latitude:40.34038203224056
; longitude:         -7.381018601998095',
                width="10"
                height="10"
                rotation="0 0 0"
                src="#frente"></a-image>

      <!-- Waypoint 55. -->
      <a-image   gps-projected-entity-place='latitude:40.34034995205814
; longitude:         -7.380977746986486',
                width="10"
                height="10"
                rotation="0 0 0"
                src="#frente"></a-image>
```

```
      <!-- Waypoint 56. -->
      <a-image   gps-projected-entity-place='latitude:40.34031652262102
; longitude:         -7.380936280860677',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

      <!-- Waypoint 57. -->
      <a-image   gps-projected-entity-place='latitude:40.34027701508381
; longitude:         -7.380893219884426',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

      <!-- Waypoint 58. -->
      <a-image   gps-projected-entity-place='latitude:40.34024625110254
; longitude:         -7.380855791560783',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

      <!-- Waypoint 59. -->
      <a-image  gps-projected-entity-place='latitude:40.340210491068866
; longitude:          -7.380816225106745',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

      <!-- Waypoint 60. -->
      <a-image   gps-projected-entity-place='latitude:40.34017535714954
; longitude:          -7.380778878851063',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

       <!-- Waypoint 61. -->
      <a-image   gps-projected-entity-place='latitude:40.34014499907118
; longitude:          -7.3807418949912496',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

      <!-- Waypoint 62. -->
      <a-image   gps-projected-entity-place='latitude:40.34010872590754
; longitude:          -7.380702490448946',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

      <!-- Waypoint 63. -->
      <a-image   gps-projected-entity-place='latitude:40.34007692217463
; longitude:          -7.380658868460447',
```

```
                width="10"
                height="10"
                rotation="0 0 0"
                src="#frente"></a-image>

      <!-- Waypoint 64. -->
      <a-image   gps-projected-entity-place='latitude:40.34003748883386
; longitude:         -7.380624322857009',
                width="10"
                height="10"
                rotation="0 0 0"
                src="#frente"></a-image>

      <!-- Waypoint 65. -->
      <a-image   gps-projected-entity-place='latitude:40.34000523001805
; longitude:         -7.380587962399465',
                width="10"
                height="10"
                rotation="0 0 0"
                src="#frente"></a-image>

      <!-- Waypoint 66. -->
      <a-image   gps-projected-entity-place='latitude:40.33997362384372
; longitude:         -7.380546496272612',
                width="10"
                height="10"
                rotation="0 0 0"
                src="#frente"></a-image>

      <!-- Waypoint 67. -->
      <a-image  gps-projected-entity-place='latitude:40.339942017654586
; longitude:          -7.380508219847827',
                width="10"
                height="10"
                rotation="0 0 0"
                src="#frente"></a-image>

      <!-- Waypoint 68. -->
      <a-image   gps-projected-entity-place='latitude:40.33990651058896
; longitude:         -7.3804684371304345',
                width="10"
                height="10"
                rotation="0 0 0"
                src="#frente"></a-image>

      <!-- Waypoint 69. -->
      <a-image   gps-projected-entity-place='latitude:40.33987178908061
; longitude:          -7.380424264270636',
                width="10"
                height="10"
                rotation="0 0 0"
                src="#frente"></a-image>

      <!-- Waypoint 70. -->
      <a-image   gps-projected-entity-place='latitude:40.33984232839285
; longitude:          -7.380396656233261',
                width="10"
                height="10"
                rotation="0 0 0"
                src="#frente"></a-image>
```

```
      <!-- Waypoint 71. -->
      <a-image   gps-projected-entity-place='latitude:40.33980865901967
; longitude:           -7.380358004980935',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

       <!-- Waypoint 72. -->
      <a-image  gps-projected-entity-place='latitude:40.339774989629674
; longitude:           -7.380313832121137',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

      <!-- Waypoint 73. -->
      <a-image  gps-projected-entity-place='latitude:40.33972137358345;
longitude:            -7.380258420356565',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

      <!-- Waypoint 74. -->
      <a-image gps-projected-entity-place='latitude:40.33966614592976 ;
longitude:           -7.3801940849098555',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#stop"></a-image>

        </a-scene>
    </body>
</html>
```

Anexo E: Código HTML da aplicação de realidade aumentada para o teste com *waypoints* distante 10 m

```
<html>
    <head>
        <title>PrunusBOT project</title>
        <script
src="https://aframe.io/releases/1.0.4/aframe.min.js"></script>
        <script                    src="https://raw.githack.com/AR-js-
org/AR.js/master/aframe/build/aframe-ar-nft.js"></script>
        <!-- Look-at component. We don't need this now, but we will
later. -->
        <script                  src="https://unpkg.com/aframe-look-at-
component@0.8.0/dist/aframe-look-at-component.min.js"></script>
    </head>
    <body>
    <a-scene
        vr-mode-ui="enabled: false"
        arjs='sourceType: webcam; videoTexture: true; debugUIEnabled:
false;'
        renderer='antialias: true; alpha: true'>
            <a-camera gps-projected-camera rotation-reader></a-camera>

            <a-assets>
              <img  id="way1"src="https://cdn.glitch.com/aee37fec-0163-
4ae7-aa0e-68f3fbc44c96%2Fesq.png?v=1633100284847">
              <img  id="stop"src="https://cdn.glitch.com/aee37fec-0163-
4ae7-aa0e-68f3fbc44c96%2Fstop.png?v=1631657495077">

              <img     id="frente"src="https://cdn.glitch.com/aee37fec-
0163-4ae7-aa0e-68f3fbc44c96%2Fkisspng-computer-icons-opposite-arrows-
symbol-right-arrow-
5ac72e63741d83.94298962152300297 94756.png?v=1633046297702">
              <img id="right"src="https://cdn.glitch.com/aee37fec-0163-
4ae7-aa0e-68f3fbc44c96%2Fdir.png?v=1633100284891">
            </a-assets>

     <!-- Waypoint distantes 10 m. -->


      <!-- Waypoint 1. -->
      <a-image gps-projected-entity-place='latitude:40.33965753778872 ;
longitude:  -7.380265850242897 ',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#way1"></a-image>



       <!-- Waypoint 2. -->
      <a-image  gps-projected-entity-place='latitude:40.339728275271995
; longitude:  -7.380336483954758 ',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>
```

```
      <!-- Waypoint 3. -->
      <a-image   gps-projected-entity-place='latitude:40.33979719142242
; longitude:   -7.380410577446708 ',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>



      <!-- Waypoint 4. -->
      <a-image   gps-projected-entity-place='latitude:40.33986753104998
; longitude:    -7.380485052876364',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>



      <!-- Waypoint 5. -->
      <a-image    gps-projected-entity-place='latitude:40.3399360193602
; longitude:    -7.380562836938312',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>



      <!-- Waypoint 6. -->
      <a-image   gps-projected-entity-place='latitude:40.34000187621993
; longitude:    -7.380644639501126',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>



      <!-- Waypoint 7. -->
      <a-image  gps-projected-entity-place='latitude:40.340070875499286
; longitude:     -7.380717059145009',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>



      <!-- Waypoint 8. -->
      <a-image   gps-projected-entity-place='latitude:40.34013741099216
; longitude:      -7.38079656552882',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>
```

```
      <!-- Waypoint 9. -->
      <a-image  gps-projected-entity-place='latitude:40.340204699684094
; longitude:      -7.380875941316188',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>



      <!-- Waypoint 10. -->
      <a-image   gps-projected-entity-place='latitude:40.34027318765206
; longitude:      -7.380950372616844',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>



      <!-- Waypoint 11. -->
      <a-image   gps-projected-entity-place='latitude:40.34034057132091
; longitude:       -7.381031113709347',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>



      <!-- Waypoint 12. -->
      <a-image   gps-projected-entity-place='latitude:40.34040829113311
; longitude:       -7.381107481563077',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>



      <!-- Waypoint 13. -->
      <a-image  gps-projected-entity-place='latitude:40.340477093313645
; longitude:       -7.38118256855622',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>



      <!-- Waypoint 14. -->
      <a-image  gps-projected-entity-place='latitude:40.340544517394314
; longitude:       -7.381261304323196',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>
```

```
      <!-- Waypoint 15. -->
      <a-image  gps-projected-entity-place='latitude:40.340612907361674
; longitude:       -7.381337380975831',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>



      <!-- Waypoint 16. -->
      <a-image  gps-projected-entity-place='latitude:40.340679561125796
; longitude:       -7.381417249308805',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>



      <!-- Waypoint 17. -->
      <a-image   gps-projected-entity-place='latitude:40.34074862335486
; longitude:       -7.381493021740581',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>



      <!-- Waypoint 18. -->
      <a-image   gps-projected-entity-place='latitude:40.34081608857092
; longitude:       -7.381570805802528',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>

      <!-- Waypoint 19. -->
      <a-image   gps-projected-entity-place='latitude:40.34089888669842
; longitude:       -7.381664012566414',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#right"></a-image>

      <!-- Waypoint 20. -->
      <a-image   gps-projected-entity-place='latitude:40.34091933066487
; longitude:       -7.38163249661028',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#right"></a-image>







      <!-- Waypoint 21. -->
```

```
      <a-image  gps-projected-entity-place='latitude:40.34086328504868;
longitude:       -7.381557438202421',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>



      <!-- Waypoint 22. -->
      <a-image   gps-projected-entity-place='latitude:40.34079499202108
; longitude:       -7.381477038537739',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>



      <!-- Waypoint 23. -->
      <a-image   gps-projected-entity-place='latitude:40.34072463699968
; longitude:        -7.381401416387059',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>



      <!-- Waypoint 24. -->
      <a-image   gps-projected-entity-place='latitude:40.34063639307608
; longitude:       -7.381299114249637',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>



      <!-- Waypoint 25. -->
      <a-image  gps-projected-entity-place='latitude:40.340587327530585
; longitude:         -7.381242935467512',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>



      <!-- Waypoint 26. -->
      <a-image   gps-projected-entity-place='latitude:40.34052367639858
; longitude:        -7.381166647764429',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>



       <!-- Waypoint 27. -->
```

```
      <a-image   gps-projected-entity-place='latitude:40.34045910573369
; longitude:          -7.381093087164952',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>



      <!-- Waypoint 28. -->
      <a-image   gps-projected-entity-place='latitude:40.34038203224056
; longitude:          -7.381018601998095',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>



      <!-- Waypoint 29. -->
      <a-image   gps-projected-entity-place='latitude:40.34031652262102
; longitude:          -7.380936280860677',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>


      <!-- Waypoint 30. -->
      <a-image   gps-projected-entity-place='latitude:40.34024625110254
; longitude:          -7.380855791560783',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>



      <!-- Waypoint 31. -->
      <a-image   gps-projected-entity-place='latitude:40.34017535714954
; longitude:          -7.380778878851063',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>



      <!-- Waypoint 32. -->
      <a-image   gps-projected-entity-place='latitude:40.34010872590754
; longitude:          -7.380702490448946',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>



      <!-- Waypoint 33. -->
```

```
      <a-image   gps-projected-entity-place='latitude:40.34003748883386
; longitude:         -7.380624322857009',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>



      <!-- Waypoint 34. -->
      <a-image   gps-projected-entity-place='latitude:40.33997362384372
; longitude:         -7.380546496272612',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>



      <!-- Waypoint 35. -->
      <a-image   gps-projected-entity-place='latitude:40.33990651058896
; longitude:         -7.3804684371304345',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>



      <!-- Waypoint 36. -->
      <a-image   gps-projected-entity-place='latitude:40.33984232839285
; longitude:          -7.380396656233261',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>



       <!-- Waypoint 37. -->
      <a-image  gps-projected-entity-place='latitude:40.339774989629674
; longitude:           -7.380313832121137',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>



      <!-- Waypoint 38. -->
      <a-image gps-projected-entity-place='latitude:40.33966614592976 ;
longitude:           -7.3801940849098555',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#stop"></a-image>

        </a-scene>
    </body>
</html>
```

Anexo F: Código HTML da aplicação de realidade aumentada para o teste com *waypoints* distante 15 m

```
<!-- Look-at component. We don't need this now, but we will later. -->
        <script                  src="https://unpkg.com/aframe-look-at-
component@0.8.0/dist/aframe-look-at-component.min.js"></script>
    </head>
    <body>
    <a-scene
        vr-mode-ui="enabled: false"
        arjs='sourceType: webcam; videoTexture: true; debugUIEnabled:
false;'
        renderer='antialias: true; alpha: true'>
            <a-camera gps-projected-camera rotation-reader></a-camera>

            <a-assets>
              <img  id="way1"src="https://cdn.glitch.com/0f872697-e3de-
4ce7-a866-dea1e06bed89%2Fesq.png?v=1633101494740">
              <img  id="stop"src="https://cdn.glitch.com/aee37fec-0163-
4ae7-aa0e-68f3fbc44c96%2Fstop.png?v=1631657495077">

              <img     id="frente"src="https://cdn.glitch.com/0f872697-
e3de-4ce7-a866-dea1e06bed89%2Fkisspng-computer-icons-opposite-arrows-
symbol-right-arrow-
5ac72e63741d83.94298962152300297 94756.png?v=1633048900768">
              <img id="right"src="https://cdn.glitch.com/0f872697-e3de-
4ce7-a866-dea1e06bed89%2Fdir.png?v=1633101494796">
            </a-assets>

     <!-- Waypoint distantes 15 m. -->


      <!-- Waypoint 1. -->
      <a-image gps-projected-entity-place='latitude:40.33965753778872 ;
longitude:  -7.380265850242897 ',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#way1"></a-image>



      <!-- Waypoint 2. -->
      <a-image   gps-projected-entity-place='latitude:40.33976363026514
; longitude:   -7.3803736124315185 ',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>



      <!-- Waypoint 3. -->
      <a-image   gps-projected-entity-place='latitude:40.33986753104998
; longitude:    -7.380485052876364',
               width="10"
               height="10"
```

```
               rotation="0 0 0"
               src="#frente"></a-image>



      <!-- Waypoint 4. -->
      <a-image  gps-projected-entity-place='latitude:40.339966927445985
; longitude:     -7.380604703229294',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>



      <!-- Waypoint 5. -->
      <a-image  gps-projected-entity-place='latitude:40.340070875499286
; longitude:     -7.380717059145009',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>



      <!-- Waypoint 6. -->
      <a-image    gps-projected-entity-place='latitude:40.3401691501391
; longitude:       -7.380835573577234',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>



      <!-- Waypoint 7. -->
      <a-image   gps-projected-entity-place='latitude:40.34027318765206
; longitude:      -7.380950372616844',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>



      <!-- Waypoint 8. -->
      <a-image   gps-projected-entity-place='latitude:40.34037316613725
; longitude:       -7.381071152370817',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>



      <!-- Waypoint 9. -->
      <a-image  gps-projected-entity-place='latitude:40.340477093313645
; longitude:       -7.38118256855622',
               width="10"
               height="10"
```

```
                rotation="0 0 0"
                src="#frente"></a-image>



      <!-- Waypoint 10. -->
      <a-image   gps-projected-entity-place='latitude:40.34057827729557
; longitude:       -7.3812990194448975',
                width="10"
                height="10"
                rotation="0 0 0"
                src="#frente"></a-image>



      <!-- Waypoint 11. -->
      <a-image  gps-projected-entity-place='latitude:40.340679561125796
; longitude:       -7.381417249308805',
                width="10"
                height="10"
                rotation="0 0 0"
                src="#frente"></a-image>



      <!-- Waypoint 12. -->
      <a-image  gps-projected-entity-place='latitude:40.340780510402936
; longitude:       -7.381531456764051',
                width="10"
                height="10"
                rotation="0 0 0"
                src="#frente"></a-image>



     <!-- Waypoint 13. -->
      <a-image   gps-projected-entity-place='latitude:40.34089888669842
; longitude:       -7.381664012566414',
                width="10"
                height="10"
                rotation="0 0 0"
                src="#right"></a-image>

      <!-- Waypoint 14. -->
      <a-image   gps-projected-entity-place='latitude:40.34091933066487
; longitude:       -7.381632496661028',
                width="10"
                height="10"
                rotation="0 0 0"
                src="#right"></a-image>



      <!-- Waypoint 15. -->
      <a-image  gps-projected-entity-place='latitude:40.340824441474844
; longitude:       -7.381509158441O5',
                width="10"
                height="10"
                rotation="0 0 0"
                src="#frente"></a-image>
```

```
      <!-- Waypoint 16. -->
      <a-image   gps-projected-entity-place='latitude:40.34072463699968
; longitude:        -7.381401416387059',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>



      <!-- Waypoint 17. -->
      <a-image   gps-projected-entity-place='latitude:40.34062700071348
; longitude:        7.381289621267384',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>



      <!-- Waypoint 18. -->
      <a-image   gps-projected-entity-place='latitude:40.34052367639858
; longitude:        -7.381166647764429',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>



      <!-- Waypoint 19. -->
      <a-image   gps-projected-entity-place='latitude:40.34042079688891
; longitude:         -7.38105325839385',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>



      <!-- Waypoint 20. -->
      <a-image   gps-projected-entity-place='latitude:40.34031652262102
; longitude:         -7.380936280860677',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>



      <!-- Waypoint 21. -->
      <a-image  gps-projected-entity-place='latitude:40.340210491068866
; longitude:          -7.380816225106745',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>
```

```
      <!-- Waypoint 22. -->
      <a-image   gps-projected-entity-place='latitude:40.34010872590754
; longitude:          -7.380702490448946',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>



      <!-- Waypoint 23. -->
      <a-image   gps-projected-entity-place='latitude:40.34000523001805
; longitude:          -7.380587962399465',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>



      <!-- Waypoint 24. -->
      <a-image   gps-projected-entity-place='latitude:40.33990651058896
; longitude:          -7.3804684371304345',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>



      <!-- Waypoint 25. -->
      <a-image   gps-projected-entity-place='latitude:40.33980865901967
; longitude:           -7.380358004980935',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#frente"></a-image>



      <!-- Waypoint 26. -->
      <a-image gps-projected-entity-place='latitude:40.33966614592976 ;
longitude:           -7.3801940849098555',
               width="10"
               height="10"
               rotation="0 0 0"
               src="#stop"></a-image>
        </a-scene>
    </body>
</html>
```