Instituto Superior
de Engenharia
Politécnico de Coimbra

DEPARTAMENTO DE ENGENHARIA
ELETROTÉCNICA

Desenvolvimento de plataforma móvel para testes de navegação para AGV industriais

Trabalho de Projeto para a obtenção do grau de Mestre em Engenharia Eletrotécnica

Especialização em Automação e Comunicações em Sistemas Industriais

Autor

Nuno Edgar Baptista Duarte

Orientadores

Doutor Fernando José Pimentel Lopes

Doutor Inácio de Sousa Adelino da Fonseca

INSTITUTO POLITÉCNICO
DE COIMBRA

INSTITUTO SUPERIOR
DE ENGENHARIA
DE COIMBRA

Coimbra, maio de 2025

RESUMO

Este projeto visa criar uma plataforma de testes com protótipos de robôs móveis para aplicar algoritmos de inteligência artificial para otimizar o funcionamento desses protótipos. Seria importante que este projeto pudesse ser utilizado futuramente para demonstrar e testar, em âmbito académico, o funcionamento de robôs móveis. Além disso, pode ser utilizado para melhorar equipamentos industriais, realizando testes iniciais em laboratório, evitando a necessidade de parar elementos essenciais de uma linha industrial ao implementar ou testar novas ideias e funcionalidades que os técnicos considerem importantes para aumentar a eficácia e fiabilidade desses equipamentos.

Com estes objetivos, foi projetado, montado e programado um computador Central, através do qual é possível monitorizar e controlar os três protótipos de robô contruídos para testes integrados num protótipo de uma fábrica. O protótipo da fábrica possui quatro linhas de produção e três zonas de estacionamento, além de sistemas de localização por RFID que serão lidos pelos protótipos de robô, e sistemas de sinalização que permitem visualizar de forma rápida e eficaz o sentido de cada um dos percursos. Esta sinalização é gerida pelo computador Central, que informa os microcontroladores de cada uma das linhas de produção.

Os protótipos de robô construídos estão equipados com sensores de ultrassons que permitem uma perceção eficaz do espaço em redor. Também possuem sensores óticos que detetam obstáculos e permitem que parem imediatamente. Adicionalmente, estão equipados com um sistema de leitura de RFID para facilitar a localização. Estes protótipos dispõem ainda de um sistema de guiamento de linhas, utilizado em testes com AGV, para que possam seguir trajetórias definidas por linhas. Uma novidade destes protótipos são as rodas "Mecanum" que possibilitam o controlo e o comportamento como robôs omnidirecionais.

Além da Central de controlo, que permite comandar os protótipos de robô para executarem tarefas específicas, foram implementados alguns conceitos de inteligência, como a auto-localização, a partilha de dados sobre o ambiente da área de produção com comunicação entre todos os protótipos via Central, e o cálculo da melhor trajetória conforme os elementos presentes na área de produção.

Por fim, o autor considera este trabalho muito desafiante, com potencial para ser continuado adicionando diversas melhorias e evoluções. A atual plataforma de testes permite o desenvolvimento de novas soluções para problemas reais na área de produção, sem a necessidade de um robô industrial, mas sim utilizando um protótipo com as mesmas funcionalidades.

Palavras Chave: Robôs móveis, AGV, AMR, Controlo, Planeamento de trajetórias, Sensores, Raspberry PI, Arduino.

ABSTRACT

This project aims to create a testing platform with mobile robot prototypes to apply artificial intelligence algorithms to optimize the operation of these prototypes. It would be important for this project to be used in the future to demonstrate and test, in an academic setting, the operation of mobile robots. In addition, it can be used to improve industrial equipment, carrying out initial tests in the laboratory, avoiding the need to stop essential elements of an industrial line when implementing or testing new ideas and functionalities that technicians consider important to increase the effectiveness and reliability of this equipment.

With these objectives, a Central Computer was designed, assembled and programmed, through which it is possible to monitor and control the three prototype robots built for integrated testing in a prototype factory. The prototype factory has four production lines and three parking areas, in addition to RFID location systems that will be read by the robot prototypes, and signaling systems that allow for quick and effective visualization of the direction of each of the routes. This signaling is managed by the Central Computer, which informs the microcontrollers of each of the production lines.

The prototype robots built are equipped with ultrasound sensors that allow for effective perception of the surrounding space. They also have optical sensors that detect obstacles and allow them to stop immediately. Additionally, they are equipped with an RFID reading system to facilitate localization. These prototypes also have a line guidance system, used in AGV tests, so that they can follow trajectories defined by lines. A novelty of these prototypes are the "Mecanum" wheels that allow control and behavior like omnidirectional robots.

In addition to the Control Center, which allows the robot prototypes to be commanded to perform specific tasks, some intelligence concepts were implemented, such as self-localization, sharing of data about the environment of the production area with communication between all prototypes via the Central, and calculation of the best trajectory according to the elements present in the production area.

Finally, the author considers this work to be very challenging, with potential to be continued by adding several improvements and evolutions. The current testing platform allows the development of new solutions for real problems in the production area, without the need for an industrial robot, but rather using a prototype with the same functionalities.

Keywords: Mobile robots, AGV, AMR, Control, Trajectory planning, Sensors, Raspberry PI, Arduino.

AGRADECIMENTOS

Este projeto não teria sido realizado com sucesso, sem o apoio de várias pessoas importantes no percurso pessoal e académico.

Começo por agradecer a toda a minha família, começando pela minha mãe, pelo seu apoio, força e principalmente pela sua paciência, nestes anos de trabalho e estudo pós-laboral. Agradeço em especial ao meu tio Luís pela ajuda, motivação e paciência na construção e fornecimento do material do protótipo da fábrica, também à sua família pelo suporte que foi dado.

Agradeço a toda a restante família mais chegada pelo apoio e motivação.

Agradeço de forma muito especial aos Professores Orientadores, Professor Doutor Fernando Lopes e Professor Doutor Inácio Fonseca, sem o apoio e persistência destes, este projeto teria sido muito difícil terminar, pois conhecem o aluno e sabem que se não insistissem para fazer primeiro a parte escrita, a parte prática nunca mais acabava. Agradeço muito o apoio dado, não só neste projeto, mas em todo o percurso académico.

Não posso deixar de agradecer à empresa onde trabalhei durante vinte anos, que sempre que foi necessário me ajudou e apoiou, a Sirmaf, empresa de grande sucesso e à qual desejo votos de continuidade. Agradeço em especial à administração e colegas de trabalho diretos.

Não podia esquecer os meus amigos especiais, que conheci nesta instituição e sem os quais este percurso não teria sido tão agradável, refiro-me em especial ao André Luís, Nuno Costa, Diogo Silva, Renato Bispo e Gonçalo Gomes, sempre que foi preciso estiveram lá para apoiar, motivar e ajudar.

Por fim agradeço a três amigos de verdade que sempre me apoiaram e motivaram durante este percurso académico, Norberto Ferreira, Mário Lama e Tiago Simões.

Além de todos os que já mencionei, não posso deixar de agradecer ao meu grande amigo Alfredo Jesus. Sempre me incentivou desde o secundário até hoje a não desistir.

A todas estas pessoas agradeço imenso o apoio que me deram e o tempo que disponibilizaram comigo.

Muito obrigado.

Nuno Edgar Baptista Duarte

1 CONTEÚDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABELAS

LISTA DE SIGLAS, ACRÓNIMOS E ABREVIATURAS

AGV – Automated Guided Vehicle

AMR – Autonomous Mobile Robot

CAN – Controller Area Network

LiDAR – Light Detection and Ranging

RFID – Radio Frequency Identification

ROS – Robot Operating System

SLAM – Simultaneous Localization and Mapping

WiFi – Wireless Fidelity

1 INTRODUÇÃO

Neste capítulo é feita a introdução ao trabalho onde são apresentadas as motivações pessoais e profissionais para a realização deste projeto, os seus principais objetivos e a organização do presente relatório.

1.1 Motivações

Com a evolução da tecnologia da automação e robótica na indústria nas últimas décadas, a robótica móvel tem sido uma das áreas a que cada vez mais as empresas recorrem de forma a diminuir acidentes e otimizar os processos de logística internos (ver Figura 1.1). Prevê-se que nos próximos anos este mercado continue a evoluir bastante.

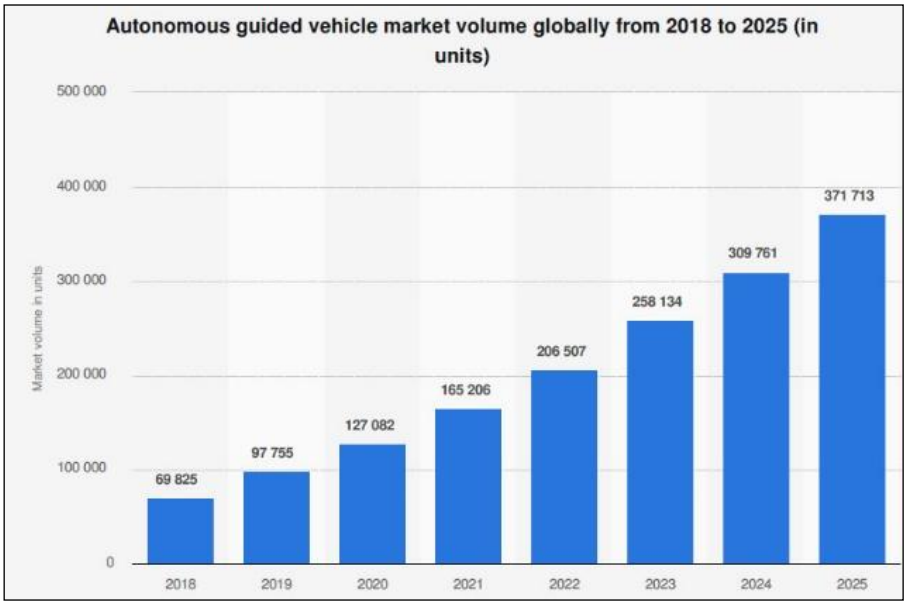


Figura 1.1 – Expectativa da evolução das frotas de AGV (Market Report 2019-2029 Visiongain).

O presente trabalho insere-se na área da automação e robótica industrial. A área da robótica móvel sempre cativou o autor para a entender, analisar e explorar possíveis melhorias nos seus equipamentos.

Na maioria das empresas onde presta serviços existem estes equipamentos. Surgiu então a ideia de desenvolver o projeto de uma plataforma que permitisse às empresas utilizadoras e de fabrico otimizar estes equipamentos sem ter de estar a parar um ou mais elementos da frota.

Por regra, as empresas fabricantes deste tipo de equipamentos desenvolvem e instalam estes equipamentos para o transporte de peças de um ponto para outro por uma rota programada. Caso surja algum problema nesse caminho pré-determinado,

o equipamento para e por regra, não consegue calcular alternativas para retomar a rota e chegar ao destino. Por norma estas falhas são detetadas com a paragem das linhas de processo, sendo estas na maioria das vezes fundamentais para a laboração das empresas.

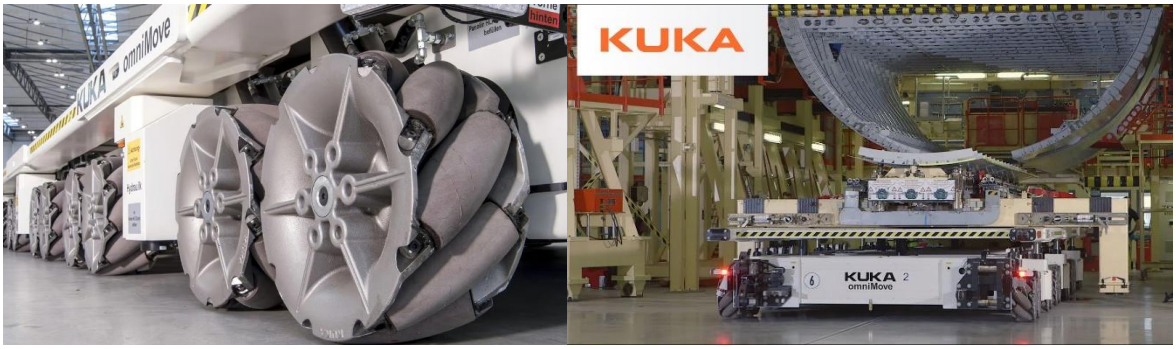
1.2 Objetivos

Este projeto pretende desenvolver uma plataforma para testes de navegação para robôs móveis industriais. Inclui uma Central de controlo e um protótipo de uma fábrica, de modo a permitir, desenvolver, comprovar e testar novas funcionalidades para possível aplicação em equipamentos industriais. Pretende-se evitar que as empresas necessitem de um equipamento de testes constantemente disponível, ou retirar um elemento da frota ativa para testar novas funcionalidades, e desta forma conseguirem uma constante melhoria da sua frota de robôs móveis sem prejudicar a produção.

Pode também ser utilizado para demonstração das tecnologias aplicadas nestes equipamentos, assim como em testes e trabalhos académicos de robótica móvel.

1.3 Trabalho

Na elaboração deste projeto decidiu-se construir três protótipos de robôs movéis, para realizar tarefas recorrendo ao conceito de enxame e partilha de informações entre os diversos protótipos e uma Central de controlo. Estes protótipos de robô dispõem de diversos sensores e atuadores que existem tanto em AGV como AMR [1]. Os equipamentos foram idealizados para funcionarem com rodas omnidirecionais (Figura 1.2), que permitem efetuar os movimentos de forma mais eficaz, não necessitando de tanto espaço como os equipamentos com rodas convencionais. Em todos os protótipos de robô, podem ser testados e verificados todos os movimentos elementares com este tipo de rodas, tendo sido implementados e testados vinte e dois movimentos elementares.



a) b)

Figura 1.2 – Rodas Omnidirecionais: a) Roda tipo "Mecanum"; b) Aplicação de rodas omnidirecionais na Airbus para transporte de partes de aviões.

Construiu-se o protótipo de uma fábrica que permite simular e testar o comportamento dos protótipos de robô em "campo" com as funções que forem desenvolvidas para testar. Localmente é possível ver o sentido de movimento de cada troço assim como efetuar pedidos em cada uma das linhas.

Foi projetada, montada e programada uma Central de controlo da frota, que permite saber a localização de cada protótipo, assim como comandar e testar as funções desenvolvidas. O utilizador interage com a Central recorrendo a uma página Web existente na mesma, podendo atuar localmente ou através de um equipamento externo.

Nos protótipos de robô são usadas diversas tecnologias que se podem encontrar em AGV e AMR industriais, tais como, leitor RFID, sensores de ultrassons, sensores óticos, sensores de seguimento de linhas, rede WiFi e elementos de sinalização sonora e luminosa.

Cada um destes elementos é apresentado e explicado mais pormenorizadamente nos capítulos seguintes.

Após a realização da montagem dos diversos elementos apresentados anteriormente, foram realizados diversos testes de demonstração das diversas funções.

Foram ainda desenvolvidas e testadas algumas funções de programação dinâmica de forma a rentabilizar a frota, tais como, cálculo de trajeto mais curto, cálculo de trajetos alternativos, partilha de bloqueio de trajetos.

1.4 Estrutura do relatório

Este relatório está dividido em cinco capítulos:

Capítulo 1: É feita a introdução ao tema deste projeto, apresentando as motivações para a realização do mesmo e fazendo uma pequena síntese de trabalhos e testes realizados.

Capítulo 2: Apresentação de um resumo da tecnologia atual na área da indústria da robótica móvel, apresentando as indústrias utilizadoras deste tipo de equipamentos, os tipos de equipamentos existentes, tipos de sensores de guiamento e localização, as arquiteturas de hardware e software usadas e as novas tendências.

Capítulo 3: São apresentados pormenorizadamente o protótipo de robô e o protótipo da fábrica desenvolvidos, descrevendo as funções implementadas, as opções de hardware e software, os equipamentos e sensores utilizados, incluindo as opções idealizadas para o protótipo da fábrica. É ainda apresentado a estrutura e conceção da Central de controlo.

Capítulo 4: São apresentados diversos testes e resultados de tarefas e funções apresentadas no Capítulo 3.

Capítulo 5: É apresentada a conclusão dos trabalhos realizados neste projeto, as possíveis melhorias e sugestões de trabalhos futuros.

2 ROBÓTICA MÓVEL E AGV INDUSTRIAIS

Neste capítulo é apresentado um resumo da tecnologia atual da robótica móvel, diferenciando os vários tipos de equipamentos existentes e as indústrias que recorrem à robótica móvel. São apresentadas as diversas tecnologias de sensorização, comunicação, arquiteturas de funcionamento e as mais recentes tecnologias utilizadas nos robôs industriais.

2.1 Tipos de equipamentos existentes

Existem na atualidade diversos tipos de robôs móveis, alguns deles já se podem encontrar em uso doméstico para ajudar nalgumas tarefas, tais como robôs de limpeza, de manutenção de jardim e com o passar do tempo mais robôs móveis tendem a aparecer.

Na área industrial os dois conceitos de robôs móveis mais conhecidos, são os AGV (*Automated Guided Vehicle*) e os AMR (*Autonomous Mobile Robot*) [1].

Os primeiros robôs móveis a serem aplicados em ambientes industriais foram os AGV. Com a evolução da robótica e da sensorização, apareceram os AMR, que cada vez mais são a opção escolhida devido às suas potencialidades, capacidades e flexibilidade de adaptação à mudança de *layouts* e ambientes industriais

2.1.1 AGV

Este tipo de robôs (Figura 2.1) tal como o nome indica, são veículos auto guiados, o que implica que este tipo de robô necessite de algum tipo de guiamento físico externo para se poder deslocar num determinado trajeto. Existem vários tipos de guiamento para este tipo de robô. Estes tipos de guiamento são referidos mais pormenorizadamente na Secção 2.4.4.

Figura 2.1 – Exemplo de um AGV.

Estes equipamentos são muito fiáveis e eficazes quando o *layout* de produção de uma determinada indústria não sofre constantes modificações e quando no caso da utilização de banda magnética não existem outros veículos que a danifiquem com muita frequência. Do conhecimento prático do autor, que verificou em certas

indústrias que tinham estes tipos de equipamentos, o espaço de manobra por vezes tem de ser grande de forma que o robô possa fazer os trajetos sem perder o seu guiamento. Se o trajeto for feito com traçados muito apertados, por vezes os AGV saem do trajeto, causando perdas de produção pois na maioria das vezes estes estão a abastecer linhas de montagem. O seu erro na trajetória por norma só é verificado quando o AGV deixou de trazer as peças necessárias para a produção.

Quando existem trajetos comuns para diversos AGV, para tomar as decisões em cruzamentos, tem de se recorrer sempre a outros elementos que ajudem na decisão do trajeto a tomar. Por norma são utilizadas *tags* RFID para que o AGV na sua tabela de trajeto possa reduzir velocidades e ou decidir qual o trajeto da guia que deve seguir (Figura 2.2).

Para este tipo de equipamentos tem sempre de existir trabalho na área de produção, colocando os sistemas de guiamento e equipamentos nos pontos de decisão.

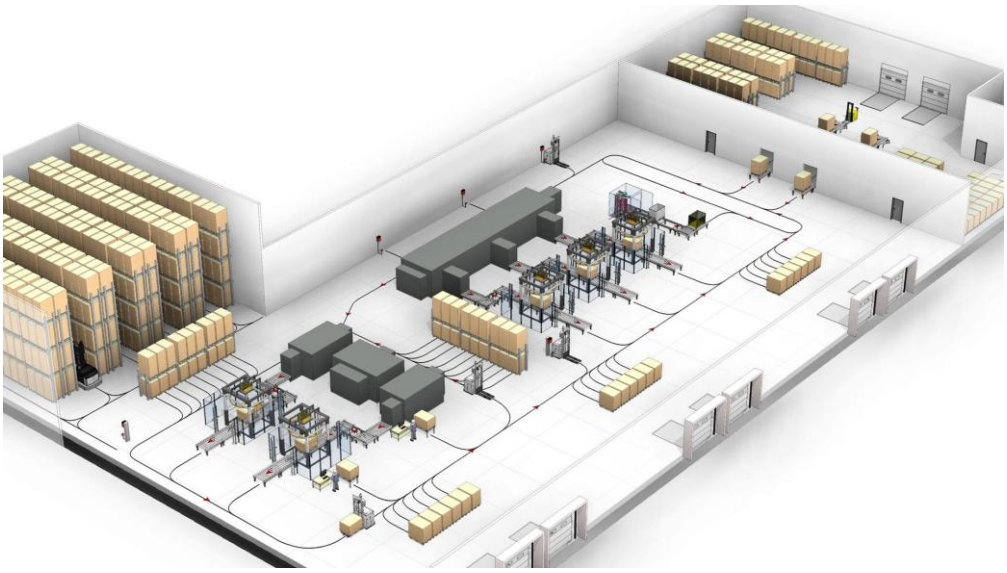


Figura 2.2 – Exemplo de complexidade de uma instalação com recursos a AGV.

2.1.2 AMR

Este tipo de robôs (Figura 2.3) são uma evolução dos AGV [1], sendo que utilizam sensores para efetuar a perceção de espaço e por norma estão dotados da capacidade de mapeamento da área de produção. Com este mapeamento não são necessários sistemas de guiamento físicos externos e não são necessários os elementos externos para decisão. Outra das capacidades que este tipo de equipamentos dispõe é que consegue fazer localização e atualização dos mapas. Esta capacidade tem a designação de *SLAM (Simultaneous Localization and Mapping)*.

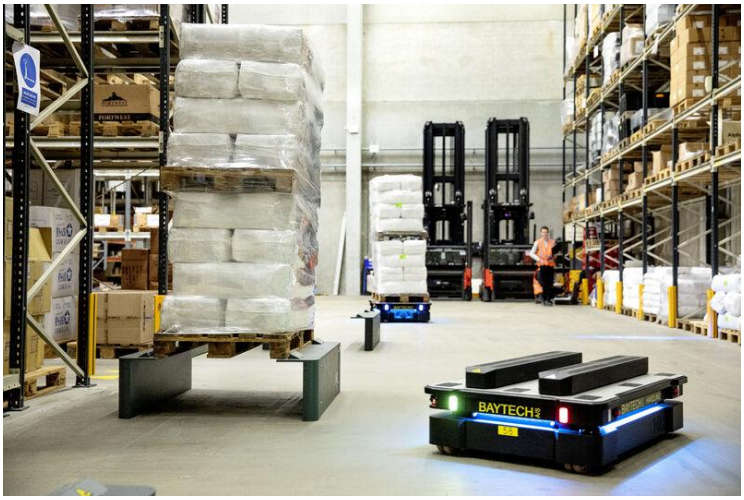


Figura 2.3 – Exemplo de um AMR.

Como o nome diz são veículos autónomos que têm a capacidade de se mover na área de produção de forma prática e fácil.

Podemos considerar que os robôs utilizados em tarefas de limpeza domésticas são AMR pois a maioria consegue fazer o mapeamento da casa e facilmente consegue saber os trajetos e otimizar as suas tarefas.

Por norma este tipo de equipamentos usa sensores de varrimento laser ou ultrassons para executar as tarefas de mapeamento e para referenciamento da sua localização. Pela investigação que o autor realizou e os equipamentos que já observou deste tipo, é usado o sensor Lidar de segurança para fazer o mapeamento e o reconhecimento da localização atual.

Comparando com os AGV, estes equipamentos são mais competitivos para os seus utilizadores, são mais flexíveis, pois adaptam-se mais facilmente à mudança de layout e são em geral mais baratos porque os AGV acrescentam o custo da infraestrutura de guiamento.

Ou seja, pode-se verificar que o facto de não precisar de meios de guiamento externos reduz custos e complexidade do sistema, aumenta a flexibilidade e diminui as necessidades de manutenção pois não é necessária a constante verificação se os guiamentos externos se encontram em bom estado.

Uma das desvantagens destes equipamentos (AMR) é a necessidade de colaboradores com conhecimentos mais técnicos a nível da complexidade do mapeamento e interpretação do mesmo, enquanto nos AGV o técnico sabe que em geral basta colocar os meios de guiamento e localização externos e testar.

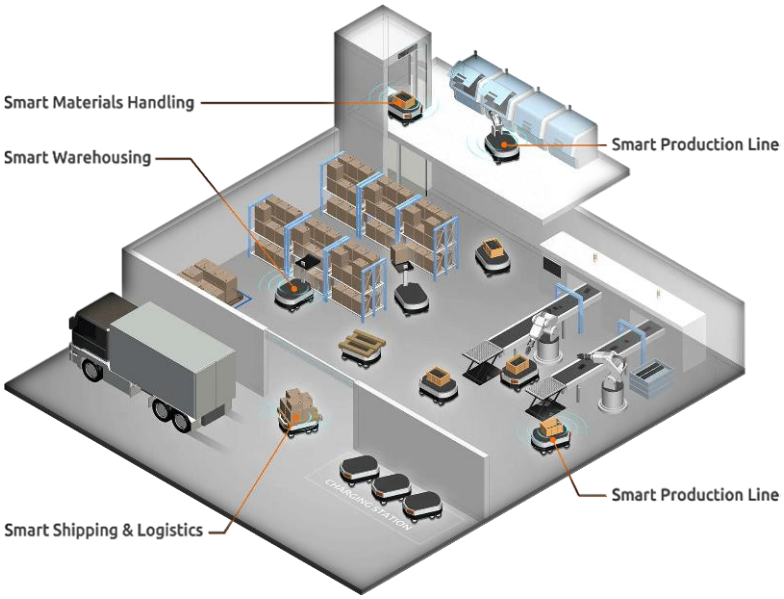


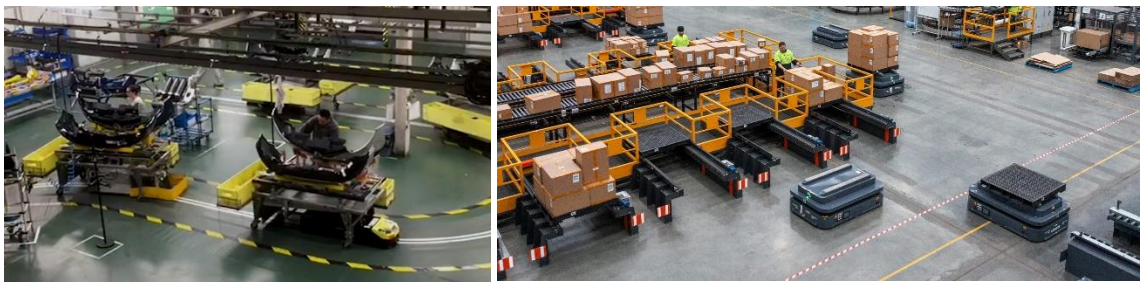
Figura 2.4 – Exemplo de complexidade de uma instalação com recursos a AMR.

Como se pode verificar na Figura 2.4 e comparando com a Figura 2.2 a instalação fica mais simples não havendo a necessidade de recorrer a meios físicos de guiamento do robô. Estes utilizam o mapeamento realizado através dos sensores, sendo que mesmo quando estão a efetuar operações de trabalho vão atualizando o mapa da instalação.

2.2 Tipos de indústria que utilizam robôs móveis

Com a evolução da robótica móvel muitas indústrias começaram a aderir a este tipo de equipamentos, para reduzir acidentes com empilhadores e otimizarem os seus processos logísticos (Figura 2.5).

Quase todos os tipos de indústria transformadora passaram a utilizar este tipo de equipamentos para tarefas logísticas, desde a alimentação de linhas e o transporte de produtos entre linhas e armazenamento e logística de produto acabado (Figura 2.5).



a) b)

Figura 2.5 – Utilização de robôs móveis em indústria transformadora: a) alimentação de para-choques para uma linha de montagem; b) operações de logística.

As empresas de logística como por exemplo a Amazon passaram a utilizar este tipo de equipamentos para preparar encomendas e efetuar a gestão logística de pedidos, fiabilizando e otimizando os processos logísticos (Figura 2.6).

a) b)

Figura 2.6 – Utilização de robôs moveis em indústrias de logística: a) AGV da Amazon Robotics usados nos armazéns da empresa; b) exemplo de processo em empresa de transportes e logística.

Um dos sectores industriais que também está a dar grandes passos na utilização de robôs moveis é o sector agrícola (Figura 2.7). Estão a ser desenvolvidas diversas aplicações, desde a utilização nos tratamentos, manutenção dos solos, e colheita.

a) b)

c) d)

Figura 2.7 – Utilização de AGV em indústrias de logística: a) robô a efetuar tarefas de fertilização; b) robô a colher uvas; c) robôs a colher alfaces recorrendo a braço robótico; d) robô a realizar a tarefa de manutenção do solo.

Existem diversos sectores industriais que começam também a estudar a utilização de drones (Figura 2.8) para trabalharem em conjunto com estes robôs não tendo assim de ser feito todo o deslocamento por solo.

Figura 2.8 – Projeto de utilização de drones na apanha de fruta com robôs móveis.

Nas indústrias da saúde e hospitais (Figura 2.9) estes equipamentos são utilizados para transportar amostras e medicamentos entre departamentos das suas instalações.

Figura 2.9 – Robôs móveis para utilização em Hospitais.

Outra das utilizações que se está a dar a este tipo de equipamentos é no processo de maquinação e montagem, sendo que o robô móvel é dotado de um braço robótico que permite retirar peças de um equipamento, colocar sobre a área de transporte, deslocar-se para outra máquina e colocar as peças retiradas do processo anterior num novo processo (Figura 2.10).

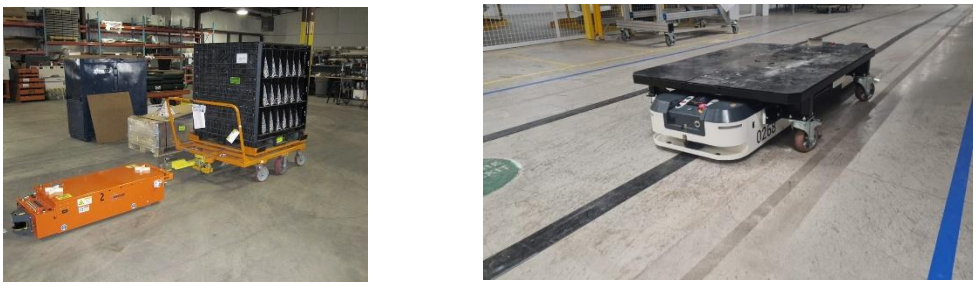
Figura 2.10 – Robô móvel com braço robótico.

2.3 Tipos de funções dos robôs móveis

Tanto na família dos AGV como na dos AMR existem diversos tipos de *designs* de robôs que permitem aos mesmos se adaptarem a diversas funções de trabalho conforme o seu *design*. Neste subcapítulo são apresentados os diversos tipos de robôs mais utilizados.

2.3.1 Robôs de reboque

Este tipo de construção de robôs é concebido para o executar o transporte de cargas em carros de transporte. Normalmente este tipo de robô não tem a capacidade de carga e descarga pelos seus meios, sendo necessárias ações externas para estas tarefas. Os carros podem ser puxados pelo AGV ou o mesmo ter capacidade de se colocar por baixo e levar o sistema de transporte (Figura 2.11).



a) b)

Figura 2.11 – Robôs de reboque: a) Robô a rebocar um carrinho atrás de si; b) Robô a transportar um carrinho sobre si.

2.3.2 Robôs de reboque tipo comboio

Este tipo de robôs é concebido para normalmente executar o processo de comboios logísticos. São utilizados essencialmente para transporte de um ou mais carros com cargas diversas. Este tipo de robôs é muito utilizado por exemplo para transportar

diversos componentes para uma linha de montagem. Tal como o robô do ponto anterior, não tem a capacidade de carga e descarga autónoma (Figura 2.12).

Figura 2.12 – Robô rebocador.

2.3.3 Robôs de transporte de cargas

Este tipo de robôs é idealizado para transportar a carga sobre si. Por norma estes robôs são dotados de meios que permitem a carga e descarga automaticamente e gerida por si. Estes robôs não necessitam assim de qualquer tipo de ação externa para realizar os transportes entre os pontos de trabalho. Os tipos de equipamentos para realizar as operações de carga e descarga podem ser rolos motorizados ou elevadores hidráulicos (Figura 2.13).

Figura 2.13 – Robô de transporte de cargas.

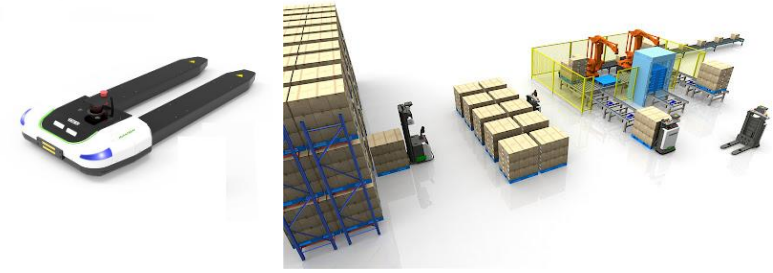
2.3.4 Robôs de garfos tipo empilhador

Neste tipo de robôs o equipamento é dotado de garfos (Figura 2.14), que permitem assim efetuar a logística entre armazéns verticais e áreas de produção, reduzindo também a área de armazenamento. Existem muitos modelos destes robôs que possuem a capacidade híbrida, ou seja, têm postos de condução para humanos.

Figura 2.14 – Robô de garfos tipo empilhador.

2.3.5 Robôs de garfos tipo porta paletes

Estes robôs têm um conceito semelhante ao do ponto anterior sendo que a capacidade de elevação é mais reduzida permitindo apenas elevar as paletes para uma altura de transporte em que a mesma não toca no chão. Este tipo de robôs pode ser utilizado no transporte de linhas de produção para a zona de armazém, onde estão os robôs tipo empilhador (Figura 2.15).



a) b)

Figura 2.15 – Robôs de garfos: a) Robô de garfos tipo porta paletes; b) Exemplo da utilização dos tipos de robôs de garfos.

2.3.6 Outros tipos de funções

Existem outros tipos de robôs que são semelhantes aos apresentados acima, mas com outras funções especificas.

Existem robôs de transporte de cargas ligeiras que levam a carga sobre si, mas como o nome diz são robôs para transportar pequenos pesos. Por norma estes robôs são muito mais rápidos e ágeis que os restantes desta categoria.

Existem no sentido inverso os robôs de cargas elevadas que ao contrário dos anteriores são muito grandes e logo são menos ágeis e mais lentos.

Existem os robôs dotados com sistema de tesoura que permite que uma plataforma mais pequena possa ser elevada a uma altura maior por exemplo para colocar uma carga numa mesa mais elevada.

Com garfos, existem também robôs que são dedicados a carga e descarga de camiões. Estes robôs possuem uma construção que lhes permite deslocar-se dentro de um camião sem o danificar (Figura 2.16).

a) b) c) d) e) f)

Figura 2.16 – Diversos tipos de funções: a) robô de cargas ligeiras; b) robôs de cargas ligeiras a trabalhar; c) robô de cargas pesadas a transportar *bogie* de comboio; d) robô de carga pesada a transportar um rolo de chapa de aço; e) frota de robôs com plataforma de tesoura; f) robô de garfos dedicado a carga e descarga de camiões.

2.4 Tipos de sensores de guiamentos e de localização utilizados

Os robôs móveis são dotados de sensores e sistemas que permitem o seu melhor funcionamento. Existem diversos sensores que são comuns em ambos os principais tipos de robôs referidos anteriormente, outros são utilizados em apenas cada um deles. Nos próximos subcapítulos são apresentados os sensores mais comuns aplicados num robô móvel. Existem outros sensores que são utilizados em aplicações especificas de robôs. No entanto, com os sensores indicados, é possível efetuar as diversas tarefas de segurança, guiamento, localização e controlo de movimento.

2.4.1 Sensores de segurança

Como qualquer equipamento autónomo a prioridade é sempre a segurança do equipamento, áreas envolventes e principalmente os humanos que trabalham perto ou com os robôs. Para isso qualquer um dos tipos de robô está equipado com sensores de segurança (Figura 2.17) que ao detetarem algum obstáculo param o robô. Cada robô, normalmente, está dotado de um sensor deste tipo. Pode estar dotado de mais do que um se o robô for bidirecional ou omnidirecional, sendo que em cada lateral do robô existe um sensor de segurança. Por norma estes sensores são sensores laser. Cada vez mais estes sensores estão equipados com a capacidade LiDAR que, para além de realizar as tarefas de segurança, são utilizados para realizar tarefas de mapeamento e navegação.

Figura 2.17 – Exemplo de sensor de segurança a laser.

Na função de segurança também são utilizados sensores de segurança anticolisão (Figura 2.18), que por norma são sensores de borracha que ao efetuarem um toque em objetos ou pessoas provocam a paragem imediata do robô. Este tipo de sensores pode ser conhecido também por bordo sensível.

Figura 2.18 – Exemplo de sensor de anticolisão.

2.4.2 Sensores de navegação e mapeamento

Tal como abordado no tópico anterior existem sensores LiDAR (Figura 2.19) que permitem efetuar o mapeamento e localização do robô em tempo real, assim como permitem que o robô se vá adaptando de forma fácil à mudança de layout da instalação onde trabalha.

Figura 2.19 – Sensor LiDAR.

Existem os sensores de visão (Figura 2.20) que permitem por exemplo através de visão artificial localizar pontos de trabalho. Estes sensores podem ser câmaras de visão ou mesmo sensores de visão mais dedicados a tarefas especificas, por exemplo, o reconhecimento de posição de objetos e leituras de códigos de barras.

Figura 2.20 – Sensor de visão.

Os sensores ultrassónicos (Figura 2.21) são sensores que permitem detetar objetos inesperados na trajetória e simultaneamente efetuar o mapeamento da instalação.

Figura 2.21 – Sensor ultrassónico para evitar obstáculos.

Os sensores óticos (Figura 2.22) podem ter duas funções num robô. Podem ser utilizados para detetar objetos e comandar a paragem do robô antes de colidir com o objeto. No entanto, uma das suas principais funções é o guiamento por linha. Nos AGV, se o cliente final do robô não pretender usar a banda magnética e optar por utilizar somente a linha pintada no solo, estes sensores são usados como seguidor de linhas, em que são utilizados vários sensores para que o AGV se consiga manter centrado na linha - este tipo de montagem tem o nome de seguidor de linhas ("Line Follower").

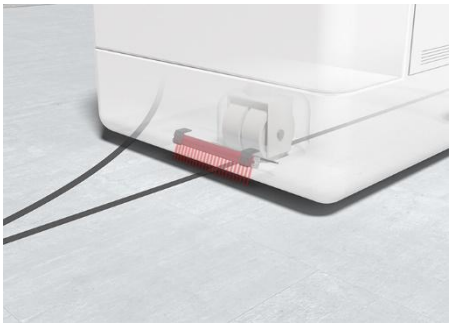


Figura 2.22 – Sensor ótico para guiamento.

Os sensores magnéticos (Figura 2.23) são utilizados normalmente para efetuar o guiamento através de banda magnética nos AGV. Estes sensores encontram-se na parte de baixo do AGV no sistema que efetua o controlo de direção do mesmo.

Figura 2.23 – Sensor magnético de guiamento.

2.4.3 Sensores de posicionamento

Qualquer robô móvel necessita de meios auxiliares para executar um melhor controlo do seu posicionamento e controlo do seu movimento. Tanto os AGV como os AMR dispõem de sensores que permitem controlar a trajetória que está a ser executada pelos seus motores. Existem vários sensores que podem ajudar a melhorar o processo de controlo do robô tais como os giroscópios, que controlam a orientação e rotação do sistema de direção das rodas, e os acelerómetros, que permitem controlar as grandezas de movimento como a aceleração e desaceleração do robô. No entanto, o principal tipo de sensor utilizado são os sensores que medem a rotação de cada uma das rodas. Por norma os sensores mais utilizados são os *encoders*.

Figura 2.24 – Exemplo de montagem de *encoder* em roda de robô.

Como se pode verificar na Figura 2.24 existem *encoders* que podem ser montados no servomotor e ou no sistema mecânico da roda. Usando *encoders* nas duas localizações é possível verificar possíveis escorregamentos no sistema e com essa informação conseguir corrigi-los com o sistema de controlo.

2.4.4 Tipos de guiamento e localização

Na robótica móvel existem diversos tipos de guiamento, os AGV, por norma recorrem a meios físicos externos para efetuar a sua localização e guiamento, enquanto os AMR utilizam os dados de alguns dos seus sensores para se localizarem e deslocarem de forma mais eficaz.

É importante referir os diversos tipos de guiamento a que um AGV pode recorrer. O mais conhecido é a instalação de linhas no chão (Figura 2.25) para o robô seguir. Estas podem ser linhas pintadas ou fita colada no chão e o AGV é dotado de sensores óticos capazes de seguir as mesmas. Este tipo de guiamento é muito frágil pois numa área de produção muito utilizada a linha vai desaparecendo causando mais tarde problemas na trajetória do robô. Para minimizar este problema, foi criado o guiamento por banda magnética, que é baseado no sistema anterior, mas usando linhas propriedades magnéticas e com fitas de proteção, o que permite uma manutenção mais fácil do trajeto, visto que ao se verificarem danos na fita, basta reforçar ou substituir a mesma.

Nos pontos de decisão destes sistemas, como em entroncamentos, é utilizado por norma o sistema RFID com *tags* ou cartões, em que quando o AGV lê os mesmos toma a decisão conforme o software que está a ser executado.

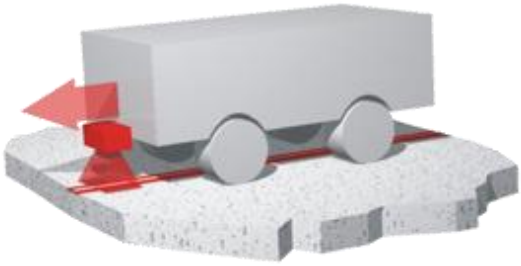


Figura 2.25 – Guiamento por linha.

Também existe o guiamento por fio magnético montado sob o chão (Figura 2.26). Este é mais eficaz no guiamento em relação aos anteriores, mas para modificar percursos é mais dispendioso pois é sempre necessária obra civil para poder efetuar modificação de traçados.

Figura 2.26 – Guiamento por guia indutiva.

Outro dos meios de guiamento físico bastante utilizado é o dos pontos de ancoragem (Figura 2.27). Este sistema recorre à instalação de pontos de referência que o AGV utiliza para se referenciar e tomar decisões. Por norma são *tags* RFID que o robô lê e na sua tabela de programação toma uma determinada decisão de movimento, descolando-se até outra *tag* onde toma a decisão seguinte. Com este sistema o AGV pode-se deslocar de forma livre entre *tags,* não sendo necessário nenhum dos meios referidos anteriormente.

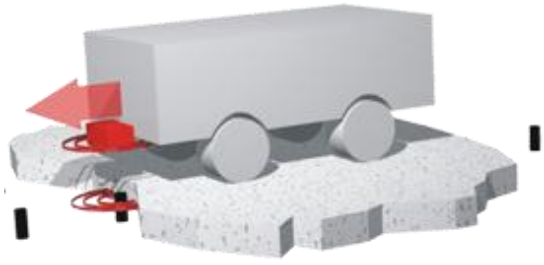


Figura 2.27 – Guiamento por pontos de ancoragem.

Existe o sistema de guiamento por laser, que é constituído por um emissor de laser e pela instalação de diversos pontos de referência, quer sejam estes refletores ou lasers emissores. Este tipo de guiamento precisa sempre de ter no mínimo três

pontos de referência, de forma a fazer a triangulação e conseguir saber a localização e orientação do robô. A desvantagem deste sistema é que é preciso sempre garantir o mínimo dos três pontos de referência e ter uma área circundante não muito elevada de forma a não obstruir pontos de referência (Figura 2.28).

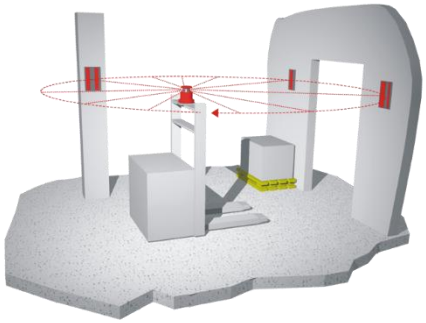


Figura 2.28 – Guiamento por triangulação laser.

Os robôs que se movem no exterior das instalações podem recorrer ao sistema GPS para localização. Normalmente neste tipo de sistema são utilizados meios fixos nas instalações para ajudar na localização e navegação do robô por este meio de guiamento (Figura 2.29).

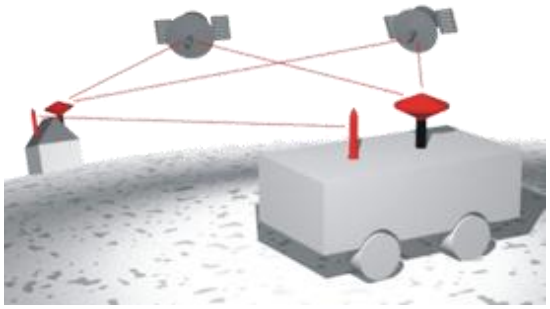


Figura 2.29 – Guiamento por GPS.

2.5 Tipos de comunicações utilizadas

Este tipo de equipamentos utiliza diversos tipos de comunicação para o seu funcionamento. Nesta secção são referidos os diversos tipos de comunicação mais utilizados neste tipo de equipamento.

2.5.1 RFID

Por norma os AGV recorrem a este tipo de comunicação para localização de pontos de decisão e referência ou mesmo para guiamento. Neste tipo de comunicação os robôs são dotados de um leitor RFID, que por norma está montado na parte inferior. No solo estão colocados em pontos estratégicos cartões ou *tags* RFID, que têm configurado um código que na tabela de programação do AGV podem desde indicar pontos de aceleração e desaceleração, pontos de decisão de trajetória e pontos de

paragem para tomar alguma decisão como cargas, descargas, cruzamentos, entre outras informações (Figura 2.30).

a)

b)

c)

d)

Figura 2.30 – Sistema RFID: a) Antena de leitura RFID; b) *Tags* RFID; c) Cartões RFID; d) Instalação de pontos RFID no chão.

2.5.2 WiFi e Ethernet

Embora este tipo de comunicação frequentemente apresente alguma latência e interferência, é um dos mais usados nos robôs móveis (Figura 2.31). Por norma serve para comunicar com o sistema central, mas também para comunicar por exemplo, com sistemas externos dos pontos de trabalho dos robôs. Tipicamente os robôs utilizam estas tecnologias de comunicação para comunicar com equipamentos externos que controlam a sua carga e descarga. Para isso o robô tem instalado um módulo WiFi que vai transmitindo dados para os elementos externos e vice-versa. Os sinais utilizados na comunicação têm de ser configurados nos diversos elementos da rede, de forma a não causar problemas inesperados no correto funcionamento de toda a estrutura de trabalho. Por exemplo, o AGV indica que está no ponto de descarga através de uma variável que normalmente todos os elementos externos recebem. O único elemento que deve dar uso a esta variável deve ser o posto de descarga. Mesmo que o posto de carga receba este sinal, deve ignorar o mesmo e não fazer uso dele de forma a não colocar material para o chão.

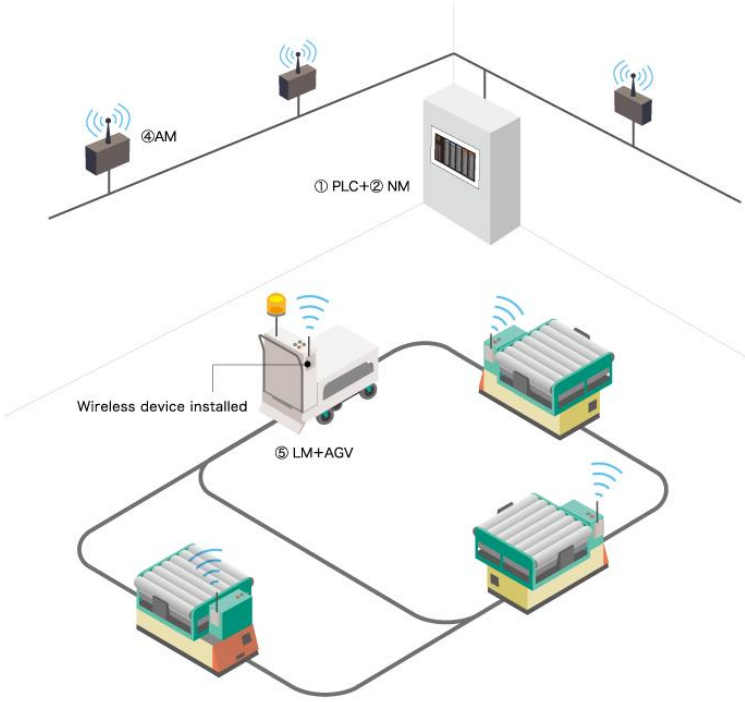


Figura 2.31 – Exemplo de instalação WiFi com AGV.

Outras utilizações que esta rede permite ter neste tipo de robôs é a atualização de software e comunicações com sistemas centrais de visualização em tempo real. É um sistema de comunicação que facilmente se pode ampliar devido ao seu baixo custo e flexibilidade de configuração. Ao colocar mais pontos de acesso, o sinal deste tipo de rede pode ser amplificado, aumentando assim a capacidade de comunicação.

A Ethernet é utilizada essencialmente para fazer a ligação entre os pontos de acesso, e sistemas externos, como por exemplo a central de controlo e mesmo PLCs que estejam a controlar os equipamentos de carga e descarga dos robôs.

2.6 Arquiteturas de hardware e software

Nesta secção são referidos os conceitos de hardware e software mais utilizados atualmente na robótica móvel.

2.6.1 Arquitetura de hardware

As arquiteturas de hardware nos robôs são constituídas por diversos elementos, desde baterias, controladores, sensores e diversos atuadores. Na Figura 2.32 pode ser vista a arquitetura de um robô de garfos.

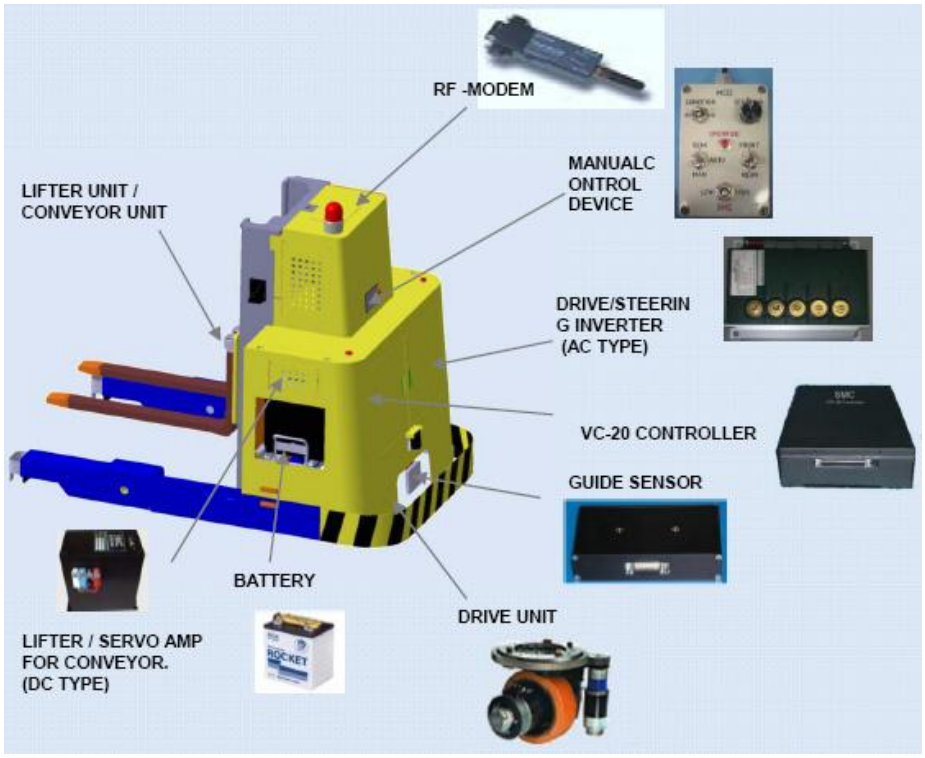


Figura 2.32 – Exemplo de arquitetura de hardware de um robô com garfos.

2.6.1.1 Baterias

O elemento essencial de um robô móvel sem o qual o mesmo não consegue funcionar é a bateria. Por regra o robô dispõe de baterias recarregáveis, que podem ter várias tensões de funcionamento. Cada vez mais o carregamento de baterias destes equipamentos deixa de ter a intervenção humana. Nos primeiros robôs era necessário alguém trocar as baterias aos robôs, ou então colocar o mesmo à carga. Com a evolução destes equipamentos, foram criados meios de carga autónomos. Por exemplo, sistemas de troca automática, sistemas de carregamento físico no ponto onde o robô está mais tempo parado em que o robô move os pinos de carregamento que fazem contacto com pinos de carga fixos no solo e por último os sistemas de carregamento por indução (Figura 2.33).

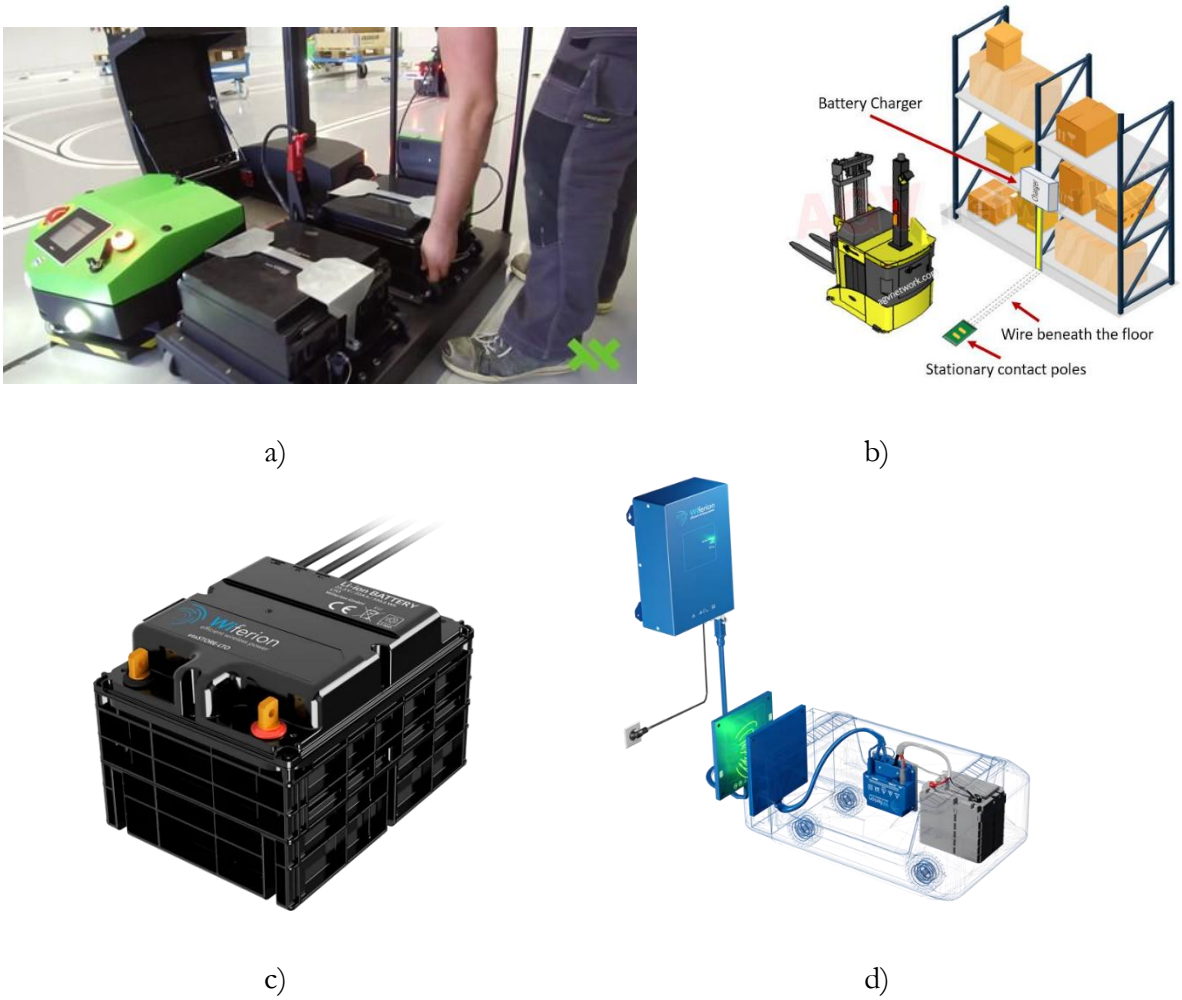


Figura 2.33 – Baterias: a) Troca de bateria manual; b) Sistema de carga por pinos; c) Bateria de um AGV d) Sistema de carga por indução.

2.6.1.2 Controlador Principal

O controlador principal (Figura 2.34) de um robô móvel é constituído por um processador que pode ser um PLC, microcontrolador ou mesmo um PC. Este equipamento é o responsável por toda a gestão de hardware e com a capacidade de gerir as comunicações e controlar o movimento do robô. Este equipamento pode ser ajudado por outros microcontroladores auxiliares que controlam elementos do robô não sobrecarregando o processador principal. Normalmente comunicam entre si recorrendo a protocolos de comunicação como por exemplo o CAN. A este equipamento estão também ligados todos os sensores (referidos no Subcapítulo 2.4) e os atuadores apresentados no Subcapítulo 2.6.1.3.

Figura 2.34 – Exemplo de controlador principal [Kinco AK800M].

2.6.1.3 Atuadores

Os principais atuadores dos robôs moveis são os motores, ou sistemas de controlo de tração, que são compostos efetivamente pelo controlador e pelos motores. Os controladores são os responsáveis por controlar velocidades, acelerações e desacelerações dos motores que se encontram ligados a si. Normalmente estes equipamentos na sua maioria são controladores de servomotores. Os servomotores são dotados de uma maior precisão e velocidade que os motores DC. Estes controladores têm normalmente o *encoder* do motor ligados a si, permitindo assim um controlo mais eficaz (Figura 2.35).

Figura 2.35 – Exemplo de servo controladores e motores [cyber® iTAS® system da WITTENTSTEIN].

Além destes controladores existem também os controladores dedicados para cada tipo de robô, por exemplo os controladores dos garfos, tapetes de rolos, mesas tipo tesoura, etc.

2.6.1.4 Interface Homem-Máquina

Todos os robôs moveis dispõem de uma interface para o utilizador poder comandar e visualizar o estado do robô. Um dos elementos obrigatórios nesta interface é a paragem de emergência que permite numa emergência parar o robô manualmente.

Existem algumas interfaces que se resumem a botões e sinalizadores que permitem ver o estado do robô e controlar como o parar e colocar em modo automático.

Algumas interfaces mais recentes já estão equipadas com consolas gráficas que permitem ter uma visualização e controlo mais detalhados (Figura 2.36).

Figura 2.36 – Interface Homem-Máquina de um AGV.

2.6.1.5 Equipamentos de sinalização obrigatórios

Com a evolução e aumento da utilização destes equipamentos as normas começam a exigir mais equipamentos de segurança e sinalização para alertar da aproximação do robô ao ser humano. Os equipamentos que são obrigatórios neste momento são o *buzzer* de sinalização que tem de ser bem audível e uma luz azul que acende no sentido do deslocamento do robô. Normalmente esta sinalização visual está montada na frente do robô (Figura 2.37).

a) b)

Figura 2.37 – Equipamentos de sinalização obrigatória: a) Sinalizador acústico; b) luz de cor azul obrigatória.

2.6.2 Arquitetura de software

No que se refere à arquitetura de software existem várias componentes que a constituem sendo o sistema operativo a componente base, sobre a qual existem várias camadas de software, cada uma dedicada a uma determinada funcionalidade. Em conjunto, estas camadas permitem construir a arquitetura de controlo de software, que pode ter três tipos de funcionamento, deliberativa, reativa e híbrida.

2.6.2.1 Componentes de arquitetura de software

A principal componente da arquitetura de software é o sistema operativo. O sistema operativo nos robôs é normalmente de tempo real (RTOS). Nos robôs são escolhidos estes tipos de sistemas de forma a garantir que tarefas críticas de controlo

e decisão são executadas de forma eficaz de modo a não existirem falhas que possam causar problemas de funcionamento e ou mesmo acidentes.

Tal como referido anteriormente existem diversas camadas de software dedicadas a uma determinada função. Estas camadas utilizam os *inputs* necessários ao seu funcionamento e geram os seus *outputs*, que utilizados pelo sistema operativo podem gerar ações em outras camadas.

A primeira camada a referir é a camada de segurança, que é a responsável por analisar todos os elementos de segurança e conforme o estado das suas entradas pode tomar decisões de reduzir velocidades na deteção de obstáculos na área circundante ou mesmo parar o robô no caso de prever que o robô vai colidir.

Outra camada muito importante é a camada de sensorização e perceção. Esta camada é responsável por analisar o estado de todos os sensores do robô. Com os dados dos sensores, esta camada deve ter a capacidade de interpretar a área circundante do robô e transmitir para outras camadas as ações que devem tomar com os dados recolhidos.

A camada de navegação também utiliza os dados criados pela camada de sensorização. Nesta camada podem ser executados diversos algoritmos, já estudados e desenvolvidos, que permitem gerir o mapeamento e a localização do robô em tempo real, sendo um dos conceitos mais usados o do SLAM (*Simultaneous Localization and Mapping*). O SLAM é um algoritmo que, com informação, por exemplo, dos sensores LiDAR e de ultrassons, permite ao robô saber de forma precisa o local onde se encontra na área de produção e em simultâneo ir atualizando e melhorando o mapa da área de trabalho.

Uma das camadas principais do robô é a que gere todos os atuadores de forma que o robô se consiga mover numa determinada trajetória entre dois pontos. Esta camada é a camada de controlo de movimento. É responsável por receber os dados de todas as camadas já referidas e com estes definir, através dos algoritmos de controlo de movimento (normalmente PID), e dos algoritmos associados à cinemática do robô, definir as melhores estratégias de velocidade, aceleração e comportamentos dos atuadores.

A camada de comunicação é responsável por atualizar os dados externos. Esta camada normalmente é a responsável por informar elementos externos de estados do robô, e comunicar ordens com os pontos de trabalho do robô, por exemplo de carga e descarga.

Por fim uma camada que também existe nos robôs é a camada de interface com o utilizador. Esta camada controla as indicações de estado do robô localmente assim como lê os equipamentos responsáveis por comandar localmente o robô. Normalmente esta camada recorre a consolas e botões, e com as últimas evoluções, pode incluir uma interface web à qual se pode aceder externamente através de um *tablet* ou um simples telemóvel.

Com as últimas evoluções da inteligência artificial [2] alguns fabricantes deste tipo de equipamento começam a aplicar uma camada de controlo utilizando inteligência artificial. Esta camada é dotada de algoritmos que permitem otimizar de forma inteligente traçados, rotas e até tomar decisões no caso de obstáculos impeditivos de atingir o destino, tal como encontrar rotas alternativas, caso existam.

2.6.2.2 Tipos de arquitetura de software

Com as componentes referidas na secção anterior, podem ser contruídos vários tipos de arquitetura. Existem três tipos principais de arquitetura de software que os construtores de robôs utilizam. São elas a deliberativa, a reativa e a híbrida.

A arquitetura deliberativa baseia-se num modo de funcionamento que é mais "ponderado", pois é um conceito que analisa os dados recolhidos, pondera a melhor decisão a tomar, planeando tarefas e decisões antecipadamente, de forma ao robô ter um comportamento mais eficaz e a não existirem decisões que podem gerar comportamentos bruscos do robô. Quando se opta por este tipo de arquitetura o nível de processamento é bastante elevado, mas o maior processamento corresponde a maior fiabilidade no comportamento do robô. As principais características desta arquitetura são a capacidade de pré-planeamento, a capacidade de sequenciar ações, e a capacidade de gerar mapas, memórias e planeamento.

A arquitetura reativa reage sempre a uma determinada perceção. Com esta arquitetura o robô toma decisões momentâneas não tendo em conta outros fatores que podem melhorar a eficácia e gestão do robô. Por exemplo, se for detetado um obstáculo para imediatamente, não verificando antecipadamente se pode ser calculada uma trajetória de forma a evitar essa paragem. Esta é uma arquitetura que é bastante simples, não necessitando de muita capacidade de processamento. As principais características desta arquitetura são a capacidade perceção-ação, capacidades sensoriais e reação imediata às mesmas.

A arquitetura híbrida é uma arquitetura que junta o melhor das duas anteriores podendo assim criar uma arquitetura que analise os dados e tome decisões rápidas, mas sem estar necessariamente a analisar todos os caminhos possíveis, e não parando por não analisar antecipadamente.

2.6.2.3 Plataformas de processamento

Após uma análise através de pesquisas realizadas, as duas plataformas de processamento mais utilizadas na robótica movel são o ROS (*Robot Operating System*) e a plataforma NVidia.

A plataforma ROS é uma plataforma de software que dispõe de diversas ferramentas, bibliotecas e ficheiros *online*, que podem ser desenvolvidas por qualquer membro e colocadas disponíveis para outros utilizarem. Por exemplo, um fabricante de um determinado equipamento pode criar a sua biblioteca e os construtores que usarem esse equipamento, em vez de desenvolverem tudo por si, podem assim utilizar as bibliotecas do fabricante no seu robô. Esta plataforma permite assim um construtor

de robôs utilizar diversos sensores e algoritmos de vários fabricantes e interligá-los, não sendo necessário desenvolver todo o código individualmente.

Uma outra plataforma de hardware de processamento é a NVidia. Esta marca tem equipamentos bastante robustos e utilizados a nível industrial, sendo a família Jetson a mais conceituada. Por norma estes equipamentos têm uma capacidade de processamento elevada, capazes de executar o processamento de várias tarefas nos robôs móveis, tais como a visão e a inteligência artificial. Outra das vantagens desta plataforma é que este equipamento já dispõe de bibliotecas dedicadas à robótica móvel, sendo que são mais complexas do que as do sistema ROS, mas segundo a pesquisa realizada, são mais robustas para o nível da robótica industrial.

No início da construção do robô um construtor tem de optar pelo sistema de processamento que vai utilizar, sendo possível optar por hardware NVidia e em simultâneo usar as bibliotecas ROS e não as da NVidia.

2.7 Novas tendências

No que se refere à evolução da robótica e da indústria, é fundamental o caminho que a inteligência artificial tem tomado nos últimos tempos. A robótica móvel não é exceção à tendência geral e continuará sempre a evoluir tendo como um dos principais campos de evolução a gestão da inteligência dos robôs.

Sendo assim, existem diversas áreas da robótica móvel com elevado potencial de evolução com a aplicação de inteligência artificial recorrendo a redes neuronais que permitam aos robôs móveis tornarem-se mais eficazes e rentáveis. Com esta tecnologia pode também ser melhorada a tecnologia SLAM para mapeamento e localização mais eficazes. Pode também ser melhorado o guiamento por visão artificial, tornando mais eficaz o seguimento de trajetórias num determinado percurso. Outra das áreas que já foi referida e que pode beneficiar desta tecnologia é o uso de braços robóticos colaborativos que permitam uma melhor gestão de cargas entre postos de trabalho. Podem ainda ser melhorados os sistemas de energia, permitindo aos robôs fazer uma melhor gestão da energia, aumentando a autonomia. Por fim, uma das razões principais que motivou o autor a realizar este trabalho foi a aplicação do conceito de enxame em robôs móveis de forma a partilharem informação e criarem uma equipa de protótipos de robô que trabalham com objetivos comuns, aumentando a rapidez, eficiência, flexibilidade e rentabilidade destes equipamentos.

3 ARQUITETURA DA PLATAFORMA DE DESENVOLVIMENTO

Neste capítulo são descritos os diversos elementos do projeto, apresentadas as funcionalidades aplicadas a cada um deles e o porquê da seleção dos equipamentos utilizados. É descrita também a interligação e funcionamento de cada um dos elementos e as comunicações entre si. São apresentadas cada uma das tarefas e funções de cada elemento ativo, protótipo da fábrica, Central e protótipos de robô.

3.1 Elementos do projeto

Neste projeto foram desenvolvidos três elementos-chave. O primeiro foi o projeto, impressão e montagem de três protótipos de robô. O segundo foi a programação de um computador que permita servir de Central onde o utilizador pode localmente ou externamente, monitorizar e controlar toda a frota, e executar testes de novas funcionalidades criadas por si. Por fim, o último elemento é um protótipo de uma fábrica onde os protótipos de robô podem executar os diversos testes, como se estivessem numa área de produção, e assim o utilizador avaliar os testes em tempo real.

3.1.1 Protótipo de robô

Foram projetados, dimensionados, programados e testados três protótipos de robô (Figura 3.1) para que assim se possa testar o conceito de funcionamento de enxame de robôs. Como já foi referido anteriormente, é um dos caminhos que o autor pretende investigar e melhorar.

Estes protótipos estão dotados de diversos sensores e atuadores que permitem simular funcionalidades de robôs industriais, tais como, sensores ultrassónicos, sensor seguidor de linhas, sensores óticos de deteção de obstáculos, interface com o utilizador e elementos de sinalização visuais e acústicos. Outra das suas características é a capacidade de movimentos omnidirecionais. Se porventura, o utilizador pretender utilizar rodas convencionais, terá de desenvolver os algoritmos de controlo das mesmas, ou então, desenvolver um sistema que se adeque a essa realidade.

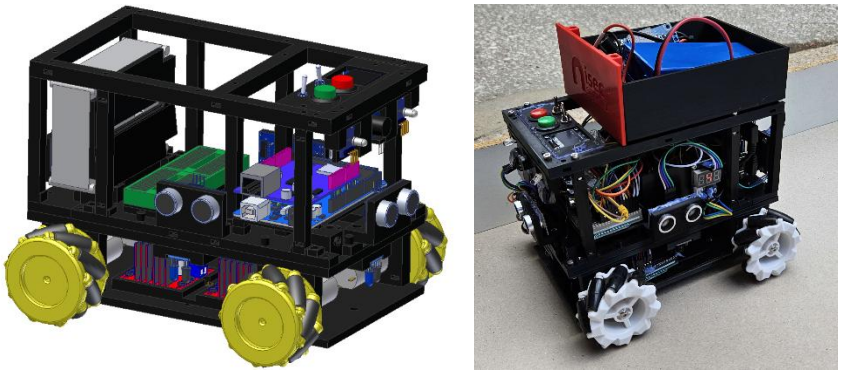


Figura 3.1 – Protótipo de robô.

3.1.2 Computador Central

Para servir de computador Central (ou "Central"), foi utilizado um Raspberry PI 4 [3], de 8 GB de RAM (Figura 3.2). Optou-se por este sistema devido à maioria dos construtores recorrerem ao sistema operativo Linux para o desenvolvimento das plataformas de robótica móvel, e principalmente devido aos custos de hardware e software mais competitivos. O autor utilizou um monitor portátil, que permite realizar localmente ensaios e alteração ao código de programação desenvolvido. No entanto, a aplicação pode ser acedida por equipamentos externos desde que ligados à rede de comunicações criada para este projeto.

Figura 3.2 – Hardware da Central.

3.1.3 Protótipo da fábrica

Para simular e testar tarefas nos protótipos de robô e verificarmos os seus resultados em tempo real foi criado um protótipo de uma fábrica (Figura 3.4). Este protótipo tem uma dimensão de dois metros e meio por dois metros e inclui três pontos de estacionamento dos protótipos e quatro pontos de trabalho a simular linhas de produção, incluindo pontos de origem e destino.

Figura 3.3 – Localização de linhas de produção e estacionamento no protótipo da fábrica.

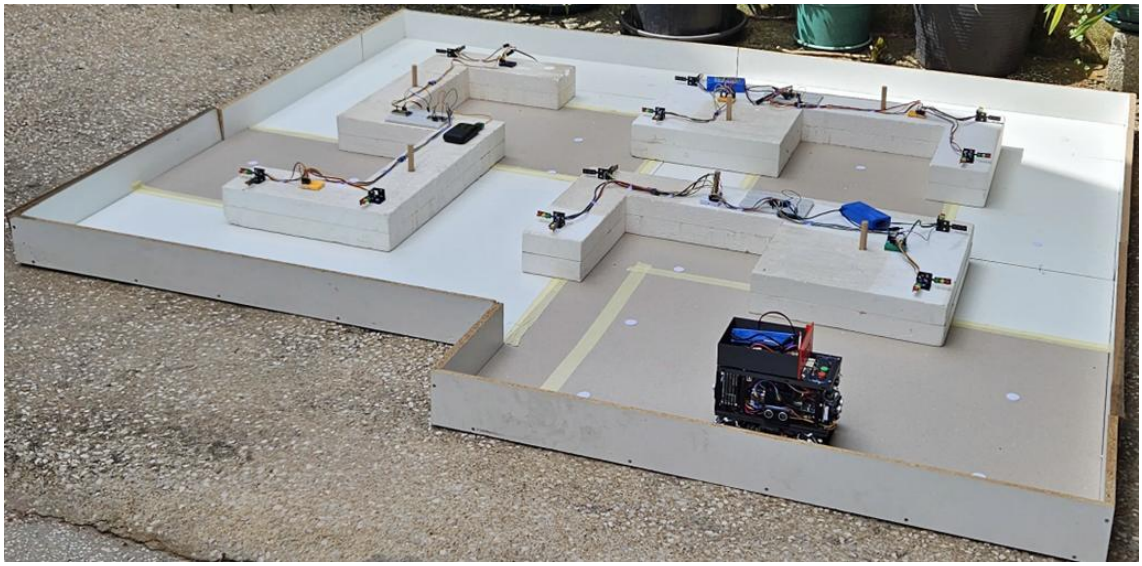


Figura 3.4 – Protótipo da fábrica.

Na Figura 3.5 são identificados com os círculos verdes os nós (pontos de decisão) e com as linhas laranjas os diversos troços (trajeto entre dois nós) do protótipo da fábrica.

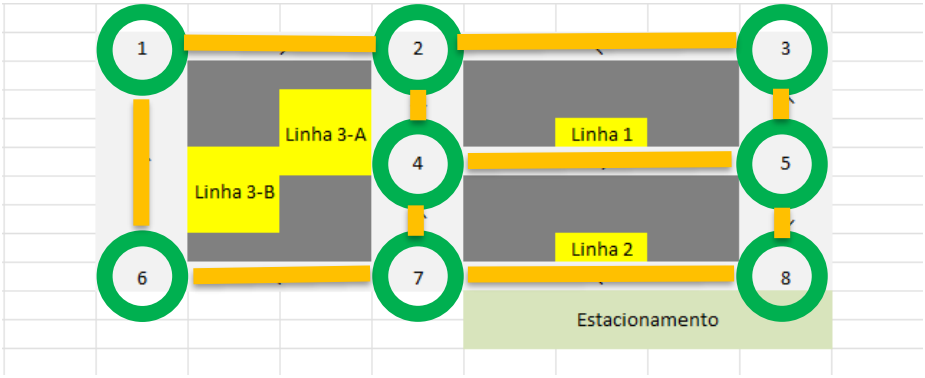


Figura 3.5 – Identificação de troços e nós no protótipo da fábrica.

3.1.4 Descrição total do sistema

Tal como referido existem então três protótipos de robô, um computador Central e um protótipo de uma fábrica. Para além destes elementos principais existem diversos componentes internos e externos que permitem a integração de todos os elementos.

Para interligar todos os elementos do projeto existe um router que faz a ligação de todos os equipamentos por WiFi. Os protótipos de robô comunicam com a Central através do seu Raspberry PI individual. A Central está ligada ao mesmo router por WiFi. Em cada uma das zonas de trabalho do protótipo da fábrica foi colocado um ESP32-WROOM, que é responsável por receber os dados da Central e mostrar visualmente o estado de cada troço adjacente a si e efetuar a simulação de pedidos de tarefas. Toda esta estrutura é apresentada na Figura 3.6.

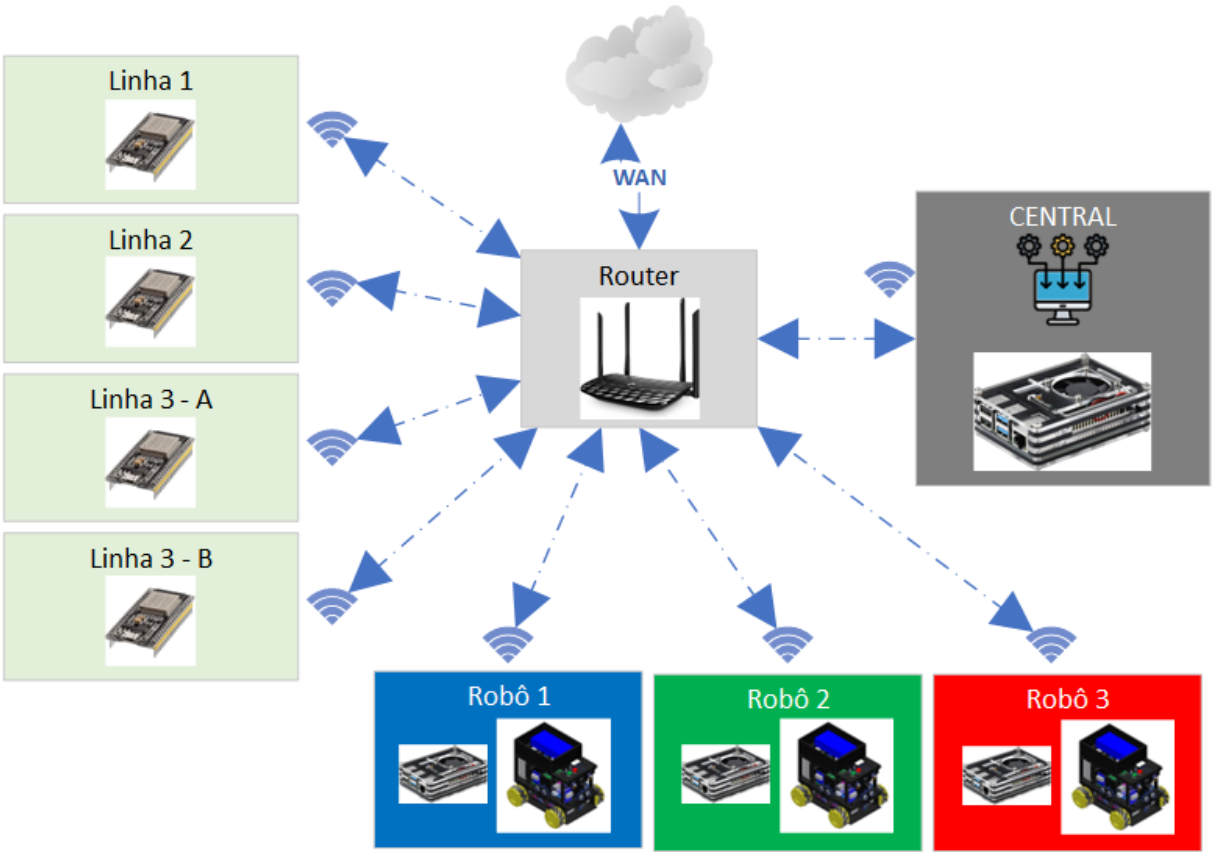


Figura 3.6 – Elementos do projeto e tipos de comunicações.

3.2 Funções implementadas

Apresentam-se nesta secção as funcionalidades implementadas em cada um dos componentes do projeto, identificados na secção anterior.

Nos protótipos de robô:

- Capacidade de testar e utilizar os vários sensores e atuadores existentes em robôs;
- Algoritmos de programação dinâmica:
    - Capacidade de auto localização no arranque sem ponto de referência;
    - Capacidade de encontrar autonomamente a melhor trajetória entre dois pontos;
    - Capacidade de localizar bloqueios de troços e partilhar essa informação com os outros protótipos de robô e Central;
- Capacidade de demonstração de vinte e dois dos movimentos elementares possíveis com as rodas omnidirecionais "Mecanum" [4].

Na Central:

- Servidor Web que permite visualizar e controlar todos os elementos do projeto;
- Capacidade de testar e despoletar as capacidades dos protótipos de robô;
- Algoritmos de programação dinâmica:
    - Algoritmo que consegue reconfigurar os sentidos dos diversos troços, de forma que nenhuma linha fique inacessível;
- Gestão do protótipo da fábrica de forma a ser fácil a visualização de todos os troços e receber pedidos das diversas zonas de trabalho.

Protótipo da fábrica:

- Permitir visualizar o estado de todos os troços em tempo real;
- Permitir efetuar pedidos de cada uma das linhas de produção.

3.3 Protótipos de Robô

Nesta secção são apresentadas as opções de hardware, comunicações e software escolhidas e as suas motivações. São apresentados também os fluxogramas das funções e tarefas implementadas.

3.3.1 Opções de Hardware

Os protótipos foram evoluindo ao longo do projeto. Estes começaram como uma simples ideia da aplicação de inteligência artificial em robôs móveis, mas com a evolução decidiu-se enveredar pelo caminho do desenvolvimento desta plataforma de testes.

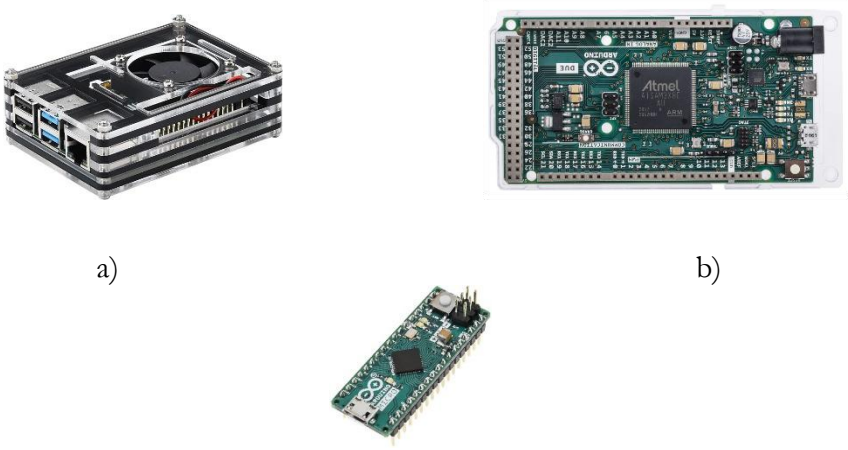
Os protótipos dispõem de uma bateria de 12V com capacidade de 6800 mAh para a alimentação. Esta bateria alimenta o circuito dos motores a 12V e um conversor DC-DC Step-Down. O modelo escolhido foi o LM2596, devido à sua flexibilidade de ajuste e capacidade de corrente. Este conversor transforma os 12 V em 5 V para alimentar os controladores e sensores existentes (Figura 3.7).

a) b)

Figura 3.7 – Equipamentos de Energia: a) Bateria de 12V; b) Conversor DC-DC.

Na parte interior do protótipo de robô encontram-se os três controladores (Figura 3.8).

O Raspberry PI, instalado na parte traseira, tem a função de gerir todo o protótipo e comunicar com a Central para enviar o estado do protótipo e receber ordens e pedidos do sistema Central. O Arduino Due [5] pode ser considerado um escravo do Raspberry PI, sendo que uma das evoluções possíveis seria passar a utilizar o mesmo como escravo Modbus TCP, visto que a comunicação entre eles é por Ethernet. Como todo o projeto começou a ser desenvolvido no Arduino este tem a potencialidade de mostrar em modo manual, os vinte e dois movimentos elementares das rodas omnidirecionais. O último controlador existente pode ser encontrado na parte inferior do protótipo. Este controlador é responsável pela deteção de elementos RFID através da carta leitora existente também na parte inferior do protótipo, e indicar ao controlador principal o que foi lido (Figura 3.9).



a) b)

c)

Figura 3.8 – Controladores: a) Controlo e Comunicação – Raspberry PI; b) Sensores e Atuadores – Arduino Due; c) RFID – Arduino Micro.

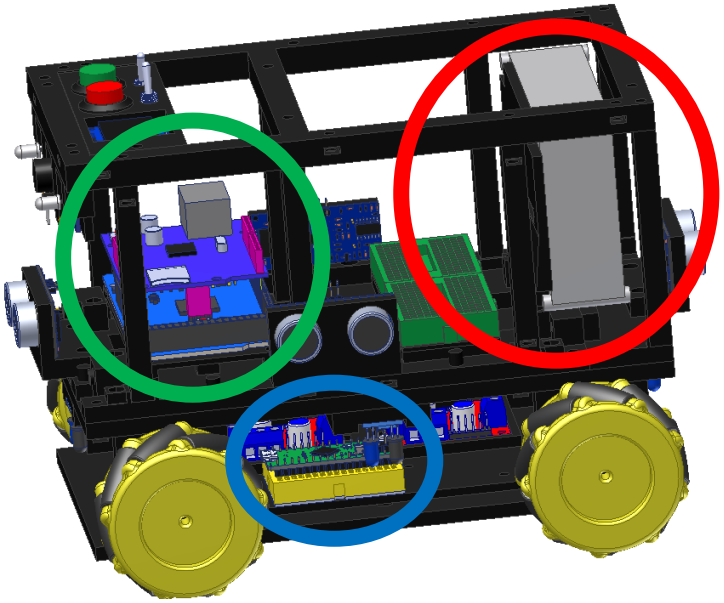


Figura 3.9 – Localização dos Controladores no protótipo de robô (Vermelho – Raspberry PI, Verde – Arduino Due e Azul – Arduino Micro).

Existe um painel de interface equipado com um interruptor de duas posições que permite ligar e desligar o protótipo, um interruptor de 3 posições que define o modo de funcionamento (manual, automático e por visão artificial), um botão verde que permite colocar o protótipo de robô em funcionamento e um botão vermelho que permite parar o protótipo (Figura 3.10). Também existe um pequeno display que permite visualizar o estado do protótipo de robô.

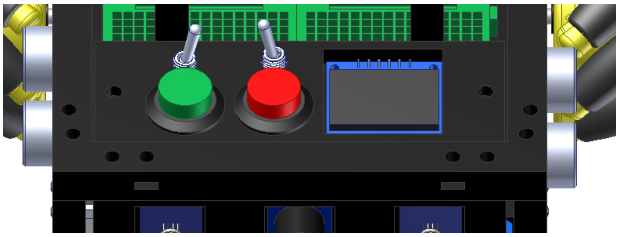


Figura 3.10 – Interface Homem-Máquina.

Os protótipos na frente estão equipados com um sistema de sinalização luminosa e acústica, sendo constituído por um besouro e por dois LED RGB. A sinalização luminosa apresenta três estados possíveis sendo estes apresentados de forma intermitente. O estado verde indica que o protótipo de robô se encontra em funcionamento automático e em movimento. O estado azul indica ao utilizador que o protótipo se encontra em modo manual e em movimento. O estado vermelho indica a existência de uma falha (Figura 3.11).

O besouro, por omissão, quando se encontra ativo de forma intermitente, indica que o protótipo de robô se encontra em movimento.

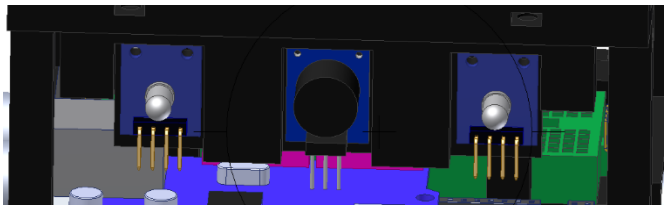
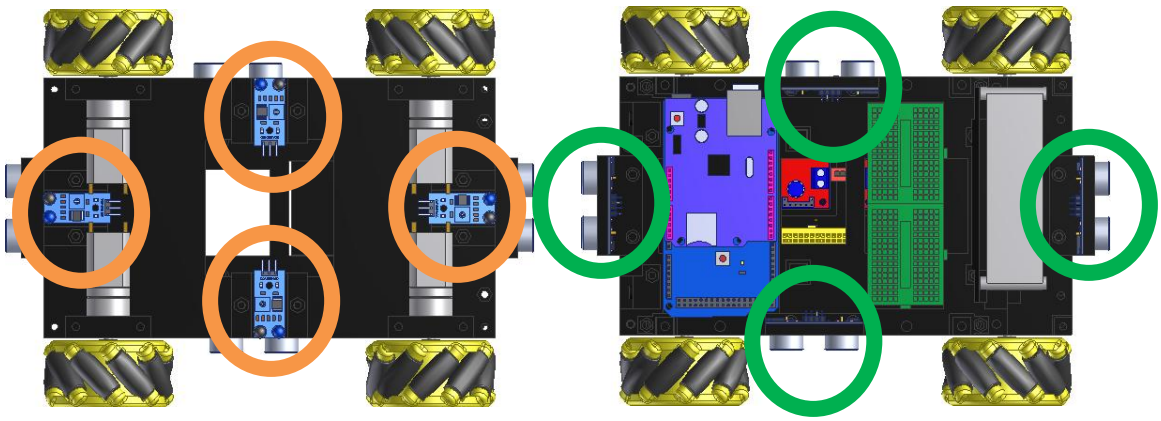


Figura 3.11 – Elementos de sinalização luminosa e acústica.

Para localização e possível mapeamento, os protótipos de robô dispõem de quatro sensores de ultrassons, um em cada face lateral do protótipo, para detetar e mapear os protótipos em seu redor, podendo ainda ser utilizados para evitar colisões.

Também em cada uma das faces do protótipo de robô existe um sensor ótico de deteção de obstáculos (Figura 3.12).



a) b)

Figura 3.12 – Localização de sensores: a) Sensores óticos(vista da face inferior); b) Sensores ultrassónicos (vista da face superior).

Na parte de baixo encontra-se o sensor seguidor de linhas, para testes de funcionalidades relacionadas com o seguimento de linhas por AGV (Figura 3.13).

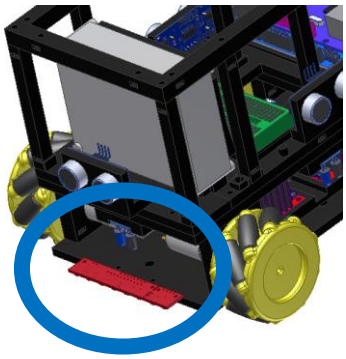


Figura 3.13 – Localização de Seguidor de Linhas.

Os motores escolhidos são motores de 12V, com trinta rotações por minuto, sendo controlados por cartas de controlo PWM do tipo L298N. Estas cartas são alimentadas a 12 V e geram impulsos PWM que permitem modificar a velocidade dos motores (ver Figura 3.14).

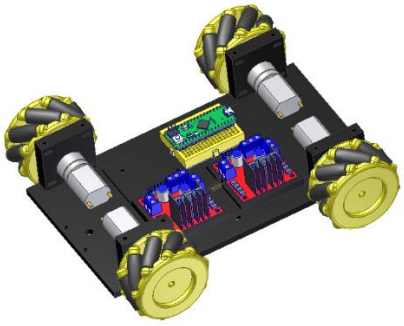


Figura 3.14 – Localização dos motores e seus controladores.

A cada um dos motores está acoplada uma roda do tipo "Mecanum". O funcionamento destas rodas é explicado na Subsecção 3.3.3.2.

O sistema de leitura RFID é constituído por um modulo PN532 NFC que está localizado na parte inferior (Figura 3.15), usado para ler os elementos de localização desta tecnologia que se encontram em pontos estratégicos do protótipo da fábrica. Este módulo utiliza o protocolo I2C para comunicar com o Arduino Micro.

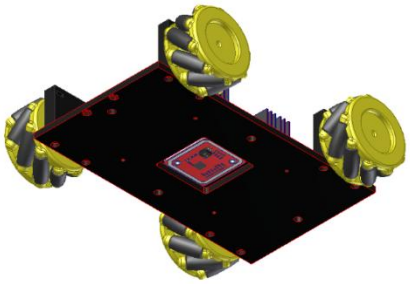


Figura 3.15 – Localização do leitor RFID.

Os sensores, atuadores e interface referidos estão ligados diretamente ao Arduino Due.

Os controladores e as opções tomadas foram evoluindo ao longo do projeto, apresentando-se nos próximos parágrafos as motivações das suas escolhas.

No início, o objetivo era controlar todo o sistema com um Arduino Due. No entanto, como surgiu a possibilidade de outro colega desenvolver um projeto de controlo por visão artificial nestes protótipos, decidiu-se então utilizar um Raspberry PI para poder realizar o processamento de imagem. Com a montagem deste elemento, o projeto passou a ter o Raspberry PI como controlador principal, enquanto o Arduino ficou responsável por controlar todos os atuadores conforme as definições do controlador principal e enviar para o mesmo, ciclicamente, o estado de todos os sensores.

Outra das decisões que levou a colocar o Arduino Micro foi que como o Raspberry PI e o Arduino Due recebem os sinais com uma tensão de 3,3 V e a carta RFID só funciona a 5V, decidiu-se então colocar, tal como nos robôs reais, um controlador só para verificar as *Tags* e quando se recebe uma leitura de uma *Tag* válida, esta é enviada ao processador de controlo e comunicações (ver Figura 3.16).

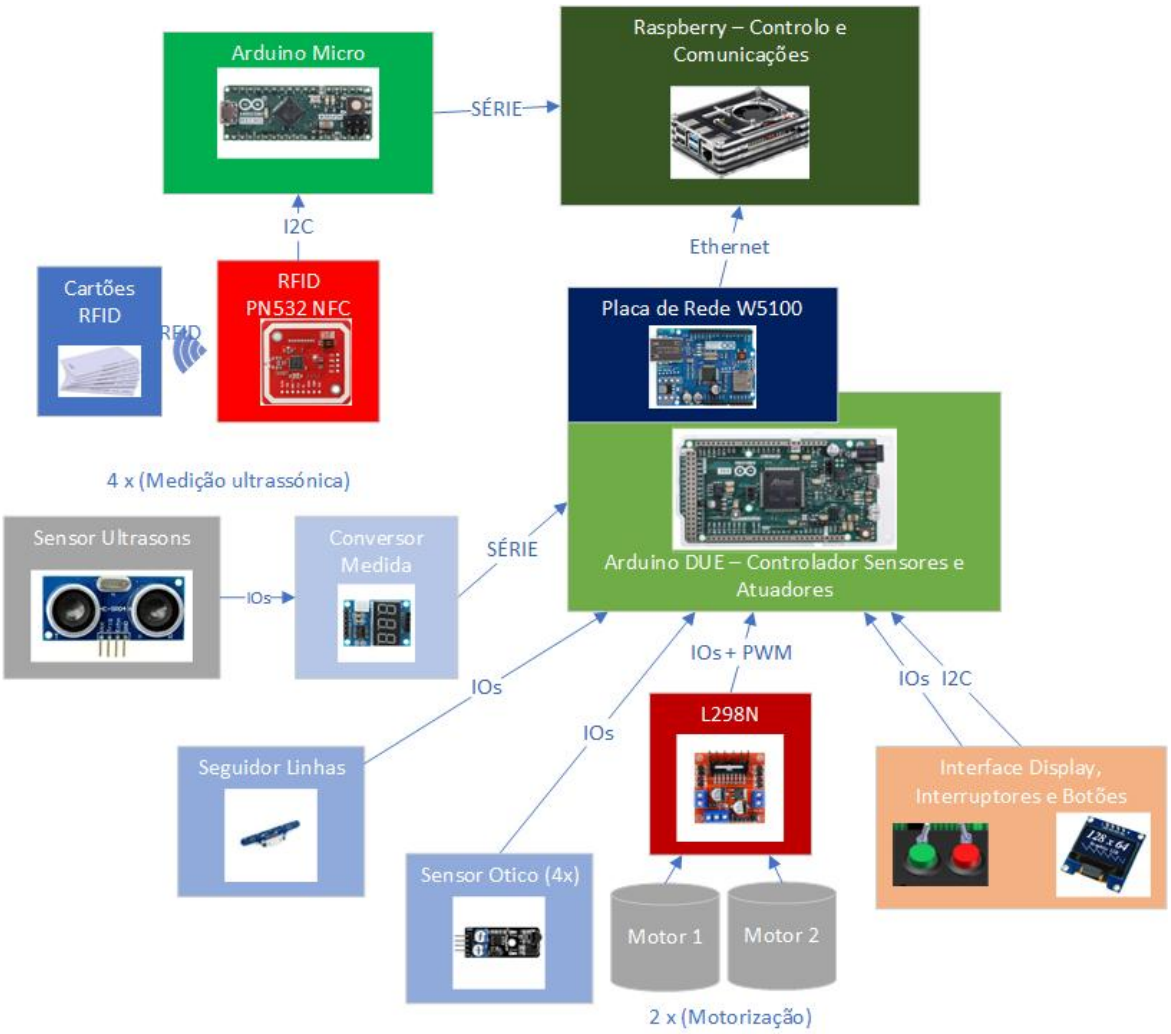


Figura 3.16 – Arquitetura geral de hardware dos protótipos de robô.

Houve um momento em que foi testada uma função de medição de distância no Arduino Due para fazer as medições de distância dos sensores ultrassónicos, mas esta não respondia como previsto, causando alguns atrasos que impediam o AGV de reagir a distâncias muito curtas a tempo de evitar a colisão com algum objeto. Por isso, optou-se pela utilização do conversor de medida, que controla internamente o sensor e envia os dados de medição já convertidos para o Arduino Due. Este, por sua vez, analisa e envia os dados para o controlador principal, que os utiliza de forma a otimizar o controlo do protótipo.

3.3.2 Software e Tarefas

Como foi referido anteriormente, os protótipos de robô dispõem de duas zonas de processamento, uma no Arduino e outra no Raspberry PI. No Arduino é efetuada a leitura e controlo de todos os sensores e atuadores do protótipo. No Raspberry PI são processados todos os algoritmos de programação dinâmica e realizado o controlo do protótipo de robô quando se encontra em modo automático.

O Arduino e o Raspberry PI comunicam entre si através do protocolo Ethernet, sendo que comunicam por websockets [6]. Optou-se por este tipo de comunicação porque com sockets convencionais quando existiam falhas na comunicação não era fácil restabelecer a comunicação sem ter de desligar os servidores e clientes.

O Arduino foi programado para funcionar como sistema operativo em tempo real. Para isso foi utilizada a biblioteca FreeRTOS. Foram programadas diversas funções e tarefas para ter um melhor controlo do protótipo.

O Raspberry PI do protótipo de robô foi programado em Python [7], utilizando multiprocessos e filas para simplificar a programação e permitir que vários processos funcionem em paralelo. Estes processos compartilham os dados com o processo principal, onde são processados, e as decisões necessárias são tomadas sem depender da execução de todos os processos. No fundo, esta programação tem como objetivo receber uma determinada mensagem, processar a sua informação e tomar a decisão mais correta com os dados recebidos.

Começando assim pela arquitetura de software do Raspberry PI, a estrutura de trabalho principal passa por ter uma zona de processamento principal, onde é executado todo o processamento de mensagens recebidas e decisões. Existem dois servidores e dois clientes de websockets, um par para comunicar com o Arduino que é responsável pela comunicação através da porta de rede física. O outro par é responsável por comunicar com a Central através da rede WiFi do Raspberry PI (Figura 3.17).

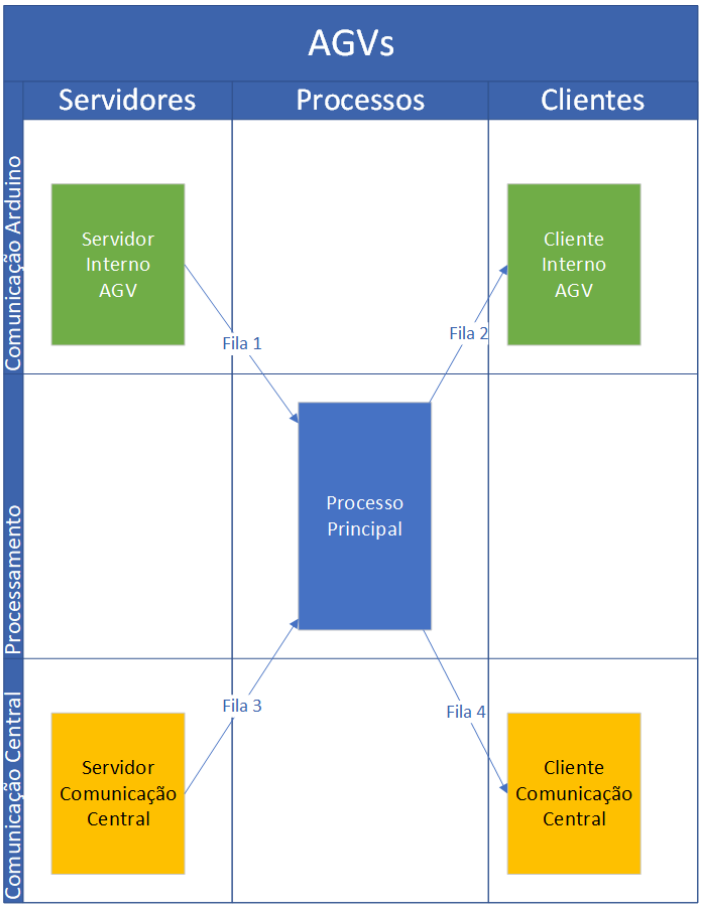


Figura 3.17 – Arquitetura do software principal do Raspberry PI nos protótipos de robô.

Nas próximas subsecções são apresentadas as funções e tarefas programadas no Arduino e no Raspberry PI. As funções são partes de código dedicados a controlar determinados elementos, por exemplo, o controlo de um motor. As tarefas são partes de código do programa, que podem invocar diversas funções e com as mesmas tomar decisões sobre diversos elementos. Por exemplo uma tarefa de controlo de movimentos, é uma tarefa que com a função de leitura de sensores adquire os dados de posicionamento do protótipo e com os dados adquiridos toma as decisões necessárias para evitar colisões, controlando os motores através da função de controlo dos mesmos. As tarefas podem, para além de invocar funções invocar tarefas.

3.3.2.1 Funções existentes no Arduino

3.3.2.1.1 Leitura de sensores de obstáculos

Esta função tem como objetivo a leitura do estado dos quatro sensores de deteção de obstáculos (Figura 3.18).

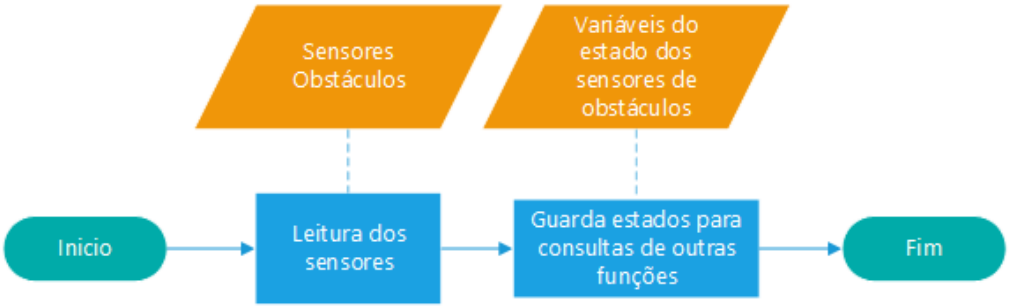


Figura 3.18 – Estrutura da função de leitura dos sensores.

Esta função executa a leitura do estado dos sensores e guarda o mesmo em variáveis internas do programa, para o estado estar disponível em qualquer tarefa que necessite dele.

Esta função é invocada na tarefa de controlo dos movimentos e se algum dos sensores detetar um obstáculo é tomada a decisão mais adequada para o protótipo de robô evitar colisões.

3.3.2.1.2 Leitura de sensores de distância

As distâncias do protótipo de robô aos objetos da área circundante são adquiridas através de uma função (Figura 3.19) que recebe uma mensagem com a informação através de porta de comunicação série.

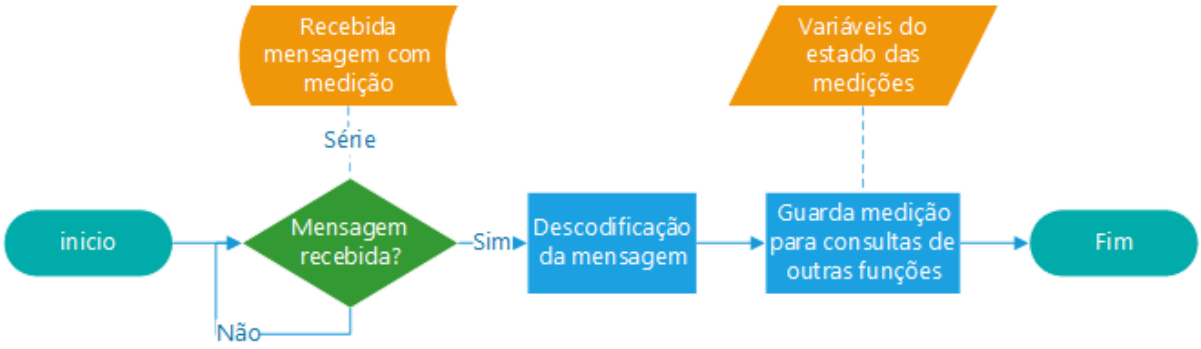


Figura 3.19 – Estrutura da função de leitura de sensores de distância.

Quando é recebida uma mensagem com a distância medida por um determinado sensor, a função executa a descodificação da medida recebida e guarda o valor numa variável interna que pode ser lida por outras tarefas.

Os valores recebidos de cada um dos sensores é lido e utilizado pela tarefa de controlo de movimento, que decide a partir dos valores recebidos qual a melhor ação a tomar para guiamento e evitar colisões.

3.3.2.1.3 Leitura dos botões de comando

Com esta função é lido o estado dos botões existentes no painel de controlo que existe nos protótipos de robô (Figura 3.20).

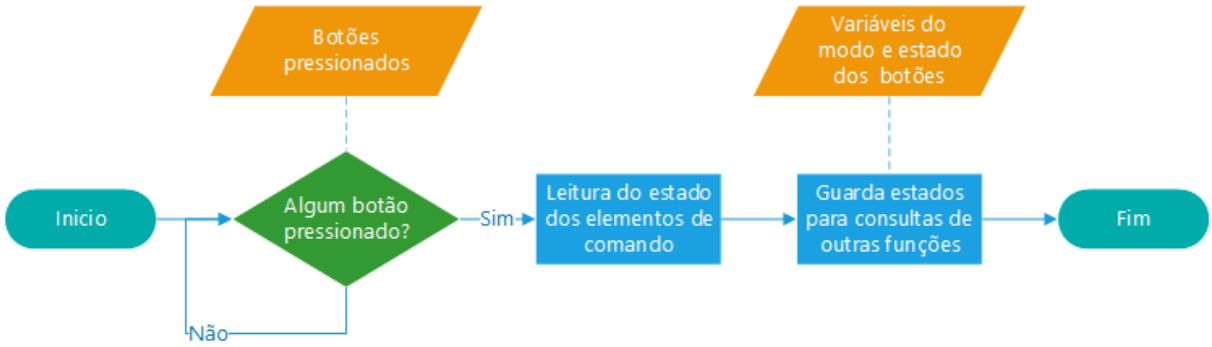


Figura 3.20 – Estrutura da função de leitura dos botões.

A função lê o modo de funcionamento do protótipo de robô e os botões que permitem iniciar e parar o protótipo localmente. É responsável por definir o modo de funcionamento selecionado, manual ou automático, e por verificar se o protótipo de robô está em ciclo de movimento e parar o mesmo caso este esteja em execução.

Ao mudar algum destes valores, o protótipo de robô reage e pode iniciar ou parar movimentos. A maioria das tarefas existentes no Arduino utilizam os sinais desta função para saberem se continuam ou não a ser executadas.

3.3.2.1.4 Controlo de motores

Esta função tem como objetivo o controlo dos motores, controlando o sentido e a velocidade dos mesmos (Figura 3.21).

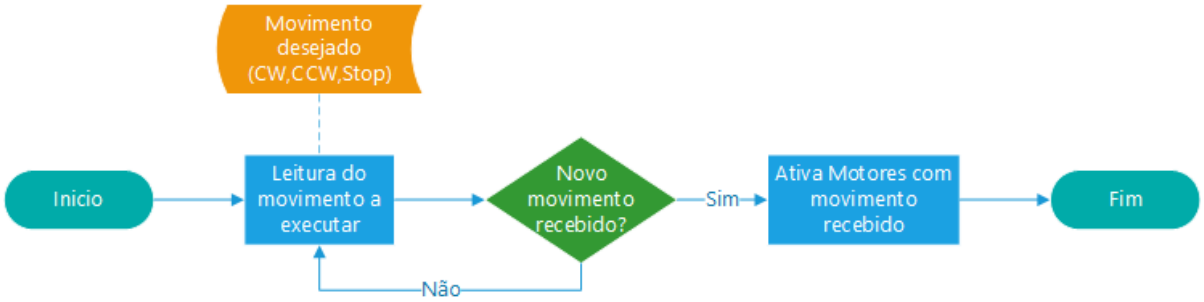


Figura 3.21 – Estrutura da função de controlo dos motores.

A função tem como parâmetros de entrada o sentido desejado (avanço ou recuo) e com estes dados a função controla o sentido do motor. O controlador de sensores e atuadores envia o sentido e o valor de PWM de controlo da velocidade do motor para o modulo de controlo dos motores.

A função é chamada e controlada pela tarefa de controlo de movimento, permitindo assim controlar cada um dos motores.

3.3.2.1.5 Controlo das sinalizações

Com esta função são controlados os diversos equipamentos de sinalização e alerta que o protótipo de robô dispõe, são eles a sinalização luminosa e acústica (Figura 3.22).

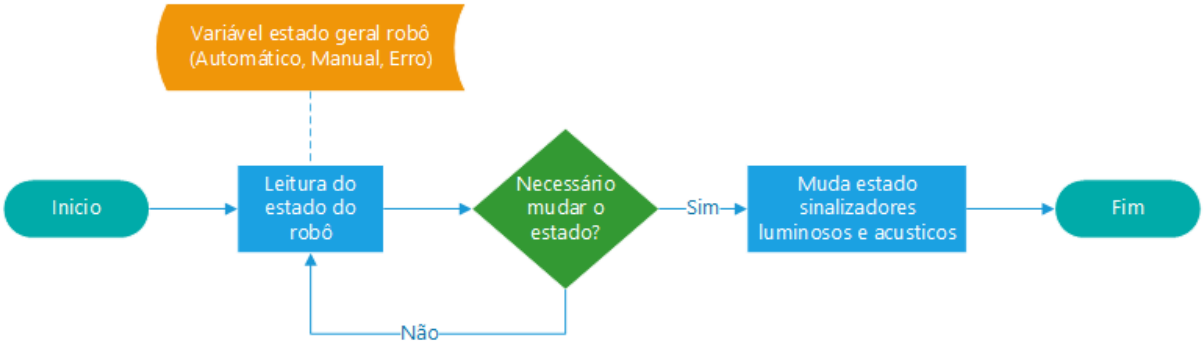


Figura 3.22 – Estrutura da função de controlo das sinalizações.

A função é responsável por ler as variáveis do sistema e, de acordo com o estado, controlar os elementos de sinalização. Por exemplo, quando o protótipo de robô se desloca sem erros, emite um sinal sonoro e luminoso (LED RGB verde) a cada três segundos, indicando que se encontra em movimento. Se encontrar um obstáculo emite um sinal sonoro a cada segundo e os LED piscam com a mesma frequência. Para alertar que está o protótipo de robô está em erro, por ter detetado um obstáculo os LED apresentam a cor vermelha.

3.3.2.1.6 Controlo do Display

Esta função é responsável por controlar o display que se encontra no painel de controlo do protótipo de robô, e por dar indicações do estado de funcionamento do protótipo de robô (Figura 3.23).

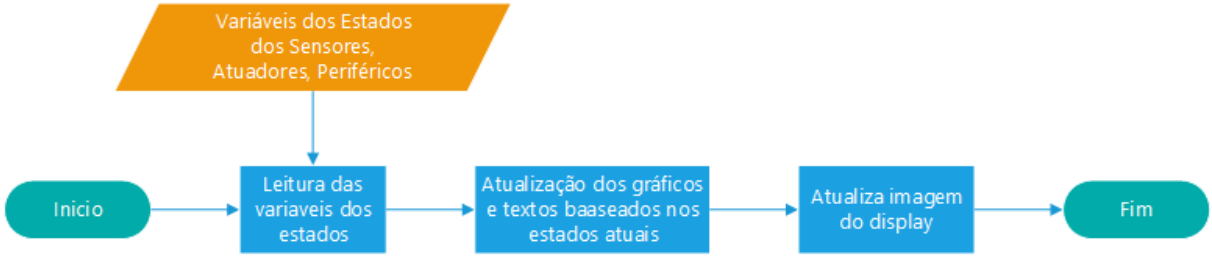


Figura 3.23 – Estrutura da função de controlo do display.

A função realiza a leitura de estado dos sensores e modos de funcionamento do protótipo de robô e apresenta os mesmos no display. Por exemplo, lê as variáveis do modo de funcionamento e apresenta graficamente no display o estado do protótipo. A função é tratada pela tarefa de controlo geral.

3.3.2.1.7 Servidor Websocket

Esta função é responsável por criar e gerir o servidor do websocket para comunicação com o Raspberry PI (Figura 3.24).

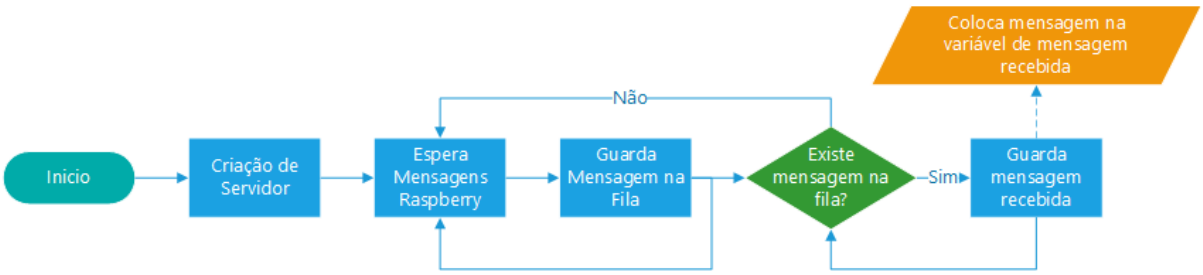


Figura 3.24 – Estrutura da função de servidor de websockets no Arduino dos protótipos de robô.

Com esta função é criado e gerido o servidor websocket. A função recebe as mensagens e coloca-as numa fila de mensagens, de forma a não se perder nenhuma. Esta fila é gerida pela tarefa geral de análise das mensagens. No fim de guardar cada uma das mensagens recebidas é enviado o *feedback* de mensagem recebida para o cliente que lha enviou.

3.3.2.1.8 A função é tratada pela tarefa de controlo geral. Cliente Websockets

Esta função é responsável por criar o cliente de websockets e gerir o envio de mensagens de estado para o Raspberry PI (Figura 3.25).

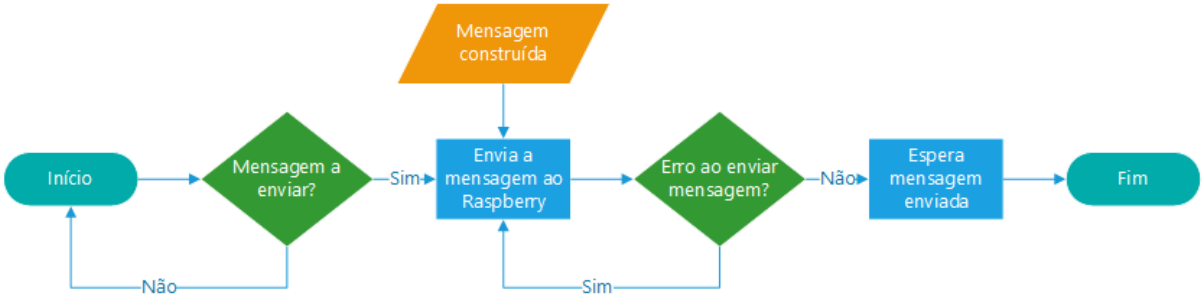


Figura 3.25 – Estrutura da função de cliente de websockets no Arduino dos protótipos de robô.

A função envia a cada segundo o estado do protótipo de robô para o Raspberry PI. Nessa mensagem vão diversos estados como distâncias, estado de sensores e modo de funcionamento atual. Em caso de erro no envio da mensagem, esta é reenviada.

A função é tratada pela tarefa de controlo geral.

3.3.2.2 Tarefas existentes no Arduino

3.3.2.2.1 Controlo geral do protótipo de robô

Esta tarefa é a responsável por efetuar o controlo global do protótipo de robô (Figura 3.26). A tarefa é responsável por gerir os modos de funcionamento, e com estes dados controlar diversos elementos como sinalizadores, displays e definir variáveis de estado gerais utilizadas noutras tarefas e funções.

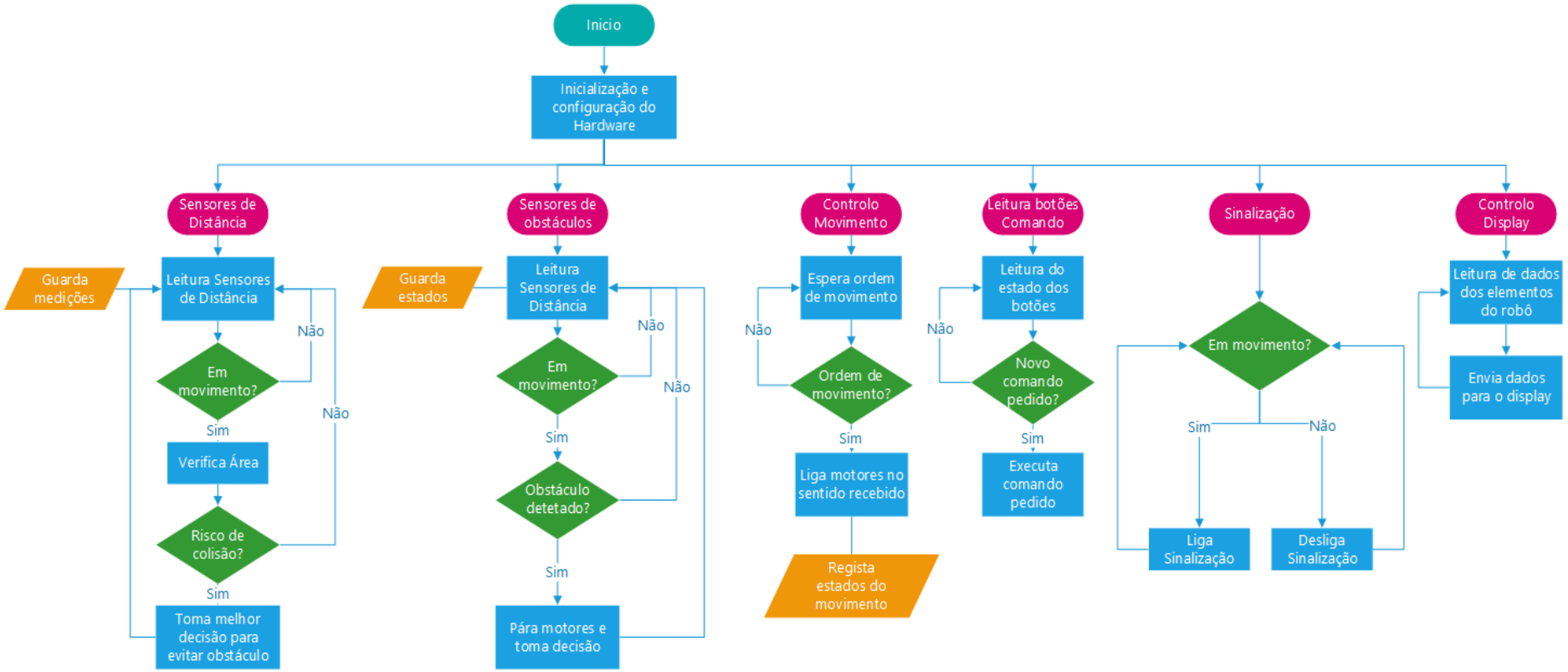


Figura 3.26 – Estrutura da tarefa de controlo global do protótipo de robô.

3.3.2.2.2 Gestão de comunicações

Esta tarefa é responsável por efetuar a comunicação com o Raspberry PI (Figura 3.27).

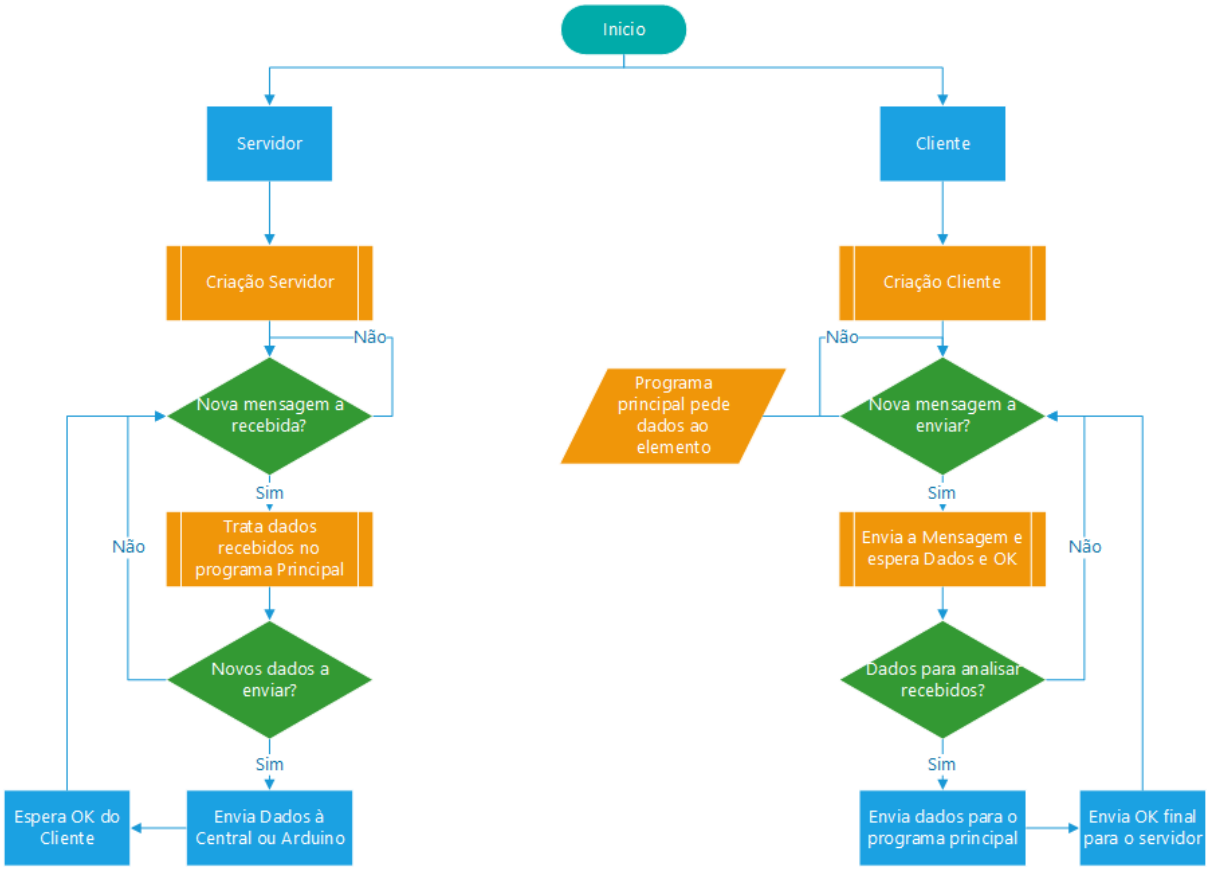


Figura 3.27 – Estrutura da tarefa de gestão de comunicações.

A tarefa gere e analisa a troca de mensagens com o Raspberry PI, depois de criados os websockets cliente e servidor. A tarefa tem a capacidade de descodificar e codificar as mensagens trocadas. Com esta tarefa são lidas as variáveis geradas por diversas funções indicadas na Figura 3.26, e contruídas as mensagens que são enviadas para o programa do Raspberry PI.

3.3.2.2.3 Controlo de movimentos

Esta tarefa é a responsável por gerir a localização e controlo do protótipo de robô (Figura 3.28).

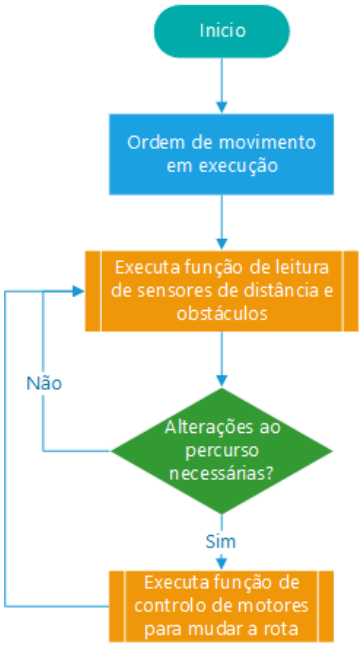


Figura 3.28 – Estrutura da tarefa de controlo de movimentos.

Nesta tarefa são adquiridos os estados dos sensores de obstáculos e dos sensores de distância por ultrassons. Com esta informação são realizadas todas as ações necessárias para que o protótipo se mova, pare, ou se desvie de obstáculos que encontra no seu percurso.

De referir que esta tarefa só é controlada pelo Arduino quando o protótipo de robô se encontra em modo de demonstração dos movimentos das rodas omnidirecionais.

3.3.2.3 Funções existentes no Raspberry PI

3.3.2.3.1 Servidor de Websocket do RaspberryPi

Esta função é responsável por criar e gerir o servidor do websocket para comunicação com o Arduino e com a Central (Figura 3.29).

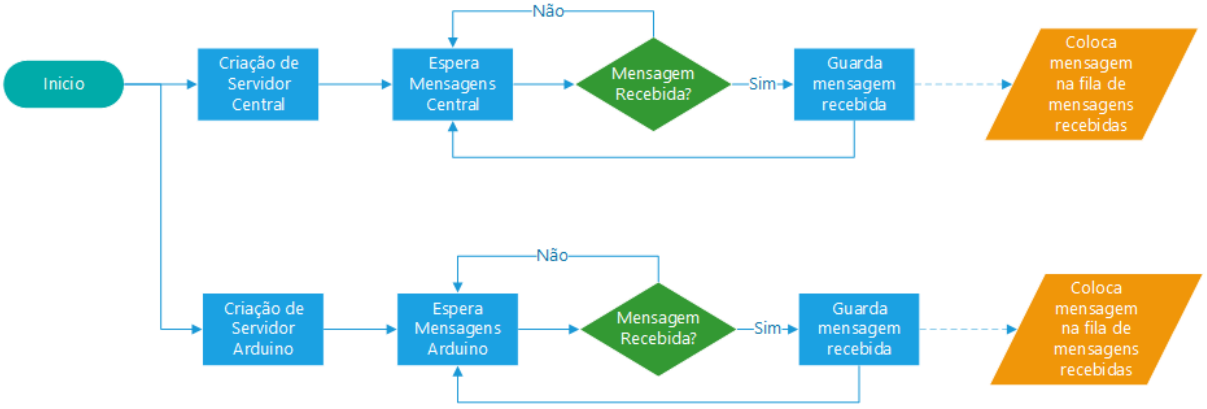


Figura 3.29 – Estrutura da função de servidor de websockets do Raspberry PI dos protótipos de robô.

Com esta função são criados e geridos os servidores websocket. A função recebe as mensagens, trata-as e encaminha-as para o processo indicado para tratar esta informação.

3.3.2.3.2 Cliente de websockets do Raspberry PI

Esta função é responsável por gerir o envio de mensagens de estado e decisões para o Arduino e para a Central (Figura 3.30).

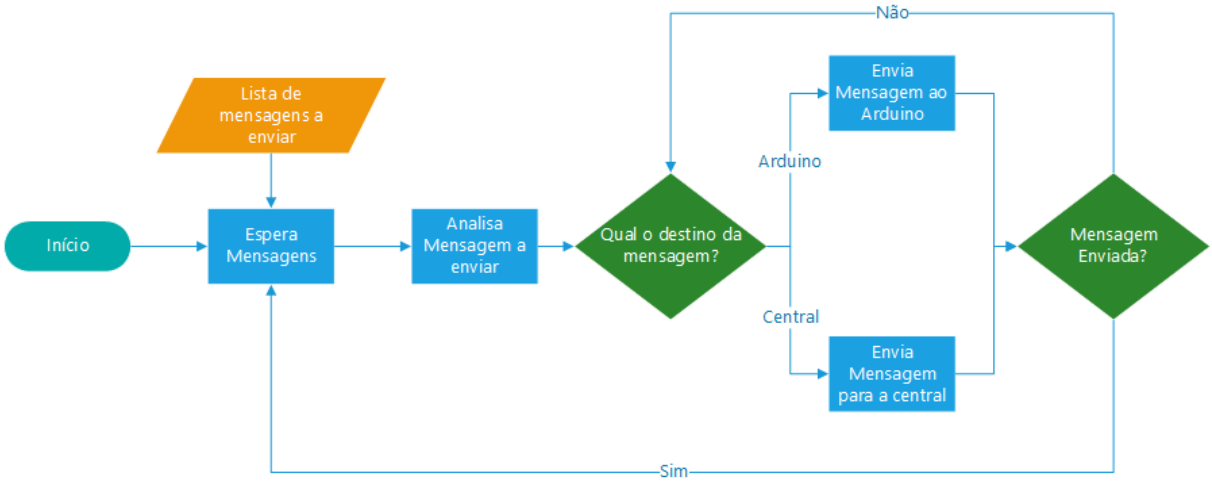


Figura 3.30 – Estrutura da função de cliente de websockets do Raspberry PI dos protótipos de robô.

Com esta função é criado e gerido o cliente websocket. Cada vez que o programa principal tem informações e estados a transmitir, esta função envia-os por mensagem para a Central e para o Arduino.

3.3.2.3.3 Criação de mensagens no Raspberry PI

Esta função é responsável por criar as mensagens a enviar para o Arduino e Central (Figura 3.31).

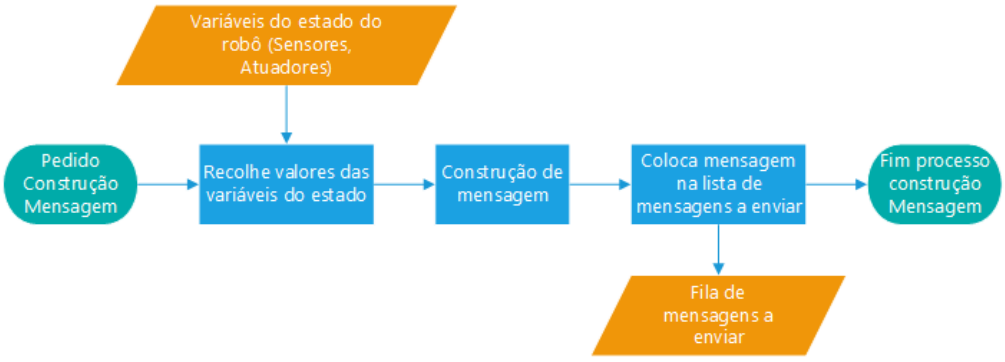


Figura 3.31 – Estrutura da função de criação de mensagens.

Nesta função são construídas as mensagens que são enviadas para os elementos como a Central e o Arduino. Estas mensagens têm um formato específico que é apresentado na Secção 4.4, que pode ser lido facilmente pelo utilizador dos protótipos de robô.

3.3.2.3.4 Descodificação de mensagens

Esta função é responsável por descodificar as mensagens recebidas (Figura 3.32).

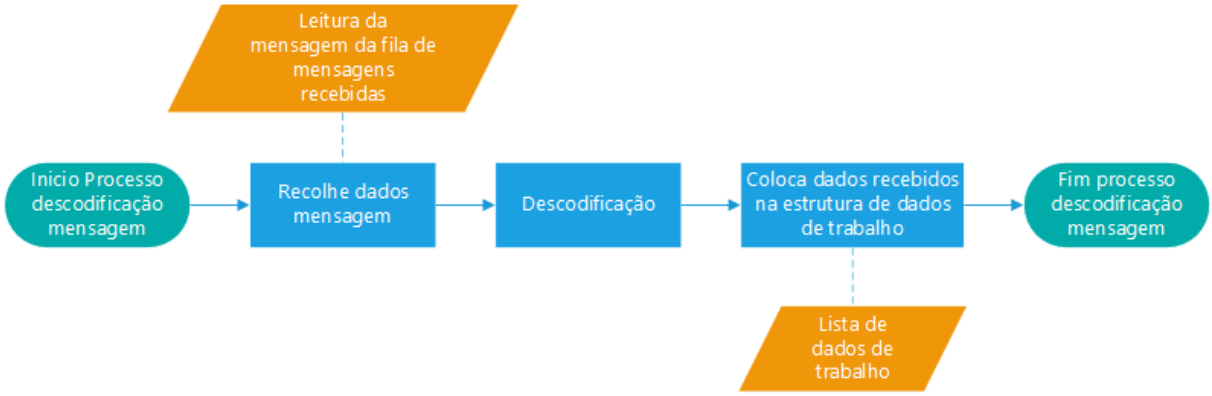


Figura 3.32 – Estrutura da função de descodificação de mensagens.

Nesta função são descodificadas as diversas mensagens recebidas e os dados recebidos são colocados em variáveis globais que podem ser utilizados por diversas funções e tarefas existentes no programa.

3.3.2.3.5 Análise dos sensores

Esta função é responsável por analisar os dados dos sensores que vêm na mensagem do Arduino (Figura 3.33).

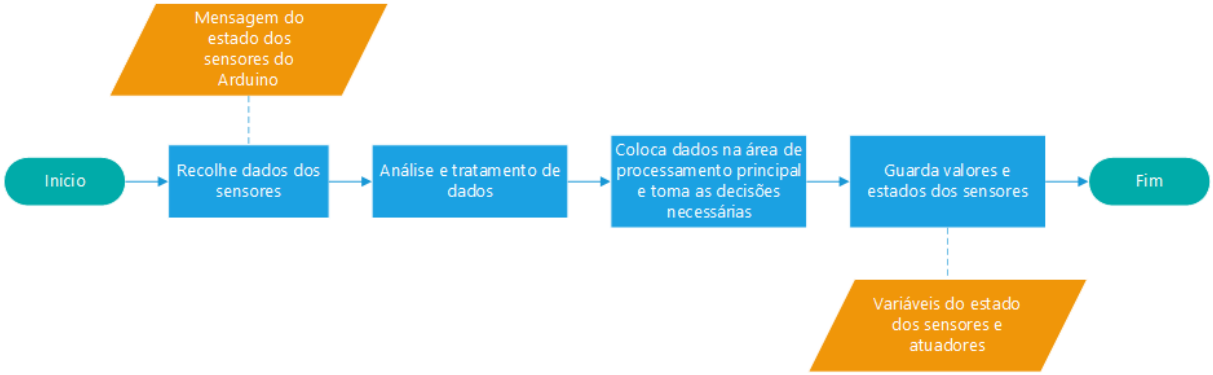


Figura 3.33 – Estrutura da função de análise dos sensores.

Com os dados recebidos esta função ajuda na definição das estratégias de controlo partilhando a informação recebida com a tarefa de controlo de movimento, definindo as ações a tomar para melhor gestão do movimento do protótipo de robô.

3.3.2.3.6 Controlo de movimento do protótipo de robô

Esta função é responsável por controlar o movimento do protótipo de robô (Figura 3.34).

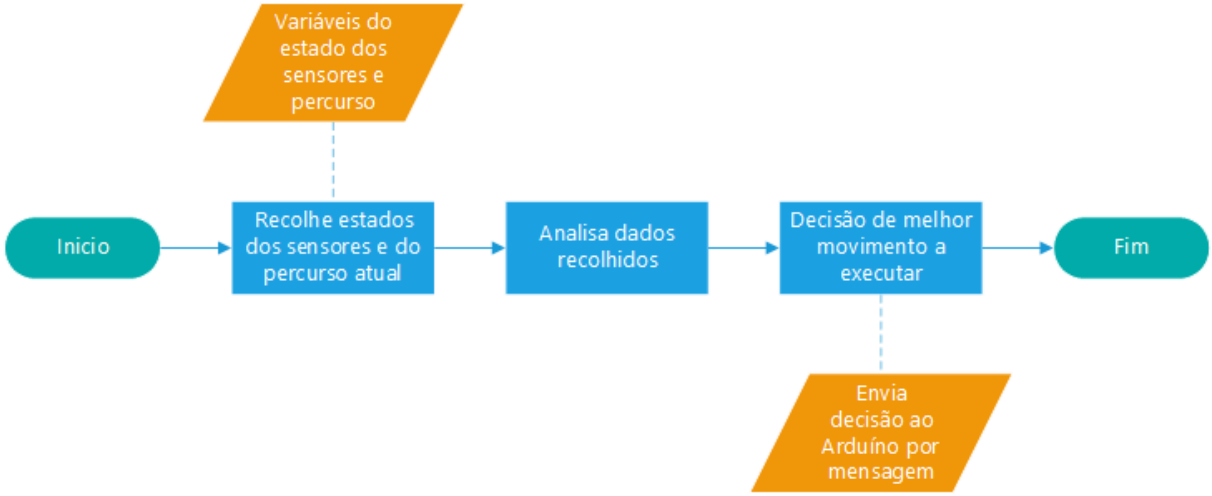


Figura 3.34 – Estrutura da função de Controlo de Movimento do protótipo de robô.

Esta função é a responsável por definir as ações de movimento do protótipo de robô, sendo que só é enviado ao Arduino o sentido de movimento pretendido e ordens de paragem.

3.3.2.3.7 Procura do melhor trajeto

Esta função é responsável por analisar o ficheiro de dados do protótipo da fábrica e definir o trajeto (Figura 3.35).

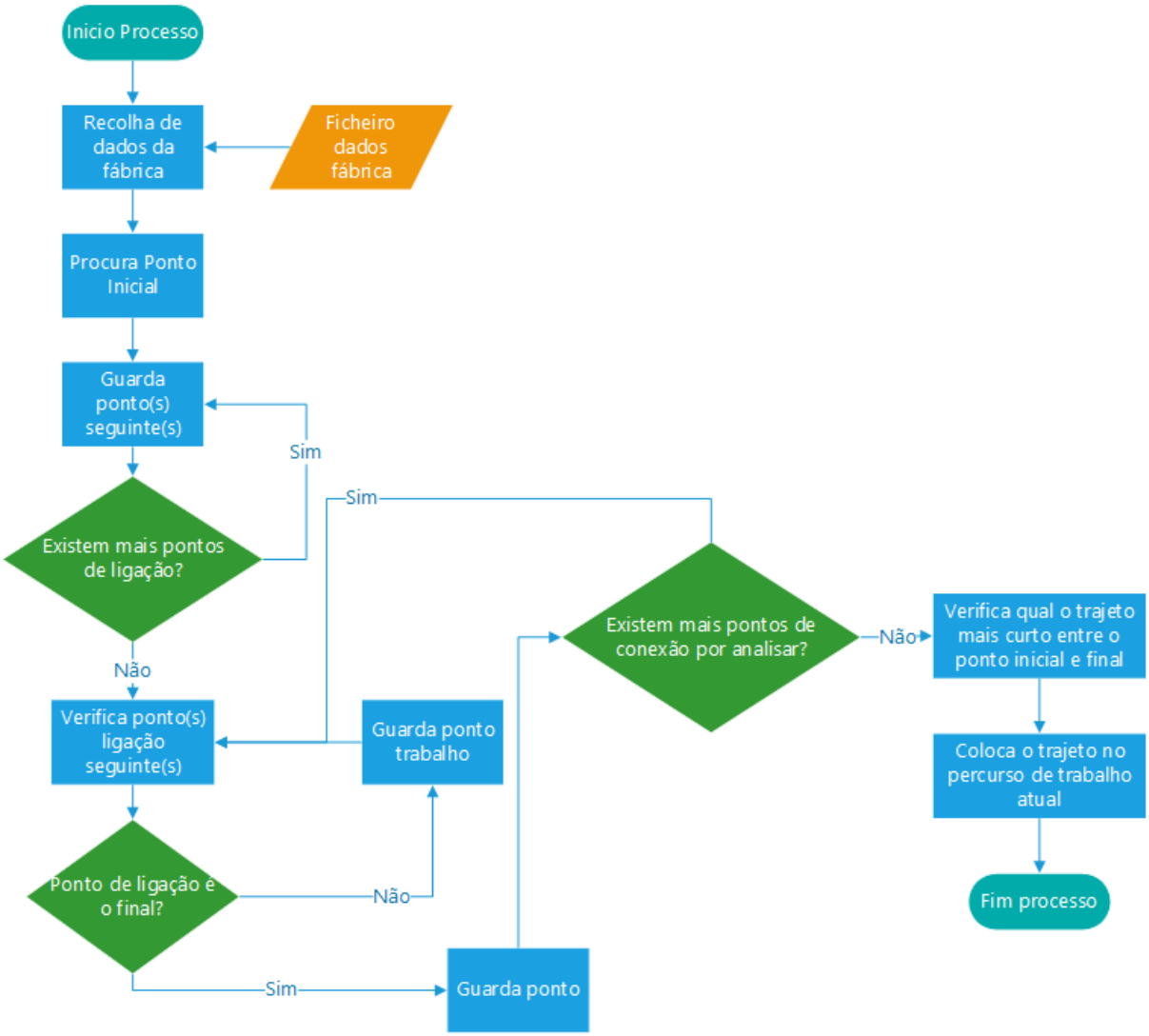


Figura 3.35 – Estrutura da função de procura de melhor trajeto.

Com esta função é consultado o ficheiro de configuração do protótipo da fábrica e com os dados recolhidos é definida a melhor rota ponto a ponto. A função tem a capacidade de calcular os trajetos possíveis entre dois pontos.

3.3.2.3.8 Criação de ficheiros de dados

Esta função é responsável por criar os ficheiros de dados recebidos da Central (Figura 3.36).

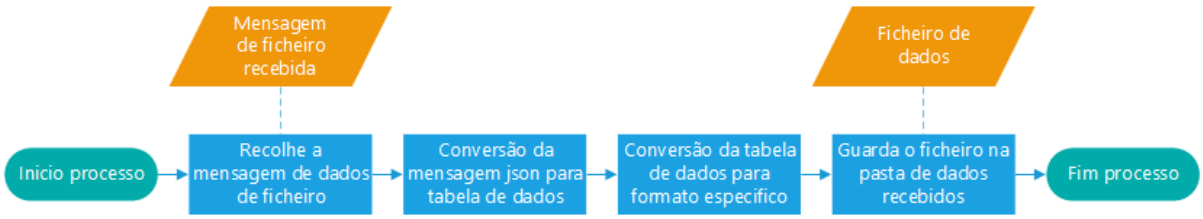


Figura 3.36 – Estrutura da função de criação de ficheiros de dados.

Sempre que o protótipo de robô é inicializado, ou existe uma alteração na configuração dos sentidos dos troços do protótipo da fábrica onde o protótipo se encontra, a Central envia uma mensagem especifica com os dados do protótipo da fábrica atualizados. Esta função recebe essa mensagem e codifica os dados num ficheiro que, sempre que necessário, é consultado por outras funções e tarefas.

3.3.2.3.9 Gestão do leitor RFID

Esta função é responsável por gerir os dados do sistema RFID (Figura 3.37).

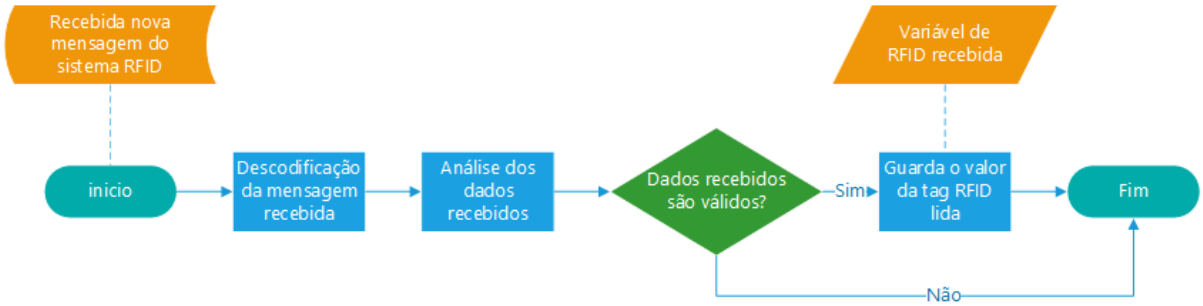


Figura 3.37 – Estrutura da função de Gestão do leitor RFID.

Sempre que o sistema RFID envia uma mensagem esta função é a responsável por descodificar a mensagem recebida e guardar essa informação, desde que relevante, numa variável global, para ser consultada por outras funções e ou tarefas.

3.3.2.4 Tarefas existentes no Raspberry PI

3.3.2.4.1 Comunicação com o Arduino

Esta tarefa é a responsável por efetuar a comunicação com o Arduino (Figura 3.38).

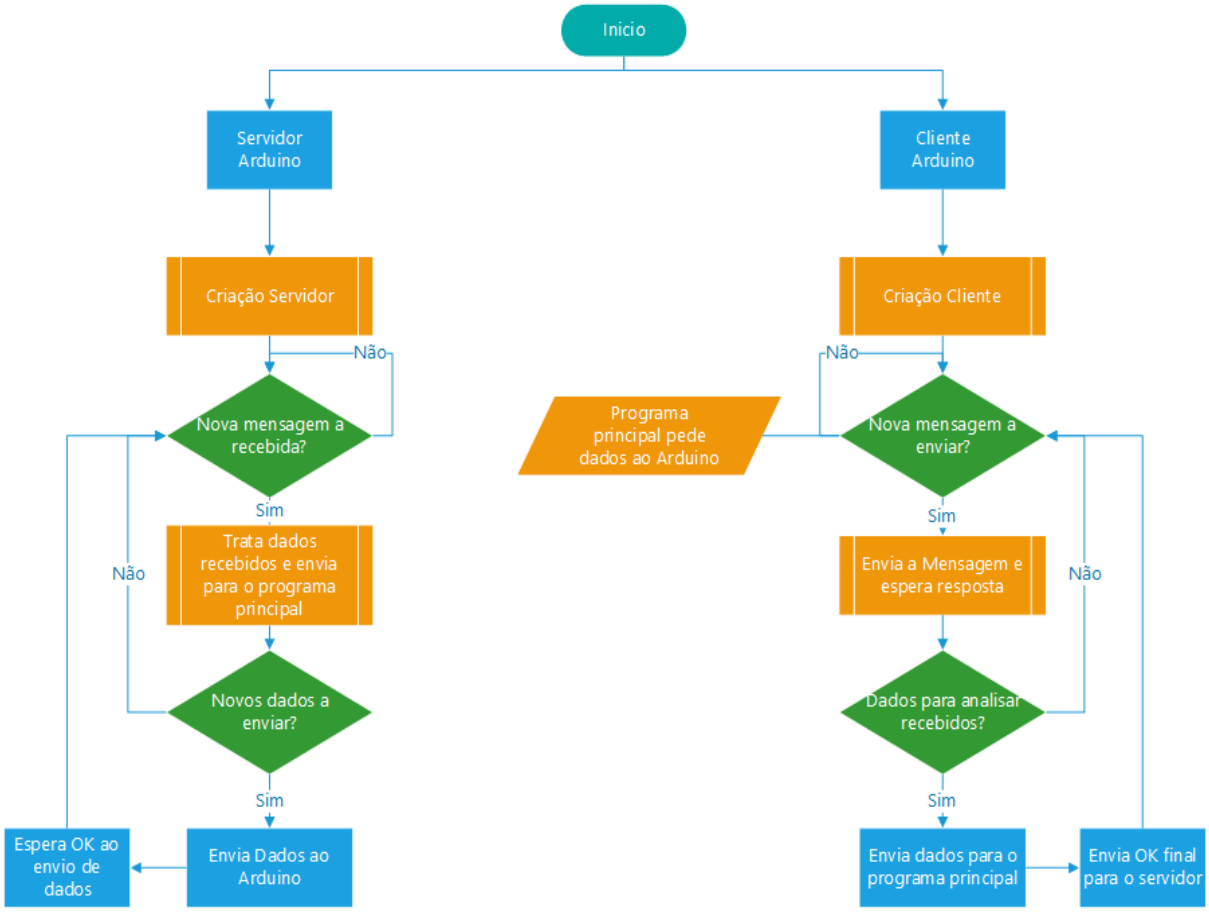


Figura 3.38 – Estrutura da tarefa de gestão de comunicações com o Arduino.

Nesta tarefa é gerida e analisada a troca de mensagens com o Arduino e criados os websockets cliente e servidor. Esta tarefa tem a capacidade de descodificar e codificar as mensagens trocadas.

3.3.2.4.2 Comunicação com a Central

Esta tarefa é a responsável por efetuar a comunicação com a Central (Figura 3.39).

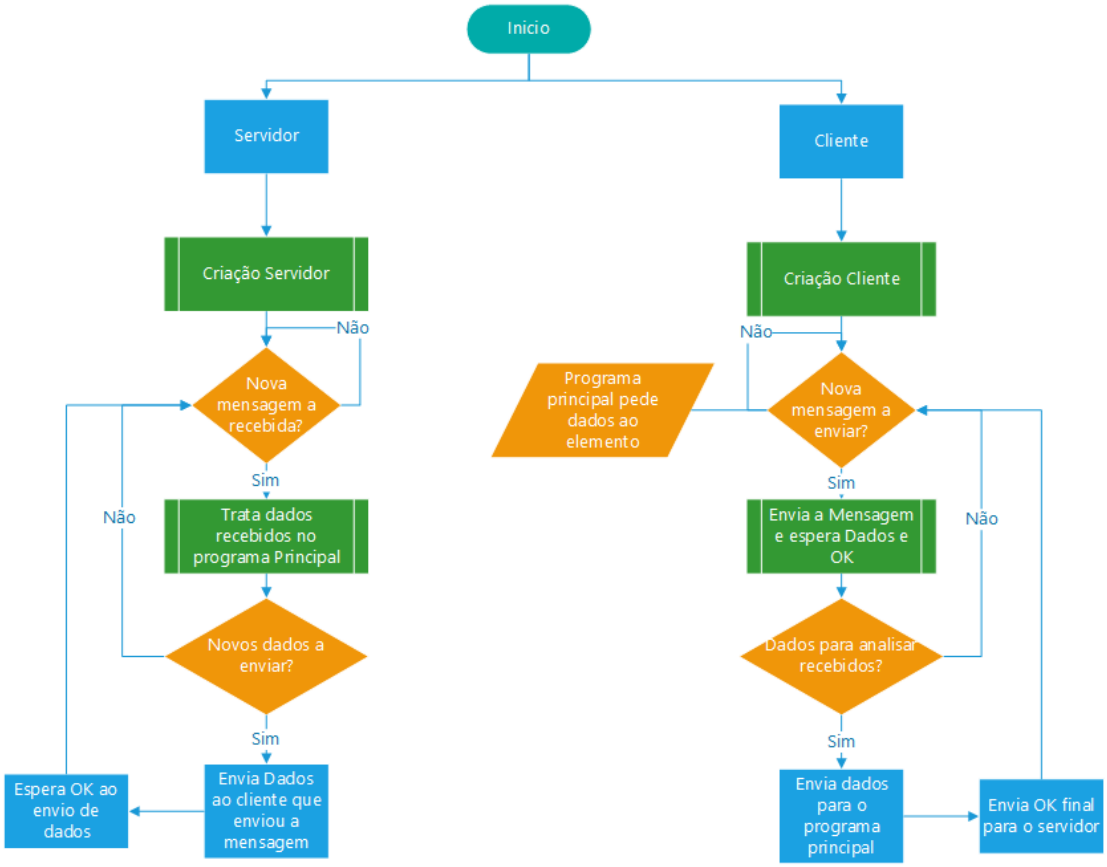


Figura 3.39 – Estrutura da tarefa de gestão de comunicações com a Central.

Esta é responsável por estabelecer a ligação à Central e trocar as informações de estado do protótipo de robô e receber os pedidos da Central, descodificar e executar as melhores estratégias para executar o pedido da Central.

3.3.2.4.3 Controlo de movimento do protótipo de robô

Esta tarefa é a responsável por efetuar o controlo do movimento do protótipo de robô (Figura 3.40).

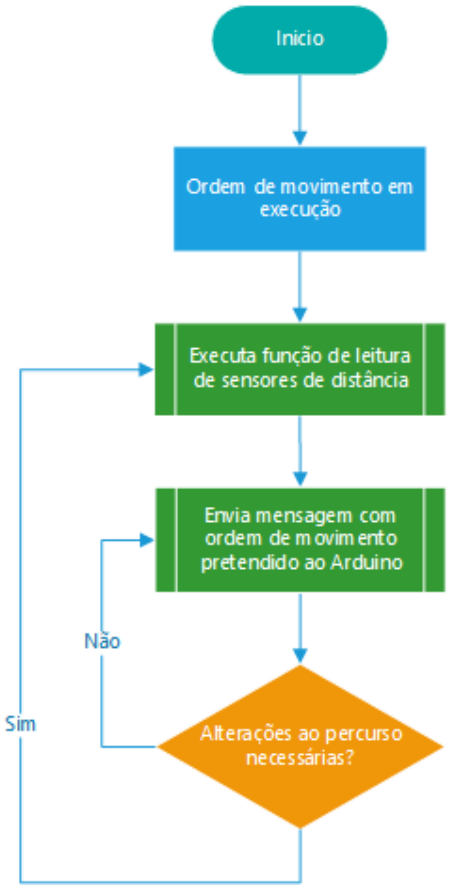


Figura 3.40 – Estrutura da tarefa de controlo de movimento do protótipo de robô.

Esta tarefa gere os movimentos do protótipo de robô sendo que analisa os dados de sensores do Arduino, e com estes define o sentido de deslocação de cada um dos motores do protótipo de robô. Esta recebe o sentido de deslocamento pretendido no trajeto atual, e com os dados dos sensores define a melhor estratégia de movimento a executar.

De referir que esta tarefa é controlada pelo Raspberry PI, recebendo os dados do Arduino e só funciona quando o protótipo está em modo automático. Ou seja, quando o protótipo de robô está em modo de demonstração, é o Arduino que faz o controlo localmente, no caso de estar em modo automático esta tarefa é a responsável por controlar todos os movimentos no Raspberry PI, trocando os sinais e comandos com o Arduino.

3.3.2.4.4 Controlo de percurso

Esta tarefa é a responsável por controlar o percurso do protótipo de robô no trajeto que foi pedido pela Central (Figura 3.41).

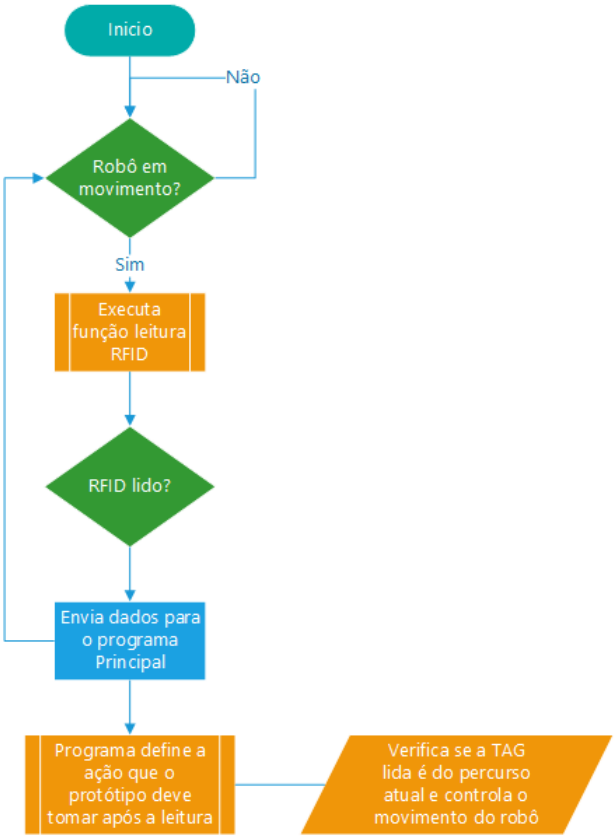


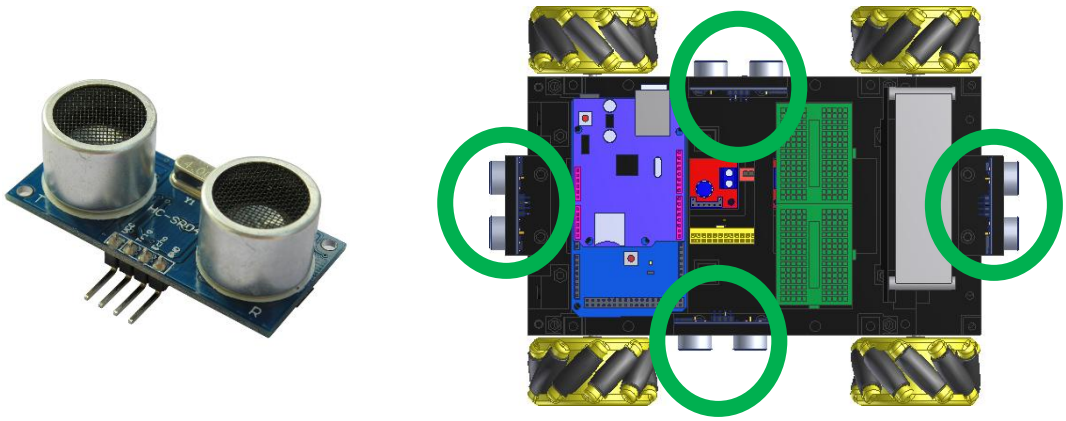
Figura 3.41 – Estrutura da tarefa de controlo de percurso.

Esta tarefa gere o percurso de um ponto para outro. Nesta tarefa são adquiridos e descodificados os dados do RFID e é indicada à tarefa de controlo de movimento (subsecção 0), o próximo movimento que o protótipo tem de executar, por análise do melhor percurso calculado através da função de melhor trajeto a executar (subsecção 0). Esta tarefa também é responsável por fazer a localização automática e retorno à posição de parque.

3.3.3 Guiamento e movimentação

3.3.3.1 Guiamento

O guiamento base dos protótipos de robô é efetuado pela perceção do espaço circundante, através dos sensores de ultrassons do tipo HC-SR04 (Figura 3.42), com capacidade de medição de três centímetros a quatro metros. Estes sensores medem as distâncias entre o protótipo de robô e os objetos circundantes, para ajustarem a trajetória do protótipo, de modo a este deslocar-se sem colidir com qualquer obstáculo. Neste projeto não foram realizados testes de mapeamento do espaço, pois o projeto não tem esse âmbito, embora seja possível em trabalhos futuros implementar essa funcionalidade.



a) b)

Figura 3.42 – a) Sensores HC-SR04; b) Localização dos sensores.

Para além dos sensores de distância ultrassónicos os protótipos de robô dispõem de um sensor seguidor de linhas com cinco sensores óticos (Figura 3.43). Este sensor é usado neste projeto para guiar o protótipo de robô por linhas e ajudar na sua orientação nos trajetos realizados. Este encontra-se na parte inferior traseira dos protótipos de robô.

Neste projeto não foi programada nem testada a funcionalidade do seguimento de linhas, por esta não se encontrar no seu âmbito.

a) b)

Figura 3.43 – a) Sensor seguidor de linhas 5 pontos; b) Localização do sensor na parte traseira.

3.3.3.2 Movimentação

Com a evolução do desenvolvimento e para uma melhor otimização de trajetórias, movimentos e espaço utilizado para manobras, foi decidido utilizar nestes protótipos rodas omnidirecionais do tipo Mecanum (Figura 3.44). Estas rodas caracterizam-se por poder efetuar todos os tipos de movimentos no protótipo de robô, sem que seja necessário efetuar uma transmissão mecânica de direção, ou seja, as rodas estão fixas ao eixo do motor e o motor fixo diretamente na estrutura do protótipo de robô.

Figura 3.44 – Exemplo do tipo de roda Mecanum utilizada.

A configuração de quatro rodas omnidirecionais Mecanum, consiste num par de rodas com inclinação para o lado esquerdo e outro par para o lado direito. Para além desta condição, as rodas devem ser montadas com a sua inclinação a apontar para o centro do protótipo de robô, como mostra a Figura 3.45.

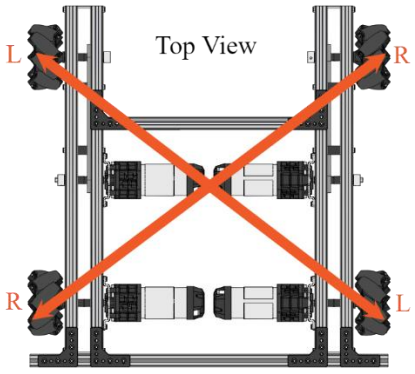


Figura 3.45 – Exemplo de montagem utilizada com rodas Mecanum.

Este tipo de rodas permite diversos movimentos do protótipo de robô, sendo que no modo de demonstração estão configurados vinte e dois tipos de movimentos elementares. Estes movimentos são possíveis através da conjugação de diferentes sentidos de rotação e diferentes velocidades em cada uma das rodas. Na Figura 3.46 e Tabela 3.1 são apresentados os movimentos elementares selecionados para este projeto. Em alguns movimentos representados na Figura 3.46 as setas têm tamanhos diferentes. Quando as setas são mais pequenas significa que a velocidade das rodas junto às mesmas é metade da velocidade das rodas com a seta maior.

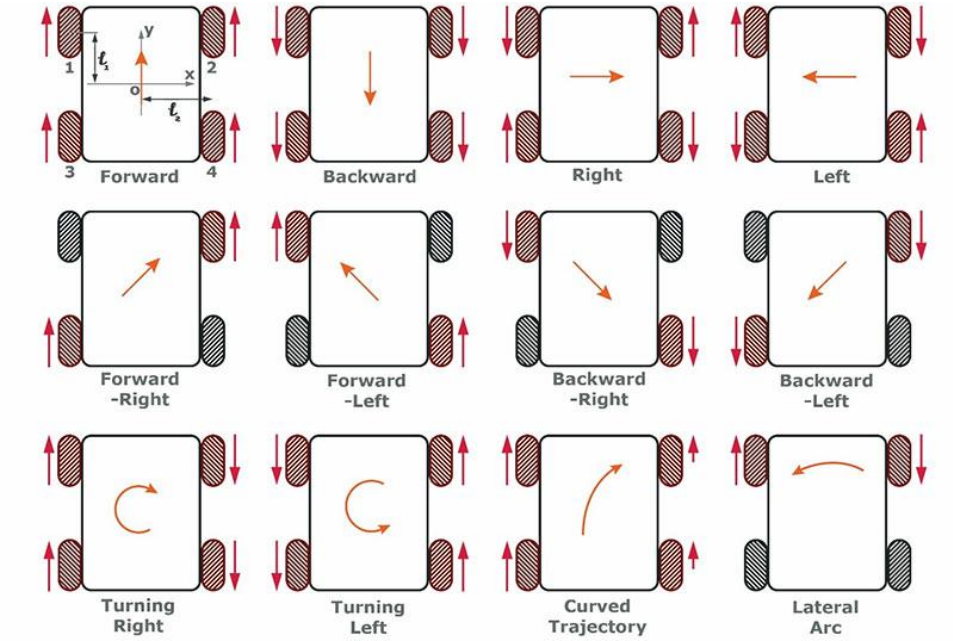


Figura 3.46 –Movimentos mais comuns utilizados com rodas Mecanum.

Durante o desenvolvimento do projeto foi decidido realizar um programa de demonstração que funciona somente no modo manual, e que permite visualizar os vinte e dois movimentos elementares do protótipo de robô que se podem obter com este tipo de rodas. Com as rodas convencionais é mais complexo obter este tipo de movimentos e por exemplo não permitem ter deslocamentos laterais lineares, a não ser que tenham um sistema de direção com ângulos de noventa graus.

Na Tabela 3.1 é possível ver os movimentos que são utilizados no modo de demostração, sendo apresentado o sentido e velocidade de cada uma das rodas para obter este movimento.

Tabela 3.1 – Tabela de movimentos considerados nas rodas omnidirecionais do tipo Mecanum.

Sentido Movimento	Roda Frontal Direita		Roda Frontal Esquerda		Roda Traseira Direita		Roda Traseira Esquerda	
	Sentido	Velocidade	Sentido	Velocidade	Sentido	Velocidade	Sentido	Velocidade
Frente	Frente	n	Frente	n	Frente	n	Frente	n
Trás	Trás	n	Trás	n	Trás	n	Trás	n
Lateral Direita	Trás	n	Frente	n	Frente	n	Trás	n
Lateral Esquerda	Frente	n	Trás	n	Trás	n	Frente	n
Diagonal Frente Direita	Frente	n	Parada	0	Parada	0	Frente	n
Diagonal Trás Direita	Parada	0	Trás	n	Trás	n	Parada	0
Diagonal Frente Esquerda	Parada	0	Frente	n	Frente	n	Parada	0
Diagonal Trás Esquerda	Trás	n	Parada	0	Parada	0	Trás	n
Centro para a Esquerda	Trás	n	Frente	n	Trás	n	Frente	n
Centro para a Direita	Frente	n	Trás	n	Frente	n	Trás	n
Meio da Frente para a Esquerda	Parada	0	Parada	0	Trás	n	Frente	n
Meio da Frente para a Direita	Parada	0	Parada	0	Frente	n	Trás	n
Meio da Traseira para a Esquerda	Trás	n	Frente	n	Parada	0	Parada	0
Meio da traseira para a Direita	Frente	n	Trás	n	Parada	0	Parada	0
Sobre Roda Frente Direita	Parada	0	Trás	n	Parada	0	Trás	n
Sobre Roda Frente Esquerda	Trás	n	Parada	0	Trás	n	Parada	0
Sobre Roda Trás Direita	Parada	0	Frente	n	Parada	0	Frente	n
Sobre Roda Trás Esquerda	Frente	n	Parada	0	Frente	n	Parada	0
Curva Frente Direita	Frente	n/2	Frente	n	Frente	n/2	Frente	n
Curva Trás Direita	Trás	n/2	Trás	n	Trás	n/2	Trás	n
Curva Frente Esquerda	Frente	n	Frente	n/2	Frente	n	Frente	n/2
urva Trás Esquerda	Trás	n	Trás	n/2	Trás	n	Trás	n/2

Este tipo de rodas, com *encoders* incorporados, permite uma maior exatidão no posicionamento. Como este tema não estava no âmbito do projeto, pode ser uma das evoluções a implementar nos protótipos desenvolvidos, sendo que existe informação científica disponível sobre a cinemática destas rodas [8].

Este tipo de rodas necessita de uma manutenção muito superior em relação às rodas convencionais, visto que as mesmas são compostas por uma roda com diversos rolos montados sobre si. Talvez seja este o motivo de as mesmas ainda não serem aplicadas globalmente neste tipo de robôs.

3.4 Estrutura da Central

Nesta secção é apresentada a interface web e interação com a mesma, assim como as funções e tarefas desenvolvidas e implementadas, incluindo os respetivos fluxogramas de funcionamento.

3.4.1 Interface Web

A interface web de interação com o utilizador foi configurada com o sistema de *backend* e *frontend* Django. Está programada nas linguagens Python [7] e HTML [9]. Através dos ficheiros de dados geridos e criados com o programa de interação com os protótipos de robô, são apresentados os estados de cada um dos protótipos de robô ligado à Central e a interface envia os pedidos de testes para o programa de gestão dos protótipos de robô. Optou-se por ter dois sistemas em paralelo, porque o sistema web por omissão, para não sobrecarregar o sistema, só deve ser executado quando é realizada uma ação na interface. Por sua vez, a interação da Central com os protótipos de robô ocorre cada vez que esta efetua um pedido, ou é recebida uma mensagem do estado de um deles.

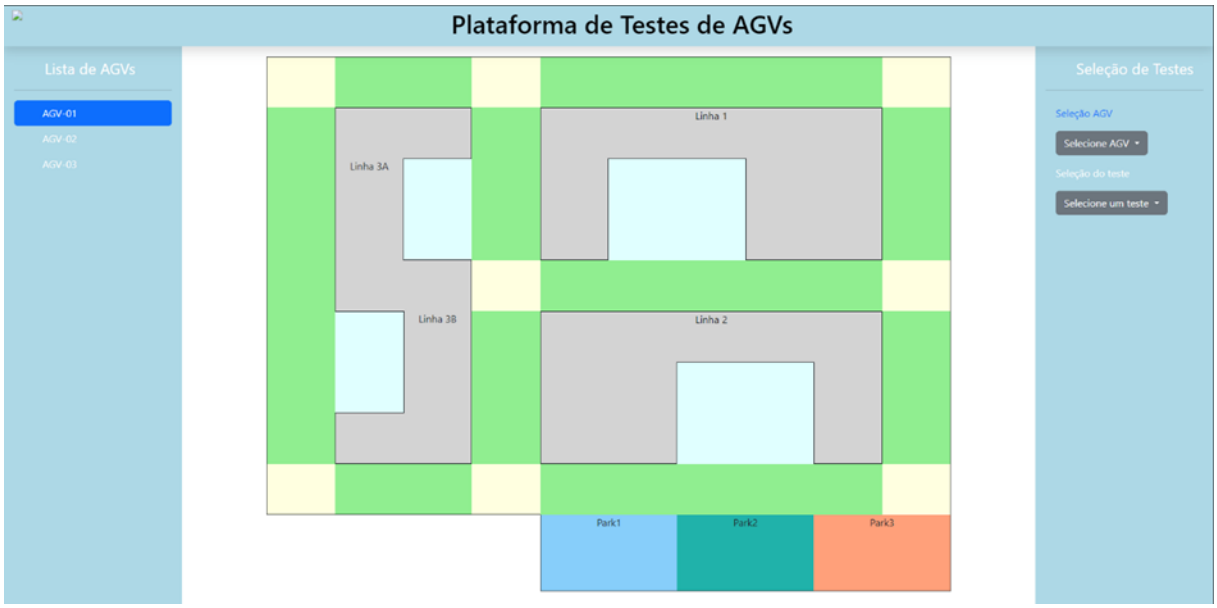


Figura 3.47 – Aspeto da interface gráfica web.

Na interface web (Figura 3.47) quando existe um ou mais protótipos de robô ligados à Central, permite-se a seleção do protótipo pretendido para testes através do campo AGV. Depois de selecionado o protótipo, deve ser selecionada uma das opções de teste a executar, e carregar no botão executar.

As tarefas programadas e testadas no decorrer deste projeto foram a auto-localização, a ida para o estacionamento e o pedido de um trajeto entre dois pontos.

Na tarefa de teste de auto-localização, o protótipo executa os movimentos necessários até encontrar um ponto de referência. Após localizar o ponto de referência, informa a Central onde o protótipo de robô se encontra.

Na tarefa de teste ida para o estacionamento, o protótipo de robô calcula o melhor trajeto e indica na Central o melhor trajeto calculado e indica que se está a deslocar do estacionamento. No final, ao chegar ao estacionamento apresenta a mensagem a indicar o resultado do mesmo.

Na tarefa de teste de trajetória entre dois pontos, o protótipo de robô começa por calcular o melhor percurso até ao ponto inicial. Ao chegar, indica que chegou e aguarda a ordem para seguir até ao próximo ponto. Após receber essa ordem, o protótipo de robô identifica o melhor percurso até ao ponto final e inicia o trajeto. Ao concluir, é indicado ao utilizador o resultado do teste.

Se, durante o percurso, um dos protótipos de robô encontrar um obstáculo, do qual não consegue desviar-se, ele informa a Central. Esta, por meio da interface, sinaliza que há um troço bloqueado e inicia automaticamente a reorganização do mapa da área de produção para criar trajetos alternativos. Ao concluir a busca por alternativas, a Central atualiza a interface com os sentidos dos trajetos modificados.

3.4.2 Software e tarefas

Como já foi referido, a Central tem dois processos a correr: o da interface com o utilizador e o processo de gestão do protótipo da fábrica e da frota de protótipos de robô. A interface web foi programada com recurso à *framework* Django, que permite a ligação de equipamentos externos à Central para realizar testes e aceder à interface web.

O programa de gestão da frota dispõe de um servidor e um cliente websockets que permite comunicar com os protótipos que estão ligados a si (Figura 3.48).

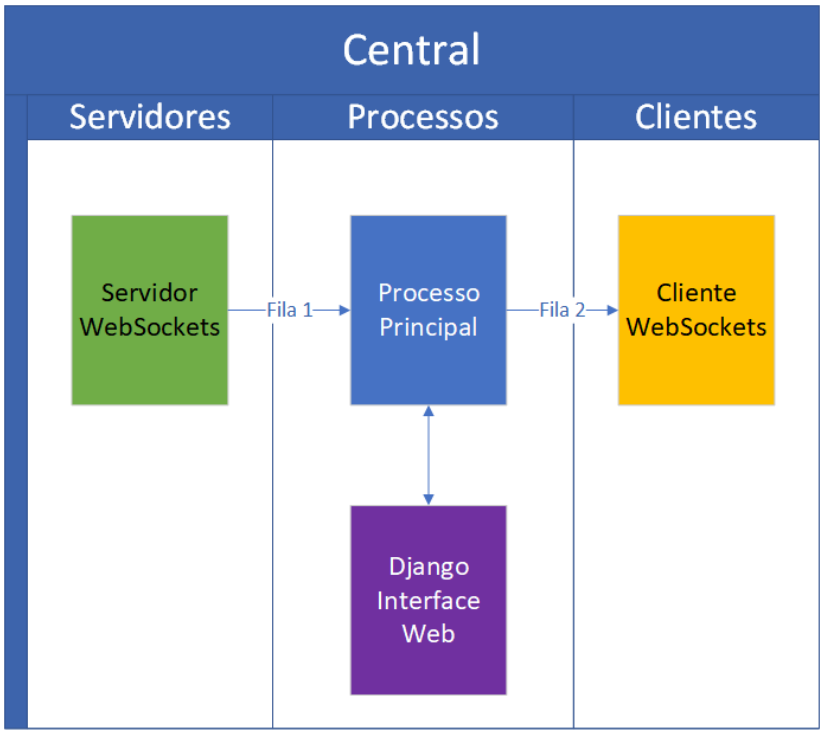


Figura 3.48 – Arquitetura de software principal no Raspberry PI da Central.

Nas próximas subsecções são apresentadas as funções e tarefas da Central programadas no Raspberry PI. As funções são partes de código dedicada a controlar determinadas funcionalidades, por exemplo refazer a configuração da área de produção.

3.4.2.1 Funções existentes no Raspberry PI

3.4.2.1.1 Servidor de websocket da Central

Esta função é responsável por criar e gerir o servidor do websocket para comunicação com os protótipos de robô (Figura 3.49).

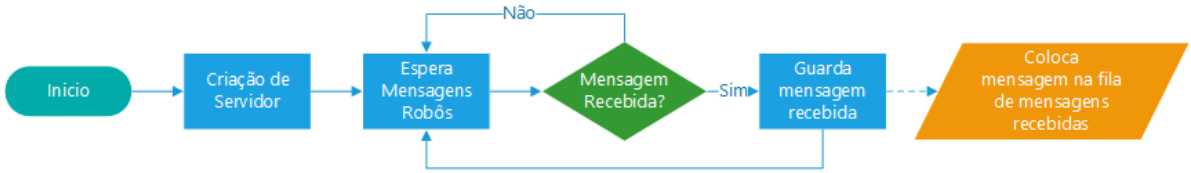


Figura 3.49 – Estrutura da função de servidor de websockets da Central.

A função cria e gere os servidores websocket e recebe as mensagens dos protótipos de robô, trata-as e encaminha-as para uma fila de mensagens. O processo principal é responsável por tratar as mensagens recebidas e tomar as decisões necessárias.

3.4.2.1.2 Cliente de websockets da Central

Esta função é responsável por gerir o envio de mensagens de estado e decisões para os protótipos de robô (Figura 3.50).

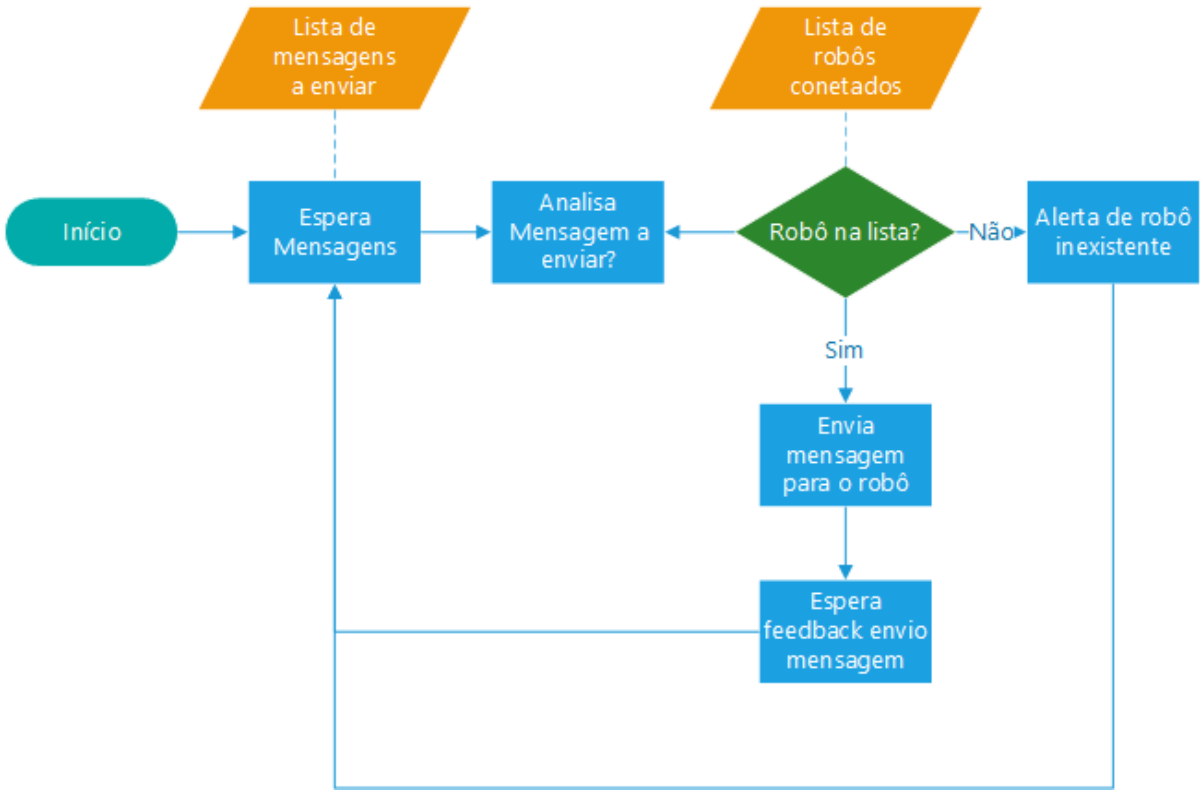


Figura 3.50 – Estrutura da função de cliente de websockets da Central.

Com esta função é criado e gerido o cliente websocket. A função envia as informações e pedidos para os protótipos de robô, cada vez que seja necessário.

3.4.2.1.3 Criação de mensagens

Esta função é responsável por criar as mensagens a enviar para os protótipos de robô (Figura 3.51).

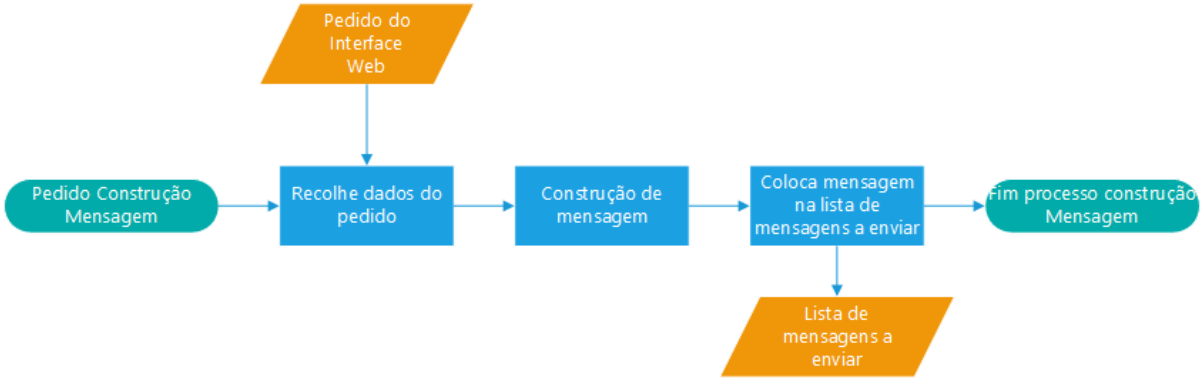


Figura 3.51 – Estrutura da função de criação de mensagens.

Nesta função são construídas as mensagens com os dados dos pedidos realizados pelo utilizador através da interface web, depois de construída a mensagem com os dados do pedido esta é enviada para os protótipos de robô que estão ligados à Central.

3.4.2.1.4 Descodificação de mensagens

Esta função é responsável por descodificar as mensagens recebidas de cada um dos protótipos de robô (Figura 3.52).

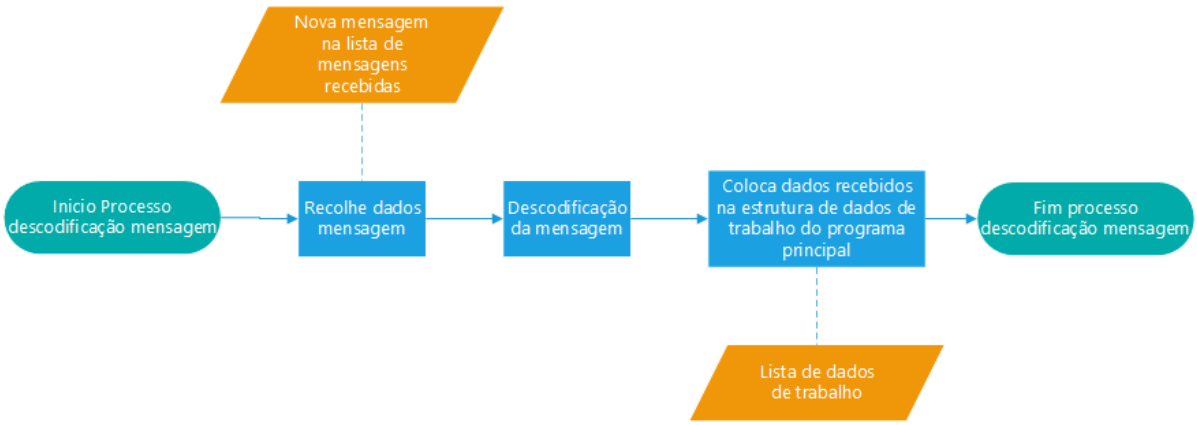


Figura 3.52 – Estrutura da função de descodificação de mensagens.

Nesta função são descodificadas as diversas mensagens recebidas e os dados recebidos são colocados na fila de espera para o programa principal tomar a decisão mais adequada ao estado recebido.

3.4.2.1.5 Procura do melhor trajeto

Esta função é responsável por analisar o ficheiro de dados do protótipo da fábrica e definir o melhor trajeto (Figura 3.53).

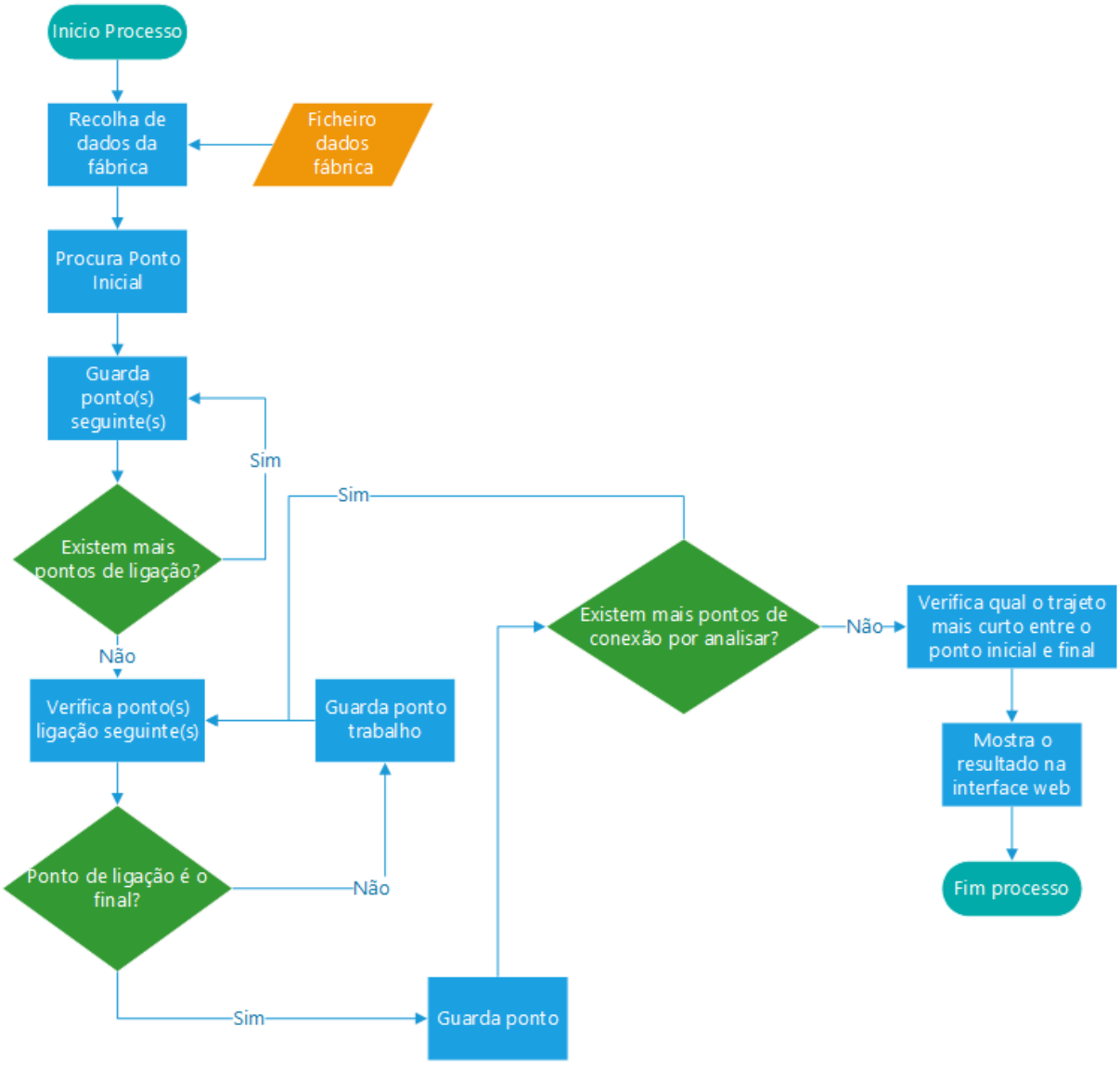


Figura 3.53 – Estrutura da função de definição de melhor trajeto.

Com esta função é possível verificar se o protótipo de robô obteve o melhor resultado para o pedido que a Central solicitou. Para isso, a função encontra todos os caminhos possíveis entre dois pontos de trabalho, escolhendo no final o que tem menor distância. O protótipo, depois de calcular o melhor trajeto entre dois pontos, envia o resultado à Central para o utilizador poder visualizar o trajeto calculado. Por sua vez, a Central implementa o mesmo algoritmo e verifica se o resultado é o mesmo enviado pelo protótipo. É indicando ao utilizador se houve incoerência com o protótipo a seguir um percurso diferente do encontrado pela Central.

3.4.2.1.6 Criação de mensagem de ficheiro de dados

Esta função é responsável por ler os ficheiros de dados e criar a mensagem a enviar para os protótipos de robô (Figura 3.54).

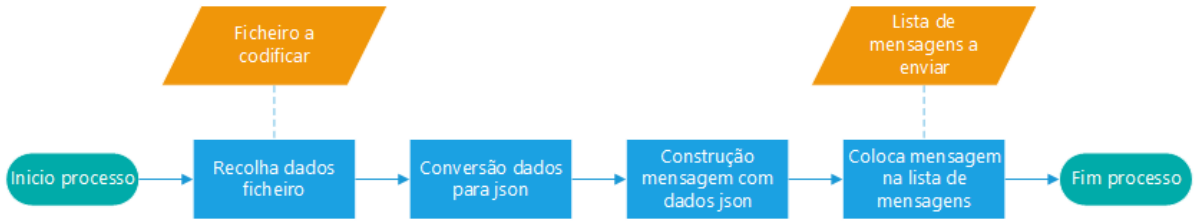


Figura 3.54 – Estrutura da função de criação da mensagem de ficheiros de dados.

Sempre que um protótipo de robô se liga de novo à Central ou existe uma alteração na configuração do protótipo da fábrica, a Central envia uma mensagem especifica com os dados do protótipo da fábrica atualizados. Esta função lê os ficheiros de dados, prepara essa mensagem e envia-a para os protótipos de robô que necessitem desta informação. Os dados do protótipo da fábrica enviados aos robôs são as informações das localizações de cada linha, dos troços e dos nós, assim como os dados das *tags* RFID para cada ponto. Os pormenores relativos a estes dados encontram-se na Secção 4.1.

3.4.2.2 Tarefas existentes na Central

3.4.2.2.1 Comunicação com os protótipos de robô

Esta tarefa é a responsável por efetuar a comunicação com os protótipos de robô (Figura 3.55).

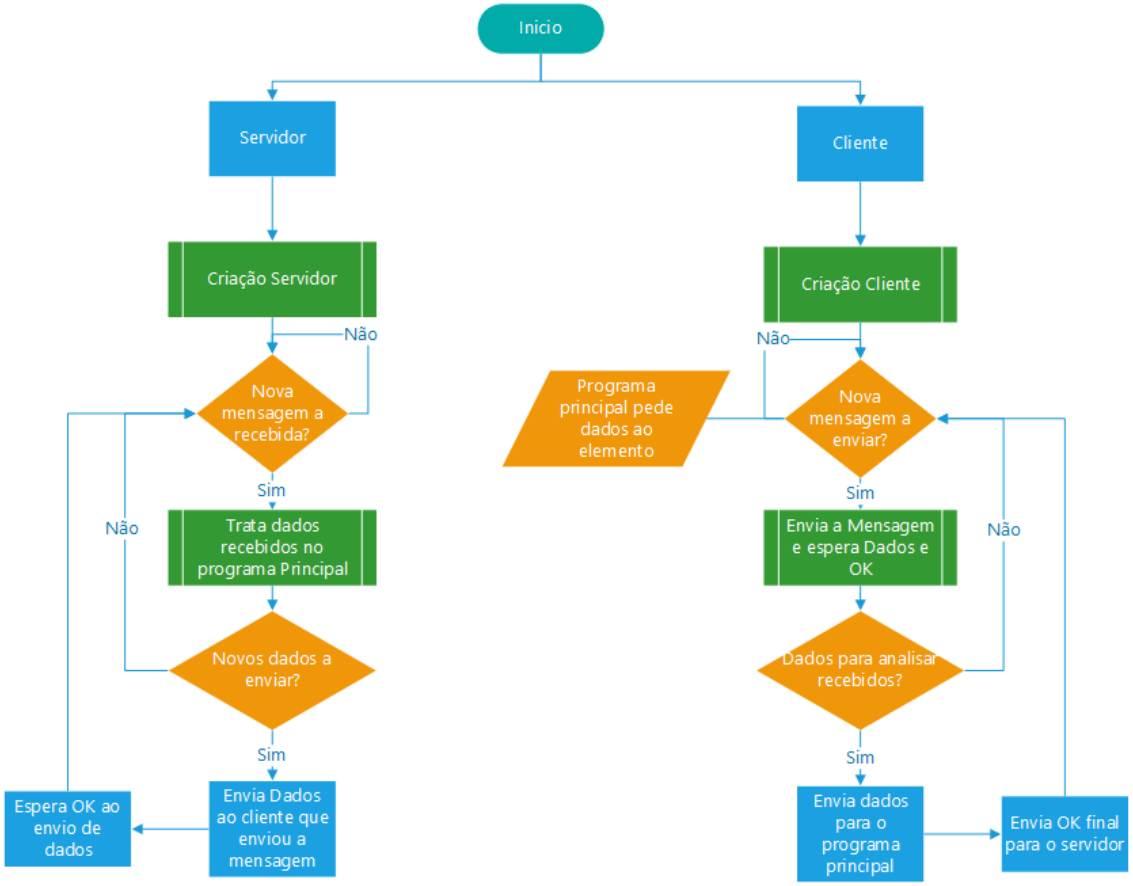


Figura 3.55 – Estrutura da tarefa de comunicação com os protótipos de robô.

Nesta tarefa é gerida e analisada a troca de mensagens com os protótipos de robô, e são criados os websockets cliente e servidor. Esta tarefa tem a capacidade de descodificar e codificar as mensagens trocadas.

3.4.2.2.2 Troca de dados com a aplicação da interface web

Nesta tarefa são controlados os dados trocados com a interface web, desenvolvida na framework Django (Figura 3.56).

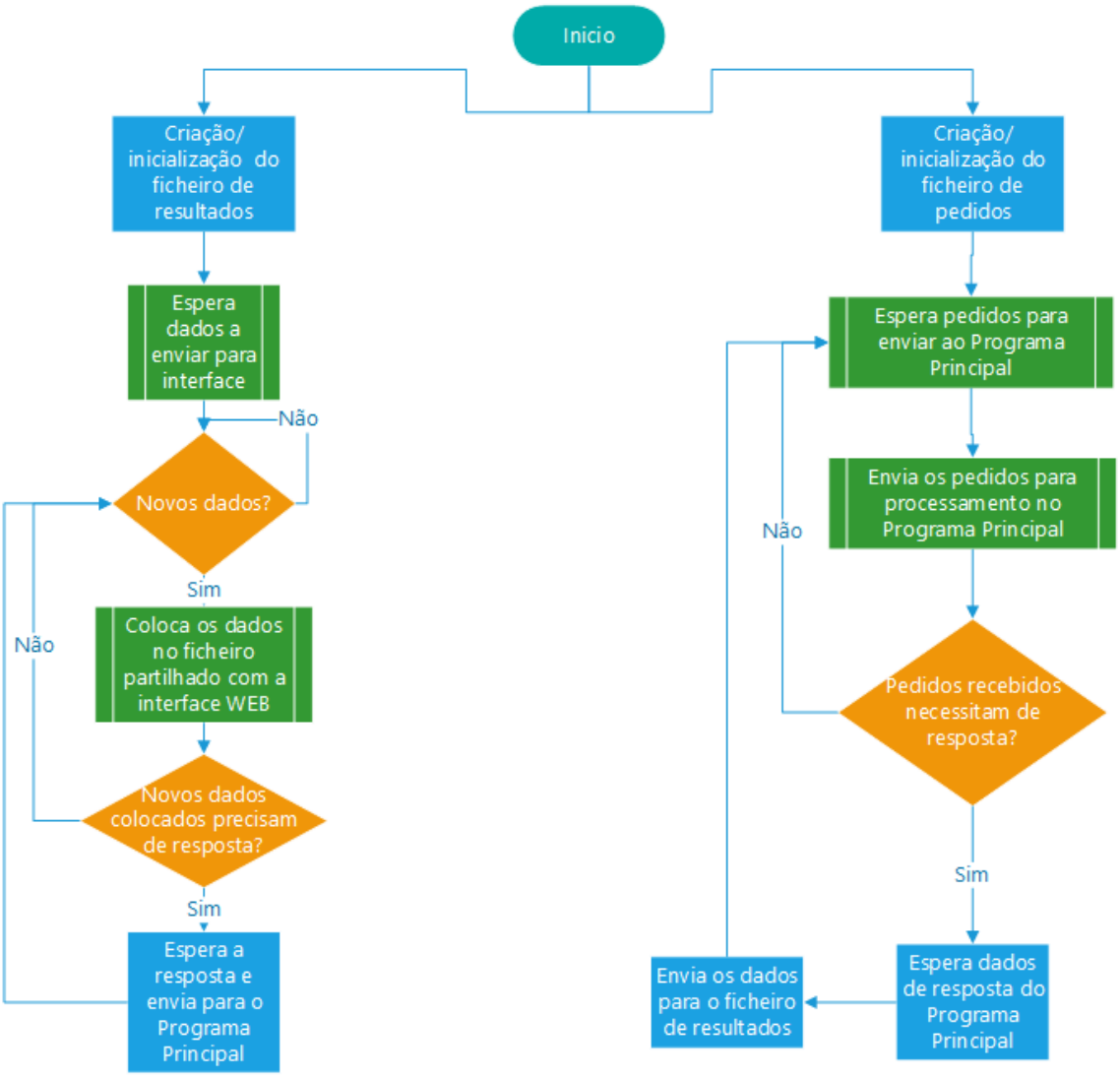


Figura 3.56 – Estrutura da tarefa de troca de dados com a interface.

Esta tarefa, sempre que recebe uma nova informação, quer seja da interface ou dos protótipos de robô, é responsável por analisar os dados e fazer a troca de informação entre os dois elementos-chave da Central, a interface e os protótipos de robô.

4 TESTES E RESULTADOS

Neste capítulo são descritos diversos testes realizados durante o desenvolvimento deste projeto e os respetivos resultados. Na descrição destes testes é indicado o objetivo de cada um e apresentado o resultado obtido, assim como alguns problemas que surgiram, para os quais é apresentada uma resolução.

4.1 Envio de ficheiros da Central para os protótipos de robô

4.1.1 Objetivo

Foram testadas diversas formas para enviar ficheiros de dados da Central para os protótipos de robô. Foi testado o envio de ficheiros por FTP, enviando os ficheiros para uma pasta partilhada que existia em cada um dos protótipos de robô. No entanto, este método resultava em erros e atrasos no envio se o ficheiro estivesse a ser usado. O autor procurou formas alternativas e uma forma rápida e eficaz que encontrou para o envio dos ficheiros de dados, que se encontram em formato "xls", foi o envio via websocket e em formato "json".

O ficheiro de dados é convertido para um ficheiro "json" e enviado por websocket para o protótipo em questão.

Do lado do protótipo de robô, sempre que é recebida uma mensagem do tipo ficheiro, o programa efetua a leitura dos dados em formato "json" e converte-os para um ficheiro "xls".

São enviados diversos ficheiros entre a Central e os protótipos de robô. Estes ficheiros contêm informação fundamental para o funcionamento dos protótipos de robô. Os dados do protótipo da fábrica foram divididos em quatro ficheiros.

O ficheiro Factory.xls (Figura 4.1) permite configurar os diversos caminhos do protótipo, nomeadamente os trajetos possíveis constituídos por vários troços. Os trajetos são descritos por um conjunto de troços. No exemplo encontra-se a informação de cada troço do trajeto, sendo o troço constituído pelo ponto inicial e final, a distância entre esses pontos e o sentido de movimento. O sentido entre os dois pontos é representado pelos números "1" e "2", respetivamente, do ponto final para o ponto inicial e do ponto inicial para o ponto final. O sentido pode ainda assumir o valor "3" indicando movimentos nos dois sentidos. Por fim, o valor "0" indica que o troço está desabilitado. A definição dos troços pressupõe que o protótipo de robô não necessita de alterar o sentido de movimento enquanto se movimenta entre o ponto inicial e o ponto final.

Troço	Nó Inicial	Nó Final	Distância (mm)	Sentido
1	1	2	1000	2
2	2	3	600	2
3	3	5	800	2
4	5	8	800	1
5	1	6	1200	1
6	6	7	500	1
7	7	8	1200	2
8	2	4	800	2
9	4	5	1200	1
10	4	7	800	2

Figura 4.1 – Dados existentes no ficheiro de dados Factory.xls.

Figura 4.2 – Indicação representativa do protótipo da fábrica com os nós e sentidos definidos.

Na Figura 4.2 as setas indicam os sentidos atuais dos troços para os oito pontos representados. Com a definição da configuração do protótipo da fábrica, o protótipo de robô pode efetuar vários trajetos, que são determinados pela Central quando indica a origem e o destino do movimento. O controlador Raspberry PI do protótipo de robô tem autonomia para decidir a rota, após receber a informação da Central. A representação gráfica da Figura 4.2 corresponde ao conteúdo do ficheiro Factory.xls (Figura 4.1) e do ficheiro NodeInfo.xls (Figura 4.3).

O ficheiro NodeInfo.xls (Figura 4.3) contém informação relacionada com cada um dos nós detalhados no Factory.xls (Figura 4.1). Para cada nó é adicionada uma linha com a seguinte informação: o número do nó, o tipo de nó que indica se o nó é uma curva (L) ou um ponto de interseção de três troços (T) e os quatro pontos que indicam nós adjacentes. Estes quatro pontos permitem determinar o sentido de rotação do protótipo. O ponto 1 indica o troço na base do T, o ponto 2 indica o braço direito do T e o ponto 3 indica o braço esquerdo. O ponto 4 foi considerado e configurado para uma possível evolução do protótipo da fábrica para pontos em cruz (+), em que o mesmo indicará o ponto ligado na linha superior da cruz. No caso da curva (L) apenas são preenchidos os pontos associados à mudança de direção (esquerda e direita). A Central tem software que permite usar estes dados (Figura

4.3) para indicar e definir a interligação de um determinado nó aos nós que estão ligados a si. Estes dados são bastante importantes para a modificação dinâmica da configuração do protótipo fábrica.

Confrontando a Figura 4.2, que representa o protótipo da fábrica, e a Figura 4.3, que apresenta os dados de cada um dos nós, podemos ter uma melhor perceção do significado dos dados do ficheiro NodeInfo.xls. Os nós 1, 3, 6 e 8 são curvas simples, em que só existem duas ligações distintas a estes nós. Já os nós 2, 4, 5 e 7 têm uma configuração em que a eles ligam sempre três nós. Por exemplo, ao colocar a definição do T no nó 2, podemos verificar que na parte inferior do T liga o nó 4, na parte esquerda liga o nó 1 e na parte direita liga o nó 3.

Se aplicarmos a configuração do T ao nó 4, obtemos então na parte inferior do T o nó 5, na parte esquerda o nó 7 e na parte direita o nó 2.

		Base	Direita	Esquerda	Frente	Numero de nós de	
Nó	Tipo	Ponto 1	Ponto 2	Ponto 3	Ponto 4	Entradas	Saídas
1	L	0	2	6	0	1	1
2	T	4	1	3	0	1	2
3	L	0	5	2	0	1	1
4	T	5	7	2	0	2	1
5	T	4	3	8	0	2	1
6	L	0	1	7	0	1	1
7	T	4	8	6	0	1	2
8	L	0	7	5	0	1	1

Figura 4.3 – Dados existentes no ficheiro de dados NodeInfo.xls.

O ficheiro Workpoints.xls (Figura 4.4) contém os dados da localização de cada ponto de trabalho. Neste ficheiro é definido o nome da Linha, e o troço do protótipo da fábrica onde a linha se encontra. Também é indicado o código RFID esperado na localização de cada uma das linhas.

ID	Nome	Localização (Troço)	RFID
1	Linha 1	9	1234
2	Linha 2	7	12345
3	Linha 3 - A	8	123456
4	Linha 3 - B	5	78941
5	Estacionamento 1	7	4259
6	Estacionamento 2	7	488
7	Estacionamento 3	7	5427

Figura 4.4 – Dados existentes no ficheiro de dados Workpoints.xls.

O ficheiro Nodeid.xls (Figura 4.5) contém os dados do RFID encontrado em cada um dos nós da fábrica, por exemplo no ponto 1 o protótipo sabe que o código esperado é o 14485111.

ID	RFID
1	15485111
2	15485112
3	15485113
4	15485114
5	15485115
6	15485116
7	15485117
8	15485118

Figura 4.5 – Dados existentes no ficheiro de dados Nodeid.xls.

4.1.2 Resultados

Os resultados obtidos foram os esperados, com a Central a converter os ficheiros do formato "xls" para "json", e a enviar os mesmos para o protótipo de robô, por websocket.

O protótipo de robô, à escuta do websocket, quando recebe a mensagem cria os ficheiros "json" e converte-os em ficheiros "xls" (Figura 4.6). Nesta figura pode-se observar a pasta vazia e depois a mesma com os ficheiros "json" e "xls".

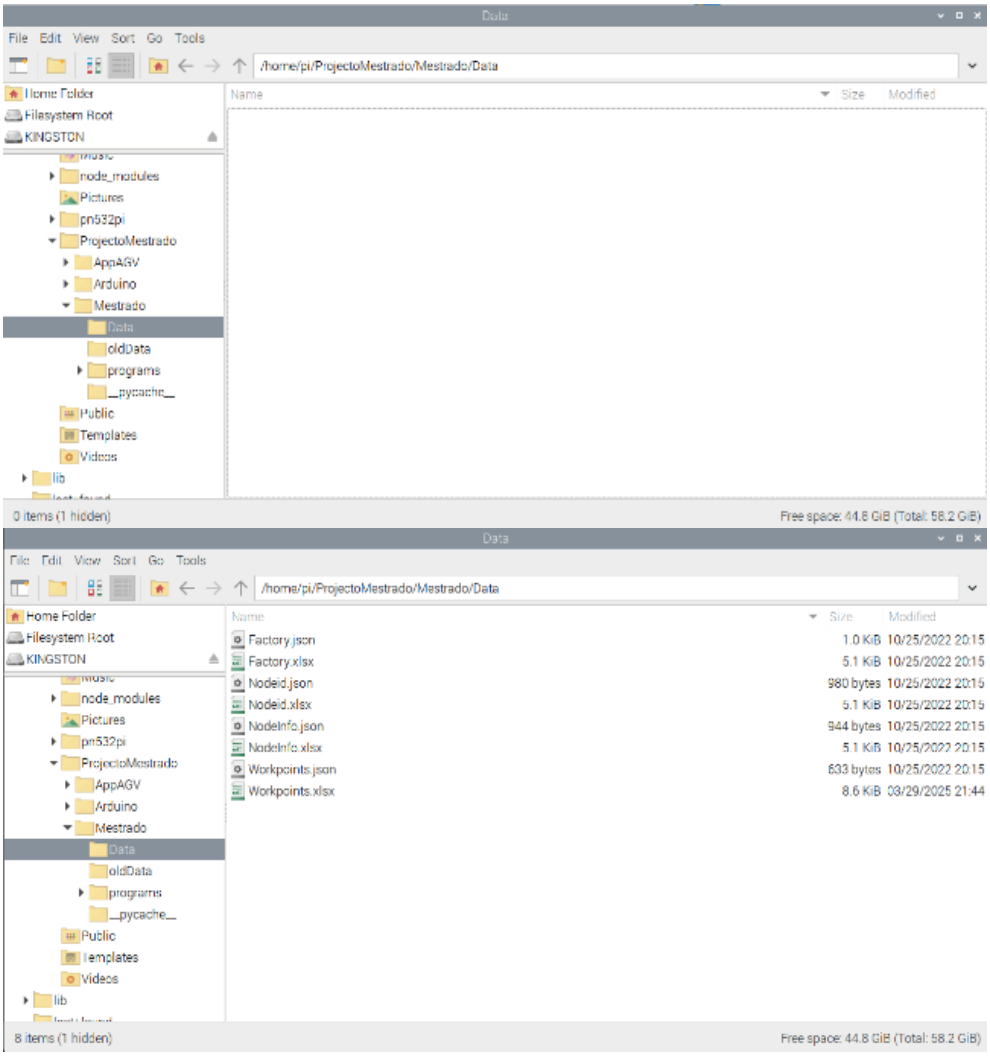


Figura 4.6 – Pasta de dados no protótipo de robô antes e após a receção de dados.

4.2 Algoritmo para encontrar a melhor trajeto entre dois pontos

4.2.1 Objetivo

Depois da troca de ficheiros da Central para o protótipo de robô foi desenvolvida a parte do código em que os protótipos de robô conseguem encontrar o melhor caminho entre dois pontos, com os dados do protótipo da fábrica fornecidos pela Central.

4.2.2 Resultado

O algoritmo desenvolvido recebe o ponto inicial e o ponto final da rota a executar, encontra os diversos trajetos entre os dois pontos, selecionando o trajeto com menor distância. Este trajeto é apresentando em conjunto com a direção de rotação em cada um dos pontos de decisão. Este algoritmo também apresenta as diversas alternativas ao percurso selecionado.

Durante os testes o autor começou por utilizar um método matemático convencional para o melhor trajeto entre dois pontos [10]. Este método funcionava plenamente se a área de produção do protótipo da fábrica não fosse dinâmica. Como existe a possibilidade de mudar os sentidos de movimento num troço, foi necessário desenvolver um algoritmo e respetiva programação, que encontrasse os percursos entre dois pontos considerando os sentidos definidos dinamicamente.

Figura 4.7 – Simulação de algoritmo de melhor trajeto.

Na Figura 4.7 é ilustrada a forma de indicar ao algoritmo que nos devolva os caminhos possíveis começando o trajeto na Linha 1 e terminando na Linha 3–A. O programa indica os dois caminhos encontrados na configuração do protótipo da fábrica que o protótipo de robô disponha na altura do pedido. Depois é apresentado o caminho mais curto encontrado que seria o utilizado pelo protótipo de robô.

Foi então desenvolvido um algoritmo e respetivo código baseado numa técnica de Otimização Combinatória [11], que encontra os percursos entre dois pontos considerando os sentidos definidos dinamicamente.

O algoritmo utilizado, desenvolvido pelo autor do presente documento, é baseado no algoritmo de Ramificar e Limitar (*Branch and Bound*) [11] [12] [13]. O algoritmo

cria numa estrutura todas as possíveis ramificações desde o ponto de origem até ao ponto de destino. Para isso, inicia no ponto de origem e vai saltando para os pontos adjacentes em função do sentido possível. Sempre que encontra o ponto de destino ou um ponto já visitado não continua a construção de novas ramificações nesse ramo. No caso de um ponto de uma ramificação ter como destino vários pontos o algoritmo recursivamente aplica o mesmo método a todos os destinos para descobrir todas as ramificações. As ramificações cujo ponto final não seja o destino são anuladas. O algoritmo termina após a construção de todas as ramificações e devolve o trajeto associado à menor distância.

Por fim a linha apresenta o sentido que o protótipo de robô deve tomar ao chegar a cada ponto de interseção do tipo "T" ou "+(cruz)", sendo que "R" significa que o protótipo de robô deve virar à direita, "L" deve virar à esquerda e por fim o "F" indica que o protótipo de robô deve seguir em frente.

4.3 Algoritmo de perceção de espaço

4.3.1 Objetivo

O objetivo deste algoritmo é dotar o protótipo de robô de alguma capacidade de inteligência. Este algoritmo está aplicado no modo de demonstração manual. O protótipo de robô pode demonstrar todos os seus movimentos sem colidir com qualquer obstáculo que se encontre na sua rota de demonstração. Sempre que o protótipo de robô se aproxima de um obstáculo reduz a velocidade e caso continue em direção ao obstáculo, para e analisa a área circundante. Conforme o resultado da análise dos sinais dos sensores, o controlador de sensores e atuadores define o melhor movimento a executar para se afastar do obstáculo e recomeçar a demonstração.

O algoritmo desenvolvido é baseado no algoritmo de busca. Quando o protótipo de robô está a efetuar a demonstração de todos os movimentos possíveis com as rodas Mecanum, ao detetar um obstáculo, decide a melhor movimentação a executar para se afastar.

4.3.2 Resultados

O autor teve os resultados esperados nesta demonstração. No entanto, foi necessário fazer alterações às velocidades do protótipo de robô, pois como o protótipo de robô se deslocava a velocidades elevadas, o tempo de reação para objetos mais pequenos não era o suficiente e por vezes tocava nos objetos. Foi mudado o diâmetro exterior das rodas e montados motores com uma velocidade de rotação de referência inferior, para que o protótipo de robô se desloque com menor velocidade, e consiga demonstrar as suas capacidades sem se correr o risco de causar danos por colisões acidentais, durante os testes de desenvolvimento.

Figura 4.8 – Fotos do vídeo de testes de demonstração.

Na Figura 4.8, o protótipo de robô está a demonstrar a sua capacidade de movimento entre dois pontos. Com a funcionalidade de perceção de espaço o robô desloca-se em várias direções sem colidir com qualquer elemento do protótipo de fábrica.

4.4 Teste de mensagens

4.4.1 Objetivo

O autor tomou a decisão de implementar as mensagens trocadas entre os diversos elementos do projeto em modo de texto e não binárias. Esta decisão foi tomada de forma a ser mais fácil durante os testes e ensaios perceber o conteúdo que está a ser enviado e recebido por cada um dos elementos.

A estrutura das mensagens é semelhante em todo o projeto, e é apresentada na Figura 4.9.

Elemento origem da mensagem	Função/Tarefa pedida	Dados da função/tarefa pedida
AGV0001|	**Hello|**	**IP:192.168.3.1|Port:5001|**

Figura 4.9 – Formato e exemplo de mensagens.

O exemplo da mensagem apresentada na Figura 4.9, indica que o protótipo de robô 1 (**AGV0001**) acabou de ser ligado e está a indicar à Central o Arranque (**Hello**) e que a comunicação com ele será feita através do seu endereço IP (**192.168.3.1**), e que a porta de comunicação que está à escuta de mensagens é a (**5001**).

4.4.2 Resultado

No início dos testes foi verificado que eram recebidos caracteres especiais no início de cada mensagem (b' (mensagem esperada)), teve de ser feita uma adaptação na codificação da mensagem para não ser recebido este caractere.

As mensagens tal como esperado são de fácil perceção depois de estudar a sua construção.

5 CONCLUSÕES E TRABALHOS FUTUROS

Neste capítulo são apresentadas as conclusões do projeto, onde são indicadas as dificuldades encontradas, as soluções de resolução, e os resultados obtidos. São também apresentadas diversas propostas de trabalhos futuros.

5.1 Conclusões

Neste trabalho foi desenvolvido pelo autor o projeto de três protótipos de robô que permitem simular e testar o funcionamento de AGV e de AMR. Estes protótipos foram sofrendo diversas alterações durante o projeto devido à possibilidade de outro mestrando pretender fazer o guiamento dos protótipos de robô por visão artificial, o que levou a colocar um controlador de nível superior. Com a utilização do Raspberry PI aumentaram as potencialidades na programação dos protótipos de robô. Com esta capacidade de processamento foi possível melhorar a eficácia e gestão dos protótipos de robô.

Como em qualquer projeto existiram dificuldades que tiveram de ser ultrapassadas. Apresentam-se nesta secção as principais dificuldades, para permitir futuros desenvolvimentos da plataforma por terceiros, de modo a não serem cometidos os mesmos erros, que levaram a este projeto ser mais longo que o esperado.

A primeira dificuldade foi a comunicação entre os protótipos de robô e a Central. O autor recorreu a sockets de comunicação convencionais. Foram realizados diversos testes em que no caso de não existir nenhuma paragem do socket, as comunicações funcionavam perfeitamente. Mas sempre que se parava o socket de um dos lados para restabelecer novamente a comunicação, tinha de se reiniciar o programa de comunicação, dos dois lados. Como a solução era complexa e após conversa com os orientadores, apresentando e demonstrando o problema, efetuaram-se com sucesso testes recorrendo ao método definitivo usado no projeto, que foram os websockets. Com estas ferramentas foi efetuado um teste de comunicação dos três protótipos de robô a enviar a cada 100 ms uma mensagem para a Central, e durante oito horas de teste não houve qualquer interrupção ou falha de comunicação.

Depois de implementar os websockets, o autor deparou-se com outro problema. Como os websockets funcionavam de forma assíncrona, e desta forma os programas principais ficavam sempre à espera de mensagens para continuarem o seu fluxo, não executavam as tarefas de análise e controlo de forma cíclica como pretendido. Optou-se então por usar os métodos de filas de dados, em que deste modo é garantido que nenhuma mensagem é perdida, e todos os dados são analisados e controlados de forma cíclica num programa principal.

No desenvolvimento do projeto a construção dos protótipos de robô foi efetuada em plástico PLA. Ao fim de algum tempo detetou-se um problema de construção mecânica: com o tempo, com a realização dos testes e ensaios, as peças de suporte

dos motores e rodas foram-se deformando, devido ao peso excessivo da estrutura base, resultante das opções tomadas na fabricação e construção. Foi então simplificada e refeita a estrutura base dos protótipos de robô para reduzir o peso dos protótipos de robô. O protótipo de robô que já tinha um peso de 1800 g passou a cerca de 900 g. Para isso foram removidos alguns elementos que não eram necessários, por exemplo:

Os protótipos de robô dispunham de um *power bank* (600 g) para permitir que o Raspberry PI parasse o sistema de forma correta e segura e um conjunto de 3 baterias 18650 (200 g) para alimentar todos os outros elementos, as cablagens e ligadores necessários para estes dois elementos alimentarem o protótipo de robô (100 g), os suportes em PLA (100 g) e os parafusos e porcas de fixação (100 g). Optou-se por colocar uma bateria (300 g) que alimenta todos os elementos do protótipo de robô e permitir que o Raspberry PI seja desligado de forma direta sem qualquer programa de *background* a encerrá-lo de forma segura. Redução dos sensores de obstáculo para metade (5 g) - existiam dois em cada face e passou a existir somente um sensor por face, cablagens (5 g) suportes em PLA (80 g) e parafusos e porcas de fixação (10 g).

Após realizar este projeto o autor considera que todos os objetivos a que se propôs foram atingidos.

Um dos objetivos principais era a implementação da interface gráfica simples e intuitiva de utilizar, o que foi realizado com sucesso.

Outro dos objetivos que o autor se propunha neste projeto era desenvolver a flexibilização dos trajetos e da funcionalidade das frotas. Este objetivo foi demonstrado e realizado com sucesso, pois podemos verificar que um protótipo de robô deixa de estar limitado a um trajeto simples de um ponto A para um ponto B de uma instalação. Com o potencial de aplicação de algoritmos inteligentes, e usando o conhecimento pelos protótipos de robô do ambiente da área de produção, facilmente estes podem ser utilizados em qualquer tarefa. Podem, por exemplo, estar numa hora a transportar peças da linha de produção 1 para a linha produção 2, como na hora seguinte estar a reforçar a frota e ajudar outros protótipos de robô a transportar peças da linha de produção 3 para a linha de produção 2.

Com o desenvolvimento deste projeto e a experiência profissional nesta área, o autor conclui que na área da robótica e da automação, com a evolução da tecnologia, as tarefas e funções a desempenhar tendem sempre a ser melhoradas e otimizadas.

Neste projeto foi desenvolvida, construída e testada uma plataforma constituída por três protótipos de robô móveis e uma estação Central de monitorização e controlo, associados a um protótipo de uma fábrica para simular percursos e atividades a executar.

Foi projetada e construída a estrutura mecânica, selecionado e montado o hardware e respetivos circuitos de alimentação incluindo as baterias, e programado o software de monitorização, de controlo e de comunicações. Esta plataforma foi testada com funções de demonstração e possui a flexibilidade para permitir desenvolver e testar

novas funcionalidades a implementar em ambiente industrial real, sem a necessidade da existência de robôs de tamanho real nem interromper o fluxo de produção.

Como referido anteriormente esta plataforma tem como um dos objetivos a utilização em ambiente académico. Na secção seguinte são apresentadas possíveis melhorias e trabalhos a realizar nesta plataforma.

5.2 Trabalhos futuros

O autor apresenta nesta secção as possíveis melhorias que identificou, e que podem ser realizadas num futuro próximo.

As melhorias que podem ser implementadas nos protótipos de robô são relativamente simples e das mesmas podem surgir diversas possibilidades de aplicação industrial e de investigação e desenvolvimento:

- Aplicação de *encoders* nas rodas omnidirecionais Mecanum e usando os seus sinais, programar a cinemática e otimizar os movimentos.
- Aplicação de sensores LiDAR para implementação de funcionalidades de segurança e melhor gestão no mapeamento.
- Desenvolver algoritmos de mapeamento, processamento e partilha de ambiente real da área de produção entre os protótipos de robô.
- Utilizar o Arduino como escravo Modbus de entradas e saídas, permitindo que todo o processamento e controlo de dados do protótipo de robô passe a estar centrado no Raspberry PI.
- Aplicação de câmaras e desenvolver o guiamento por visão artificial, para referenciamento por localização de pontos chave. Podem ser usados códigos QR que podem ser uma alternativa ao sistema de localização por RFID.
- Desenvolver uma página web que permita mover os protótipos de robô de forma manual, com um dispositivo externo como um telemóvel, tablet ou computador.

Podem ser implementadas na Central outras funções para flexibilizar a sua utilização:

- Criação de uma base de dados para partilhar com os protótipos de robô, em vez dos ficheiros Excel utilizados neste projeto.
- Otimização e melhoria do controlo das tarefas de teste.
- Criar uma ferramenta que permita enviar o código fonte para todos os protótipos de robô de forma simples a partir da Central, sem ter de estar a colocar os ficheiros manualmente em cada um dos protótipos.

- Aplicação de algoritmos de *Machine Learning* e redes neuronais para dotar os protótipos de robô de Inteligência Artificial;

6 REFERÊNCIAS BIBLIOGRÁFICAS

[1] R. A. F. Romero, E. Prestes, F. Osório e D. Wolf, Robótica Móvel, Rio de Janeiro: LTC - Livros Técnicos e Cientificos Editora Ltda., 2014.

[2] G. Simons, Introdução à Inteligência Artificial, Lisboa: Clássica Editora, 1984.

[3] R. P. Foundation, "Rasperry PI," [Online]. Available: https://www.raspberrypi.com/. [Acedido em maio de 2025].

[4] F. &. D. I. Adascalitei, "Practical applications for mobile robots based on Mecanum wheels - a systematic survey.," *Romanian Review Precision Mechanics, Optics and Mechatronics.,* nº 40, pp. 21-29, 2011.

[5] A. Inc., "Arduino," [Online]. Available: https://www.arduino.cc/. [Acedido em maio de 2025].

[6] A. Augustin e contributors, "Websockets library," [Online]. Available: https://websockets.readthedocs.io/en/stable/. [Acedido em maio de 2025].

[7] E. Costa, Programação em Python - Fundamentos e Resolução de Problemas, Lisboa: FCA - Editora de Informática, Lda, 2015.

[8] F. Becker, O. Bondarev, I. Zeidis, K. Zimmermann, M. Abdelrahman e B. Adamov, "An approach to the kinematics and dynamics of a four-wheeled mecanum vehicles," *Problems of Mechanics,* nº 2, pp. 27-37, 2014.

[9] L. Abreu, HMTL 5, Lisboa: FCA - Editora de Informática, Lda, 2011.

[10] E. Martins, M. Pascoal, D. Rasteiro e J. Santos, "The Optimal Path Problem.," *Investigação Operacional,* nº 19, pp. 43-60, 1999.

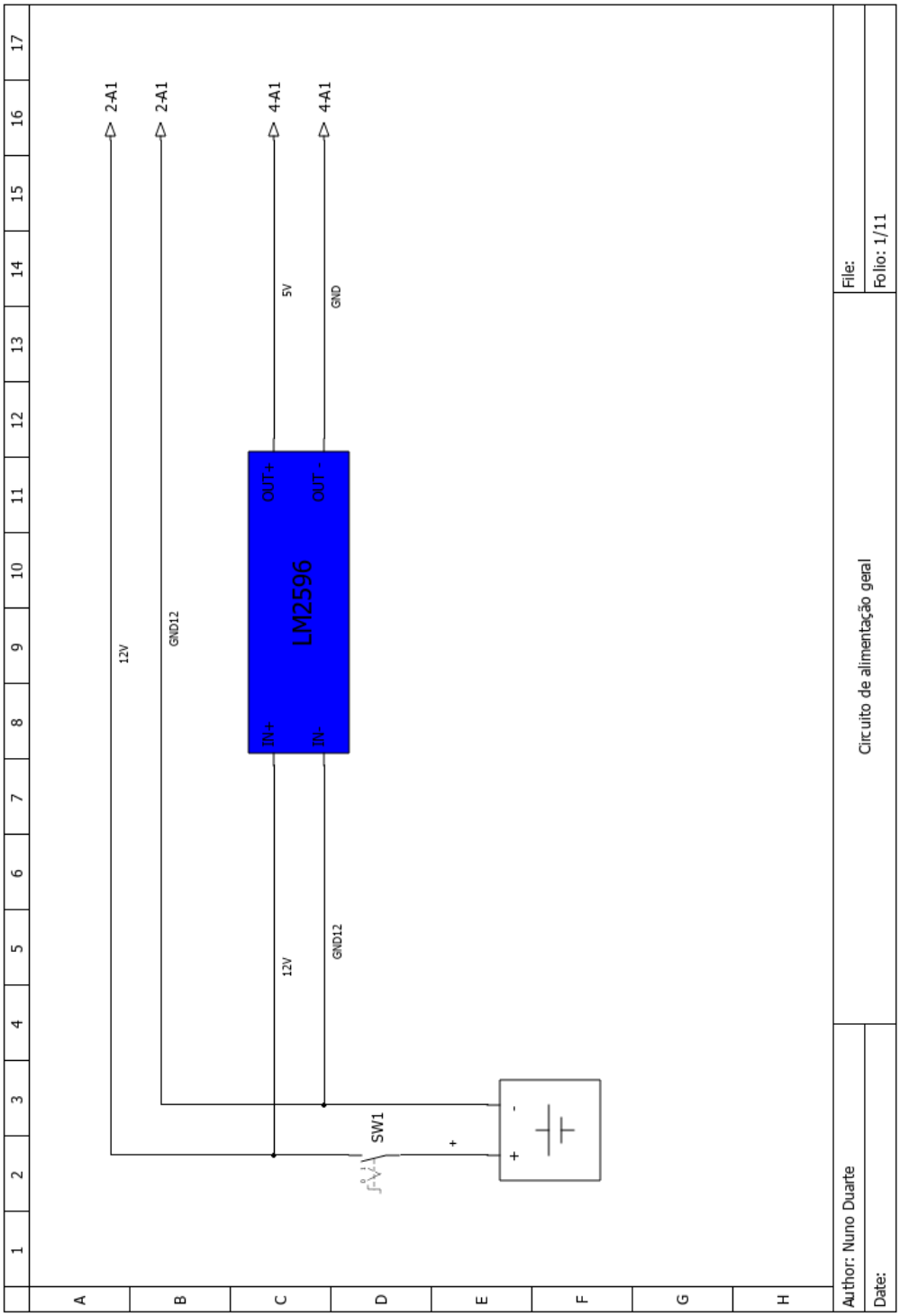
[11] J. V. Bernhard Korte, Combinatorial Optimization, Springer, 2018.

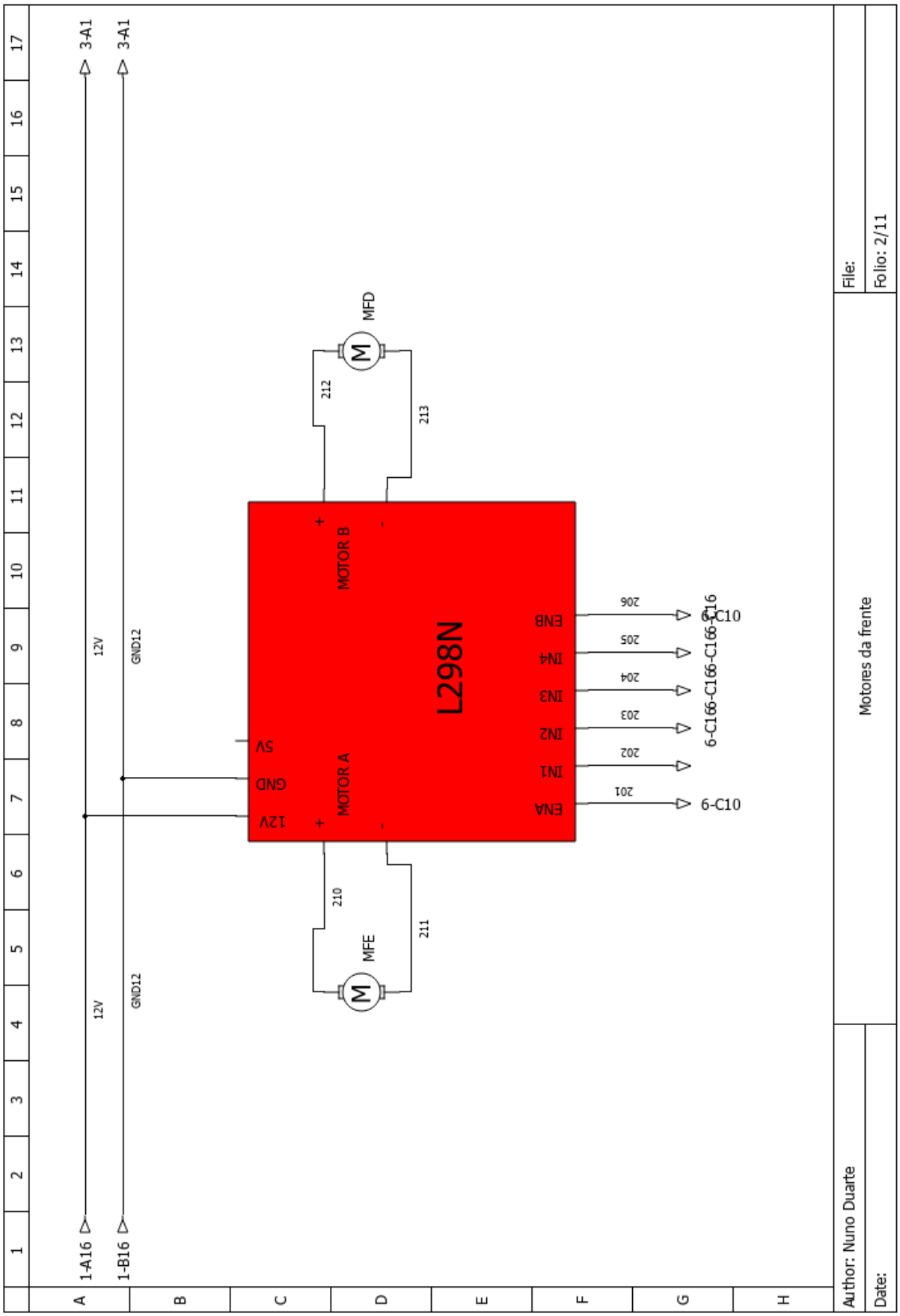
[12] J. Binney e G. S. Sukhatme, "Branch and bound for informative path planning," *2012 IEEE International Conference on Robotics and Automation, Saint Paul, MN, USA, doi: 10.1109/ICRA.2012.6224902.,* pp. 2147-2154, 2012.

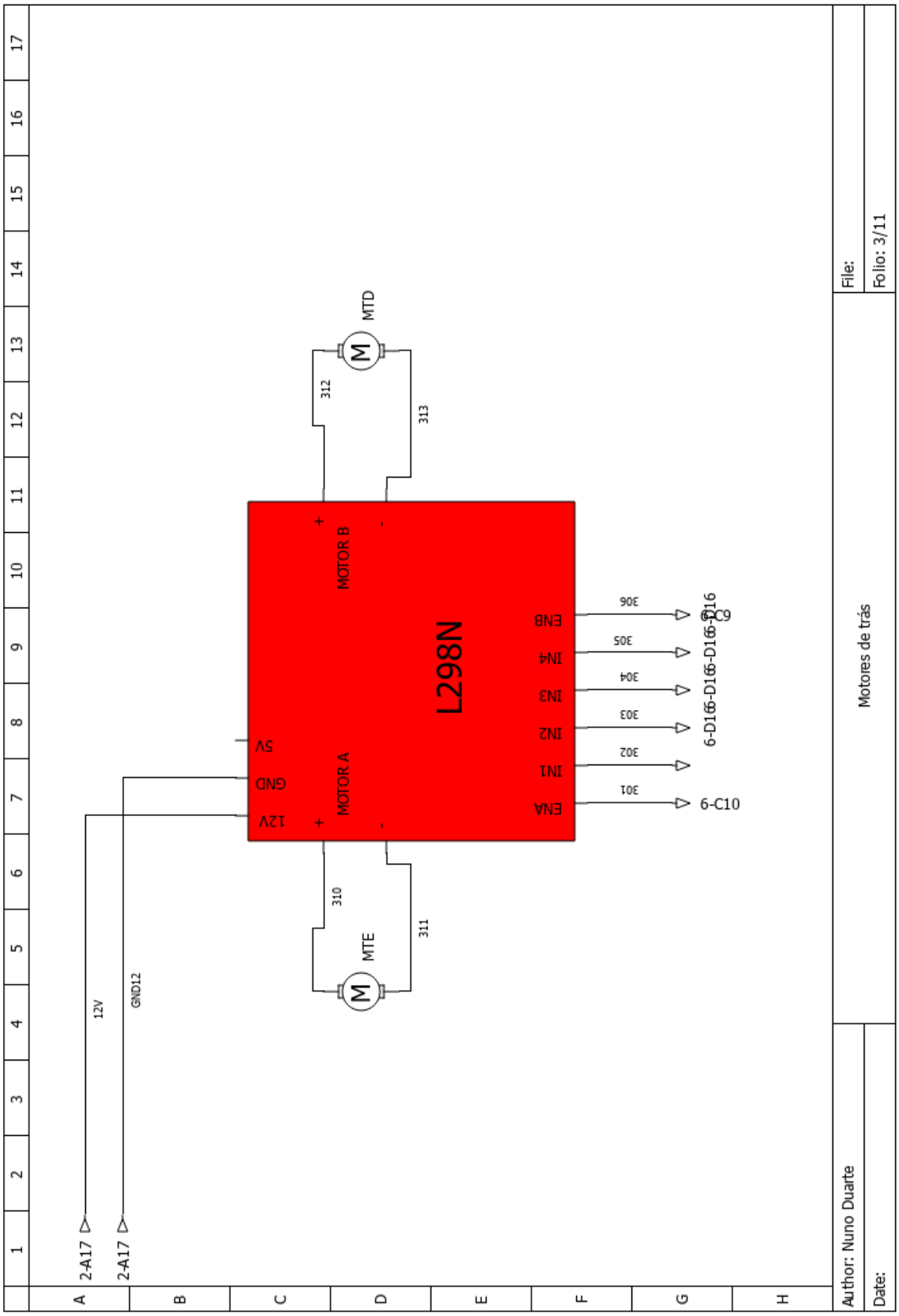
[13] M. Hasan, M. A. Haque, S. M. Mominuzzaman e T. G. Troyee, "Optimization of CDPR Path Planning using a Branch and Bound Algorithm," *2023 6th International Conference on Electrical Information and Communication Technology (EICT), Khulna, Bangladesh, doi: 10.1109/EICT61409.2023.10427901.,* pp. 1-6, 2023.

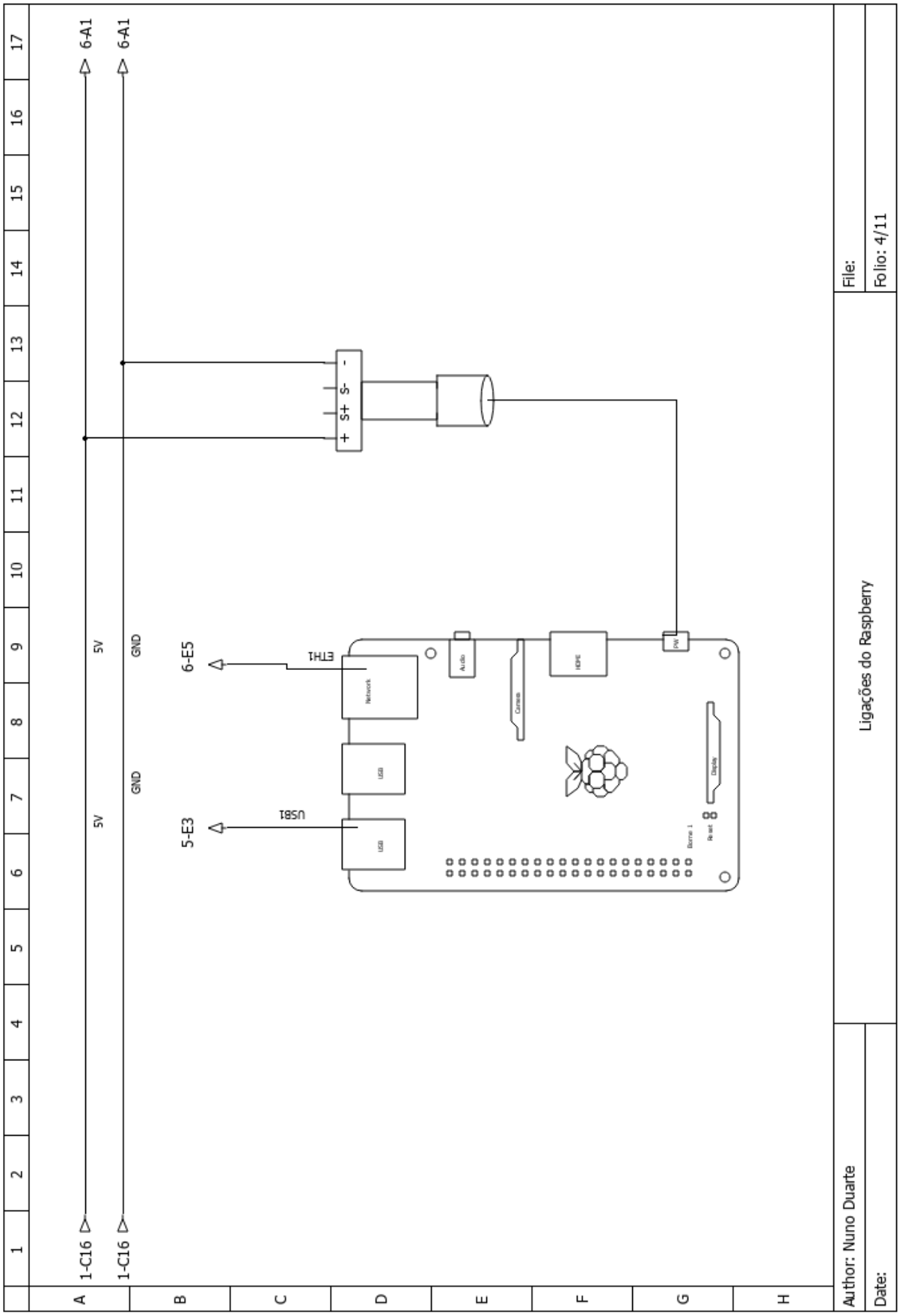
ANEXOS

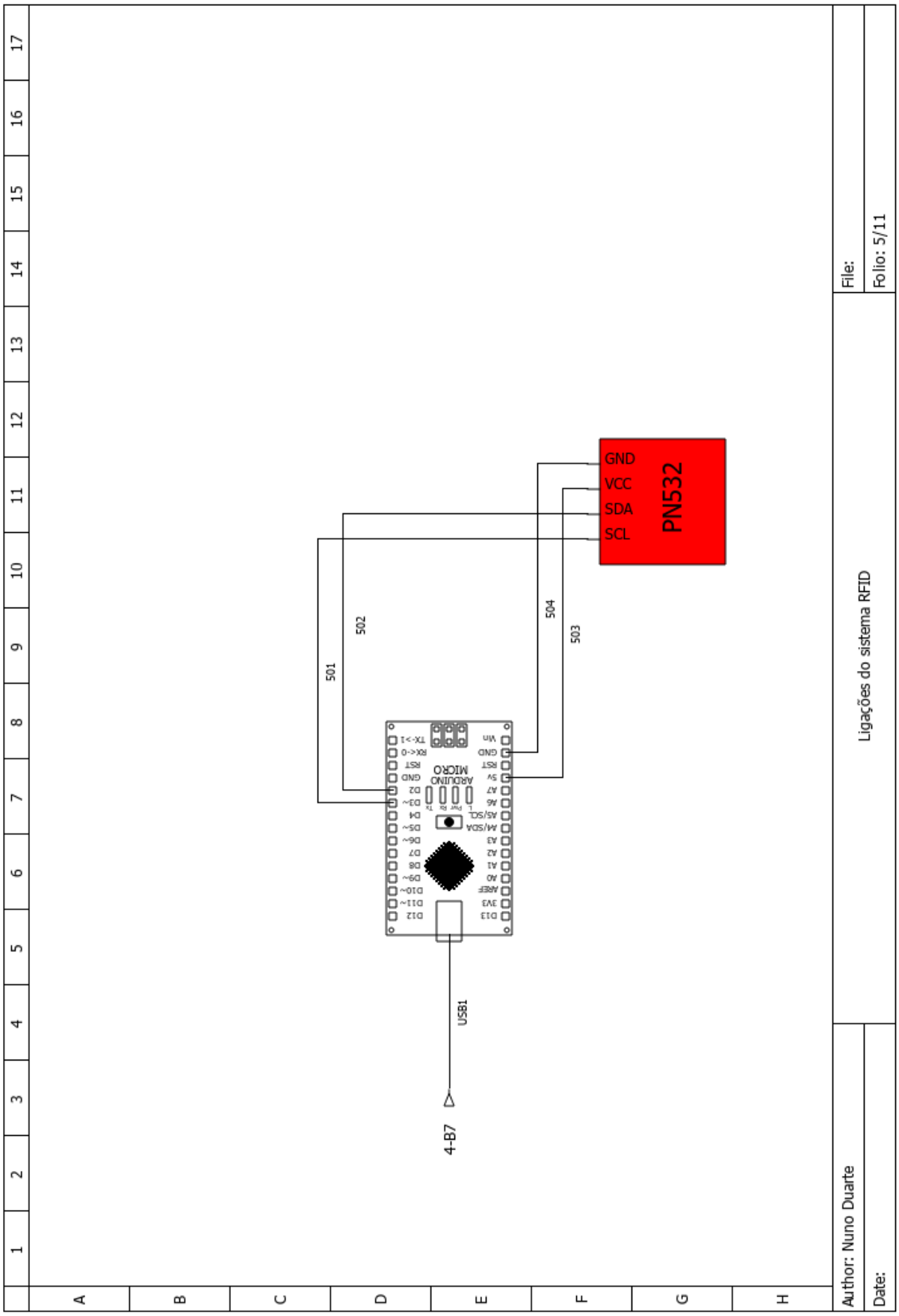
Esquema Elétrico do protótipo do robô

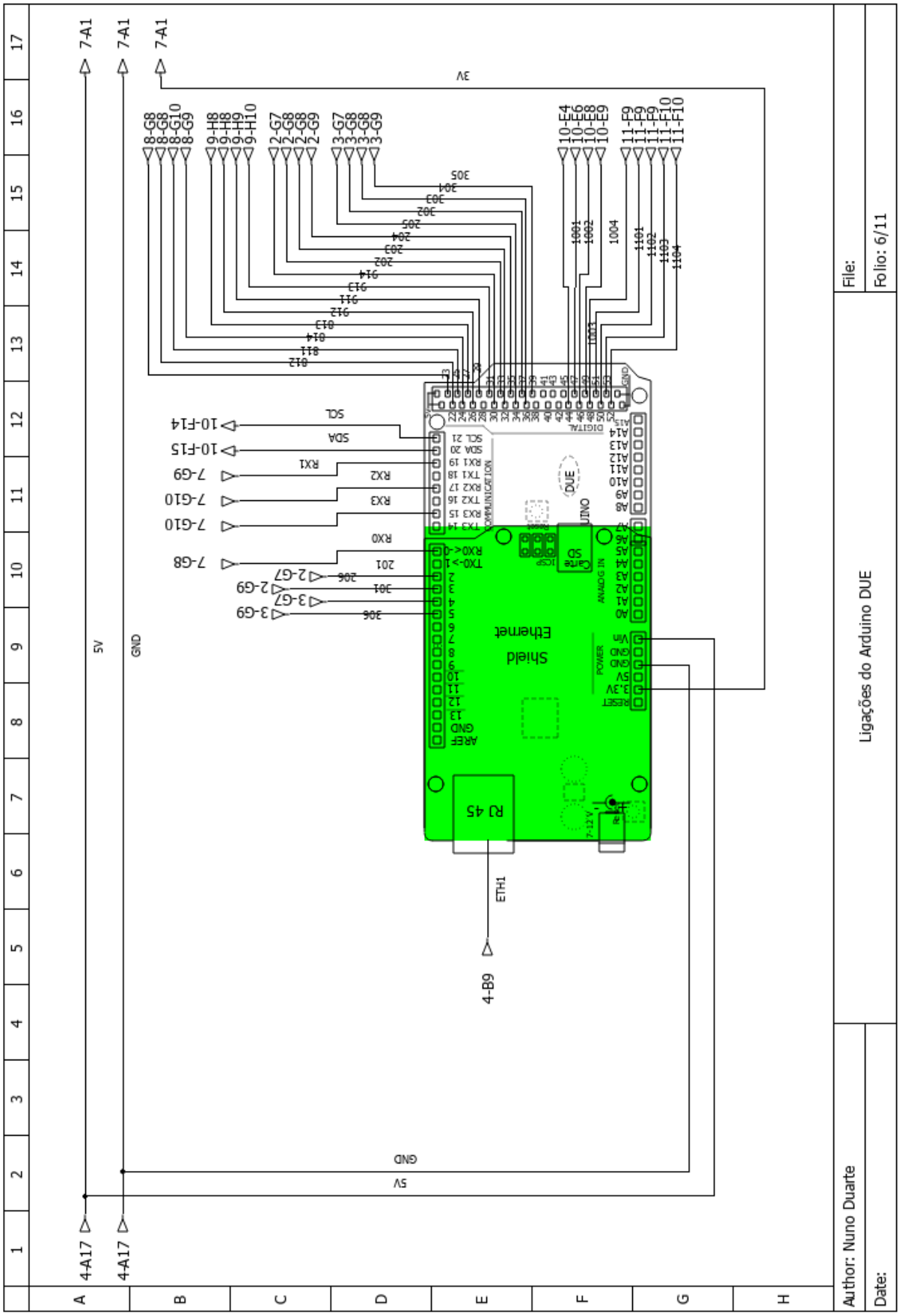


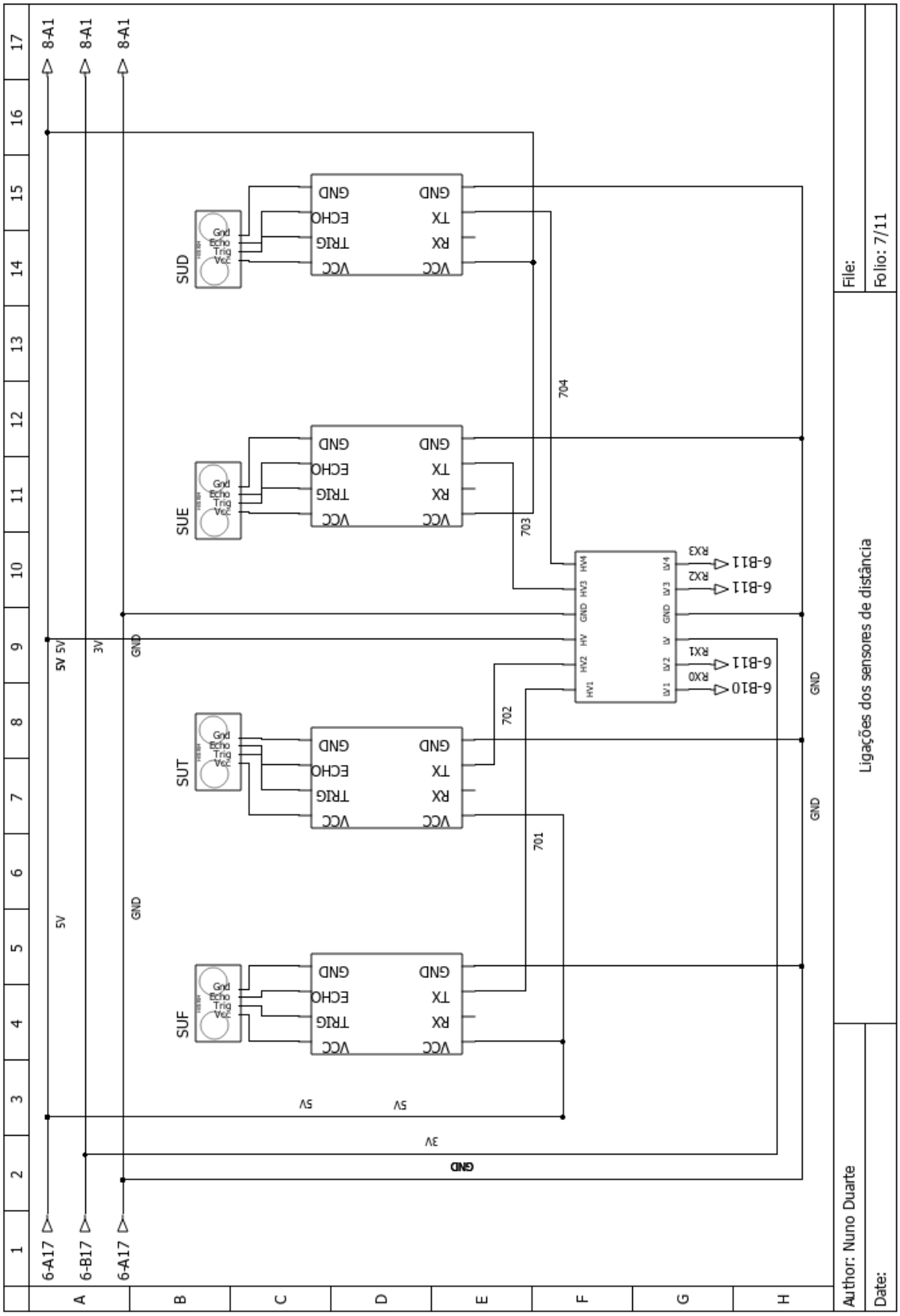

3-A1
3-A1
MFD
212
213
MOTOR B
L298N
206
205
204
203
202
201
6-C166-C166-C166-C166
6-C10
6-C10
ENB
IN4
IN3
IN2
IN1
ENA
5V
GND
12V
MOTOR A
12V
GND12
12V
GND12
210
211
MFE
1-A16
1-B16
Author: Nuno Duarte
Date:
Motores da frente
File:
Folio: 2/11

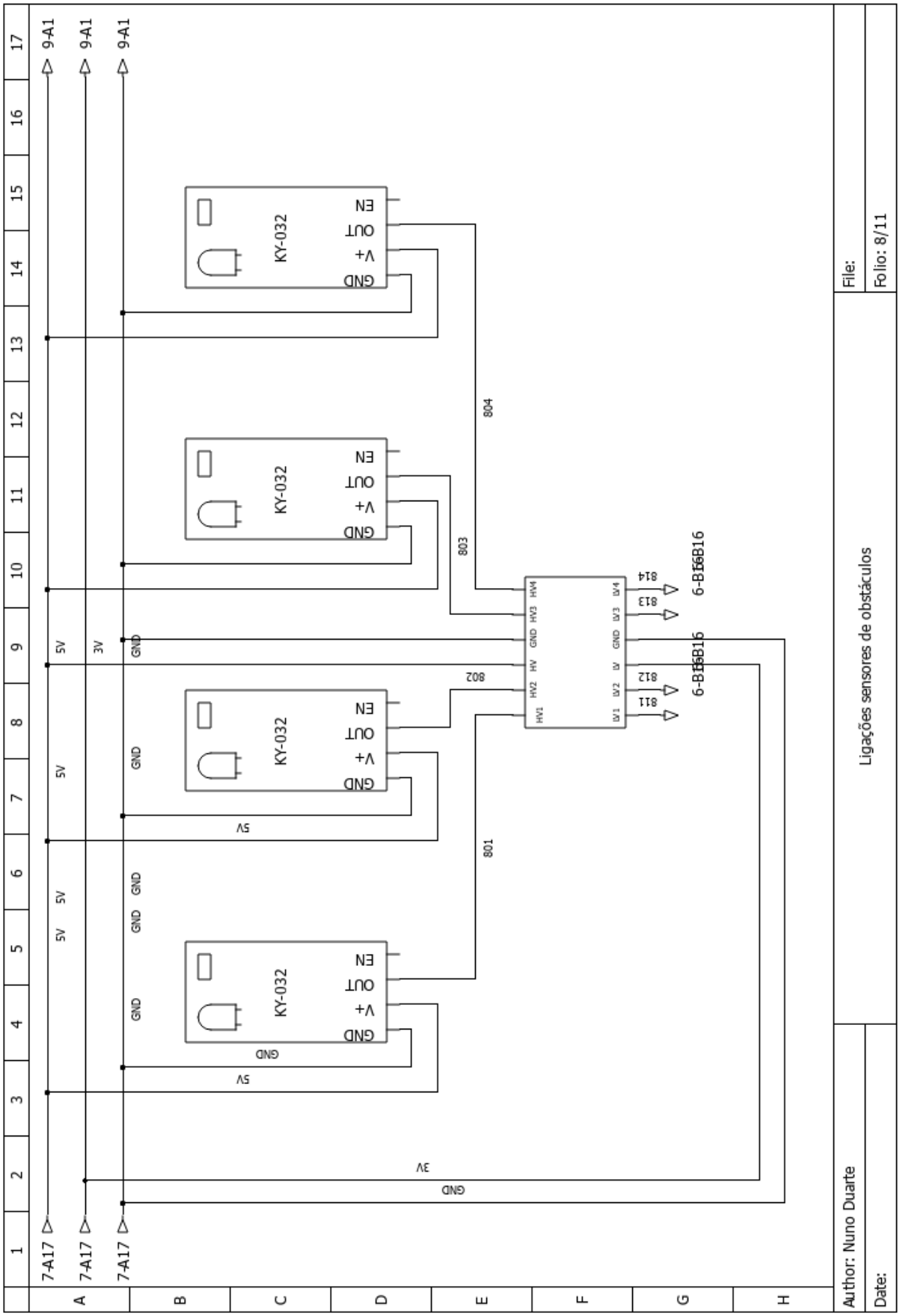

L298N
MOTOR A
MOTOR B
12V
GND
5V
ENA
IN1
IN2
IN3
IN4
ENB
MTD
MTE
312
313
310
311
306
305
304
303
302
301
6-C10
6-D166-D166-D166-D166-C9
2-A17
2-A17
12V
GND12
Motores de trás
File:
Folio: 3/11
Author: Nuno Duarte
Date:

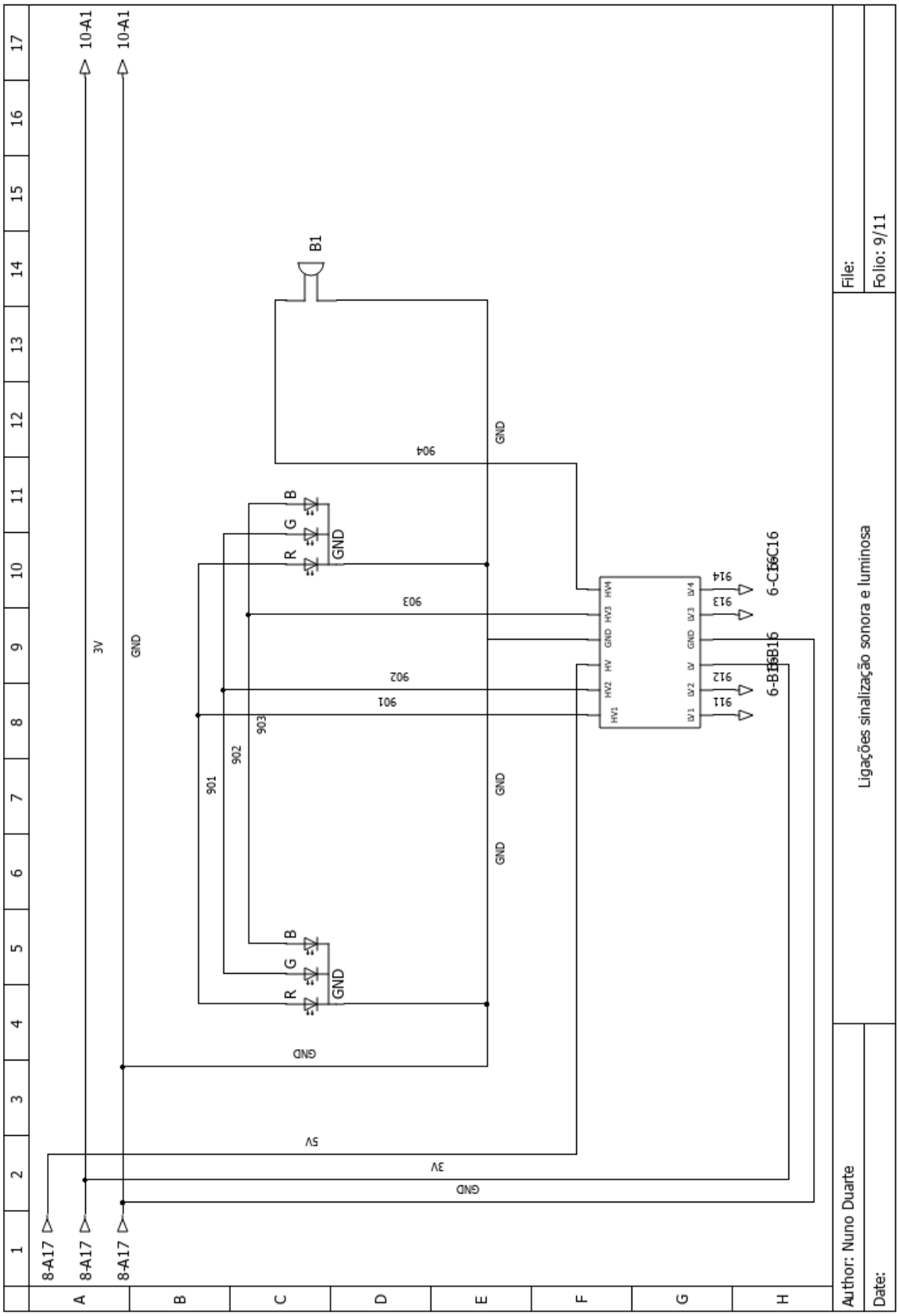

6-A1
6-A1
1-C16
1-C16
5V
5V
GND
GND
6-E5
ETH1
5-E3
USB1
+ S+ S- -
Author: Nuno Duarte
Date:
Ligações do Raspberry
File:
Folio: 4/11

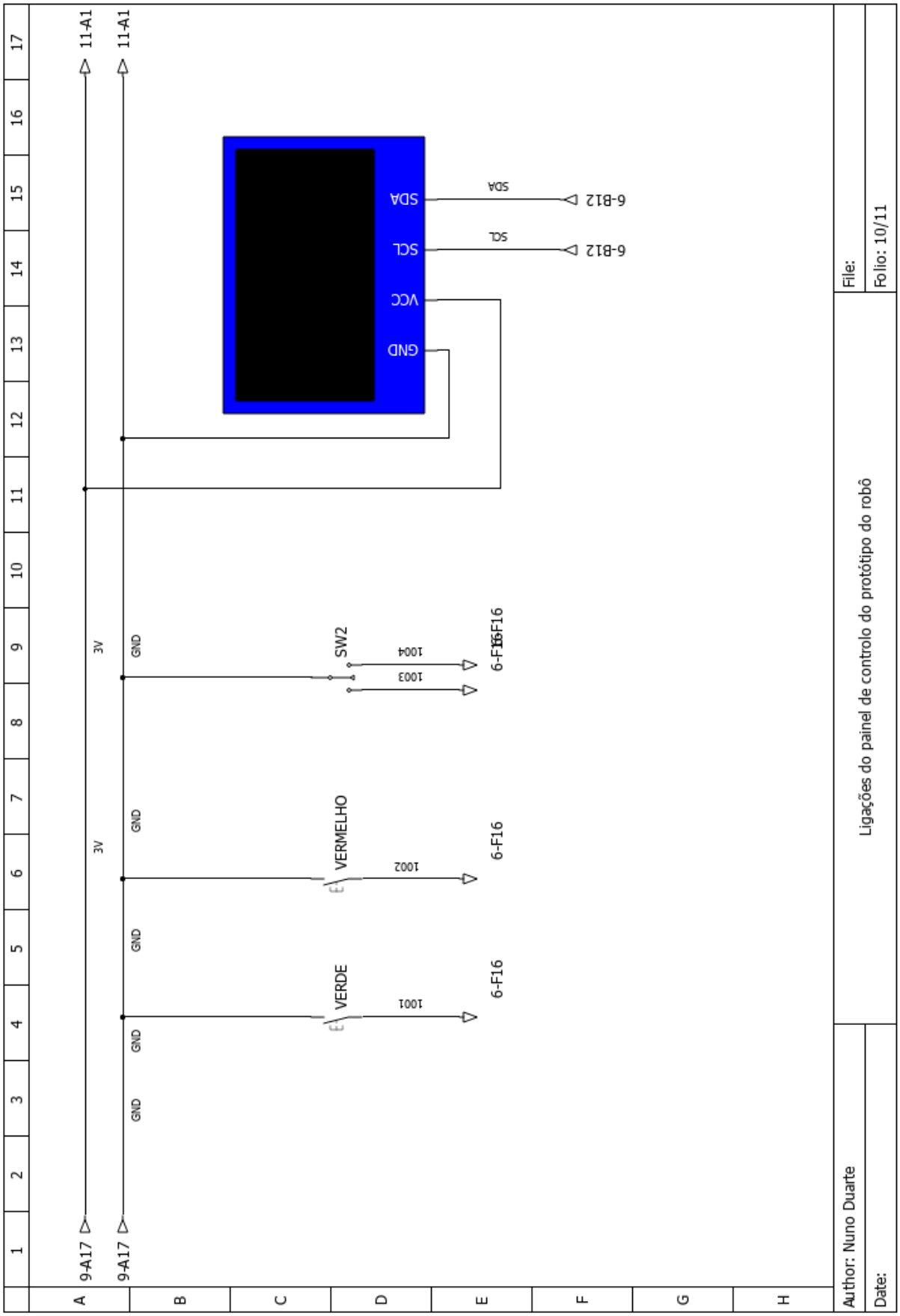

GND
VCC
SDA
SCL
PN532
501
502
503
504
USB1
4-B7
Ligações do sistema RFID
Author: Nuno Duarte
Date:
File:
Folio: 5/11

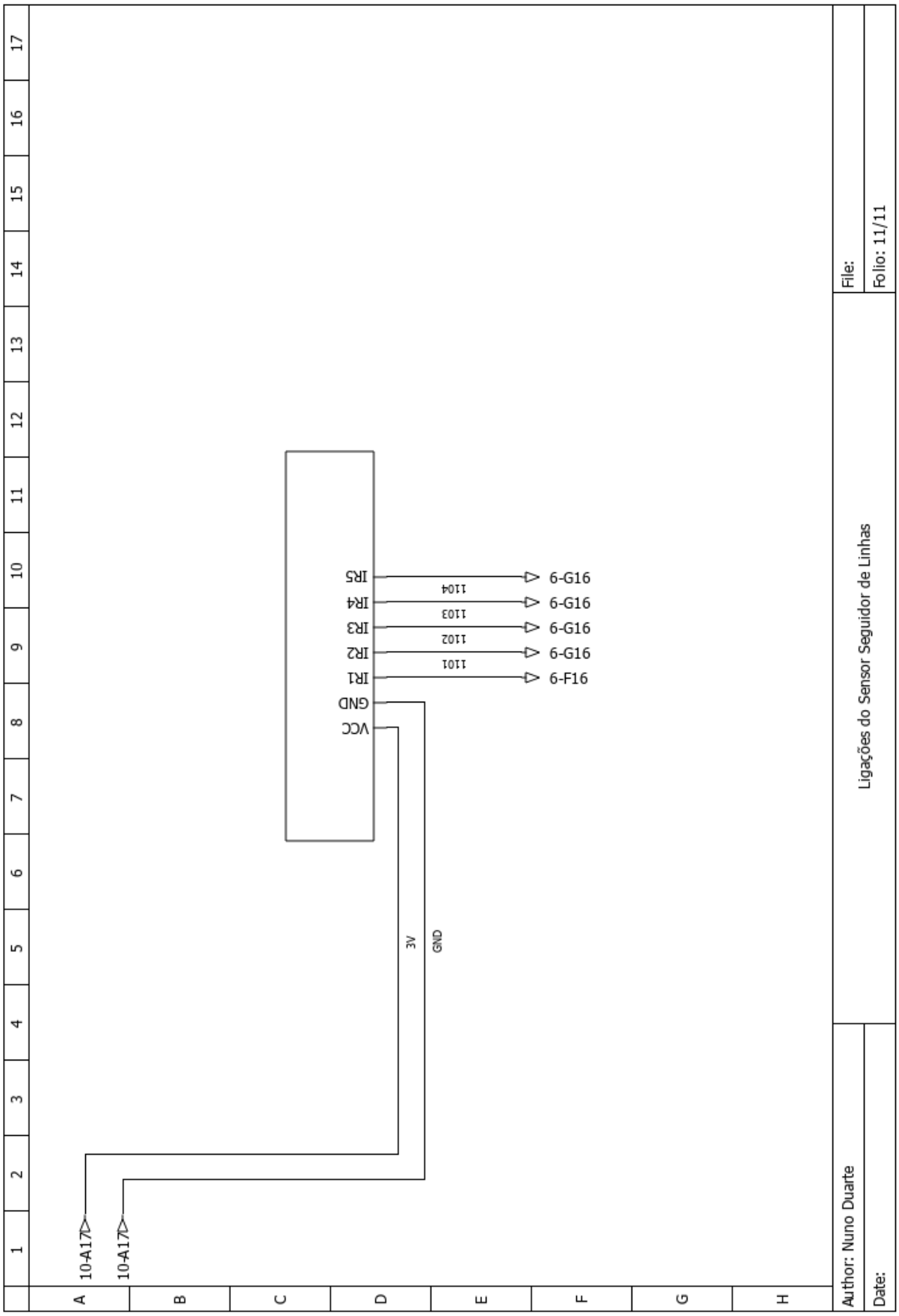

Ligações do Arduino DUE
Author: Nuno Duarte
Date:
File:
Folio: 6/11
Shield
Ethernet
RJ 45
ETH1
4-B9
7-A1
4A17
5V
GND
3V
SCL
SDA
RX1
RX2
RX3
RX0
10-F14
10-F15
7-G9
7-G10
7-G10
7-G8
2-G7
2-G9
3-G7
3-G9
8-G8
8-G8
8-G10
8-G9
9-H8
9-H8
9-H9
9-H10
2-G7
2-G8
2-G8
2-G9
3-G7
3-G8
3-G8
3-G9
10-E4
10-E6
10-E8
10-E9
11-F9
11-F9
11-F9
11-F10
11-F10


8-A1
8-A1
8-A1
6-A17
6-B17
6-A17
SUD
SUE
SUT
SUF
VCC
TRIG
ECHO
GND
TX
RX
704
703
702
701
5V
3V
GND
6-B11
6-B10
HV1
HV2
HV
GND
HV3
HV4
LV1
LV2
LV
GND
LV3
LV4
RX0
RX1
RX2
RX3
Ligações dos sensores de distância
Author: Nuno Duarte
Date:
File:
Folio: 7/11

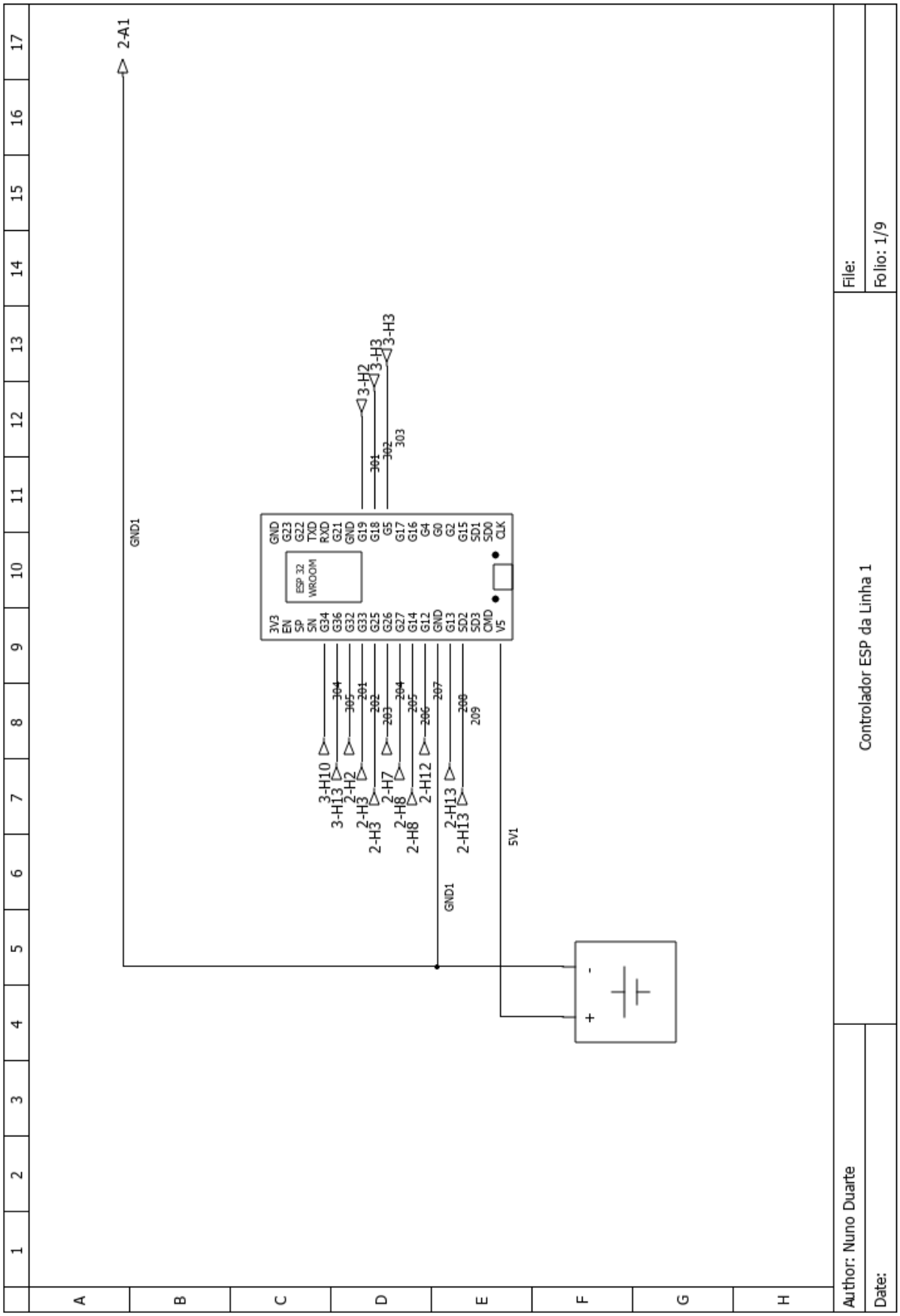

KY-032
GND
V+
OUT
EN
804
803
802
801
5V
3V
GND
9-A1
7-A17
Ligações sensores de obstáculos
Author: Nuno Duarte
Date:
File:
Folio: 8/11

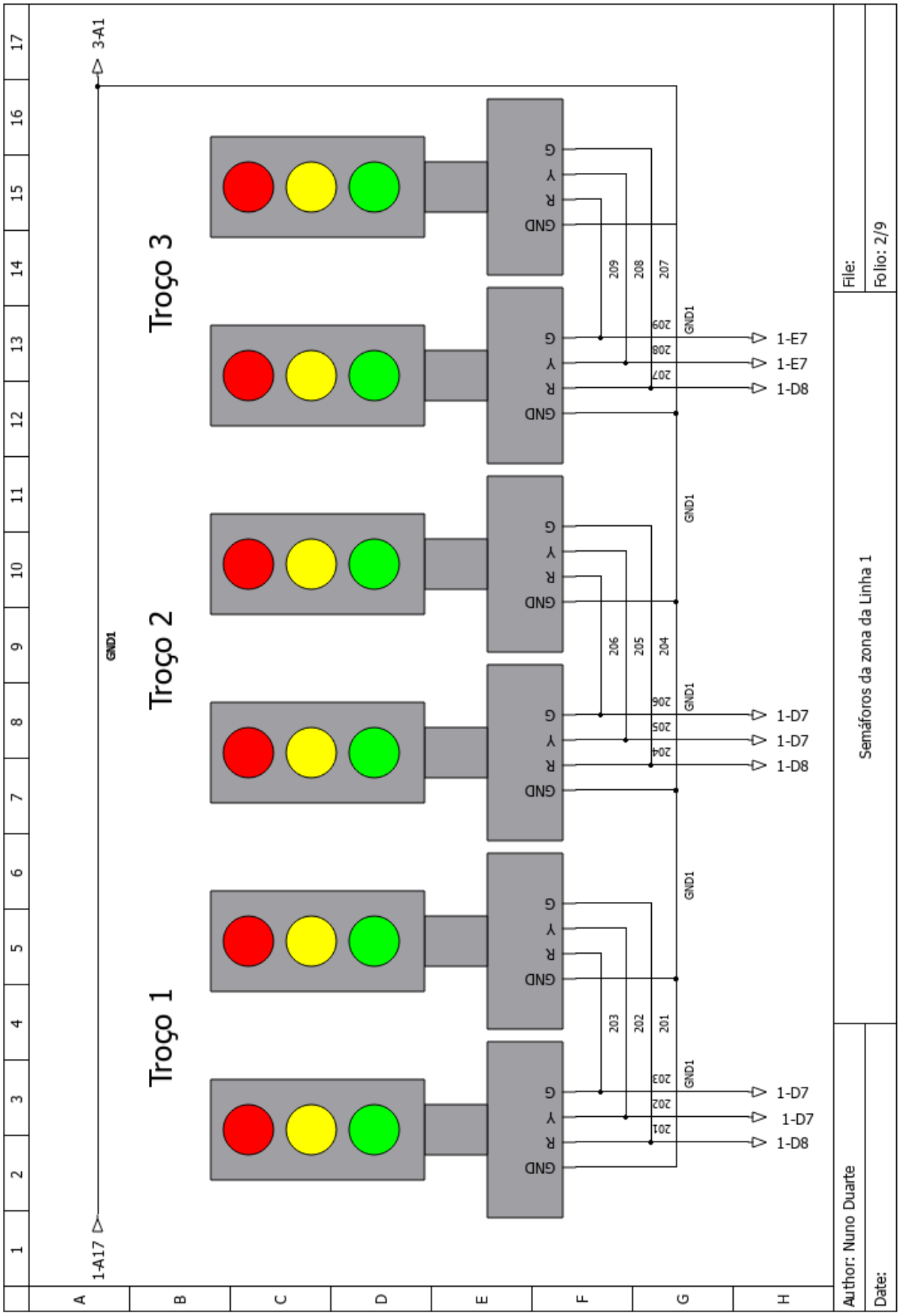

Ligações sinalização sonora e luminosa
Author: Nuno Duarte
Date:
File:
Folio: 9/11
8-A17
10-A1
3V
GND
5V
B1
R
G
B
901
902
903
904
911
912
913
914

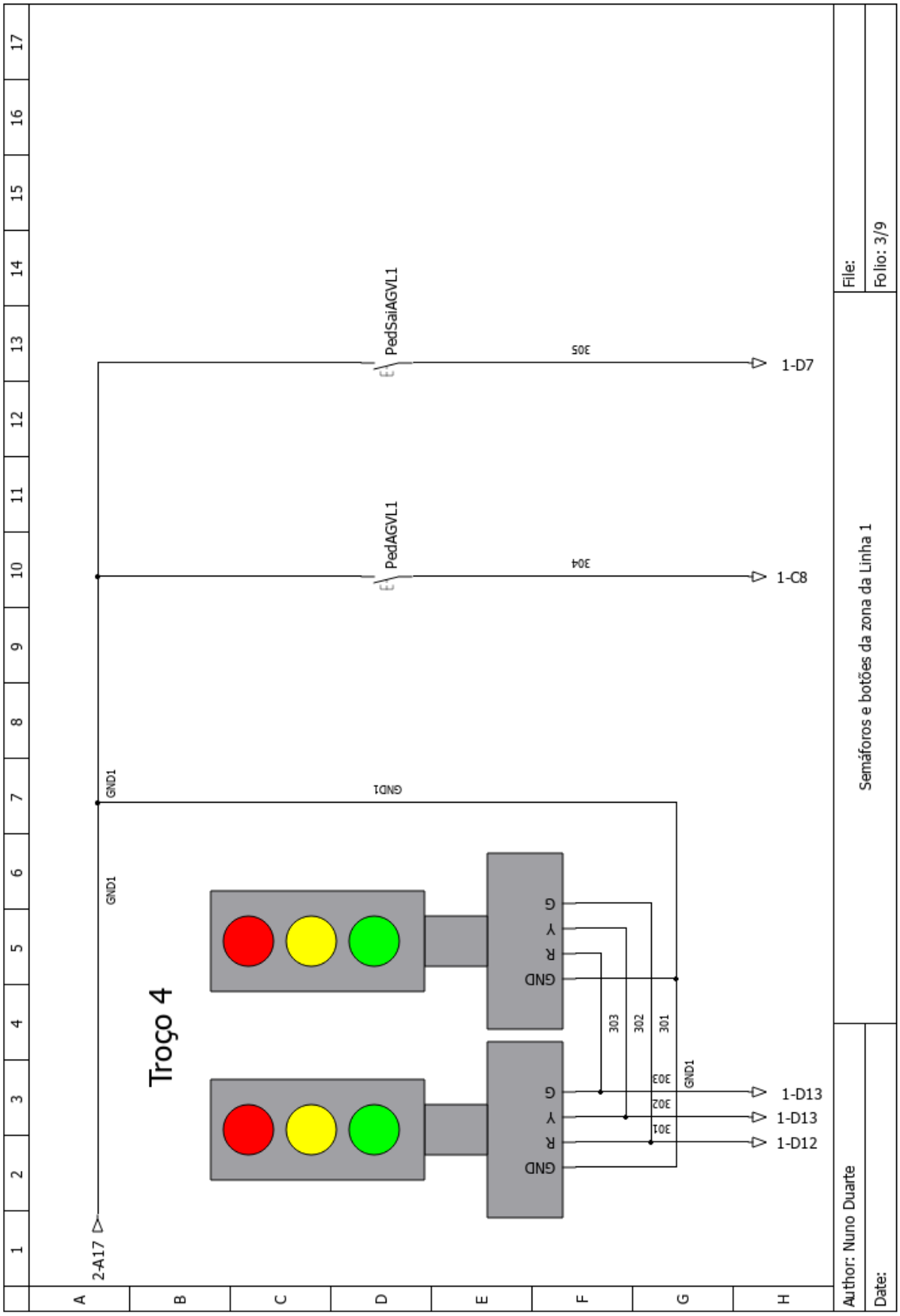

11-A1
11-A1
SDA
SCL
VCC
GND
SDA
SCL
6-B12
6-B12
3V
3V
GND
GND
GND
GND
GND
SW2
1004
1003
6-F16
VERMELHO
1002
6-F16
VERDE
1001
6-F16
9-A17
9-A17
Ligações do painel de controlo do protótipo do robô
Author: Nuno Duarte
Date:
File:
Folio: 10/11

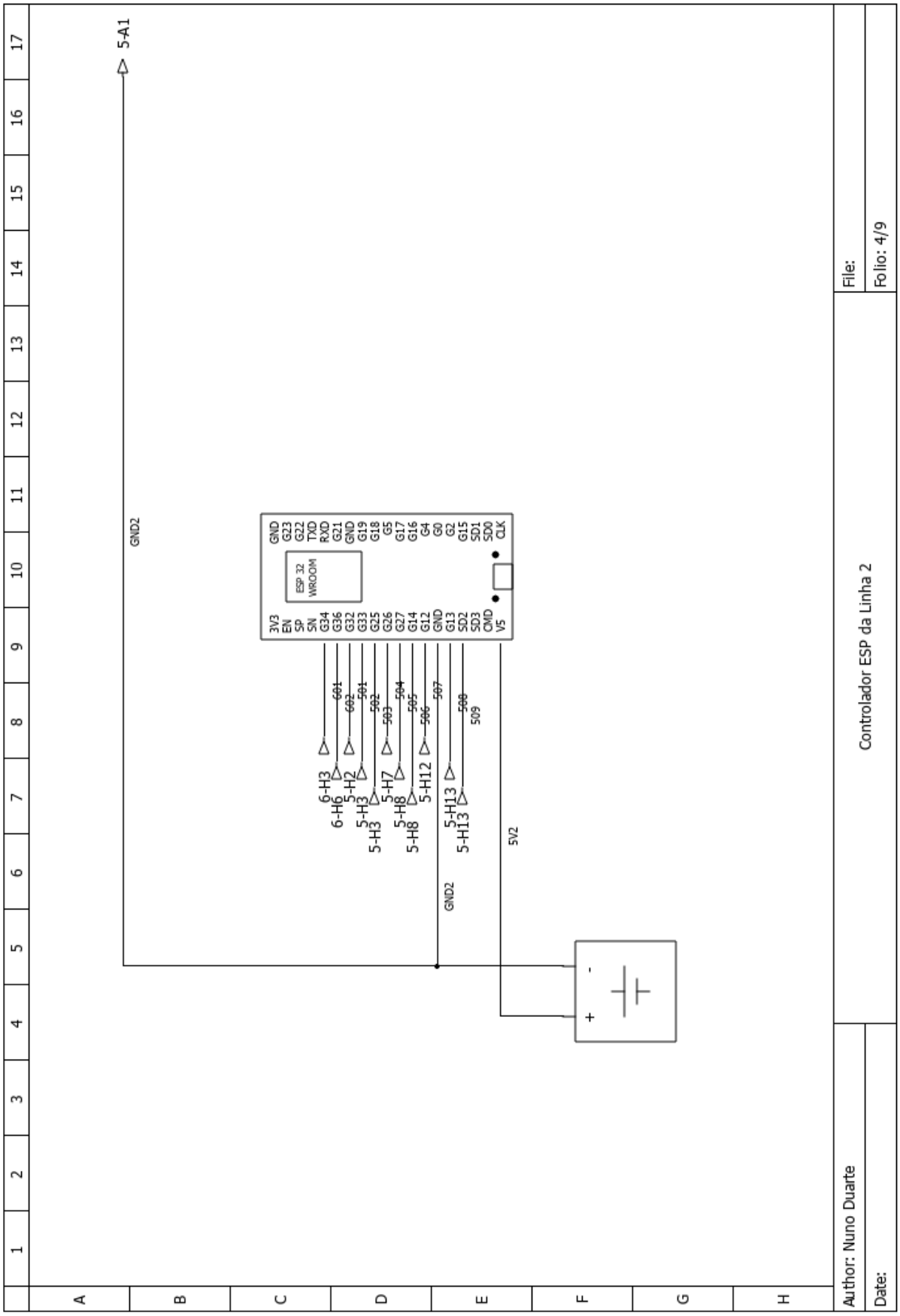

Ligações do Sensor Seguidor de Linhas
Author: Nuno Duarte
Date:
File:
Folio: 11/11
10-A17
10-A17
3V
GND
VCC
GND
IR1
IR2
IR3
IR4
IR5
1101
1102
1103
1104
6-F16
6-G16
6-G16
6-G16
6-G16

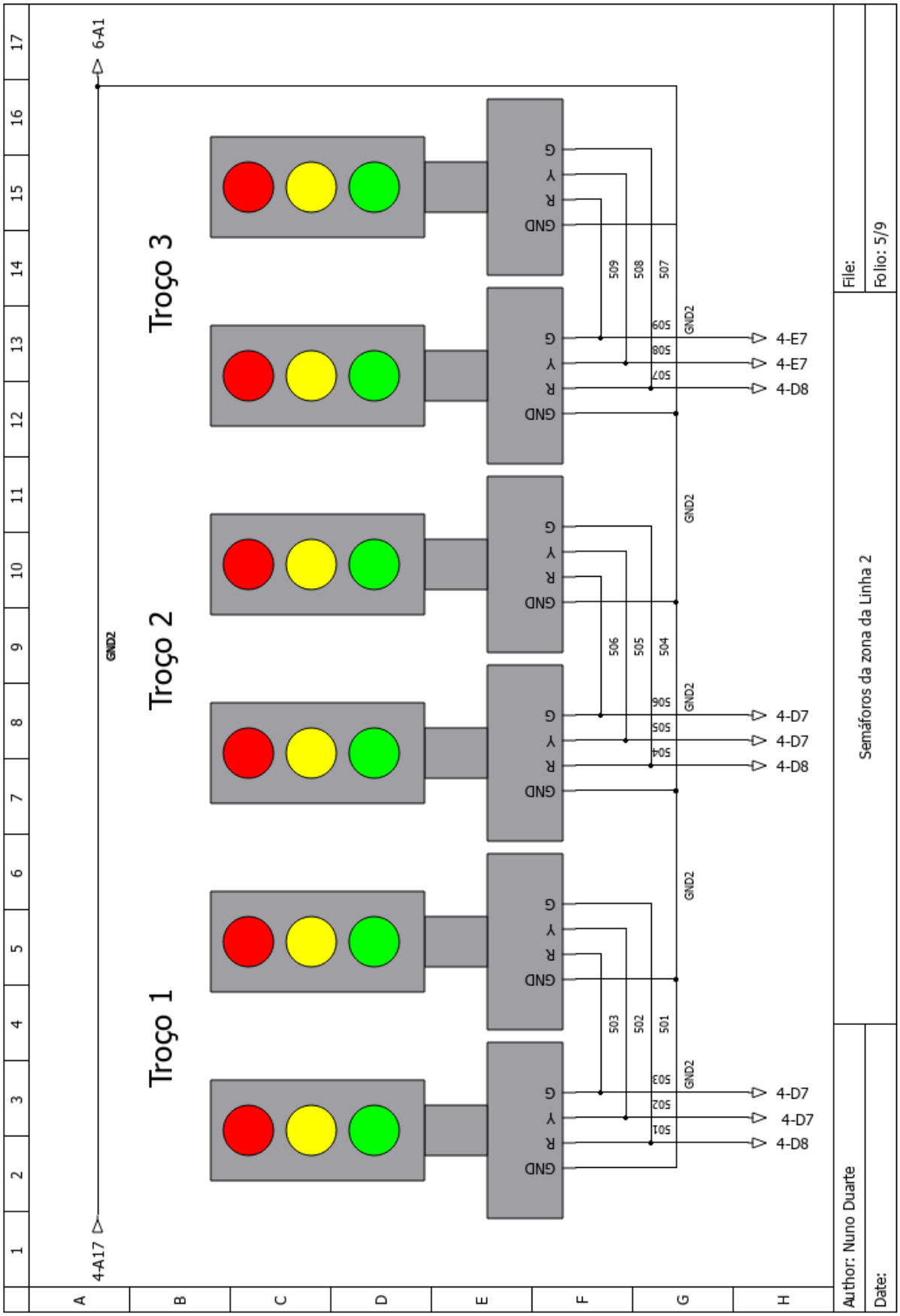
Esquema Elétrico do Protótipo da Fábrica

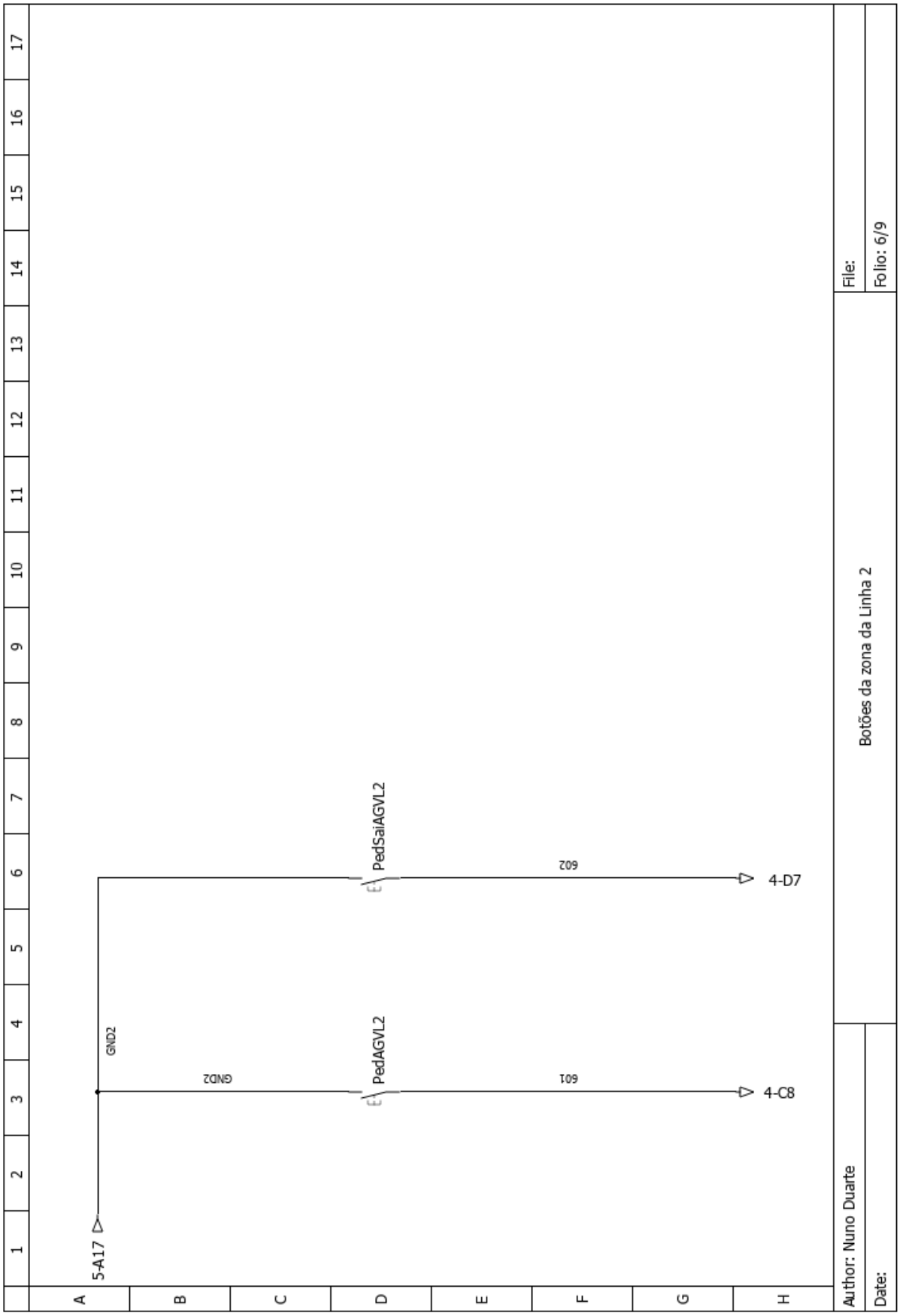


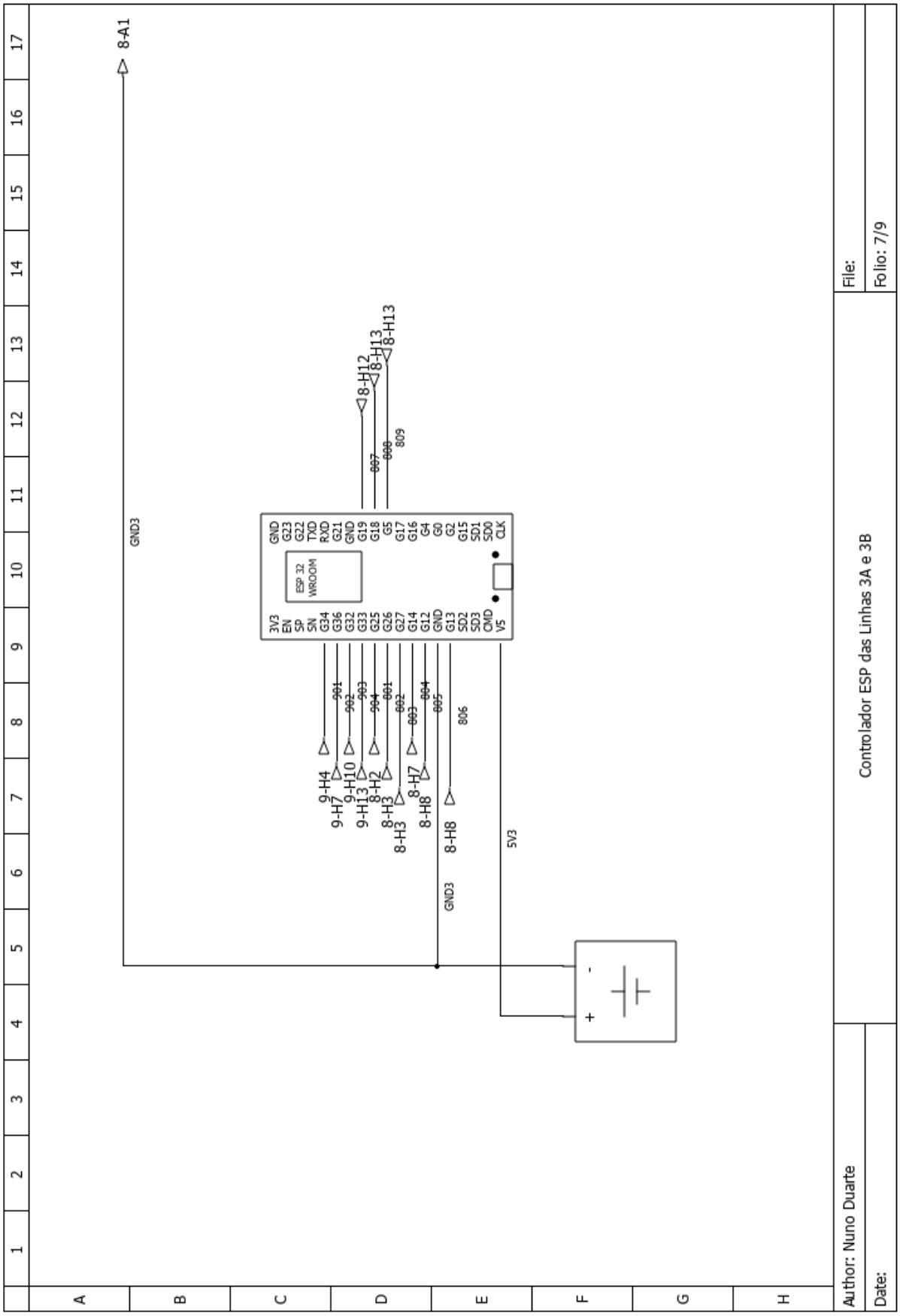

Troço 3
Troço 2
Troço 1
3-A1
1-A17
GND1
G
Y
R
GND
209
208
207
206
205
204
203
202
201
1-E7
1-E7
1-D8
1-D7
1-D7
1-D8
1-D7
1-D7
1-D8
Semáforos da zona da Linha 1
Author: Nuno Duarte
Date:
File:
Folio: 2/9

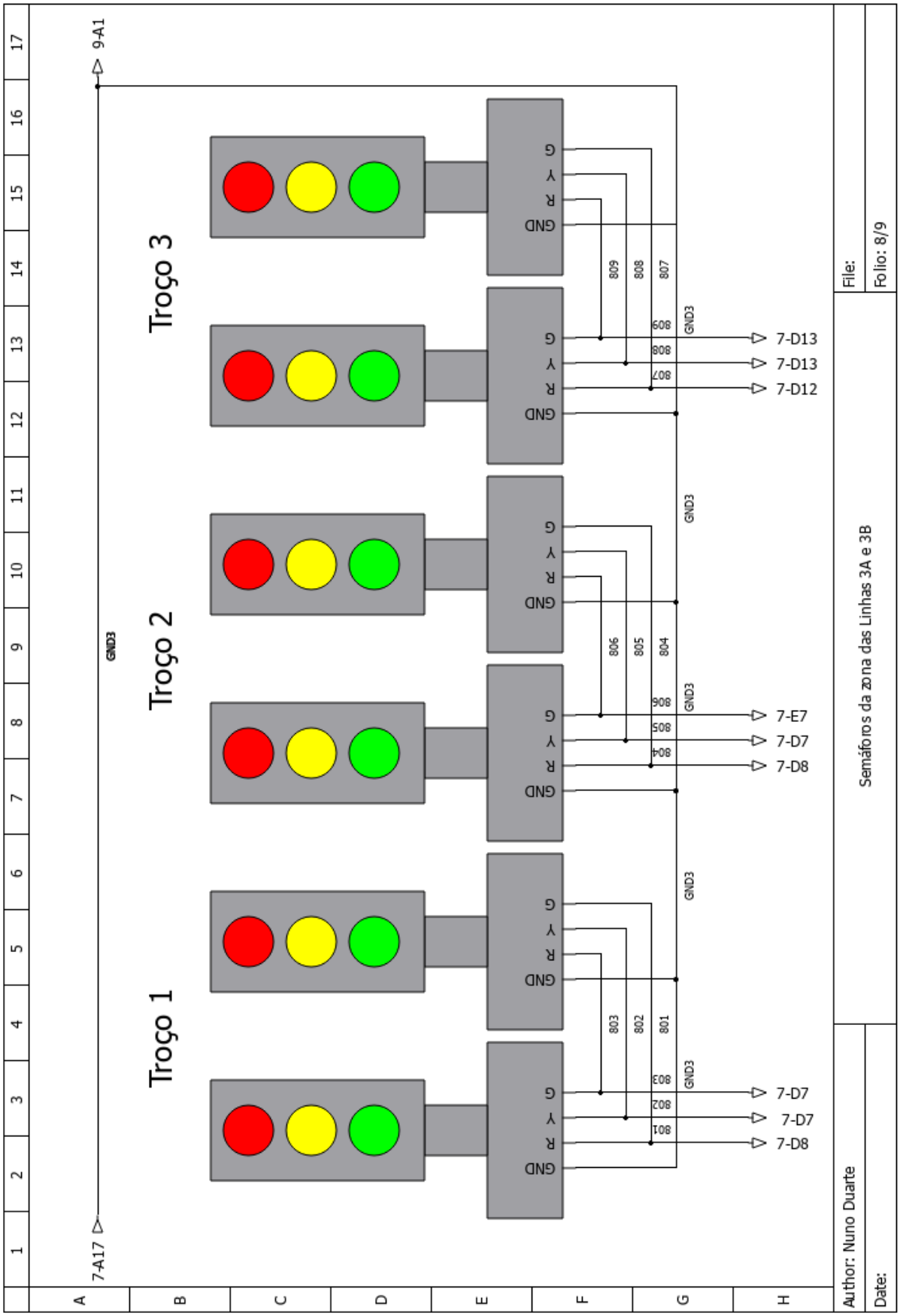

Troço 4
PedSaiAGVL1
PedAGVL1
305
304
1-D7
1-C8
GND1
303
302
301
1-D13
1-D13
1-D12
2-A17
G
Y
R
GND
Semáforos e botões da zona da Linha 1
Author: Nuno Duarte
Date:
File:
Folio: 3/9

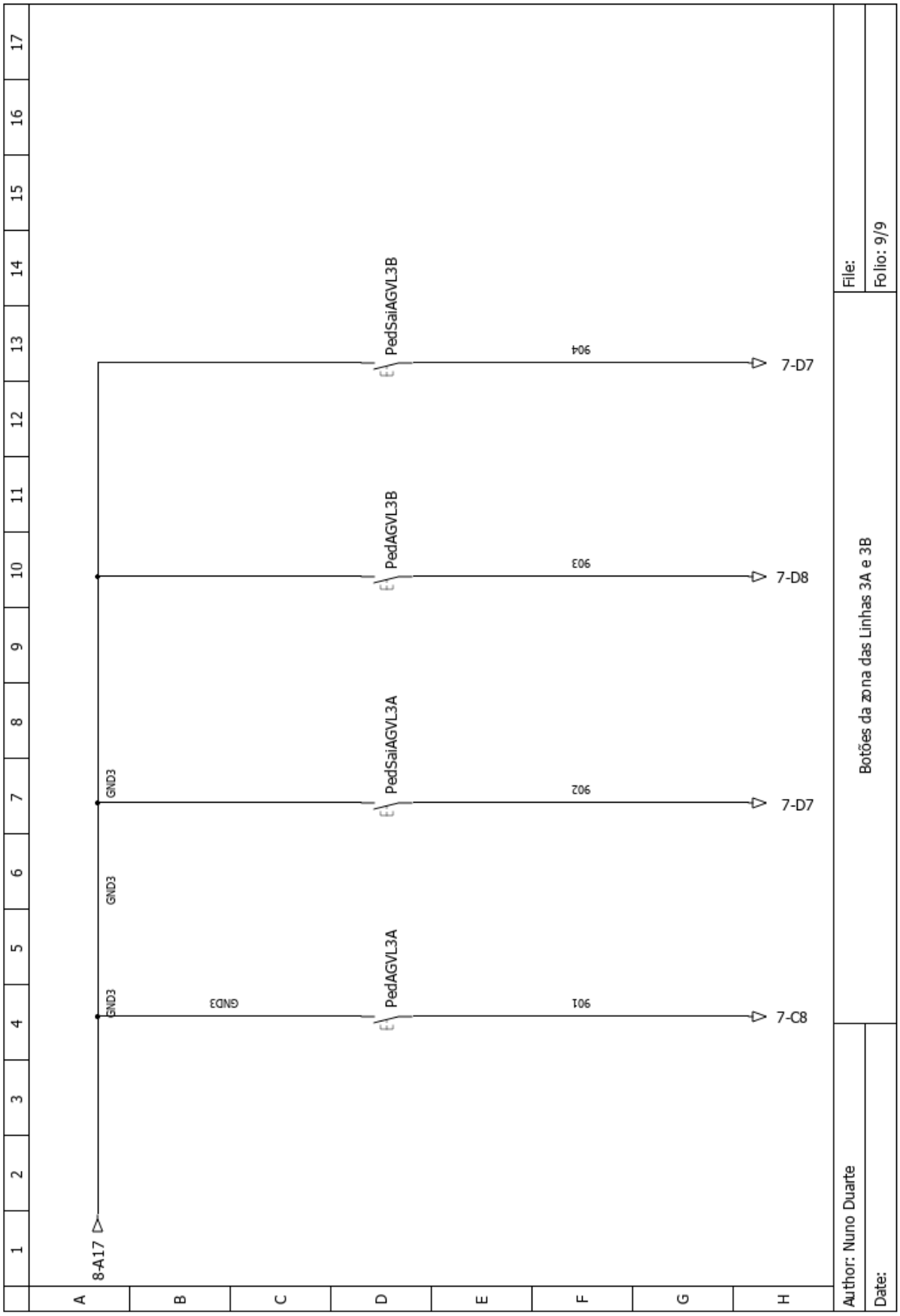

5-A1
GND2
ESP 32 WROOM
3V3
EN
SP
SN
G34
G36
G32
G33
G25
G26
G27
G14
G12
GND
G13
SD2
SD3
CMD
V5
GND
G23
G22
TXD
RXD
G21
GND
G19
G18
G5
G17
G16
G4
G0
G2
G15
SD1
SD0
CLK
601
602
501
502
503
504
505
506
507
508
509
6-H3
6-H6
5-H2
5-H3
5-H3
5-H7
5-H8
5-H8
5-H12
5-H13
5-H13
GND2
5V2
-
+
Controlador ESP da Linha 2
File:
Folio: 4/9
Author: Nuno Duarte
Date:


Troço 3
Troço 2
Troço 1
G
Y
R
GND
GND2
509
508
507
506
505
504
503
502
501
4-E7
4-E7
4-D8
4-D7
4-D7
4-D8
4-D7
4-D7
4-D8
6-A1
4-A17
Semáforos da zona da Linha 2
Author: Nuno Duarte
Date:
File:
Folio: 5/9


5-A17
GND2
GND2
PedAGVL2
PedSaiAGVL2
601
602
4-C8
4-D7
Botões da zona da Linha 2
Author: Nuno Duarte
Date:
File:
Folio: 6/9


8-A1
GND3
8-H12
8-H13
8-H13
807
808
809
ESP 32 WROOM
3V3
EN
SP
SN
G34
G36
G32
G33
G25
G26
G27
G14
G12
GND
G13
SD2
SD3
CMD
V5
GND
G23
G22
TXD
RXD
G21
GND
G19
G18
G5
G17
G16
G4
G0
G2
G15
SD1
SD0
CLK
9-H4
9-H7
9-H10
9-H13
8-H2
8-H3
8-H3
8-H7
8-H8
8-H8
801
802
803
804
801
802
803
804
805
806
GND3
5V3
-
+
Author: Nuno Duarte
Date:
Controlador ESP das Linhas 3A e 3B
File:
Folio: 7/9


Troço 3
Troço 2
Troço 1
9-A1
7-A17
GND3
G
Y
R
GND
809
808
807
806
805
804
803
802
801
608
808
807
7-D13
7-D13
7-D12
806
805
804
7-E7
7-D7
7-D8
803
802
801
7-D7
7-D7
7-D8
Semáforos da zona das Linhas 3A e 3B
Author: Nuno Duarte
Date:
File:
Folio: 8/9


8-A17
GND3
GND3
GND3
GND3
PedAGVL3A
PedSaiAGVL3A
PedAGVL3B
PedSaiAGVL3B
901
902
903
904
7-C8
7-D7
7-D8
7-D7
Botões da zona das Linhas 3A e 3B
Author: Nuno Duarte
Date:
File:
Folio: 9/9

Exemplos de código de funções e tarefas do Arduino

Código de configuração de tarefas em tempo real

```
/*-----------------------------------------------------*/
/*---------------------- Tasks ---------------------*/
/*-----------------------------------------------------*/
void TasksCreation(){

  //          function       name                                          st
ack  parameter         priority   task                    //Comentário
  xTaskCreate(TaskSensores, (const portCHAR *)"Leitura
Obstaculos",         128,   NULL,             1,         NULL);
   //Tarefa de leitura dos sensores
  xTaskCreate(TaskSinal,    (const portCHAR *)"Controlo
Sinalização",       128,   NULL,             3,         &xSinaliza);
 //Tarefa de controlo da sinalização
  vTaskSuspend(xSinaliza);
  xTaskCreate(TaskDemo,     (const portCHAR *)"Controlo Modo Demonstração",
128,   NULL,             2,         &xDemo);                //Tarefa de
controlo do modo de demonstração
  vTaskSuspend(xDemo);
  xTaskCreate(TaskModo,     (const portCHAR *)"Modo de
funcionamento",      128,   NULL,             2,         NULL);
   //Tarefa de controlo de modo de funcionamento


  //Inicia o escalunador
  vTaskStartScheduler();
}
```

Função de definição de movimento

```
/* CONTROLO DE MOVIMENTOS */
void MoveControl (int Step){
  switch(Step){
     case 1:
      Frente();
      break;
     case 2:
      Tras();
      break;
     case 3:
      Esquerda();
      break;
     case 4:
      Direita();
      break;
```

```
case 5:
 DiagFrDir();
 break;
case 6:
 DiagTrDir();
 break;
case 7:
 DiagFrEsq();
 break;
case 8:
 DiagTrEsq();
 break;
case 9:
 RodaCentEsq();
 break;
case 10:
 RodaCentDir();
 break;
case 11:
 RodaMeioFrenteDir();
 break;
case 12:
 RodaMeioFrenteEsq();
 break;
case 13:
 RodaMeioTrasDir();
 break;
case 14:
 RodaMeioTrasEsq();
 break;
case 15:
 RodaRodaTrDir();
 break;
case 16:
 RodaRodaFrDir();
 break;
case 17:
 RodaRodaTrEsq();
 break;
case 18:
 RodaRodaFrEsq();
 break;
case 19:
 CurvaFrEsq();
 break;
case 20:
 CurvaTrEsq();
 break;
```

```
      case 21:
       CurvaFrDir();
       break;
      case 22:
       CurvaTrDir();
       break;
      default:
       StopMot();
      break;
  }
  ActualMove = Step;
}
```

Função de controlo dos motores

```
/* CONTROLO DA RODA FRONTAL ESQUERDA */
void RodaFrEsq (int sentido){
  switch (sentido) {
  case 1:
    digitalWrite(FrEsqCW, HIGH);
    digitalWrite(FrEsqCCW, LOW);
    break;
  case 2:
    digitalWrite(FrEsqCW, LOW);
    digitalWrite(FrEsqCCW, HIGH);
    break;
  default:
    digitalWrite(FrEsqCW, LOW);
    digitalWrite(FrEsqCCW, LOW);
    break;
  }
}
/* CONTROLO DA RODA FRONTAL DIREITA */
void RodaFrDir (int sentido){
  switch (sentido) {
  case 1:
    digitalWrite(FrDirCW, HIGH);
    digitalWrite(FrDirCCW, LOW);
    break;
  case 2:
    digitalWrite(FrDirCW, LOW);
    digitalWrite(FrDirCCW, HIGH);
    break;
  default:
    digitalWrite(FrDirCW, LOW);
    digitalWrite(FrDirCCW, LOW);
    break;
  }
}
```

```
/* CONTROLO DA RODA TRASEIRA ESQUERDA */
void RodaTrEsq (int sentido){
  switch (sentido) {
  case 1:
    digitalWrite(TrEsqCW, HIGH);
    digitalWrite(TrEsqCCW, LOW);
    break;
  case 2:
    digitalWrite(TrEsqCW, LOW);
    digitalWrite(TrEsqCCW, HIGH);
    break;
  default:
    digitalWrite(TrEsqCW, LOW);
    digitalWrite(TrEsqCCW, LOW);
    break;
  }
}
/* CONTROLO DA RODA TRASEIRA DIREITA */
void RodaTrDir (int sentido){
  switch (sentido) {
  case 1:
    digitalWrite(TrDirCW, HIGH);
    digitalWrite(TrDirCCW, LOW);
    break;
  case 2:
    digitalWrite(TrDirCW, LOW);
    digitalWrite(TrDirCCW, HIGH);
    break;
  default:
    digitalWrite(TrDirCW, LOW);
    digitalWrite(TrDirCCW, LOW);
    break;
  }
}
```

Função de medição dos sensores

```
//Programa de leitura dos 4 sensores
void MedDists(){
  //Leitura das quatro distâncias
  DistFr = LeituraDist(EchoPinFr,TriggerFr);
  //InitSensDist();
  DistTr = LeituraDist(EchoPinTr,TriggerTr);
  //InitSensDist();
  DistEsq = LeituraDist(EchoPinEsq,TriggerEsq);
  //InitSensDist();
  DistDir = LeituraDist(EchoPinDir,TriggerDir);
  //InitSensDist();
  //Construção da string a enviar para o Raspberry PI
```

```
}
//Função de leitura de distância
int LeituraDist(uint8_t PinEcho, uint8_t PinTrigger){
  int distance,duration;
  //Pedido de activação do trigger durante 10 microsegundos
  digitalWrite(PinTrigger, LOW);
  delayMicroseconds(2);
  digitalWrite(PinTrigger, HIGH);
  delayMicroseconds(10);
  digitalWrite(PinTrigger, LOW);
  //Leitura do tempo de feedback do som em microsegundos
  duration = pulseIn(PinEcho, HIGH);
  //Calculo da distância recorrendo à velocidade do som
  distance = duration * 0.034 / 2; // Velocidade do som é dividida por 2
(devido ao efeito de ida e volta)
  //Mostra a medição na consola.
  //Serial.print("Distance: ");
  //Serial.print(distance);
  //Serial.println(" cm");
  return distance;
  delay(100);
}
```

Exemplos de código de funções e tarefas do Raspberry PI da Central

Biblioteca de conversão e envio de ficheiros

```
import asyncio
from time import sleep
import websockets
import pandas as pd
import json

ServerIP = '192.168.3.10'
Port = 8000
SrvAdr = 'ws://'+ServerIP+':'+str(Port)
#Diretoria dos ficheiros
facfile = 'Data/Factory.xlsx'
wpfile = 'Data/Workpoints.xlsx'
Nodefile = 'Data/NodeInfo.xlsx'
NodeIDfile = 'Data/Nodeid.xlsx'
#imprime os dados do servidor
print("Dados Servidor: "+SrvAdr)

async def SendDataToServer(Msg):
    if len(Msg)>0:
        print("Send: "+  Msg)
        MyMsg="|Mens|: "+Msg
        async with websockets.connect(SrvAdr) as ws:
            await ws.send(str(MyMsg))
            print("Espera resposta!!!")
            reply= await ws.recv()
            print("Received: "+str(reply))
            await ws.close()
    if reply == 'OK':
        print("Mensagem enviada com sucesso para o servidor")


async def SendFileToServer(Data,filename):
    if len(Data)>0:
        myData=filename+"|e|"+str(Data)
        print("Send: "+  str(myData))
        async with websockets.connect(SrvAdr) as ws:
            await ws.send(str(myData))
            print("Espera resposta!!!")
            reply= await ws.recv()
            print("Received: "+str(reply))
            await ws.close()
    if reply == 'OK':
        print("Ficheiro enviado com sucesso para o servidor")
```

```
def xlstojson(File):
    excel_data_df = pd.read_excel(File, sheet_name='Folha1')
    thisisjson = excel_data_df.to_json(orient='records')
    return json.loads(thisisjson)

def Main():
    print("Iniciou o programa")
    #conversão do ficheiro excel para json
    Data = xlstojson(facfile)
    asyncio.run(SendFileToServer(Data,facfile))
    Data = xlstojson(wpfile)
    asyncio.run(SendFileToServer(Data,wpfile))
    Data = xlstojson(Nodefile)
    asyncio.run(SendFileToServer(Data,Nodefile))
    Data = xlstojson(NodeIDfile)
    asyncio.run(SendFileToServer(Data,NodeIDfile))
    while True:
        Msg = input('Enter a message: ')
        asyncio.run(SendDataToServer(Msg))

if __name__=="__main__":
    Main()
```

Exemplos de código de funções e tarefas do Raspberry PI dos protótipos de robô

Biblioteca de descodificação dos ficheiros recebidos

```
import asyncio
from time import sleep
import websockets
import pandas as pd
import json

ServerIP = '192.168.3.10'
Port = 8000
SrvAdr = 'ws://'+ServerIP+':'+str(Port)
#Diretoria dos ficheiros
facfile = 'Data/Factory.xlsx'
wpfile = 'Data/Workpoints.xlsx'
Nodefile = 'Data/NodeInfo.xlsx'
NodeIDfile = 'Data/Nodeid.xlsx'
#imprime os dados do servidor
print("Dados Servidor: "+SrvAdr)

async def SendDataToServer(Msg):
    if len(Msg)>0:
        print("Send: "+  Msg)
        MyMsg="|Mens|: "+Msg
        async with websockets.connect(SrvAdr) as ws:
            await ws.send(str(MyMsg))
            print("Espera resposta!!!")
            reply= await ws.recv()
            print("Received: "+str(reply))
            await ws.close()
    if reply == 'OK':
        print("Mensagem enviada com sucesso para o servidor")


async def SendFileToServer(Data,filename):
    if len(Data)>0:
        myData=filename+"|e|"+str(Data)
        print("Send: "+  str(myData))
        async with websockets.connect(SrvAdr) as ws:
            await ws.send(str(myData))
            print("Espera resposta!!!")
            reply= await ws.recv()
            print("Received: "+str(reply))
            await ws.close()
    if reply == 'OK':
        print("Ficheiro enviado com sucesso para o servidor")
```

```
def xlstojson(File):
    excel_data_df = pd.read_excel(File, sheet_name='Folha1')
    thisisjson = excel_data_df.to_json(orient='records')
    return json.loads(thisisjson)

def Main():
    print("Iniciou o programa")
    #conversão do ficheiro excel para json
    Data = xlstojson(facfile)
    asyncio.run(SendFileToServer(Data,facfile))
    Data = xlstojson(wpfile)
    asyncio.run(SendFileToServer(Data,wpfile))
    Data = xlstojson(Nodefile)
    asyncio.run(SendFileToServer(Data,Nodefile))
    Data = xlstojson(NodeIDfile)
    asyncio.run(SendFileToServer(Data,NodeIDfile))
    while True:
        Msg = input('Enter a message: ')
        asyncio.run(SendDataToServer(Msg))

if __name__=="__main__":
    Main()
```

Biblioteca para encontrar o melhor trajeto

```
import factMan,rwxlsfiles

files = rwxlsfiles.xlsfiles
factory = factMan.factoryMan

class bestwaySupport():
    #verifica se o ponto já está no trajecto
    def checkway(actualway, chkval):
        try:
            if actualway.index(chkval)>0:
                return False
            else:
                return True
        except:
            return True

    #Função para encontrar o caminho com menor parametro
    def getbestweightway(ways):
        #Variaveis auxiliares
        params=[]
        #leitura do ficheiro da fábrica
        fabrica = files.readfile(files.facfiledirectory)
```

```
        #Calculo dos pesos de cada caminho
        for way in ways:
            #tamanho do equipamento
            tam=len(way)-2
            par = 0
            idx=0
            #calculo de cum caminho
            while (idx-1 < tam):
                for line in fabrica:
                    if line[1] == int(way[idx]) and line[2] == int(way[idx+1])
and line[4] == 1:
                        par = par + line[3]
                    if line[1] == way[idx+1] and line[2] == way[idx] and
line[4] == 2:
                        par = par + line[3]
                idx=idx+1
            #Save weight
            params.append(par)
        #procura o maior valor e o seu index
        minweight= min(params)
        best=ways[params.index(minweight)]
        print(best)
        return best

    #função para acrescentar a decisão de lado a virar no caso de T's e L's
(pode ser ponderado fazer o mesmo no caso dos pontos em "L")
    def insertTurns(Orig,Dest,way):
        #variaveis
        primPoint = False
        newWay=[]
        tam=len(way)
        pnt=0
        #Encontra os pontos anterior e seguinte ao caminho
        befPrim = factory.getSrcStrPoint(Orig)
        aftLast = factory.getSrcLastPoint(Dest)

        #print(befPrim)
        #leitura de ficheiros
        nos = files.readfile(files.nodesfiledirectory)
        lista = list(way)
        for line in nos:
            if line[0]==way[0]:
                if line[1]=='T' or line[1]=='+':
                    lista.insert(0,befPrim)
                    primPoint = True
                    pnt = 1
                break
        lista.append(aftLast)
```

```
        idx=0
        for point in lista:
            if primPoint:
                #print("Primeiro ponto foi inserido só para análise do T")
                primPoint = False
            else:
                if idx<tam:
                    for line in nos:
                        if line[0]==point:
                            if line[1]=='T':
                                newWay.append(point)
                                newWay.append(factory.getTDirection(lista,poin
t,pnt))
                            elif line[1]=='+':
                                newWay.append(point)
                                newWay.append(factory.getPlusDirection(lista,p
oint,pnt))
                            else:
                                newWay.append(point)
                    idx=idx+1
                    pnt=pnt+1
                else:
                    break
        return newWay


class bestway(bestwaySupport):

    #função para encontrar os caminhos possiveis
    def getways(Orig,Dest):
        #Variaveis internas
        newval=0
        tempactual=[]
        actual=[]
        temp=[]
        ways=[]
        #Encontra o primeiro ponto
        StrPoint = factory.getStrPoint(Orig)
        #Encontra o ultimo ponto
        LastPoint = factory.getLastPoint(Dest)
        #leitura de ficheiro da fábrica
        fabrica = files.readfile(files.facfiledirectory)
        #definição do primeiro ponto de origem
        MyOrig=StrPoint
        actual.append(MyOrig)
        #Ciclo de procura de caminhos válidos
        while len(actual)>0 or len(temp)>0 or len(ways)<=0:
            way=0
```

```
            for line in enumerate(fabrica):
                newval=0
                #encontrou nó inicial como origem
                if line[1][1] == MyOrig and line[1][4]==1:
                    newval=line[1][2]
                    if way==0:
                        #encontrou a primeira opção
                        if bestway.checkway(actual, newval):
                            actual.append(newval)
                            way=newval
                    else:
                        tempactual = []
                        #encontrou outra opção
                        #copia a trajectoria actual para uma temporária
                        lista = list(actual)
                        #com o tamanho do tupple irá ser retirada a ultima
posição do caminho atual para definir a outra possibilidade de rota
                        tamatual=len(tempactual)
                        del lista[tamatual-1]
                        tempactual = (lista)
                        #remove o valor do array
                        try:
                            if actual.index(newval)>0:
                                print("Está na
lista")
                        except:
                            #adiciona a nova opção
                            tempactual.append(newval)
                            temp.append(tempactual)

                #encontrou nó inicial como origem
                if line[1][2] == MyOrig and line[1][4]==2:
                    newval=line[1][1]
                    if way==0:
                        #encontrou a primeira opção
                        if bestway.checkway(actual, newval):
                            actual.append(newval)
                            way=newval
                    else:
                        tempactual = []
                        #encontrou outra opção
                        #copia a trajectoria actual para uma temporária
                        lista = list(actual)
                        #com o tamanho do tupple irá ser retirada a ultima
posição do caminho atual para definir a outra possibilidade de rota
                        tamatual=len(tempactual)
                        del lista[tamatual-1]
                        tempactual = (lista)
```

```
                        #remove o valor do array
                        try:
                            if actual.index(newval)>0:
                                print("Está na
lista")
                        except:
                            #adiciona a nova opção
                            tempactual.append(newval)
                            temp.append(tempactual)

            try:
                if actual.index(way)<(len(actual)-1):
                    if len(temp)>0:
                        actual=temp[0]
                        lista1 = list(temp)
                        del lista1[0]
                        temp=(lista1)
                        #vai buscar o ultimo valor e define que é o novo way
                        NovoTam=len(actual)
                        way=actual[NovoTam-1]
            except:
                actual=[]


            #verifica se o novo elemento já se encontra na ultima posição
            tamnovo=len(actual)
            #Ponto encontrado é o ponto final
            try:
                if way == LastPoint or actual[tamnovo-1]==LastPoint:
                    ways.append(actual)
                    if len(temp)>0:
                        actual=temp[0]
                        lista = list(temp)
                        del lista[0]
                        temp=(lista)
                        #vai buscar o ultimo valor e define que é o novo way
                        NovoTam=len(actual)
                        way=actual[NovoTam-1]
                    else:
                        actual=[]
            except:
                actual=[]

            MyOrig=way
        best=bestway.getbestweightway(ways)
        final=bestway.insertTurns(Orig,Dest,best)
        print(final)
```

Lista de mensagens criadas durante o projecto

Mensagens AGV para Central

Protótipo de robô arrancou e indica dados de comunicação

AGVxxxx|Hello|IP:xxx.xxx.xxx.xxx|Port:xxxx|

Estado e Localização atual

AGVxxxx|MyLoc|CurrentLocal:xxxxxxxxxxxxxxx|Work: xxxxx|Volt:xxxx|Dists:F-xxxxx/T-xxxxx/E-xxxxx/D-xxxxx|Sens:xxxxxxxx|Modo:xxxxxx|

Estado e indicação de troço bloqueado

AGVxxxx|WayBl|CurrentLocal:xxxxxxxxxxxxxxx|Work: xxxxx|Volt:xxxx|Dists:F-xxxxx/T-xxxxx/E-xxxxx/D-xxxxx|Sens:xxxxxxxx|Modo:xxxxxx|

Indicação da tag lida e o trabalho a ser executado

AGVxxxx|RFID |Read:xxxxxxxxxxxxxxxxxxxxxxxx|Work: xxxxx|

Indicação de que protótipo de robô chegou a linha para tarefa de carga/descarga

AGVxxxx|Charg|Line:xxxx

Indicação de que o protótipo de robô a sair de uma linha

AGVxxxx|LChrg|Line:xxxxx

Protótipo de robô em espera e estado do mesmo

AGVxxxx|Free |CurrentLocal:xxxxxxxxxxxxxxx|Work: xxxxx|Volt:xxxx|Dists:F-xxxxx/T-xxxxx/E-xxxxx/D-xxxxx|Sens:xxxxxxxx|Modo:xxxxxx|

Mensagens Central para AGV

Envio de ficheiros

Central|File |File Data|xxxxxxxxxxxxxxxxx

Indicação de comunicação OK

Central|Alive|

Ordem de paragem de todos os protótipos de robô

Central|Stop |

Pedido de trabalho a um protótipo de robô

Central|Job |Work: xxxxx|Start:xxxxxxxxxxxxxxx|End:xxxxxxxxxxxxxxx|

Pedido de estado de um protótipo de robô

Central|StAsk|

Mensagens Raspberry PI para Arduino

Pedido de Movimento

Raspber|Movem|Way:xx|Speed:xxx|

Comunicação Ativa

Raspber|Alive|

Mensagens Arduino para Raspberry PI

Indicação das distâncias medidas

Arduino|Statu|Dists:F-xxxxx/T-xxxxx/E-xxxxx/D-xxxx|

Estado dos sensores de obstaculos

Arduino|Statu|Sens:xxxxxxxx|

Tensão da bateria

Arduino|Statu|Volt:xxxxx|

Modo de funcionamento atual

Arduino|statu|Modo:xxxxx|

Estado do sensor de deteção de linhas

Arduino|Statu|Line:xxx,xxx,xxx,xxx,xxx|

Mensagens da Central para as Linhas

Estado do protótipo da fábrica:

Central|Statu|1 :xx|2 :xx|3 :xx|4 :xx|5 :xx|6 :xx|7 :xx|8 :xx|9 :xx|10 :xx|

Estado do AGV numa determinada linha:

Central|Statu|Line-xx|AGV: xxxxxxxxxxxxxxxxxxxxxxxx|

Estados possíveis: NoAGV, AGVxxxx-InLine, AGVxxxx-Leaving, AGVxxxx-OnWay

Protótipo de robô pronto para descarga numa determinada linha

Central|AGVt|Line-xx|AGVxxxx|Ready To Unload|

Protótipo de robô a sair de uma determinada linha

Central|AGVt|Line-xx|AGVxxxx|Leaving|

Protótipo de robô saiu de uma determinada linha

Central|AGVt|Line-xx|AGVxxxx|Out|

Mensagens das linhas para a Central

Linha arrancou e indica dados de comunicação

Line-xx|Hello|IP:xxx.xxx.xxx.xxx|Port:xxxx|

Linha pede que sejam lhe entregue peças

Line-xx|RequP|Request Parts

Linha informa que descarregou o protótipo de robô

Line-xx|UnldD|AGV Unloaded

Linha envia o seu estado

Line-xx|AskSt|Ask My Status

Instituto Superior de Engenharia
Politécnico de Coimbra