Desenvolvimento de um Veículo Autónomo e Controlável para o Centro de Ciência Viva de Bragança

Alfredo André Rodrigues Padrão

Dissertação apresentada à Escola Superior de Tecnologia e de Gestão de Bragança para obtenção do Grau de Mestre em Engenharia Industrial ramo de Engenharia Eletrotécnica.

Trabalho orientado por:

Prof. Dr. José Luís Sousa de Magalhães Lima

Bragança

2020

Desenvolvimento de um Veículo Autónomo e Controlável para o Centro de Ciência Viva de Bragança

Alfredo André Rodrigues Padrão

Dissertação apresentada à Escola Superior de Tecnologia e de Gestão de Bragança para obtenção do Grau de Mestre em Engenharia Industrial ramo de Engenharia Eletrotécnica.

Trabalho orientado por:

Prof. Dr. José Luís Sousa de Magalhães Lima

Bragança

2020

Dedicatória

Dedico este trabalho aos meus queridos pais Alfredo e Fernanda, incansáveis incentivadores da minha formação profissional, exemplos de luta e determinação, que nunca deixaram de me apoiar nos momentos de maior dificuldade, a eles lhes dedico.

Agradecimentos

Queria aproveitar esta oportunidade para agradecer a todos os que me acompanharam ao longo desta etapa e que contribuíram de diversas formas para que a realização deste trabalho fosse possível.

Um obrigado à Instituição (IPB) que me acolheu durante estes 4 anos, onde pude fazer amigos para toda a vida e encontrar docentes com vontade de ensinar e comprometidos com o sucesso dos seus alunos, em especial obrigado ao orientador desta dissertação, Professor Lima, que me deu todas as sugestões para a melhor elaboração possível desta dissertação, sempre pronto para ajudar e para me fazer evoluir neste contexto.

Agradecer também aos meus tios, Agostinho Padrão e Amélia Padrão e também ao meu primo, Tiago Padrão, por todo o apoio, ajuda e incentivo durante todo o desenvolvimento desta dissertação, nunca me deixando desistir e sempre com uma enorme predisposição em ajudar quando necessário.

E por fim, mas mais importante, à minha mãe, Fernanda Padrão e ao meu pai, Alfredo Padrão, que sempre foram o pilar de tudo na minha vida e a minha fonte de inspiração. Aqui fica um grande e humilde obrigado e eterna gratidão, por sempre terem acreditado nas minhas capacidades e nunca deixarem de me incentivar a ir mais além.

Resumo

Esta dissertação tem como base a eficiência de um sensor de linha num carro eletrónico. O foco incidiu sobre o desenvolvido de código a aplicar e circuitos elétricos necessários para o seu devido funcionamento e, paralelamente, foi usada uma placa Printed Circuit Board (PCB) com um circuito elétrico com um LCD para a contagem dos tempos e um botão para a seleção do modo de funcionamento.

Através da análise dos tempos obtidos conseguiremos compreender melhor qual a utilidade deste tipo de sensor, a sua implementação em contexto real e vantagens que ele iria trazer no mundo automóvel, percebendo a diferença de tempo em percorrer um circuito com um sensor de linha e, posteriormente, manualmente através de um Joystick.

Palavras-Chave: Sensor de linha, Alphabot, Joystick, Robótica.

Abstract

This dissertation is based on the efficiency of a line sensor in an electronic car. The focus is on the code applied and the electrical circuits necessary for its malfunction and, in parallel, a PCB board with an electrical circuit with LCD for counting the times and a button for selecting the operating mode.

Through the analysis of times, it is possible to better understand what is the use of this type of sensor, the implementation in a real context and the advantages that it could generate in the automotive world, realizing the difference in time in traversing a circuit with a sensor line and, later, manually with a joystick.

Keywords: Line Sensor, Alphabot, Joystick, Robotics.

Conteúdo

Lista de Tabelas

Lista de Figuras

Siglas

ARM Advanced RISC Machine. 12

CC Corrente Contínua. 12

CCVB Centro de Ciência Viva de Bragança. 5

DARPA Agência de Projetos de Pesquisa Avançada de Defesa. 23

EEPROM Electrically-Erasable Programmable Read-Only Memory. 32

EUA Estados Unidos da América. 10, 23

FNR Festival Nacional de Robótica. 14

GPS Global Positioning System. 24, 26, 28

HW Hardware. 24, 67

IA Inteligência Artificial. 10

IBM International Business Machines. 30

IMU Inertial Measurement Unit. 26–28

IPC Inter Process Communication. 28

LIDAR Light Imaging Detection And Ranging. 24, 26, 28

MIT Massachusetts Institute of Technology. 11

MU Movimento Uniforme. 4

MUV Movimento Uniformemente Variado. 4

PCB Printed Circuit Board. vii, viii

PWM Pulse Width Modulation. 56

RADAR Radio Detection And Ranging. 24, 26, 28

RAM Random Access Memory. 12, 29, 30

RGB Red Green Blue. 12

ROM Read Only Memory. 29, 30

RPi2 Raspberry PI 2. 12, 44

SPR Sociedade Portuguesa de Robótica. 14

STEAM Science Technology Engineering Arts and Mathematics. 3

SW Software. 11, 24, 28, 29, 34, 53, 67

Capítulo 1

Introdução

Há muito tempo que o homem vem idealizando reproduzir-se por meios mecânicos, criando máquinas capazes de executar e suprir com propriedade as tarefas humanas. Grandes pensadores, historicamente conhecidos, como Leonardo da Vinci [1] e Nikola Tesla [2] dedicavam grande parte do tempo em estudos direcionados à projeção da construção de mecanismos apropriados para copiar ou simular algumas características da capacidade humana, podendo dizer-se que estas invenções já tinham como base a ideia da Robótica.

No entanto, é no princípio do século XX, com a Revolução Industrial, que a Robótica aparece com maior força, devido à necessidade de intensificar a produção e melhorar a qualidade dos produtos, surgindo neste período os primeiros robôs com aplicações industriais, repetindo infinitamente e com precisão milimétrica, uma série de operações previamente programadas [3].

Definimos então a robótica como uma área de estudo multidisciplinar que se apoia nos conhecimentos de mecânica, eletrónica, física, matemática, biologia, informática e inteligência artificial, tendo por objetivo automatizar tarefas através de técnicas de programação e algoritmos orientados à construção de robôs. Na área educacional, a robótica visa levar o aluno a questionar, pensar e procurar soluções, saindo assim da teoria para a prática através de conhecimentos obtidos numa sala de aula, na vivência quotidiana, nos relacionamentos, nos conceitos e nos valores [4] .

Esta é considerada uma ferramenta educacional atrativa e estimulante, servindo de

apoio à aprendizagem, pela possibilidade de atuar com uma ampla gama de estratégias no ensino das diferentes componentes curriculares [5] [6] [7].

Por tudo isto, a robótica torna-se numa ciência interdisciplinar de grandes possibilidades na educação, pois a interdisciplinaridade é considerada a atitude positiva perante o conhecimento, o que implica uma mudança de comportamento diante da tomada de decisões, promovendo assim a cooperação, o trabalho, o diálogo entre pessoas, entre disciplinas e outras formas de conhecimento [8].

O termo autónomo, que aparece diretamente associado à robótica, refere-se à tentativa de dotar máquinas com uma capacidade inerente ao homem, capacitando-as na recolha e análise de dados sensoriais, tomando decisões e adotando comportamentos com base nessa mesma análise. No decurso das últimas três décadas foi percorrido um longo caminho e foram dados passos determinantes no sentido de se alcançarem sistemas completamente autónomos e inteligentes, sistemas semelhantes aos que são vulgarmente retratados, por exemplo, no cinema e na televisão [9].

É notório que, com os avanços neste campo e também devido às novas tecnologias, a relação com a sociedade tem mudado drasticamente. Hoje, o processo de comunicação é feito de forma quase que instantânea, e isso não está restrito apenas à comunicação, uma vez que a tecnologia está fortemente inserida no processo de trabalho, lazer, saúde, entre outras tantas áreas, tendo grandes centros de tecnologia como objetivo principal as novas tecnologias, criando assim robôs capazes de lidar com tomadas de decisão cada vez mais complexas [10].

Assumimos então que a sociedade, de forma geral, se rende às tecnologias e à modernidade. Porém, o processo de ensino ainda se mostra resistente quanto ao uso destes meios e, com isso, torna-se ineficiente e de certa forma fechado para estas inserções tecnológicas [11].

1.1 Contextualização

O desenvolvimento de máquinas inteligentes está cada vez mais presente no quotidiano, sendo esta uma tarefa especializada. Porém, existe nos dias de hoje uma robótica para crianças e adolescentes, sendo esta assim reconhecida como uma forma de estimular e desenvolver competências cognitivas, contribuindo para uma boa aprendizagem. É fundamental as crianças terem acesso a este tipo de informação, para terem cada vez mais a capacidade de adaptação a este novo mundo que é a automação e a robótica.

Uma das competências que ajuda a desenvolver é o raciocínio lógico, pois um robô é uma máquina desenvolvida e programada para responder a estímulos aos quais é submetida, tendo como finalidade cumprir uma ação de acordo com o comando que lhe for dado. É fundamental que a sua criação siga uma lógica, ajudando assim desta forma no desenvolvimento do raciocínio nos mais jovens.

Existe uma metodologia que engloba este tipo de aprendizagem , apelidada de Science Technology Engineering Arts and Mathematics (STEAM). Esta metodologia é baseada em projetos, que têm como objetivo, através de diversos conhecimentos, desenvolver valores juntamente com os conteúdos abordados, para formar alunos e preparar assim as pessoas para os desafios futuros [12].

O STEAM incentiva a formação dos mais jovens por meio de cinco etapas:

- **Investigar**

  O aluno é incentivado a se aprofundar num tema, procurando dados e informações sobre o assunto que está a ser abordado. Também é nesse momento que as conexões entre as áreas do saber são realizadas.

- **Descobrir**

  Desenvolvimento da técnica, na qual o aluno deve procurar por soluções. Neste processo, ele entende também o que não demonstra bons resultados no projeto que está a ser trabalhado.

- **Conectar**

  Todos os pontos até então trabalhados devem ser conectados de forma a criar um caminho para a solução do problema apresentado.

- **Criar**

  As soluções são aplicadas e as dúvidas são solucionadas de maneira estratégica, gerando conexões entre o problema e a resolução e todas as habilidades que estão a ser desenvolvidas precisam ser exploradas.

- **Refletir**

  Após apresentar todos os pontos trabalhados e as soluções conquistadas, devem ser feitas trocas de feedbacks e de orientações para lapidar o processo criativo.

Uma grande vantagem deste método é que a integração das disciplinas através de projetos, fazendo com que as crianças se tornem muito mais proativas, estando elas assim prontas para a resolução de problemas, aptas a trabalhar em equipa através da colaboração e ainda saberão melhor como utilizar o tempo que têm [13].

Com o apoio da tecnologia e por meio da experimentação, as crianças podem ainda estimular a criatividade e outras características que são muito importantes no século XXI, como capacidade de lidar com o diferente, adaptabilidade, comunicação e habilidades sociais.

Com a implementação deste sistema e tendo com base esta metodologia, o objetivo principal será ensinar de uma forma lúdica algumas leis da física de modo a tornar a aprendizagem mais fácil e o conhecimento mais consolidado, assim como maximizar a compreensão do contexto, através do Movimento Uniforme (MU) e do Movimento Uniformemente Variado (MUV).

Deste modo, será possível que qualquer pessoa, nomeadamente os mais jovens, ao utilizar este equipamento, perceber quais as diferenças entre carro em modo autónomo, utilizando o sensor de linha ou ser controlado manualmente, tirando assim ilações sobre as vantagens e desvantagens de cada um destes mecanismos.

Este projeto tem tudo para cativar as gerações mais jovens para a robótica, para o raciocínio lógico e para a curiosidade em perceber e saber mais sobre este tipo de implementações e projetos. Este propósito vem de encontro a tudo o que engloba o plano de ação do Centro de Ciência Viva de Bragança (CCVB), que pretende estabelecer bases para a criação de um exemplo participativo que promove a utilização de recursos científicos, tecnológicos e pedagógicos, através da organização anual de um conjunto diversificado de eventos e atividades científicas tais como:

- Cafés de Ciência;
- Oficinas Científicas;
- Workshops;
- Dias temáticos e abertos;
- Formação/Orientação de estagiários.

Deste modo, a promoção do conhecimento científico, a divulgação e a disseminação da ciência e da tecnologia acabam por ser os pontos fundamentais de atuação do CCVB [14].

1.2 Objetivos

Este projeto tem como finalidade desenvolver um robô seguidor de linha autónomo, que execute o trajeto do robô sobre uma linha preta, em pistas diferentes, reduzindo erros no percurso, através da construção de uma plataforma de hardware e de software livre (open source) e, também, a implementação de um comando (joystick) que permita a qualquer utilizador controlar o carro.

O objetivo principal de todo o sistema desenvolvido é comparar o tempo que o carro demorará a percorrer o percurso através do robô autónomo com aquele que é feito através do controlo humano. Deste modo será possível ter uma maior perceção de qual a utilidade

do sensor e das suas vantagens e desvantagens, comparativamente a um controlo humano do robô.

1.3 Estrutura da Dissertação

O restante desta dissertação está organizado em mais cinco capítulos, que são sucintamente definidos abaixo.

- No **Capítulo 2** é descrita a fundamentação teórica desta dissertação, sendo um capítulo inteiramente dedicado a descrever o estudo que foi feito relativamente à temática e aos conteúdos que existem atualmente e que podem ser associados.

- No capítulo seguinte, **Capítulo 3**, é feita uma descrição mais detalhada de todos os componentes e das tecnologias que são utilizadas. Para isso, são descritos cada um destes componentes, fazendo uma referência à sua especificação e imagens ilustrativas do componente ao qual nos estamos a referir. Para além disso, é feito o registo de todo o software e hardware desenvolvido, fazendo uma análise geral das especificidades do mesmo, uma especificação dos circuitos elétricos e também do desenvolvimento do código que foi utilizado.

- No **Capítulo 4** é feita uma interpretação dos resultados que foram obtidos, fazendo uma separação dos vários tipos de resultados pelo tipo de atividade que o carro desenvolveu (automática / controlo manual) e fazendo uma comparação entre ambos. É também feita uma análise final ao projeto em si, retirando todas as ilações e conclusões referentes ao mesmo.

- Por fim, no **Capítulo 6**, através dos resultados obtidos, é feita uma conclusão geral dos resultados, fazendo uma comparação com o contexto atual ao nível da robótica e analisando o que correu melhor e pior durante toda a elaboração da dissertação. Paralelamente, também é feita uma apreciação do trajeto que este projeto poderia seguir, de forma a ter uma maior utilidade no futuro.

Capítulo 2

Fundamentação Teórica

Os termos “robótica” e “robô” estão intimamente associados. Enquanto o primeiro está associado à área do conhecimento, o segundo diz respeito ao produto desta ciência. Robô vem do tcheco “robota” e significa “trabalho forçado”.

A Robótica é uma área de estudo multidisciplinar que se apoia nos conhecimentos de:

- Computação;
- Matemática;
- Física;
- Biologia;
- Engenharia Elétrica;
- Engenharia Mecânica.

O seu principal objetivo é “automatizar tarefas através de técnicas de programação e algoritmos orientados à construção de robôs"[15].

Nos dias atuais, a Robótica é uma área de crescente desenvolvimento, decorrendo dos inúmeros recursos que os sistemas de 13 microcomputadores oferecem. Devido a isto, grandes centros de tecnologia têm como objetivo criar novas tecnologias, fazendo robôs capazes de lidar com tomadas de decisão cada vez mais complexas [16] [17].

A robótica móvel lida com três níveis de autonomia, os robôs teleoperados, os semiautónomos e os robôs autónomos. Cada vez mais tem crescido o interesse por robôs móveis autónomos, os quais são capazes de operar no ambiente (interno ou externo), apreendendo e tomando decisões corretas de modo a que as suas tarefas sejam realizadas com êxito e sem a necessidade da intervenção humana, tanto em atividades de pesquisa como industriais [18] [19].

Nos últimos 5 anos, os robôs móveis autónomos são utilizados com sucesso em diversas aplicações externas (ambientes não estruturados), como por exemplo:

- Operações militares [20];
- Explorações espaciais [21];
- Minas [22];
- Portos Marítimos [23];
- Agricultura, entre outras aplicações [24].

Parte da robótica móvel especializa-se na navegação autónoma, que é uma tarefa fundamental em robôs móveis. O conceito de navegação autónoma é abordado como sendo a tarefa de conduzir um robô autonomamente de forma adequada, onde o robô é equipado com sensores para a leitura das propriedades do ambiente [25] [26].

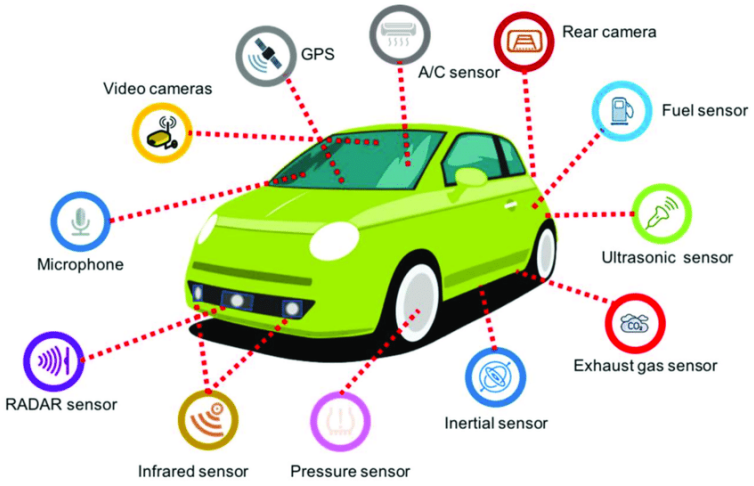


Figura 2.1: Sensores Automóveis

Este capítulo fornece uma visão geral dos principais conceitos que sustentam o trabalho de pesquisa, no capítulo 2.2, estando focado em rever o estado da arte atual de competições/projetos similares de mercado existentes, fornecendo uma análise de suas vantagens e desvantagens.

Para além disso, no capítulo 2.3, é feita uma retrospetiva a nível de microcontroladores e sensores, analisando detalhadamente cada um deles para, depois, ser possível uma melhor contextualização do sensor de linha.

2.1 Pesquisa Bibliográfica

Fazendo um estudo do mercado atual e do que ele nos oferece, é notório que existem diversos eventos associados a tudo o que é competição robótica e condução autónoma. Estes eventos são, hoje em dia, preponderantes para uma cativação da sociedade mais jovem para esta área, podendo eles, desde bastante novos, ter um contato direto com muitas das ferramentas que estão associadas á robótica.

2.1.1 Duckietown

A Fundação Duckietown é uma fundação sem fins lucrativos que desenvolve e promove o projeto Duckietown, estando registada no estado de Massachusetts, Estados Unidos da América (EUA). Tem como missão tornar o mundo mais animado com a diversão, a importância e os desafios da robótica e da Inteligência Artificial (IA), através de experiências de aprendizagens tangíveis, acessíveis e inclusivas, projetando plataformas e currículos de robótica disponíveis gratuitamente para todos os níveis de ensino e promover o seu uso no mundo. Com uma visão muito própria, na qual a IA e a robótica são as áreas mais interessantes de desenvolvimento, mostrando a tentativa da humanidade de criar seres artificiais que pensem e atuem como nós.

Figura 2.2: Logotipo Duckietown

Uma grande vantagem da plataforma Duckietown é derivada da complexa arquitetura

de Software (SW) fornecida, que contém componentes como calibração do sensor, configuração, perceção de baixo nível, reconhecimento de objeto, localização global, planificação de alto nível e coordenação descentralizada, tendo como objetivo fornecer uma completa arquitetura de "livro didático"que é comparável em complexidade com implementações em grande escala, sendo compreensível para iniciantes.

A Duckietown pretende que a sua utilização seja, principalmente, apoiar turmas de graduação ou pós-graduação em autonomia, sendo especialmente adequado para aprendizagem baseada em trabalho de equipa, onde grupos de alunos trabalham na melhoria de diferentes subsistemas que funcionam em conjunto. Era assim usada no Massachusetts Institute of Technology (MIT) na primavera de 2016, onde um grupo, com mais de 15 pós-doutorados e 5 professores, esteve envolvido no desenvolvimento inicial deste projeto [27].

Figura 2.3: Fundação Duckietown - 2016

Em alternativa, a Duckietown é adequada para aulas focadas num campo ou subsistema específico (por exemplo, visão computacional ou controlo não linear), tendo, neste caso, a vantagem da sua integração ser imediatamente observada.

Torna-se então numa plataforma de pesquisa, diminuindo suposições anteriores conhecidas sobre o meio ambiente (como dimensões conhecidas, cores, posicionamento de sinalização, topologias de rede permitidas, etc.). Os autores usam a Duckietown para

tópicos de pesquisa, como perceção restrita de recursos [28], co-design [29] e métodos formais para coordenação segura de veiculos [30].

O Duckiebot é uma plataforma que consiste, principalmente, num conjunto de componentes interligados. A computação é realizada num Raspberry PI 2 (RPi2), com 4 núcleos Advanced RISC Machine (ARM) de 900 MHz e 1 GB de Random Access Memory (RAM). A atuação é fornecida através de duas rodas controladas por motor Corrente Contínua (CC), incluídas com o quadro no kit do chassi.

Figura 2.4: Plataforma Duckiebot

O chassi é a parte mais dispensável do robô e pode ser substituído por qualquer configuração que use dois motores CC no acionamento diferencial de configuração. A placa do motor CC Adafruit foi projetada para se conectar diretamente ao RPi2. A deteção é fornecida exclusivamente por meio de uma câmara de alta definição, suportando o RPi2 uma câmara com design personalizado por meio de uma conexão paralela dedicada, estabelecendo comunicação com o Duckiebot através de WiFi.

Para grandes implementações, uma solução para a saturação da rede é um ponto de acesso integrado em cada Duckiebot. Esses hotspots móveis criam uma rede de 5 GHz, que são alimentados de forma independente, e que se conectam ao RPi2 por meio da Ethernet. Além disso, um joystick sem fios pode ser usado para tornar o controlo manual mais conveniente.

Os Duckiebots também podem ser equipados com Red Green Blue (RGB) LEDs para

permitir comportamentos globais que requerem comunicação entre veiculos.

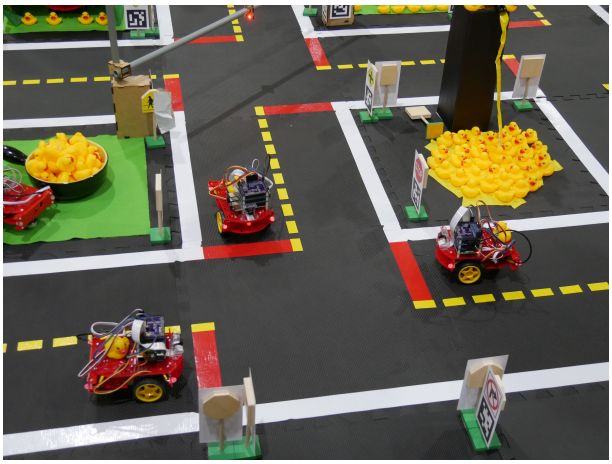


Figura 2.5: Duckiebots implementados no mesmo contexto

Em suma, o Duckiebot apresenta três configurações [31]:

- A configuração de autonomia mínima é suficiente para suportar todos os comportamentos de um único robô;
- A configuração estendida adiciona alguns recursos de "luxo"que são convenientes para o desenvolvimento, como um joystick e um ponto de acesso sem fio integrado;;
- A configuração da frota inclui díodos emissores de luz (LEDs) como meio de comunicação entre Duckiebot e permite os comportamentos de coordenação multi-robô.

Atualmente, 735 estudantes de 10 países usam Duckietown nas aulas da universidade preparando-se para carreiras em IA e robótica, estando a fundação neste momento a aperfeiçoar a experiência de alunos e instrutores [32].

2.1.2 Festival Nacional da Robótica

O Festival Nacional de Robótica (FNR) é atualmente o maior encontro científico nacional em competições de robótica, cuja a primeira edição foi realizada em 2001, na cidade de Guimarães, por iniciativa da Sociedade Portuguesa de Robótica (SPR).

A SPR foi criada por escritura pública, outorgada em 28 de Abril de 2006, como associação sem fins lucrativos, com o objetivo de promover e estimular o ensino, a investigação científica, o desenvolvimento tecnológico e as aplicações (indústria e serviços) na área da robótica, sendo este conseguido através de várias ações, nas quais se incluem, entre outras, o Festival Nacional de Robótica, publicações regulares, seminários e encontros [33].

Ao longo das suas várias edições, o FNR tem vindo a ser realizado por todo o País, proporcionando a divulgação da Robótica e áreas afins, de uma forma geograficamente equilibrada[34].

Este festival atua em 3 áreas principais:

- Competições robóticas (juniores e seniores);
- Encontro Científico;
- Demonstrações e atividades afin (ateliers, entre outros).

Reúne anualmente cerca de 500 participantes, incluindo alunos das escolas básicas, secundárias, instituições de ensino superior e investigadores nas áreas da robótica e automação, nacionais e internacionais, pretendendo atrair mais jovens para as áreas da ciência, tecnologia, engenharia e matemática, promovendo o espírito inovador e empreendedor das crianças e jovens através de métodos ativos de ensino e a aquisição de competências transversais.

(a) 2019 (b) 2020

Figura 2.6: Festival Nacional da Robótica

Os objetivos deste evento, de uma maneira generica, são:

- Dar um contributo positivo para o desenvolvimento da investigação em Robótica e Automação;
- Motivar os alunos das escolas básicas e secundárias para uma área tecnologicamente avançada e altamente multidisciplinar;
- Contribuir para a divulgação da Ciência e Tecnologia desenvolvida em Portugal;
- Mostrar ao público em geral como a robótica, a tecnologia e a ciência podem ser interessantes;
- Promover o encontro e as sinergias entre os grupos de trabalho nacionais na área da robótica, e também entre estes e grupos internacionais com trabalho na mesma área;
- Efetuar o apuramento das equipas Portuguesas para a competição mundial do RoboCup.

As competições inseridas no Festival Nacional de Robótica estão organizadas em dois escalões: Júnior e Sénior [35]:

- As competições Júnior integram as seguintes modalidades:
  - Busca e Salvamento Júnior;
  - Futebol Robótico Júnior;
  - Futebol Robótico Júnior;
- As competições Sénior integram as seguintes modalidades:
  - Futebol Robótico Médio (MSL - Middle Size League);
  - Simulação 2D;
  - Simulação 3D;
  - Liga de Plataforma Standard;
  - Robot@Factory;
  - Freebots;
  - Condução Autónoma;

Uma das competições de maior interesse é a de condução autónoma. Trata-se de uma competição técnica de média complexidade, na qual um robô móvel e autónomo deve executar um percurso ao longo de uma pista fechada, que apresenta semelhanças com a condução de um veiculo automóvel numa estrada convencional.

A pista utilizada tenta reproduzir, num determinado contexto, um cenário real, embora a competição decorra num ambiente estruturado. Esta pista, em formato de "8" ou similar, representa uma estrada de duas vias, uma passadeira com painéis semafóricos (um em cada sentido), um túnel, uma zona de obras, um obstáculo, sinais de trânsito e uma área de estacionamento com dois lugares, estando um ocupado [36].

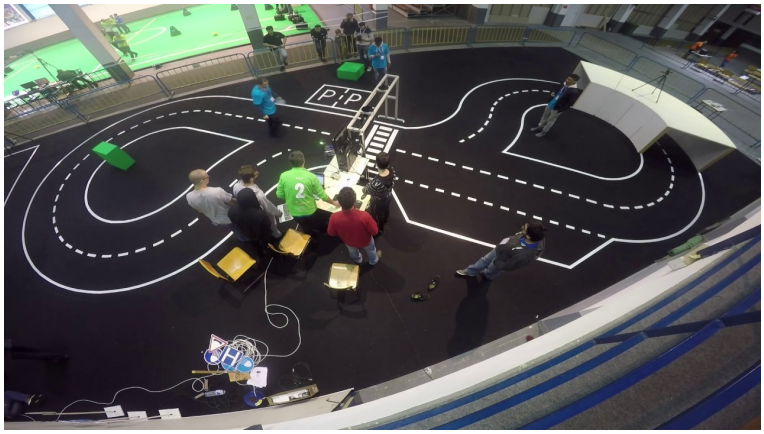


Figura 2.7: FNR - Desafio Condução Autónoma

Esta competição desenvolve-se em quatro fases, realizadas em três dias consecutivos, com diferentes desafios organizados em quatro categorias [36]:

- Desafios de condução, com quatro tipos de desafios;
- Desafios de estacionamento, com quatro tipos de desafios;
- Desafios de deteção de sinalização de tráfego vertical, com um único tipo de desafio.

No final, a equipa que tiver a melhor pontuação, considerando a soma dos resultados nos vários desafios, é a vencedora.

2.1.3 RoboParty

A RoboParty consiste num evento pedagógico que reúne equipas de 4 pessoas, durante 3 dias e duas noites, para ensinar a construir robôs móveis autónomos, de uma forma simples, divertida e com acompanhamento por pessoas qualificadas. Tem um objetivo pedagógico e não competitivo, visto que a ideia deste evento é ensinar a construir robôs e, no final, as equipas podem participar em desafios, sem competição. Com isto, para além da diversão, os participantes vão também aprender ciência e tecnologia, onde os orientadores (alunos e docentes universitários de Eletrónica Industrial), fazem um acompanhamento de perto.

Figura 2.8: Logotipo RoboParty

Inicialmente, é dada uma curta formação para:

- Aprender a dar os primeiros passos em Eletrónica;
- Programação de robôs;
- Construção mecânica.

Depois disto é entregue um KIT robótico desenvolvido pela *botnroll.com* e pela Universidade do Minho. Este KIT é chamado Bot'n Roll ONE A e é compatível com Arduino, tendo os participantes que montar a parte mecânica, a placa eletrónica (soldar os componentes) e programar o robô. O Kit é fácil de construir, é expansível, e pode ser usado em várias competições de robótica nacionais e internacionais [37].

Este robô tem também disponíveis as ligações existentes no Arduíno UNO, de forma a ser possível a ligação de módulos/shields ou de outros dispositivos sensores/atuadores, sendo dotado de uma ótima performance do hardware e software, já incluindo na sua versão base um display LCD 16x2 e vários botões de pressão para uma melhor interação com o utilizador.

Para além disto, também inclui [37]:

- Sistema de Infravermelhos para deteção de obstáculos, com os respetivos LEDs de estado;
- Dois LEDs de debug;

- Barramentos de comunicação I2C;
- Conectores para ligação direta de servos e seguidor de linha;
- Plataforma acrílica robusta com dois potentes motores DC.

Figura 2.9: Logotipo RoboParty

No final do evento, o robô pertence à equipa que o construiu. De seguida, é possível participar nos desafios oferecidos pela RoboParty, entre os quais se destacam [38]:

- **D**esafio de Obstáculos

  O robô realiza um percurso sem colidir com as paredes no menor tempo possível, sendo este o primeiro desafio para testar as capacidades de programação.

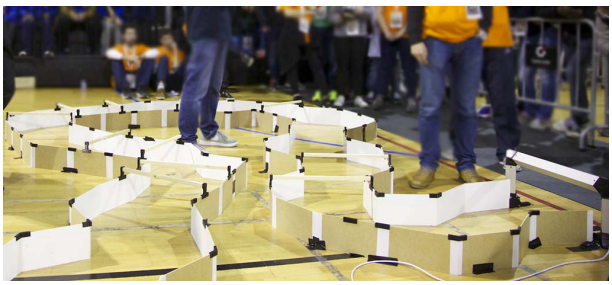


Figura 2.10: Desafio de Obstáculos - RoboParty

- **P**rova de Dança

  Desafio à criatividade e ao trabalho em equipa. As equipas de uma ilha têm que trabalhar em conjunto para enfeitar os seus robôs e criar uma coreografia robótica, de acordo com a temática que definirem. O júri decidirá de acordo com criatividade e a melhor prestação em palco.

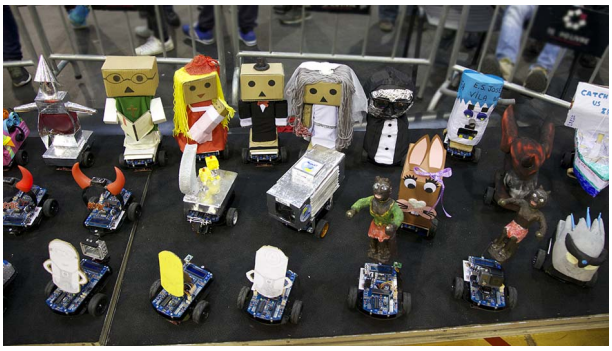


Figura 2.11: Prova de Dança - RoboParty

- **D**esafio Race of Champions

  O robô terá que seguir uma linha preta e completar uma volta ao circuito no menor tempo possível, com a aliciante de correr lado a lado com um adversário. Este desafio é uma prova aos recetores de infravermelhos dos sensores de obstáculos.

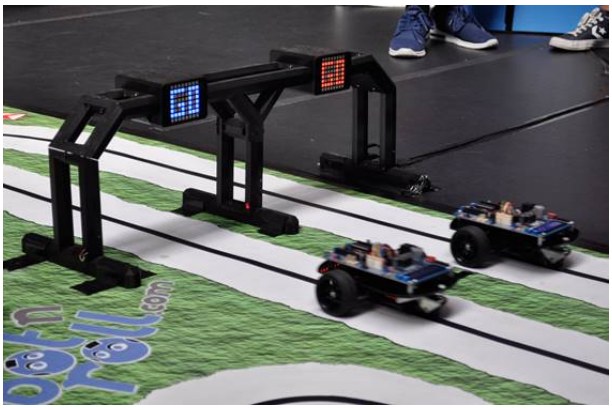


Figura 2.12: Desafio Race of Champions - RoboParty

2.1.4 Robotex

A Robotex é uma organização com sede na Estónia, cujo objetivo é organizar Robotex International: um festival anual de robótica que reúne milhares de engenheiros, executivos e estudantes, onde interagem com líderes da indústria de tecnologia, examinam novas start-ups e constroem robôs. Em 2020, realizaram-se pré-competições em 8 países diferentes: Finlândia, Grécia, China, Índia, Chipre, Armênia, Turquia e Coreia do Sul.

A Robotex International é considerada a maior competição de robótica do planeta, realizada pela primeira vez em 2001, no Taltech Assembly Hall, mudando-se em 2008 para as instalações do Taltech Sports Hall, onde foram garantidas melhores condições. Além de competições, este evento também oferece várias exposições, exibições e workshops para jovens, englobados na feira de tecnologia, na qual várias empresas e organizações, conhecidas em escala global, cooperam com este evento [39].

Figura 2.13: Robotex Internacional Logo

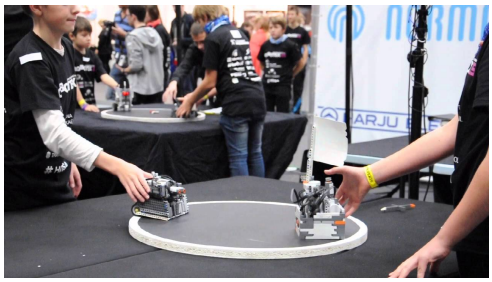
Desde 2010 que o evento se realiza durante vários dias, permitindo organizar competições com diferentes graus de dificuldade e reunir todos os entusiastas da robótica, ciência e engenharia, tendo o mesmo ganho em 2015 o título não oficial de maior competição de robótica da Europa.

Ao longo dos anos, foram realizadas competições em vários campos, tornando-se cada vez maiores com o aumento da adesão e o seu desenvolvimento. Estas competições envolviam desafios que, de acordo com nível de dificuldade designados pelos organizadores da competição, se dividiam em [40]:

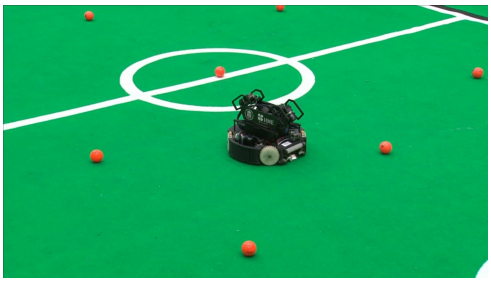
- seguimento de linha;
- passagem de labirinto;

- Voleibol;
- Basquetebol;
- futebol;
- Luta de sumo;
- Escalada à corda;
- Missão de resgate no ma.

De entre estas, o Sumo e o Futebol sempre foram das competições mais atrativas para os intervenientes, devido ao dinamismo, à programação e ao hardware que está envolvido em todos os robôs que participam nestas provas.



(a) Luta de Sumo



(b) Futebol

Figura 2.14: Competições - Robotex

Também as competições de seguimento de linha foram das que mais suscitaram interesse aos participantes, especialmente devido à capacidade autónoma de um robô realizar um percurso marcado por uma linha preta, no mais curto espaço de tempo, competindo dois robôs em pistas espelhadas e paralelas.

Os desafios existentes para este tipo de competição são:

- Line Following

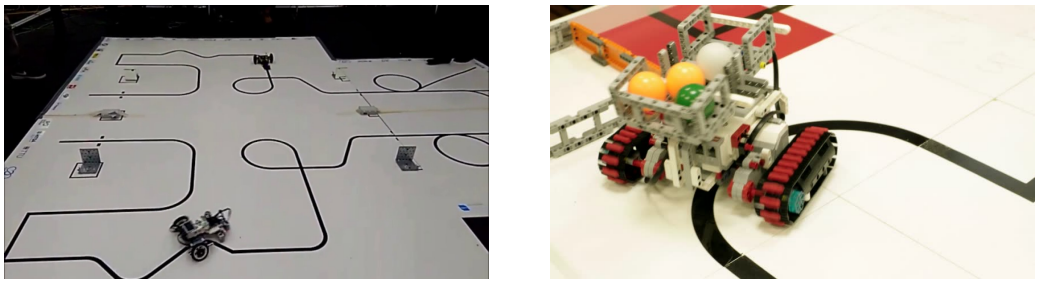
  Dois robôs, sem nenhuma especificação, devem percorrer o percurso.

- LEGO Line Following

  Os robôs, neste desafio, devem ser feitos de LEGOs.

- Line Following Enhanced

  Neste desafio a dificuldade é superior, devido à existência de obstáculos na pista.



(a) Line Following Enhanced (b) LEGO Line Following

Figura 2.15: Desafios Seguidores de Linha da Robotex

Esta organização tem como grande missão estimular o interesse das crianças e jovens pela engenharia e ciência. Para isso, vê como fundamental organizar competições de robótica de alta qualidade e eventos de tecnologia, como forma de estimular a cativação e o interesse por toda esta área.

2.2 Condução Autónoma

A ideia de automatizar veiculos é antiga. As pessoas tentam automatizar carros e aeronaves desde 1930, mas o hype para carros autónomos disparou entre 2004 e 2013. Para incentivar a tecnologia dos carros autónomos, o departamento de pesquisa do Departamento de Defesa dos EUA, a Agência de Projetos de Pesquisa Avançada de Defesa (DARPA), criou um desafio chamado "Grand DARPA Challenge"em 2004 com o objetivo de projetar um veiculo que pudesse conduzir autonomamente através do ambiente do mundo real. O vencedor do desafio foi o Team Taran Racing da Carnegie Mellon University.

Um carro capaz de conduzir autonomamente deve poder deslocar-se sem qualquer contribuição humana. Para conseguir isso, o carro autónomo precisa sentir o ambiente,

navegar e reagir sem interação humana. Uma ampla gama de sensores, como Light Imaging Detection And Ranging (LIDAR), Radio Detection And Ranging (RADAR), Global Positioning System (GPS), sensores e câmaras de odometria das rodas, são usados pelos carros autónomos para perceber o ambiente [41].

Para que um veiculo opere de forma autónoma, vários sistemas em tempo real devem trabalhar juntos. Esses sistemas, em tempo real, incluem mapeamento e entendimento do ambiente, localização, planeamento de rotas e controlo de movimento. Para que esses sistemas em tempo real tenham uma plataforma para trabalhar, o próprio carro autónomo precisa estar equipado com os sensores apropriados, Hardware (HW) computacional, rede e infraestrutura de SW [42].

A condução autónoma tem muitos benefícios para a humanidade, reduzirá o custo da viagem, permitirá que as crianças viajem sem adultos e aliviará as pessoas de conduzir, o que resultará numa experiência de viagem aprimorada. Os carros autónomos serão mais seguros, escolherão rotas mais ideais e aumentarão a produtividade da rodovia, podendo também estacionar, se necessário, longe de seus proprietários, o que reduziria os custos de estacionamento. O interesse em desenvolvedores autónomos de direção está a aumentar juntamente com o crescimento do setor da direção autónoma [43].

Para que uma máquina seja chamada de robô, ela deve estar de acordo em pelo menos três capacidades importantes: ser capaz de sentir, planear e agir. Para que um carro seja chamado de carro autónomo, ele deve atender aos mesmos requisitos, uma vez que são essencialmente carros robóticos que podem tomar decisões sobre como ir do ponto A ao ponto B, precisando o passageiro apenas de especificar o destino e, o carro autónomo deve ser capaz de o levar para lá com segurança, sendo necessária a adição de sensores e um computador de bordo para transformar um carro comum num carro autónomo.

2.2.1 Níveis de Autonomia de um Veiculo

É possível identificar seis níveis de autonomia do veiculo:

- **Autonomia Nível 0**

  Totalmente manuais, com condutor.

- **Autonomia Nível 1**

  Veiculos com condutor, mas também possuem um sistema de assistência que pode controlar automaticamente o motor ou os sistemas de direção usando informações ambientais.

- **Autonomia Nível 2**

  Automação parcial, o veiculo pode executar as funções de controlo do motor e de direção, enquanto todas as outras tarefas são controladas pelo condutor.

- **Autonomia Nível 3**

  Chamado de automação condicional, esperando-se que todas as tarefas sejam executadas pelo carro de forma autónoma, embora também seja esperado que um condutor que intervenha sempre que necessário.

- **Autonomia Nível 4**

  Não há necessidade de um condutor humano, pois tudo é tratado por um sistema automatizado. Apelidado de alta automação, funciona num tipo definido de área sob condições climáticas especificadas.

- **Autonomia Nível 5**

  Chamado de automação total, pois, neste nível, tudo é altamente automatizado e o carro pode funcionar autonomamente em qualquer estrada, em qualquer clima, não havendo necessidade de um condutor.

O nível mais alto de autonomia atualmente alcançado pelos carros modernos é o nível 3, o que significa que a tecnologia atual não está no nível de autonomia total nem no nível de alta autonomia, embora haja promessas de grandes empresas do setor automotivo que isso mudará no futuro próximo [44].

2.2.2 Sensores de Veiculos Autónomos

Os seguintes sensores devem estar presentes em todos os carros autónomos:

- **GPS**

  O sistema de posicionamento global é usado para determinar a posição de um carro autónomo, triangulando os sinais recebidos dos satélites GPS. É frequentemente usado em combinação com dados recolhidos de um codificador Inertial Measurement Unit (IMU) e de odometria de roda para posicionamento e estado do veiculo mais precisos, usando algoritmos de fusão de sensores.

- **LIDAR**

  Um sensor central de um carro autónomo, mede a distância de um objeto enviando um sinal de laser e recebendo seu reflexo. Pode fornecer dados 3D precisos do ambiente, calculados a partir de cada sinal de laser recebido, usando-os os veiculos autónomos o para mapear o ambiente, detetar e evitar obstáculos.

- **Câmara**

  A câmara de um carro autónomo é usada para detetar sinais de trânsito, semáforos, pedestres etc., usando algoritmos de processamento de imagem.

- **RADAR**

  O RADAR é usado para os mesmos fins que o LIDAR. Tem como vantagem ser mais leve e ter a capacidade de operar em diferentes condições. Embora tenha alcance maior, todas as categorias RADAR têm um campo de visão limitado [45].

- **Sensores de ultrassom**

  Os sensores de ultrassom desempenham um papel importante no estacionamento de veiculos autónomo e na prevenção e deteção de obstáculos em pontos cegos, pois seu alcance geralmente é de até 10 metros.

- **Codificador de Odometria das Rodas**

  Os codificadores de roda fornecem dados sobre a rotação das rodas do carro por segundo. A odometria utiliza esses dados, calcula a velocidade e estima a posição e a velocidade do carro com base neles. A odometria é frequentemente usada com os dados de outros sensores para determinar a posição de um carro com mais precisão [46].

- **IMU**

  Uma IMU consiste em giroscópios e acelerómetros, com um par orientado para cada um dos eixos. Esses sensores fornecem dados sobre o movimento rotacional e linear do carro, que é usado para calcular o movimento e a posição do veiculo, independentemente da velocidade ou de qualquer tipo de obstrução do sinal [47].

- **Computador de Bordo**

  Parte principal de qualquer carro autónomo. Como qualquer computador, ele pode ter uma potência variável, dependendo da quantidade de dados do sensor que ele precisa processar e da eficiência que precisa ter. Todos os sensores estão conectados a este computador, que deve fazer uso dos dados do sensor, entendendo-os, planeando a rota e controlando os atuadores do carro. O controlo é realizado enviando os comandos de controlo, como o ângulo de direção, acelerador e travagem para as rodas e motores.

2.2.3 Diagrama de Blocos SW de Veiculos Autónomos

A Figura ilustra o diagrama de blocos SW do carro autónomo padrão:

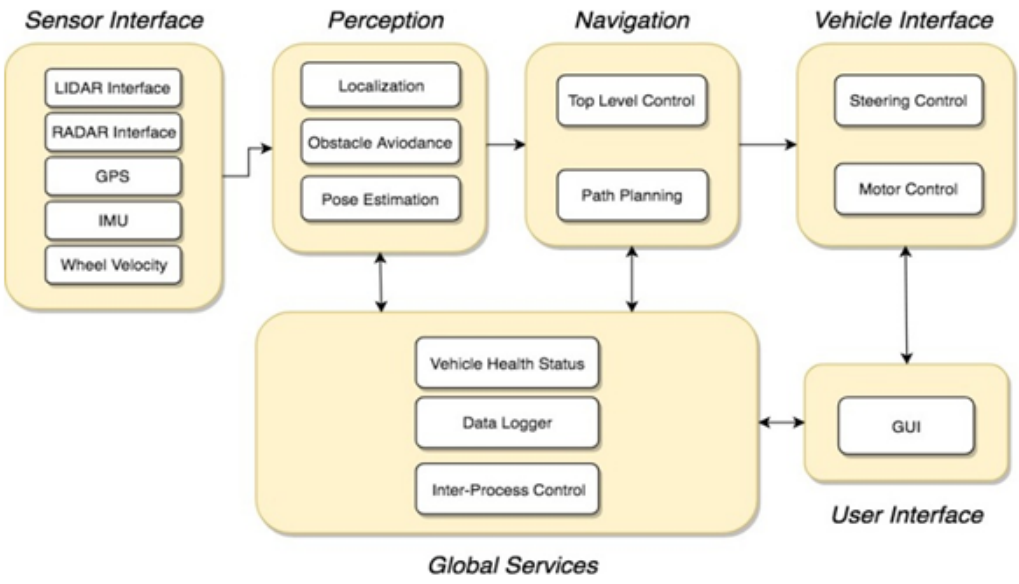


Figura 2.16: Diagrama de Blocos de Veiculos Autónomos

Cada bloco visto na Figura pode interagir com outros blocos usando Inter Process Communication (IPC) ou memória compartilhada, portanto, o sistema de mensagens ROS funciona muito bem aqui. É possível identificar os seguintes blocos para o diagrama de blocos SW de um carro autónomo típico [48]:

- **Módulos de Interface do Sensor**

  Toda a comunicação entre os sensores e o carro é realizada neste bloco, pois permite que os dados adquiridos dos sensores sejam compartilhados com outros blocos. Os sensores principais, dos quais os dados são conectados neste bloco, são: LIDAR, RADAR, GPS, IMU, câmaras e codificadores de roda.

- **Módulos de Percepção**

  Estes módulos processam dados de percepção de sensores como LIDAR, RADAR e câmaras e, em seguida, segmentam os dados processados para localizar diferentes

objetos que ficam imóveis ou em movimento. Esses módulos também ajudam na localização de carros autónomos, em relação ao mapa gerado do ambiente.

- **Módulos de Navegação**

  Determinam o comportamento do carro autónomo, pois possuem planeadores de rota e movimento, bem como uma máquina de estado do comportamento do carro. Para gerar a rota mais ideal para o carro ir do ponto A ao ponto B, os módulos de navegação comunicam com os módulos de perceção.

- **Interface do Veiculo**

  Envia comandos de controle, como direção, aceleração e travagem para o carro depois do caminho ser recebido no módulo de navegação.

- **Interface de Utilizador**

  Fornece controlos que podem estar relacionados à definição de um destino ou à visualização do mapa. Normalmente, um botão de paragem de emergência também deve estar disponível para o utilizador.

- **Serviços Globais**

  Controla a confiabilidade do SW do carro, permitindo o registo da data, hora e dos dados do sensor do carro.

2.3 Microcontroladores e Sensores

2.3.1 Microcontroladores

O microcontrolador foi inventado pela Texas Instruments no início da década de 70. Os primeiros microcontroladores eram basicamente microprocessadores com memória incorporada, tal como RAM e Read Only Memory (ROM). Posteriormente, evoluíram para

uma vasta variedade de dispositivos adaptados para aplicações específicas de sistemas embebidos, tais como carros, telefones sem fio e eletrodomésticos.

- **O Primeiro Microcontrolador**

  Inventado por 2 engenheiros na Texas Instruments, de acordo com o Instituto Smithsoniano. Gary Boone e Michael Cochram criaram o TMS 1000, que era um microcontrolador de 4 bits com ROM e RAM incorporados, sendo utilizado internamente pela empresa nas suas calculadoras (de 1972 a 1974) e melhorado ao longo dos anos.

Figura 2.17: Microcontrolador TMS 1000

- **Microcontroladores Intel**

  Além de produzir o primeiro microprocessador, a Intel também desenvolveu muitos microcontroladores importantes, tais como 8048 e o 8051:

  – – Introduzido em 1976, o microcontrolador 8048 foi um dos primeiros microcontroladores da Intel e foi utilizado como o processador no teclado do computador da International Business Machines (IBM), sendo estimado que mais de um bilião tenham sido vendidos.

  – O microcontrolador 8051 apareceu em 1980, tornando-se uma das famílias mais populares de microcontroladores. Variantes da sua arquitetura ainda são produzidas atualmente, tornando o 8051 um dos projetos eletrónicos mais duradouros da história.

Figura 2.18: Microcontrolador 8248

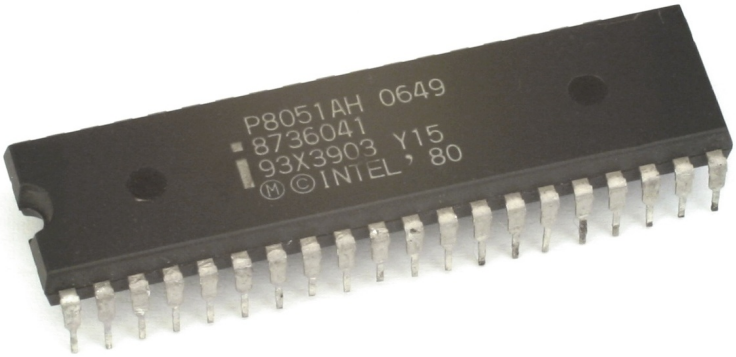


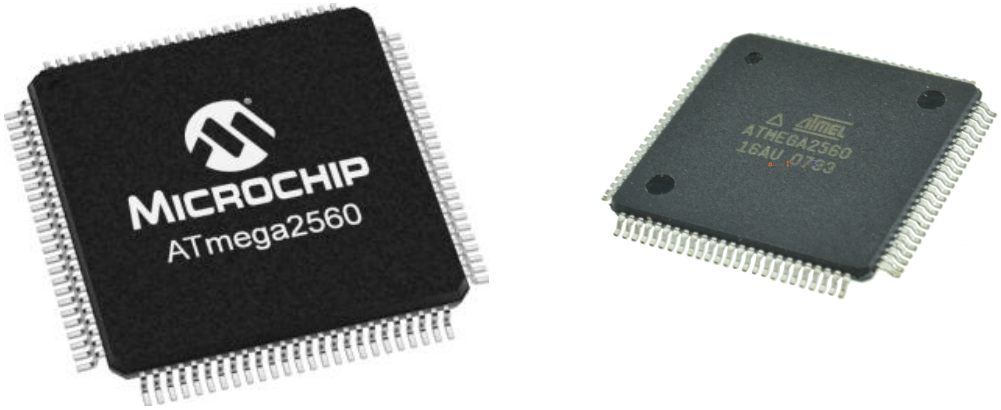
Figura 2.19: Microcontrolador 8051

- **Microcontroladores de Memória Eletricamente Apagável**

  Durante os anos 90 apareceram microcontroladores com memórias Electrically-Erasable Programmable Read-Only Memory (EEPROM), eletricamente apagáveis e programáveis, tal como a memória flash. Estes microcontroladores poderiam ser programados, apagados e reprogramados utilizando sinais elétricos.

  Têm memória não-volátil, e são usados em computadores e outros dispositivos eletrónicos para armazenar pequenas quantidades de dados que precisam ser salvos quando a energia é removida, por exemplo, dados de configuração do dispositivo. Apesar de no nome se declarar Read-Only, eles podem Read/Ler e Write/Escrever, visto que os códigos de erro no OBD-II podem ser removidos independentemente [49].

  Muitos microcontroladores atuais, como os da Microchip e da Atmel, incorporam a tecnologia de memória flash.



(a) Microchip (b) Atmel

Figura 2.20: Microcontrolador Microchip e Atmel ATmega2560

Além de dispositivos de uso geral, os microcontroladores especializados são produzidos para áreas como automotive, iluminação, comunicação e dispositivos de baixo consumo de energia. Eles também se têm tornado menores e mais potentes [50].

Figura 2.21: Microcontrolador Atmel 1401

Podemos então deduzir com isto que os microcontroladores assumem uma posição muito vantajosa na sociedade atual e, relativamente aos anos primordiais da sua criação, as vantagens para os seus utilizadores são enormes [51]:

- **Custo reduzido**

  Projetos com microcontroladores em geral dispensam diversos outros componentes, reduzindo o custo final.

- **Tamanho reduzido**

  São grandes circuitos dentro de um único chip. Essa característica permite a confeção de placas com pouquíssimos componentes de apoio ao microcontrolador.

- **Praticidade**

  Por se tratar em geral de circuitos digitais, o desenvolvimento de projetos com microcontroladores não esbarra em diversos problemas que geralmente ocorrem com outros tipos de circuitos.

– **Facilidade de desenvolvimento**

  Apesar de ser necessário um certo conhecimento de lógica digital e de programação, desenvolver um projeto utilizando um microcontrolador é em geral mais rápido e fácil que de outra maneira, visto que a maior parte dos projetos que utilizam microcontroladores não poderiam ser feitos de outra forma.

– **Facilidade de manutenção**

  Como os circuitos microcontrolados são menores que os circuitos discretos e utilizam menos componentes, a sua manutenção é mais ágil.

– **Facilidade de modificação**

  Por se tratar de um SW interno, a modificação de características do projeto microcontrolado é muito mais rápida.

Apesar do necessário conhecimento prévio para criar um projeto microcontrolado, só existem vantagens a partir do momento em que se domina essa tecnologia. Grande parte dos projetos que no passado eram feitos com centenas de componentes discretos, podem hoje ser feitos com microcontroladores e alguns circuitos de apoio, reduzindo o tempo de desenvolvimento e principalmente o custo desses produtos.

2.3.2 Sensores

Criados em 1950, os sensores tornaram-se ao longo dos anos peças fundamentais à automação. Estes produtos são responsáveis pela deteção de quaisquer movimentações no ambiente em redor, seja para contagem de material, controlo de direção, nível de fluídos e verificação de material dentro do recipiente, controlos de temperatura, proximidade, entre outros.

São dispositivos que têm a função de detetar e responder com eficiência a algum estímulo. Existem vários tipos de sensores que respondem a estímulos diferentes como por exemplo: calor, pressão, movimento, luz e outros. Depois do sensor receber o estímulo,

a sua função é emitir um sinal que seja capaz de ser convertido e interpretado pelos outros dispositivos [52].

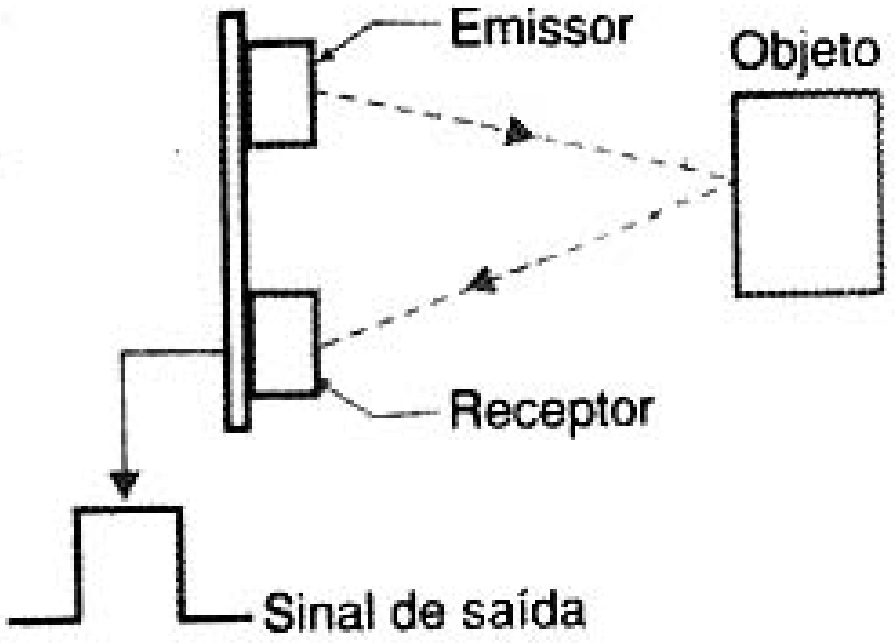


Figura 2.22: Exemplo de funcionamento de um Sensor

Os sensores dividem-se em [53]:

- **Analógicos**

  Podem assumir qualquer valor no seu sinal de saída ao longo do tempo, desde que esteja dentro da sua faixa de operação. Estas variáveis são mensuradas por elementos sensíveis, com circuitos eletrónicos não digitais.

- **Digitais**

  Assumem apenas dois valores no seu sinal de saída ao longo do tempo, que podem ser interpretados como 0 ou 1, não existindo grandezas físicas que assumam esses valores, mas eles são mostrados ao sistema de controlo após serem convertidos por um circuito eletrónico.

Em suma, a diferença entre sensores analógicos e digitais está no tipo de saída que eles fornecem. A escolha do sensor depende exclusivamente do objetivo da sua instalação, sendo preciso avaliar as condições do ambiente e optar pelo sensor mais adequado para a atividade. Para isso, existem vários tipos de sensores, cada um com o seu tipo de input e consoante atuação:

- **Sensores Elétricos**

  Dispositivo ou circuito integrado que deteta um parâmetro físico específico e o converte num sinal elétrico, sendo o sinal de saída processado e usado para fornecer uma medição ou para adicionar uma ação [54].

- **Sensor Magnético**

  Composto por uma pequena caixa plástica que tem no seu interior duas lâminas metálicas milimetricamente afastadas, deteta a magnitude do magnetismo e geomagnetismo gerado por um íman ou por uma corrente [55].

- **Sensor Indutivo**

  Cria um pequeno campo magnético na sua ponta e quando um metal passa próximo dele, perturbando o campo magnético, o sensor consegue captar essa perturbação e envia um sinal que pode ser interpretado por algum circuito ligado ao sensor, como um microprocessador [56].

- **Sensor Capacitivo**

  Deteta qualquer tipo de massa, logo, é aplicado onde existe a necessidade de deteção de materiais não metálicos como plásticos, madeiras e resinas, sendo utilizado também para deteção do nível de líquidos e sólidos [57].

- **Sensor Mecânico**

  Possui a capacidade para detetar as posições, movimentos ou presença por meio de recursos mecânicos. Utilizado para detetar a presença de objetos num determinado

lugar, a deteção de fechamento ou abertura de portas, sendo que o sensor de fim de curso é um dos mais conhecidos [58].

- **Sensor Térmico**

  Fornece uma determinada resposta quando é submetido a uma mudança de temperatura. Este tipo de sensor de temperatura é muito utilizado em ambientes onde é necessário manter uma determinada temperatura, enviando uma resposta quando percebe que a temperatura está fora do ideal, e de acordo com esta resposta, a refrigeração é ligada [59].

- **Sensor Acústico**

  Utiliza o retorno do eco que se propaga na velocidade do som, sendo um dos tipos de sensores usados para captar distâncias. É usado para medir/sentir um ambiente e converte essas informações num sinal de dados digital ou analógico que pode ser interpretado por um computador ou um observador [60].

- **Sensor de Fibra Ótica**

  Sensor que tem uma fibra ótica conectada a uma fonte de luz para permitir a deteção em espaços apertados ou onde um pequeno perfil é benéfico. Usado para fibras, modelos microprocessados, sistema de deteção da fibra por barreira ou foto-sensores e lentes opcionais para diversas aplicações [61].

- **Sensor Ultrassónico**

  Mede a distância de um objeto alvo, emitindo ondas sonoras ultrassônicas e converte o som refletido num sinal elétrico. Têm dois componentes principais: o transmissor, que emite o som usando cristais piezoelétricos e o receptor, que encontra o som depois de ele se deslocar de e para o alvo [62].

- **Sensor de Pressão**

  Deteta a pressão e converte-a num sinal elétrico analógico cuja magnitude depende da pressão aplicada, sendo esta definida como a força por unidade de área que um

fluido exerce ao seu redor. Como eles convertem a pressão num sinal elétrico, eles também são chamados de transdutores de pressão [63].

- **Sensor de Imagem**

  Converte fotões (luz) em sinais elétricos que podem ser interpretados pelo dispositivo. As primeiras câmaras digitais usavam dispositivos de carga acoplada, facilitando o movimento da carga elétrica através do dispositivo para que pudesse ser modulada [64].

- **Sensor Fotoelétrico**

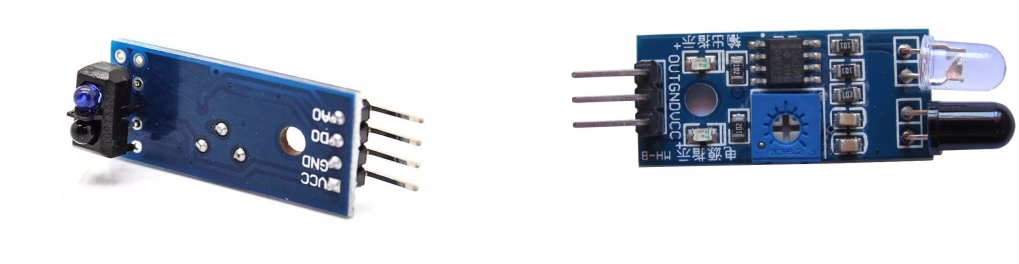
  Utilizam um emissor para emitir luz e um recetor para receber luz. Quando a luz emitida é interrompida ou refletida pelo objeto sensor, ela muda a quantidade deluz que chega ao recetor. Por sua vez, o recetor deteta essa alteração e converte- a numa saída elétrica [65].

Os sensores de linha, como é o caso do sensor utilizado neste projeto, fazem parte do conjunto de sensores fotoelétricos. O seu funcionamento é simples: quando algum obstáculo refletor (cor branca) é colocado à frente do Sensor IR Infravermelho, o sinal é refletido para o recetor. Quando isso acontece, o pino de saída OUT é colocado em nível baixo (0). Por outro lado, se a superfície do obstáculo for de cor preta, a radiação infravermelha não é refletida, permitindo que o sensor reconheça linhas pretas em fundo branco (e vice-versa), fazendo com que o robô siga essa linha.

Existe no mercado um conjunto de diversos tipos de sensores de linha, cada um com as suas características e com uma eficiência maior ou menor, tendo em conta o seu objetivo. Porém, a diferença entre eles, a nível de alimentação e de modo de atuação, é relativamente pequena. A maior diferença consiste no facto de alguns já virem equipados com 5 canais, sendo essa uma vantagem para a execução da leitura e seguimento da linha. Outro ponto que pode ser considerado vantajoso é que alguns destes sensores já vêm equipados com uma saída tanto analógica como digital, permitindo assim uma maior versatilidade.

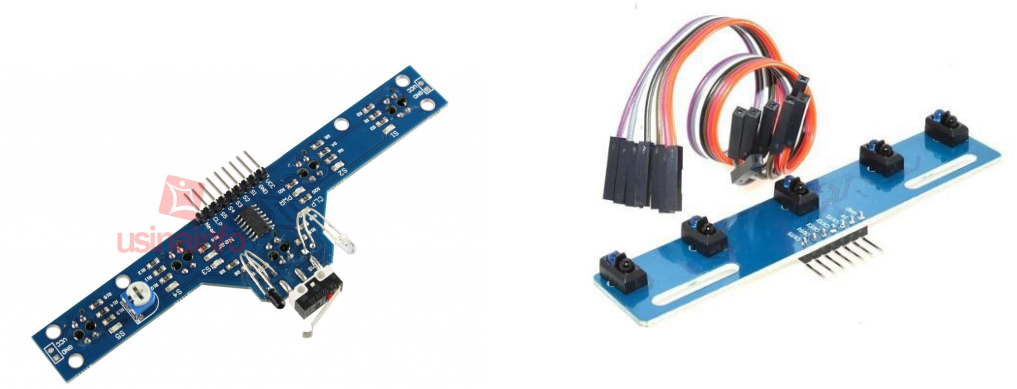
Em alguns sensores de 1 canal, como podemos observar na Figura 2.23, têm associados

na no seu circuito um potenciómetro, que permite ajustar a sensibilidade do sensor e a distância até uma linha. A melhor distância entre objetos como o solo e o sensor é de 1 a 3 cm.



(a) TCRT5000 (b) LM393

Figura 2.23: Sensores de linha de 1 canal



(a) BFD1000 (b) KB05

Figura 2.24: Sensores de linha de 5 canais

Capítulo 3

Desenvolvimento Prático

Feita uma contextualização teórica e prática de tudo o que envolve o projeto, desde o historial de várias tecnologias, o enquadramento recente das mesmas e algumas especificidades mais práticas desta temática, é necessário então fazer uma separação e entender quais são os materiais, tecnologias e especificações utilizadas.

3.1 Tecnologias Utilizadas

3.1.1 Arduíno UNO

Placa composta por um microcontrolador Atmel, circuitos de entrada/saída e que pode ser facilmente conectada à um computador e programada via IDE (Integrated Development Environment, ou Ambiente de Desenvolvimento Integrado), utilizando uma linguagem baseada em C/C++, sem a necessidade de equipamentos extras além de um cabo USB [66].

É uma placa de microcontrolador de código aberto baseada no microcontrolador Microchip ATmega328P e desenvolvida pela Arduíno.

A sua placa está equipada com um conjunto de pinos de entrada/saída digital e analógica (E/S) que podem ser conectados a várias placas de expansão (blindagens) e outros

circuitos. A placa possui 14 pinos digitais, 6 pinos analógicos e programável com o Arduíno IDE (ambiente de desenvolvimento integrado) por meio de um cabo USB tipo B, podendo ser alimentado pelo cabo USB ou por uma bateria externa de 9 volts, embora aceite voltagens entre 7 e 20 volts.

Esta placa é constituída pelos seguintes componentes [67]:

- Um microcontrolador;
- Reguladores de tensão que estão adequados ao microcontrolador;
- Conjuntos de entradas/saídas analógicas e digitais;
- Interface USB para a programação e interação com o computador.

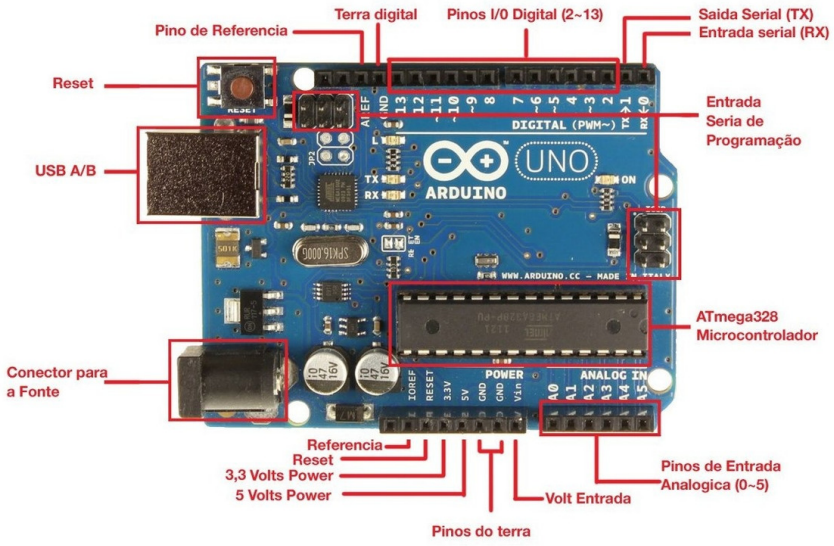


Figura 3.1: Arduino UNO

No circuito regulador para entrada externa, o CI responsável pela regulação de tensão é o NCP1117, da OnSemi, usando também um díodo D1 que protege o circuito caso uma fonte com tensão invertida seja ligada.

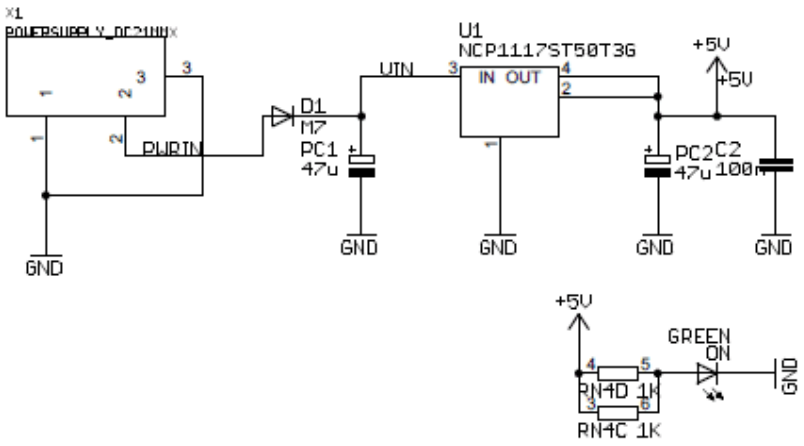


Figura 3.2: Arduino - Circuito Regulador para Entrada Externa

Quando o cabo USB é ligado a um PC, a tensão não precisa ser estabilizada pelo regulador de tensão. Dessa forma a placa é alimentada diretamente pela USB [68].

O circuito da USB apresenta alguns componentes que protegem a porta USB do computador em caso de alguma anormalidade. Na figura abaixo é exibido o circuito de proteção da USB da placa Arduíno UNO.

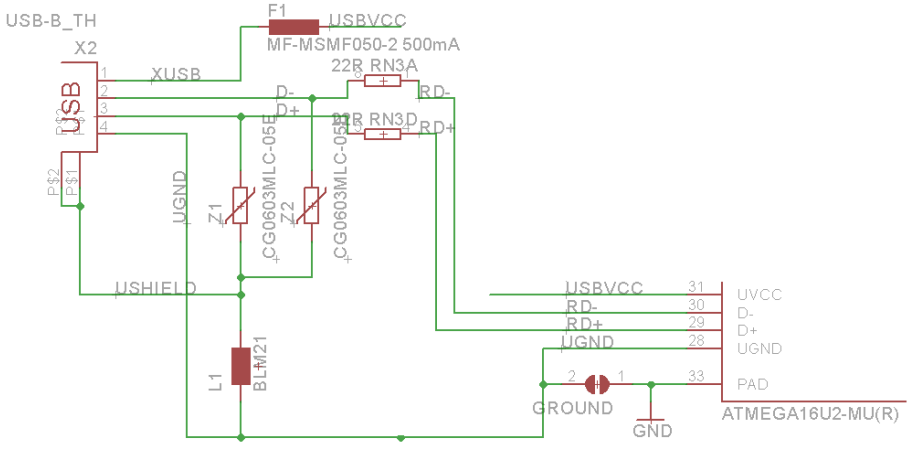


Figura 3.3: Arduino - Circuito Regulador para USB

3.1.2 Carro AlphaBot

AlphaBot é uma plataforma de desenvolvimento robótico compatível com RPi2 e Arduíno. Consiste na placa principal AlphaBot, no chassi móvel e em tudo que é necessário para o mover.

Basta conectar uma placa controladora, no caso deste projeto, o Arduíno, e combinada com o código que mais for conveniente de modo a executar as funcionalidades pretendidas. Esta plataforma esta diretamente relacionado com as seguintes tecnologias [69]:

- Linha – rastreamento;
- Prevenção de obstáculos;
- Monitorização de vídeo;
- WiFi;
- Bluetooth;
- ZigBee;
- Infravermelho;
- Controlo remoto, etc.

O Alphabot tem na sua constituição os seguintes componentes [70]:

- Um shield para a placa Arduino;
- Dois monitores;
- Um modulo seguidor de linha,
- Uma ponte-H L298P;
- Um regulador de tensão LM2596.

Figura 3.4: Carro AlphaBot

O Alphabot usas duas baterias da série 18650 para fornecer energia com uma entrada tensão de 7,4V. A tensão de entrada é tratada pelo regulador LM2596, que posteriormente a enviará para o Arduíno.

3.1.3 Sensor Ótico IR – Sensor de Linha

O sensor ótico IR (Tracker module), consiste num sensor infravermelho ITR20001/T que foi desenvolvido para aplicação em projetos robóticos e, é principalmente aplicado no desenvolvimento de projetos de "robôs seguidores de linha".

Os "robôs seguidores de linha" são projetos que funcionam a partir de linhas pretas demarcadas no chão, desde que o chão seja totalmente branco, ou de linhas brancas, desde que o chão seja totalmente preto. O robô seguirá estas linhas, que definirão o trajeto a ser seguido graças ao sensor presente.

Para que o robô possa seguir a linha predefinida é necessária a presença do sensor de linha infravermelho, que é o responsável por emitir diferentes frequências de luzes para identificação do caminho, que ao refletirem na linha demarcada voltam ao sensor, que por sua vez processa as informações e mantém o "robô seguidor de linha" no caminho correto. Este módulo é baseado em 5 sensores infravermelhos inferiores que emitem e recebem sinais luminosos, possibilitando, deste modo a identificação do trajeto, por meio de linhas pré-demarcadas, mesmo em trajetos com curvas agudas.

Figura 3.5: Sensor Infravermelho ITR20001/T

3.1.4 Módulo NRF24L01

O módulo NRF24L01 é um componente que reúne em si a capacidade de realizar a transmissão e receção de um sinal. Usa a banda de 2,4 GHz e pode operar com taxas de transmissão de 250 kbps a 2 Mbps. Se usado em espaço aberto e com menor taxa de transmissão, o seu alcance pode atingir até 100 metros.

O consumo de energia deste módulo é de apenas 12mA durante a transmissão, que é ainda mais baixo que um único LED. A tensão de operação do módulo é de 1,9 a 3,6V, mas o bom é que os outros pinos toleram a lógica de 5V, para que possamos conectá-lo facilmente a um Arduíno sem usar conversores de nível lógico [71].

Três desses pinos destinam-se à comunicação SPI e precisam estar conectados aos pinos SPI do Arduíno. Os pinos CSN e CE podem ser conectados a qualquer pino digital da placa Arduíno e são usados para configurar o módulo no modo de espera ou ativo, bem como para alternar entre o modo de transmissão ou comando. O último pino é de interrupção que não precisa ser usado.

A comunicação entre o módulo NRF24L01 e o Arduíno é feita através do protocolo SPI e opera a uma tensão de 3,3V [72].

Figura 3.6: Módulo NRF24L01

3.1.5 LCD1602 Qapass

O LCD é utilizado com a finalidade de nos permitir fazer um registo do tempo que o carro demora a fazer o percurso. Para isso, é utilizado paralelamente a este LCD 2 botões, de modo a poder iniciar a contagem, fazer a sua paragem e também ser possível fazer um RESET para reiniciar.

Tem 4 pinos na lateral esquerda do módulo, dois pinos são para alimentação (VCC e GND), e os outros dois pinos são da interface I2C (SDA e SCL). O potenciómetro da placa serve para ajuste do contraste do display, e o jumper na lateral do lado oposto permite que a luz de fundo (backlight) seja controlada pelo programa ou permaneça apagada, o que facilita a visualização do dados, contribuindo para o aperfeiçoamento do projeto [73].

Figura 3.7: LCD1602 Qapass

3.1.6 Módulo Joystick KY-023

O modelo de Joystick utilizado neste projeto é do tipo 3 eixos. Este Joystick tem o seu princípio de funcionamento através do controlo de 2 potenciómetros e um botão [74].

Neste projeto, porém, o eixo Z não é utilizado devido à pinagem reduzida do modelo. Este joystick tem como objetivo o controlo manual do carro pelo utilizador.

Figura 3.8: Módulo Joystick KY-023

3.2 Desenvolvimento de Hardware e Software

Para a elaboração deste projeto, foi necessária a realização do desenvolvimento do projeto a nível de programação e, paralelamente, ajustar todo o hardware necessário para o seu correto funcionamento. Para além disto também foram desenhados alguns modelos de pistas, de modo a que o carro percorresse um determinado percurso.

3.2.1 Pistas Desenvolvidas

Para a realização das Pistas para o carro percorrer, foram tomados em conta alguns pontos que pareceram importantes:

- Fazer 2 tipos de pistas e, para cada tipo, fazer 2 com dimensões diferentes, de modo a compreender, com a diferença de distâncias percorridas, como o sensor de linha se comportaria relativamente ao controlo manual;
- Em algumas pistas utilizar curvas e em outras apenas retas, nas quais as curvas seriam simuladas com retas mais pequenas, de modo a perceber o comportamento do sensor nas curvas relativamente às retas.

Tendo como base estes pontos foram criadas as seguintes pistas:

- A pista A e a B com dimensões mais reduzidas e apenas com retas.

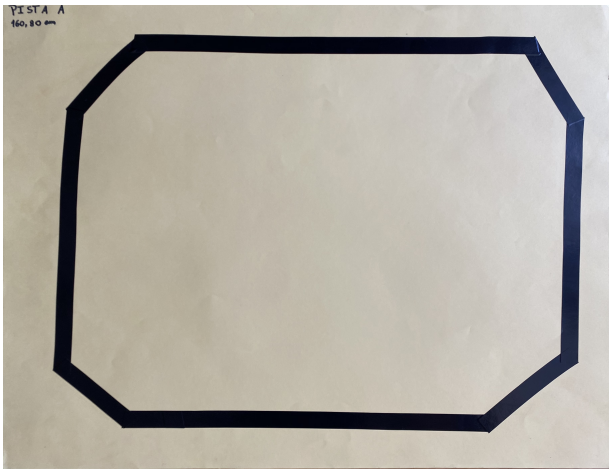


Figura 3.9: Pista A - 160.80 cm

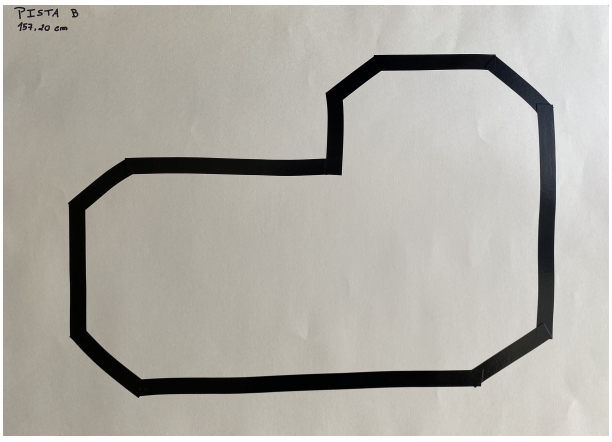


Figura 3.10: Pista B - 157.20 cm

- As pistas C e D já com dimensões maiores e com retas e curvas no seu percurso.

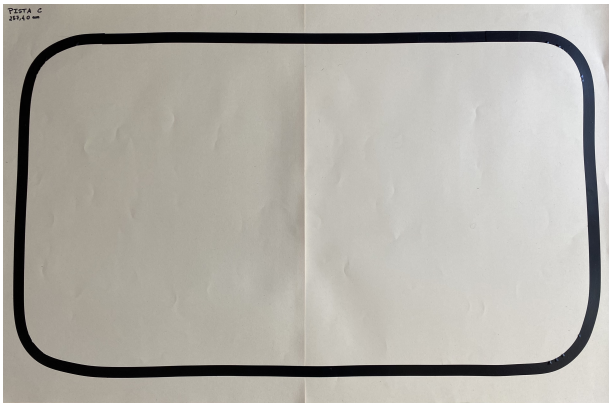


Figura 3.11: Pista C - 257.40 cm

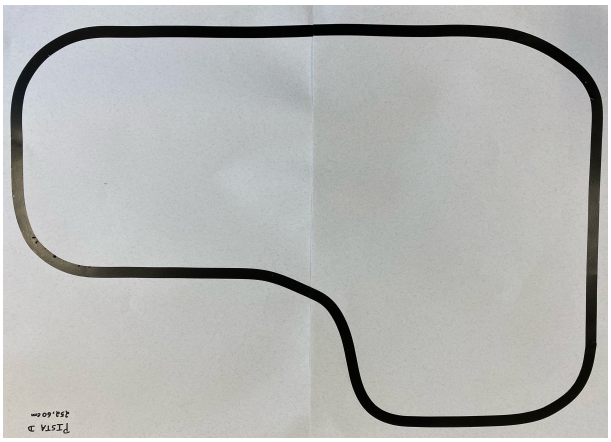


Figura 3.12: Pista D - 252.60 cm

3.2.2 Circuitos Elétricos Desenvolvidos

Com vista a um melhor entendimento de todo o processo necessário para gerar este projeto, foi necessário haver um estudo e uma pesquisa inicial de como desenvolver todos os circuitos elétricos necessários para uma correta funcionalidade.

Após esse estudo e depois de, em todos os circuitos, terem sido feitas várias tentativas com a finalidade a aprimorar ao máximo a funcionalidade, chegou-se a um estado final. Através da utilização do SW *Fritzing*, é feita uma representação dos circuitos que compõem todo este projeto. Na Figura 3.13 temos representado o esquema elétrico do joystick, na Figura 3.14 o do carro e na Figura 3.15 o do cronómetro.

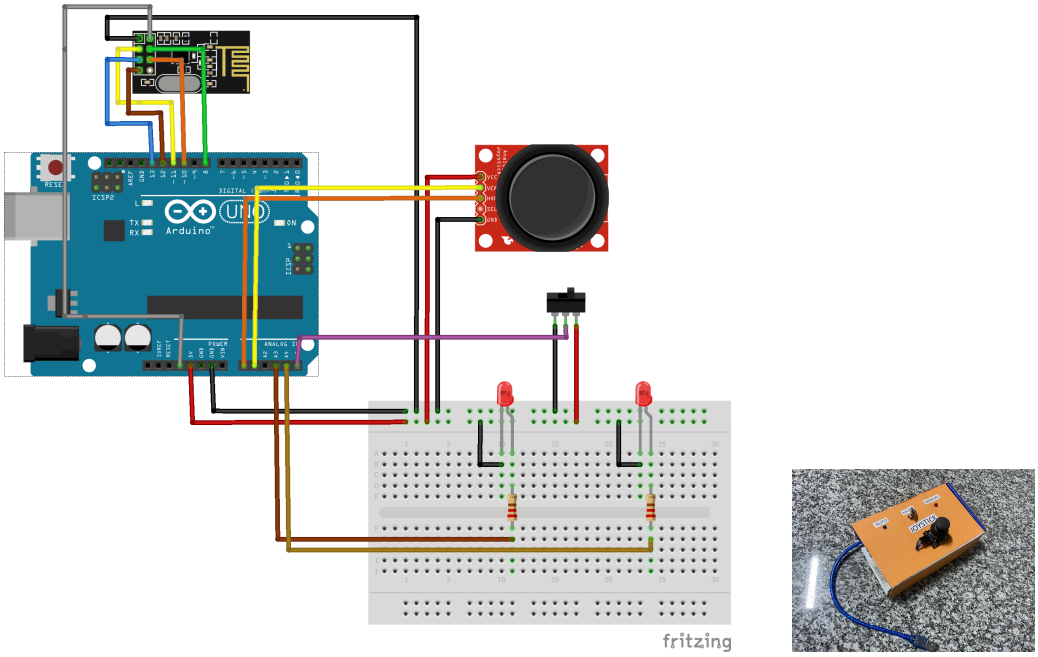


Figura 3.13: Circuito Elétrico do Comando

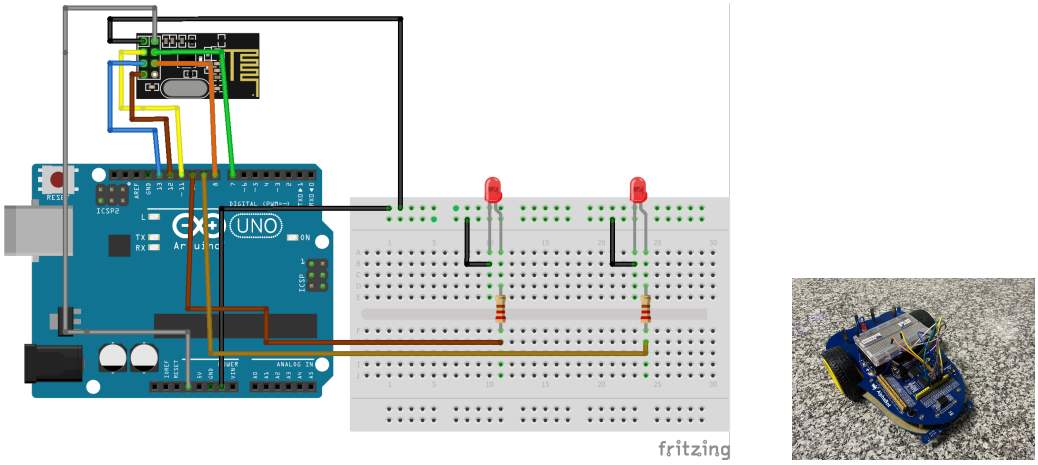


Figura 3.14: Circuito Elétrico do Carro

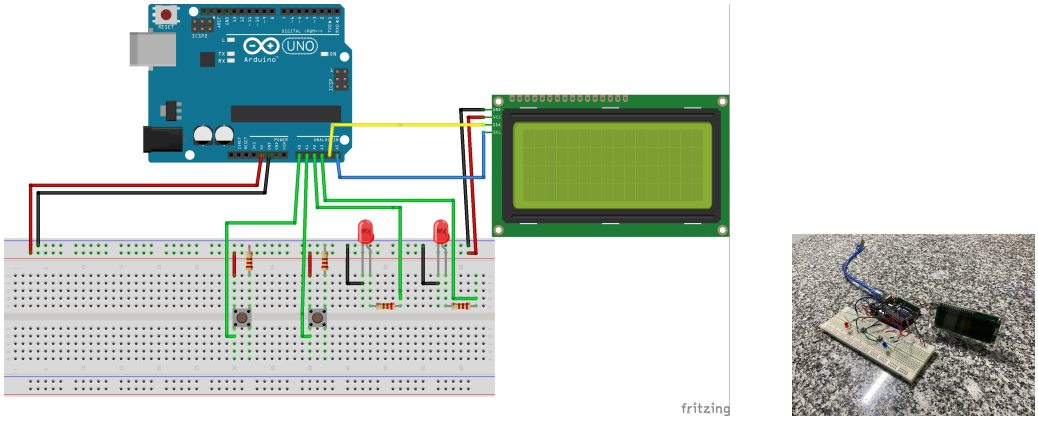


Figura 3.15: Circuito Elétrico do Cronómetro

3.2.3 Bibliotecas e Interfaces - Arduíno

Como já foi referido anteriormente, a nível programático, foi utilizado neste projeto o Arduíno IDE. A programação foi bastante delicada, tendo em conta a complexidade do projeto, uma vez que era necessário aliar várias funcionalidades do sistema a interagir, de modo a retirar os melhores resultados possíveis.

Para a implementação desta programação, foram necessárias a instalação de vários tipos de bibliotecas para que todos os módulos pudessem funcionar corretamente e interagir entre si. As bibliotecas necessárias foram:

- <RH_NRF24.h> - Utilização do modulo NRF24L01 [75];
- <SPI.h> - Para o módulo NRF24L01 já que usa o protocolo SPI [76];
- <EEPROM.h> - Armazenar na memória os dados dos cartões [77];
- "TRsensors.h" – Utilizada para o correto funcionamento de Sensor de Linha [78];
- <Wire.h> - Necessária para o funcionamento do LCD1602 Qapass no Arduíno IDE [79];
- <Adafruit_GFX.h> - fornece uma sintaxe comum e um conjunto de funções gráficas para todos os nossos monitores LCD e LED, permitindo que o Arduíno se adapte facilmente entre os tipos de exibição com o mínimo de barulho [80];
- <LiquidCrystal_I2C.h> - Permite controlar displays I2C com funções extremamente semelhantes à biblioteca LiquidCrystal [81];
- <Bounce2.h> - A parte mecânica dos botões e interruptores vibram ligeiramente quando fechados ou abertos, causando vários estados falsos indesejados (semelhantes ao ruído). Esta biblioteca filtra essas mudanças de estado indesejadas [82].

Para além disto e através de uma revisão de todo o manual alphabot, foi também necessário fazer uma correta definição das interfaces para os motores. Foi feita a definição seguinte:

Interface	**Arduino**
ENA	5
IN1	A1
IN2	A0
ENB	6
IN3	A2
IN4	A3

Tabela 3.1: Definição de Interfaces para Arduíno

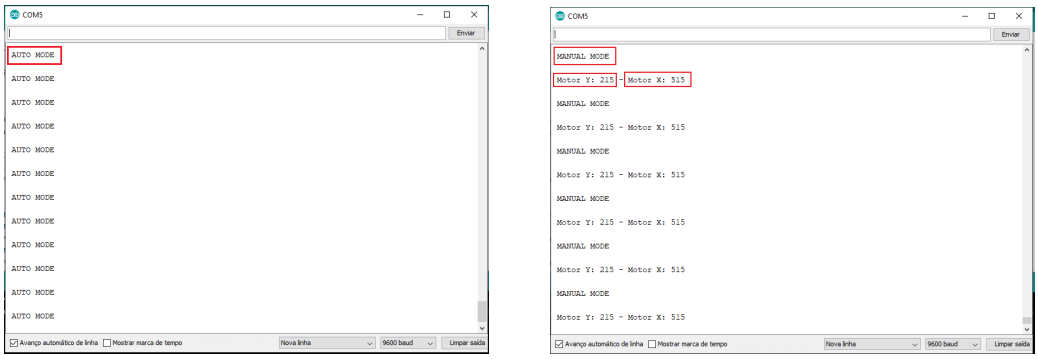
As interfaces IN1 e IN2 estão ligadas ao motor esquerdo, enquanto IN3 e IN4 estão conectados ao motor direito. Por sua vez, as interfaces ENA e ENB são pinos de saída. Quando são levados a HIGH, o pulso Pulse Width Modulation (PWM) será enviado de IN1, IN2, IN3 e IN4, com a finalidade de controlar a velocidade do robô [83].

Após a instalação de todas as bibliotecas, da correta montagem e ligação de todos os componentes, foi, então, feita toda a programação necessária para o correto funcionamento dos vários intervenientes. Estes códigos desenvolvidos encontram-se nos **Anexos** desta dissertação.

3.2.4 Programação e Valores Lógicos - Arduíno

Através de uma análise de todos os componentes utilizados e de uma posterior reflexão sobre os mesmos, deparamo-nos com o facto de que, para conseguirmos controlar o carro, temos que gerir sempre toda a programação consoante os valores lógicos registados a partir do comando, fazendo com isto a leitura do valor registado para$X$ e para Y respetivamente.

Aqui, existe um botão que permite escolher entre a opção de ter o carro a funcionar de modo automático e de modo manual (aliado a 2 leds que permite a sua visualização). Esta informação, utilizando o Módulo NRF24L01, é enviada para o carro , como se pode observar na Figura 3.16 (a) e, caso o modo manual seja o escolhido, os valores analógicos do joystick serão lidos e enviados, tal como na Figura 3.16 (b) .



(a) Modo Automático (b) Modo Manual

Figura 3.16: Monitor Série do Comando - Modo

Para um correto envio de toda esta informação, toda a programação relativamente ao NRF24L01 assume um papel preponderante, tanto a nível do comando como no carro. Utilizando corretamente a biblioteca RF24, mencionada anteriormente, conseguimos fazer uma configuração exata de toda esta comunicação.

Inicialmente, é necessário criar um "objeto RF24", utilizando dois argumentos, os pinos CSN e CE:

$$RF24\,radio(8, 10);\ //CE, CSN \tag{3.1}$$

Os pinos CSN e CE podem ser selecionados do modo que seja mais conveniente para as ligações existentes. Assim, estes pinos no comando e no carro não necessitam de ser

obrigatoriamente os mesmos.

Em seguida, foi necessário criar uma matriz de bytes que representa o endereço, através do qual os dois módulos comunicam entre si. Aqui, tanto a nível do comando, como no carro, a direção tem que ser a mesma:

$$const\, byte\, address[6] = \text{"00001"}; \tag{3.2}$$

Podemos alterar o valor desta direção para qualquer string de 5 letras, o que permite escolher com qual recetor comunicaremos. Neste caso, teremos a mesma direção em ambos os recetores e o transmissor.

No transmissor (comando), seção de configuração (*void setup()*), é necessário inicializar o "objeto RF24"e usar o função *radio.openWritingPipe()* com o objetivo de definir a direção do recetor para o qual enviaremos a string de 5 letras definida anteriormente:

$$radio.openWritingPipe(address); \tag{3.3}$$

Por outro lado, no recetor (carro), é necessário usar a função *radio.setReadingPipe()*, definindo a mesma direção e, assim, permitimos a comunicação entre os dois módulos:

$$radio.openReadingPipe(0, address); \tag{3.4}$$

Depois, em ambos os módulos, utilizamos a função *radio.setPALevel()*, no qual é definida o nível do amplificador de potência que, no nosso caso, é definido no máximo, uma vez que a possibilidade dos módulos se afastarem um do outro é alta:

$$radio.setPALevel(RF24\_PA\_MAX); \tag{3.5}$$

Neste caso, é recomendado o uso de capacitores de bypass através do GND e 3,3 V dos módulos, para que tenham uma tensão mais estável durante a operação.

Seguidamente, temos a função *radio.stopListening()* que define o módulo como transmissor e , do outro lado, temos a função *radio.startListening()* que define o módulo como recetor [84]:

$$radio.stopListening(); \; //Transmitter \tag{3.6}$$

$$radio.startListening(); \; //Receiver \tag{3.7}$$

Com esta inicialização correta, foi então feito do lado do utilizador a criação de um array que nos permitisse enviar os valores de X e Y do joystick:

$$int\, joystick[2]; \tag{3.8}$$

Com isto, dentro do *void loop()* fazemos a leitura dos valores lógicos dos pinos onde estão ligados as entradas X e Y do joystick e, depois disso, enviamos essa informação para o carro, utilizando o *radio.write()*:

$$joystick[0] \; = \; analogRead(A0); \tag{3.9}$$

$$joystick[1] \; = \; analogRead(A1); \tag{3.10}$$

$$radio.write(joystick, \; sizeof(joystick)); \tag{3.11}$$

Estes valores registados são importantes para o modo de funcionamento, sendo enviados apenas quando o modo manual está selecionado. Quando o modo automático é o que esta a ser usado, então, para ser feita uma distinção, em vez de serem enviados os *analogRead()* de cada um, são enviados 2 valores negativos, para assim, do lado do carro, quando estes valores são recebidos, saberá imediatamente que o modo automático é o selecionado. Isto deve-se ao fato de que, quando são lidos os valores lógicos nos pinos,

estes nunca assumem valores negativos, daí ser possível esta diferenciação:

$$joystick[0] = -20; \tag{3.12}$$

$$joystick[1] = -20; \tag{3.13}$$

Do lado do carro criamos duas variáveis que permitem armazenar os valores recebidos do joystick, definindo também o array para receber os valores de X e Y do joystick, tal como no transmissor:

$$int\, xAxis,\, yAxis; \tag{3.14}$$

$$int\, joystick[2]; \tag{3.15}$$

No *void loop()* fazemos a leitura dos valores de X e Y recebidos do joystick, através da utilização do *radio.read()* e, depois disso, fazemos com que as variáveis que criamos anteriormente assumam os valores que vão sendo recebidos, permitindo depois com isso toda a lógica a partir destes valores:

$$radio.read(joystick,\, sizeof(joystick)); \tag{3.16}$$

$$yAxis = joystick[0]; \tag{3.17}$$

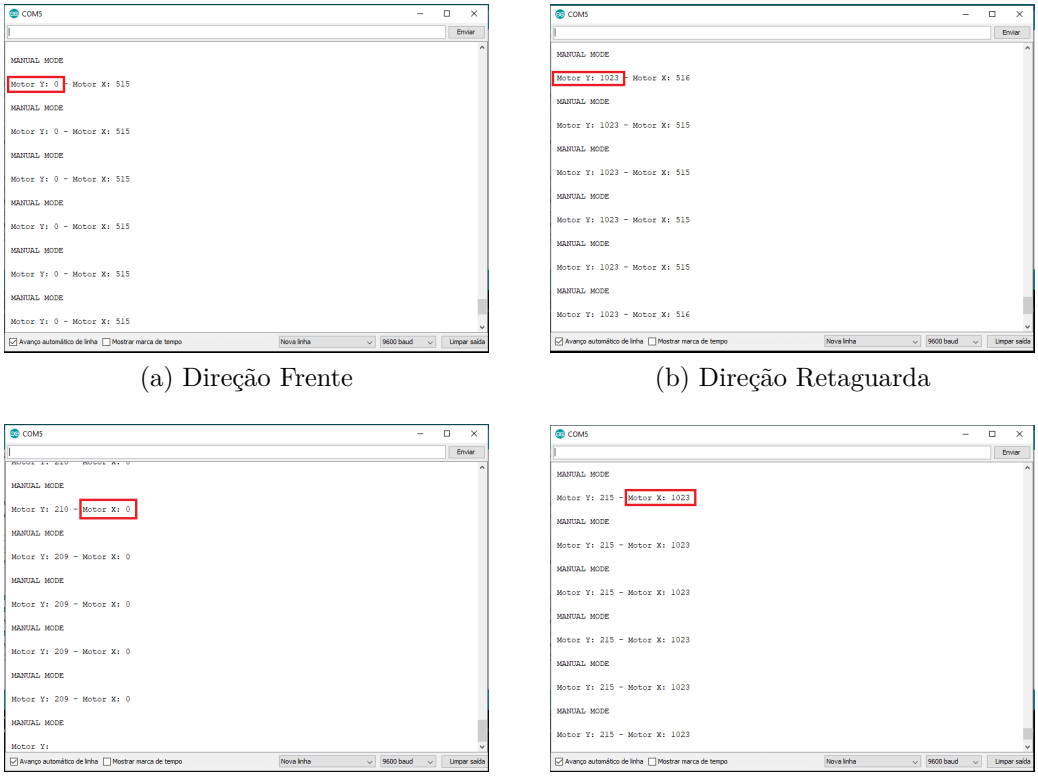
$$xAxis = joystick[1]; \tag{3.18}$$

Depois de esta comunicação ser feita de modo correta, é então necessária uma reflexão sobre os valores que são lidos nas portas lógicas para que, a partir deles, o carro se mova na direção pretendida.

Através de uma análise destes valores, é possível assumir que quando a leitura destes

valores é feita (modo manual) e, se o joystick não for mexido, o carro terá que estar parado, sendo os valores registados os observados na Figura 3.16 (b), sendo textitX=215 e para Y=515.

Com isto, é possível saber, do lado da programação do carro, quais os valores no qual ele não se irá mover. Assim sendo,o próximo passo foi perceber quais os valores que as portas lógicas enviam em cada caso (seleção no joysitck para a frente, retaguarda, esquerda e direita), como podemos observar na Figura 3.17.



(a) Direção Frente (b) Direção Retaguarda

(c) Direção Esquerda (d) Direção Direita

Figura 3.17: Monitor Série do Comando - Direção

Analisando detalhadamente os valores que são recebidos, é possível observar que existem especificações comuns às 4 direções:

- Se o joystick for selecionado para a frente ou para a retaguarda, o valor de X mantém-se igual ao valor de quando este está parado (X=215);

- Do mesmo modo, se a seleção for feita para a esquerda ou para a direita, então é o valor de Y que não se irá alterar relativamente a quando está parado (Y=515).

Assim sendo, é possível depois retirar os valores para cada caso e, assim, poder preparar o carro para agir de acordo com cada valor recebido:

- Direção - Frente : Y=0 (Figura 3.17 (a));
- Direção - Retaguarda : Y=1023 (Figura 3.17 (b));
- Direção - Esqueda : X=0 (Figura 3.17 (c));
- Direção - Direita : X=1023 (Figura 3.17 (d)).

De notar que as alterações destes valores são feitas de forma crescente ou decrescente, respetivamente, e não de modo instantâneo. Esta caraterística é importante na medida em que o carro, ao ler todos estes valores, tem que estar preparado para lidar com eles e não simplesmente lidar apenas com determinados valores específicos.

Depois de toda esta análise e de todos estes dados recolhidos, foi então possível preparar o carro para todo o tipo de informação que iria receber. Analisando o código elaborado do carro, presente nos **Anexos**, é possível observar que, caso o modo manual esteja selecionado do lado do comando, então vai ser feita a receção dos valores das portas lógicas e, com isso, o carro irá atuar de acordo com os valores recebidos. Por outro lado, se o modo automático for o selecionado, então o sensor de linha presente irá atuar e o carro agirá sem qualquer tipo de controlo, como referido anteriormente.

Para preparar o carro, no caso do modo manual, para os valores que este irá receber, foi importante a utilização da função *map()* [85]:

$$Z = map((value, fromLow, fromHigh, toLow, toHigh); \tag{3.19}$$

No qual :

- value: o número a ser mapeado;

- fromLow: o menor limite do intervalo atual do valor;
- fromHigh: o maior limite do intervalo atual do valor;
- toLow: o menor limite do intervalo alvo
- toHigh: o maior limite do intervalo alvo

Esta permite efetuar o mapeamento de um intervalo numérico noutro intervalo numérico desejado, significando isto que num intervalo numérico, que vai de um valor mínimo até um valor máximo, o valor mínimo será mapeado num novo mínimo, e o valor máximo será mapeado num novo máximo, sendo os valores intermédios remapeados em novos valores intermédios, de forma correspondente [86]. Este remapeamento vai permitir ao carro assumir determinada trajetória deste valor estipulado até ao valor máximo (no caso de se mover para a retaguarda ou para a direita o valor é 1023, respetivamente para Y e X)) ou mínimo de referência (no caso de se mover para a frente ou para a esquerda o valor é 0, respetivamente para Y e X), respetivamente.

Tendo isto em mente, foi possível então mapear cada caso, tendo em conta os valores recebidos e também o mapeamento que lhe é mais conveniente:

- Direção - Frente:
  - Condição: $Y < 150$;
  - Mapeamento: *map(Y, 150, 0, 0, 255).*
- Direção - Retaguarda:
  - Condição: $Y > 565$;
  - Mapeamento: *map(Y, 565, 1023, 0, 255).*
- Direção - Esquerda:
  - Condição: $X < 300$;
  - Mapeamento: *map(X, 300, 0, 0, 255).*

- Direção - Direita:
  - Condição: $X > 565$;
  - Mapeamento: *map(X, 565, 1023, 0, 255).*

De notar também que este mapeamento é sempre feito para os motores das duas rodas. A única diferencia de mapeamento entre os casos é que, para a frente e para a retaguarda, o mapeamento, para ambas as rodas, é feito da mesma maneira, visto que as 2 rodas têm o mesmo comportamento. Porém para a esquerda e para a direita, este mapeamento é feito de forma similar, visto que para cada uma das direções, uma das rodas vai diminuir o seu valor e a outra vai aumentar, para o carro se conseguir mover de forma correta. Para isso, depois de mapeado o valor recebido, este valor vai ser somado ao valor atual de uma das rodas e diminuído à outra:

- Direção - Esquerda:

$$xMapped = map(X, 300, 0, 0, 255); \tag{3.20}$$

$$motorSpeedA = motorSpeedA + xMapped; \tag{3.21}$$

$$motorSpeedB = motorSpeedB - xMapped; \tag{3.22}$$

- Direção - Direita:

$$xMapped = map(X, 565, 1023, 0, 255); \tag{3.23}$$

$$motorSpeedA = motorSpeedA - xMapped; \tag{3.24}$$

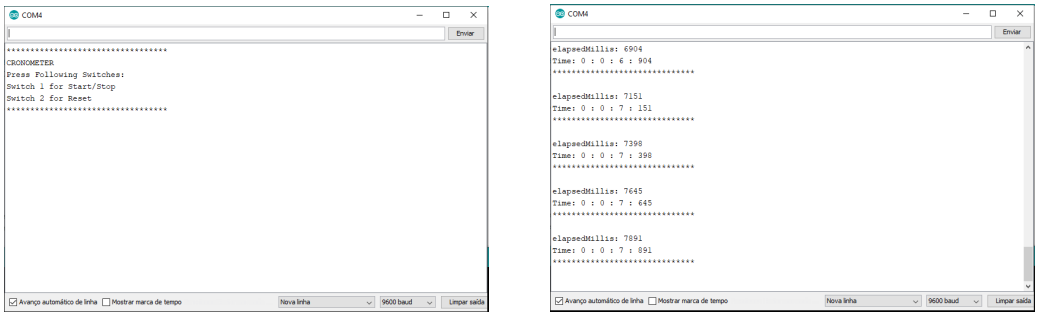
$$motorSpeedB = motorSpeedB + xMapped; \tag{3.25}$$

Foi também possível delinear, como é observável nas funções utilizadas, que os valores mínimo e máximo foram sempre de 0 a 255, podendo deste modo a velocidade máxima ser a mesma para qualquer direção selecionada e, foi usado uma proteção no código

de, caso o valor fosse menor que o mínimo ou maior que o máximo, então o carro iria automaticamente assumir os valores mínimo e máximo respetivamente.

Através de toda esta programação, aliada a alguma investigação necessária para conseguir fazer o melhor alinhamento neste contexto, foi então possível colocar o carro e o joystick a funcionar e a comunicar corretamente.

Relativamente ao cronómetro, foi criado uma programação que permitisse, através de uso de 2 botões, executar início, a pausa e o reset.



(a) Opções (b) Cronómetro a correr

Figura 3.18: Monitor Série do Cronómetro

Através disto, foi possível a sua utilização para cronometrar o tempo que o carro demorou a executar cada uma das pistas que foram referidas anteriormente, tanto no modo automático como no modo manual e, assim, fazer o seu registo. Este código também se encontra nos **Anexos**.

Capítulo 4

Resultados Obtidos

Depois de finalizado todo o processo de desenvolvimento, tanto a nível de SW como de HW, foi possível fazer uma extração detalhada de valores para todos os percursos desenvolvidos. Neste processo, foram feitas amostras de 10 repetições para percorrer cada uma das pistas, sendo isto feito tanto para o carro a mover-se de modo automático, como para o mesmo ser feito de modo manual e, posteriormente, é feita uma comparação para ser compreendida a diferença de tempos entre ambos.

Após esta comparação, também é feito o cálculo da velocidade média para cada um dos contextos, permitindo assim averiguar a efetividade do sensor, perceber os contextos em que ele tem um melhor comportamento e, aliado a isso, ter uma perceção mais específica da diferença entre executar o percurso pretendido em modo automático ou modo manual.

4.1 Pista A

	Tempo(s)		
	Auto	**Manual**	**Dif**
Valor 1	14.90	08.33	06.57
Valor 2	09.77	07.02	02.75
Valor 3	08.82	08.06	00.76
Valor 4	09.42	05.32	04.10
Valor 5	08.55	06.37	02.18
Valor 6	08.29	06.89	01.40
Valor 7	14.13	07.97	06.16
Valor 8	08.91	07.36	01.55
Valor 9	08.94	07.12	01.82
Valor 10	09.97	06.16	03.81
Média	10.17	07.06	03.11

Tabela 4.1: Tempos percorridos na Pista A

Através de uma análise dos valores retirados na primeira pista, é possível observar que a diferença de tempo entre ambas as formas de controlo de carro, de um ponto de vista geral, têm alguma margem.

Conseguimos observar também que no modo automático foram registadas 2 amostras com valores um pouco dispares relativamente aos outros, amostras estas que obviamente têm influência no valor final.

A diferença da média de valores assume um valor muito alto, porém, visto ser uma pista pequena, ainda é um valor com algum impacto.

4.2 Pista B

	Tempo(s)		
	Auto	**Manual**	**Dif**
Valor 1	08.63	07.63	01.00
Valor 2	09.32	07.99	01.33
Valor 3	11.01	09.45	01.56
Valor 4	09.08	07.23	01.85
Valor 5	09.05	07.82	01.23
Valor 6	11.96	09.64	02.32
Valor 7	08.47	07.19	01.28
Valor 8	08.58	07.43	01.15
Valor 9	09.19	07.93	01.26
Valor 10	09.88	06.79	03.09
Média	09.52	07.91	01.61

Tabela 4.2: Tempos percorridos na Pista B

Nesta segunda experiência é possível observar que a variação de tempo entre o tipo de controlo do carro não assume valores muito diferentes, pelo contrário.

Neste caso, não existem amostras com uma grande disparidade relativamente às anteriores, o que leva à dedução que caso o carro não tenha nenhum tipo de atraso, nestas pistas mais pequenas, a diferença de valores será sempre reduzida.

4.3 Pista C

	Tempo(s)		
	Auto	**Manual**	**Dif**
Valor 1	19.33	08.37	10.96
Valor 2	17.23	09.86	07.37
Valor 3	16.05	10.29	05,76
Valor 4	10.50	11.11	-00.61
Valor 5	16.30	12.50	03.80
Valor 6	16.07	09.80	06.27
Valor 7	18.73	09.92	08.81
Valor 8	15.11	09.73	05.38
Valor 9	20.48	09.25	11.23
Valor 10	19.16	10.85	08.31
Média	16.90	10.19	06.71

Tabela 4.3: Tempos percorridos na Pista C

Nesta pista, uma pista com uma dimensão já bastante maior que as 2 anteriores, observa-se que a diferença de tempo entre ambos já assume um valor bastante maior.

Observamos neste caso que, no modo Automático, houve 3 amostras que assumiram valores bastantes dispares relativamente à média final, entre os quais 1 deles é notório por ser bastante mais pequeno que os outros.

Estes valores acabam por ser referência para o facto de, num ponto de vista geral, a diferença de valores ser tão grande, já que no modo manual apenas 1 amostra é que têm um valor ligeiramente maior. Porém, se notarmos a diferença entre as amostras dispares e a média, a amostra do modo manual continua a estar bastante mais próxima da média que as amostras do modo automático para a sua média, respetivamente.

De notar também que, nesta pista, foi possível observar uma amostra no qual o tempo no modo automático foi mais baixo que no modo Manual.

4.4 Pista D

	Tempo(s)		
	Auto	**Manual**	**Dif**
Valor 1	17.08	10.96	06.12
Valor 2	15.86	13.75	02.11
Valor 3	13.76	11.95	01.81
Valor 4	15.37	12.09	03.28
Valor 5	16.45	12.59	03.86
Valor 6	16.37	12.55	03.82
Valor 7	18.16	16.17	01.99
Valor 8	16.80	12.46	04.34
Valor 9	12.36	12.34	00.02
Valor 10	16.65	11.06	05.54
Média	15.87	12.59	03.28

Tabela 4.4: Tempos percorridos na Pista D

Nesta última pista e, tendo em conta a sua dimensão, denota-se que a diferença de valores é bastante reduzida.

Tal como na Pista B, não existem amostras que assumam um valor demasiado díspar relativamente à média global, no caso manual, o que nos permite dizer que, caso não ocorra nenhum tipo de problema a percorrer o circuito, o tempo que este é percorrido no modo automático não é assim tão diferente do modo Manual.

Porém, observando mais atentamente o modo manual, é notório que foi recolhida uma amostra com um valor díspar e, apesar de esta acontecer por "erro humano", este valor acaba também por influenciar a média global e favorecer a aproximação entre ambas as médias.

4.5 Comparação de Velocidades

	Velocidade(cm/s)		
Pistas	**Auto**	**Manual**	**Dif**
A - 160.80 cm	15.81	22.07	06.26
B - 157.20 cm	16.51	19.87	03.36
C - 257.40 cm	15.23	25.26	10.03
D - 252.60 cm	15.91	20.10	04.19

Tabela 4.5: Comparação de Velocidades nas Pistas percorridas

Numa apreciação final de todo este estudo, foram feitos os cálculos para, em cada pista e em cada modo de atuar o carro, retirar o valor da velocidade, com vista a tirar ilações mais detalhadas sobre os resultados obtidos.

Com isto, é possível destacar os seguintes pontos:

- Observa-se que, de um ponto de vista geral, como era evidente, os valores das velocidades no modo Manual acabam por ser bastante superiores aos valores em modo Automático;
- Dentro daquilo que é a comparação direta em cada pista, observamos que a diferença de velocidades nas pistas B e D (pistas com mais curvas) são bastante menores que nas restantes;
- Através de uma comparação direta entre as pistas com curvas e as pistas apenas com retas, observamos através da velocidade em modo Automático que o sensor de linha reagiu melhor às que não têm curvas, que são também aquelas que têm uma dimensão mais pequena;
- Contrariamente, no modo Manual, é visível que as pistas com curvas e, também tendo em conta que são pistas maiores, fizeram com que a velocidade nessas fosse maior.

Capítulo 5

Conclusões e Perspetivas Futuras

Ao longo deste documento, foi apresentado e descrito todo o trabalho realizado no âmbito desta dissertação. Além do levantamento bibliográfico realizado, foi apresentado um relatório detalhado da construção/projeção de um robô autónomo e controlável.

5.1 Conclusão

A concretização da dissertação foi um projeto especialmente enriquecedor e uma mais-valia para o presente e futuro. A dedicação, o tempo, as técnicas e as metodologias de pesquisa envolvidas na recolha da informação relevante, permitiram a aquisição e aprofundamento de conhecimentos em programação no domínio Arduíno, com montagem de circuitos eletrónicos. Possibilitou conhecer e diferenciar a importância e as diferentes utilidades no contexto atual, ao nível de sensores e microcontroladores, organizações que investem neste tipo de projetos inovadores, a sua dimensão e o interesse no desenvolvimento tecnológico na sociedade.

O processo de seleção de material e a aplicação dos conhecimentos teóricos e práticos em programação Arduíno, proporcionaram a interiorização de conhecimentos e a aplicabilidade de técnicas, resultando numa visão mais objetiva e integrada sobre ferramentas, equipamentos e processos que envolvem a conceção de um carro-robô, especialmente quanto às suas vantagens, desvantagens e possíveis melhorias. Salientando o impacto que

este tipo de projeto inovador poderá assumir num contexto real de utilidade futura.

Da análise detalhada dos resultados obtidos, foi possível extrair várias conclusões relativamente ao "sensor de linha" no seu modo de utilização, comparativamente com um controlo remoto do carro-robô através de "joystick":

- O "sensor de linha" permite que o carro percorra o circuito delineado, sem movimentos demasiado bruscos e sem se afastar da trajetória prevista;

- Por vezes, quando o carro está em modo autónomo, não executa o movimento programado, pelo facto de não conseguir ler a linha corretamente. Esta circunstância está relacionada com a existência de maior ou menor luminosidade a que a mesma pista está exposta, verificando-se que num ambiente com maior claridade a linha é lida com facilidade e assertividade;

- Paralelamente, as cores das cartolinas onde as pistas foram feitas também tiveram influencia na leitura correta da linha. As linhas cujas pistas tinham uma tonalidade mais clara foram lidas de modo correto com maior facilidade;

- Com a utilização do "joystick", verificamos que o circuito é percorrido com maior rapidez, porém, contrariamente ao funcionamento com o "sensor de linha", constatamos que a probabilidade de nos afastarmos do circuito delineado é muito maior, propiciando a ocorrência de mais erros/desvios. Neste caso, o movimento do carro está sempre dependente do utilizador. Esta probabilidade de erro acaba por ser, evidentemente, maior;

- Constatamos ainda que, por vezes, as rodas do carro, autonomamente e não devido ao código implementado, diminuem o seu movimento e, por conseguinte, este demora mais tempo a realizar o percurso. Esta condição que ocorre, tanto a nível autónomo como manual, pode ser superada com simples impulso na parte traseira do caro-robô. Contudo, e conforme resulta da pesquisa realizada, este acontecimento é comum em projetos similares estudados e apresentados.

O acima referido, permite-nos concluir que o "sensor de linha" é um componente essencial e que, nas condições funcionais, apresenta uma margem de erro reduzida, percorrendo o seu trajeto de forma exata, sem desvios nem desníveis.

No desenvolvimento deste projeto, também nos deparamos com alguns constrangimentos:

- O módulo LCD, inicialmente, apresentava alguns erros de compreensão, que foram sendo solucionados com o fator tempo e pesquisa;
- O módulo NRF24L01, sobretudo o implementado no carro, apresentava bastante sensibilidade, verificando-se que várias vezes não comunicava adequadamente, não recebendo os valores corretos, apesar destes dados serem sempre enviados corretamente.
- Quando se ligava o Arduíno presente no carro ao computador, recebia sempre os valores corretos, contudo quando se executava ON no carro, aconteceu muitas vezes não receber corretamente a informação;
- A existência de delays para a comunicação e a melhor seleção das portas para os módulos (normalmente geravam conflitos);
- O sensor de linha, devido à presença do módulo NRF24L01, não conseguia fazer a sua calibração corretamente e, devido a isso, quando o modo automático era o selecionado, a linha não era lida corretamente.

Estas limitações foram sendo superadas através de pesquisas, na tentativa de melhoria de algumas ferramentas a que se conseguiu aceder, processo esse que também propiciou conhecimento e operacionalidade funcional.

Pelo exposto, acreditamos que não obstante a sua complexidade e dificuldade de execução, o projeto foi bem-sucedido. A condição de trabalhador-estudante a exercer funções no setor da programação, também na área eletrónica, facilitou a superação de dificuldades e resolução prática de problemas, nomeadamente a nível de ligações geradoras de conflito,

mas também para uma melhoria constante do código com vista a uma solução final mais próxima do ideal.

5.2 Perspetiva Futura

É presentemente possível ter uma perspetiva dos próximos passos que um projeto desta dimensão e natureza poderia perseguir, de modo a que a sua implementação e funcionamento possa ser mais benéfico e intuitivo para futuros utilizadores. Os próximos avanços passariam pela implementação de sensores (por exemplo infravermelhos) nos locais de partida e chegada do circuito que o carro teria que percorrer. Deste modo, automático, o cronómetro iniciaria e parava quando o carro atravessasse estes sensores, sendo assim possível fazer uma análise mais circunstanciada e intuitiva das variáveis possíveis e do comportamento, tanto a nível do sensor de linha como do joystick.

Acreditamos que a implementação deste tipo de componentes/sensores, seria um valor acrescido neste tipo de projeto, se estes valores, depois de captados fossem direcionados para uma folha (Excel por exemplo) e fossem aglomerados numa tabela, para que esse registo também pudesse ser feito de forma automática. Claro que, tendo em conta o tipo de projeto a que nos estamos a reportar, teriam necessariamente de existir 2 (duas) tabelas, uma para os valores recolhidos em modo automático e outra para o valores recolhidos em modo manual. Assim, seria possível retirar todas as ilações pretendidas de um modo mais percetível, desde médias até à própria comparação de tempos e velocidades, calculadas no seu todo diretamente nas células do Excel.

A criação de uma pista com uma maior complexidade, que levasse o sensor a percorrer esse circuito e, paralelamente, o utilizador também tivesse de fazer esse mesmo percurso, permitiria recolher valores e aferir melhor a efetividade do sensor.

Como referido anteriormente, também o processo de melhoramento de leitura e calibração do sensor, quando este está no mesmo circuito do módulo NRF24L01 e o modo automático é o selecionado, será um dos pontos de maior preponderância a ser aprimorado futuramente.

Em suma e numa perspetiva contemporânea e futura, acreditamos que este tipo de projetos serão de extrema utilidade, dado a forma realista, didática e intuitiva que permitirá conhecer e melhor entender todo o contexto e importância da Robótica Autónoma. O mundo progressista caminhará para este tipo de abordagem e conceções que, em nosso entender, permitirão o conhecimento específico e amplo do fenómeno da robotização, vantagens e desvantagens, que vão sendo diferenciadas com a evolução social e tecnológica. Estes mecanismos e tudo que envolve autonomia, estará necessariamente ligado a sensores e microcontroladores, elementos essenciais e preponderantes a curto, médio e longo prazo, num panorama global em que os avanços na inovação tecnológica e robotização serão centrais e envolventes em tudo que nos rodeia numa sociedade dinâmica.

Bibliografia

[1] F. S. Martın, F. M. Rodrıguez, J. S. Bayarri, J. P. Redorta, F. R. Escovar, S. E. Fernández e H. V. Mavrich, "Historia de la robótica: de Arquitas de Tarento al robot Da Vinci (Parte I)," *Actas Urológicas Españolas*, vol. 31, n.º 2, pp. 69–76, 2007.

[2] *Nikola Tesla: Father of Robotics*, `https://www.teslasociety.com/robotics.htm`, Accessed: 2020-03-04.

[3] S. d. R. Zilli, "A Robótica Educacional no Ensino Fundamental: Perspectivas e Práticas," p. 89, 2004. URL: `https://repositorio.ufsc.br/xmlui/handle/123456789/86930`.

[4] F. B. V. Benitti, A. Vahldick, D. L. Urban, M. L. Krueger e A. Halma, "Experimentação com Robótica Educativa no Ensino Médio: ambiente, atividades e resultados," em *Anais do Workshop de Informática na Escola*, vol. 1, 2009, pp. 1811–1820.

[5] D. A. Medeiros Filho e P. C. Gonçalves, "Robótica educacional de baixo custo: Uma realidade para as escolas brasileiras," em *Anais do Workshop de Informática na Escola*, vol. 1, 2008.

[6] E. de Nez, A. M. da Silva e E. M. da Silva, "TRANSDISCIPLINARIDADE ATRAVÉS DA ROBÓTICA: Um relato de experiência na Escola Pública do Estado de Mato Grosso," em *Anais do Workshop de Informática na Escola*, vol. 1, 2010, pp. 1433–1436.

[7] P. C. Ribeiro, C. B. Martins e F. C. Bernardini, "A Robótica como Ferramenta de Apoio ao Ensino de Disciplinas de Programação em Cursos de Computação e Engenharia," em *Anais do Workshop de Informática na Escola*, vol. 1, 2011, pp. 1108–1117.

[8] I. C. A. Fazenda, *Interdisciplinaridade: história, teoria e pesquisa*. Papirus editora, 1994.

[9] R. Pitta, S. Thomaz, A. Aglaé, S. Azevedo, A. Burlamaqui e L. M. Gonçalves, "Roboeduc: Um software para programação em nıveis," em *Anais do Workshop de Informática na Escola*, vol. 1, 2010, pp. 1425–1428.

[10] A. L. Guedes, F. L. Guedes e T. B. Castro, "Perspectivas do uso da robótica educativa na educação infantil e no ensino fundamental," em *Anais do Workshop de Informática na Escola*, vol. 1, 2013, p. 410.

[11] J. V. V. d'Abreu e B. L. Bastos, "Robótica pedagógica: Uma reflexao sobre a apropriaçao de professores da escola elza maria pellegrini de aguiar," em *Anais do Workshop de Informática na Escola*, vol. 1, 2013, p. 280.

[12] *Ctrl+Play - Escola de Programação e Robótica para crianças e jovens de todas as idades.* `https://www.ctrlplay.com.br/o-que-e-steam-conheca-essa-metodologia-de-ensino-revolucionaria/`, Accessed: 2020-04-30.

[13] *Novos Alunos*, `https://novosalunos.com.br/steam/`, Accessed: 2020-04-30.

[14] *braganca*, `https://braganca.cienciaviva.pt/2233/historia-e-memoria`, Accessed: 2020-04-30.

[15] A. F. da Silva, P. R. D. E. P. Ós, R. Em, E. N. E. Létrica, A. Ferreira e A. F. da Silva, "RoboEduc: Uma Metodologia de Aprendizado com Robótica Educacional," *Tese de Doutorado*, p. 133, 2009. URL: `http://bdtd.bczm.ufrn.br/tedesimplificado/tde%7B%5C_%7Dbusca/processaArquivo.php?codArquivo=2427%7B%5C%%7D5Cnpapers2://publication/uuid/3E1438B1-8563-4624-AA9A-6C185A76F649`.

[16] G. Dudek e M. Jenkin, *Computational Principles of Mobile Robotics*. 2010, ISBN: 9780521692120. DOI: 10.1017/cbo9780511780929.

[17] Y. Ichikawa e N. Ozaki, *Autonomous Mobile Robot.* 1. 1985, vol. 2, pp. 135–144, ISBN: 026219502X.

[18] S. Thrun, "Probabilistic robotics," *Communications of the ACM*, vol. 45, n.º 3, pp. 52–57, 2002, ISSN: 00010782. DOI: 10.1145/504729.504754.

[19] D. d. C. Moreira e D. C. C. K. Kowaltowski, "Discussão sobre a importância do programa de necessidades no processo de projeto em arquitetura," *Ambiente Construído*, n.º 19, pp. 31–45, 2009. URL: http://www.seer.ufrgs.br/ambienteconstruido/article/download/7381/5484.

[20] C. Lin, C. Standing e Y. C. Liu, "A model to develop effective virtual teams," *Decision Support Systems*, vol. 45, n.º 4, pp. 1031–1045, 2008, ISSN: 01679236. DOI: 10.1016/j.dss.2008.04.002.

[21] R. Madhavan, E. Tunstel e E. Messina, *Performance evaluation and benchmarking of intelligent systems*. Springer, 2009.

[22] S. L. Murphy, J. C. Robinson e S. H. Lin, "Conducting Systematic reviews to inform occupational therapy practice," *American Journal of Occupational Therapy*, vol. 63, n.º 3, pp. 363–368, 2009, ISSN: 02729490. DOI: 10.5014/ajot.63.3.363.

[23] V. Sukumaran, Q. Chen, F. Liu, N. Kumbhat, T. Bandyopadhyay, H. Chan, S. Min, C. Nopper, V. Sundaram e R. Tummala, "Through-package-via formation and metallization of glass interposers," em *2010 Proceedings 60th Electronic Components and Technology Conference (ECTC)*, IEEE, 2010, pp. 557–563.

[24] N. T. Son, C. Chen, C. Chen, L. Chang e V. Q. Minh, "Monitoring agricultural drought in the Lower Mekong Basin using MODIS NDVI and land surface temperature data," *International Journal of Applied Earth Observation and Geoinformation*, vol. 18, pp. 417–427, 2012.

[25] S. Thrun, M. Montemerlo, H. Dahlkamp, D. Stavens, A. Aron, J. Diebel, P. Fong, J. Gale, M. Halpenny, G. Hoffmann et al., "Stanley: The robot that won the DARPA Grand Challenge," *Journal of field Robotics*, vol. 23, n.º 9, pp. 661–692, 2006.

[26] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer et al., "Autonomous driving in urban environments: Boss and the urban challenge," *Journal of Field Robotics*, vol. 25, n.º 8, pp. 425–466, 2008.

[27] P. Furgale, U. Schwesinger, M. Rufli, W. Derendarz, H. Grimmett, P. Mühlfellner, S. Wonneberger, J. Timpner, S. Rottmann, B. Li et al., "Toward automated driving in cities using close-to-market sensors: An overview of the v-charge project," em *2013 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2013, pp. 809–816.

[28] L. Paull, G. Huang e J. J. Leonard, "A unified resource-constrained framework for graph SLAM," em *2016 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2016, pp. 1346–1353.

[29] A. Censi, "A mathematical theory of co-design," *arXiv preprint arXiv:1512.08055*, 2015.

[30] D. Hoehener, G. Huang e D. Del Vecchio, "Design of a lane departure driver-assist system under safety specifications," em *2016 IEEE 55th Conference on Decision and Control (CDC)*, IEEE, 2016, pp. 2468–2474.

[31] J. Tani, L. Paull, M. T. Zuber, D. Rus, J. How, J. Leonard e A. Censi, "Duckietown: an innovative way to teach autonomy," em *International Conference EduRobotics 2016*, Springer, 2016, pp. 104–121.

[32] *Duckietown Foundation*, `https://www.duckietown.org/`, Accessed: 2020-05-30.

[33] *Sociedade Portuguesa de Robótica*, `http://www.sprobotica.pt`, Accessed: 2020-10-27.

[34] *Festival Nacional de Robótica*, http://www.sprobotica.pt/index.php?option=com_content&view=article&id=108&Itemid=62&fbclid=IwAR2ppjhCXa_gIk-9eEq_KPzIGSijz_F2SHEH8x7im4FlK1hwARa2jNjUG8E, Accessed: 2020-10-27.

[35] *Programa do Festival Nacional de Robótica 2019*, https://web.fe.up.pt/~robotica2019/index.php/pt/, Accessed: 2020-10-27.

[36] *Condução Autónoma*, https://web.fe.up.pt/~robotica2019/index.php/pt/conducao-autonoma, Accessed: 2020-10-27.

[37] *BOT'N ROLL ONE A*, https://www.roboparty.org/RPLGW/BrnOneA.php, Accessed: 2020-10-27.

[38] *ROBOPARTY – O QUE É?* https://www.roboparty.org/RPGuimaraes/abstract.php, Accessed: 2020-10-27.

[39] *Robotex history*, https://robotex.international/robotex-organization/, Accessed: 2020-10-27.

[40] *Robotex Competitions*, https://robotex.international/roboticscompetitions/, Accessed: 2020-10-27.

[41] K. Jo, J. Kim, D. Kim, C. Jang e M. Sunwoo, "Development of autonomous car—Part I: Distributed system architecture and development process," *IEEE Transactions on Industrial Electronics*, vol. 61, n.º 12, pp. 7131–7140, 2014.

[42] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt et al., "Towards fully autonomous driving: Systems and algorithms," em *2011 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2011, pp. 163–168.

[43] R. Zakharenko, "Self-driving cars will change cities," *Regional Science and Urban Economics*, vol. 61, pp. 26–37, 2016.

[44] *Car Autonomy Levels Explained*, https://www.thedrive.com/article/15724/what-are-these-levels-of-autonomy-anyway, Accessed: 2020-05-25.

[45] A. Živković, "Development of Autonomous Driving using ROS,"

[46] M. Killpack, T. Deyle, C. Anderson e C. C. Kemp, "Visual odometry and control for an omnidirectional mobile robot with a downward-facing camera," em *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, 2010, pp. 139–146.

[47] J. Chung, "Mind, Machine, and Society: Legal and Ethical Implications of Self-Driving Cars," tese de doutoramento, 2018.

[48] B. Sudlow, "Review of Joseph E. Aoun (2017). Robot Proof: Higher Education in the Age of Artificial Intelligence," *Postdigital Science and Education*, vol. 1, n.º 1, pp. 236–239, 2019.

[49] *What is EEPROM Memory Technology*, https://www.electronics-notes.com/articles/electronic_components/semiconductor-ic-memory/eeprom-e2prom-technology.php, Accessed: 2020-06-23.

[50] *A História dos Microcontroladores*, http://oincrivelmundonerd.blogspot.com/2014/03/a-historia-dos-microcontroladores.html, Accessed: 2020-06-23.

[51] *Por que utilizar microcontroladores*, https://www.palpitedigital.com.br/wp/2007/05/12/porque-utilizar-micrcontroladores/, Accessed: 2020-06-25.

[52] *O que são sensores e quais as suas aplicações?* https://www.mundodaeletrica.com.br/o-que-sao-sensores-e-quais-as-suas-aplicacoes/, Accessed: 2020-06-27.

[53] *O que são sensores e quais as suas aplicações?Qual é a diferença entre sensores analógicos e digitais (uma explicação em palavras simples)?* https://svcministry.org/pt/dictionary/what-is-the-difference-between-analog-and-digital-sensors-an-explanation-in-simple-words/, Accessed: 2020-06-27.

[54] *Electrical Sensors: Maxim Integrated*, https://www.maximintegrated.com/en/products/sensors.html, Accessed: 2020-06-27.

[55] *What's a Magnetic Sensor?* https://www.akm.com/global/en/technology/technical-tutorial/basic-knowledge-magnetic-sensor/magnetic-sensor/, Accessed: 2020-06-27.

[56] *Purpose and Working Principle of Inductive Sensors*, https://electrical-engineering-portal.com/purpose-and-working-principle-of-inductive-sensors, Accessed: 2020-06-27.

[57] *What is a Capacitive Sensor?* https://automation-insights.blog/2017/06/07/what-is-a-capacitive-sensor/, Accessed: 2020-06-27.

[58] S. Beeby, G. Ensel, N. M. White e M. Kraft, *MEMS mechanical sensors*. Artech House, 2004.

[59] *Thermal Sensor*, https://www.sciencedirect.com/topics/materials-science/thermal-sensor, Accessed: 2020-06-27.

[60] *Acoustic sensor*, http://wikid.io.tudelft.nl/WikID/index.php/Acoustic_sensor, Accessed: 2020-06-27.

[61] *What is a Fiber Optic Sensor?* https://www.keyence.com/ss/products/sensor/sensorbasics/fiber/info/, Accessed: 2020-06-27.

[62] *What is an Ultrasonic Sensor?* https://www.fierceelectronics.com/sensors/what-ultrasonic-sensor, Accessed: 2020-06-27.

[63] *Pressure Sensors*, https://pt.farnell.com/sensor-pressure-sensors-technology, Accessed: 2020-06-30.

[64] *What Is An Image Sensor?* https://www.visiononline.org/blog-article.cfm/What-Is-An-Image-Sensor/32, Accessed: 2020-07-10.

[65] *Photoelectric Sensors*, http://www.ia.omron.com/support/guide/43/introduction.html, Accessed: 2020-06-27.

[66] *O que é Arduino: conceito, benefícios e como utilizar*, https://www.filipeflop.com/blog/o-que-e-arduino/, Accessed: 2020-04-01.

[67] *Arduíno UNO R3 detalhes técnico*, http://suadica.com/dica.php?d=373, Accessed: 2020-04-01.

[68] *Arduino UNO - Conheça o hardware da placa Arduino em detalhes*, https://www.embarcados.com.br/arduino-uno/, Accessed: 2020-04-01.

[69] *Alphabot*, https://www.waveshare.com/wiki/AlphaBot, Accessed: 2020-04-30.

[70] *Alphabot, THE OPEN SOURCE ROBOT*, https://www.open-electronics.org/alphabot-the-open-source-robot/, Accessed: 2020-04-30.

[71] *Módulo Wireless NRF24L0*, https://www.curtocircuito.com.br/modulo-wireless-2-4g-nrf24l01.html, Accessed: 2020-05-10.

[72] *Módulo nRF24L01*, http://mundoprojetado.com.br/modulo-nrf24l01/, Accessed: 2020-05-10.

[73] *Display I2C LCD1602 16x2 c/ Driver - Azul*, https://www.botnroll.com/pt/lcds-e-displays/2989-display-i2c-lcd1602-16x2-c-driver-p-funduino-azul.html, Accessed: 2020-05-10.

[74] *Como usar com Arduino - Módulo Joystick KY-023*, https://blogmasterwalkershop.com.br/arduino/como-usar-com-arduino-modulo-joystick-ky-023/, Accessed: 2020-05-12.

[75] *About Version of RadioHead library for Teensy boards*, https://github.com/PaulStoffregen/RadioHead, Accessed: 2020-05-12.

[76] *SPI Library*, https://www.pjrc.com/teensy/td_libs_SPI.html, Accessed: 2020-05-12.

[77] *Memória EEPROM*, ://pt.slideshare.net/mariokleber31/ee-prom.

[78] *Tracker Sensor*, https://www.waveshare.com/wiki/Tracker_Sensor, Accessed: 2020-05-10.

[79] *Wire library used on Teensy boards*, https://github.com/PaulStoffregen/Wire, Accessed: 2020-05-12.

[80] *Adafruit GFX graphics core library, this is the 'core' class that all our other graphics libraries derive from*, https://github.com/adafruit/Adafruit-GFX-Library, Accessed: 2020-05-12.

[81] *Library for the LiquidCrystal LCD display connected to an Arduino board*, https://github.com/fdebrabander/Arduino-LiquidCrystal-I2C-library, Accessed: 2020-05-12.

[82] *Debouncing library for Arduino or Wiring*, https://github.com/heman4t/Arduino-Bounce2, Accessed: 2020-05-12.

[83] A. U. Manual, "AlphaBot User Manual," pp. 1–54, 2017.

[84] Dejan, "Arduino Wireless Communication – NRF24L01 Tutorial," *HowToMechatronics*, 2017. URL: https://howtomechatronics.com/tutorials/arduino/arduino-wireless-communication-nrf24l01-tutorial/.

[85] *map()*, https://www.arduino.cc/reference/en/language/functions/math/map/, Accessed: 2020-09-12.

[86] *Arduino – Função map() – Mapeando intervalos de valores numéricos*, http://www.bosontreinamentos.com.br/eletronica/arduino/funcao-map/, Accessed: 2020-10-13.

Apêndice A

Código do Carro

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
#include "TRSensors.h"

#define SENSORS_NUMBER 5

// Motor A
int enA = 5;
int in1 = A1;
int in2 = A0;

// Motor B
int enB = 6;
int in3 = A2;
int in4 = A3;


TRSensors trs =   TRSensors();
unsigned int sensorValues[SENSORS_NUMBER];
unsigned int last_proportional = 0;
long integral = 0;

RF24 radio(7,8); // CE, CSN
const byte address[6] = "00001";
char receivedData[32] = "";
int  xAxis, yAxis;
int motorSpeedA = 0;
int motorSpeedB = 0;
int joystick[2];


const int LEDAuto = 10;
const int LEDManual = 9;

void setup() {
  pinMode(enA, OUTPUT);
  pinMode(enB, OUTPUT);
  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);
  pinMode(LEDAuto, OUTPUT);
  pinMode(LEDManual, OUTPUT);
```

```
  Serial.begin(9600);
  radio.begin();

  radio.openReadingPipe(0, address);
  radio.setPALevel(RF24_PA_MAX);
  radio.startListening();

  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
  analogWrite(enA,0);
  analogWrite(enB,0);

  for (int i = 0; i < 400; i++)               // make the calibration take about 10 seconds
  {
    trs.calibrate();
  }
  Serial.println("calibrate done");

  // print the calibration minimum values measured when emitters were on
  for (int i = 0; i < SENSORS_NUMBER; i++)
  {
    Serial.print(trs.calibratedMin[i]);
    Serial.print(' ');

  }
  Serial.println();

  // print the calibration maximum values measured when emitters were on
  for (int i = 0; i < SENSORS_NUMBER; i++)
  {
    Serial.print(trs.calibratedMax[i]);
    Serial.print(' ');
  }
  Serial.println();
  delay(1000);
}
void loop() {

  if (radio.available())
  {
    radio.read( joystick, sizeof(joystick) );
    radio.read(&receivedData, sizeof(receivedData));
```

```
  yAxis = joystick[0];
  xAxis = joystick[1];

  Serial.print("Motor Y: ");
  Serial.print(yAxis);
  Serial.print(" - Motor X: ");
  Serial.print(xAxis);
  Serial.print("\n");

  delay (100);
 }

 if ((yAxis < -10) && (xAxis < -10))
 {
  digitalWrite(LEDAuto, HIGH);
  digitalWrite(LEDManual, LOW);
  Serial.print("\n AUTO MODE \n");

  // Get the position of the line.
  unsigned int position = trs.readLine(sensorValues);
  for (unsigned char i = 0; i < SENSORS_NUMBER; i++)
  {
   Serial.print(sensorValues[i]);
   Serial.print('\t');
  }
  Serial.println(position);

  int proportional = (int)position - 2000;
  // improve performance.
  int power_difference = proportional/15;

  // Compute the actual motor settings
  const int maximum =100;

  if (power_difference > maximum)
   power_difference = maximum;
  if (power_difference < - maximum)
   power_difference = - maximum;

  Serial.println(power_difference);

  if (power_difference < 0)
  {
   analogWrite(enB,maximum + power_difference);
   analogWrite(enA,maximum);
```

```
  }
  else
  {
    analogWrite(enB,maximum);
    analogWrite(enA,maximum - power_difference);
  }
}
else
{
  digitalWrite(LEDAuto, LOW);
  digitalWrite(LEDManual, HIGH);
  Serial.print("\n MANUAL MODE \n");

  if ((yAxis < 150) && (yAxis >= 0))
  {
    digitalWrite(in1, HIGH);
    digitalWrite(in2, LOW);
    digitalWrite(in3, HIGH);
    digitalWrite(in4, LOW);

    motorSpeedA = map(yAxis, 150, 0, 0, 255);
    motorSpeedB = map(yAxis, 150, 0, 0, 255);
  }
  else if (yAxis > 564)
  {
    digitalWrite(in1, LOW);
    digitalWrite(in2, HIGH);
    digitalWrite(in3, LOW);
    digitalWrite(in4, HIGH);

    motorSpeedA = map(yAxis, 564, 1023, 0, 255);
    motorSpeedB = map(yAxis, 564, 1023, 0, 255);
  }

  else {
    motorSpeedA = 0;
    motorSpeedB = 0;
  }

  if ((xAxis < 300) && (xAxis >= 0))
  {
    int xMapped = map(xAxis, 300, 0, 0, 255);

    motorSpeedA = motorSpeedA + xMapped;
    motorSpeedB = motorSpeedB - xMapped;
```

```
    // Confine the range from 0 to 255
    if (motorSpeedA < 0) {
      motorSpeedA = 0;
    }
    if (motorSpeedB > 255) {
      motorSpeedB = 255;
    }
   }
   else if (xAxis > 564)
   {
    int xMapped = map(xAxis, 564, 1023, 0, 255);

    motorSpeedA = motorSpeedA - xMapped;
    motorSpeedB = motorSpeedB + xMapped;

    if (motorSpeedA > 255) {
      motorSpeedA = 255;
    }
    if (motorSpeedB < 0) {
      motorSpeedB = 0;
    }
   }

   if (motorSpeedA < 150) {
    motorSpeedA = 0;
   }
   if (motorSpeedB < 150) {
    motorSpeedB = 0;
   }
   analogWrite(enA, motorSpeedA); // Send PWM signal to motor A
   analogWrite(enB, motorSpeedB); // Send PWM signal to motor B
  }
}
```

Apêndice B

Código do Comando

```
#include <SPI.h>
#include <nRF24L01.h>
#include <RF24.h>
RF24 radio(8,10); // CE, CSN
const byte address[6] = "00001";

char xyData[32] = "";
int joystick[2];


const byte controlButton = A5;
const int LEDAuto = A3;
const int LEDManual = A4;

int BUTTONControlState = 0;

void setup()
{
 Serial.begin(9600);
 radio.begin();
 radio.openWritingPipe(address);
 radio.setPALevel(RF24_PA_MAX);
 radio.stopListening();

 pinMode(controlButton, INPUT_PULLUP);
 pinMode(LEDAuto, OUTPUT);
 pinMode(LEDManual, OUTPUT);
}

bool controlState = LOW;

void loop()
{

 BUTTONControlState = digitalRead(controlButton);

 if (BUTTONControlState == HIGH )
 {
  digitalWrite(LEDAuto, HIGH);
  digitalWrite(LEDManual, LOW);

  joystick[0] = -20;
  joystick[1] = -20;
  radio.write( joystick, sizeof(joystick) );
```

```
  Serial.print("\n AUTO MODE \n");
 }
 else
 {
  digitalWrite(LEDAuto, LOW);
  digitalWrite(LEDManual, HIGH);

  joystick[0] = analogRead(A0);
  joystick[1] = analogRead(A1);
  radio.write( joystick, sizeof(joystick) );

  Serial.print("\n MANUAL MODE \n");
  Serial.print("\n Motor Y: ");
  Serial.print(joystick[0]);
  Serial.print(" - Motor X: ");
  Serial.print(joystick[1]);
  Serial.print("\n");
 }
}
```

Apêndice C

Código do Cronómetro

```
#include <Wire.h> // Library for I2C communication
#include <LiquidCrystal_I2C.h> // Library for LCD
#include <Bounce2.h>

LiquidCrystal_I2C lcd = LiquidCrystal_I2C(0x27, 20, 4);

const byte switchButton = 9;

//right button
const byte startButton = A0;
// Instantiate a Bounce object
Bounce startDebouncer1 = Bounce();


//center button
const byte resetButton = A1;
// Instantiate another Bounce object
Bounce resetDebouncer2 = Bounce();

 const int LEDStart = A2;
 const int LEDReset = A3;

 int BUTTONStartState = 0;
 int BUTTONResetState = 0;


void setup() {
  pinMode(LEDStart, OUTPUT);
  pinMode(LEDReset, OUTPUT);


  pinMode(startButton, INPUT_PULLUP);
  // After setting up the button, setup the Bounce instance :
  startDebouncer1.attach(startButton);
  startDebouncer1.interval(5); // interval in ms

  pinMode(resetButton, INPUT_PULLUP);
  // After setting up the button, setup the Bounce instance :
  resetDebouncer2.attach(resetButton);
  resetDebouncer2.interval(5); // interval in ms
  lcd.init();
  lcd.backlight();

  // set up the LCD's number of columns and rows:
  lcd.begin(16, 2);
```

```
  lcd.setCursor(0, 0);
  // Print a message to the LCD.
  lcd.print("*** StopWatch **");
  lcd.setCursor(0, 1);
  lcd.print("***  Project ***");

  delay(100);

  // clear screen for the next loop:
  lcd.clear();

  lcd.setCursor(0, 0);
  lcd.print("Press B-Strt/Stp");

  lcd.setCursor(0, 1);
  lcd.print("Press R-Reset");

  Serial.begin(9600);

  Serial.println("********************************** ");
  Serial.println("CRONOMETER ");
  Serial.println("Press Following Switches: ");
  Serial.println("Switch 1 for Start/Stop");
  Serial.println("Switch 2 for Reset");
  Serial.println("********************************** ");


}

bool startState = LOW;
bool resetState = LOW;

unsigned long startMillis;
unsigned long currentMillis;
unsigned long elapsedMillis;

void loop() {

  BUTTONStartState = digitalRead(startButton);
  BUTTONResetState = digitalRead(resetButton);


  if (BUTTONStartState == HIGH)
  {
    digitalWrite(LEDStart, HIGH);
```

```
}
else{
  digitalWrite(LEDStart, LOW);
}

if (BUTTONResetState == HIGH)
{
  digitalWrite(LEDReset, HIGH);
}
else{
  digitalWrite(LEDReset, LOW);
}
// Update the Bounce instances :
startDebouncer1.update();

 if ( startDebouncer1.fell() ) {    // Call code if button transitions from HIGH to LOW
   startState = !startState;        // Toggle start button state
   startMillis = millis();
 }

if (startState)
{
  currentMillis = millis();
  elapsedMillis = (currentMillis - startMillis);

  lcd.setCursor(0, 0);
  lcd.print("SW (hh:mm:ss:ms)");
  lcd.setCursor(0, 1);
  lcd.print("              ");

  Serial.print("elapsedMillis: ");
  Serial.print(elapsedMillis);
  Serial.println("");

  unsigned long durMS = (elapsedMillis%1000);      //Milliseconds
  unsigned long durSS = (elapsedMillis/1000)%60;   //Seconds
  unsigned long durMM = (elapsedMillis/(60000))%60; //Minutes
  unsigned long durHH = (elapsedMillis/(3600000));  //Hours
  durHH = durHH % 24;

  Serial.print("Time: ");
  Serial.print(durHH);
  Serial.print(" : ");
  Serial.print(durMM);
  Serial.print(" : ");
```

```
  Serial.print(durSS);
  Serial.print(" : ");
  Serial.print(durMS);
  Serial.println("");

  String durMilliSec = timeMillis(durHH, durMM, durSS,durMS);
  lcd.setCursor(0, 1);
  lcd.print(durMilliSec);

  Serial.println("*****************************");
  Serial.println("");

  delay(150);
 }

 resetDebouncer2.update();

 if (resetDebouncer2.fell())
 {
  resetState = HIGH;
 }

 if (resetState)
 {
  // clear screen for the next loop:
  lcd.clear();

  lcd.setCursor(0, 0);
  lcd.print("Press B-Strt/Stp");

  lcd.setCursor(0, 1);
  lcd.print("Press R-Reset");

  Serial.println("StopWatch using Arduino Nano");
  Serial.println("Press Following Switches:- ");
  Serial.println("Black Switch for Start/Stop");
  Serial.println("Red Switch for Reset");
  Serial.println("");

  delay(100);

  resetState = LOW;
 }

}
```

```
String timeMillis(unsigned long Hourtime,unsigned long Mintime,unsigned long
Sectime,unsigned long MStime)
{
 String dataTemp = "";

 if (Hourtime < 10)
 {
  dataTemp = dataTemp + "0" + String(Hourtime)+ "h:";
 }
 else{
  dataTemp = dataTemp + String(Hourtime)+ "h:";
 }

 if (Mintime < 10)
 {
  dataTemp = dataTemp + "0" + String(Mintime)+ "m:";
 }
 else{
  dataTemp = dataTemp + String(Mintime)+ "m:";
 }

 if (Sectime < 10)
 {
  dataTemp = dataTemp + "0" + String(Sectime)+ "s:";
 }
 else{
  dataTemp = dataTemp + String(Sectime)+ "s:";
 }

 dataTemp = dataTemp + String(MStime);

 //Serial.print("String Time: ");
 //Serial.println(dataTemp);
 lcd.print(dataTemp);

 return dataTemp;
}
```

Apêndice D

Código do Carro Autónomo

```
#include "TRSensors.h"

#define NUM_SENSORS 5

// sensors 0 through 5 are connected to analog inputs 0 through 5, respectively
TRSensors trs =   TRSensors();
unsigned int sensorValues[NUM_SENSORS];
unsigned int last_proportional = 0;
long integral = 0;

// Motor A
int ENA = 5;
int IN1 = A1;
int IN2 = A0;

// Motor B
int ENB = 6;
int IN3 = A2;
int IN4 = A3;

void setup()
{
 Serial.begin(9600);
 Serial.println("TRSensor example");
 pinMode(ENA, OUTPUT);
 pinMode(ENB, OUTPUT);
 pinMode(IN1, OUTPUT);
 pinMode(IN2, OUTPUT);
 pinMode(IN3, OUTPUT);
 pinMode(IN4, OUTPUT);
 digitalWrite(IN1, LOW);
 digitalWrite(IN2, LOW);
 digitalWrite(IN3, HIGH);
 digitalWrite(IN4, LOW);
 analogWrite(ENA,0);
 analogWrite(ENB,0);

 for (int i = 0; i < 400; i++)              // make the calibration take about 10 seconds
 {
  trs.calibrate();
 }
 Serial.println("calibrate done");

 // print the calibration minimum values measured when emitters were on
 for (int i = 0; i < NUM_SENSORS; i++)
```

```
 {
  Serial.print(trs.calibratedMin[i]);
  Serial.print(' ');
 }
 Serial.println();

 // print the calibration maximum values measured when emitters were on
 for (int i = 0; i < NUM_SENSORS; i++)
 {
  Serial.print(trs.calibratedMax[i]);
  Serial.print(' ');
 }
 Serial.println();



 delay(1000);
}


void loop()
{
 // Get the position of the line.  s.
 unsigned int position = trs.readLine(sensorValues);
   for (unsigned char i = 0; i < NUM_SENSORS; i++)
   {
    Serial.print(sensorValues[i]);
    Serial.print('\t');
   }
    Serial.print("Position : "); // comment this line out if you are using raw values

 // The "proportional" term should be 0 when we are on the line.
 int proportional = (int)position - 2000;

 int power_difference = proportional/15; //+derivative;

 // Compute the actual motor settings.
 const int maximum =100;

 if (power_difference > maximum)
   power_difference = maximum;
 if (power_difference < - maximum)
   power_difference = - maximum;

   Serial.println(power_difference);
```

```
  if (power_difference < 0)
  {
    analogWrite(ENB,maximum + power_difference);
    analogWrite(ENA,maximum);
  }
  else
  {
    analogWrite(ENB,maximum);
    analogWrite(ENA,maximum - power_difference);
  }

  delay(1000);
}
```