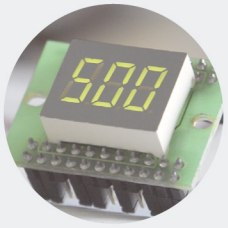
INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA ELECTROTÉCNICA E DE COMPUTADORES






Simulador 3D para Robótica Educativa

ANDRÉ FILIPE COSTA RIBEIRO
Setembro de 2024

POLITÉCNICO
DO PORTO

Politécnico do Porto
Instituto Superior de Engenharia do Porto

Simulador 3D para Robótica Educativa

André Filipe Costa Ribeiro

Mestrado em Engenharia Electrotécnica e de Computadores
Área de Especialização em Automação e Sistemas

Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto

Setembro, 2024

Esta dissertação satisfaz, parcialmente, os requisitos que constam da Ficha de Unidade Curricular de Tese/Dissertação, do 2º ano, do Mestrado em Engenharia Electrotécnica e de Computadores, Área de Especialização em Automação e Sistemas.

Candidato: André Filipe Costa Ribeiro, Nº 1150763, 1150763@isep.ipp.pt

Orientação Científica: André Dias, apd@isep.ipp.pt

Departamento de Engenharia Eletrotécnica
Instituto Superior de Engenharia do Porto
Rua Dr. António Bernardino de Almeida, 431, 4200–072 Porto

Setembro, 2024

Agradecimentos

Gostaria de expressar o meu sincero agradecimento a todas as pessoas que contribuíram para a realização desta dissertação. Em primeiro lugar, ao meu orientador, André Dias, por me apresentar este projeto e pela sua orientação constante na área da robótica educativa, sempre disponível para discutir ideias e ajudar na superação dos desafios encontrados ao longo do caminho. Agradeço também aos formadores da Academia de Robótica pela ajuda essencial na organização e condução das aulas, e aos alunos que participaram nas sessões de testes, cuja colaboração foi fundamental para a validação do trabalho desenvolvido. Aos meus pais, deixo um agradecimento especial pelo incentivo contínuo e pela motivação que me transmitiram, sem os quais a conclusão deste projeto não teria sido possível. Por fim, à Bernarda Santos, expresso a minha profunda gratidão por estar presente em todas as fases deste projeto, oferecendo-me o seu apoio incondicional e ajuda incansável.

Resumo

A implementação de conhecimentos de áreas como Ciência, Tecnologia, Engenharia e Matemática (STEM) no currículo educacional têm-se demonstrado relevantes na formação de jovens por todo o mundo para um mercado de trabalho cada vez mais complexo e competitivo na área das novas tecnologias. A robótica educativa surge como uma ferramenta STEM, pois permite que os alunos desenvolvam competências técnicas, *hard skills*, em programação, matemática, áreas da ciência e competências transversais, *soft skills* como a resolução de problemas. No entanto, os custos elevados associados à robótica ainda representam um obstáculo para a sua ampla adoção. Nesse sentido a dissertação endereça o desenvolvimento de um ambiente simulação 3D da prova First Challenger do Festival Nacional de Robótica no *software* Webots. Esta abordagem permite que os alunos desenvolvam e testem os seus códigos num ambiente simulado antes da aplicação no robô real, reduzindo os custos e os riscos de danificação do material. A solução foi testada em três aulas experimentais, demonstrando a viabilidade e eficácia do uso do robô virtual na robótica educativa. Os resultados obtidos permitiram aferir que o uso do robô simulado não só é intuitivo, como também facilita a implementação do código, otimizando o processo de ensino e consequentemente contribuindo para uma melhor aprendizagem.

Palavras-Chave: Robótica Educativa, STEM, Simulação, Webots, First Challenger

Abstract

The integration of knowledge from fields such as Science, Technology, Engineering, and Mathematics (STEM) in the educational curriculum has proven to be crucial in preparing young people around the world for an increasingly complex and competitive job market, especially in emerging technologies. Educational robotics serves as a STEM tool, enabling students to develop technical skills, such as programming and math, as well as scientific knowledge, alongside transversal competencies like problem-solving. However, the high costs associated with robotics still pose a barrier to its widespread adoption. To address this challenge, this dissertation focuses on the development of a 3D simulation environment for the First Challenger competition, part of the National Robotics Festival, using the Webots software. This approach allows students to develop and test their code in a simulated environment before applying it to a real robot, reducing both costs and the risk of equipment damage. The solution was tested in three experimental classes, demonstrating the feasibility and effectiveness of using a virtual robot in educational robotics. The results showed that the simulated robot is not only intuitive to use but also simplifies code implementation, optimizing the teaching process and thereby contributing to improved learning outcomes.

Keywords: Educational Robotics, STEM, Simulation, Webots, First Challenger

Índice

Lista de Figuras

Listagens

Capítulo 1

Introdução

Nas últimas décadas, o aumento exponencial da complexidade e a constante mudança no mundo motivou a necessidade de aplicar novas reformas no ensino. A implementação de áreas como Ciência, Tecnologia, Engenharia e Matemática, em inglês, Science, Technology, Engineering and Mathematics (STEM) [1], nas várias disciplinas do currículo atual na educação dos jovens é uma solução que traz vantagens fundamentais para o desenvolvimento de novos profissionais neste mundo altamente complexo e competitivo, onde as *soft skills*, assumem um papel preponderante.

Nesta nova década praticamente todos os novos trabalhos e os já existentes vão necessitar de conhecimentos nas áreas das STEM, por isso ao incorporá-las nas várias disciplinas não só torna o ensino mais intuitivo e motivadora para o aluno, como também desenvolve de forma quase impercetível estas competências mandatárias.

Umas das grandes dificuldades na aplicação desta nova metodologia de ensino em todas as escolas é o custo associado à aplicação das STEM, como tal, existe então uma necessidade de desenvolver produtos que podem ser rapidamente aplicados em contexto de sala de aula. Nesse sentido, a área da robótica poderá assumir um papel relevante, visto que a introdução da mesma na educação traz múltiplas vantagens, uma vez que um robô tem o poder de ser bastante versátil e pode ser aplicado para lecionar várias disciplinas. Além disso, no mercado, já existem vários robôs educativos que as escolas e os alunos podem adotar para aprenderem programação, matemática, ciências e outras disciplinas.

A Academia de Robótica [2] é um exemplo de um projeto educativo de robótica em Portugal que procura promover as áreas das STEM através do ensino da robótica. Este projeto promove a realização de desafios que através do método de tentativa e erro permite que o aluno aprenda a lidar com os vários problemas que vão surgindo no decorrer das atividades. Os alunos quando entram para o primeiro nível têm como tarefa aprender a programar em linguagem de alto nível Scratch [3], assim como efetuar a montagem de todos os componentes que compõem um robô. A finalidade é interligar estas duas componentes e obter um robô capaz de seguir uma linha preta que efetua uma paragem de emergência quando deteta um objeto. Já no segundo nível, o objetivo final é que os alunos participem no Festival Nacional de Robótica na competição First Challenger [4].

No decorrer das diversas aulas existe quase sempre perda de material o que eleva o custo da formação por parte da Academia. Como já referido anteriormente, o custo é um dos grandes problemas na adoção das STEM e o processo de montagem, bem como a validação de todos os sensores e atuadores do robô requerem que a equipa de professores tenha uma elevada experiência de robótica, o que poderá ser uma limitação se o objetivo for massificar a utilização desta área como uma ferramenta STEM.

Nesse sentido, a dissertação irá endereçar o desenvolvimento de uma ferramenta de simulação tridimensional que permita de uma forma mais controlada os alunos e os professoras possam desenvolver atividades STEM sem o esforço adicional de validação dos sensores e atuadores existentes numa plataforma robótica. Com esta abordagem não é pretendido a remoção de componentes de montagem do robô, mas sim alcançar um caminho que permita aos professores com menos conhecimentos da área avançarem com a aplicação de metodologias STEM nas escolas, ficando o *know-how* de *hardware* para uma segunda etapa.

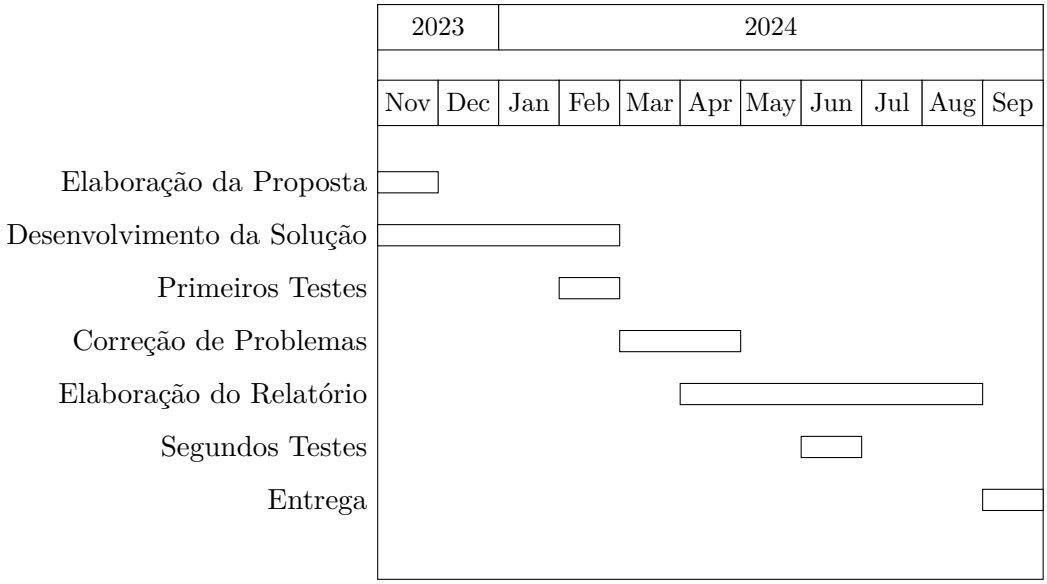
1.1 Objetivos

Para facilitar a integração das STEM na educação, minimizar os custos com a compra de plataforma robóticas e reduzir os recursos necessários para a introdução destas, é proposto o desenvolvimento da prova First Challenger num simulador. Assim, o aluno terá uma nova fase intitulada, fase de simulação, no desenvolvimento do seu robô, onde irá efetuar a programação e testá-la num ambiente virtual. Esta fase inicial, também poderá ser utilizada por novos alunos para ganhar motivação e vontade de participar nesta prova.

Assim os objetivos para este projeto são:

- Modelar as diferentes provas do First Challenger num simulador;
- Modelar um robô com os componentes (sensores, atuadores, etc.) necessários para a realização da prova;
- Desenvolver blocos de código/bibliotecas para o simulador idênticos ao do robô real;
- Efetuar a validação do simulador com um grupo heterogéneo de alunos.

1.2 Calendarização



1.3 Organização da Dissertação

Este documento está organizado em sete capítulos, começando pela "Introdução", que apresenta o tema e os objetivos. De seguida o capítulo do "Estado da Arte"que explora o panorama das STEM na educação focando-se na área da robótica e da simulação como ferramentas educativas. No capítulo de "Proposta de Solução"é descrita a solução encontrada para satisfazer os objetivos propostos, já o do "Desenvolvimento"descreve o procedimento para a modelação do robô virtual. O antepenúltimo capítulo "Resultados e Análise"apresenta os testes efetuados e resultados obtidos aos vários componentes do robô simulado, enquanto que o capítulo "Validação do Simulador em Contexto de Aula"cinge-se nos resultados obtidos da aplicação prática e validação em contexto de sala de aula. A Conclusão responde aos objetivos estabelecidos, expõe as limitações e apresenta trabalho futuro para o presente projeto.

Capítulo 2

Estado da Arte

No presente capítulo, vão ser explorados o uso de ferramentas STEM na educação, com ênfase na robótica como uma ferramenta pedagógica. Em seguida, será abordada a importância das simulações no contexto da robótica educativa, destacando a forma como estas auxiliam os alunos a compreenderem e aplicarem conceitos das várias áreas do ensino de forma prática e segura. Por fim, será analisado o papel dos simuladores nas competições de robótica júnior.

2.1 As STEM na Educação

Como referido no capítulo anterior, as STEM são uma grande ferramenta para motivar e melhorar o ensino nas várias áreas do currículo do aluno. A procura por este tema e a investigação dos seus benefícios têm vindo a ter um crescimento exponencial na última década [5].

Nos Estados Unidos da América, no primeiro ano da presidência de Barack Obama houve mudanças no sistema de educação, fomentando a aplicação de metodologias STEM devido à "da fraca performance dos estudantes nos Estados Unidos da América em matemática e ciências" [6]. Até ao momento o país encontrava-se atrasado a nível de inovação e implementação destas áreas comparativamente com outros países. Carla C. Johnson defende ainda que "muitas escolas em todo o país, particularmente no ensino básico, deixaram de ensinar as áreas da ciência e negaram o acesso às novas tecnologias aos alunos" devido à implementação de políticas relacionadas com a Lei "No Child Left Behind"nessas escolas [6].

Tal como os Estados Unidos da América, outros países começaram a apostar mais nas STEM. Na Suiça foi criada a comunidade Roteco que se debruça-se na área da robótica, na qual os professores integrantes têm acesso a vários recursos e ajudam-se mutuamente [7]. Do estudo efetuado conclui-se que a maior parte dos professores juntaram-se à comunidade para se manterem informados e na vanguarda sobre as atividades ligadas à robótica. Só alguns utilizadores da plataforma é que desenvolviam conteúdos para serem carregados na mesma.

Na Europa o projeto EuroSTEAM [8] com o objetivo de desenvolver materiais e formação para os professores utilizarem em contexto de sala de aula, juntou sete escolas de vários países para identificar problemas existentes na educação. Todos os países apresentaram valores não desejáveis no alcance dos conhecimentos de matemática e ciência e detetou uma grande dificuldade em fazer comparações entre os países por não haver uma estrutura universal para a implementação das STEM.

Em Portugal existe uma emergente adesão por parte das escolas às salas do futuro, inspiradas no projeto "Future Classroom Lab" [9]. Este projeto visa reestruturar a sala de aula convencional, de modo a utilizar as STEM como suporte na educação. Como demonstra a Figura 2.1, estas salas estão divididas em zonas distintas com propósitos diferentes e, não só, utilizam material como robôs e computadores mas também material de desenvolvimento de conteúdo digital para promover os vários projetos.

Figura 2.1: Exemplo de uma sala do Futuro [10].

A pandemia provocada pelo o vírus SARS-COV-2 veio fomentar ainda mais a revolução digital. A necessidade de manter o ensino de forma remota acelerou o processo e também expôs as dificuldades que algumas famílias tinham para aceder aos meios necessários. As escolas em Portugal passaram a apostar ainda mais nas novas tecnologias e na adaptação dos alunos e professores a esta nova realidade.

Dos vários estudos e artigos identificados todos reconhecem a importância das STEM no desenvolvimento pessoal e profissional do estudante. Nesta nova década cada vez mais estas valências são agentes diferenciadores e que podem vir a ter no futuro um grande impacto na seleção de novos colaboradores. É reconhecido pela comunidade escolar que a utilização das STEM promove a capacidade de resolução de problemas, as competências comunicacionais, o pensamento crítico e criativo. Estas são aptidões que devem ser exploradas através das STEM nas várias disciplinas do aluno durante o seu percurso escolar.

2.2 A Robótica como um instrumento educativo

O uso de robôs na sala de aula começa a ter um número de alunos bastante elevado pelo mundo fora. A utilização destes traz bastantes vantagens não só a nível de ensino de programação, bem como no ensino de boas práticas de utilização de dispositivos de hardware [11]. Estes são geralmente vistos como agentes facilitadores do ensino nas áreas das Tecnologias da Informação e Comunicação (TIC), mas na realidade, muito provavelmente, podem ser aplicadas a todas as outras disciplinas presentes no currículo escolar.

Com o interesse pela Robótica Educativa (RE) a aumentar começam a surgir projetos que promovem esta área de forma mais assertiva. Um dos casos é a sociedade Scuola di Robotica em Itália, que desde 2008 é um centro educativo aprovado e certificado pelo Ministério da Educação italiano [12]. A sua missão é organizar conferências e preparar aulas para as escolas, formar professores em RE, entre outros. Começou por implementar um projeto denominado por Robot-in-the-Classroom. Este projeto não só visa implementar robôs na educação dos jovens e facilitar o ensino das STEM, como também fornecer formações aos docentes sobre os vários conceitos. Nos dias de hoje a sociedade já desenvolve o próprio software que o aluno e professor vão utilizar, operando em mais ou menos 130 escolas que vão desde o pré-escolar até ao 12º ano de escolaridade.

Mesmo existindo um grande avanço no uso da RE nas escolas, um estudo revela que ainda existe um grande receio por parte dos professores em utilizar um robô como meio educativo [13]. Destaca também que existe uma grande ansiedade provocada pelo o próprio conceito da robótica, no entanto realça que os professores percebem as grandes vantagens que o uso destas ferramentas podem trazer. Assim, este estudo aponta que devem ser feitos mais esforços para estudar as dificuldades e relutâncias que os professores possam apresentar face à Robótica Educativa.

O Departamento da Educação italiano continua apostar na RE, visto que compreende as grandes vantagens que esta pode trazer. Um estudo realizado em escolas de primeiro ciclo [14] foi determinado que o uso de robôs na sala de aula pode aumentar a capacidade de resolução de problemas. Neste foram implementadas várias

sessões com diferentes robôs e softwares, e no final após a realização de vários testes concluíram que o aluno melhorou a capacidade de compreensão, representação e categorização por parte do aluno. Determina ainda que o processo de ensino através da robótica pode ter várias vertentes, podendo passar pela corroboração da teoria com a pratica ao uso da narrativa para promover a literacia digital.

Com vários projetos por todo o mundo a utilizarem a Robótica Educativa, nasce uma nova necessidade: o desenvolvimento de robôs especializados para o uso da comunidade escolar. Uma das grandes empresas que tem apostado fortemente nesta área é a LEGO [15] que utilizou o seu produto para facilitar a construção de um robô e desenvolveu plataformas digitais capazes de conter tutoriais e zonas para programar os robôs. A LEGO apresenta um catálogo extenso de *kits* para o desenvolvimento e programação de vários robôs com variados graus de complexidade consoante a idade a que o *kit* é destinado. Uma das grandes vantagens que estes têm face à competição é que são compatíveis com todas a peças LEGO, o que leva a que ao invés de um kit ser utilizado para um único fim, possa ser utilizado para desenvolver inúmeros robôs com intuitos distintos. O produto mais recente e mais versátil desta empresa é o "LEGO MINDSTORMS Robô Inventor", que tem a possibilidade de montar cinco robôs com propósitos diferentes, como apresentado na Figura 2.2, e a possibilidade de programar pela a aplicação informática prioritária, "LEGO MINDSTORMS App", que se encontra disponível para todas as plataformas digitais.

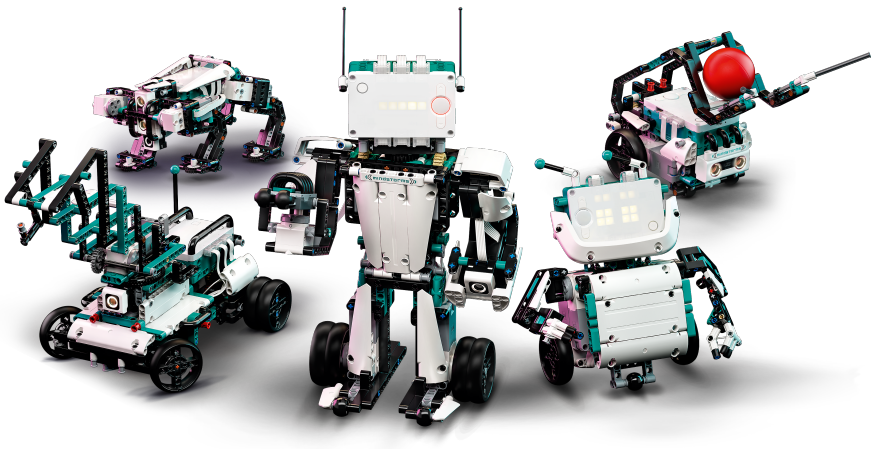


Figura 2.2: Robôs standard do LEGO kit "MINDSTORMS"[16].

Outra empresa que tem contribuido muito na área da robótica educativa é a Dobot [17]. Esta empresa oriunda da China cresceu de forma exponencial nos últimos anos contendo uma variedade enorme de produtos mas focando-se essencialmente na área dos braços robóticos. Estes podem vir acompanhados por vários kits de extensão que permitem ao utilizador recriar exemplos presentes na indústria que

utilizem braços robóticos. Os kits podem ser de visão e de inteligência artificial, como também de tapetes rolantes para transporte de peças. Outros produtos educacionais presentes no catálogo da empresa são as impressoras 3D e as plataformas robótica móveis.

Na Figura 2.3 encontra-se o braço robótico denominado Magician produzido e vendido pela Dobot e na Figura 2.4 está presente a interface do *software* para programação do mesmo.

Figura 2.3: Kit Braço robótico da empresa Dobot [17].

Figura 2.4: Interface do Dobot Studio [17].

Outra empresa que compete neste mercado é a Makeblock [18]. que se foca maioritariamente na venda de plataformas robótica móveis para a educação. O mBot é o produto exemplar da empresa. Tem vários sensores e atuadores que permitem ao utilizador programar o robô para efetuar enumeras operações possibilitando assim uma maior aprendizagem de conteúdos STEM de forma fácil e eficaz. Também apresenta algumas extensões adicionais que proporcionam a adição de novas funções ao robô como por exemplo a possibilidade de se movimentar em terrenos mais complexos ou a possibilidade de ter uma garra para manipulação de objetos. Mais recentemente foi lançada uma nova versão do mBot, o mBot Neo, representado na

Figura 2.5, que traz uma unidade de processamento mais avançada com um maior número atuadores e sensores, tais como microfone e colunas capazes de produzir música, enriquecendo assim a experiência da sua utilização.

Figura 2.5: Robô móvel mBot Neo [18].

Com a grande evolução e dependência da tecnologia nas várias áreas de trabalho, a BBC decidiu reviver e atualizar uma das suas premissas de introduzir a programação nas escolas da Grã-Bretanha e mais recentemente por todo o mundo. O BBC Micro:bit [19] é o sucessor do BBC Micro que promete encorajar e introduzir a programação no currículo escolar. O BBC Micro:bit, apresentado na Figura 2.6, é um dispositivo com diversos sensores e atuadores, já embutidos na placa, que permite ao aluno programar diversos desafios. Atualmente, esta plataforma é dirigida por uma organização sem fins lucrativos.

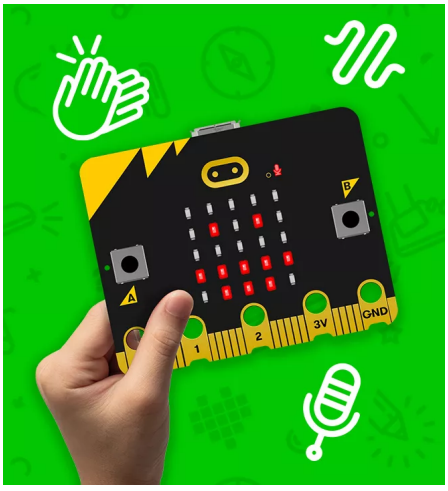


Figura 2.6: Plataforma Micro:bit [20].

Na sua versão mais recente, contém uma matriz de LEDs totalmente programáveis, sensores de temperatura, luminosidade e acelerómetro tornando o dispositivo

bastante versátil. Também apresenta um conector que pode ser utilizado para incrementar as suas funcionalidades, com um circuito numa placa de ensaios ou numa plataforma robótica com mais sensores e atuadores.

É de notar que estas soluções utilizam todas uma linguagem de programação comum, mas também têm sempre a possibilidade de programar através de blocos (programação gráfica) que são baseados na biblioteca Blockly [21] desenvolvida pela Google. A programação por blocos utiliza o processo de arrastar e largar, em inglês, *drag-and-drop*, e cada bloco contém linhas de código em texto embutidos. Este tipo de abordagem tem bastantes vantagens ao nível educacional dos mais novos sobre a programação. A maior vantagem da programação por blocos sobre a programação por texto é não haver preocupação relativamente a como o utilizador escreve o código. Cada linguagem de programação utiliza uma sintaxe diferente e quando mal aplicada o código irá apresentar erros. Na linguagem por blocos, como é utilizada uma linguagem visual, o aluno só tem de se preocupar em colocar os blocos nos locais corretos para efetuar o que pretende, evitando assim a ortografia e pontuação que numa linguagem de programação por texto requer.

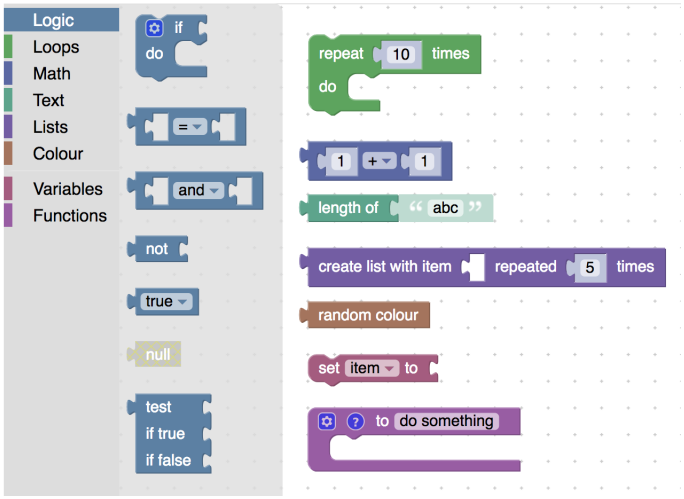


Figura 2.7: Linguagem de programação por blocos "Blockly"[21].

2.3 A Simulação como auxílio na Robótica Educativa

A introdução dos robôs no ensino regular também trouxe algumas desvantagens em relação ao custo de aquisição dos mesmos, bem como a manutenção que estes exigem numa escola. Outra desvantagem a anunciar é a dificuldade de transporte [22] que pode danificar o material, por isso a maioria das escolas só disponibilizam os robôs numa sala restrita, não permitindo, muitas das vezes, a utilização diária por todos os alunos.

Apesar do uso de simuladores não substitui o uso de robôs, pode ser uma solução a considerar, visto que ajudam a desenvolver aulas adicionais que permitem aumentar o uso da RE no plano curricular, minimizar custos de manutenção, e demonstrar ao aluno o método que muitas empresas e investigadores utilizam para aplicação de robôs no mundo real.

Na última década ao serem desenvolvidos robôs destinados à educação, surge por consequência o desenvolvimento de simuladores destinados ao ensino. Existem dois tipos de simuladores: os que precisam de ser instalados no computador e os que estão disponíveis online com acesso através do browser [22]. Ambos têm as suas vantagens e desvantagens e é preciso compreende-las para escolher qual o mais adequado para satisfazer cada tipo de necessidade. A grande vantagem dos simuladores online é que não precisam de muito processamento gráfico e não requerem qualquer tipo de instalação, basta o utilizador ter uma conexão à internet para puder utilizar. No entanto, neste tipo de conexão pode ser uma desvantagem, visto que a rapidez e a qualidade variam de escola para escola, e aquelas que não tenham uma conexão à internet excelente podem não conseguir ter a melhor experiência. O Open Roberta Lab e o VEXcode VR são exemplos de simuladores online.

O Open Roberta Lab [23] é um software que provém da iniciativa Roberta – Learning with Robots que começou em 2002 e tem como objetivo motivar o aluno a aprender conteúdos sobre tecnologia e ciências naturais. Este software foi desenvolvido para que todos os alunos em todo o mundo aprendam a programar um robô sem necessitar de ter um real. Este tem a vantagem de ter vários robôs didáticos modelados no próprio simulador, como os robôs da LEGO em configuração de Veículo Automático Guiado, em inglês, Automated Guided Vehicle (AGV), as duas versões do mBot desenvolvido pela MakeBlock e até o robô NAO, que é um robô humanoide usado para a educação [24]. O software contém os blocos de programação idênticos aos blocos que os permitem programar. Assim o aluno tem a oportunidade de programar em casa um robô que tenha experimentado dentro da sala de aula.

A Figura 2.8 apresenta a disposição do Open Roberta Lab, onde do lado esquerdo está o menu dos blocos de código, no centro é a zona de trabalho e no lado direito está a janela do mundo simulado.

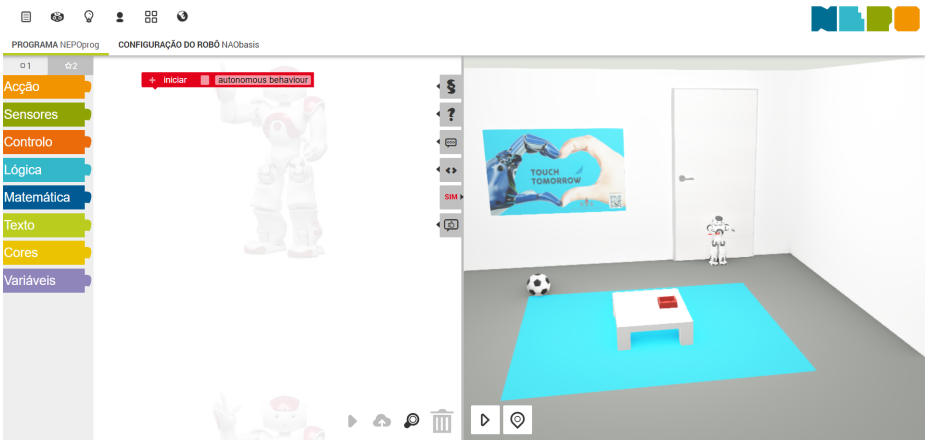


Figura 2.8: Ambiente de simulação com zona de programação do simulador online Open Roberta Lab [23].

O VEXcode VR [25] permite que o aluno programe um AGV em diferentes cenários. Este robô está equipado com vários sensores, como por exemplo, sensor de distância, sensor de posição e sensores de pressão. Além dos motores, o AGV apresenta um íman para capturar alguns objetos para o robô executar tarefas de transporte, e uma caneta para que este consiga escrever ou desenhar o que o aluno pretender. Ao ter bastantes cenários onde o AGV pode operar, este software torna-se interessante pois permite que o professor desenvolva várias aulas com base nos vários desafios que os cenários apresentam. A Figura 2.9 expõe o ambiente de trabalho do software VEXcode VR e o cenário mais básico que o robô simulado pode operar.

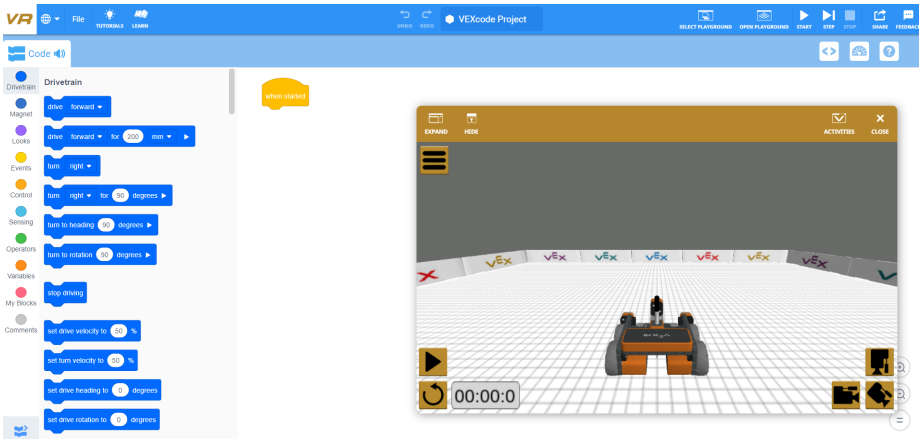


Figura 2.9: Simulador VEX com o AGV virtual presente na arena de desenho [25].

Em relação aos simuladores que necessitam de instalação, existem muitas opções

com as suas vantagens e desvantagens. Simuladores como o Khepera Simulator e o USARSim [22] permitem a programação de um robô predefinido em linguagem de programação C. O Gazebo [26] e o SimTwo [27] permitem a modelação do robô que o utilizador vai utilizar, o cenário tridimensional (com simulação realista) e a programação das tarefas que o robô terá de efetuar. Ambos são softwares gratuitos e de código aberto, permitindo assim uma maior versatilidade durante a sua utilização. Uma das vantagens que estes softwares apresentam relativamente a outros, é a possibilidade de utilizar o hardware em conjugação com o software. Por exemplo, é possível utilizar um microcontrolador atmega328 (Arduino UNO) [28] para receber os dados dos sensores simulados e controlar os atuadores do robô simulado. Assim, o utilizador poderá minimizar o tempo de programação, uma vez que o código utilizado para a simulação poderá ser muito idêntico para o robô real, e se for totalmente igual, ao ser realizado dentro do microcontrolador, rapidamente o aluno transita da simulação para o ambiente real. Este método tem como denominação Hardware-in-the-loop.

Na Figura 2.10 é possível observar as várias janelas necessárias na utilização do software SimTwo. As opções de conexão ao mundo simulado apresentam-se na janela do lado esquerdo do *software*. Ao centro é apresentado o mundo simulado. Por fim, o utilizador consegue obter dados e enviar comandos para validação a partir da tabela posicionada do lado direito.

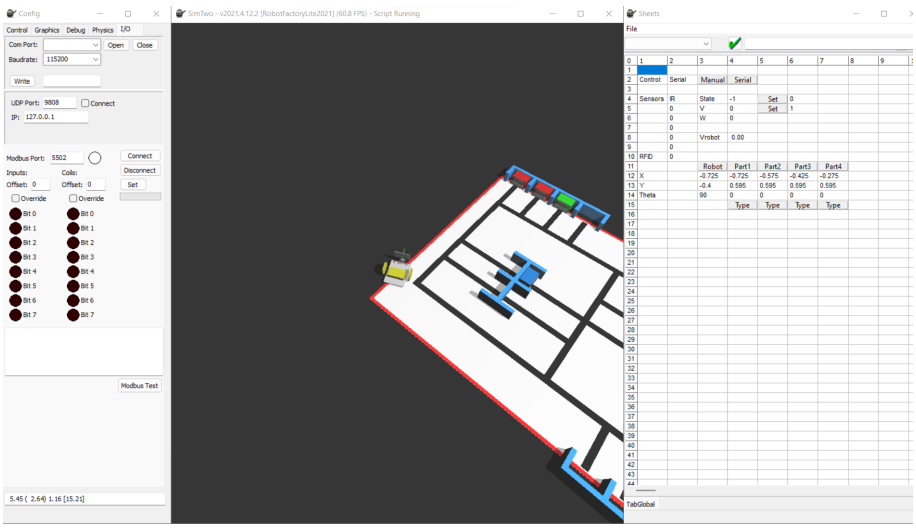


Figura 2.10: Software Simtwo. [27].

O software Webots [29] é um simulador de robôs móveis totalmente gratuito que permite ao utilizador desenvolver ambientes tridimensionais muito detalhados e de forma muito intuitiva. O Webots contém uma interface gráfica para o desenvolvimento do robô e do ambiente. Este já contém os sensores e atuadores mais utilizados na robótica móvel para tornar o processo de criação do robô mais eficiente. O simulador também apresenta vários robôs e ambientes pré instalados, com tutoriais

disponíveis para cada um deles para ajudar o utilizador na sua primeira aventura no mundo da robótica simulada. O Webots apresenta várias interfaces para com robôs reais, sendo possível desenvolver um código no software e carregar diretamente no robô real, por exemplo o e-puck, NAO, entre outros. Ao contrário de outros simuladores, o Webots permite programar em diferentes linguagens, sendo estas C, C++, Java, Python e MATLAB, tornando o software acessível a um maior grupo de utilizadores.

2.4 A Simulação nas Competições de Robótica Júnior

No ano de 2023, em Tomar, no Festival Nacional de Robótica [30] das sete provas júnior realizadas, uma delas é totalmente em ambiente de simulação e outra apresenta um simulador como auxílio. Como referido anteriormente o uso de simuladores apresenta grandes vantagens, assim o uso dos mesmos nas competições contribuem para apelo de um maior leque de pessoas, pois não precisam de adquirir um robô para participarem. As provas RoboCup Junior Rescue e Robot Factory Lite utilizam ambiente de simulação. São provas júnior que simplificam as provas originais, tornando-as mais acessíveis ao público jovem. Na Figura 2.11 demonstra o labirinto e o robô E-Puck da prova RoboCup Junior Rescue.

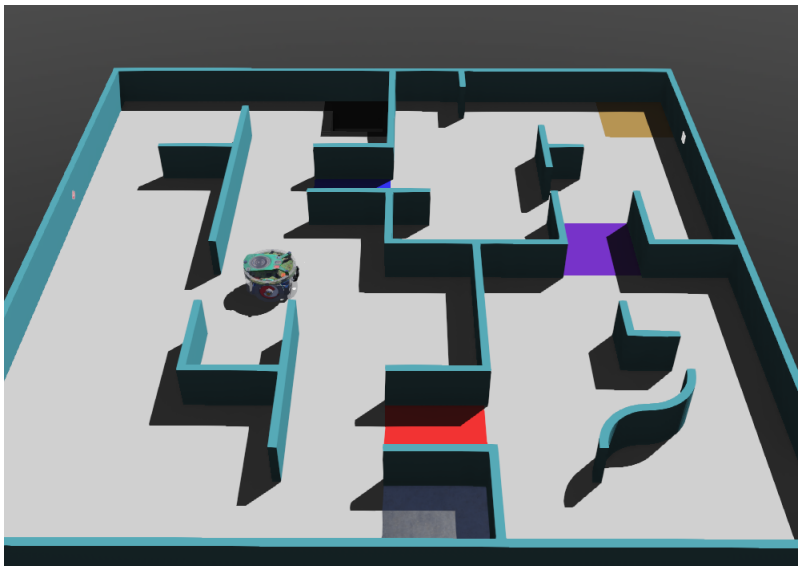


Figura 2.11: Labirinto e robô RoboCup Junior Rescue [31].

A prova RoboCup Junior Rescue [31] permite aos alunos mais jovens explorarem o mundo da robótica e da programação utilizando um ambiente simulado da prova RoboCup Rescue. Nesta o robô virtual tem que se deslocar num labirinto com diversas divisões nas quais podem estar "vítimas"que o robô terá de identificar. O software de simulação utilizado é o Webots e foi escolhido após alguns testes com equipas voluntárias. A primeira demonstração foi numa competição virtual em junho

de 2020 (após a primeira quarentena provocada pelo o vírus SARS-Cov-2) e nos anos seguintes foi refinado para aprimorar a competição.

Os participantes têm um orçamento para os diversos componentes a colocar na plataforma robótica E-Puck (robô já existente no simulador Webots). As localizações de cada componente também são escolhidas pela equipa tornando todos os robôs virtuais únicos, com métodos de exploração e deteção diferentes. A prova também pode apresentar níveis mais difíceis com terrenos e labirintos distintos.

A prova Robot Factory Lite [32] é uma simplificação da Robot Factory destinada ao público mais jovem. Nesta os participantes têm que programar um robô para deslocar umas caixas de um ponto de origem a um ponto de destino. Durante a prova é simplificado o método de localização e movimentação que o robô deve efetuar (seguimento de linha preta). A Robot Factory Lite é composta por três mangas, na primeira o robô tem de arrastar umas caixas utilizando um eletroíman do ponto A para o ponto B. Na segunda manga a caixa tem de passar por um ponto intermédio e na terceira manga algumas caixas terão de passar por dois pontos intermédios.

Os dirigentes desta prova desenvolveram o simulador SimTwo para os alunos poderem efetuar o código e testa-lo em ambiente de simulação. Uma das grandes vantagens é utilizar o próprio Arduino para comunicar com o simulador, visto que, os dados dos sensores simulados são enviados para o Arduino e este controla os atuadores do robô simulado, tornando a transição do simulador para o ambiente real e o tempo de desenvolvimento mais rápido. Permitem também que equipas desenvolvam diferentes códigos sem necessitarem de ter a pista e o robô disponíveis.

Capítulo 3

Proposta de Solução

Neste capítulo serão explorados todos os tópicos necessários para executar uma proposta de solução que satisfaça todos os objetivos definidos para esta dissertação.

Será recriado a prova e o robô no simulador Webots uma vez que contém, já de origem, muitos componentes virtuais capazes de simular os componentes reais mais utilizados na prova First Challenger. O Webots encontra-se atualizado, com muita documentação e tutoriais, sendo o simulador utilizado nas provas internacionais de robótica da Robocup. Estas valências tornam o processo de implementação mais rápido, permitindo avançar para a fase de testes e corrigir os diversos erros e dificuldades que poderão surgir durante a utilização do simulador. Ao permitir várias linguagens de programação, os participantes poderão utilizar para concorrer uma maior diversidade de microcontroladores.

Como a Academia de Robótica visa utilizar este simulador em contexto de sala de aula, é igualmente necessário desenvolver todos os mecanismos para o mesmo efeito. O objetivo final é que o aluno consiga desenvolver o código e testá-lo no simulador e de forma quase impercetível aplicá-lo no robô real. Para isso todos os blocos e funções/procedimentos utilizados têm de ser idênticos aos que já existem na Academia de Robótica. Para além disso, é fundamental o desenvolvimento de um conjunto de aulas e materiais de apoio ao aluno, de forma a conseguir desenvolver uma solução de código no Webots, assim como aprender a utilizar o simulador da forma correta.

3.1 Prova First Challenger do Festival Nacional de Robótica

O Festival Nacional de Robótica [33] é um evento anual que se realiza em diferentes cidades de Portugal e tem como propósito promover as STEM, assim como diversas competências nomeadamente o espírito crítico, o trabalho em equipa, a capacidade de resolução de problemas, entre outros através de competições de robótica. Uma dessas competições é a prova "First Challenger"que tem como propósito ser uma competição introdutória ao mundo da robótica para jovens estudantes, das escolas básicas até ao secundário e ensino profissional. A prova encontra-se dividia em três níveis, nos quais a complexidade e dificuldade vai aumentando. O aluno para poder participar tem de utilizar um robô desenvolvido e montado por ele, não podendo utilizar plataformas robóticas já prontas ou que contenham todo o código já disponível, como por exemplo o robô LEGO Inventor.

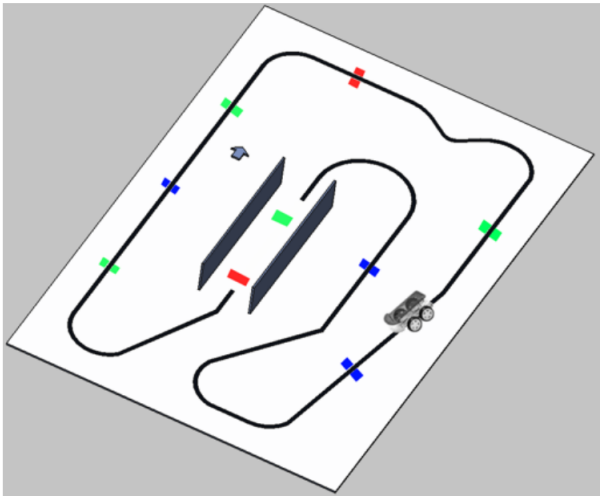


Figura 3.1: Imagem ilustrativa do circuito apresentada nas regras da prova First Challenger do circuito [34]

3.1.1 Nível 1

Neste primeiro nível o robô programado pelo o aluno tem de ser capaz de seguir uma pista, utilizando um sensor de pista/linha, no menor tempo possível.

3.1.2 Nível 2

No segundo nível a pista apresenta várias tiras coloridas com três cores principais: verde, vermelho e azul ao longo da linha preta. O robô tem de segui-la e ao mesmo tempo ser capaz de detetar as cores, utilizando um sensor de cor. Após a deteção deve parar, ativar um *buzzer* e ligar a cor correspondente utilizando um LED RGB.

3.1.3 Nível 3

No terceiro nível uma zona de linha preta é substituída por duas paredes. O robô deverá se orientar utilizando sensores de distância e continuar a seguir o percurso da linha, parar e identificar as cores quando detetadas.

3.2 Sensores e Atuadores utilizados na prova

- **Sensor Ultrassónico**

  O sensor ultrassónico HC-SR04 [35] é um sensor capaz de medir a distância entre ele e um objeto, através do fenómeno do eco (retorno de uma onda sonora), utilizando ondas da gama de frequência dos ultrassons. O HC-SR04 é considerado um sensor de Tempo de Voo, em inglês *Time of Flight* (ToF), pois consegue determinar a distância através do tempo que a onda demora a efetuar uma viagem, desde que é emitida até ser captada pelo o recetor. Sabendo que a velocidade do som é de 340 m/s e ao determinar o tempo que a onda demorou durante a viagem, é possível obter a distância através do cálculo das funções 4.3 e 3.2.

$$Velocidade = \frac{Distância}{Tempo} \tag{3.1}$$

$$Distância = Velocidade * Tempo \tag{3.2}$$

  Como o tempo contabilizado da viagem é referente a um trajeto de ida e volta é necessário dividir a distância por dois, conforme apresentado na função 3.3.

$$Distância = \frac{Velocidade * Tempo}{2} \tag{3.3}$$

O HC-SR04 apresenta quatros pinos, representado na Figura 3.2, dois de alimentação, um de entrada (trigger) e um de saída (echo).

Figura 3.2: Sensor Ultrassónico HC-SR04 [36]

O pino "trigger"é aquele que ativa o emissor para produzir a onda ultrassónica. O pino "echo"vai enviar um pulso para o microcontrolador quando recebe e deteta a onda ultrassónica. O tempo que este pino se encontra elevado é proporcional ao tempo que a onda demorou a fazer a viagem de ida e volta. A Figura 3.3 apresenta o gráfico de tempos de controlo para controlar o sensor.

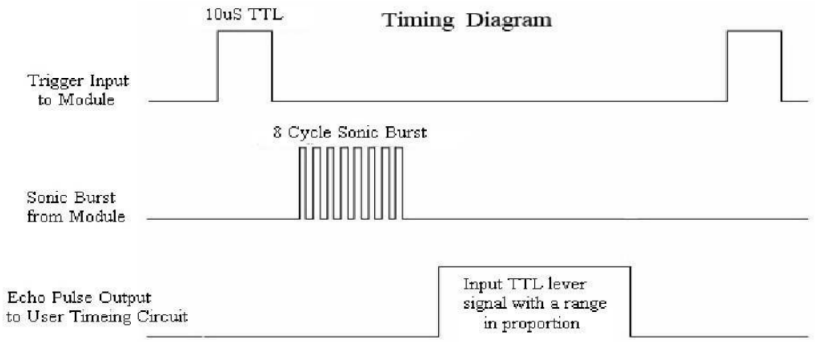
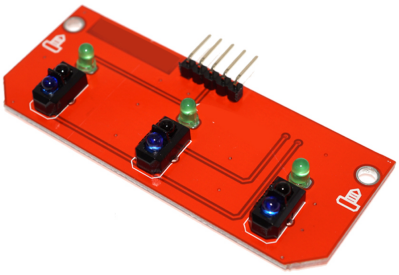


Figura 3.3: Diagrama de tempo para controlo do sensor [35]

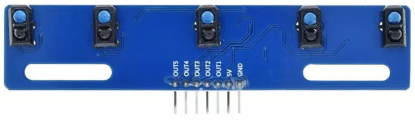
- **Sensor de Pista**

  Os sensores de pista utilizados são na verdade módulos com vários sensores de infravermelho ativos (produzem e detetam o sinal), o TCRT5000 [37]. Estes são compostos por um emissor e um recetor que mede a quantidade de luz infravermelha que é refletida. Existem vários módulos para o mesmo efeito alterando o número de sensores constituintes ou também a sensibilidade e altura indicada para efetuar a montagem.

Alguns exemplos destes módulos de sensores de pista podem ser visualizados na Figura 3.4.



(a) Módulo com três sensores [38]



(b) Módulo com cinco sensores [39]

Figura 3.4: Exemplos de Módulos de Deteção de Linha

Ao longo dos anos da competição "First Challenger"foram utilizados vários módulos de deteção de linha mas os mais comuns são os com três e cinco sensores infravermelhos, que apresentam uma saída digital (0 e 1) para cada um dos sensores. De realçar que os alunos da Academia utilizam uma biblioteca, que será explorada no próximo capítulo, que junta todas as saídas digitais do sensor tornando-a num número binário e convertendo-o num número decimal. A Tabela 3.1 demonstra um exemplo de valor de saída para um módulo com três sensores TCRT5000.

Esquerda	Centro	Direita	Binário	Decimal
0	0	0	000	0
0	0	1	001	1
0	1	1	011	3
0	1	0	010	2
1	1	0	110	6
1	0	0	100	4
1	1	1	111	7
1	0	1	101	5

Tabela 3.1: Valores de Saída Binária e Decimal com três sensores de linha

- **Sensor de Cor**

  O sensor de cor TCS3200 [40], apresentado na Figura 3.5 é composto por uma matriz de fotodíodos com quatro diferentes filtros e é utilizado pelos alunos da Academia para detetarem as cores presentes na pista da prova First Challenger. Estes filtros permite ao sensor captar informações sobre as diferentes intensidades das cores primárias, bem como da luz que lhe incide.

Tem 16 fotodíodos com um filtro de cor vermelha, 16 com filtro de cor verde, 16 com filtro de cor azul e 16 sem qualquer filtro.

Figura 3.5: Sensor de Cor TCS3200 [41]

O TCS3200 contém um conversor de corrente para frequência que vai converter o sinal ótico num sinal com onda quadrada. A frequência deste sinal vai variar consoante a intensidade da luz. O sensor apresenta dois pinos para seleção da cor que o utilizador pretende ler e dois pinos para escalonar o valor da frequência do sinal. Tem ainda um pino de saída que envia o sinal de frequência para o microcontrolador. A Academia de Robótica tem um extensão que faz a leitura do sensor de cor e também a atribuição dos valores RGB para as diferentes cores que estão presentes na pista.

- **Motor de Corrente Contínua**

  A plataforma robótica incluída no kit contém quatro motores de corrente contínua [42]. Estes têm uma tensão de 5V e um máximo de corrente de 180mA. O motor tem um redutor de velocidade com um rácio de 1:48 e um máximo de 208rpm. Estes estão representados na Figura 3.6 com a roda e pneu que também está incluído.

Figura 3.6: Motor DC com caixa redutora 1:48.

Os motores DC, responsáveis pelo o movimento do robô, são componentes essenciais em muitos projetos de robótica móvel e de automação devido à rápida e fácil aplicação. Quando combinados com uma caixa redutora, estes motores tornam-se ainda mais versáteis, oferecendo um maior controlo sobre o binário e a velocidade.

- **LED RGB**

  Um LED RGB [43], *Light Emitting Diode - Red*, *Green*, *Blue*, é um tipo de díodo emissor de luz que pode reproduzir várias cores (vermelho, verde, azul) e combiná-las. Este tipo de LED, apresentado na Figura 3.7, é amplamente utilizado em diversas aplicações, desde dispositivos eletrónicos, iluminação decorativa/natal e no caso dos participantes do "First Challenger"serve para indicar a cor que o robô está a detetar.

Figura 3.7: LedRGB com 4 pinos. [44]

O LED RGB é composto por três díodos individuais, comprimidos numa única cápsula, cada um capaz de emitir uma das três cores primárias. É um dispositivo digital que tem sempre a mesma corrente e tensão enquanto se encontra ligado. Para variar e representar várias cores tem que ser controlado com um sinal de Modulação de Largura de Pulso, em inglês *Pulse With Modulation*, PWM. Ao variar a largura do pulso , é possível ajustar a luminosidade de cada cor primária e, consequentemente obter novas cores. Com este sinal o LED irá "piscar"de forma muito rápida que a olho nu parece que diminuiu a intensidade da luz emitida.

Na Figura 3.8 é possível observar as cores primárias e secundárias do espetro de cores para um LED RGB, bem como um sinal de PWM e as diferentes percentagens entre ligado e desligado.

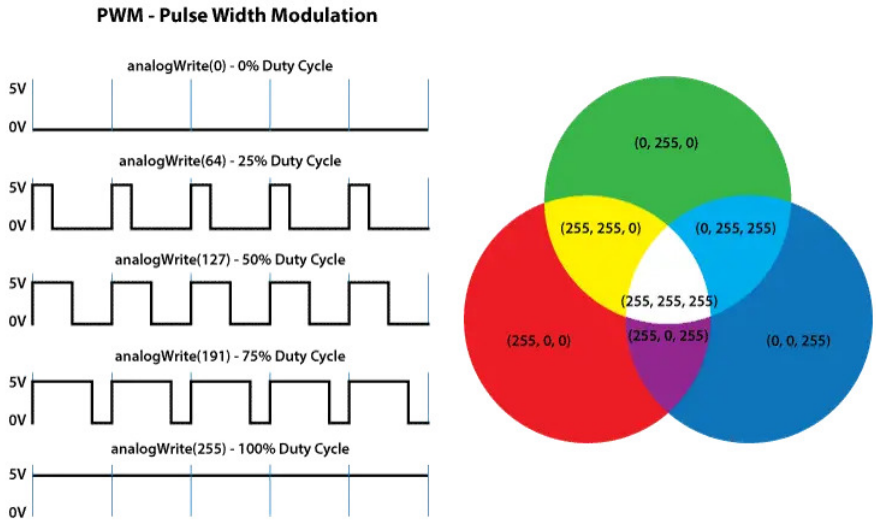


Figura 3.8: Sinal PWM. [43]

Os LEDs RGB podem partilhar o terminal positivo, no qual os três LEDs estão conectados pelo pino positivo (ânodo comum) ou partilhar o terminal negativo, onde os três LEDs estão conectados pelo pino negativo (cátodo comum). Por isso, é que o LED RGB só apresenta quatro pinos.

- **Buzzer**

  O Buzzer de 5V [45] é um dispositivo eletromecânico ou piezoelétrico utilizado para gerar ondas sonoras e é um componente amplamente aplicado em diversos sistemas eletrónicos pela sua simplicidade e baixo custo. Durante a realização da prova "First Challenger"o Buzzer de 5V permite ao aluno colocar o robô a emitir um sinal sonoro ao parar numa cor.

Figura 3.9: Buzzer 5V [46]

Existem dois tipos principais de buzzers: os eletromecânicos e os piezoelétricos. O buzzer electromecânico, presente no kit da Academia de Robótica, consiste numa bobina magnética, uma membrana e um íman. Quando a corrente elétrica passa pela bobina cria um campo magnético, que movimenta a membrana dando origem a um som. A frequência do som é controlado por um sinal PWM.

3.3 mBlock

O mBlock [47] é uma plataforma de programação por blocos desenvolvida pela Makeblock e é baseada no Scratch que foi desenvolvida pelo MIT [48]. Esta ferramenta facilita a aprendizagem de conceitos de programação e robótica ao utilizar uma interface visual muito intuitiva. O mBlock permite que utilizadores mais novos possam criar programas complexos. sem a necessidade de escrever qualquer linha de código, promovendo assim um ambiente de aprendizagem bastante interativo. A Figura 3.10 demonstra a interface de programação do mBlock Versão 3.

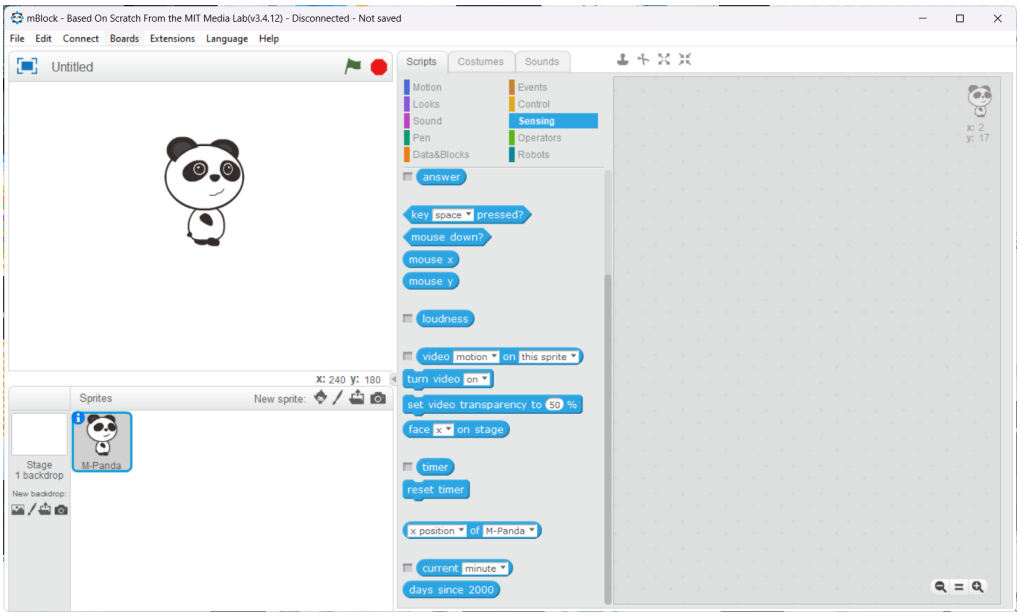


Figura 3.10: mBlock 3 - versão utilizada pelos os alunos da Academia de Robótica [18]

Os blocos standard do mBlock apresentam-se organizados por categorias:

- *Motion*: controla o movimento das personagens, *sprites*, criadas pelo o utilizador
- *Looks*: altera a aparência dos sprites.

- *Sound*: permite adicionar e controlar sons.
- *Pen*: controlos para desenhar na tela/palco.
- *Events*: blocos para inicio de código através do acontecimento de um certo evento.
- *Control*: blocos para controlo do código, como o *loop* infinito ou condições.
- *Sensing*: permite ao utilizador interagir com o *sprite* através dos vários "sensores"que este tem.
- *Operators*: operações matemáticas e lógicas de comparação.
- *Data&Blocks*: criação de variáveis/constantes e também de novos blocos.
- *Robots*: todos os blocos para o controlo de robôs reais que o mBlock permite programar.

O mBlock permite não só programar os robôs da empresa Makeblock, como também a programação de vários microcontroladores, como o Arduino UNO, Leonardo, MEGA e NANO, ou até o microcontrolador Pi Pico . Assim, o utilizador pode programá-los usando blocos predefinidos ou criando novos. Este último é o caso da Academia de Robótica que apresenta umas bibliotecas/-blocos, que serão explorados no Capítulo 4 do presente relatório, criados para simplificar o processo de programação aos alunos.

3.4 Academia de Robótica

A Academia de Robótica tem como objetivo de impulsionar as competências do séculos XI e das áreas da STEM através da robótica educativa [2]. É uma atividade lúdica que junta a teoria com a prática, desafiando os alunos, de idades entre os oito e 18 anos, a conhecer o mundo da eletrónica e da programação de forma interativa. Ao longo do ano, os alunos terão que desenvolver raciocínios lógicos e abstratos, e também vão trabalhar em equipa para se prepararem para competir em provas do Festival Nacional de Robótica.

O curso de robótica está dividido em quatro níveis:

- **Nível 1** - Introdução à robótica

  Este é o nível introdutório onde os alunos vão ter a sua primeira experiência com duas das grandes áreas da robótica, a programação e a eletrónica. Utilizam o microcontrolador Arduino [49] para programar diversos circuitos elétricos e no final obterem um robô capaz de seguir uma linha preta e parar quando detetado um obstáculo. Os alunos utilizam a ferramenta de programação mBlock [50], que permite desenvolver códigos e carregá-los diretamente para o Arduino.

A Figura 3.11 apresenta a bancada de um aluno que frequenta o curso de robótica na Academia.

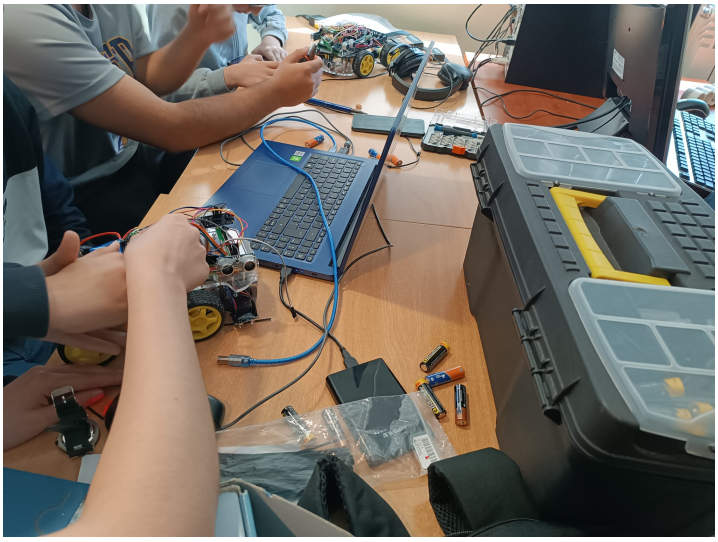


Figura 3.11: Bancada de trabalho dos alunos da Academia de Robótica

- **Nível 2** - Preparação e participação numa competição de robótica

  Neste nível, o aluno irá receber novos sensores e atuadores para preparar-se e competir na prova "First Challenger"do Festival Nacional de Robótica. Vão aprender novos conceitos de programação como a criação de procedimentos e funções. Na Figura 3.12 é possível observar os alunos da Academia de Robótica que participaram na prova do ano 2023.

Figura 3.12: Grupo de alunos da Academia de Robótica a participarem no FNR 2023

- **Nível +** - Introdução à linguagem de programação C

  Repetição do Nível 2 mas com a grande alteração do método de programação utilizado. Os alunos aprendem a programar o Arduino pela linguagem de programação C, identificando as vantagens e desvantagens de a utilizar. No final também se preparam para competirem de novo na prova "First Challenger".

- **Nível ++** - Vários projetos de robótica e outras linguagens de programação

  Este é o quarto e último ano que os alunos frequentam a Academia de Robótica e neste nível é promovido uma aprendizagem por projeto e de outras plataformas robóticas/microcontroladores. Os alunos também podem escolher participar noutras provas do Festival Nacional de Robótica.

Ao iniciarem o curso de nível 1 os alunos recebem um kit de robótica com os componentes necessários para alcançar o objetivo final desse ano, assim como outros que serão utilizados para promover diversas atividades durante as aulas. No segundo ano, este kit é incrementado com os componentes essenciais para os alunos participarem na prova do "First Challenger"e aprimorarem o seu robô. Em ambos os anos de programação por blocos e no nível +, devido à limitação de tempo e divergência do objetivo do curso, os alunos não programam todos os sensores e atuadores de raiz, isto é, muitos destes dispõem de bibliotecas já previamente programadas e instaladas nos computadores utilizados. Estas são:

- Controlo Motores - **Motores.c**
- Controlo Módulo Deteção Pista - **SensorDePista.c**
- Controlo Sensor Ultrassónico - **Sonar.c**
- Controlo Sensor de Cor - **ColorSensor.c**

Nestas bibliotecas estão presentes todas as funções e procedimentos que o aluno necessitará para programar cada um destes componentes, permitindo-lhe assim aprender e dedicar mais tempo a programar a lógica para resolver os problemas propostos durante as aulas e na prova.

Capítulo 4

Desenvolvimento

Neste capítulo serão exploradas todas as etapas de desenvolvimento para obter a prova First Challenger do Festival Nacional de Robótica em ambiente 3D.

4.1 O Mundo Virtual

Para a criação da prova no Webots é necessário, em primeiro lugar, criar um novo mundo/projeto. Este terá todas as informações, desde a quantidade de objetos, robôs, o mapa, as definições do mundo físico (por exemplo gravidade e forças de atrito) que vão definir como a simulação deve operar consoante os objetivos do utilizador. No Webots cada objeto é denominado por Nós, em inglês *Nodes*, que estão organizados de forma hierárquica numa Árvore da Cena, *Scene Tree*.

Cada novo mundo tem um diretório com diferentes pastas nas quais serão guardados os mundos, as bibliotecas externas e os controladores que posteriormente serão utilizados para programar os robôs. Cada projeto pode ter diferentes mundos, funcionalidade essa que vai ser utilizada para recriar as diferentes iterações da pista da prova First Challenger.

A Figura 4.1 apresenta o diretório criado para a prova que será instalado nos diversos computadores para lecionar as aulas correspondentes.

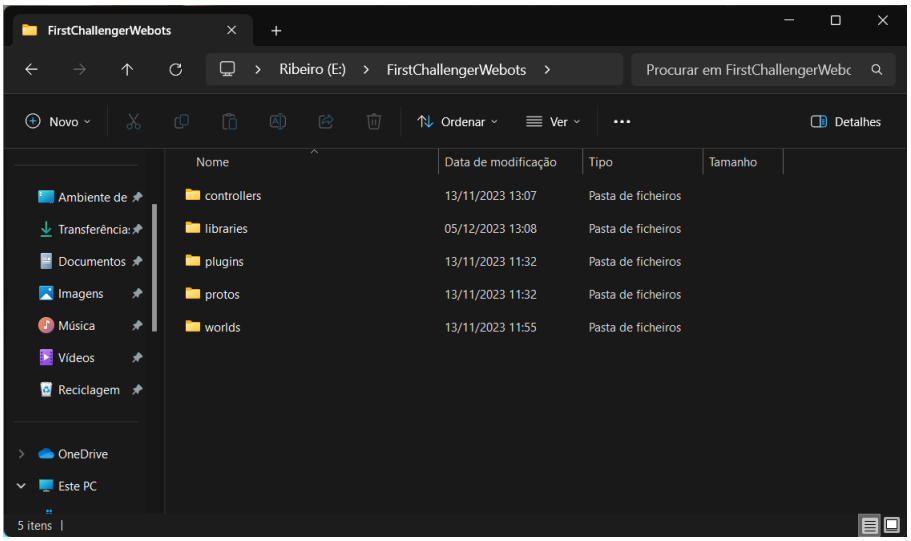


Figura 4.1: Diretório do projeto First Challenger

Ao abrir o projeto no Webots somos presenteados com uma janela, mais ou menos, dividida em quatro secções. A Figura 4.2 apresenta-a com as secções identificadas.

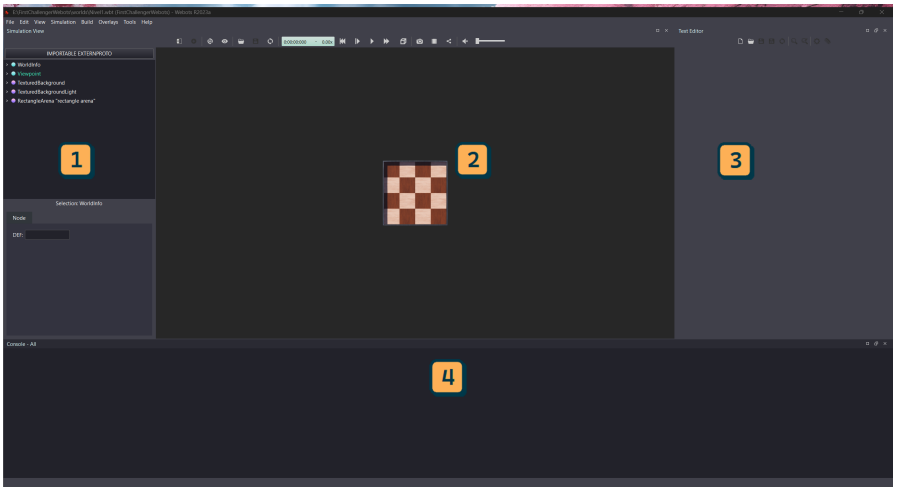


Figura 4.2: Janela Webots após criação de um novo diretório

A primeira secção, no lado esquerdo da janela, contém os diferentes Nós que o utilizador poderá alterar ou criar novos para complementar e assim desenvolver o mundo.

Ao criar um novo mundo são também criados, por predefinição, os seguintes Nós:

- **Worldinfo** - contém os parâmetros do mundo físico a simular
- **Viewpoint** - parâmetros da câmara que o utilizador vai utilizar para interagir e visualizar a simulação
- **TextureBackground** - contém a textura visível no fundo da simulação
- **TextureBackgroundLight** - contém as definições da luz para o fundo anteriormente definido e da simulação em geral (iluminação global)
- **RectangleArena** - arena retangular que o Webots apresenta por predefinição

Na segunda secção, ao centro da janela, temos o mundo simulado, no qual serão visíveis a arena, os robôs, os diferentes objetos criados e as janelas de controlo, como por exemplo, se o utilizador usar uma câmara. Estas janelas de controlo são compostas por uma barra de ferramentas, no topo, que permite guardar as alterações feitas aos vários Nós que compõem o mundo e ainda controlar o decorrer da simulação com botões de *reset*, pausa, reprodução e de reprodução na velocidade máxima. A Figura 4.3 demonstra os diversos botões desta barra de ferramentas.

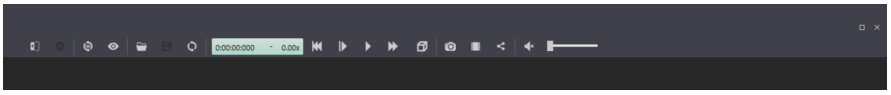


Figura 4.3: Barra de Ferramentas para o Mundo Simulado

A terceira secção, no lado direito da janela, é um editor de texto que vai permitir desenvolver o controlador na linguagem de programação pretendida. Na parte superior da mesma está também presente uma barra de ferramentas para gerir os controladores. Além dos botões de guardar, têm os botões de compilar e de limpar a compilação anterior. A Figura 4.4 apresenta os botões da barra de ferramentas para os controladores.

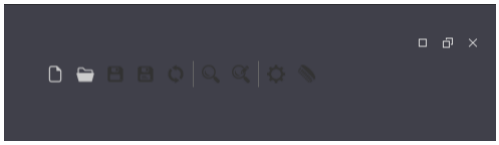


Figura 4.4: Barra de Ferramentas para os controladores

A quarta e última secção, é uma consola onde o programa indica erros e emite avisos que poderá identificar ao compilar o controlador/mundo e, demonstra também algum tipo de informação que o utilizador pretenda evidenciar durante o decorrer da simulação.

4.1.1 A Pista First Challenger

Uma das vantagens de utilizar o *software* Webots é que este permite a criação de objetos tridimensionais (3D) utilizando modelos desenvolvidos em programas de desenho assistido a computador, em inglês *Computer Aided Design*(CAD). Foi utilizado o *software* online gratuito Onshape para criar um modelo 3D da pista para depois ser introduzido no projeto final. A Figura 4.5 apresenta uma das pistas utilizadas pelos alunos para testarem os seus robôs antes do FNR.

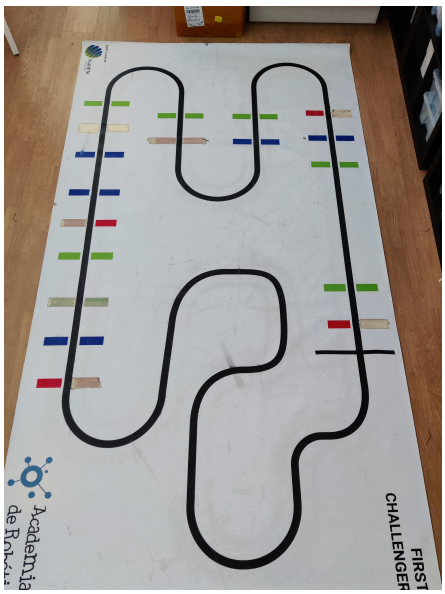


Figura 4.5: Pista da prova First Challenger

Uma vez que é uma pista de teste muitas das cores encontram-se tapadas para tornar o processo de testes mais rápido. O objetivo é que os alunos encontrem os erros na deteção da cor, seguimento de linha ou parede de forma mais eficaz.

Ao utilizar a imagem como referência e após a realização de medições à pista foi possível desenvolver um desenho tridimensional aproximado da pista real, conforme apresenta a Figura 4.6.

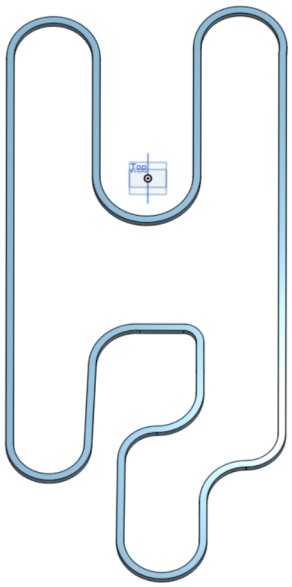


Figura 4.6: Pista 3D desenvolvida no software Onshape

Os blocos com as cores a serem detetadas serão criadas no próprio Webots. Para introduzir a pista no mundo virtual, em primeiro lugar, foi aumentada a arena, pelo que no projeto final apresenta-se na forma de um quadrado com três metros de comprimento nos lados. A Figura 4.7 demonstra a arena com o tamanho final.

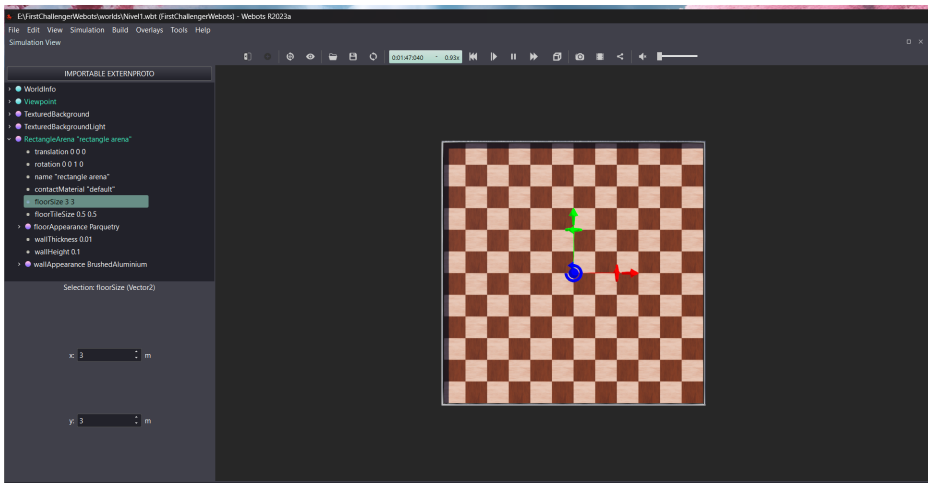
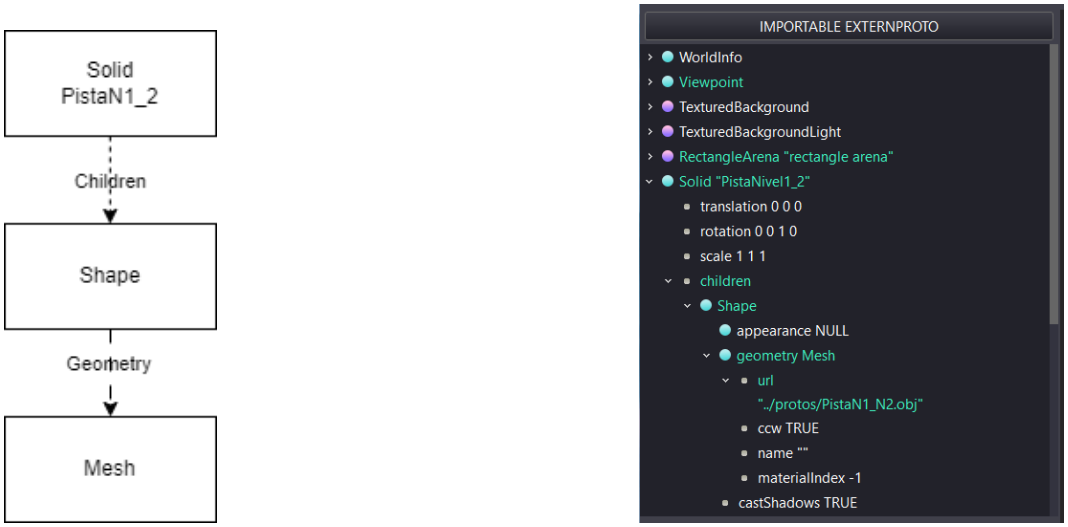


Figura 4.7: Arena quadrangular com três metros de comprimento

Após fazer a exportação do ficheiro do desenho 3D em formato .obj, este deve ser colocado numa das pastas do projeto. Para isso foi criado um novo Nó no Webots

para a introdução da pista. A sua hierarquia será um objeto sólido com o nome "PistalN1_2"que terá como dependência (*children*) uma forma (*shape*) de uma *mesh* em que a localização será o ficheiro .obj anteriormente desenvolvido. A Figura 4.8 apresenta o gráfico de dependências deste objeto e da hierarquia de Nós no projeto Webots.



(a) Hierarquia de Nós da pista

(b) Hierarquia de Nós da pista no Webots

Figura 4.8: Hierarquia de Nós para a pista

O Webots é um software que desde já contém muitos sensores que o utilizador pode usar no seu projeto. Um desses, é o sensor infravermelho que será utilizado para a deteção da pista. Como apresentado no Capítulo 3, os robôs dos alunos utilizam um módulo com sensores infravermelhos para detetar a pista através da reflexão da luz infravermelha. No entanto, uma das limitações que o Webots apresenta é que é um sensor utilizado para detetar a distância. Este software, como muitos outros, não consegue replicar a refração da luz, visto que necessitaria de muito poder de computação para calcular as trajetórias de todas as luzes. Logo para utilizar este tipo de sensor, a pista terá que ter um certo relevo (0.2mm) para seja possível a deteção.

Após mudar a cor da pista e centra-la na arena pode-se observar o resultado presente na Figura 4.9.

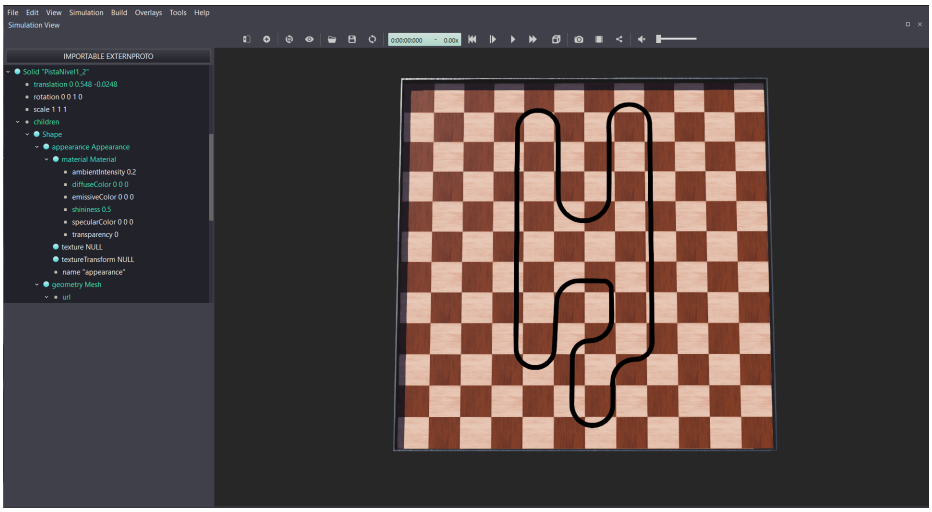


Figura 4.9: Pista First Challenger no software Webots

Para os blocos de deteção de cor o robô terá uma câmera capaz de reconhecer cores (não existe um sensor que replique o sensor de cor utilizado no robô real, mas também teria a mesma limitação que o sensor infravermelhos). Estes blocos também têm uma definição ativa de reconhecimento de uma certa cor, estão agrupados por cores e terão a hierarquia apresentada na Figura 4.10.

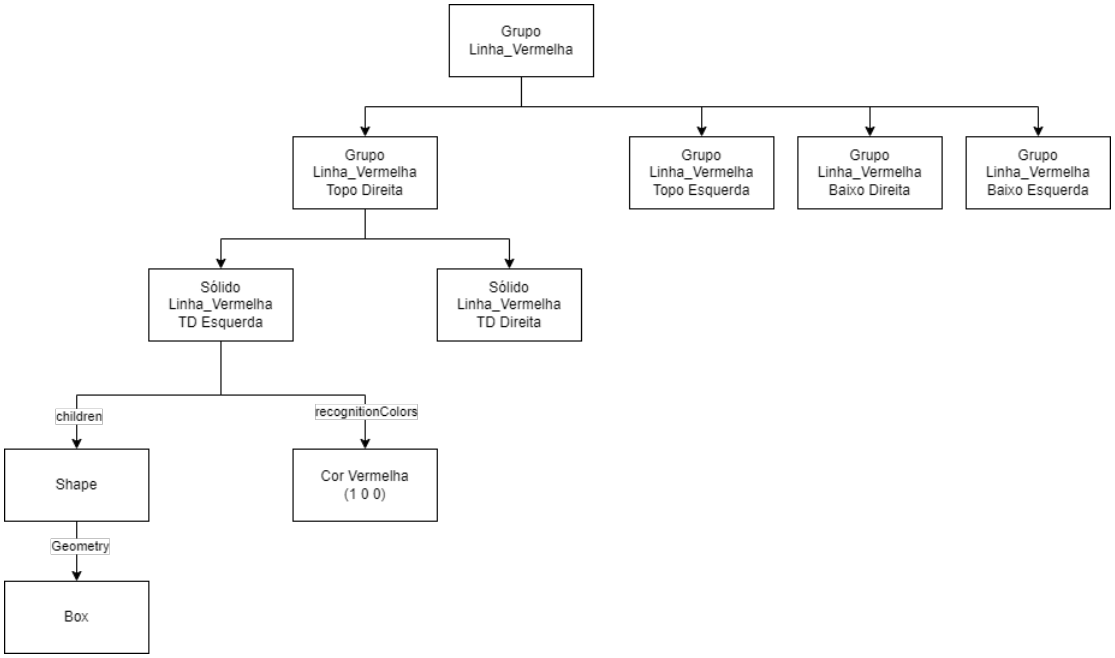


Figura 4.10: Hierarquia das linhas de cor Vermelha

Cada tira de cor terá um comprimento de 0,1m, uma largura de 0,03m e uma profundidade de 0,001m. Após as várias tiras terem sido organizadas e alteradas as respetivas cores, a pista já se encontra criada e pronta a utilizar. Esta pode ser observada na Figura 4.13.

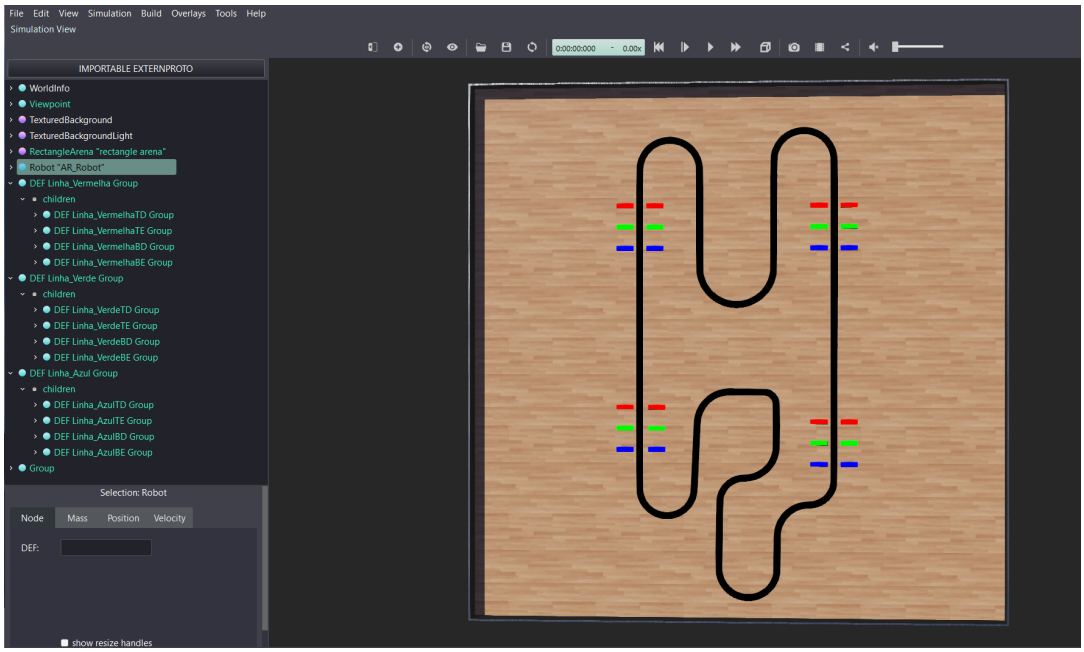


Figura 4.11: Pista Virtual prova First Challenger Nível 1 e 2

Para a pista do nível 3 foi duplicado o mundo e substituído o ficheiro .obj por um sem um troço da pista, Figura 4.12.

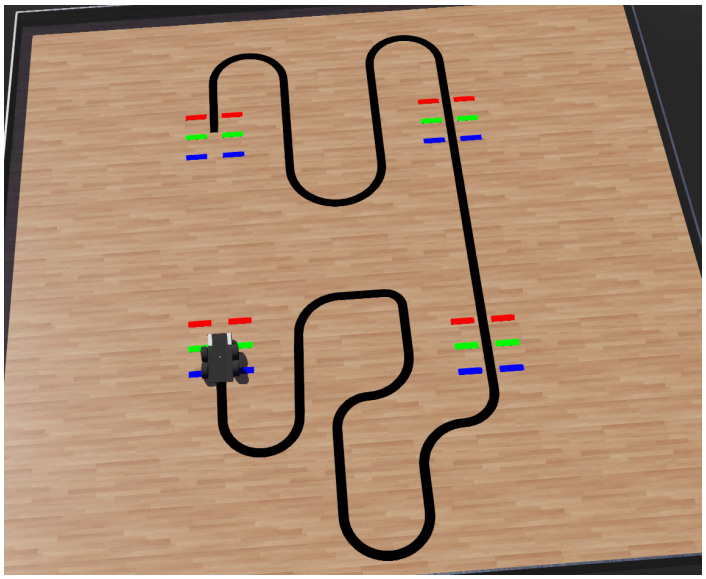


Figura 4.12: Pista Virtual prova First Challenger Nível 3 sem a parede

Para definir as paredes foram criados dois nós do tipo "Solid"e atribuída uma forma de caixa a estas. Cada uma das "shapes"foram denominadas por Parede_1 e Parede_2 para então definir o boundingObject deste sólido a cada uma destas formas, assim tornam-se objetos que podem ser detetados e colididos pelo o robô.

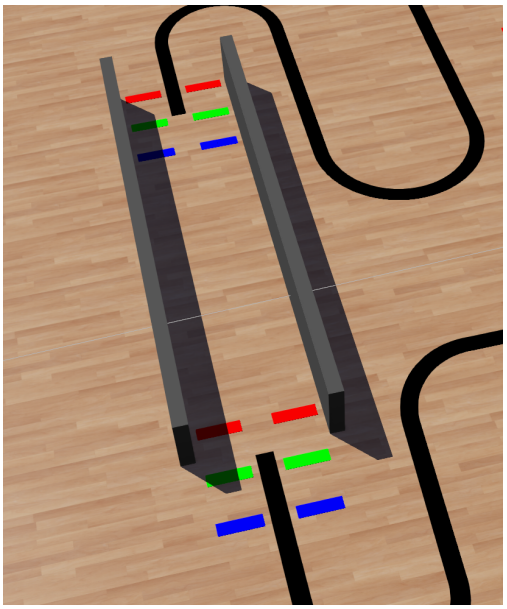


Figura 4.13: Pista Virtual prova First Challenger Nível 3

4.1.2 O Robô Virtual Academia de Robótica

Como referido no Capítulo 3, o robô que os alunos da Academia de Robótica desenvolvem contém vários sensores e atuadores para conseguirem participar em todos os níveis da Prova First Challenger. O Webots apresenta vários módulos que replicam no mundo virtual o funcionamento dos vários componentes reais a utilizar.

A Figura 4.14 apresenta um dos modelos 3D utilizado nas aulas da Academia de Robótica para auxílio da montagem do robô.

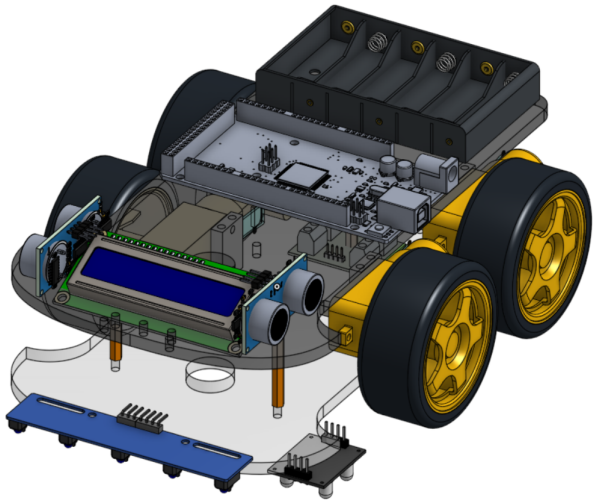


Figura 4.14: Modelo 3D do robô que os alunos montam ao longo do curso Nível 2

Para criar o robô num ambiente de simulação, em primeiro lugar, é necessário criar um novo nó de base "Robot"que tem o nome de "AR_Robot"e tem a opção "Physics"ativada. Assim serão aplicadas forças no robô como a gravidade e atrito, tornando a simulação um pouco mais realista. Para simular o robô real é necessário ter uma plataforma onde todos os componentes vão estar montados, quatro motores agrupados dois a dois em cada lado, um sensor capaz de detetar cores (camêra), um módulo de deteção de linha (três ou cinco sensores infra-vermelhos), dois sensores ultrassónicos para a deteção das paredes no nível 3 da Prova First Challenger e um LED RGB.

Sabendo os componentes que o constituem é possível então criar a árvore hierárquica, apresentada na Figura 4.15, do robô que depois será explicada detalhadamente no decorrer deste capítulo.

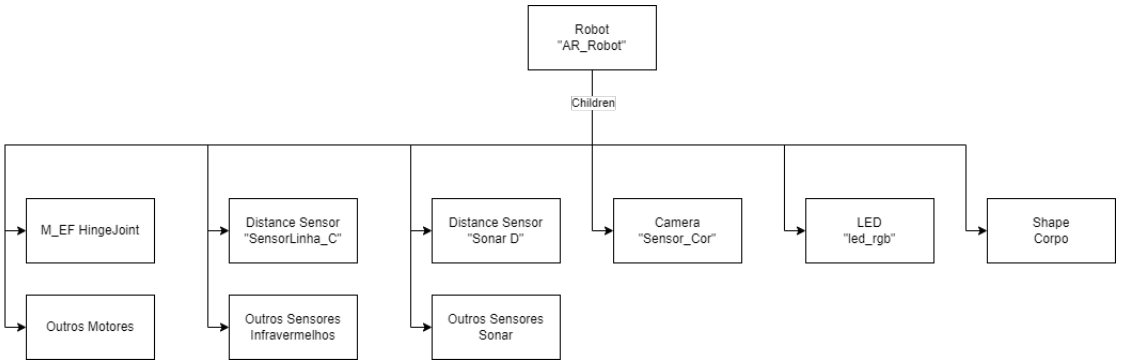


Figura 4.15: Hierarquia simplificada do Robô

Corpo do robô - Base

Para montar e posicionar todos os sensores e atuadores, o robô tem uma base "Corpo"que é um "Children"do AR_Robot. O "Corpo"vai ser definido como uma "Shape", um tipo de nó que permite criar objetos renderizados no mundo virtual, como referido anteriormente. O "Corpo"tem a hierarquia apresentada na Figura 4.16.

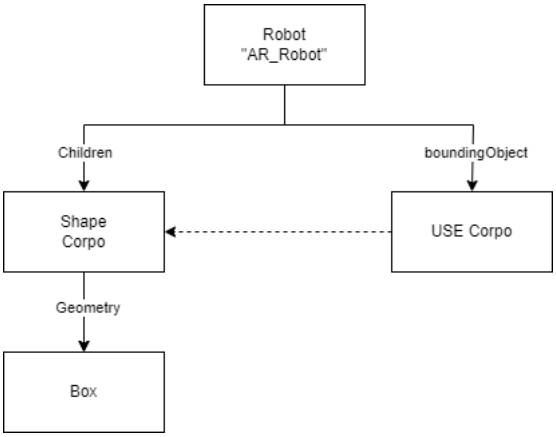


Figura 4.16: Hierarquia do corpo do Robô

Esta "Shape"vai ser um dos objetos que vai colidir com os obstáculos e igualmente definir o peso do robô para o software calcular e aplicar as forças no robô, e consequentemente calcular a inércia do mesmo. Como tal, é necessário definir o "Corpo"como boundingObject do AR_Robot, e estabelecer os parâmetros anteriormente referidos.

A base "Corpo"terá uma dimensão idêntica às bases utilizadas no robô real. A Figura 4.17 apresenta as medidas que constituem esta base.

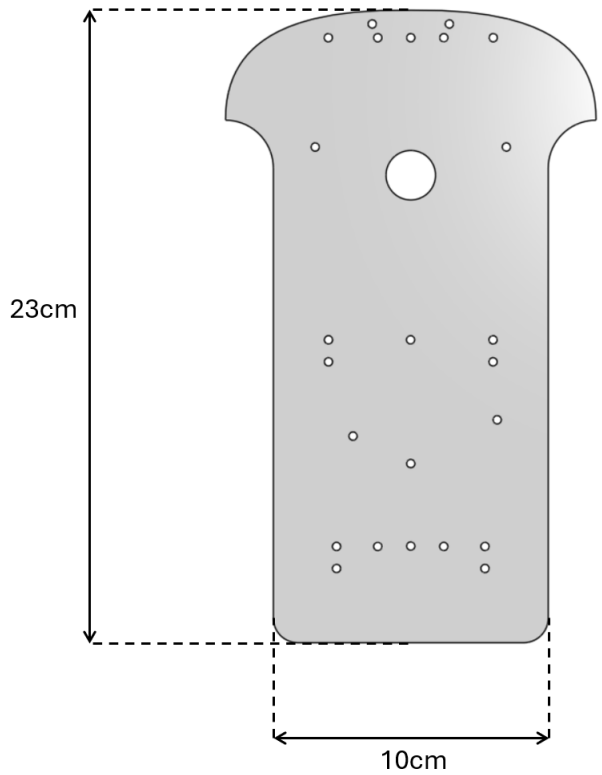


Figura 4.17: Medidas da Base do Robô Real.

Para tornar o processo mais simples e por não trazer grandes vantagens, a base do robô virtual foi definida como um simples retângulo com as medidas referidas na Figura 4.17.

Motores e rodas do robô

Com a base criada é possível definir as rodas do robô. Uma das roda é considerada uma articulação com um grau de liberdade, ou seja, irá girar em torno de um dos eixos permitindo movimentar-se para a frente ou para trás. O Webots tem a opção de criar articulações com um grau de liberdade, "HingeJoint"ou dois graus de liberdade, "Hinge2Joint".

A Figura 4.18 apresenta uma "HingeJoint"e os vários parâmetros a definir no Webots.

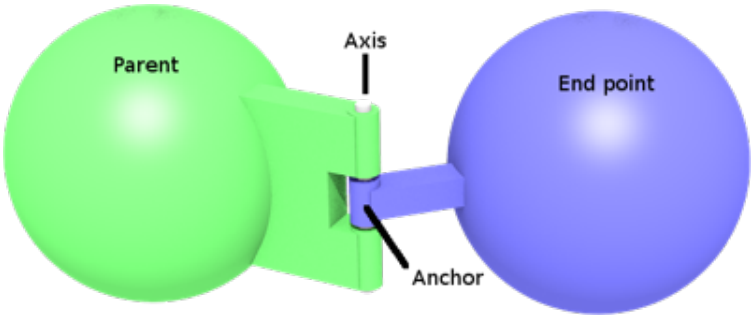


Figura 4.18: Articulação com um grau livre. [29]

No caso do robô simulado o parente é a base "corpo"do robô, a âncora é o ponto onde o veio dos motores está localizado no robô real, bem como no robô simulado. O "End point"é a roda/pneu que fará contacto com o piso e permitirá a movimentação do robô.

Ao saber os parâmetros a alterar conseguimos determinar a árvore hierárquica de uma das rodas que se encontra apresentada na Figura 4.19.

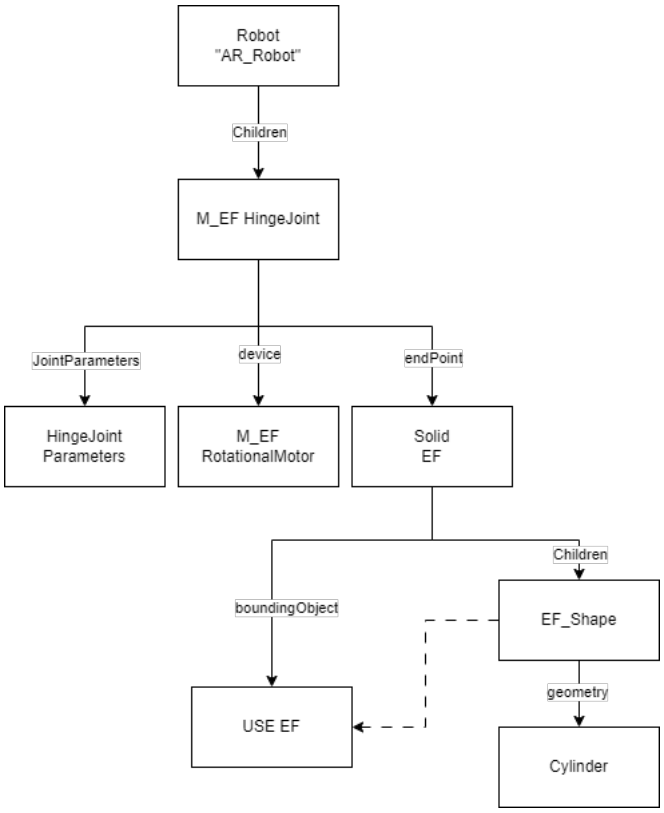


Figura 4.19: Hierarquia das rodas e motores do Robô

Na opção JointParameters estão definidas as configurações desta articulação, como a posição, o ponto de ancoragem e o eixo em que ela irá girar. A seguinte lista apresenta os diferentes parâmetros aplicados a cada um dos motores:

- **Motor Esquerdo Frontal**
  - **Posição**: 97.4
  - **Eixo**: x= 0 ; y = 1; z = 0
  - **Ponto de ancoragem**: x= 0.012m ; y = 0.05m; z = 0.015m
- **Motor Esquerdo Frontal**
  - **Posição**: 97.4
  - **Eixo**: x= 0 ; y = 1; z = 0
  - **Ponto de ancoragem**: x= -0.065m ; y = 0.05m; z = 0.015m
- **Motor Direito Frontal**
  - **Posição**: 97.4
  - **Eixo**: x= 0 ; y = 1; z = 0
  - **Ponto de ancoragem**: x= 0.012m ; y = -0.05m; z = 0.015m
- **Motor Direito Traseiro**
  - **Posição**: 97.3
  - **Eixo**: x= 0 ; y = 1; z = 0
  - **Ponto de ancoragem**: x= -0.065m ; y = -0.05m; z = 0.015m

O "device"a utilizar é um motor de rotação denominado "Rotational Motor"que permite criar uma força capaz de girar as rodas que estão acopladas ao mesmo. Na opção endPoint estão definidos todos os parâmetros para definir a roda.

A Figura 4.20 apresenta a profundidade e o diâmetro da roda (pneu mais jante) do robô real.

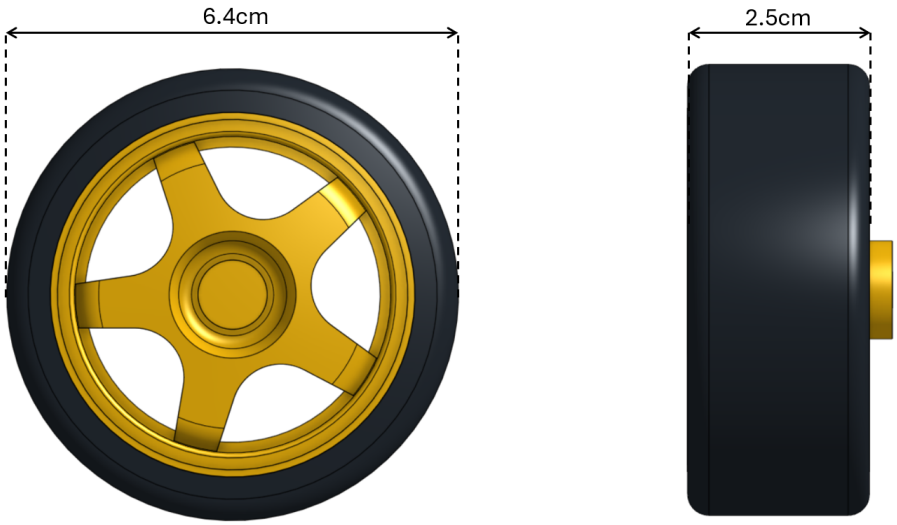


Figura 4.20: Profundidade e diâmetro da roda utilizada no robô da Academia de Robótica

Como este objeto irá estar a colidir com a arena é necessário ativar a opção "physics"e definir o boundingObject como o sólido criado anteriormente, de forma ao software saber qual é a área que este tem.

Para definir a velocidade máxima que este motor pode alcançar teremos de converter o máximo de rotações por minuto em radianos por segundo para isso:

$$V\_rad/s = RPM \times \frac{2pi}{60} \tag{4.1}$$

$$V\_rad/s = 140 \times \frac{2pi}{60} \tag{4.2}$$

$$V\_rad/s = 14.66rad/s \tag{4.3}$$

Após colocar as várias rodas no corpo do robô com as mesmas localizações do robô real e definida para cada uma a velocidade máxima, foi obtido o robô presente na Figura 4.27.

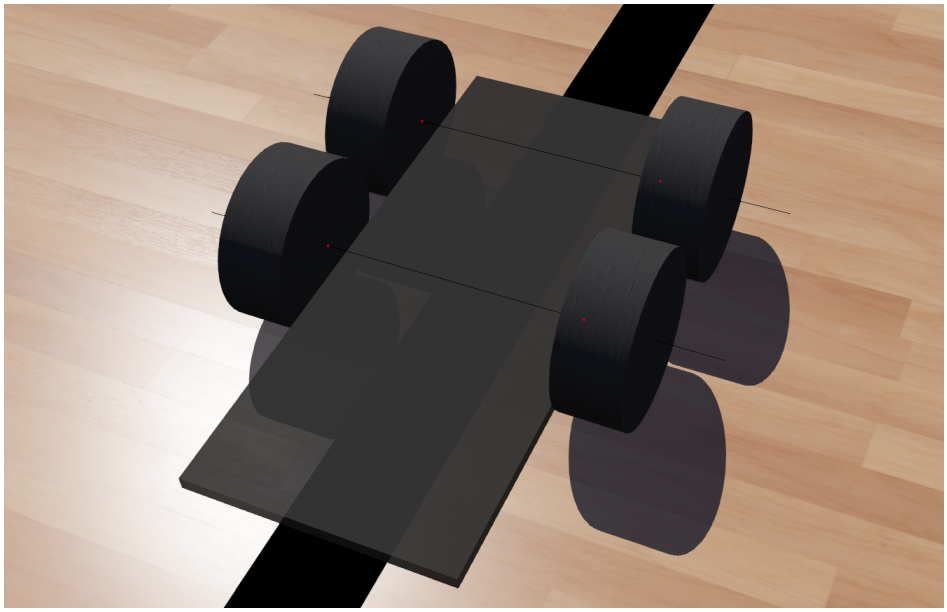


Figura 4.21: Robô AR_Robot com a plataforma e motores/roda

Na Figura 4.27 é possível observar os pontos de ancoragem a vermelho para cada uma das rodas, bem como o eixo (linhas pretas que originam dos pontos vermelhos) em que as rodas vão girar.

Sensores Infravermelhos do módulo de deteção da pista

O próximo componente a introduzir no robô serão os sensores infravermelhos para a deteção da pista. Como visto no Capítulo 3, os alunos da Academia de Robótica utilizam um módulo com três ou mais sensores infravermelhos, por isso serão criados três sensores no simulador para alcançar este efeito. Estes sensores são utilizados para medir distâncias, mas não pelo o mesmo fenómeno que os robôs reais devido às limitações identificadas anteriormente.

Estes componentes seguem a árvore hierárquica da Figura 4.22.

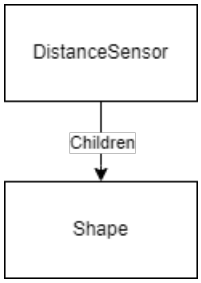


Figura 4.22: Hierarquia de nós para o sensor infravermelhos.

Em primeiro lugar, ao criar um sensor de distância este pode ser de quatro tipos diferentes: genérico, infravermelhos, sonar e laser. Após a seleção do sensor infravermelhos foi selecionada a forma de cápsula para representá-lo. Os parâmetros para este nó estão apresentados na próxima lista:

- **Forma**
  - **Geometria**: Cápsula
    - Altura: 0.003
    - Raio: 0.001
- **Tipo: Infravermelhos**
- **Tabela *LookUp 1*** x = 0.001 y = 1 z = 0
- **Tabela *LookUp 2*** x = 0.03 y = 30 z = 0

A Tabela LookUp é onde o utilizador define as distâncias mínimas (tabela 1) e máximas (tabela 2) que o sensor deve detetar. O sensor TCRT5000 tem uma distância máxima de 30mm e mínima de aproximaamente 1mm. Na primeira linha da tabela foi definda a distância mínima e o valor a devolver ao utilizador. Na segunda linha aplicam-se os mesmos critérios, mas para a distância máxima. Este sensor foi duplicado duas vezes e, assim os três sensores foram colocados na mesma posição em que o módulo se apresenta no robô real, espaçados com uma distância de 2,1 cm, que pode ser confirmada pela Figura 4.23.

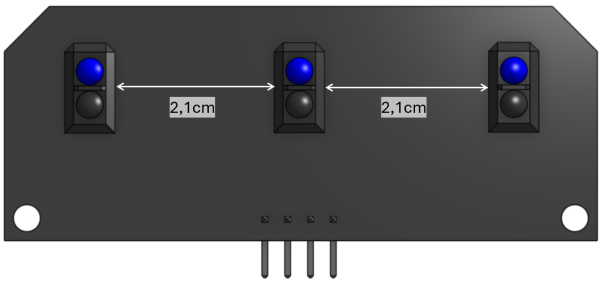


Figura 4.23: Distância dos sensores TCRT5000 no módulo.

O posicionamento destes sensores são:

- **Sensor IR Esquerdo**
  - Translação x = 0.1 y = 0.021 z = 0.003
  - Rotação x = 0 y = 1 z = 0 ângulo = 1.57

- **Sensor IR Centro**
  - Translação x = 0.1 y = 0 z = 0.003
  - Rotação x = 0 y = 1 z = 0 ângulo = 1.57
- **Sensor IR Direito**
  - Translação x = 0.1 y = -0.021 z = 0.003
  - Rotação x = 0 y = 1 z = 0 ângulo = 1.57

Importante realçar que é necessário dar um nome a estes nós referentes à posição em que se encontram. Na Figura 4.24 pode-se observar os três sensores de distância infravermelhos presentes no robô virtual, bem como as linhas que indicam o máximo de distância captada.

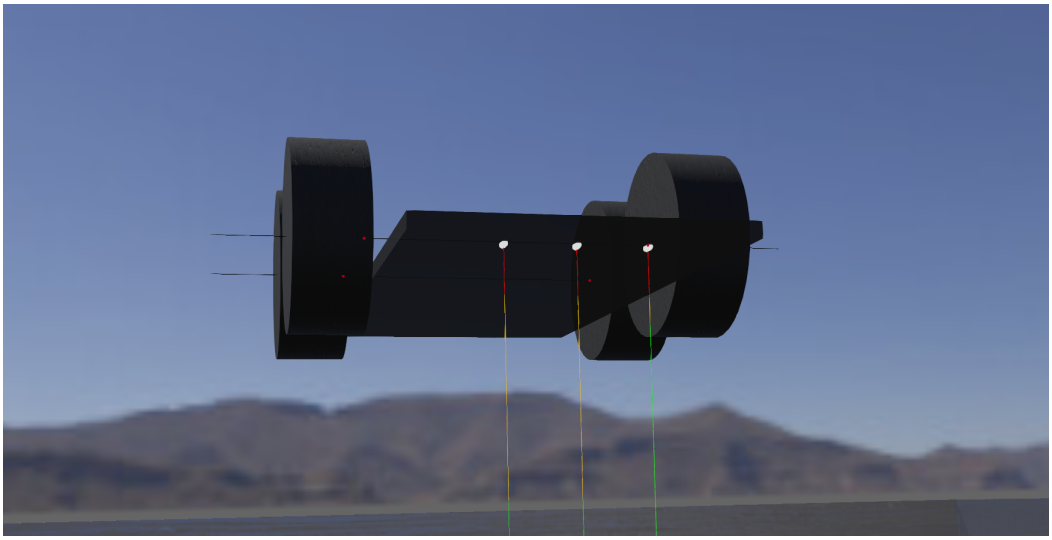


Figura 4.24: Robô AR_Robot com a plataforma e motores/roda

Sensor de Cor - Câmera com deteção de cores

Para a deteção dos blocos com cores o robô encontra-se equipado com um sensor capaz de detetar os valores RGB emitidos. No Webots não existe um sensor que efetue a simulação do mesmo, mas é possível utilizar uma câmera capaz de reconhecer as cores dos objetos. Estes valores são introduzidos pelo o utilizador para cada um dos blocos nunca variando com as diferenças de luz do ambiente simulado.

A câmera tem que ter o parâmetro "recognition"ativado para que consiga detetar as cores que foram colocadas nos diferentes blocos.

Os parâmetros definidos para este sensor são:

- **Forma**
  - **Geometria**: Caixa
    - Altura: x = 0.001 y = 0.02 z = 0.01
- **Localização** : x = 0.07 y = -0.06 z = 0

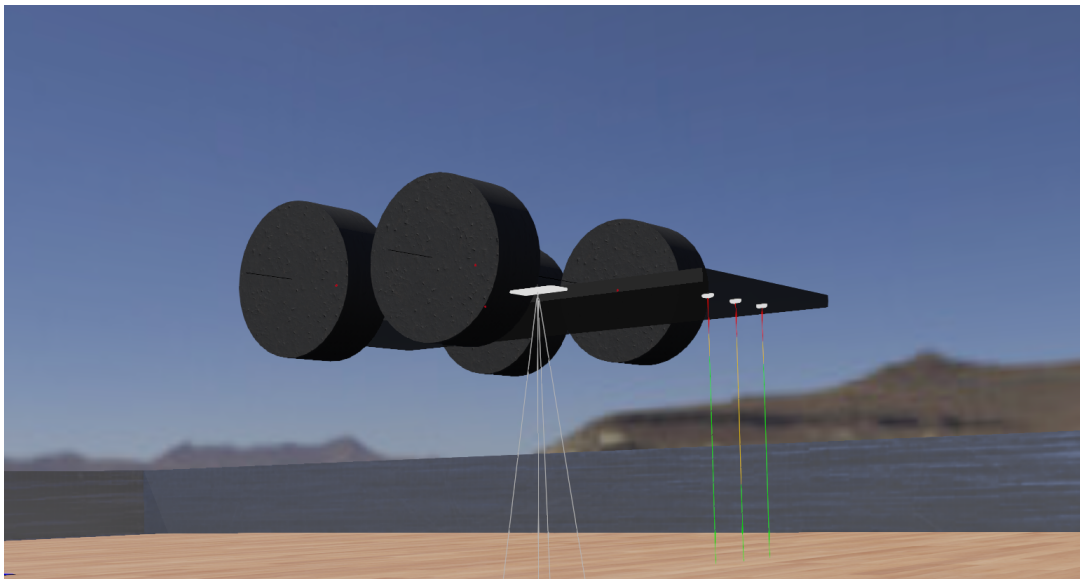


Figura 4.25: Robô AR_Robot com os vários sensor Infravermelhos.

LED RGB

Durante as aulas de montagem e desenvolvimento do robô, os alunos têm a liberdade para escolher onde e como posicionar o LED RGB, logo a sua localização no robô virtual não precisa de ser totalmente exata. Este LED tem que estar bem visível, por isso apresenta-se no centro da base "Corpo"do gémeo digital.

O Webots contém um nó que modela um LED e que é derivado de um nó sólido, apresentado na árvore hierárquica da Figura 4.26.

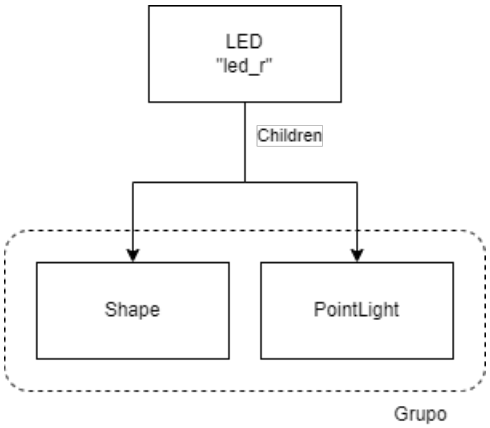


Figura 4.26: Hierarquia Nó LED

Neste tipo de componente, o primeiro nó dependente será aquele que terá a cor alterada consoante a cor do LED. Neste caso específico foi escolhido como nós dependentes uma forma cápsula e uma fonte de luz, um nó Ponto de Luz, em inglês *Point Light*. Este representa um ponto que vai emitir luz em todas as direções onde o utilizador o colocar. Para este caso, foi escolhida como primeiro nó dependente a forma "cápsula"para que esta mude de cor e torne ainda mais visível o efeito pretendido. Os parâmetros que foram definidos para a cápsula apresentam-se na seguinte lista:

- **Forma**
  - **Aparência**: PBRAppearance
  - **Geometria**: Cápsula
    - Altura: 0.001
    - Raio: 0.005
- **Ponto de Luz**
  - Atenuação: x = 0 y = 0.1 z = 0.1
  - Cor: 0, 0, 0
  - Intensidade: 0.3

Ao ser aplicado uma translação no eixo y e z torna mais notável a presença da luz quando ativado o LED, pois irá apresentar um foco de luz na lateral do robô. Este fenómeno está apresentado na Figura 4.27.

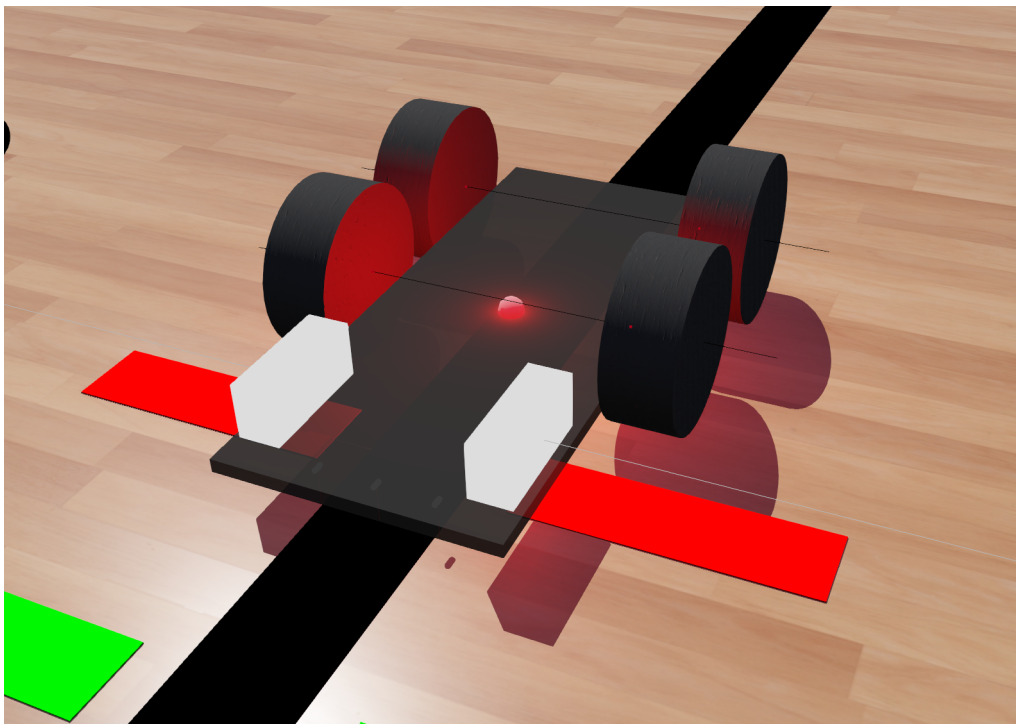


Figura 4.27: Nó LED

Sensor Ultrassónico

O último componente a introduzir é o sensor de distância para deteção das paredes, no qual será utilizado o mesmo nó que o do sensor infravermelho para deteção da pista, mas com uma configuração diferente. O Webots contém uma opção para simular um sensor ultrassónico, opção essa que será utilizada neste nó. A árvore hierárquica deste componente está representada na Figura 4.28.

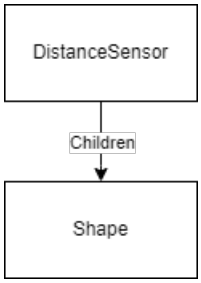


Figura 4.28: Hierarquia de nós para o sensor infravermelhos.

Este nó foi definido ligeiramente diferente do que o dos sensores infravermelhos, uma vez que aumenta a área que o robô tem e pode influenciar as colisões com as

paredes. Assim foi definido como um sólido que tem como nó dependente o sensor, que por sua vez tem como dependente a forma que o vai definir. A forma escolhida foi a de um paralelepípedo com comprimento, largura e profundidade idênticas ao sensor real. Ao sólido é possível então indicar que esta forma é o boundingObject e assim dar volume extra ao robô. Como parâmetros foram definidos na lookUp table os valores máximos e mínimos que este sensor consegue detetar. Como referido no Capítulo 3, este deteta distâncias entre os 2cm e os 400cm, acima deste valor o sensor é considerado não fiável, por isso os parâmetros da lookUp Table são:

- **LookUpTable**
  - **Valor mínimo**: x = 0.02 y = 2 z = 0
  - **Valor máximo**: x = 4 y = 400 z = 0

A biblioteca da Academia para o controlo dos sonares retorna o valor da distância em centímetros. Como o Webots permite relacionar o que deteta e a saída que apresenta ao utilizador, foi já feita a conversão de metros para centímetros na LookUp Table. O outro campo importante a alterar é a abertura ou ângulo de leitura deste sensor. O real apresenta uma abertura de mais ou menos 30º que em radianos é 0,52. Os restantes parâmetros definidos foram:

- **SONAR Esquerdo**
  - **Abertura**: x = 0,52
  - **Localização**: x = 0,04 y = 0,08 z = 0,013
- **SONAR Direito**
  - **Abertura**: x = 0,52
  - **Localização**: x = 0,04 y = -0,08 z = 0,013
- **Forma**
  - **Geometria**: Caixa
  - **Localização**: x = 0,015 y = 0,045 z = 0,02

Na Figura 4.29 pode ser visualizado a localização dos sensores ultrassónicos do robô digital, assim como as retas que indicam a direção de cada sensor ao utilizador.

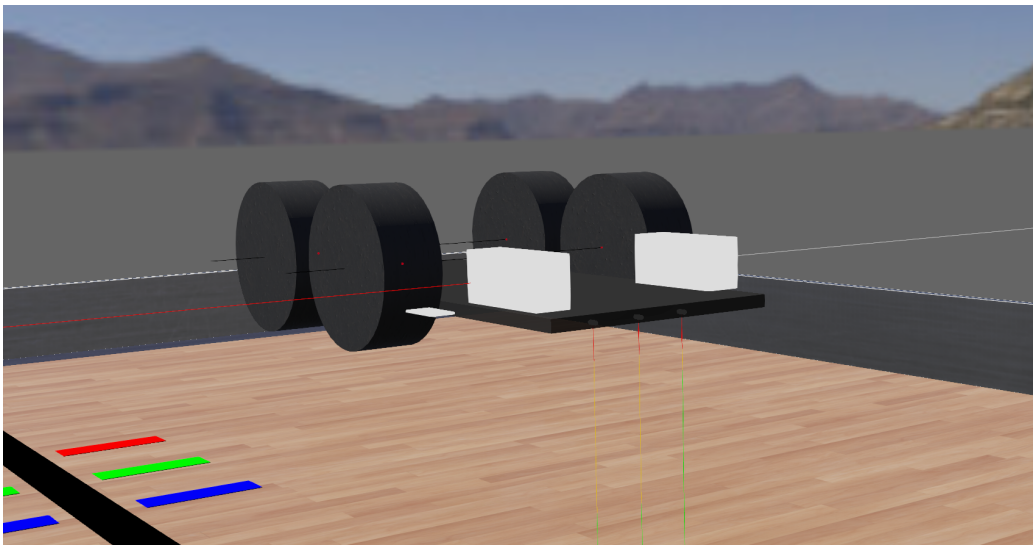


Figura 4.29: Robô final com todos os componentes.

4.2 Bibliotecas e algoritmos

Cada um dos sensores e atuadores recriados no ambiente virtual que pertencem ao robô podem ser considerado como um dos módulos fundamentais lecionados durante o curso da Academia de Robótica. Como já foi referido anteriormente neste relatório, o grande objetivo do curso é que o aluno detenha conceitos de programação e eletrónica, mas mais importante ainda, que compreenda o processo de como um programador aplica a lógica para resolver um determinado problema. Por isso, o aluno aprende, por exemplo, a lógica para controlar e obter valores do sensor ultrassónico e depois todos os outros sensores e atuadores têm bibliotecas fornecidas pela Academia. O procedimento é o aluno desenvolver e experimentar o código primeiramente no simulador e de seguida no robô real, por isso tem de ser o mais intuitivo possível e não conter passos desnecessários. As bibliotecas e o código têm que ser totalmente iguais para permitir simplesmente copiar do editor de texto do Webots para o editor de texto do Arduino IDE e vice-versa. É fundamental que o processo seja bidirecional, pois o aluno pode detetar algum erro no seu robô real ou no seu algoritmo e querer experimentá-lo na simulação para o resolver. Outro aspeto importante de realçar, especialmente nos primeiros níveis da Academia, é que ao ser um processo bidirecional é possível desenvolver o código no mBlock, copiar/colar no simulador e carregá-lo no robô do aluno.

O Webots utiliza um ficheiro de configuração denominado Makefile que é utilizado na compilação de ficheiros em linguagem de programação C/C++. Este é criado automaticamente caso o software não o detete, sendo portanto imprescindível para a compilação do código. A Makefile especifica uma lista de ficheiros de origem e bibliotecas que o controlador/código necessitará para compilar. Numa primeira fase foram identificadas as bibliotecas utilizadas na Academia e posteriormente indicadas no ficheiro Makefile do projeto. Para podermos proceder com o preenchimento da Makefile é necessário adicionar um controlador ao projeto. Para isso, é preciso criar um novo controlador, indicado na Figura 4.30.

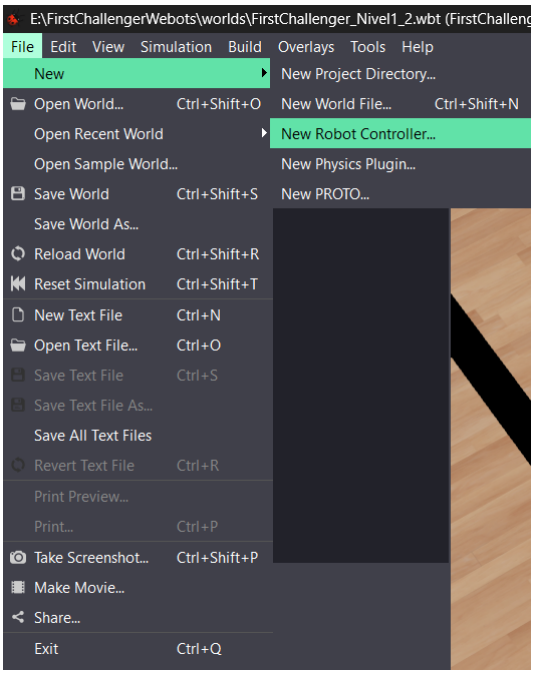


Figura 4.30: Opções no Webots para criar um novo controlador.

Este controlador vai ser compilado na Linguagem de Programação C, com o IDE do Webots e com o nome de "FirstChallenger_N1_N2". Estes passos estão indicados na Figura 4.31.

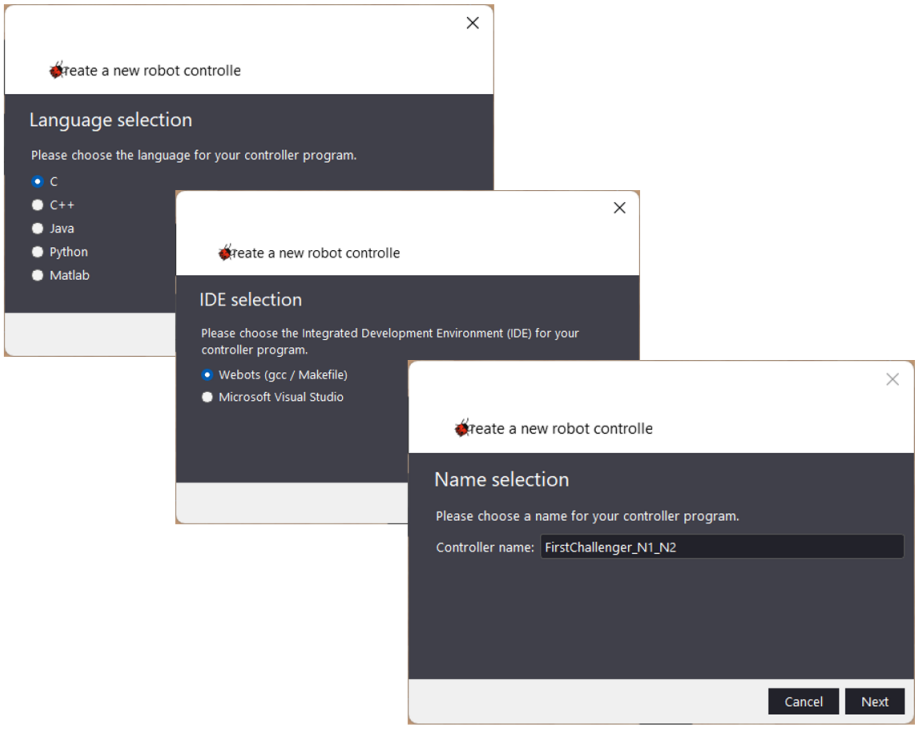


Figura 4.31: Procedimento para criar um controlador em C.

Este ficheiro acabado de criar localiza-se na pasta "Controllers"do mundo e é também nele que vão ser criados os ficheiros de origem das bibliotecas, bem como os ficheiros de cabeçalho, em inglês *Header* que vão ser explorados adiante.

Ao eliminar os vários comentários ao longo do ficheiro de controlo foi obtido o seguinte código:

```
1     #include <webots/robot.h>
2     #define TIME_STEP 64
3
4     int main(int argc, char **argv) {
5
6         wb_robot_init();
7
8
9         while (wb_robot_step(TIME_STEP) != -1) {
10
11        };
12
13
14        wb_robot_cleanup();
15
16    return 0;
17 }
```

Listagem 4.1: Código base do controlador Webots.

A primeira linha é de inclusão da biblioteca para controlo do robô simulado no Webots onde estão incluídas as funções representadas na linha 6 que inicia o robô. Esta função permite criar um ciclo infinito da linha 9 e também terminar a ligação do robô com o controlador e o código em geral é terminado, linha 14. O "TIME_STEP"indica o tempo que os vários sensores/atuadores e o código inteiro levam a ser atualizado. Neste caso foi utilizado o valor predefinido de 64 milissegundos de taxa de atualização. O mBlock utiliza o Arduino IDE para compilar o código, por isso tem de ser aplicado um método rápido de transição entre este e o Webots. Ao iniciar um novo projeto para programar o Arduino o utilizador é presenteado com o seguinte código:

```
1     void setup(){
2
3     }
4
5     void loop(){
6
7     }
```

Listagem 4.2: Procedimentos base idênticos ao do Arduino IDE

Como não é pretendido que o aluno interfira ou apague, no código do ciclo infinito

do Webots foram aplicadas as funções iguais às do Arduino IDE e, respetivamente as chamadas no código principal do controlador, conforme demonstra o código abaixo:

```
1   #include <webots/robot.h>
2   #define TIME_STEP 64
3
4   //Includes
5
6   //Procedimentos e variaveis
7
8
9   void setup(){
10
11
12  }
13
14  void loop(){
15
16
17  }
18
19
20  int main(int argc, char **argv) {
21
22      wb_robot_init();
23
24      setup();
25
26
27  while (wb_robot_step(TIME_STEP) != -1) {
28
29      loop();
30
31  };
32
33  wb_robot_cleanup();
34
35  return 0;
36  }
```

Listagem 4.3: Chamada dos procedimentos nos respetivos locais no controlador Webots.

As linhas 4 e 6 contêm os comentários a indicar as zonas onde os alunos terão que colocar as linhas de código que vêm antes da configuração. Estes comentários não são fundamentais, mas poderão ser muito úteis durante a aula, principalmente no que toca a melhorar a exposição e a explicação de como o aluno deve passar o código ou no caso de estar a realizar diretamente no Webots, orientá-lo como deve

programar para não cometer nenhum erro.

Um dos blocos e consequentemente o procedimento muito utilizado durante as aulas de robótica é o bloco de espera, em inglês *Delay*. O Webots não apresenta nenhuma função predefinida para colocar o robô/controlador em espera, por isso foi criada uma igual ao procedimento do mBlock e do Arduino (ambos são diferentes). A forma mais intuitiva de desenvolver uma função com esta especificidade é aplicando a função de "While"para fazer com que o código aguarde que o tempo atual seja igual ao tempo em que o procedimento foi chamado mais os segundos/milissegundos que o utilizador decidiu.

O diagrama de blocos da Figura 4.32 apresenta o raciocínio lógico para alcançar este objetivo.

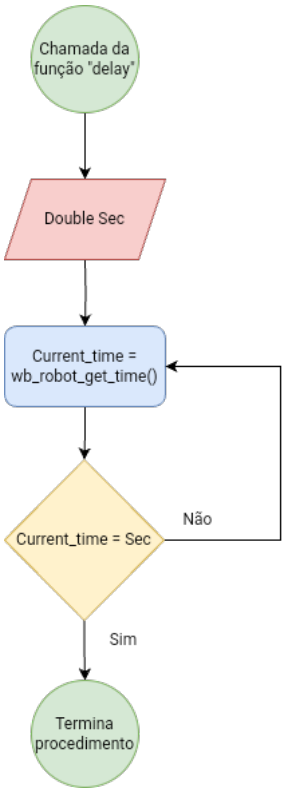


Figura 4.32: Fluxograma da função de espera.

No mBlock o procedimento de *Delay* é denominado por "_delay"e o utilizador coloca o tempo em segundos, enquanto que no Arduino IDE o procedimento é denominado por "delay"e recebe como entrada o tempo em milissegundos.

Logo foram criadas as duas funções com a mesma nomenclatura no controlador:

```
1     //Procedimentos de espera
2     void delay(double millisec);
3     void _delay(double sec);
```

```
1  void delay(double millisec){
2
3    double current_time = wb_robot_get_time();
4
5    double sec = millisec / 1000;
6
7    double time = current_time + sec;
8
9    while (current_time <= time){
10
11     current_time = wbrobot_get_time();
12     wb_robot_step(TIME_STEP);
13
14   }
15 }
16
17 void _delay(double sec){
18
19   double current_time = wb_robot_get_time();
20
21   double time = current_time + sec;
22
23   while (current_time <= time){
24
25     current_time = wb_robot_get_time();
26     wb_robot_step(TIME_STEP);
27
28   }
29 }
```

Listagem 4.4: Procedimento de Espera idêntico ao do Arduino IDE e do mBlock

É possível observar nas linhas 9 e 23 que o controlador não vai realizar nenhuma ação no robô simulado enquanto aquela condição não for alcançada. As duas linhas de código presentes neste ciclo "while"simplesmente atualizam o tempo da simulação para efeitos de comparação.

Como referido anteriormente, os alunos na Academia de Robótica utilizam blocos e procedimentos/funções de bibliotecas externas ao código principal. Para aplicar

o mesmo mecanismo no Webots é necessário indicar na Makefile todos os ficheiros ".c"que irão ser necessários para compilar o código executado.

Ao abrir pela a primeira vez a Makefile o utilizador é presenteado, Figura 4.33, com múltiplas linhas de código comentadas e um procedimento que não deve ser apagado (comandos para compilação das bibliotecas de origem do simulador).

Figura 4.33: Makefile gerada pelo o software Webots.

De seguida é preciso descomentar a linha que vai permitir compilar múltiplos ficheiros de origem em C e colocar os diferentes ficheiros a utilizar. O seguinte código apresenta a introdução nesta linha de código, a indicação que o controlador terá que compilar o ficheiro principal "FirstChallenger_N1_N2.c", a biblioteca do Arduino, que tem as funções para controlar os pinos digitais predefinidos, e a primeira biblioteca da Academia a implementar, "ARSC_Motores.c".

```
1    ### ---- C Sources ----
2    ### if your program uses several C source files:
3    C_SOURCES = FirstChallenger_N1_N2.c Arduino.c ARSC_Motores.c
```

Listagem 4.5: Linhas de configuração no ficheiro Makefile do controlador Webots

Uma biblioteca em C é constituída por dois ficheiros: um de origem, em inglês, *Source*, e um de cabeçalho, em inglês, *Header*. Estes têm a extensão '.c' e '.h', respetivamente. No ficheiro *Source* estarão todas as funções e procedimentos que o utilizador poderá utilizar daquela biblioteca, bem como todas aquelas que são necessárias para essa biblioteca funcionar. Já o ficheiro *Header* indicará o nome das funções e procedimentos que o utilizador poderá chamar no seu código. O

método aplicado para programar os vários componentes será diferente entre ambos os mundos, o real e o virtual, por isso não é crucial explorar como estes funcionam, mas sim conhecer os nomes dos procedimentos de cada biblioteca e, no caso das funções, é fundamental não só conhecer o nome, mas também saber o que estas retornam e como.

4.2.1 Biblioteca Arduino

A primeira biblioteca a ser desenvolvida terá procedimentos originais do Arduino para controlo dos LEDs e do Buzzer. Os alunos não utilizam uma biblioteca predefinida para o controlo destes, pois só necessitam de ligar ou desligar os componentes em determinadas situações. Os alunos aprendem este processo logo nas primeiras aulas e acabam por utilizar o procedimento de "digitalWrite", que ativa ou desativa os pinos digitais, para controlar o estado dos componentes referidos. Este pode ser enviado como um número inteiro, um para ligado e zero para desligado, ou pode ser introduzido a palavra "HIGH"e "LOW"que correspondem a um e zero, obtendo o mesmo resultado.

```
1 #include "Arduino.h"
2 #include <webots/led.h>
3 #include <webots/speaker.h>
4 #include <stdio.h>
5
6 #define HIGH 1
7 #define LOW 0
8 #define OUTPUT 1
9 #define INPUT 0
```

Listagem 4.6: Inclusão das bibliotecas necessárias para a Biblioteca Arduino

Função `DigitalWrite`

A função do "digitalWrite"é um procedimento que dependendo do pino chamado e do estado indicado liga a cor pretendida no LED RGB e também liga ou desliga o próprio Buzzer.

O fluxograma da Figura 4.40 apresenta o código de "digitalWrite"definido no Webots.

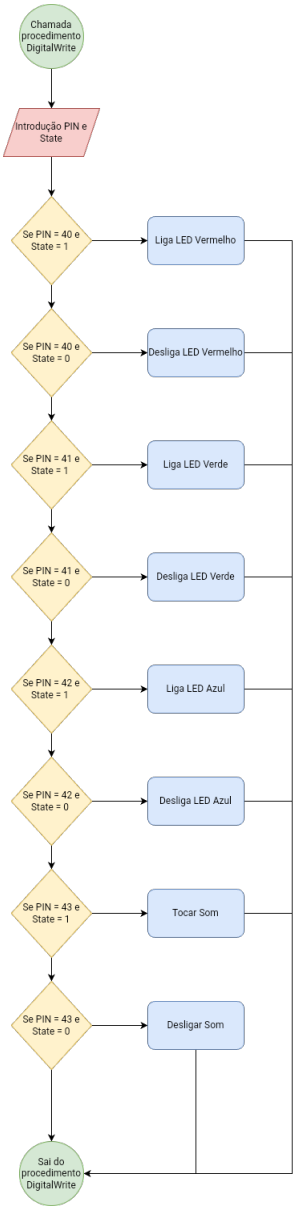


Figura 4.34: Fluxograma do procedimento DigitalWrite.

O código seguinte é referente ao fluxograma da Figura 4.40 :

```
void digitalWrite (int Pin, int State){

  WbDeviceTag led_rgb = wb_robot_get_device("led_r");
  WbDeviceTag buzzer = wb_robot_get_device("Buzzer");

  if (Pin == 40 && State == 1){
    wb_led_set(led_rgb, 0xff0000);
  }
   if (Pin == 40 && State == 0){
    wb_led_set(led_rgb, 0x000000);
  }
  if (Pin == 41 && State == 1){
    wb_led_set(led_rgb, 0x00ff00);
  }
  if (Pin == 41 && State == 0){
    wb_led_set(led_rgb, 0x000000);
  }
  if (Pin == 42 && State == 1){
    wb_led_set(led_rgb, 0x0000ff);
  }
  if (Pin == 42 && State == 0){
    wb_led_set(led_rgb, 0x000000);
  }
```

Listagem 4.7: Biblioteca Arduino.h com os procedimentos para controlo dos LEDs

O aluno para controlar o LED Vermelho deve chamar o pino 40, para o LED Verde o pino 41 e, por fim para o LED Azul o pino 42.

Para controlo do buzzer encontra-se o pino 43, no qual o aluno ao ativá-lo será chamada a função "wb_speaker_play_sound"que necessita de entradas no dispositivo para o canal da esquerda de som, o dispositivo para o canal da direita de som (neste caso será o mesmo dispositivo), o som a tocar (pode ser qualquer som), a frequência e o volume.

```
1 if (Pin == 43 && State == 1){
2     wb_speaker_play_sound(buzzer, buzzer, "Sounds/BEEPBUZZER.wav",
          1.0, 1.0, 0.0,       true);
3      printf("A Tocar \n");
4
5   }
6   if (Pin == 43 && State == 0){
7     wb_speaker_stop(buzzer,NULL);
8     printf("Desligado \n");
9   }
```

Listagem 4.8: Procedimento de controlo do Buzzer.

Para evitar erros e como no código Arduino é necessário a função pinMode, que define se é entrada ou saída, esta foi definida sem qualquer tipo de instrução. Assim o aluno pode chamá-la, mantendo a lógica do microcontrolador e não obter qualquer tipo de erro.

```
1
2 void pinMode(int Pin, int Type){
3
4 }
```

Listagem 4.9: Procedimento pinMode

4.2.2 Biblioteca Controlo Motores

O robô real tem quatro motores que estão agrupados dois a dois, no lado esquerdo e no lado direito. No Webots vai ser necessário agrupar os motores da mesma forma, portanto serão programados procedimentos que irão controlar os dois motores, de cada lado, em simultâneo.

Na biblioteca real os alunos utilizam os seguintes procedimentos e blocos da Figura 4.35.

Figura 4.35: Blocos de código para controlo dos motores no mBlock.

```
1    void AR_InitMotores(int AR_PWM_D, int AR_Dir_D, int AR_PWM_E,
         int AR_Dir_E);
2    void AR_MotorDir(unsigned int pwm, int Dir);
3    void AR_MotorEsq(unsigned int pwm, int Dir);
```

Listagem 4.10: Procedimentos do ficheiro .h da biblioteca de controlo dos motores

O primeiro procedimento tem como função inicializar e indicar os pinos utilizados para o controlo do motor. No robô simulado estes pinos não existem, mas como são fundamentais para que a transição seja o mais suave possível (o aluno basicamente só copia e cola) estas constantes têm também que ser definidas, mesmo que não sejam utilizadas em nenhum aspeto relevante para o código no simulador. Já o segundo e terceiro procedimento são aqueles que o aluno utiliza para controlar os motores. O procedimento permite controlar a velocidade (variável "pwm") e a direção com que os motores giraram (variável "Dir"). Sabendo quais os procedimentos necessários já é possível criar o ficheiro *Header*.

```
1    #ifndef ARSC_Motores_h
2    #define ARSC_Motores_h
3    #define FRENTE  1
4    #define TRAS  0
5
6    void AR_InitMotores(int AR_PWM_D, int AR_Dir_D, int AR_PWM_E,
         int AR_Dir_E);
7    void AR_MotorDir(unsigned int pwm, int Dir);
8    void AR_MotorEsq(unsigned int pwm, int Dir);
9    #endif
```

Listagem 4.11: Ficheiro header da biblioteva ARSC Motores

As primeiras duas linhas de código tem como propósito evitar a múltipla declaração do mesmo ficheiro *Header*. As linhas de "#define FRENTE 1"e "#define TRAS 0"indicam que quando estes dois termos forem utilizados no código, o valor é de um ou zero respetivamente. O procedimento de inicialização do Webots vai conter as linhas de código para referenciarem os nós dos motores e também para indicar o tipo de movimento dos mesmos, neste caso um movimento continuo sem paragens. A função "WbDeviceTag wb_robot_get_device(const char *name)"retorna um identificador para o dispositivo referido em "*name". Neste campo estarão os nomes dados aos motores ou a qualquer outro dispositivo quando criados no simulador. Para definir o motor como um com movimento contínuo é preciso utilizar a função do Webots "wb_motor_set_position(WbDeviceTag motor, double position)"com a variável "position"como "INFINITY". Assim já é possível desenvolver o procedimento de inicialização na totalidade.

```
1     void AR_InitMotores(int AR_PWM_D, int AR_Dir_D, int AR_PWM_E,
          int AR_Dir_E){
2
3         WbDeviceTag motor_ef = wb_robot_get_device("M_EF");
4         WbDeviceTag motor_et = wb_robot_get_device("M_ET");
5         WbDeviceTag motor_df = wb_robot_get_device("M_DT");
6         WbDeviceTag motor_dt = wb_robot_get_device("M_DF");
7
8         wb_motor_set_position(motor_ef,INFINITY);
9         wb_motor_set_position(motor_et,INFINITY);
10        wb_motor_set_position(motor_df,INFINITY);
11        wb_motor_set_position(motor_dt,INFINITY);
12
13 }
```

Listagem 4.12: Procedimento de configuração dos motores

Para colocar os motores a funcionar no Webots utiliza-se a função "wb_motor _set_ velocity(WbDeviceTag motor, double velocity)"que define a velocidade do dispositivo.

A 4.36 apresenta o fluxograma do procedimento para controlo dos motores.

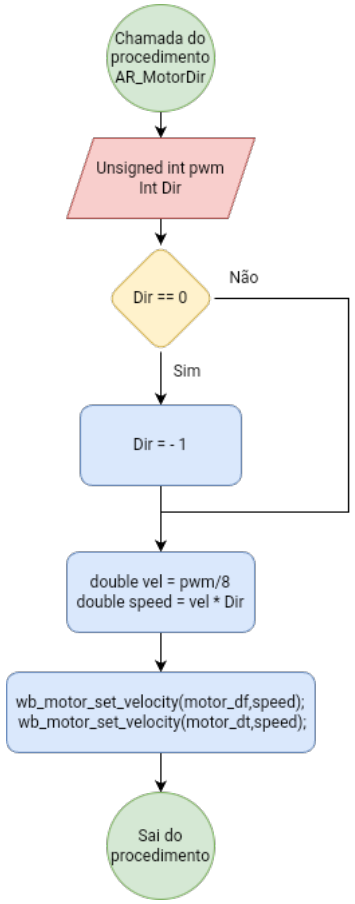


Figura 4.36: Diagrama de blocos do procedimento de controlo dos motores.

A direção do motor é definida pelo sinal da variável velocidade. Quando é negativa move-se para trás e quando é positiva move-se para a frente. Como nas funções do Arduino do robô real a direção é indicada pelo o valor "1"para andar para a frente e "0"para andar para trás, é preciso executar uma condição que indique que se o valor da direção for zero este irá mudar de "0"para -1". Como visto anteriormente, nos procedimentos para controlar o movimento dos motores o utilizador terá que introduzir o valor da velocidade e da direção.

Com estes valores é possível calcular o valor a colocar na variável "velocity".

```
1     void AR_MotorDir(unsigned int pwm, int Dir){
2
3     if (Dir == 0){
4         Dir=-1;
5     }
6
7     double vel = pwm * 14/255;
8
9     double speed = vel*Dir;
10
11    WbDeviceTag motor_df = wb_robot_get_device("M_DT");
12    WbDeviceTag motor_dt = wb_robot_get_device("M_DF");
13
14    wb_motor_set_velocity(motor_df,speed);
15    wb_motor_set_velocity(motor_dt,speed);
16
17 }
```

Listagem 4.13: Procedimento de controlo da velocidade e direção paracada conjunto de motores.

A velocidade introduzida pelos alunos está dentro do intervalo de valores zero (parado) e 255 (velocidade máxima), por isso a variável "vel"efetua o cálculo para determinar a velocidade do robô simulado, multiplicando esse valor pela a razão entre as velocidades máximas (valor 14 no Webots). Como estes são procedimentos e o código tenta ser o mais aproximado do código real, é necessário repetir o "wb_robot_get_device", pois os definidos anteriormente não são passados para fora do procedimento.

4.2.3 Biblioteca Controlo Módulo Deteção Linha

Nesta biblioteca estão os diferentes procedimentos que transformam os valores de distância de cada um dos sensores para valores binários (zero e um) e de seguida, como explorados no capitulo 3, conforme as combinações entre estes devolver o valor decimal que representa o número em binário obtido.

Nesta biblioteca, como na dos motores, tem de existir um procedimento de inicialização e um de leitura dos sensores, Figura 4.37. Neste último é onde serão realizados os cálculos para retornar o valor decimal ao utilizador.

Figura 4.37: Blocos mBlock para obtenção dos valores do módulo dd deteção de linha.

Esta biblioteca permitem ao aluno ler os vários sensores de infravermelhos para a deteção da pista. Estes sensores tanto na vida real, como no simulador têm a capacidade de detetar distâncias muito curtas. Como referido no Capítulo 3 estes módulos utilizam para a deteção da linha a indicação de presença de luz infravermelha no recetor ou não, o que no Webots não é permitido. Como ao definir a pista foi lhe dada uma pequena altura relativamente ao piso, este sensor terá que detetar as diferenças de distância e a partir desses devolver um valor de um quando deteta pista e zero quando deteta o piso.

O código da abaixo apresenta os procedimentos e variáveis declaradas no ficheiro *Header* desta biblioteca.

```
1  #ifndef ARSC_SensorDePista_h
2  #define ARSC_SensorDePista_h
3
4  void AR_InitSensorPista3(int ...);
5  void AR_LerSensorPista3();
6
7  #endif
```

Listagem 4.14: Ficheiro Header da biblioteca ARSC$_{S}ensorDePista$

Na função InitSensorPista 3 os sensores são inicializados e ativados. Os três parâmetros de entrada têm como propósito indicar os pinos em que o sensor da direita, centro e esquerda se encontram conectados no Arduino.

Em seguida obtém os dispositivos correspondentes aos sensores utilizando a função `wb_robot_get_device` e ativa-os com `wb_distance_sensor_enable`, definindo o intervalo de atualização como `TIME_STEP` e por fim, imprime os valores dos pinos de entrada.

```
1  void AR_InitSensorPista3(int IR_D, int IR_C, int IR_E) {
2    WbDeviceTag line_c = wb_robot_get_device("SensorLinha_C");
3    WbDeviceTag line_e = wb_robot_get_device("SensorLinha_E");
4    WbDeviceTag line_d = wb_robot_get_device("SensorLinha_D");
5
6    wb_distance_sensor_enable(line_c, TIME_STEP);
7    wb_distance_sensor_enable(line_e, TIME_STEP);
8    wb_distance_sensor_enable(line_d, TIME_STEP);
9
10   printf("Pino Direita %d, Pino Centro %d, Pino Esquerda %d\n",
         IR_D, IR_C, IR_E);
11 }
```

Listagem 4.15: Procedimento de configuração dos vários sensores de distância para deteção da linha

Função `AR_LerSensorPista3`

A função AR_LerSensorPista3 tem como propósito transformar os valores detetados pelos vários sensores num valor decimal. Para efetuar a transformação dos valores de distância em valores decimais, como visto na Tabela 3.1, são aplicadas várias condições que comparam as distâncias com valores de referência.

Esse processo está representado no diagrama de fluxos da Figura 4.40.

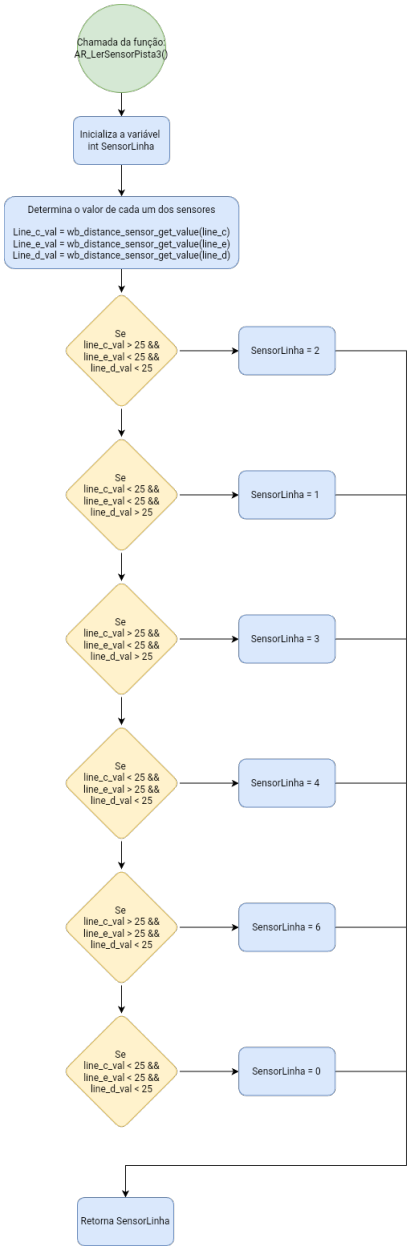


Figura 4.38: Fluxograma da função de Leitura do Modulo de deteção de linha.

Em primeiro lugar é inicializado uma variável "SensorLinha"que armazena o valor decimal que a função retorna:

```
1 int AR_LerSensorPista3() {
2   int SensorLinha;
```

Listagem 4.16: Declaração da variável SensorLinha que irá ser retornada pela função.

De seguida são chamados os vários sensores e efetua-se a leitura das distâncias de cada um.

```
1   WbDeviceTag line_c = wb_robot_get_device("SensorLinha_C");
2   WbDeviceTag line_e = wb_robot_get_device("SensorLinha_E");
3   WbDeviceTag line_d = wb_robot_get_device("SensorLinha_D");
4
5   double line_c_val = wb_distance_sensor_get_value(line_c);
6   double line_e_val = wb_distance_sensor_get_value(line_e);
7   double line_d_val = wb_distance_sensor_get_value(line_d);
```

Listagem 4.17: Configuração dos vários sensores virtuais na biblioteca SensorDeLinha

Tendo os valores das várias distâncias consegue-se aplicar uma condição que as compara com os valores de referência de menor de 25 quando deteta a linha e maior que 25 quando deteta o piso.

```
1   if (line_c_val > 25 && line_e_val < 25 && line_d_val < 25) {
2     SensorLinha = 2;
3   }
4
5   ...
6
7   return SensorLinha;
8 }
```

Listagem 4.18: Condição para retornar o valor decimal 2.

No final a função retorna o valor da variável "SensorLinha"com o valor decimal calculado.

4.2.4 Biblioteca Controlo do Módulo para Reconhecimento de Cor

A biblioteca ARCC_ColorSensor tem todos os diferentes procedimentos que vão detetar os objetos com reconhecimento de cor ativo e indicar os valores de vermelho, verde e azul ao utilizador. Conforme acontece nas outras bibliotecas esta tem um procedimento de inicialização e um de leitura, Figura 4.39. Este tem também um procedimento de configuração inicial de cor que tem como propósito calibrar os valores de referência do sensor de cor, uma vez que de aula para aula as condições de iluminação mudam e o aluno deve efetuar esta calibração.

No Webots as condições são sempre iguais, porém o aluno deve à mesma compreender o processo de calibração e indicar os valores de referência.

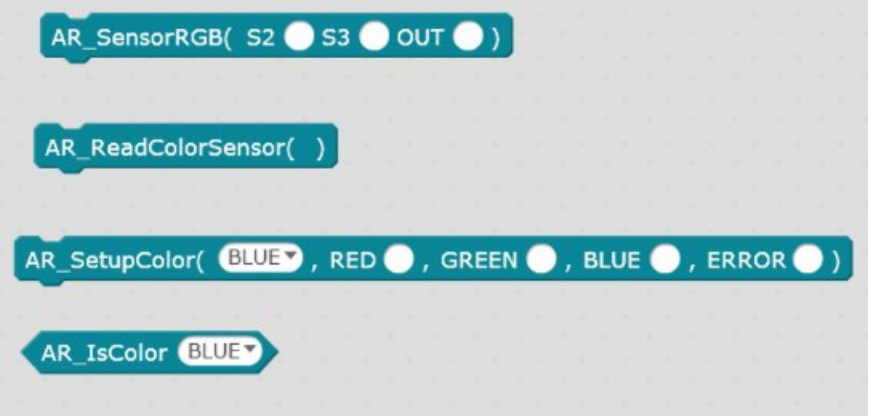


Figura 4.39: Blocos mBlock para leitura dos valores do sensor de cor.

A biblioteca original encontra-se dividida por classes e não procedimentos, de forma a manter as bibliotecas todas no mesmo formato. Devido a alguns erros de compilação detetados ao tentar compilar um controlador em C++ foi decidido também alterar este procedimento e dos sensores ultrassónicos para um em linguagem C, somente com procedimentos. O ficheiro *Header* desta biblioteca tem também um procedimento de leitura, calibração e comparação destes valores de referência com os lidos. E por fim um que define as cores RED, GREEN E BLUE como valores numéricos.

```
1  #ifndef ARCC_ColorSensor_h
2  #define ARCC_ColorSensor_h
3
4  #define RED 0
5  #define GREEN 1
6  #define BLUE 2
7
8  void AR_SensorRGB(int AR_S2, int AR_S3, int AR_OUT);
9  void AR_ReadColorSensor();
10 void AR_SetupColor(int _InitColor, double _R, double _G, double _B
       , double _Erro);
11 int AR_IsColor (int _Color);
12
13 #endif
```

Listagem 4.19: Ficheiro Header ARCC ColorSensor

As bibliotecas "camera.h", "camera_recognition_object.h"e "led.h"pertencem ao Webots e vão ser necessárias para o desenvolvimento da "ARCC_ColorSensor.h".

```
#include <webots/robot.h>
#include <webots/camera.h>
#include <webots/camera_recognition_object.h>
#include <webots/led.h>
#include <stdio.h>

#include "ARCC_ColorSensor.h"

#define TIME_STEP 64

int i = 0;

double R, G, B;
double AR_Red[4] = {0, 0, 0, 0};
double AR_Green[4] = {0, 0, 0, 0};
double AR_Blue[4] = {0, 0, 0, 0};
```

Listagem 4.20: Declaração das variáveis a serem utilizadas na Biblioteca ARCC Color Sensor

Em primeiro lugar são declarados as variáveis R,G,B que vão armazenar as componentes das cores lidas e os vetores AR_Red, AR_Green, AR_Blue que detêm os valores de referência para comparação.

Função `AR_SensorRGB`

O primeiro procedimento é o de configuração, que como nos anteriores, é ativada a câmera e o módulo de reconhecimento de cor. No final imprime no ecrã os pinos onde o utilizador tem ligado o sensor de cor ao Arduino.

```
void AR_SensorRGB(int AR_S2, int AR_S3, int AR_OUT) {
  WbDeviceTag cor = wb_robot_get_device("Sensor_Cor");

  wb_camera_enable(cor, TIME_STEP);
  wb_camera_recognition_enable(cor, TIME_STEP);

  printf("Pino S2: %d, Pino S3: %d, Pino OUT: %d\n", AR_S2, AR_S3,
       AR_OUT);
}
```

Listagem 4.21: Configuração da camêra virtual do robô simulado.

Função `AR_ReadColorSensor`

O procedimento de leitura do sensor de cor terá como propósito efetuar a leitura do sensor e guardar os valores das componentes da cor nas variáveis correspondentes. A Figura 4.40 apresenta o diagrama de fluxo para efetuar este procedimento.

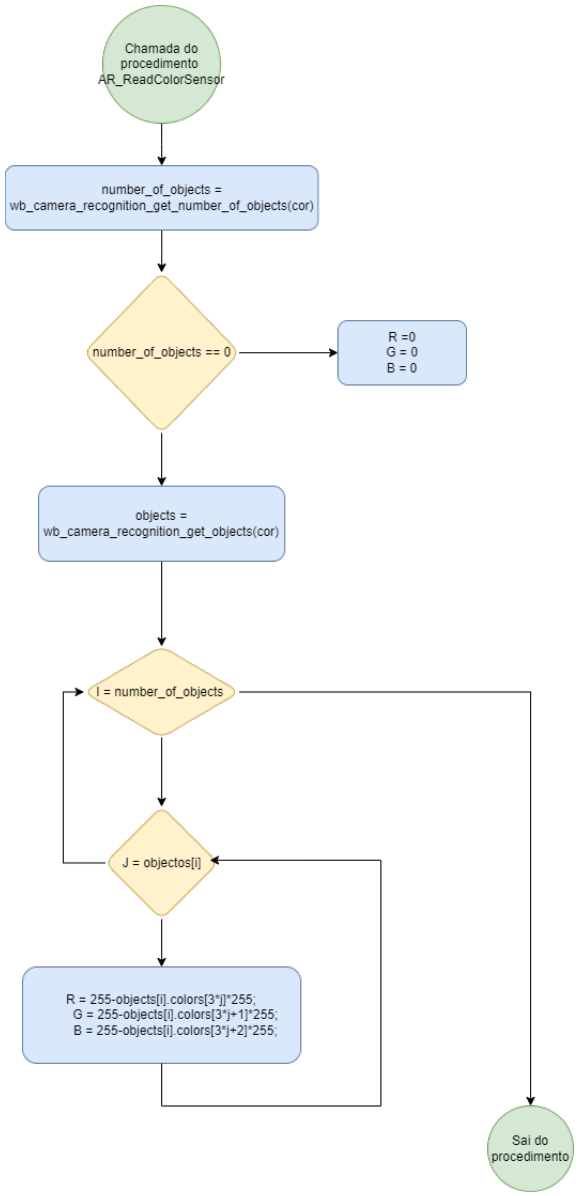


Figura 4.40: Fluxograma da função de Leitura do Sensor de Cor virtual.

Numa primeira fase ao chamar o procedimento, este irá verificar se a câmera está a visualizar um objeto com reconhecimento de cor ativo. Se não for o caso este coloca a zero as variáveis R,G e B.

```
1        WbDeviceTag cor = wb_robot_get_device("Sensor_Cor");
2
3      int n_objetos = wb_camera_recognition_get_number_of_objects(
           cor);
4
5      if (n_objetos == 0){
6
7      R=0;
8      G=0;
9      B=0;
10     }
```

Listagem 4.22: Colocação das variáveis das cores a zero se nenhum objeto for reconhecido.

Caso esteja presente um ou mais objetos no campo de visão da câmera será devolvido um ponteiro para uma lista de objetos detetados. Depois irá iterar para cada objeto e verificar o vetor "colors"que tem os valores das cores do objeto detetado, invertendo-as e convertendo os valores de 0 a 1 para 0 e 255. Por fim, o procedimento atribui a cada variável de R,G,B a componente da cor respetiva.

```
1     const WbCameraRecognitionObject *cores =
          wb_camera_recognition_get_objects(cor);
2      for (i = 0; i < n_objetos; ++i) {
3        for (int j = 0; j < cores[i].number_of_colors; ++j){
4          R = 255-cores[i].colors[3*j]*255;
5          G = 255-cores[i].colors[3*j+1]*255;
6          B = 255-cores[i].colors[3*j+2]*255;
7          }
8      }
9  }
```

Listagem 4.23: Gravação nas váriaveis RGB os valores das cores.

Função AR_SetupColor

Com esta função são guardadas as componentes de referência para as três cores para depois o programa comparar com os valores lidos. O código seguinte refere-se à cor vermelha:

```
1 void AR_SetupColor (int _InitColor, double _R, double _G, double
      _B, double _Erro) {
2   switch (_InitColor) {
3     case 0:
4       AR_Red[0] = _R;
5       AR_Red[1] = _G;
6       AR_Red[2] = _B;
7       AR_Red[3] = _Erro;
8       break;
```

Listagem 4.24: Declara o valor de referência para a cor vermelha.

Função AR_IsColor

Será no procedimento AR_IsColor que são comparados os valores de referência com os valores lidos. O valor da quarta posição de cada vetor indica um valor de "erro"originando assim, um intervalo de valores , no qual o valor mínimo é a subtração do valor de erro com o de referência e o valor máximo é a adição do valor de erro com o de referência.

O diagrama de fluxo da 4.41 demonstra os passos deste procedimento.

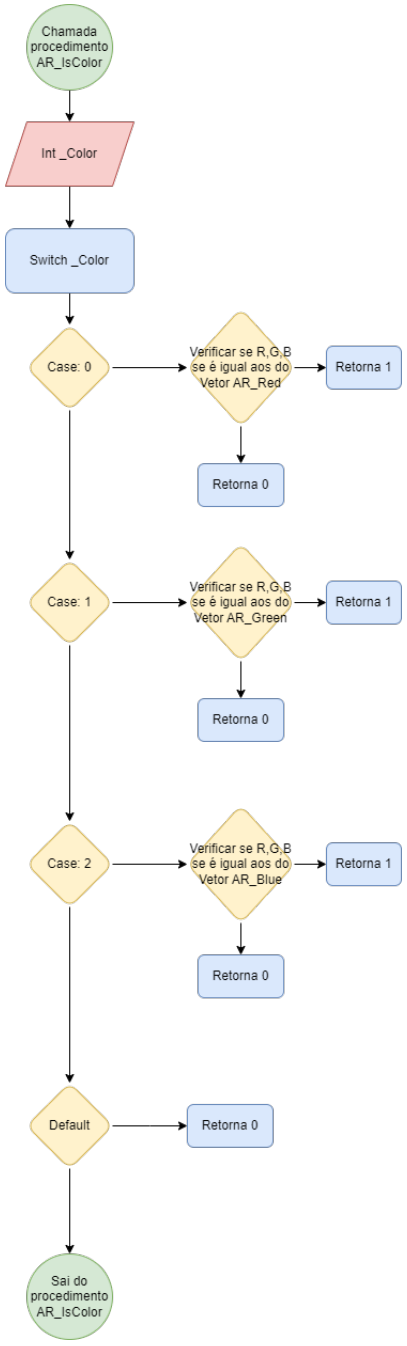


Figura 4.41: Fluxograma do procedimento de comparação dos valores de referência.

O código seguinte apresenta a programação para verificar e comparar o valor do sensor de cor com o referenciado pela função AR_SetupColor.

```
1 int AR_IsColor(int _Color) {
2   switch (_Color) {
3     case 0:
4       if ((R > AR_Red[0] - AR_Red[3] && R < AR_Red[0] + AR_Red[3])
             &&
5           (G > AR_Red[1] - AR_Red[3] && G < AR_Red[1] + AR_Red[3])
                &&
6           (B > AR_Red[2] - AR_Red[3] && B < AR_Red[2] + AR_Red[3])
               )
7         return 1;
8       else
9         return 0;
```

Listagem 4.25: Comparação dos valores das cores detetados com os valores referência.

4.2.5 Biblioteca Controlo dos sensores Ultrassónicos

Na bibleotaca AR_Sonar estão os procedimentos para configuração dos vários sensores e também a função de leitura dos mesmos. O ficheiro *Header* irá conter três procedimentos de configuração e três procedimentos de leitura.

```
1  #ifndef AR_Sonar_h
2  #define AR_Sonar_h
3
4  void ARCC_SonarDireito();
5  void ARCC_SonarEsquerdo();
6  void ARCC_SonarCentro();
7
8  double SonarDireito_LerSonar();
9  double SonarEsquerdo_LerSonar();
10 double SonarCentro_LerSonar();
11
12 #endif
```

Listagem 4.26: Procedimentos presentes na biblioteca Sonar.

A Figura 4.42 apresenta o procedimento de configuração e leitura do sensor ultrassónico direito no mBlock.

Figura 4.42: Blocos mBlock para leitura dos sensores ultrassonicos.

No inicio do código do ficheiro .c são incluídas as bibliotecas Webots e indicado o ficheiro *header* para esta biblioteca. Para controlo dos sensores distância no Webots, igualmente ao módulo de deteção de linha, a biblioteca necessária é a "webots/distance_sensor.h"

```
1 #include <webots/robot.h>
2 #include <webots/distance_sensor.h>
3
4 #include "ARCC_Sonar.h"
5
6 #define TIME_STEP 64
```

Listagem 4.27: Inclusão das bibliotecas necessárias auxiliares para a leitura dos sensores ultrassónicos.

O procedimento ARCC_SonarDireito inicializa o sensor da direita e tem, como os outros anteriormente programados, a chamada e identificação do sensor que vai ser posteriormente ativado. Os procedimentos para o sensor do centro e da esquerda são idênticos.

```
1 void ARCC_SonarDireito() {
2   WbDeviceTag sonar_D = wb_robot_get_device("Sonar_D");
3   wb_distance_sensor_enable(sonar_D, TIME_STEP);
4 }
```

Listagem 4.28: Procedimento de configuração do sensor ultrassónico Direito

Por último a função de leitura é extremamente básica, comparadas com outras já exploradas neste capítulo. Esta só tem de efetuar a leitura do sensor e como este já indica os valores dentro dos limites estabelecidos na *Lookup Table* de cada um destes, somente terá que retornar o valor da distância detetada para o utilizador.

```
1 double SonarDireito_LerSonar() {
2   WbDeviceTag sonar_D = wb_robot_get_device("Sonar_D");
3   double sonar_D_val = wb_distance_sensor_get_value(sonar_D);
4   return sonar_D_val;
5 }
```

Listagem 4.29: Função de leitura do sensor ultrassónico Direito.

Capítulo 5

Resultados e Análise

Nesta capítulo serão demonstrados e analisados todos os testes realizados aos vários sensores e atuadores do robô digital de forma individual, assim como em conjunto para concluir os vários níveis da prova First Challenger. Será também testada a habilidade de realizar o código no mBlock/Arduino e transferi-lá para o Webots.

5.1 Testes LED RGB e Buzzer

Para testar o LED RGB e o Buzzer presentes no robô foi utilizado um código, escrito em linguagem C, acrescentando somente linhas de código antes e dentro da função "Setup", e também na função "Loop". Foi realizado um código como apresentado a seguir onde neste irá iterar por cada uma das cores no LED e no final ativar o Buzzer. Para delimitar o tempo para alterar a cor foi utilizada a função "delay".

```
1  #include "Arduino.h"
2
3  //Procedimentos e variaveis
4
5  void setup(){
6
7     pinMode(40,OUTPUT);
8     pinMode(41,OUTPUT);
9     pinMode(42,OUTPUT);
10    pinMode(43,OUTPUT);
11
```

```
12 }
13
14 void loop(){
15
16   digitalWrite(40,1);
17   delay(1000);
18   digitalWrite(40,0);
19   digitalWrite(41,1);
20   delay(1000);
21   digitalWrite(41,0);
22   digitalWrite(42,1);
23   delay(1000);
24   digitalWrite(42,0);
25   delay(1000);
26
27 }
```

Listagem 5.1: Código de teste para o LED RGB e Buzzer

Na Figura 5.1 é possível observar as várias cores (vermelho, azul, verde) apresentadas pelo LED. Apesar de pela imagem não ser possível demonstrar a saída de som, este fenómeno ocorreu como previsto. Este procedimento demorou a repetir, tempo de ciclo, mais ou menos cinco segundos o que comprova que o procedimento de delay funcionou corretamente, esperando de um em um segundo para alterar o estado do LED e Buzzer.

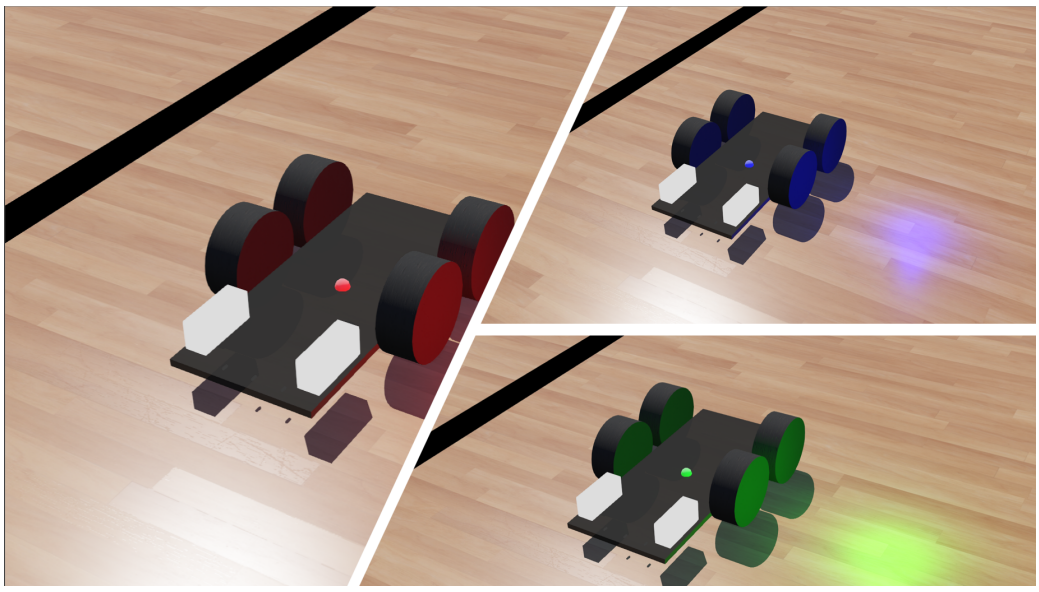


Figura 5.1: Resultado dos testes aos LEDs RGB

5.2 Motores

Para testar o funcionamento dos motores foi realizado um código muito idêntico ao produzido durante as aulas da Academia que coloca o robô a executar diferentes movimentos. Assim é possível testar a direção, velocidade e a atribuição correta do lado esquerdo e direito. O código seguinte demonstra o procedimento produzido, denominado "TesteMotores".

```
1  //Includes
2
3  #include "Arduino.h"
4  #include "ARSC_Motores.h"
5
6  void setup(){
7
8      AR_InitMotores(11,13,3,12);
9  }
10
11 void loop(){
12
13   AR_MotorDir(200,1);
14   AR_MotorEsq(200,1);
15   delay(1000);
16   AR_MotorDir(200,0);
17   AR_MotorEsq(200,0);
18   delay(1000);
19   AR_MotorDir(200,1);
20   AR_MotorEsq(200,0);
21   delay(1000);
22   AR_MotorDir(200,0);
23   AR_MotorEsq(200,1);
24   delay(1000);
25
26 }
```

Listagem 5.2: Código de teste para os motores.

A Figura 5.2 apresenta a posição do robô a alterar-se conforme os movimentos que vai fazendo.

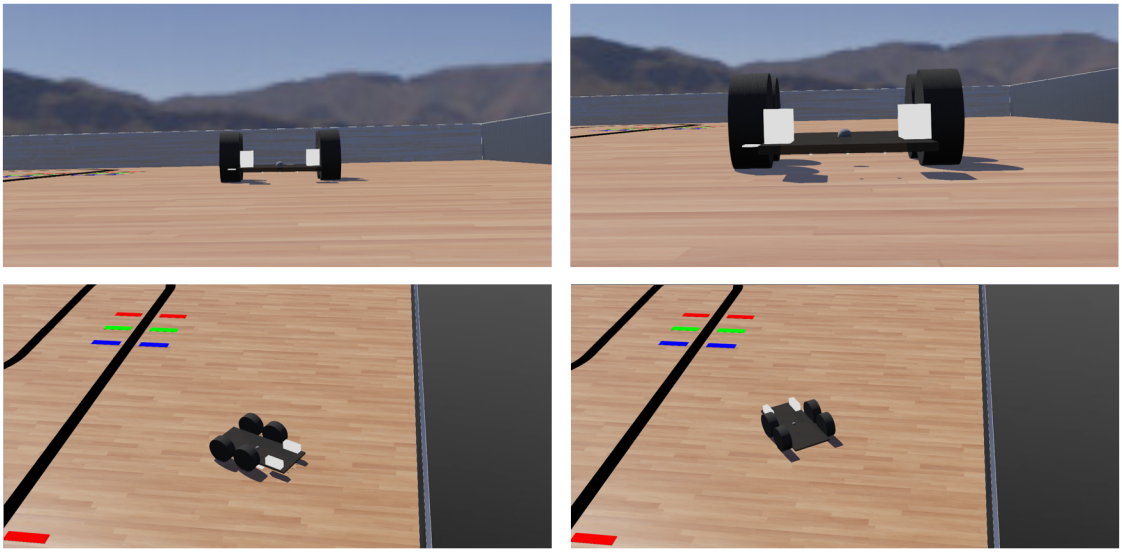


Figura 5.2: Resultado dos testes aos Motores

5.3 Sensor de Pista

Para testar o módulo de deteção de pista presente no robô desenvolveu-se um código, presente no código seguinte, que efetua uma leitura do sensor e imprime no ecrã o valor já transformado para decimal na consola.

```
1
2  //Includes
3
4  #include "Arduino.h"
5  # include "ARSC_SensorDePista.h"
6
7  //Procedimentos e variaveiis
8
9  int sensorpista = 0;
10
11 void setup(){
12
13   AR_InitSensorPista3(5,6,7);
14
15    pinMode(40,OUTPUT);
16    pinMode(41,OUTPUT);
17    pinMode(42,OUTPUT);
18    pinMode(43,OUTPUT);
19
```

```
20 }
21
22 void loop(){
23
24 sensorpista = AR_LerSensorPista3 ();
25 printf("Pista = %d \n" ,sensorpista);
26 delay(1000);
27
28 }
```

Listagem 5.3: Teste e leitura do Sensor de Pista.

Para este código funcionar é preciso efetuar a chamada da biblioteca ARSC _SensorDePista.h e guardar numa variável o valor lido pela AR_LerSensorPista3(). A função "printf"é utilizada para imprimir o valor da variável na consola. Como pode ser observado na Figura 5.3 o robô ao estar posicionado no centro da linha imprime o valor 2, quando se encontra à direita da linha imprime o valor 4 e quando está à esquerda da linha imprime o valor 1. É possível afirmar que o procedimento de leitura encontra-se a funcionar como esperado, podendo então avançar para o código do primeiro nível do First Challenger.

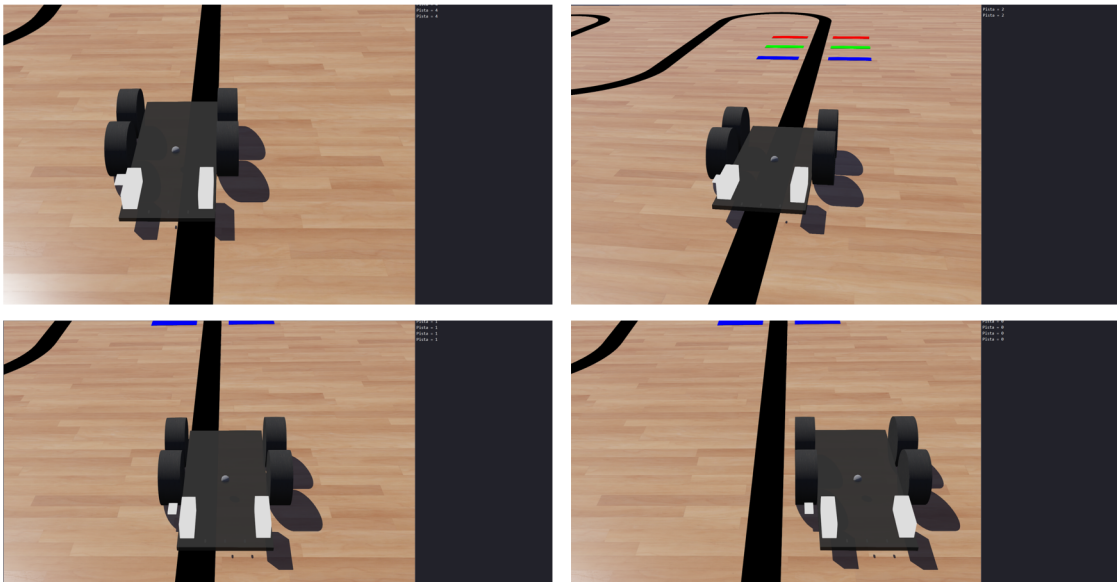


Figura 5.3: Resultado dos testes ao módulo deteção da pista

5.4 Nível 1 - Seguimento de Linha

Após efetuados os testes ao movimento dos motores e módulo de deteção de linha é possível desenvolver o código para seguimento de linha que é o objetivo do primeiro nível da prova First Challenger. O código realizado em mBlock, Figura 5.10, é o

demonstrado aos alunos de nível 1 e 2 da Academia de Robótica. Este foi copiado e colado diretamente no Webots sem ter sido necessário qualquer tipo de alteração.

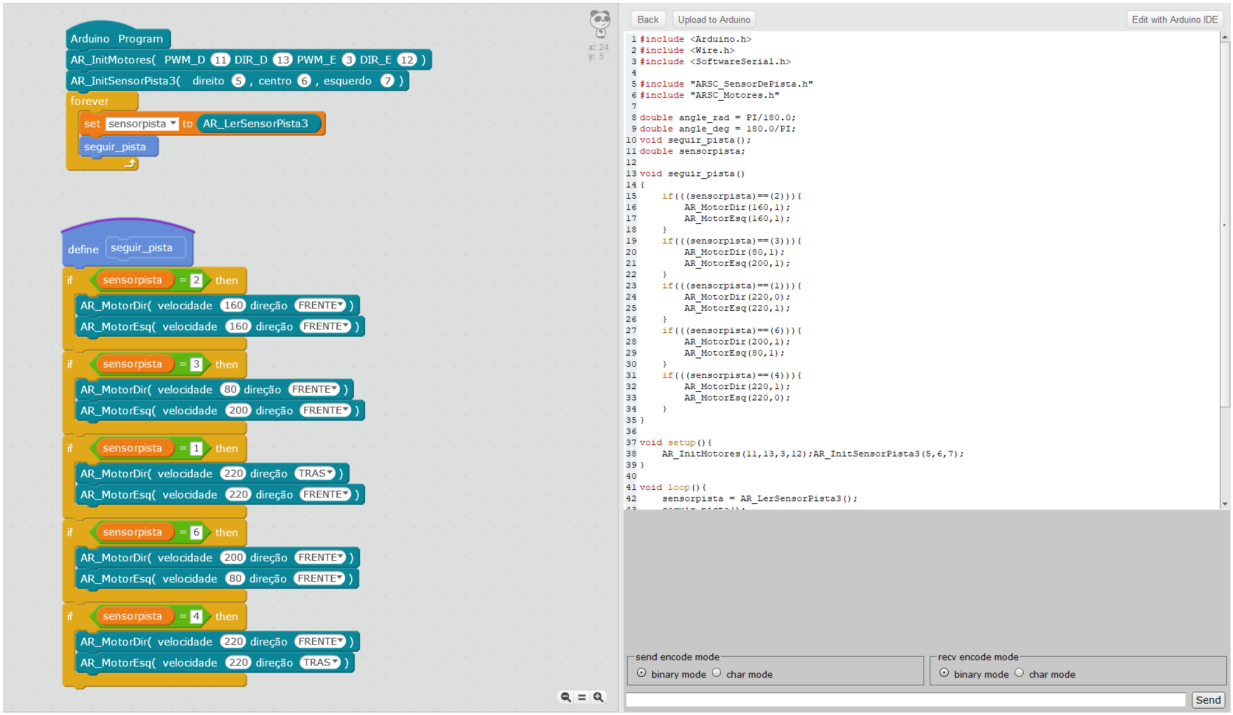


Figura 5.4: Código efetuado no mBlock para seguimento da linha

Com as velocidades definidas o robô apresenta correções muito drásticas o que é muito parecido com o que acontece no robô real. Estas velocidades foram ajustadas diretamente no Webots e o comportamento do robô tornou-se muito mais suave.

O robô simulado conseguiu completar o desafio em aproximadamente 35 segundos, Figura 5.5, o que é um valor muito próximo da realidade para essas velocidades.

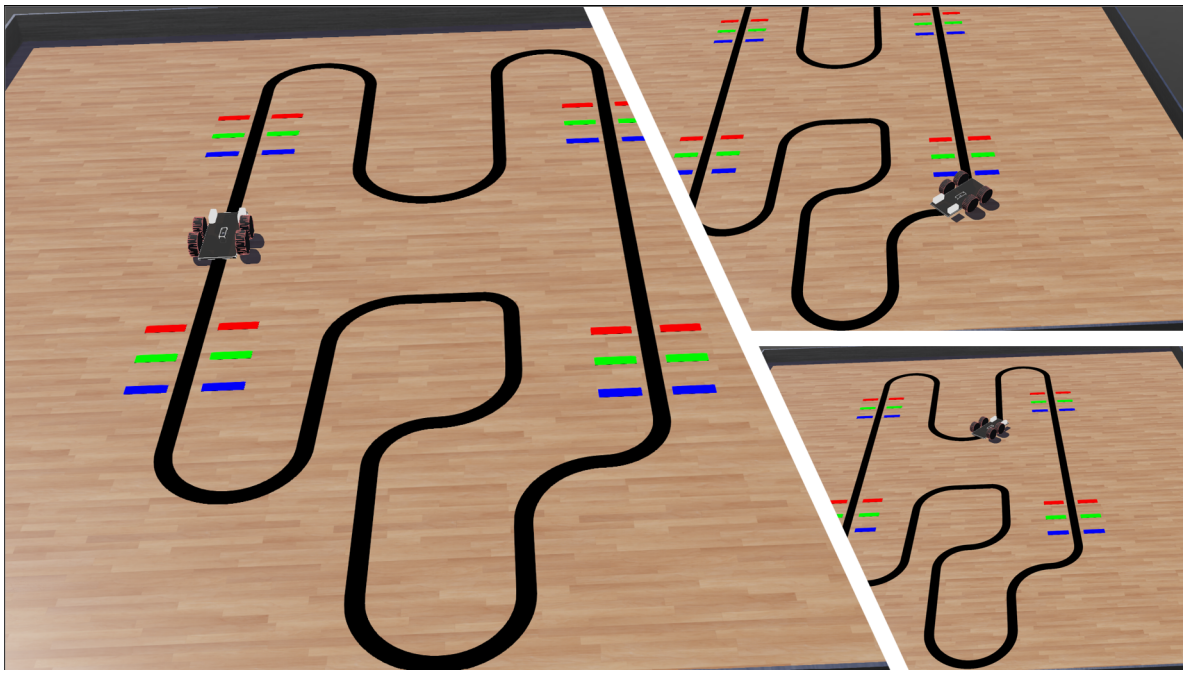


Figura 5.5: Resultado dos testes ao código para o Nível 1 da prova First Challenger

O código seguinte foi o obtido via mBlock e copiado para dentro do controlador Webots.

```
1
2  //Includes
3
4  #include "Arduino.h"
5  #include "ARSC_Motores.h"
6  #include "ARSC_SensorDePista.h"
7
8  //Procedimentos e variaveis
9
10 void seguir_pista();
11
12 int sensorpista = 0;
13
14 void setup(){
15
16   AR_InitMotores(11,13,3,12);
17   AR_InitSensorPista3(5,6,7);
18
19   pinMode(40,OUTPUT);
20   pinMode(41,OUTPUT);
```

```
21   pinMode(42,OUTPUT);
22   pinMode(43,OUTPUT);
23
24 }
25
26 void loop(){
27
28   sensorpista = AR_LerSensorPista3();
29
30   seguir_pista();
31
32 }
33
34 void seguir_pista()
35 {
36     if(((sensorpista)==(2))){
37         AR_MotorDir(160,1);
38         AR_MotorEsq(160,1);
39     }
40     if(((sensorpista)==(3))){
41         AR_MotorDir(80,1);
42         AR_MotorEsq(200,1);
43     }
44     if(((sensorpista)==(1))){
45         AR_MotorDir(220,0);
46         AR_MotorEsq(220,1);
47     }
48     if(((sensorpista)==(6))){
49         AR_MotorDir(200,1);
50         AR_MotorEsq(80,1);
51     }
52     if(((sensorpista)==(4))){
53         AR_MotorDir(220,1);
54         AR_MotorEsq(220,0);
55     }
56 }
```

Listagem 5.4: Código para efetuar o primeiro nível da prova First Challenger.

5.5 Testes ao Sensor de Cor

Para testar o Sensor de Cor foi aplicado um código em que somente efetua a leitura do sensor e indica quando detetado um objeto.

```
1  //Includes
2
3  #include "Arduino.h"
4  #include "ARCC_ColorSensor.h"
5
6  //Procedimentos e variaveiis
7
8  void setup(){
9
10  AR_SensorRGB(30,31,4);
11 }
12
13 void loop(){
14
15 AR_ReadColorSensor();
16
17 }
```

Listagem 5.5: Teste ao Sensor de Cor Virtual.

Durante o teste do sensor de cor detetou-se uma anomalia significativa no comportamento do robô. A Figura 5.6 demonstra que o sensor deteta a cor antes que esta aparecesse na janela da imagem capturada pela câmera, resultando em uma paragem prematura do robô. Esta deteção antecipada faz com que o robô pare antes de alcançar a linha desejada.

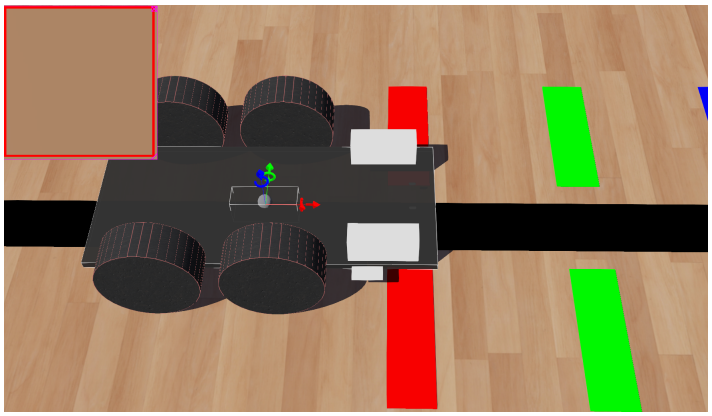


Figura 5.6: Módulo câmera a detetar a cor de forma prematura

Para solucionar este problema foram implementadas várias correções. Começou-se por reduzir o tamanho do objeto que permite a câmera detetar uma cor (linhas a

volta da pista preta) acrescentado outro objeto com o tamanho original, mas sem a opção de reconhecimento ativa, Figura 5.7.

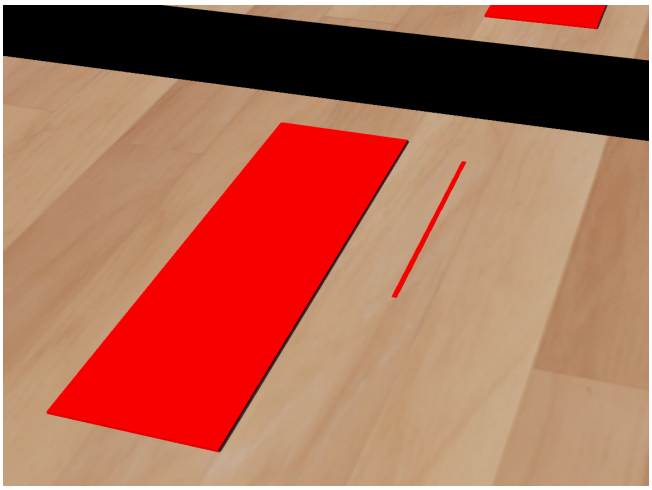


Figura 5.7: Bloco com tamanho reduzido com reconhecimento de cor ativo

Desta forma, o robô conseguirá detetar uma área bastante menor, no entanto para o utilizador parece que nada mudou visualmente na pista e assim o robô simulado efetua a paragem na posição correta.

Ainda assim foi efetuado um ajuste o ajuste à posição do sensor no eixo das abcissas, de forma a proporcionar uma posição de deteção mais adequada. Estas mudanças permitiram que o robô simulado efetue a paragem mais em cima da linha como acontece no robô real, Figura 5.8.

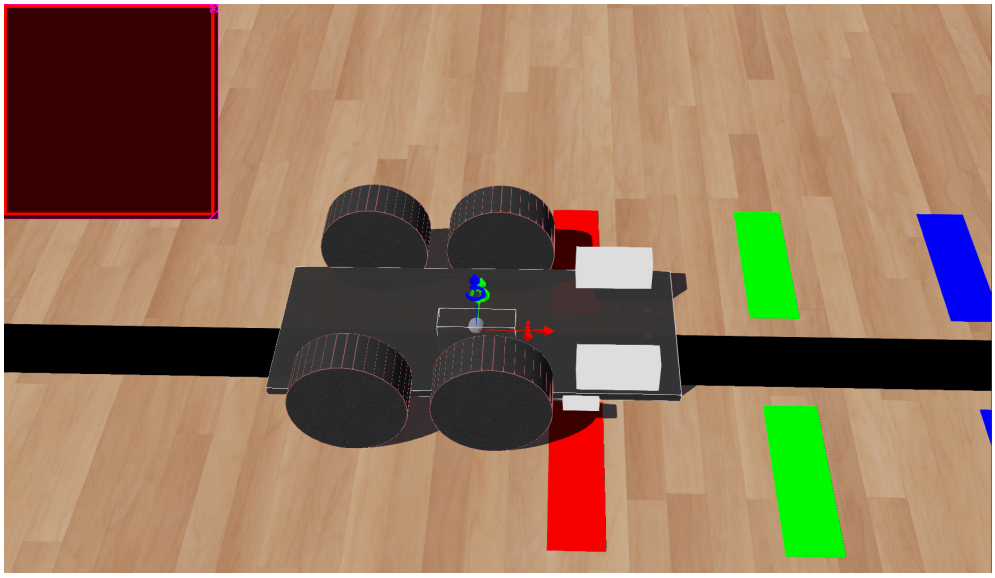


Figura 5.8: Paragem correta do robô simulado ao detetar a linha vermelha.

5.6 Nível 2 - Seguimento de Linha e Deteção de Cores

No nível 2 do desafio o robô foi configurado para seguir uma pista e detetar cores específicas. O objetivo é que o robô pare na cor detetada e ative o LED correspondente, conforme observado na Figura 5.9. O robô completou o percurso em menos de dois minutos, sem anomalias, demonstrando a eficácia das modificações implementadas.

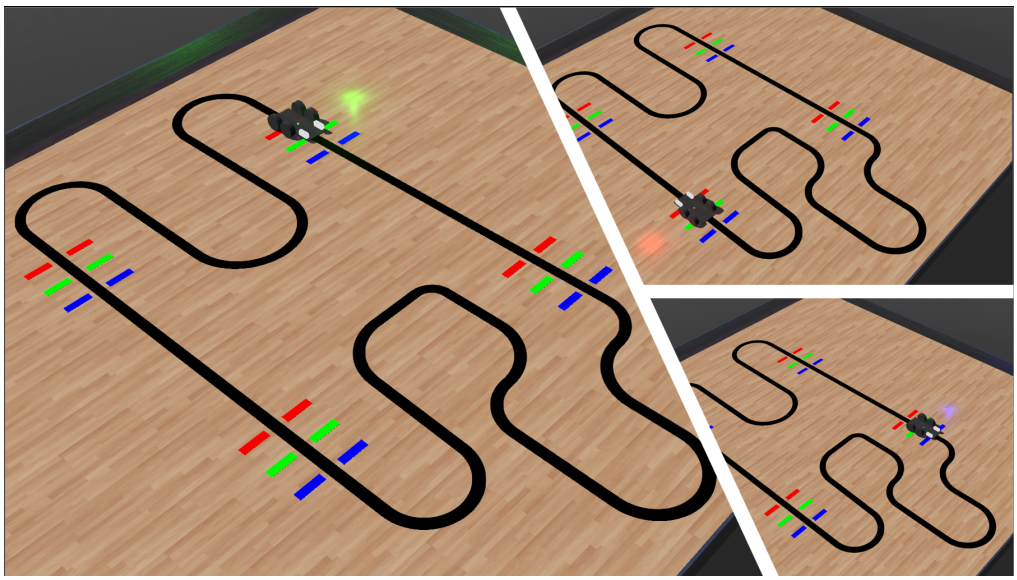


Figura 5.9: Seguimento de linha com deteção das cores.

Foi aplicado uma pequena movimentação ao robô após a sua paragem para o obrigar a sair da linha e não ficar bloqueado a detetar sempre a mesma linha de cor. Apesar deste fenómeno não acontecer em todos os robôs reais, é um episódio que por vezes pode ocorrer. O desenvolvimento deste código é importante para evitar essas particularidades que podem surgir durante o torneio.

```
1  #include "Arduino.h"
2  #include "ARSC_Motores.h"
3  #include "ARSC_SensorDePista.h"
4  #include "ARCC_ColorSensor.h"
5
6  //Procedimentos e vari veis
7
8  void seguir_pista();
9  double sensorpista;
10 void LedRGB(double CorLED);
11 void seguir_pista_cor();
12
13 void setup(){
14
```

```
15      AR_InitMotores(11,13,3,12);AR_InitSensorPista3(5,6,7);
16      AR_SensorRGB(30,31,4);
17      AR_SetupColor(0,0,255,255,10);
18      AR_SetupColor(1,255,0,255,10);
19      AR_SetupColor(2,255,255,0,10);
20
21      AR_MotorDir(0,1);
22      AR_MotorEsq(0,1);
23      pinMode(40,OUTPUT);
24      pinMode(41,OUTPUT);
25      pinMode(42,OUTPUT);
26      pinMode(43,OUTPUT);
27  }
28
29  void loop(){
30
31    sensorpista = AR_LerSensorPista3();
32
33    seguir_pista_cor();
34
35    //seguir_pista();
36
37  }
38
39  void seguir_pista()
40  {
41      if(((sensorpista)==(2))){
42          AR_MotorDir(160,1);
43          AR_MotorEsq(160,1);
44      }
45      if(((sensorpista)==(3))){
46          AR_MotorDir(160,1);
47          AR_MotorEsq(180,1);
48      }
49      if(((sensorpista)==(1))){
50          AR_MotorDir(220,0);
51          AR_MotorEsq(220,1);
52      }
53      if(((sensorpista)==(6))){
54          AR_MotorDir(180,1);
55          AR_MotorEsq(160,1);
56      }
57      if(((sensorpista)==(4))){
58          AR_MotorDir(220,1);
59          AR_MotorEsq(220,0);
60      }
61  }
62
63  void LedRGB(double CorLED)
```

```
64 {
65     if(((CorLED)==(0))){
66         digitalWrite(40,0);
67         digitalWrite(41,0);
68         digitalWrite(42,0);
69     }
70     if(((CorLED)==(1))){
71         digitalWrite(40,1);
72     }
73     if(((CorLED)==(2))){
74         digitalWrite(41,1);
75     }
76     if(((CorLED)==(3))){
77         digitalWrite(42,1);
78     }
79 }
80
81 void seguir_pista_cor()
82 {
83     AR_ReadColorSensor();
84     if(AR_IsColor(0)){
85         AR_MotorDir(0,1);
86         AR_MotorEsq(0,1);
87         LedRGB(1);
88         digitalWrite(43,1);
89         _delay(3);
90         digitalWrite(43,0);
91         _delay(2);
92         AR_MotorDir(150,1);
93         AR_MotorEsq(150,1);
94         _delay(0.2);
95     }
96     if(AR_IsColor(1)){
97         AR_MotorDir(0,1);
98         AR_MotorEsq(0,1);
99         LedRGB(2);
100        digitalWrite(43,1);
101        _delay(3);
102        digitalWrite(43,0);
103        _delay(2);
104        AR_MotorDir(150,1);
105        AR_MotorEsq(150,1);
106        _delay(0.2);
107    }
108    if(AR_IsColor(2)){
109        AR_MotorDir(0,1);
110        AR_MotorEsq(0,1);
111        LedRGB(3);
112        digitalWrite(43,1);
```

```
113             _delay(3);
114             digitalWrite(43,0);
115             _delay(2);
116             AR_MotorDir(150,1);
117             AR_MotorEsq(150,1);
118             _delay(0.2);
119         }
120         LedRGB(0);
121         seguir_pista();
122     }
```

Listagem 5.6: Código para efetuar o Nível 2 da Prova First Challenger.

O código utilizado para este nível de desafio foi adaptado do mBlock e integrado no Webots. Este código é responsável por seguir a linha, detetar a cor e acionar o LED correspondente.

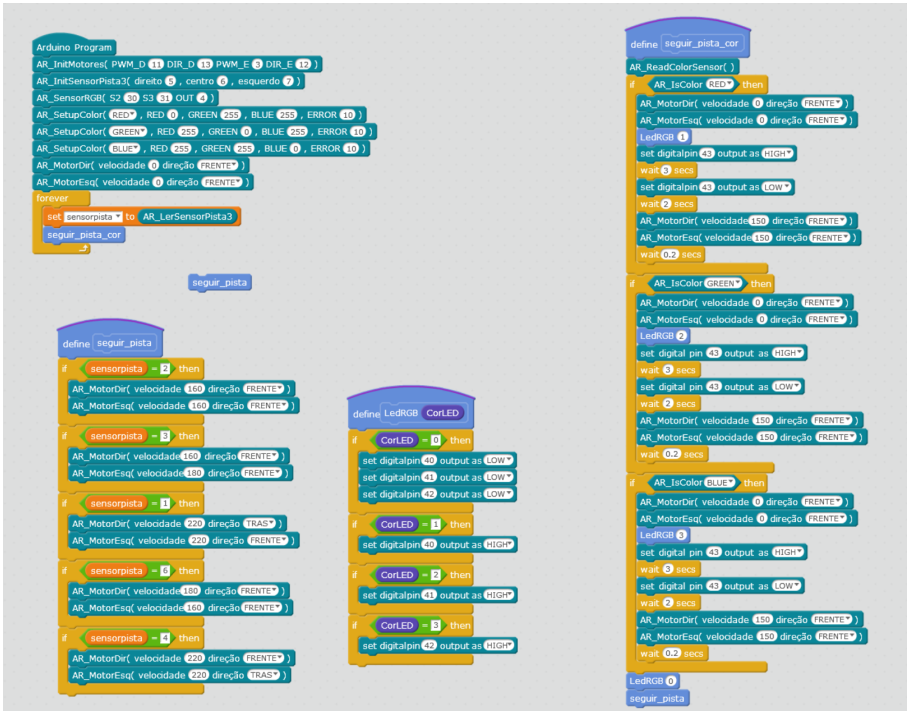


Figura 5.10: Código efetuado no mBlock para seguimento da linha

5.7 Sensores Ultrassónicos

Ambos os sensores ultrassónicos do robô virtual foram testados, simplesmente, com uma leitura e impressão na consola dos resultados obtidos. Como comprovado pela Figura 5.11 pode-se afirmar que estes encontram-se a funcionar corretamente.

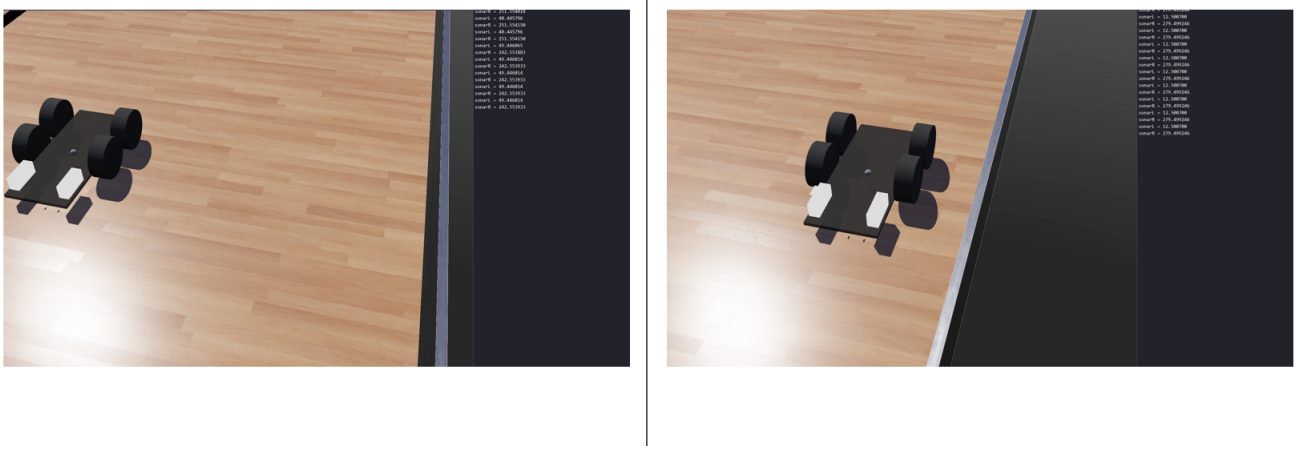


Figura 5.11: Resultado dos testes aos sonares dos Robô digital

Estes são os únicos componentes que necessitam de alguma adaptação do mBlock para a definição dos sensores. Para tornar o processo mais simples e sem usar duas linguagens de programação um pouco distintas, foi decidido que seria colocado tudo em linguagem C, só com procedimentos e sem utilizar classes.

```
1
2  //Includes
3
4  #include "Arduino.h"
5  #include "ARCC_Sonar.h"
6
7  double sonarL = 0.0;
8  double sonarR = 0.0;
9
10 //Procedimentos e variaveis
11
12 int sensorpista = 0;
13
14 void setup(){
15
16    ARCC_SonarEsquerdo();
17    ARCC_SonarDireito();
18
19    pinMode(40,OUTPUT);
20    pinMode(41,OUTPUT);
```

```
21      pinMode(42,OUTPUT);
22      pinMode(43,OUTPUT);
23
24  }
25
26  void loop(){
27
28  sonarL = SonarEsquerdo_LerSonar();
29  sonarR = SonarDireito_LerSonar();
30
31  printf("sonarL = %f\n",sonarL);
32  printf("sonarR = %f\n",sonarR);
33
34  delay(1000);
35
36  }
```

Listagem 5.7: Código para teste dos sensores ultrassónicos.

5.8 Nível 3 - Seguimento de Pista, Deteção de Cor e Seguimento de Parede

Após o teste e verificação de todos os componentes é possível aplicar os algoritmos de seguimento de parede lecionados no curso da Academia de Robótica. Os procedimentos e variáveis foram obtidos do mBlock e ajustados de seguida no Webots, conseguindo o robô completar uma volta em cerca de dois minutos. Conforme indica a Figura 5.12, o robô segue a linha e quando esta deixa de existir, ele passa a seguir as paredes, continuando a detetar todas as cores presentes em todo o circuito.

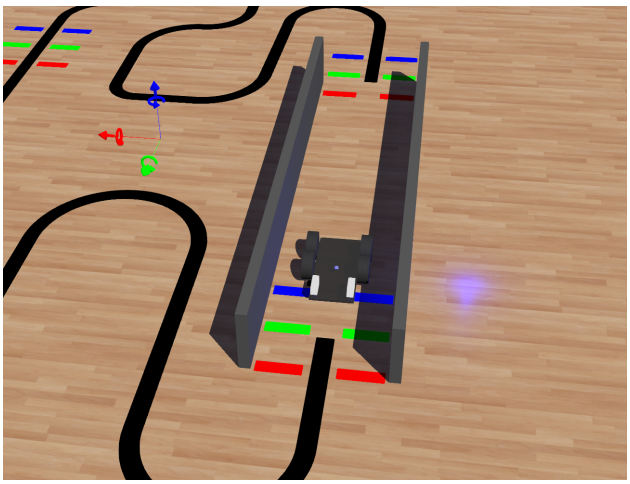


Figura 5.12: Robô a detetar a cor azul enquanto se move por entre as paredes.

O seguinte código é o que aluno deve conter no final do curso e no dia da prova do First Challenger.

```
#include "Arduino.h"
#include "ARSC_Motores.h"
#include "ARSC_SensorDePista.h"
#include "ARCC_ColorSensor.h"
#include "ARCC_Sonar.h"

Procedimentos e variaveis

void seguir_pista();
double sensorpista;
void LedRGB(double CorLED);
void seguir_pista_cor();

void seguir_parede();
void paredes_parar_cor();
void pista_paredes_parar_cor();

double sonarL;
double sonarR;

double DifDist = 0.0;

void setup(){

    AR_InitMotores(11,13,3,12);AR_InitSensorPista3(5,6,7);
    AR_SensorRGB(30,31,4);
    AR_SetupColor(0,0,255,255,10);
    AR_SetupColor(1,255,0,255,10);
    AR_SetupColor(2,255,255,0,10);
    ARCC_SonarDireito();
    ARCC_SonarEsquerdo();

    AR_MotorDir(0,1);
    AR_MotorEsq(0,1);
    pinMode(40,OUTPUT);
    pinMode(41,OUTPUT);
    pinMode(42,OUTPUT);
    pinMode(43,OUTPUT);

}

void loop(){

  sensorpista = AR_LerSensorPista3();

```

```
46    sonarL = SonarEsquerdo_LerSonar();
47    sonarR = SonarDireito_LerSonar();
48    seguir_pista_cor();
49
50    //seguir_pista();
51
52    //AR_ReadColorSensor();
53
54    pista_paredes_parar_cor();
55
56  }
57
58  void seguir_pista()
59  {
60      if(((sensorpista)==(2))){
61          AR_MotorDir(160,1);
62          AR_MotorEsq(160,1);
63      }
64      if(((sensorpista)==(3))){
65          AR_MotorDir(160,1);
66          AR_MotorEsq(180,1);
67      }
68      if(((sensorpista)==(1))){
69          AR_MotorDir(220,0);
70          AR_MotorEsq(220,1);
71      }
72      if(((sensorpista)==(6))){
73          AR_MotorDir(180,1);
74          AR_MotorEsq(160,1);
75      }
76      if(((sensorpista)==(4))){
77          AR_MotorDir(220,1);
78          AR_MotorEsq(220,0);
79      }
80  }
81
82  void LedRGB(double CorLED)
83  {
84      if(((CorLED)==(0))){
85          digitalWrite(40,0);
86          digitalWrite(41,0);
87          digitalWrite(42,0);
88      }
89      if(((CorLED)==(1))){
90          digitalWrite(40,1);
91      }
92      if(((CorLED)==(2))){
93          digitalWrite(41,1);
94      }
```

```
95      if(((CorLED)==(3))){
96          digitalWrite(42,1);
97      }
98  }
99
100 void seguir_pista_cor()
101 {
102     AR_ReadColorSensor();
103     if(AR_IsColor(0)){
104         AR_MotorDir(0,1);
105         AR_MotorEsq(0,1);
106         LedRGB(1);
107         digitalWrite(43,1);
108         _delay(3);
109         digitalWrite(43,0);
110         _delay(2);
111         AR_MotorDir(150,1);
112         AR_MotorEsq(150,1);
113         _delay(0.2);
114     }
115     if(AR_IsColor(1)){
116         AR_MotorDir(0,1);
117         AR_MotorEsq(0,1);
118         LedRGB(2);
119         digitalWrite(43,1);
120         _delay(3);
121         digitalWrite(43,0);
122         _delay(2);
123         AR_MotorDir(150,1);
124         AR_MotorEsq(150,1);
125         _delay(0.2);
126     }
127     if(AR_IsColor(2)){
128         AR_MotorDir(0,1);
129         AR_MotorEsq(0,1);
130         LedRGB(3);
131         digitalWrite(43,1);
132         _delay(3);
133         digitalWrite(43,0);
134         _delay(2);
135         AR_MotorDir(150,1);
136         AR_MotorEsq(150,1);
137         _delay(0.2);
138     }
139     LedRGB(0);
140     seguir_pista();
141 }
142
143 void paredes_parar_cor(){
```

```
144
145     AR_ReadColorSensor();
146     if (AR_IsColor(RED))
147     {
148      AR_MotorDir(0,FRENTE);
149      AR_MotorEsq(0,FRENTE);
150      digitalWrite(43,1);
151      LedRGB(1);
152      delay(2000);
153      digitalWrite(43,0);
154      delay(3000);
155      AR_MotorDir(150,1);
156      AR_MotorEsq(150,1);
157      delay(200);
158     }
159     else if (AR_IsColor(GREEN))
160     {
161      AR_MotorDir(0,FRENTE);
162      AR_MotorEsq(0,FRENTE);
163      digitalWrite(43,1);
164      LedRGB(2);
165      delay(2000);
166      digitalWrite(43,0);
167      delay(3000);
168      AR_MotorDir(150,1);
169      AR_MotorEsq(150,1);
170      delay(200);
171     }
172     else if (AR_IsColor(BLUE))
173     {
174      AR_MotorDir(0,FRENTE);
175      AR_MotorEsq(0,FRENTE);
176      digitalWrite(43,1);
177      LedRGB(3);
178      delay(2000);
179      digitalWrite(43,0);
180      delay(3000);
181      AR_MotorDir(150,1);
182      AR_MotorEsq(150,1);
183      delay(200);
184     }
185     LedRGB(0);
186     seguir_parede();
187   }
188
189 void pista_paredes_parar_cor(){
190
191     if (sensorpista == 0)
192     {
```

```
193       if ((sonarL > 0 && sonarR < 100) && (sonarR > 0 && sonarR <
              100))
194       {
195       paredes_parar_cor();
196       }
197     }
198     else
199     {
200       seguir_pista_cor();
201     }
202
203   }
204
205  void seguir_parede(){
206
207     DifDist = sonarL - sonarR;
208
209     if (DifDist > 0)
210     {
211      AR_MotorDir(200,FRENTE);
212      AR_MotorEsq(120,FRENTE);
213     }
214     if (DifDist < 0)
215     {
216      AR_MotorDir(120,FRENTE);
217      AR_MotorEsq(200,FRENTE);
218     }
219   }
```

Listagem 5.8: Código para efetuar o Nível 3 da prova First Challenger.

Capítulo 6

Avaliação do Simulador em Contexto de Aula

Neste capítulo será apresentada a validação do simulador da competição First Challenger com alunos que frequentam as aulas da Academia de Robótica. A formação irá permitir aferir a facilidade de utilização, bem como identificar pontos de melhoria que podem vir a ser implementados.

As aulas lecionadas no curso da Academia de Robótica utilizam todos os métodos de formação estabelecidos:

- **Método Expositivo** - muito utilizado no inicio de cada módulo para expor e esclarecer o tema;
- **Método Demonstrativo** - método utilizado para exemplificar e demonstrar os métodos, lógicas de programação e também como devem efetuar a montagem do robô para alcançar o objetivo final;
- **Método Interrogativo** - são efetuadas muitas questões ao longo da aula para verificar o ponto de situação e também o conhecimento prévio de alguns alunos, tornando o processo de aprendizagem mais dinâmico e criando assim mais referências na mente do aluno;

- **Método ativo** - dependendo do decorrer da aula, muitos alunos terminam as tarefas principais com muito tempo de antecedência, ficando com disponibilidade para explorarem e aplicarem os conhecimento aprendidos de forma autónoma e proactiva.

A formação realizada, presente no Anexo A, usa os vários métodos para tornar o processo mais ativo e dinâmico. Será utilizado o método expositivo juntamente com o método interrogativo para explorar o mundo da simulação, um tema introduzido com base nas experiências que os alunos vão partilhando. Quando a aula se foca no simulador Webots a metodologia inicial será demonstrativa, Figura 6.1, onde o formador demonstra as funcionalidades do software, assim como deve executar a passagem de código do mBlock ou do Arduino.

Figura 6.1: Utilização do método demonstrativo na realização de código.

Este plano de formação contou com três aulas experimentais. A validação do simulador foi efetuada por dois grupos distintos de alunos:

- alunos que frequentam o curso do Nível 2 - apenas utilizam a programação por blocos
- alunos que frequentam o curso do Nível + - programam o robô em linguagem de programação C e C++.

As primeiras duas aulas foram lecionadas a meio do ano letivo durante a preparação para o FNR/bootcamp (neste ano os alunos não participaram no First Challenger do Festival Nacional de Robótica, porém foi efetuado um torneio entre todas as turmas dos diferentes colégios e do ISEP, muito idêntico ao First Challenger) o que permitiu que todos os alunos já estivessem familiarizados com todos os sensores e atuadores, não sendo necessário entrar em detalhe sobre estes durante o período de aulas experimentais. Após as duas primeiras aulas e mais tarde na terceira formação foram realizados questionários de avaliação do software/atividade aos alunos. Nas primeiras duas formações, em cada turma estiveram presentes 12 e 15 alunos respetivamente. Ambas as turmas são consideradas heterogéneas no parâmetro da idade, variando dos 10 anos até aos 18 anos. Os gráficos apresentados nas Figuras 6.2 e 6.3 apresentam a dispersão de idades em cada uma das turmas.

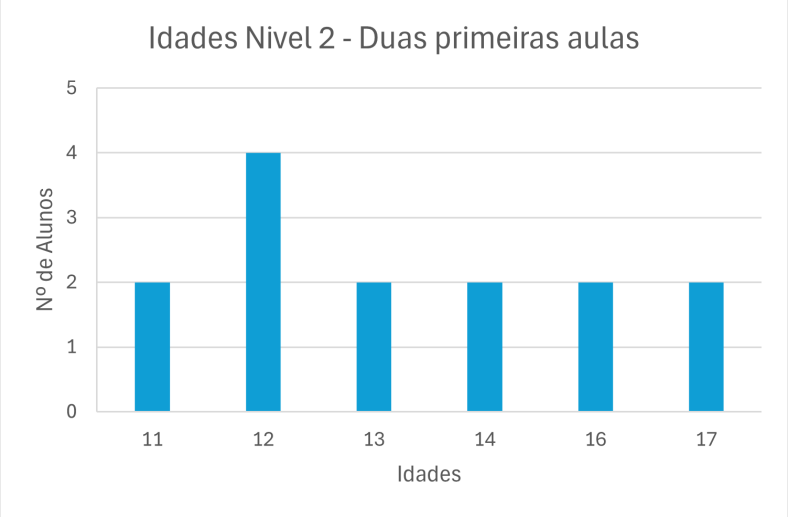


Figura 6.2: Idades dos alunos nas primeiras formações Nível 2

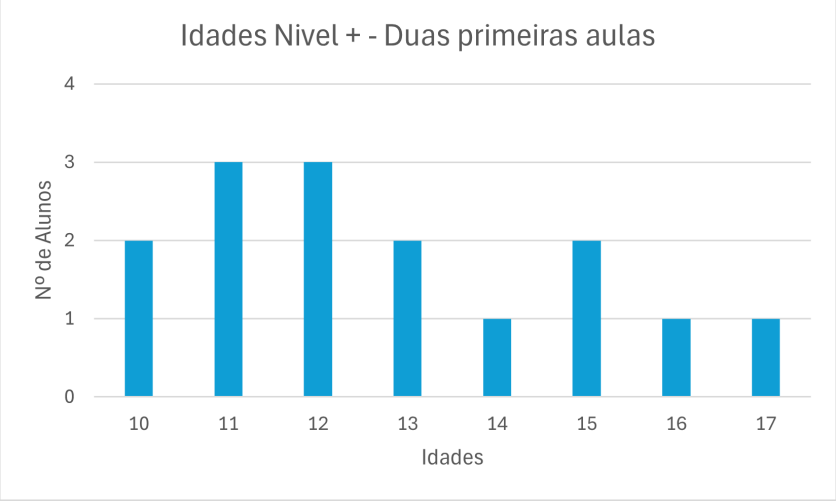


Figura 6.3: Idades dos alunos nas primeiras formações Nível +

Na primeira aula introduziu-se o simulador e o foco consistiu em aprender a utilizar as diferentes janelas do mesmo. No final os alunos utilizaram a programação já executada durante o decorrer das aulas normais para colocarem o robô a seguir a linha preta. Em alguns alunos detetou-se dificuldades em conseguir utilizar os controlos de manipulação da câmera do simulador, não conseguindo muitas vezes ver aquilo que pretendiam no ecrã, bem como na passagem do código devido a opção de copiar e colar, via atalhos do teclado, não funcionar corretamente em todos os computadores. Contudo, a maioria dos alunos conseguiu colocar o robô a circular na pista no final da primeira aula e notou-se um grande entusiasmo com a utilização de uma ferramenta nova. Na segunda aula os alunos utilizaram o Webots para aprenderem a programar o sensor de cor. Era expectável que os alunos de Nível 2, que ainda não o tinham ainda aprendido a utilizar este tipo de sensor tivessem mais dificuldades em concluir a tarefa, no entanto no final verificou-se o contrário .Todos os alunos de ambos os níveis conseguiram colocar o robô a seguir linha e parar quando detetada a cor. Na Figura 6.4 é possível identificar um conjunto de alunas a trabalharem em grupo para obterem um código de seguimento de linha via mBlock.

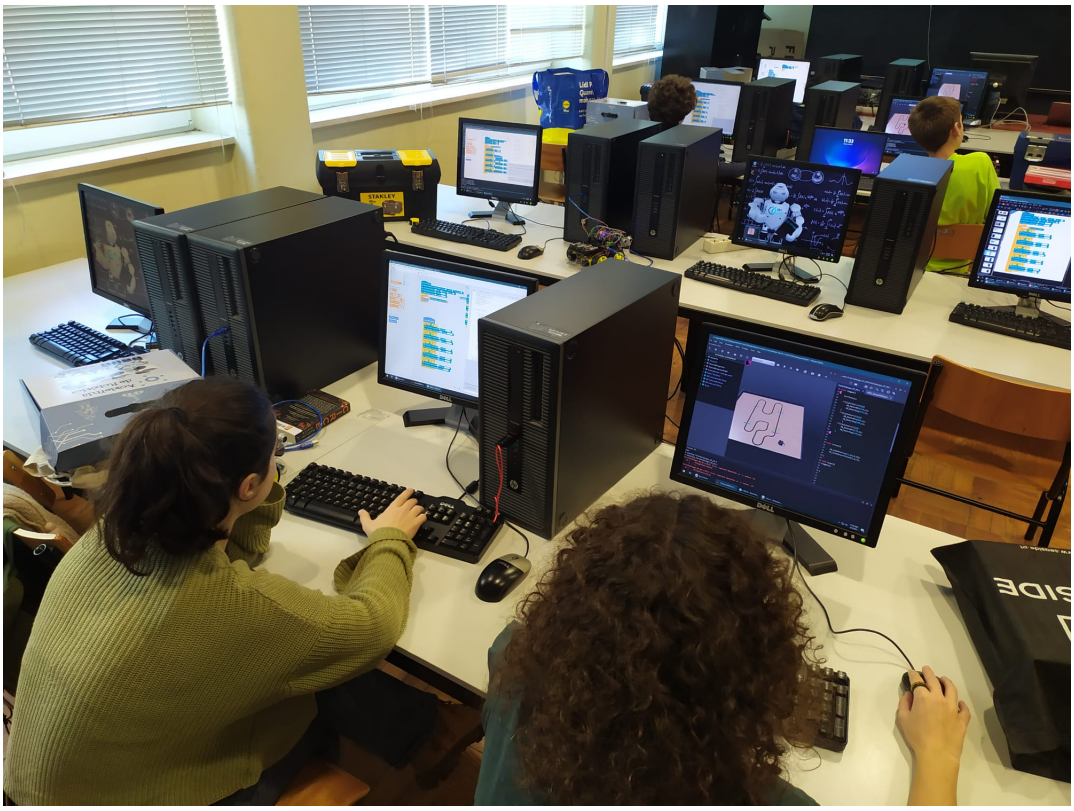


Figura 6.4: Alunos de Nível 2 a programar no mBlock e a testarem no Webots

No final destas duas formações foi realizado um questionário, presente no Anexo B, com o intuito de obter o *feedback* dos alunos sobre a dificuldade da utilização do simulador, da dificuldade da interação do mBlock/Arduino com o software de simulação e a opinião sobre a sua utilidade para a preparação do First Challenger.

Os gráficos presentes nas Figuras 6.5 e 6.6 apresentam os resultados sobre a dificuldade da atividade e utilização.

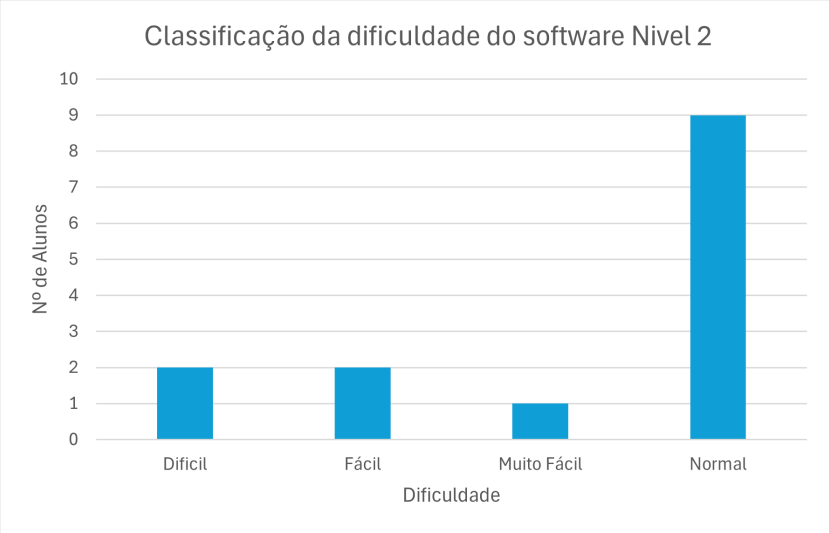


Figura 6.5: Nível de dificuldade sentida na utilização do software Webots - Nível 2

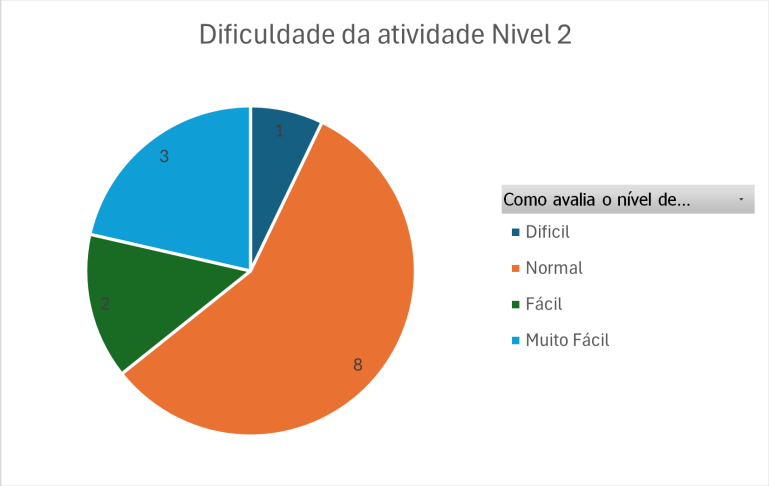


Figura 6.6: Nível de dificuldade sentida durante a formação - Nível 2

Analisando ambos os gráficos nenhum aluno apresentou muita dificuldade durante a aula experimental e em média todos os alunos realizaram-na sem se verificar alguma dificuldade anormal que inviabiliza-se a utilização do robô simulado durante o ano letivo.

Os alunos quando questionados sobre a utilidade do Webots revelaram que entenderam a importância da sua utilização e utilidade, como pode ser confirmado no gráfico da Figura 6.7

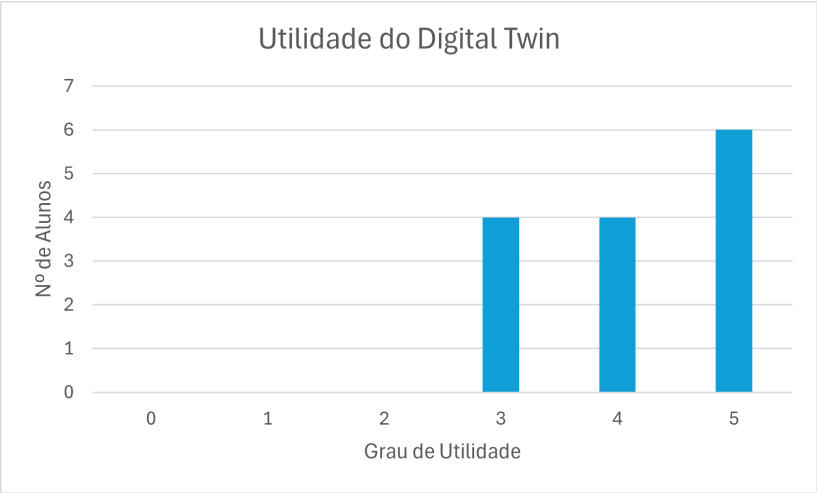


Figura 6.7: Utilidade do Robô Virtual - Nível 2 1a formação

Nenhum aluno considerou a utilidade do software como não útil, sendo que cerca de 28,5% consideraram a atividade normal em termos de utilidade e os restantes 28,5% consideraram a atividade com uma boa utilidade para a preparação do FNR. Por fim, aproximadamente 43% determinou que a utilização desta ferramenta é muito benéfica para testarem e produzirem o seu código para o First Challenger.

Já nos alunos do Nível+ a classificação dada para a dificuldade do software, Figura 6.8, e da atividade, Figura 6.9, revela que os alunos ao trabalharem com uma linguagem de programação considerada normal apresentam maior à vontade na utilização do software e consequentemente realizaram a atividade com menos dúvidas e paragens.

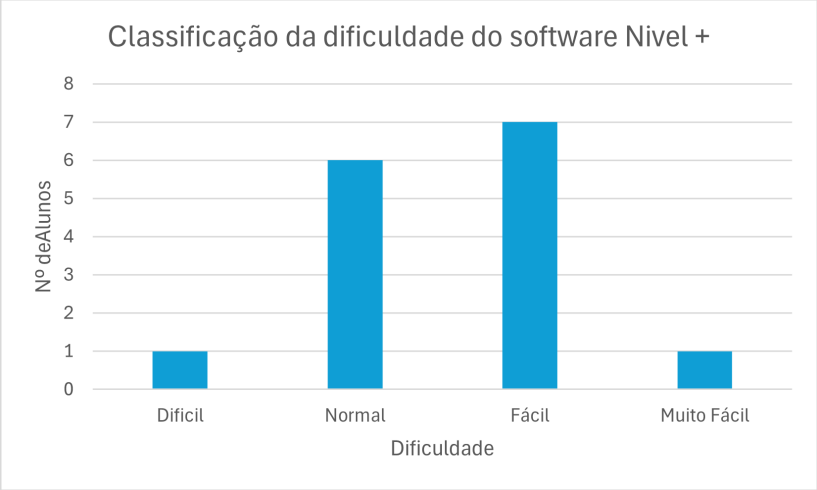


Figura 6.8: Dificuldade sentida na utilização do software Webots - Nível+

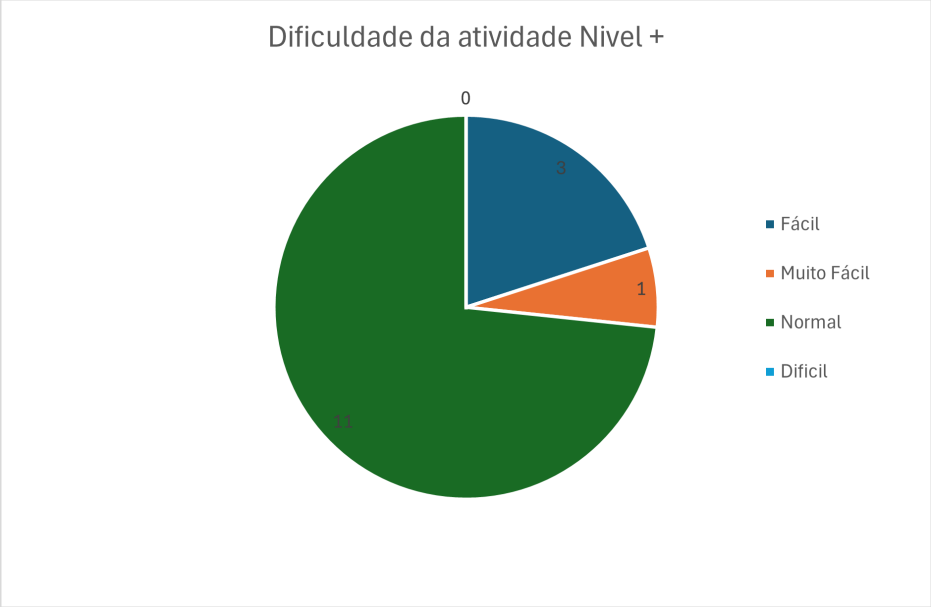


Figura 6.9: Nível de dificuldade sentida durante a formação - Nível+

Um dos alunos revelou dificuldade em utilizar o software Webots nomeadamente nos fatores de manipulação das ferramentas de visualização, mas ao mesmo tempo um aluno considerou as bastante intuitivas. É possível afirmar que esta dificuldade poderá estar associada à idade e também ao nível de literacia digital que cada

aluno tem. Nesta turma 73% dos alunos consideraram a atividade com um grau de dificuldade normal e também quase 27% entenderam a utilização do Webots como um grau de dificuldade baixo.

Neste nível os alunos também consideraram sempre a utilidade do simulador entre os níveis normal até muito útil, Figura 6.10.

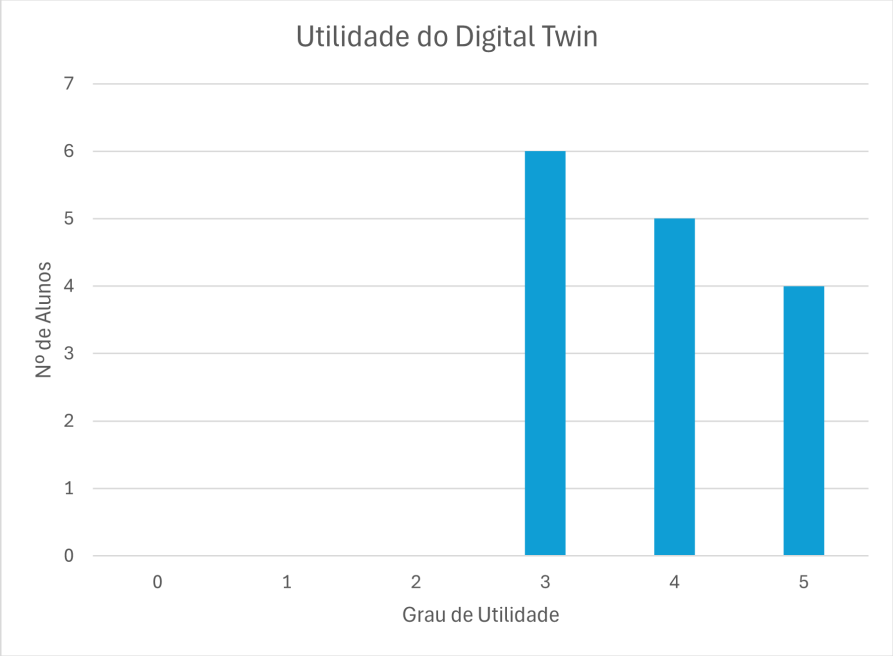


Figura 6.10: Utilidade do Robô Virtual - Nível+ 1a formação

Apesar de considerarem a atividade relativamente acessível quase metade dos alunos consideraram a sua utilidade como normal e 60% classificaram o software como muito útil, querendo utilizá-lo para aprimorar o código e até explorar outros algoritmos para realizar a prova.

No final do ano letivo e após o Bootcamp decidiu-se realizar uma terceira aula, uma vez que se verificou uma grande dificuldade pelos vários grupos de alunos ao explorarem e alcançarem o Nível 3 da prova. Por isso foi preparado um código com o nível 2 completo e com a configuração dos sensores ultrassónicos, onde foi explicada a diferença entre as bibliotecas. Assim os alunos tiveram a oportunidade de explorar diferentes algoritmos para o robô seguir as paredes. A última aula realizou-se durante o período de férias do ensino regular, como tal muitos alunos faltaram e as turmas tiveram menos alunos do que nas primeiras formações. Ao estar presente um núcleo mais pequeno notou-se uma grande diferença na progressão da aula e no tempo para alcançar os objetivos, pois os formadores resolverem e explicarem os vários problemas que foram surgindo de forma mais rápida, visto que conseguiram dar um apoio mais próximo aos alunos.

Na turma Nível 2 estiveram presentes onze alunos e na turma Nível + nove alunos, mantendo ambas as turmas bastante heterogéneas, Figura 6.11 e Figura 6.12. Desse modo, foi possível manter a qualidade da amostra para estes resultados.

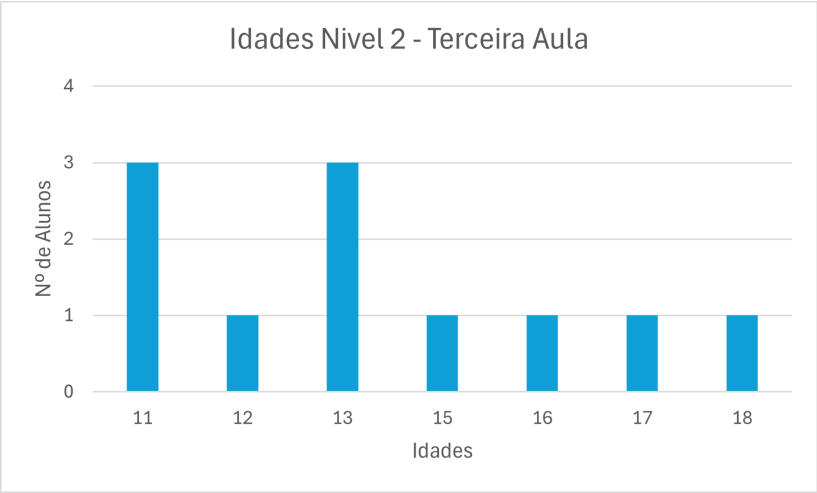


Figura 6.11: Distribuição das idades dos alunos Nível 2 na última aula experimental.

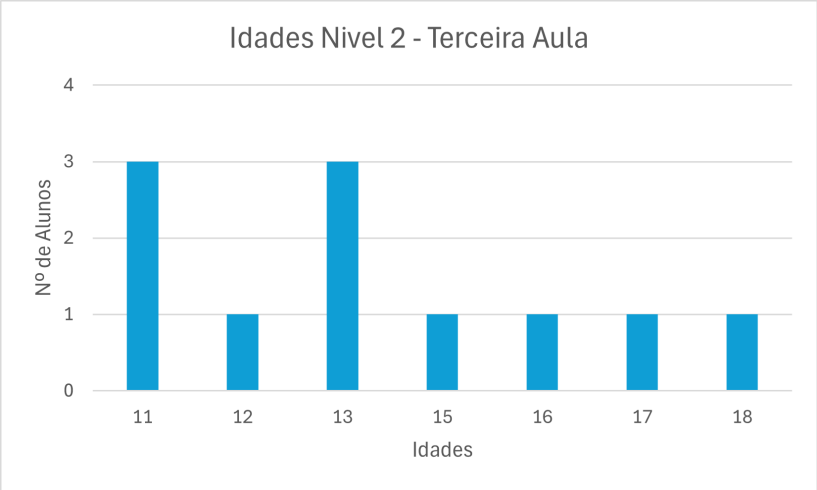


Figura 6.12: Distribuição das idades dos alunos Nível + na última aula experimental.

Esta terceira aula apostou fundamentalmente na demonstração, havendo assim pouca exposição por parte do formador, uma vez que os alunos já tinham lecionado todos os componentes do robô que vão necessitar de utilizar durante esta aula. No inicio da aula fez-se uma revisão de como utilizar o Webots e o restante tempo foi dedicado à programação do robô para seguir as paredes. A parte demonstrativa da aula só foi utilizada para alcançar um código preliminar, evoluindo assim para uma dinâmica ativa deixando os alunos que alcançaram o objetivo experimentarem mais soluções e, ainda se sobrasse tempo passar esses procedimentos para o robô real.

Na Figura 6.13 é visível um aluno Nível + que desenvolveu um algoritmo para seguimento de parede no Webots, transferiu-o para o Arduino IDE/robô real e também experimentou a viabilidade do código.

Figura 6.13: Aluno do Nível+ a testar o algoritmo de seguimento de parede.

Após a conclusão da formação foi realizado, novamente nos dois níveis, o mesmo questionário realizado na primeira fase, de forma a verificar se houve evolução positiva ou negativa entre aulas e perceber a opinião do aluno em relação ao uso do robô simulado do real da Academia em contexto de sala de aula.

Concluiu-se que a maioria dos alunos do Nível 2, Figura 6.14 continuou a considerar a atividade acessível ou até muito fácil de execução, porém nesta terceira aula houve um aluno que considerou a atividade muito difícil.

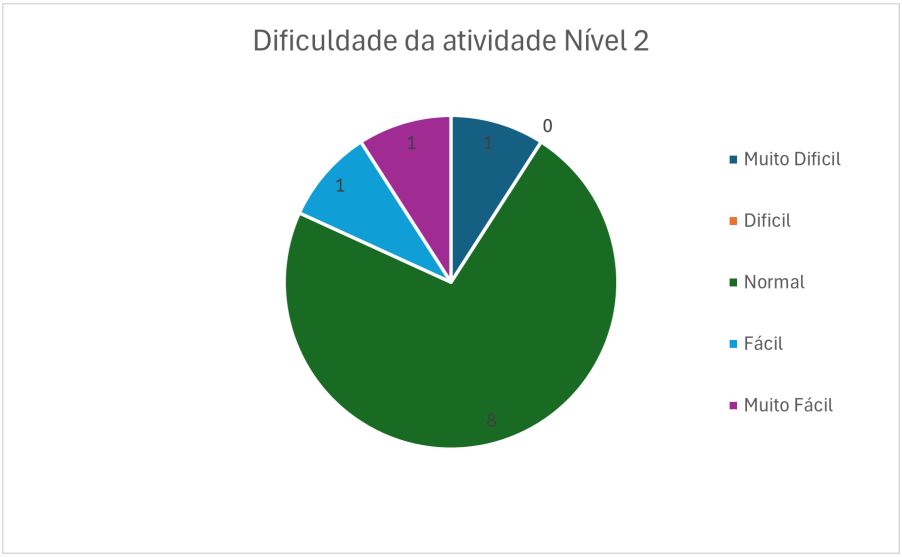


Figura 6.14: Classificação dificuldade Nível 2 última formação

Esta opinião pode-se justificar pelo facto do aluno em questão não ter participado nas primeiras aulas, tendo assim mais dificuldades na execução da aula. No entanto

com a ajuda dos formadores conseguiu alcançar o resultado final de colocar o robô a executar o Nível 3 do First Challenger.

Os alunos quando questionados sobre a utilidade do robô virtual, Figura 6.15, em geral, as opiniões mantiveram-se iguais aos primeiros resultados, no qual o simulador foi considerado útil para a preparação do Festival Nacional de Robótica. O aluno que apresentou muitas dificuldades considerou a sua utilidade nula, talvez por o aluno ao longo da formação ter revelado algumas frustrações no desenvolvimento das tarefas, contribuindo assim para uma opinião negativa.

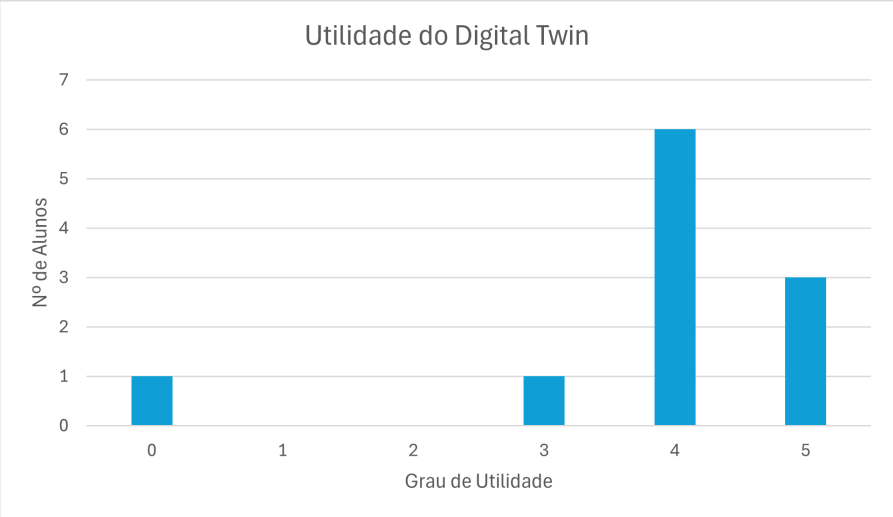


Figura 6.15: Classificação utilidade do Webots - Nivel 2 terceira aula

Já no Nível+ os alunos mantiveram a opinião sobre a dificuldade da utilização do software Webots e da atividade em geral, Figuras 6.16 e 6.17. Nenhum aluno as considerou difíceis e a maior parte avalia-a com um grau de dificuldade normal ou fácil.

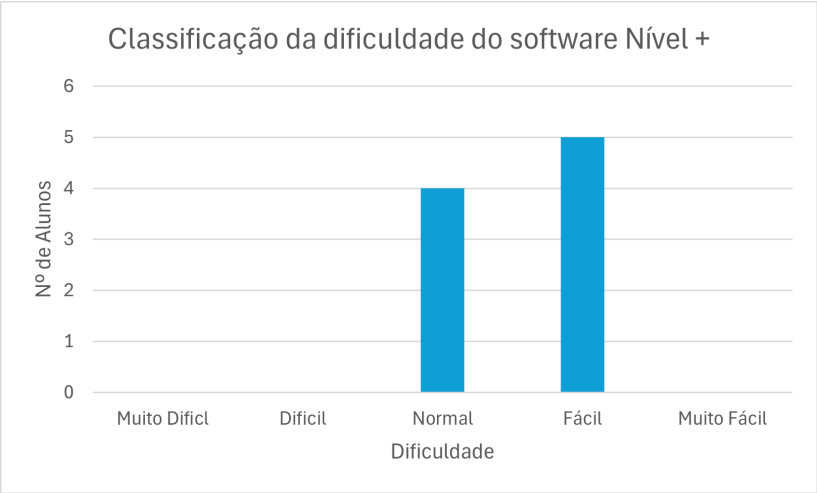


Figura 6.16: Classificação dificuldade sentida por parte dos alunos Nível +

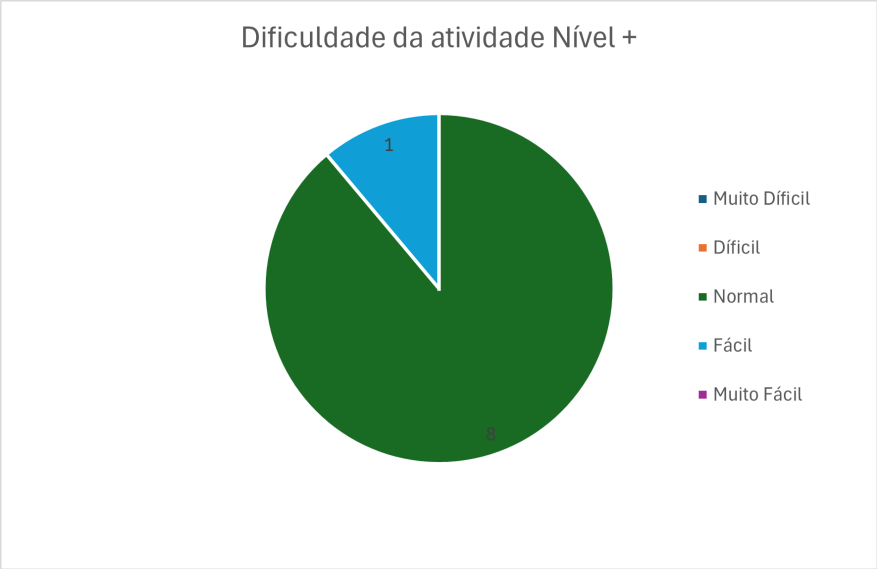


Figura 6.17: Classificação da dificuldade da formação por parte dos alunos Nível +

Em relação à utilidade a opinião do Nível+ também não se alterou, mantendo a média no grau bastante útil, Figura 6.18, para a preparação do First Challenger e, também para o decorrer das aulas. Mais de 66% considerou esta ferramenta de grande valor o que é um resultado muito positivo para a validação deste software.

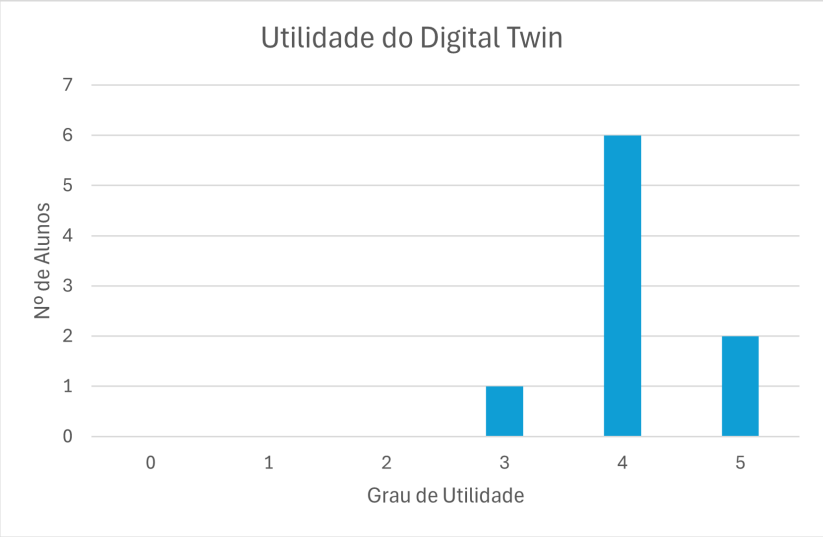


Figura 6.18: Classificação da utilidade do software por parte dos alunos Nível +.

Para verificar se a dificuldade sentida ao longo das várias aulas tinha correlação com a idade dos alunos foi produzido o gráfico da Figura 6.19.

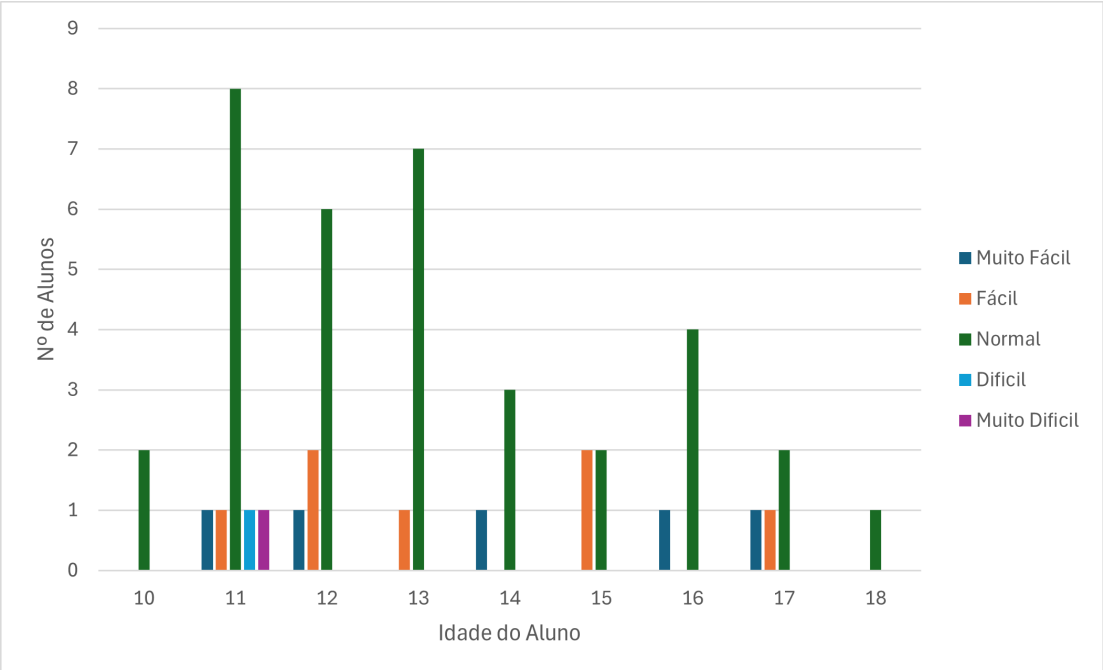


Figura 6.19: Correlação entre dificuldades sentidas e idades dos alunos.

Através deste gráfico pode-se verificar que os alunos que identificaram a atividade como difícil ou muito difícil têm uma das idades mais baixas do espectro geral dos alunos, entre os dez e onze anos. Ao mesmo tempo, em média, os alunos dessa idade consideraram a atividade como normal. Estes resultados podem derivar de vários fatores que não estão relacionados com o software/atividade em si. Uma das grandes causas, não só nesta experiência mas também durante o curso todo de robótica, são as diferenças de literacia digital que se podem observar nos vários alunos. Dependendo muito do panorama escolar e também da educação sobre os meios digitais estes são dois fatores que podem influenciar o modo como estes alunos resolvem os diferentes problemas. A literacia digital é um dos obstáculos que as atividades STEM tentam combater, por isso a utilização de um software em paralelo com experiências reais torna as possibilidades de sucesso muito mais elevadas.

Capítulo 7

Conclusões

A implementação das áreas de Ciência, Tecnologia, Engenharia e Matemática (STEM) no currículo das escolas em todo o mundo tem demonstrado ser uma estratégia fulcral na preparação dos jovens para um mundo profissional cada vez mais dinâmico e complexo. A integração de robôs nas salas de aula não só enriquece o ensino das STEM, mas também contribui para o desenvolvimento de competências pessoais, nomeadamente a resolução de problemas, o pensamento crítico e o trabalho em equipa, valências essas que são altamente valorizadas no mercado de trabalho.

Contudo, a adoção desta metodologia enfrenta desafios significativos, principalmente, associados aos custos na aquisição e manutenção de equipamentos de robótica nas escolas. A proposta de desenvolver um ambiente simulado para a competição First Challenger surge como uma solução inovadora e viável para mitigar esses custos. Esta abordagem permite aos alunos aprender e testar as suas ideias e soluções, reduzindo a necessidade de aquisição de componentes físicos e, consequentemente, dos custos relacionados.

Assim foram realizadas as provas do First Challenger no simulador Webots para que o aluno possa desenvolver e testar o próprio código antes de implementá-lo no robô real. Como o simulador tenta imitar a realidade, o aluno pode utilizar o mesmo código para o robô físico e para o robô virtual. Este processo foi testado em três aulas experimentais que permitiram comprovar a viabilidade e utilidade do uso deste robô simulado. Os alunos presentes nas várias atividades revelaram uma grande facilidade de uso da ferramenta, conseguindo, com o mesmo código, alcançar com sucesso o objetivo de cada nível da prova. Ao produzir os diferentes códigos

nos respetivos *softwares* e posteriormente migrados para o Webots permite que o aluno teste, numa primeira fase, num ambiente de simulação e ao obter resultados positivos, consiga transferir o mesmo código para o robô real. Este processo reduz o tempo de testes e calibração do código tornado-o mais eficiente. É de notar que os alunos consideraram todo este processo de integração do Webots na preparação do First Challenger/Bootcamp bastante intuitivo e de fácil compreensão, pelo que pode-se avaliar esta solução como um meio positivo a introduzir nas aulas de robótica.

Apesar de os resultados não serem suficientes para comprovarem a diminuição de custos, pode-se concluir que a aplicação deste robô virtual irá contribuir para a redução dos mesmos, uma vez que o aluno terá uma fase de testes que não influencia o seu robô físico, diminuindo assim o risco de danos provocados por códigos realizados de forma incorreta ou falta de conhecimento para o manuseamento dos equipamentos.

7.1 Trabalho Futuro

Os alunos ao serem questionados sobre o que gostariam de ver a ser modificado ou acrescentado neste ambiente simulado referiram dois aspetos de forma quase unânime. O primeiro aspeto diz respeito aos gráficos visuais do simulador, que nas suas opiniões não são totalmente idênticos ao robô montado nas aulas. O Webots permite o carregamento de ficheiros desenvolvidos em CAD, por isso como trabalho futuro é proposto a modificação e alteração do visual do robô para que se aproxime do real. O segundo aspeto referenciado pelos alunos relaciona-se com a criação de um maior número de desafios com diversas pistas/ambientes e robôs com diferentes atuadores/sensores. Um das características do uso de um simulador é a versatilidade na modelação/recriação de robôs e ambientes onde estes vão atuar. Deste modo, propõe-se a exploração de um maior leque de desafios no decorrer do curso de robótica.

Por último propõe-se o desenvolvimento de uma solução *hardware-in-loop* para aumentar a eficácia e abrangência da utilização deste simulador. Este método permitirá que os alunos integrem e testem componentes reais juntamente com a simulação, proporcionando uma experiência de aprendizagem mais completa e realista. O *hardware-in-the-loop* ajudará os alunos a compreender melhor a interação entre o software e o hardware, preparando-os de forma mais eficaz para desafios reais e, consequentemente, contribuir para o aumento da literacia digital.

Referências

[1] G. of Western Australia, "What is stem?." https://www.education.wa.edu.au/what-is-stem. [Citado na página 1]

[2] A. de Robótica, "Academia de robótica." https://www.academiarobotica.pt. [Citado nas páginas 2 e 26]

[3] MIT Media Lab, "Scratch: Imagine, program, share." https://scratch.mit.edu/. Accessed: 2024-07-03. [Citado na página 2]

[4] S. P. de Robótica, "Festival nacional de robótica." http://www.sprobotica.pt/index.php?option=com_content&view=article&id=108&Itemid=62. [Citado na página 2]

[5] Y. Li, K. Wang, Y. Xiao, and J. E. Froyd, "Research and trends in stem education: A systematic review of journal publications," *International Journal of STEM Education*, vol. 7, no. 1, 2020. [Citado na página 5]

[6] C. C. Johnson, "Implementation of stem education policy: Challenges, progress, and lessons learned," *School Science and Mathematics*, vol. 112, no. 1, p. 45–55, 2012. [Citado na página 5]

[7] L. N. S. R. M. D. M. P. M. . M. Skweres, *Teachers' Reasons to Join a Community About Educational Robotics and STEAM: A Swiss Experience*, pp. 179–1189. 05 2021. [Citado na página 6]

[8] S. Haesen and E. Van de Put, *STEAM Education in Europe: A Comparative Analysis Report*. 1 ed., 2018. [Citado na página 6]

[9] P. Simões, "Salas de aula do futuro estão a crescer em portugal," *Eco News*, 2021. [Citado na página 6]

[10] Dorothy, "Future classroom lab learning zones." https://edufor.pt/icl/documentos/FCL_LearningZones-Description_ING.pdf, June 2016. European Schoolnet, 2013. [Citado nas páginas vii e 6]

[11] D. Peykova and K. Garov, "Educational robotics for stem education," 10 2021. [Citado na página 7]

[12] W. are the Makers, "Scuola di robotica." https://www.wemakers.eu/partners/scuola-di-robotica-english/. [Citado na página 7]

[13] S. Di Battista, M. Pivetti, B. Simaku, G. Beraldo, E. Menegatti, and M. Moro, *Educational Robotics Acceptance by Italian Teachers, Educators, Psychologists and Psychotherapists*, pp. 167–178. 05 2021. [Citado na página 7]

[14] F. Gratani, L. Giannandrea, A. Renieri, and M. Annessi, *Fostering Students' Problem-Solving Skills Through Educational Robotics in Primary School*, pp. 3–14. 05 2021. [Citado na página 7]

[15] LEGO®, "Mindstorms®." https://www.lego.com/en-pt/themes/mindstorms/about. [Citado na página 8]

[16] LEGO Group, "LEGO MINDSTORMS Robot Inventor 51515." https://www.lego.com/en-pt/product/robot-inventor-51515, 2024. Accessed: 2024-07-03. [Citado nas páginas vii e 8]

[17] Dobot, "Creating robot automation of the future." https://en.dobot.cn/about/about-dobot. Last accessed in 06/10/2022. [Citado nas páginas vii, 8 e 9]

[18] Makeblock, "About us: Makeblock - global steam education solution provider." https://www.makeblock.com/about. [Citado nas páginas vii, 9, 10 e 25]

[19] L. Kelion, "Micro bit mini-computer heads overseas." https://www.bbc.com/news/technology-37682405, Oct 2016. [Citado na página 10]

[20] Micro:bit Educational Foundation, "Meet the new micro:bit — micro:bit." https://microbit.org/new-microbit/, 2024. Accessed: 2024-09-02. [Citado nas páginas vii e 10]

[21] Google, "Introduction to blockly." https://developers.google.com/blockly/guides/overview. [Citado nas páginas vii e 11]

[22] S. Tselegkaridis and T. Sapounidis, "Simulators in educational robotics: A review," *Education Sciences*, vol. 11, p. 11, 01 2021. [Citado nas páginas 11, 12 e 14]

[23] F. Initiative, "Learning with robots – learning to program in a playful way." https://www.roberta-home.de/en/. [Citado nas páginas vii, 12 e 13]

[24] Aldebaran, "Nao the humanoid and programmable robot: Softbank robotics." https://www.aldebaran.com/en/nao. [Citado na página 12]

[25] V. Robotics, "Vexcode virtual robots (vr)." https://www.vexrobotics.com/vexcode/vr. [Citado nas páginas vii e 13]

[26] Open Source Robotics Foundation, "Gazebo." `https://gazebosim.org/home`, 2024. Accessed: 2024-07-03. [Citado na página 14]

[27] P. Quaresma, "Simtwo." `https://github.com/P33a/SimTwo`, 2024. Accessed: 2024-07-03. [Citado nas páginas vii e 14]

[28] A. Abreu, *Autopilot Simulator Prototype for Autonomous Driving based on SimTwo*. PhD thesis, 2020. [Citado na página 14]

[29] Webots, "http://www.cyberbotics.com." Open-source Mobile Robot Simulation Software. [Citado nas páginas viii, 14 e 41]

[30] Festival Nacional de Robótica, "Festival nacional de robótica 2023." `https://www.festivalnacionalrobotica.pt/2023/`, 2023. Accessed: 2024-09-02. [Citado na página 15]

[31] RoboCupJunior, "Robocupjunior rescue league simulation." `https://junior.robocup.org/robocupjuniorrescue-league-simulation/`, 2024. Accessed: 2024-09-02. [Citado nas páginas vii e 15]

[32] Festival Nacional de Robótica, "Robot factory lite - festival nacional de robótica 2023." `https://www.festivalnacionalrobotica.pt/2023/robot-factory-lite/`, 2023. Accessed: 2024-09-02. [Citado na página 16]

[33] A. Dias and V. Cerqueira, "Regras prova junior first challenger v1.5c." `https://www.festivalnacionalrobotica.pt/2023/wp-content/uploads/2023/03/Regras_Prova_JuniorFirstChallengerV1.5c.pdf`, 2023. Accessed: 2024-07-08. [Citado na página 18]

[34] André Dias, *Regras Prova Junior First Challenger*, Mar. 2023. Accessed: 2024-09-02. [Citado nas páginas vii e 18]

[35] ElecFreaks, *Ultrasonic Ranging Module HC-SR04*, May 2011. Accessed: 2024-07-08. [Citado nas páginas vii, 19 e 20]

[36] Robotparts, "Sonar hc-sr04 – sensor distância." `https://robotparts.pt/produto/hc-sr04-sensor-distancia-ultrassons/`, Jul 2023. [Citado nas páginas vii e 20]

[37] I. Vishay Intertechnology, "Tcrt5000 datasheet." `https://www.vishay.com/docs/83760/tcrt5000.pdf`, 2018. Accessed: 2024-07-08. [Citado na página 20]

[38] Robotparts, "Sensores." `https://robotparts.pt/categoria-produto/sensores/`, 2024. Accessed: 2024-07-08. [Citado na página 21]

[39] Robotparts, "5x tcrt5000 sensor de pista." https://robotparts.pt/produto/5x-tcrt5000-sensor-de-pista/, 2024. Accessed: 2024-07-08. [Citado na página 21]

[40] TAOS Inc., "TCS3200 Datasheet." https://www.mouser.com/catalog/specsheets/tcs3200-e11.pdf, 2011. Accessed: 2024-07-08. [Citado na página 21]

[41] Hobby 'n' Hobby, "TCS3200 Color Sensor." https://www.hnhcart.com/blogs/sensors-modules/tcs-3200-color-sensor, 2024. Accessed: 2024-07-08. [Citado nas páginas vii e 22]

[42] Robu.in, "Working Principle of DC Motor." https://robu.in/working-principle-of-dc-motor/, 2024. Accessed: 2024-07-08. [Citado na página 22]

[43] D. Nedelkovski, "How to use a rgb led with arduino." https://howtomechatronics.com/tutorials/arduino/how-to-use-a-rgb-led-with-arduino/, 2024. Accessed: 2024-07-08. [Citado nas páginas vii, 23 e 24]

[44] Geek Factory, "LED RGB 5 mm de Diámetro Anodo o Cátodo Comum." https://www.geekfactory.mx/producto/led-rgb-5-mm-de-diametro-anodo-o-catodo-comun/, 2024. Accessed: 2024-07-08. [Citado nas páginas vii e 23]

[45] Picobricks, "What is Buzzer?." https://picobricks.com/blogs/info/what-is-buzzer, 2024. Accessed: 2024-07-08. [Citado na página 24]

[46] ABC Escolar, "Buzzer Ativo 5V." https://abcescolar.pt/products/buzzer-ativo-5v, 2024. Accessed: 2024-07-08. [Citado nas páginas vii e 24]

[47] Makeblock, "What is mBlock 5?." https://support.makeblock.com/hc/en-us/articles/4408619146519-What-is-mBlock-5, 2024. Accessed: 2024-07-08. [Citado na página 25]

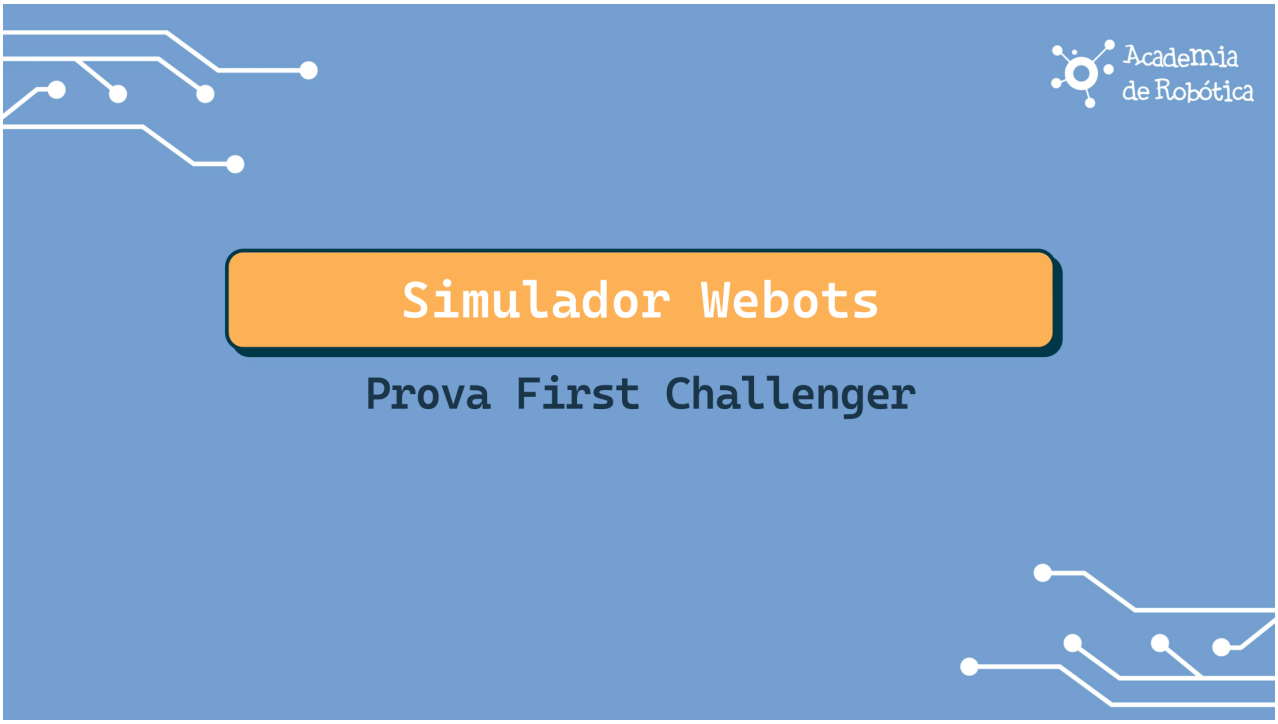
[48] Massachusetts Institute of Technology, "Title of the Webpage." https://www.mit.edu/. Accessed: 2024-07-08. [Citado na página 25]

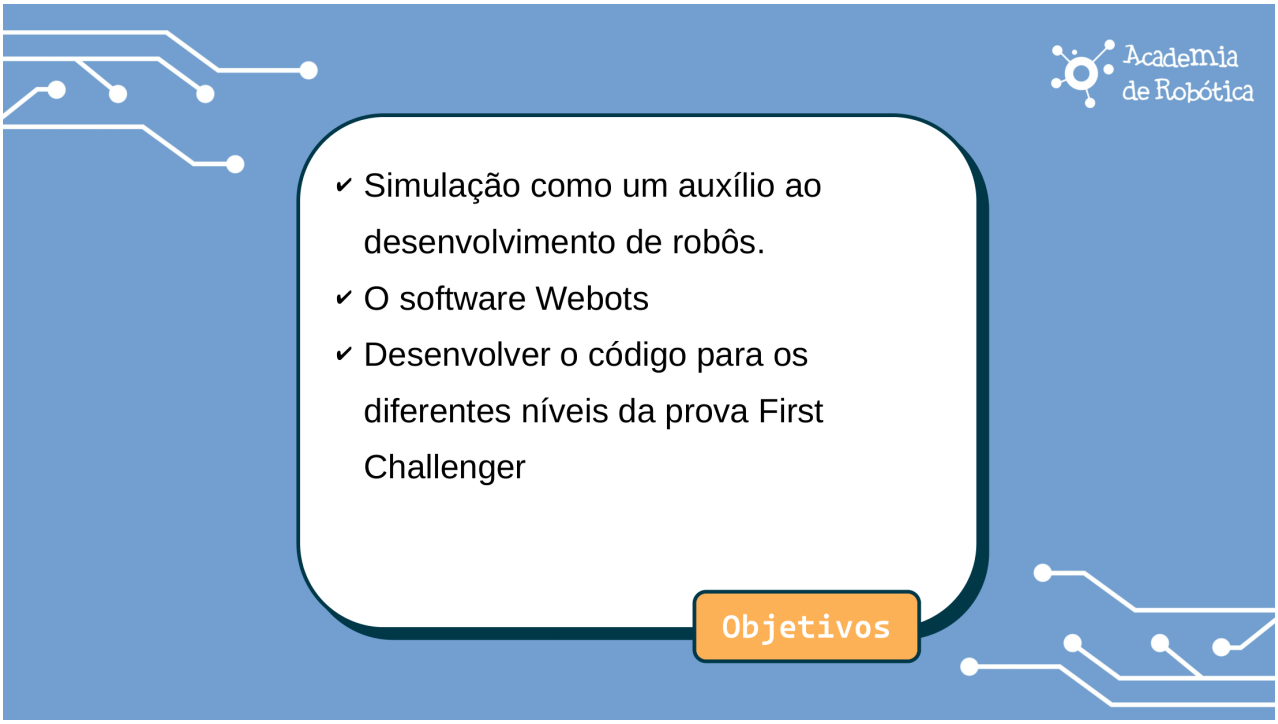
[49] T. Arduino, "About arduino." https://www.arduino.cc/en/about. [Citado na página 26]

[50] MakeCode, "mblock - one-stop coding." https://www.mblock.cc/en/. [Citado na página 26]

Anexo A

Documento de Auxílio à Formação com base no simulador da Prova First Challenger

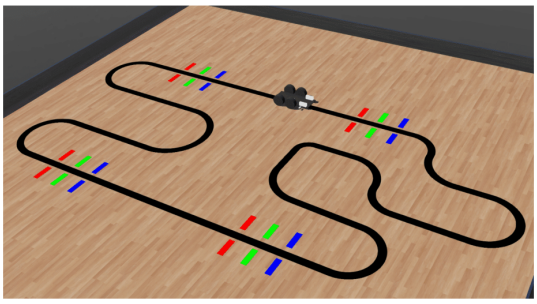

Academia de Robótica
Simulador Webots
Prova First Challenger

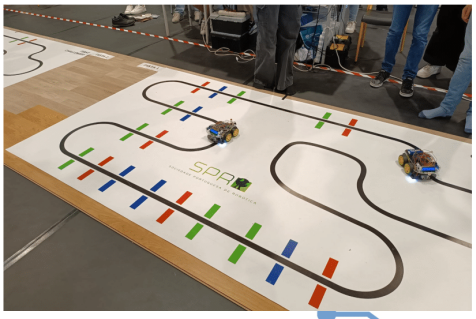

Academia de Robótica
✔ Simulação como um auxílio ao desenvolvimento de robôs.
✔ O software Webots
✔ Desenvolver o código para os diferentes níveis da prova First Challenger
Objetivos

Simulação

Academia de Robótica

- Imitação/recriação do mundo real





Simulação Vantagens

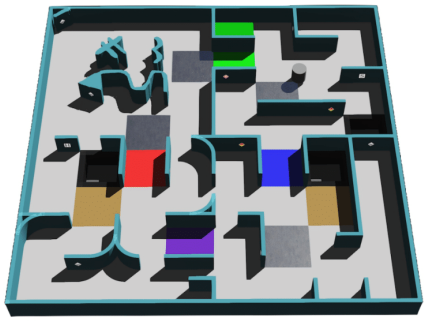
Academia de Robótica

- 1ª – Agilidade no momento do desenvolvimento
- 2ª – Redução de custos de desenvolvimento
- 3ª – Aumento da segurança e eficácia dos equipamentos

Software Webots

Academia de Robótica

- O software Webots é um simulador de robôs móveis totalmente gratuito
- Utilizado na prova RoboCup Junior Rescue



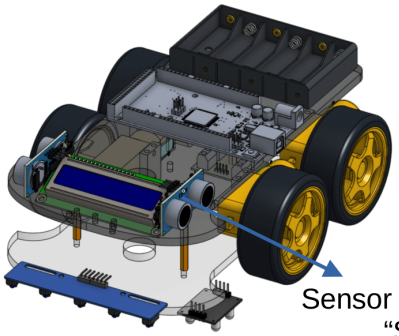
Webots

robot simulation

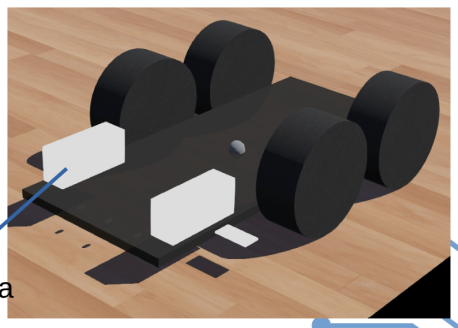
Robô Webots VS Robô Academia

Academia de Robótica

- O robô simulado utiliza os mesmos sensores e atuadores que o robô real da Academia de Robótica



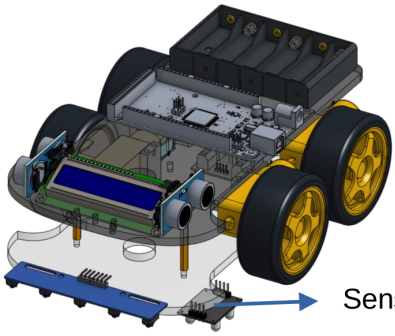
Sensor de distância "Sonar"



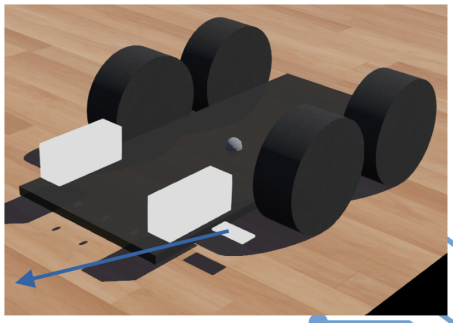
Robô Webots VS Robô Academia

Academia de Robótica

- O robô simulado utiliza os mesmos sensores e atuadores que o robô real da Academia de Robótica



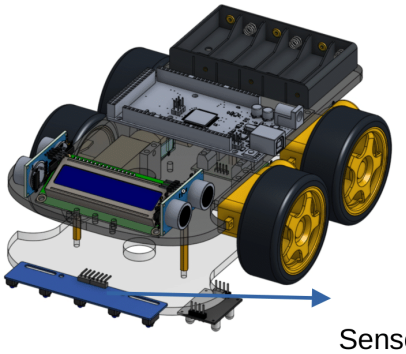
Sensor de Cor



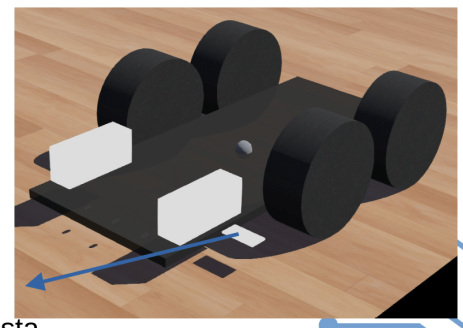
Robô Webots VS Robô Academia

Academia de Robótica

- O robô simulado utiliza os mesmos sensores e atuadores que o robô real da Academia de Robótica



Módulo
Sensores de Pista

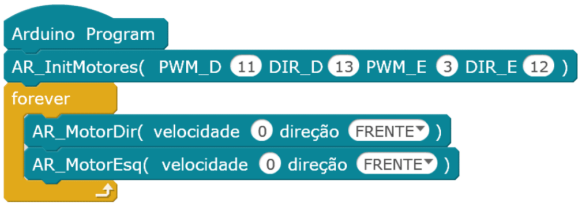


Robô Webots VS Robô Academia

Academia de Robótica

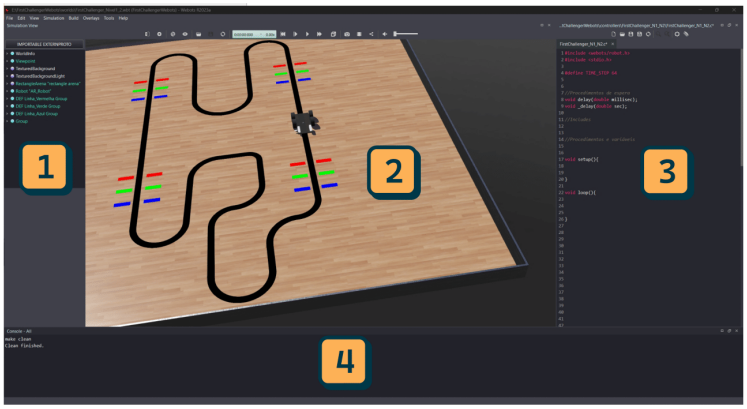
- As mesmas funções são utilizadas para controlo do robô

```
#include "ARSC_Motores.h"

void setup() {

  AR_InitMotores(11,13,3,12);

}

void loop() {

  AR_MotorDir(0,FRENTE);
  AR_MotorEsq(0,FRENTE);
}
```



Software Webots

Academia de Robótica



1. Configurações do mundo e robôs
2. Simulação
3. Controlador (programação)
4. Linha de comando

Software Webots

Academia de Robótica

Menu para controlo da simulação

Compilar o ficheiro

Menu para gravar/localizar ficheiros de controlo

Estrutura do Controlador

Academia de Robótica

```
1 #include <webots/robot.h>
2 #include <stdio.h>
3
4 #define TIME_STEP 64
5
6
7 //Procedimentos de espera
8 void delay(double millisec);
9 void _delay(double sec);
10
11 //Includes
12
13
14 //Procedimentos e variáveis
15
16
17 void setup(){
18
19
20 }
21
22 void loop(){
23
24
25
26 }
```

Inclusão das bibliotecas

Declaração de variáveis e procedimentos/funções

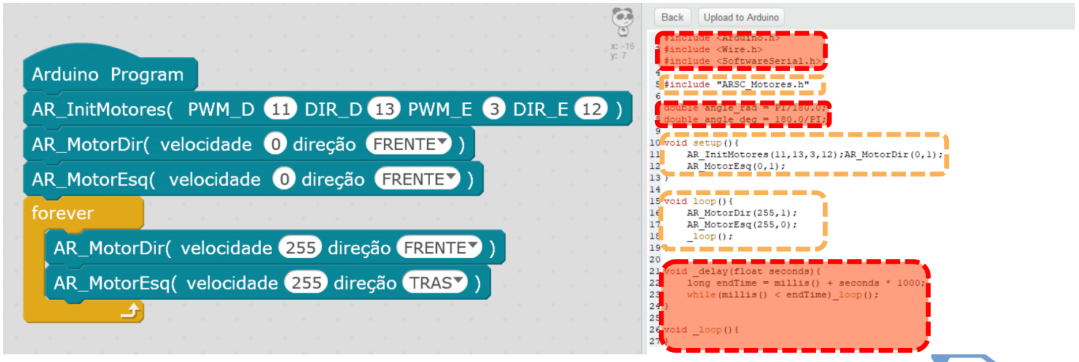
Configurações

Forever

Teste Motores mBlock

Academia de Robótica

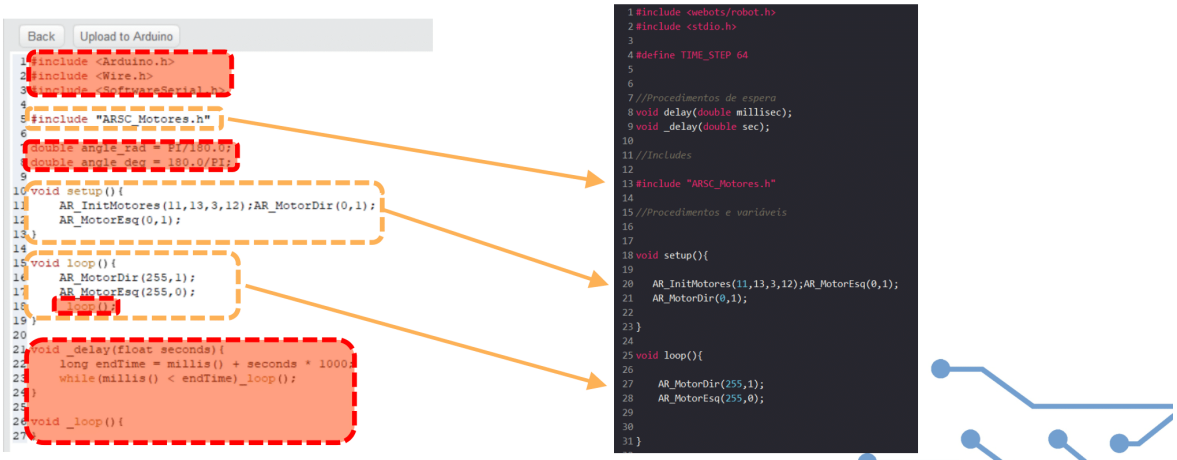
- O que copiar para o webots?

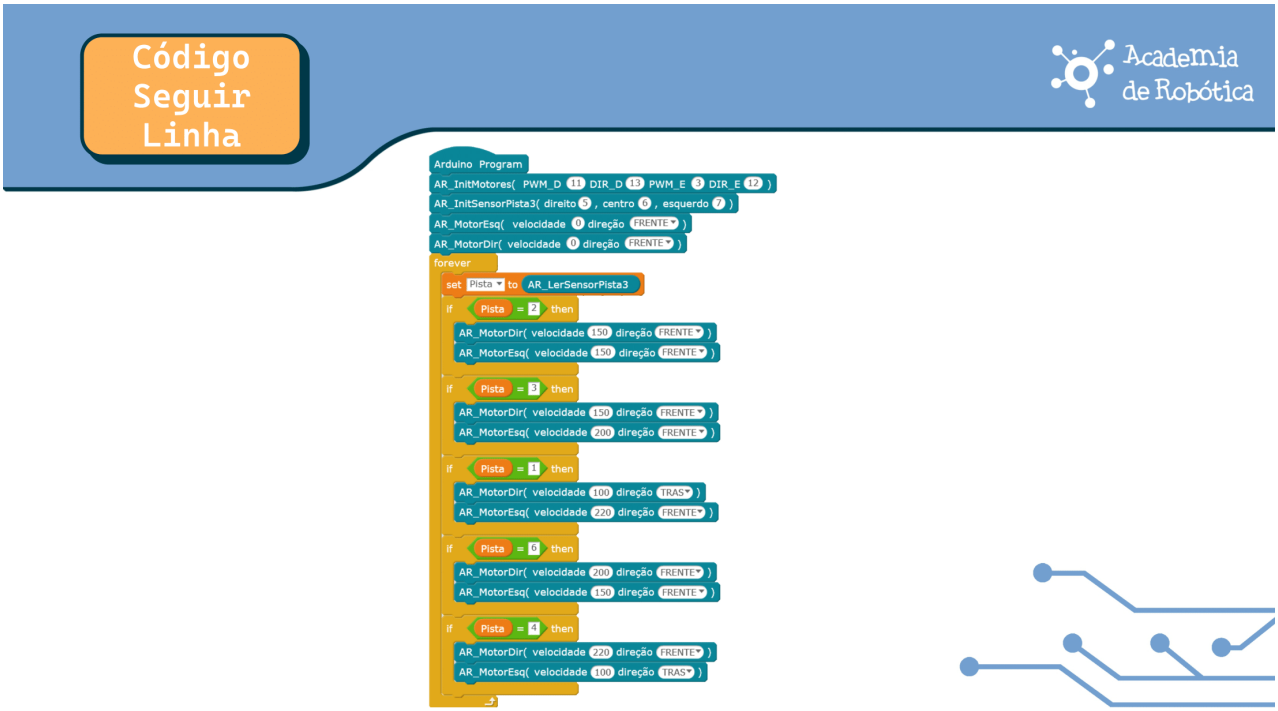


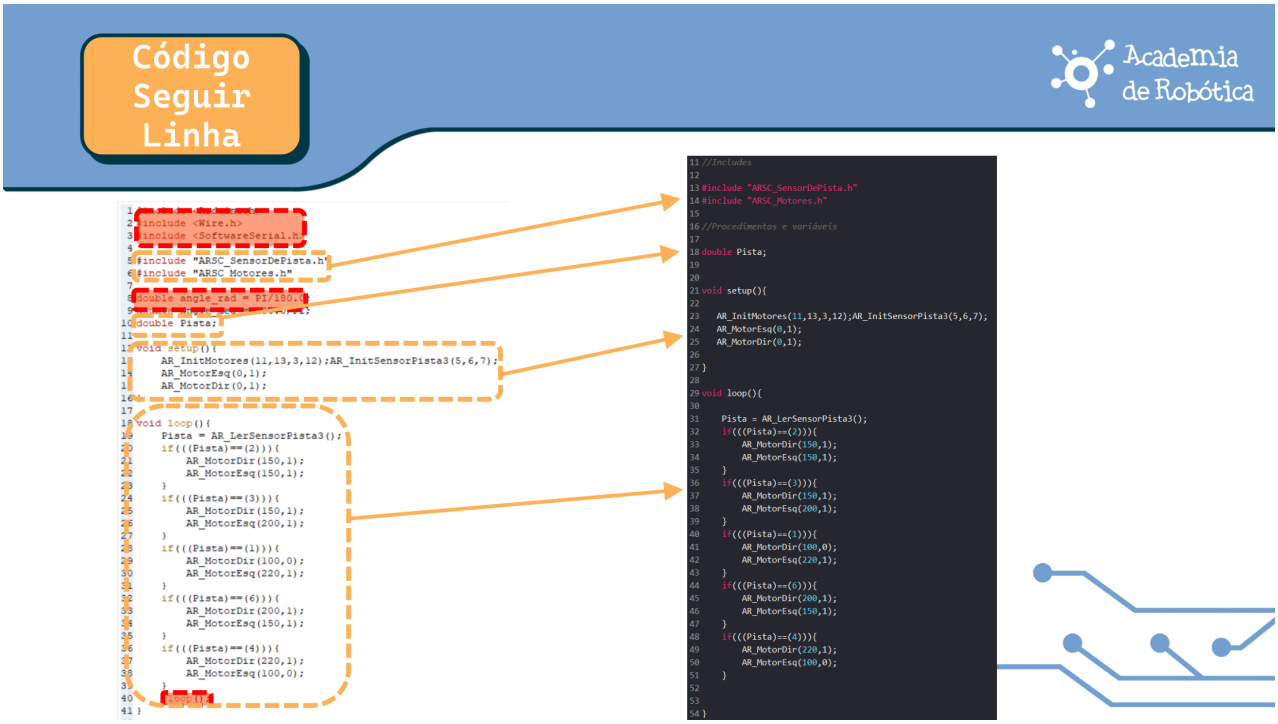
Teste Motores mBlock

Academia de Robótica

- O que copiar para o webots?




Código Seguir Linha
Academia de Robótica
Arduino Program
AR_InitMotores(PWM_D 11 DIR_D 13 PWM_E 3 DIR_E 12)
AR_InitSensorPista3(direito 5 , centro 6 , esquerdo 7)
AR_MotorEsq(velocidade 0 direção FRENTE)
AR_MotorDir(velocidade 0 direção FRENTE)
forever
set Pista to AR_LerSensorPista3
if Pista = 2 then
AR_MotorDir(velocidade 150 direção FRENTE)
AR_MotorEsq(velocidade 150 direção FRENTE)
if Pista = 3 then
AR_MotorDir(velocidade 150 direção FRENTE)
AR_MotorEsq(velocidade 200 direção FRENTE)
if Pista = 1 then
AR_MotorDir(velocidade 100 direção TRAS)
AR_MotorEsq(velocidade 220 direção FRENTE)
if Pista = 6 then
AR_MotorDir(velocidade 200 direção FRENTE)
AR_MotorEsq(velocidade 150 direção FRENTE)
if Pista = 4 then
AR_MotorDir(velocidade 220 direção FRENTE)
AR_MotorEsq(velocidade 100 direção TRAS)


Código Seguir Linha
Academia de Robótica
#include <Wire.h>
#include <SoftwareSerial.h>
#include "ARSC_SensorDePista.h"
#include "ARSC_Motores.h"
double Pista;
void setup(){
AR_InitMotores(11,13,3,12);AR_InitSensorPista3(5,6,7);
AR_MotorEsq(0,1);
AR_MotorDir(0,1);
void loop(){
Pista = AR_LerSensorPista3();
if(((Pista)==(2))){
AR_MotorDir(150,1);
AR_MotorEsq(150,1);
if(((Pista)==(3))){
AR_MotorDir(150,1);
AR_MotorEsq(200,1);
if(((Pista)==(1))){
AR_MotorDir(100,0);
AR_MotorEsq(220,1);
if(((Pista)==(6))){
AR_MotorDir(200,1);
AR_MotorEsq(150,1);
if(((Pista)==(4))){
AR_MotorDir(220,1);
AR_MotorEsq(100,0);
loop();
//Includes
#include "ARSC_SensorDePista.h"
#include "ARSC_Motores.h"
//Procedimentos e variáveis
double Pista;
void setup(){
AR_InitMotores(11,13,3,12);AR_InitSensorPista3(5,6,7);
AR_MotorEsq(0,1);
AR_MotorDir(0,1);
void loop(){
Pista = AR_LerSensorPista3();
if(((Pista)==(2))){
AR_MotorDir(150,1);
AR_MotorEsq(150,1);
if(((Pista)==(3))){
AR_MotorDir(150,1);
AR_MotorEsq(200,1);
if(((Pista)==(1))){
AR_MotorDir(100,0);
AR_MotorEsq(220,1);
if(((Pista)==(6))){
AR_MotorDir(200,1);
AR_MotorEsq(150,1);
if(((Pista)==(4))){
AR_MotorDir(220,1);
AR_MotorEsq(100,0);

Teste Motores Arduino

Academia de Robótica

- O que copiar para o webots?

```
#include "ARSC_Motores.h"

void setup() {

  AR_InitMotores(11,13,3,12);
  AR_MotorDir(0,FRENTE);
  AR_MotorEsq(0,FRENTE);

}

void loop() {

  AR_MotorDir(255,FRENTE);
  AR_MotorEsq(255,TRAS);

}
```

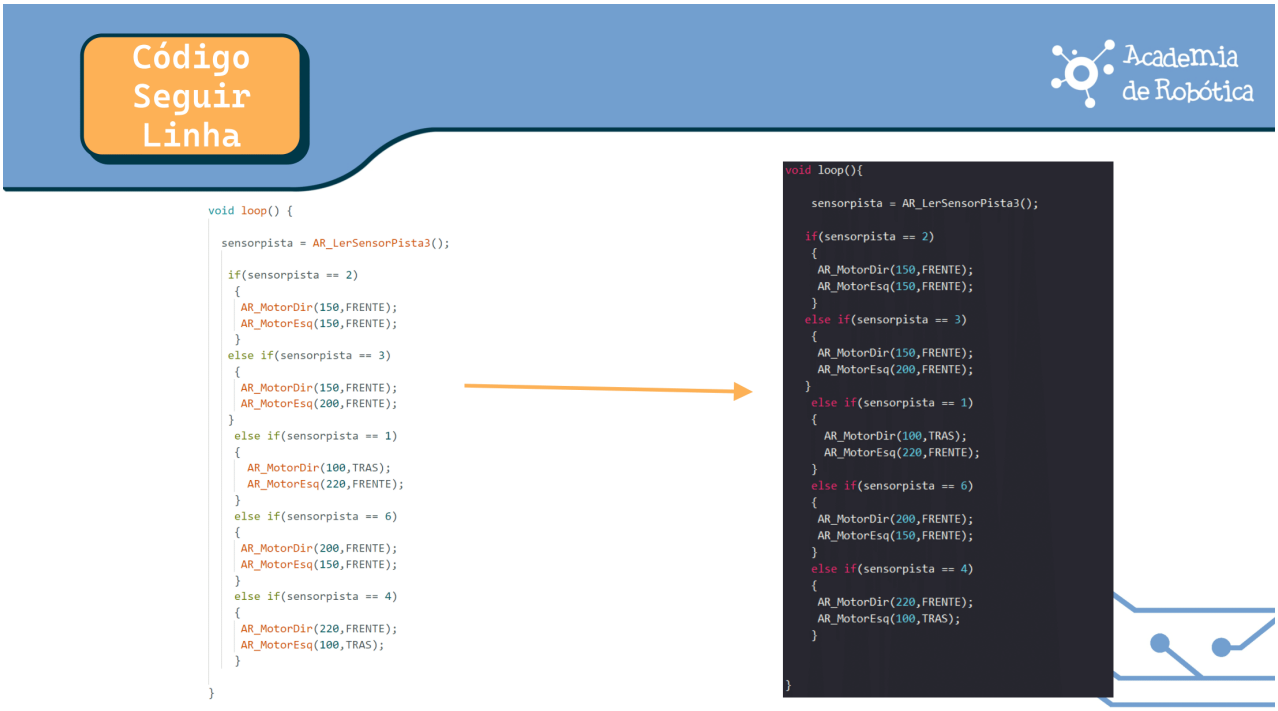
```
#include <webots/robot.h>
#include <stdio.h>

#define TIME_STEP 64


//Procedimentos de espera
void delay(double millisec);
void _delay(double sec);

//Includes

#include "ARSC_Motores.h"

//Procedimentos e variáveis


void setup(){

  AR_InitMotores(11,13,3,12);AR_MotorEsq(0,1);
  AR_MotorDir(0,1);

}

void loop(){

   AR_MotorDir(255,1);
   AR_MotorEsq(255,0);


}
```

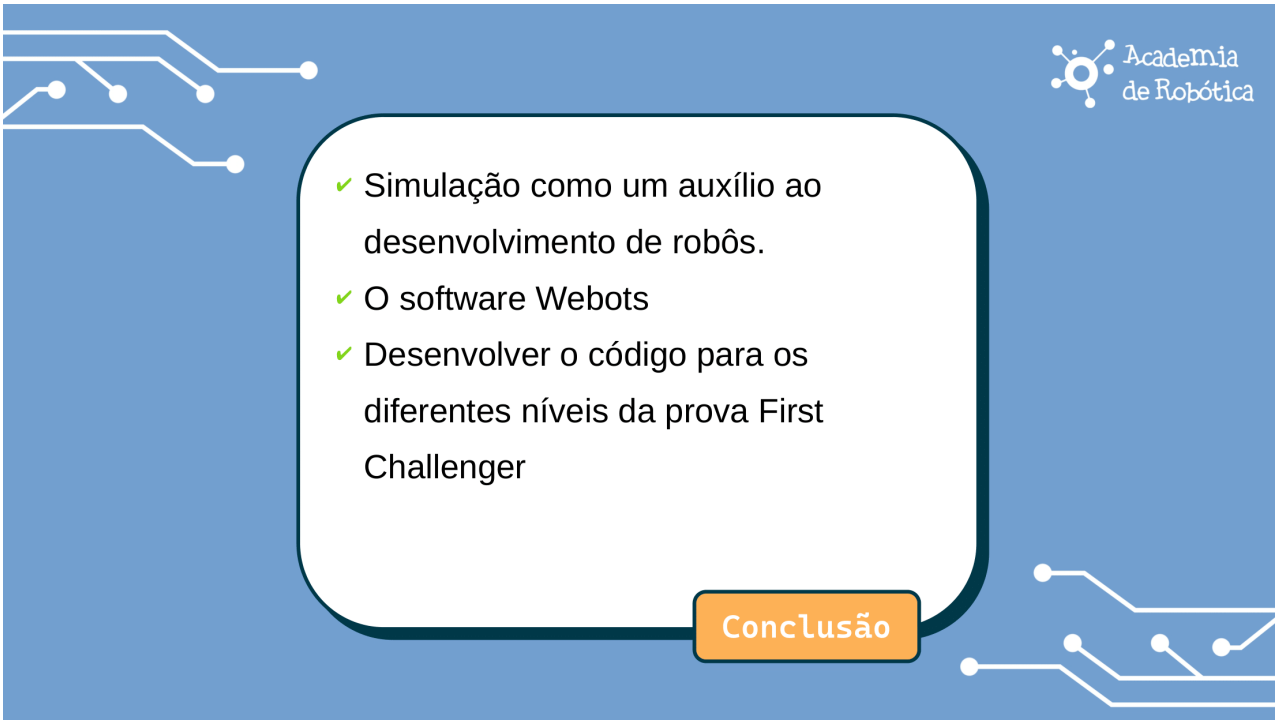
Código Seguir Linha

Academia de Robótica

```
#include "ARSC_Motores.h"
#include "ARSC_SensorDePista.h"

int sensorpista = 0;

void setup() {

  AR_InitMotores(11,13,3,12);
  AR_InitSensorPista3(5,6,7);
  AR_MotorDir(0,FRENTE);
  AR_MotorEsq(0,FRENTE);

}
```

```
//Includes

#include "ARSC_SensorDePista.h"
#include "ARSC_Motores.h"

//Procedimentos e variáveis

int sensorpista = 0;


void setup(){

  AR_InitMotores(11,13,3,12);
  AR_InitSensorPista3(5,6,7);
  AR_MotorDir(0,FRENTE);
  AR_MotorEsq(0,FRENTE);


}
```


Código Seguir Linha
Academia de Robótica
void loop() {
sensorpista = AR_LerSensorPista3();
if(sensorpista == 2)
{
AR_MotorDir(150,FRENTE);
AR_MotorEsq(150,FRENTE);
}
else if(sensorpista == 3)
{
AR_MotorDir(150,FRENTE);
AR_MotorEsq(200,FRENTE);
}
else if(sensorpista == 1)
{
AR_MotorDir(100,TRAS);
AR_MotorEsq(220,FRENTE);
}
else if(sensorpista == 6)
{
AR_MotorDir(200,FRENTE);
AR_MotorEsq(150,FRENTE);
}
else if(sensorpista == 4)
{
AR_MotorDir(220,FRENTE);
AR_MotorEsq(100,TRAS);
}
}
void loop(){
sensorpista = AR_LerSensorPista3();
if(sensorpista == 2)
{
AR_MotorDir(150,FRENTE);
AR_MotorEsq(150,FRENTE);
}
else if(sensorpista == 3)
{
AR_MotorDir(150,FRENTE);
AR_MotorEsq(200,FRENTE);
}
else if(sensorpista == 1)
{
AR_MotorDir(100,TRAS);
AR_MotorEsq(220,FRENTE);
}
else if(sensorpista == 6)
{
AR_MotorDir(200,FRENTE);
AR_MotorEsq(150,FRENTE);
}
else if(sensorpista == 4)
{
AR_MotorDir(220,FRENTE);
AR_MotorEsq(100,TRAS);
}
}


Academia de Robótica
Simulação como um auxílio ao desenvolvimento de robôs.
O software Webots
Desenvolver o código para os diferentes níveis da prova First Challenger
Conclusão


Academia de Robótica
https://forms.gle/DgyPDA1uooE8LCss9
Questionário

Anexo B

Questionário de Avaliação da Formação

Atividade Webots - FirstChallenger

Primeira Aula

Idade *

Short answer text

Nível - Academia de Robótica *

- Nível 1
- Nível 2
- Nível +
- Nível ++

Como classifica a dificuldade de utilização do software Webots? *

- Muito Fácil
- Fácil
- Normal
- Dificil
- Muito Dificil

Como avalia o nível de dificuldade da atividade realizada em aula? *

- Muito Fácil
- Fácil
- Normal
- Dificil
- Muito Dificil

Se considerou a atividade dificil, quais foram as dificuldades? *

Long answer text

Considera útil a utilização deste software para o curso/festival? (0 - Não tem utilidade e 5 - Indispensável) *

- 0
- 1
- 2
- 3
- 4
- 5

Na sua opinião acha que a utilização deste simulador pode ajudar no desenvolvimento do código? *

Long answer text

O que gostaria de acrescentar/melhorar no simulador? *

Long answer text