FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Projeto e Casos de Estudo em Robótica Educativa envolvendo Visão Tempo Real

Valter Costa

Mestrado Integrado em Engenharia Electrotécnica e de Computadores

Orientador: Armando Jorge Miranda de Sousa

Co-orientador: Ana Rosanete Lourenço Reis

28 de Julho de 2015

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

MIEEC - MESTRADO INTEGRADO EM ENGENHARIA ELETROTÉCNICA E DE COMPUTADORES **2014/2015**

A Dissertação intitulada

"Projeto e Casos de Estudo em Robótica Educativa Envolvendo Visão em Tempo Real"

foi aprovada em provas realizadas em 24-07-2015

o júri

Presidente Professor Doutor Paulo José Cerqueira Gomes da Costa
Professor Auxiliar do Departamento de Engenharia Eletrotécnica e de Computadores
da Faculdade de Engenharia da Universidade do Porto

Doutor Luis Pina
Investigador do Instituto de Engenharia Mecânica e Gestão Industrial

Professor Doutor Armando Jorge Miranda de Sousa
Professor Auxiliar do Departamento de Engenharia Eletrotécnica e de Computadores
da Faculdade de Engenharia da Universidade do Porto

O autor declara que a presente dissertação (ou relatório de projeto) é da sua exclusiva autoria e foi escrita sem qualquer apoio externo não explicitamente autorizado. Os resultados, ideias, parágrafos, ou outros extratos tomados de ou inspirados em trabalhos de outros autores, e demais referências bibliográficas usadas, são corretamente citados.

Autor - Valter Joaquim Ramos Costa

Faculdade de Engenharia da Universidade do Porto

Resumo

A robótica é uma área apelativa para ensino e demonstrações. Nesta dissertação são abordados 5 casos de estudo para diversos públicos alvo.

Os 5 casos de estudo incluem plataformas robóticas compradas e adaptadas (i) LEGO® EV3 e o (ii) Pololu 3Pi; plataformas desenvolvidas anteriormente na FEUP tal como (iii) o robô em que o controlo é definido por intermédio de ligações "elétricas" (robô "REDI"), (iv) o robô de condução autónoma (robô "Conde") para a participação no Festival Nacional de Robótica (FNR) e ainda (v) o "robô telemóvel" desenvolvido no âmbito deste trabalho. A nível de definição de regras de controlo, foram exploradas diversas abordagens desde a programação por fios (no robô (iii)), programação imperativa (robôs (i), (ii) e (iv)), programação por blocos (robô (i)), programação por intermédio de FEUPAutom/Grafcet (robôs (i) e (ii)). Foi ainda definida uma abordagem para programação via máquinas de estado utilizando o software da LEGO® NXT-G. O robô "Conde" é apenas estudado ao nível de programação imperativa para permitir futuros desenvolvimentos.

Este trabalho utilizou um sistema de *tracking* de robôs para comparar o desempenho dos diversos casos de estudo em controlo por histerese e linear para o seguimento de linha - tal como aquele que os estudantes mais jovens poderão vir a desenvolver.

Foi desenhada ainda uma nova estratégia de programação dos robôs (i) e (ii) a partir do PC não recorrendo à reprogramação do microcontrolador embebido. Desta forma é possível ver ainda o estado de execução no PC enquanto se avalia o desempenho no mundo do robô, contornando até algumas limitações inerentes ao código executado. A limitação desta estratégia é a dependência de comunicação permanente com cada um dos robôs alvo. Em caso de perda de comunicação, o robô pára. Esta estratégia de controlo do robô à distância é denominada de "programação transparente": os sensores são lidos no robô, enviados para o PC, o código é executado no PC e as ordens de atuação enviadas de novo para o robô. Foram comparados resultados (trajetórias) com programação embebida e com "programação transparente"para a aplicação de seguimento de linha. Os resultados demonstram que o desempenho é perfeitamente adequado para os objetivos do público alvo.

Relativamente ao caso de estudo do robô "Conde" para condução autónoma do FNR, foi desenvolvido um sistema de visão Tempo Real para a identificação de semáforos e pista. São discutidas as decisões e apresentados os resultados obtidos, incluindo o tempo de atraso desde a câmara até ao processamento em PC. Adicionalmente foi desenvolvido um controlador para o movimento deste robô que permitiu participar na competição do FNR 2015 e obter demonstrações simples e apelativas. A arquitetura deste sistema tira proveito do meta sistema operativo ROS (*Robotic Operating System*). Neste quadro de trabalho tira-se o proveito máximo de comunicações normalizadas e espera-se dar azo a futuro reaproveitamento de código e a correta separação das diversas partes de código.

Espera-se assim tornar a Robótica mais fácil e apelativa em situações tal como a Universidade Júnior da FEUP e em escolas secundárias para ensino e demonstrações.

Abstract

Robotics is an interesting area for teaching and demonstration purposes. In this dissertation five case studies are approached for diverse target audience. The five case studies include robotics platforms bought and adapted, such as, (i) Lego EV3 and (ii) Polulu 3Pi; Platforms previously developed in FEUP such as, (iii) REDI robot, a robot that's controlled through intermediate connections, (iv) autonomous conduction Robot (CONDE) that participates every year in the national robotics festival (FNR) and yet (v) the cellphone robot developed in the scope of this work. While defining control rules, several approaches were explored such as the wire programming in (iii), imperative programming in (i), (ii) and (iii), block programming (i), using FEUP Autom/Grafcet (i) and (ii). A state-machine programming approach was defined using the LEGO NXT-G software. CONDE robot is only studied at imperative programming level to allow future developments. This work uses a tracking robot system to match the robots performance through hysteresis control and linear for the line following - such as the ones young students might come and design in the future. A new programming strategy was designed for robots (i) and (ii) using the PC avoiding the embedded micro controller re-programming. This way it is possible to see the task execution state on the PC while evaluating the robot's real world performance, bypassing some limitations inherent to the code executed.

The limitation to this strategy is the dependency of permanent communication with each of the target robots. In case of loss of communication the robot stops. This distance-control strategy is called transparent programming: The sensors state are read in the robot, sent to the PC; the code is executed in the PC and the actuation orders are re-sent to the robot.

Trajectory results are compared between the embedded programming and the transparent programming for the line following application. The results show that the performance is adequate for the target audience objectives. A real-time vision system was developed for the CONDE case study for autonomous driving of the FNR (to identify the semaphores and the track).

Results obtained are presented and discussed, including lag from the camera to PC processing. Additionally, a motion controller was developed in a way that allowed this robot to participate in the FNR 2015. This system's architecture takes advantage of the meta operative system ROS (Robotic Operating System). In the scope of this work, maximum profit is taken between the normalized communications and in the future, the code will be re-used, documented and released so it can be used in other works. This work main goal is to provide another step towards making robotics easier and appealing in situations like the Junior University and in high-schools for teaching and demonstrations.

Agradecimentos

A realização deste trabalho só foi possível graças a todas as pessoas que, direta ou indiretamente, contribuíram para o meu percurso pessoal e académico, tendo-me ajudado a superar este desafio. Às seguintes pessoas gostaria de deixar um agradecimento especial:

Ao meu orientador Prof. Armando Jorge Miranda de Sousa pelo seu indispensável apoio, pela simpatia demonstrada, pelo conhecimento transmitido e pelo incentivo prestado, sobretudo nas alturas de maior frustração.

Ao meu co-orientador Prof. Ana Rosanete Lourenço Reis pela sua constante disponibilidade quer para a transmissão de informações quer para a passagem do material necessário à realização desta tese.

Aos meus amigos, que me acompanharam ao longo deste percurso e que me irão acompanhar nos próximos, pela sua inestimável amizade e pelo seu apoio nos bons e maus momentos.

Por fim, apesar de não haver palavras que possam alguma vez exprimir o que fizeram por mim, a toda à minha família, pais e irmão, pelo seu amor e aprovação incondicionais e por me terem guiado até este momento, e pelos mais que hão de vir.

Valter Costa

"Sometimes by losing a battle
you find a new way to win the war."

Donald Trump

Conteúdo

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

FOV	Field Of View
IPM	Inverse Perspective Mapping
PC	Personal Computer
RF	Rádio Frequência
ROS	Robot Operating System
SI	Sistema Internacional
SO	Sistema Operativo
STEM	Science Technology Engineering Maths
UDP	User Datagram Protocol

Capítulo 1

Introdução

Neste primeiro capítulo é apresentada a estrutura da dissertação, os objetivos e a sua motivação e contexto.

1.1 Motivação e Contexto

Este trabalho aborda o projeto e a análise de diversas demonstrações na área da Robótica quando esta área é entendida como um veículo demonstrativo e educacional.

O projeto de sistemas robóticos deve tirar partido do ambiente lúdico e das características dinâmicas deste tipo de sistemas para diminuir a distância que por vezes se cria entre os utilizadores menos habilitados na criação tecnológica e sistemas que podem apresentar alguma complexidade.

Os casos de estudo apresentados mostram uma variedade de abordagens adequada a uma gradação de objetivos que pretende alcançar o maior público alvo possível. A primeira demonstração é um robô programável por fios, outro caso é um robô programável por blocos visuais, seguidamente um robô programável em linguagem genérica para microcontrolador, um robô que tira proveito das características de um *smartphone* e finalmente um robô adequado a condução autónoma.

Para além da análise de algumas questões de projeto envolvidas em cada demonstração, são também analisadas questões relativas a sistemas que envolvem uma interessante demonstração acerca do funcionamento do robô de condução autónoma, quer na vertente de mapeamento realidade e visão quer na análise de questões relevantes na área do processamento em Tempo Real.

O grande objetivo desta dissertação é criar e otimizar um conjunto de demonstrações robóticas apelativas para uma abrangência máxima de audiências.

1.2 Objetivos

O ensino da robótica no ensino básico e secundário é benéfico para os estudantes como aspeto complementar nas áreas STEM (Science Technology Engineering Mathematics). Recentemente, o número de estudantes envolvidos nestas áreas tem decrescido [8]. Como forma de combate

a esta tendência, nasce o grande objetivo desta dissertação: Criar e optimizar um conjunto de demonstrações robóticas apelativas para uma abrangência máxima de audiências.

1.3 Contribuições

Como contribuições mais relevantes ao longo desta dissertação destacam-se:

- Artigo submetido, aceite e apresentado na conferência indexada EDULEARN15, anexo A.
- Programar 3pi usando FEUPAutom;
- Programar EV3 usando FEUPAutom;
- Programação Transparente no 3pi;
- Programação Transparente no EV3;
- Criação do robô telemóvel;
- Integração de todos os sistemas em ROS para o robô de condução autónoma - Conde;
- Elaboração de documentação para cada casos de estudo abordado.

1.4 Estrutura da Dissertação

Este relatório encontra-se dividido em cinco capítulos. No capítulo 1 é apresentada a motivação e contexto da dissertação e a sua estrutura documental. No capítulo 2 é apresentada a revisão bibliográfica que se encontra dividida em duas grandes secções, sendo a primeira sobre demonstrações e competições robóticas, e a segunda sobre processamento digital de imagem. O capítulo 3 apresenta o trabalho desenvolvido e o respetivo processo de desenvolvimento para os casos de estudo propostos. Os resultados obtidos ao longo do processo de desenvolvimento estão no capítulo 4. Para finalizar, no capítulo 5 são apresentadas as conclusões, a contribuição dada e o trabalho futuro.

Capítulo 2

Revisão Bibliográfica

Neste capítulo é apresentado o estado da arte relativo aos objetivos e motivações deste projeto. Este estudo incidirá por um lado sobre algumas estratégias de ensinar robótica no secundário apresentando algumas soluções de mercado, e por outro sobre a competição de condução autónoma, caso de estudo mais relevante neste projeto. Neste contexto, são estudadas algumas técnicas de sistemas baseados em visão de tempo real que são bastante promissoras para a competição de condução autónoma.

2.1 Competições e Demonstrações Robóticas

Ensinar robótica no ensino básico e secundário é benéfico para os estudantes como aspeto complementar para o seu estudo nas áreas STEM (*Science Technology Engineering Math*). Recentemente, o número de estudantes nestas áreas tem vindo a decrescer nestas áreas [8]. Algumas iniciativas como *Robocup Junior Competition* [9], *RoboParty* [10], *CEABOT* [11], *RobotChallenge* [12], *IstRobot* [13], Micro-Rato [14], *Micro-Mouse* [15] tentam combater esta tendência.

A área da robótica pode ser usada para o ensino e aprendizagem em diferentes contextos [16]. As primeiras cinco iniciativas incluem atividades educativas focadas em criar um ambiente de aprendizagem adequado para alunos com temas relacionados com a Robótica. As seguintes, propõem que a Robótica pode ser uma ferramenta de ensino e aprendizagem para outras áreas STEM. Devido à multidisciplinaridade da robótica várias iniciativas são apresentadas à comunidade. Um exemplo deste tipo de iniciativas é *The Science Engineering NASA Site of Remote Sensing (SENSORS)*. Este projeto permite aos estudantes de ensino básico e secundário controlar LEGO®RCX (fingindo ser *rovers*) pela *internet* fazendo upload dos seus próprios programas [17].

A rede Robot@Scuola [18] é uma comunidade virtual para partilha de conhecimento ligado à robótica, coordenada pela escola de robótica de Genoa, Itália. Esta comunidade inclui escolas diferentes que vão desde jardins de infância até escolas secundárias. A partilha de informação, conhecimento e atividades é feita *online*. Para além de comunidades *online* existem equipas focadas no desenvolvimento de novas tecnologias para crianças que incentivem a participação em atividades criativas como a *The Lifelong Kindergarten group (LLK)* que pertence ao *MIT Media*

Lab [19]. Esta equipa é responsável pelo projeto SCRATCH [20] que consiste num *software* livre para programação básica de forma gráfica.

Existem outros *softwares* livres e/ou pagos que baseiam a sua existência na programação de forma gráfica e que oferecem desafios apelativos para os estudantes. Como exemplos, existem o VISUINO [21] que permite a programação das placas de Arduino usando a estratégia de *drag and drop*. Cada bloco em VISUINO representa os componentes de hardware numa placa de Arduino. Uma vez feito o programa, basta ligar a placa ao pc e fazer o *upload* do mesmo. Existe também o Ardublock que é um *plugin* que é adicionado ao IDE do Arduino que também baseia a sua existência na programação com blocos e fornece a vantagem de gerar o código linha a linha dos blocos gerados. Outro exemplo apelativo à programação gráfica e ligado à robótica que usa a mesma estratégia é o LEGO®NXT ou EV3 que será objeto de estudo mais à frente.

Desde pequenas que a paixão das crianças por objetos que se movem aparece. A introdução dos computadores modernos permite-lhes criarem o seu próprio mundo e brincar com ele (por exemplo, vídeo-jogos). Só em 1998, com aparecimento do primeiro computador desenhado para as crianças é que foi possível criar objetos e fazer ações para os mover [22]. Ao longo dos últimos quinze anos que a LEGO®investe muito na educação e nas escolas, espalhando os seus *kits* em todo o mundo. Muitas escolas como [23] oferecem aos seus alunos a oportunidade de se introduzirem no mundo da robótica com o objetivo de fomentar o seu interesse nas áreas STEM. Os estudantes de ensino secundário devido ao seu maior conhecimento quando comparados com estudantes de ensino básico podem avaliar esta experiência muito melhor que os seus colegas porque em paralelo com estas atividades estão a aprender matemática e física. Estas duas atividades são complementares uma à outra e, como proposto por [24] é benéfico para o desenvolvimento a longo prazo das *STEM skills*.

2.1.1 Competição de Condução Autónoma

As competições robóticas são importantes no sentido da promoção e desenvolvimento de tecnologias e sua apresentação na comunidade académica e ao público [6]. A competição de condução autónoma é uma competição para robôs completamente autónomos que percorrem uma pista Fig. 2.1 com a forma de uma estrada comum, detetam semáforos Fig. 2.2, sinais de trânsito Fig. 2.3, evitam e desviam-se de obstáculos e estacionam numa área destinada para o efeito.

Esta competição está dividida em quatro mangas sendo que a equipa tem a liberdade de escolher em cada manga qual ou quais as provas que pretendem realizar. As provas que se podem escolher estão representadas na tabelas 2.1 e 2.2.

Desafio	**ID**
Condução a alta velocidade	D1
Condução com sinais	D2
Condução com sinais, túnel e obstáculos	D3
Condução com tudo	D4

Tabela 2.1: Descrição das provas do tipo de condução.

Desafio	**ID**
Estacionamento paralelo sem obstáculos	P1
Estacionamento paralelo à frente de um obstáculo	P2
Estacionamento paralelo entre dois obstáculos	P3
Estacionamento de garagem sem obstáculo	B1
Estacionamento de garagem com obstáculo	B2

Tabela 2.2: Descrição das provas do tipo de estacionamento.

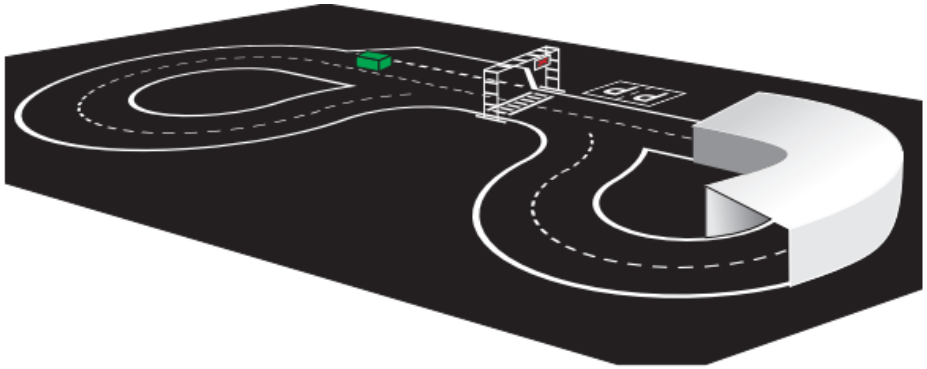


Figura 2.1: Pista da competição de condução autónoma 2015.

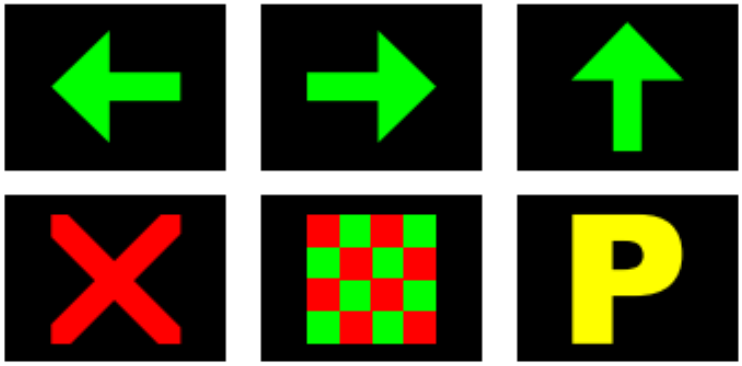


Figura 2.2: Semáforos indicativos da decisão que o robô deve tomar para a competição de 2015.

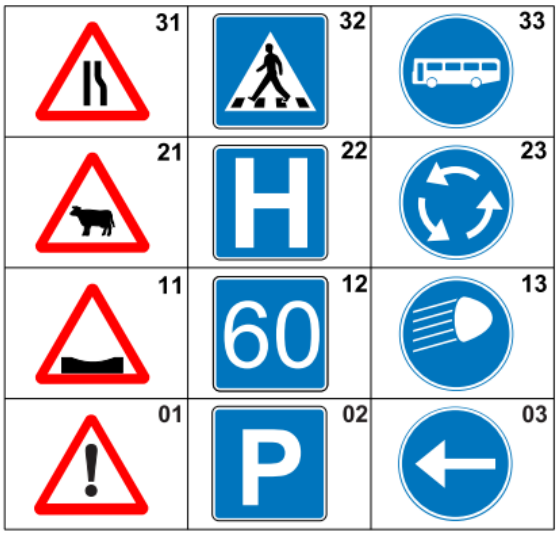


Figura 2.3: Sinais de transito selecionados para a competição de 2015.

A pontuação bem como as pontuações de referência desta prova estão previstas em [25].

2.1.2 *DARPA Robotics Challenge*

Estes desafios foram criados pela *Defense Advance Research Projects Agency (DARPA)* e são provas de condução Autónoma a grande escala. Até agora, existiram cinco competições:

- *DARPA Grand Challenge* 2004 [26];
- *DARPA Grand Challenge* 2005 [27];
- *DARPA Urban Challenge* 2007 [28];
- *DARPA Robotics Challenge* 2012 [29];
- *DARPA FANG Challenge* 2013 [30].

Estas competições são bastante apelativas pelos prémios que oferecem. O primeiro prémio são dois milhões de dólares, o segundo um milhão de dólares e o terceiro com meio milhão de dólares. O objetivo destas competições é promover o desenvolvimento de veículos autónomos para fins militares [6].

2.1.3 Plataformas móveis dotadas de *smartphones*

Dispositivos móveis como telemóveis, tablets e computadores portáteis são generalizados e cada vez mais baratos, devido à sua produção em massa, o que os torna uma opção interessante para controlar sistemas mecatrónicos [31].

Para usar um dispositivo móvel como principal centro de processamento dos sistema, uma via de comunicação tem que ser estabelecida com a unidade de controlo. Normalmente uma porta de série RS-232 é usada [32, 31], mas Bluetooth é também uma opção [32].

Dispositivos móveis têm várias funcionalidades que podem ser úteis em robótica. Estas incluem [33]:

1. CPU: *smartphones* modernos já têm processadores com ciclos de relógio de 1 GHz ou mais. Alguns modelos têm também processadores multi-core. Estas unidades de processamento são capazes de fazer bastantes operações complexas e computacionalmente intensivas, necessárias para a navegação autónoma de robôs;
2. Câmara: pode ser usada com uma variedade de algoritmos para odometria visual, reconhecimento de objetos, deteção e evasão de obstáculos, e rastreamento de objetos;
3. Bússola: pode ser usada como sensor da direção de movimento dos robôs;
4. GPS: pode ser usado para obter a posição do robô em ambientes variáveis, altitude e rapidez;
5. Acelerómetro: pode ser usado para detetar mudanças de velocidade (se o robô chocou com um objeto em alguma direção), e para detetar a orientação do robô;
6. Internet: WiFi ou outra ligação à Internet pode ser usada para monitorizar o robô e enviar comandos;
7. Bluetooth: pode ser usado para comunicação com robôs próximos e para localização;
8. ROS: o Robot Operating System de Willow Garage é já suportado em dispositivos móveis, usando o ramo ROS-java ou ministo e C++ [34, 35].

Tendo em conta estas características, os dispositivos móveis podem ser uma boa aposta na hora de escolha e desenvolvimento de um protótipo.

2.2 Sistemas baseados em Visão

2.2.1 Aquisição e Processamento de Imagem

2.2.1.1 Iluminação

O condicionamento do ambiente de aquisição é fulcral num sistema baseado em visão. Este refere-se ao aproveitamento do conhecimento prévio acerca da aplicação e dos objetos em estudo onde é inserido o sistema. Um ambiente de aquisição de imagem adequado permite maximizar o conteúdo de informação das imagens adquiridas e reduzir a complexidade dos processos de análise necessários à extração dessa informação [36]. Alguns aspetos a ter em conta no condicionamento da cena são as propriedades dos materiais, características intrínsecas dos objetos e a localização dos mesmos.

As principais formas de condicionamento são:

- Controlo de características dos objetos;
- Controlo da posição dos objetos;
- Controlo das condições de iluminação;
- Utilização de técnicas de iluminação estruturada.

O objetivo da iluminação é tornar os objetos da cena visíveis, evidenciando as suas características mais importantes e suprimir as indesejáveis [36].

As técnicas de iluminação dependem de como é aplicada a luz. Estas, dependem da direção da luz (iluminação direta ou difusa), da posição da fonte de luz relativamente ao objeto e à câmara (iluminação frontal ou em contraluz), e da quantidade de luz incidente no objeto ou captada pela câmara (iluminação de campo claro ou escuro)[36].

Em seguida, são apresentadas as principais características de alguns tipos de iluminação [36].

Iluminação difusa origina situações de reflexão homogénea, facilita a eliminação de sombras, a redução de reflexões especulares e a segmentação de objetos.

Iluminação direta permite evidenciar os detalhes da superfície dos objetos.

Campo claro evidencia a silhueta dos objetos opacos.

Campo escuro facilita a visualização dos contornos de objetos transparentes.

Tendo em consideração estas propriedades, a iluminação difusa pode ser uma mais valia tendo em conta o contexto deste trabalho. Pode ainda ser interessante o teste em condições de iluminação frontal direta de campo escuro ou claro, que normalmente são usadas para o realce de saliências e concavidades dos objetos em cena.

Após a aquisição de imagem segue-se o processamento da mesma. Neste contexto a deteção de características é uma ferramenta que ganha importância pelo que será apresentada no próximo ponto.

2.2.1.2 Deteção de Características

Atualmente, são vários os detetores de características usados. Na tabela 2.3 são apresentados alguns exemplos bem como a sua classificação.

Tabela 2.3: Classificação dos detetores de características

Detetor de Características	***Edge***	***Corner***	***Blob***
Canny	X		
Sobel	X		
SUSAN	X	X	
Shi & Tomasi		X	
Level curve curvature		X	
Laplacian of Gaussian		X	X
Difference of Gaussians		X	X
Determinant of Hessian		X	X
FAST		X	X
SIFT			X
SURF			X
BRIEF			X
ORB			X

De entre os métodos acima descritos o estudo incidirá sobre os método mais promissores, sendo estes FAST, SIFT, SURF e BRIEF.

FAST é um algoritmo usado para a identificação de pontos chave numa imagem. Um ponto chave numa imagem é um pixel que está numa posição bem definida e pode ser detetado de forma robusta. Este algoritmo passa pelas seguintes etapas [37]:

1. Seleciona-se um pixel p na imagem. Assume-se que a intensidade deste pixel é I_p. Este é o ponto que será testado como ponto de interesse.

2. Defini-se um nível de *threshold* T.

3. Considera-se um circulo em volta de p com raio r.

4. Para ser considerado ponto de interesse terão de existir pelo menos N pixeis do circulo definido com níveis de intensidade I_p acima ou abaixo de T.

5. Se pelo menos n pontos (estes n pontos possuem localizações estratégicas) deste circulo não tiverem níveis de intensidade acima ou abaixo de $I_p + T$, então p não é um ponto de interesse. Senão se pelo menos k pixeis tiverem níveis de intensidade acima ou abaixo de $I_p + T$, então verificar para todos os N pixeis pelo menos k destes verificam neste critério.

6. Repetir este procedimento para todos os pixeis da imagem.

As variáveis N, k e n dependem do raio r do circulo. Um exemplo deste algoritmo é abordado em [37].

SIFT a ideia deste algoritmo é transformar o conteúdo da imagem em coordenadas de características locais que são invariantes à translação, rotação, escala e outros parâmetros de imagem.

Com este método é possível gerar muitas características até mesmo para pequenos objetos, como estas propriedades são locais este método é robusto à oclusão de objetos (não é necessária uma primeira segmentação de imagem) e é possível fazer corresponder estas características com uma base de dados grande. De um ponto de vista global este algoritmo é dividido em quatro passos [38]:

1. *Scale-space extrema detection* - procurar em várias escalas e posições na imagem.
2. *Keypoint localization* - ajustar o modelo que determine a posição e escala. Selecionar os pontos chave baseado na medição da estabilidade dos mesmos.
3. *Orientation assigment* - calcular a melhor orientação para cada ponto chave.
4. *Keypoint description* - Descrever cada região de um ponto chave com base no cálculo de gradientes para uma escala e rotação selecionadas.

Casos de estudo deste algoritmo e comparação entre outros são apresentados em [39].

SURF Tal como no SIFT, este algoritmo propõe a extração de características (ponto chave) que são invariantes à translação, rotação, etc. Este algoritmo pode ser dividido em duas grandes fases [40]:

1. Fixar uma orientação repetível baseada na informação da região circular em torno do ponto de interesse.
2. Construir uma região quadrada alinhada com a orientação obtida e extrair informação (SURF *descriptor*).

A explicação mais detalhada deste algoritmo encontra-se em [40] e comparação entre SIFT e SURF em [39].

BRIEF Este algoritmo baseia a sua existência na representação de uma imagem numa *string* binária. A sua não invariância à rotação e o seu número menor de testes do que no SIFT, SURF ou FAST permite uma maior rapidez no processo [41]. As duas grande etapas deste algoritmo são:

1. *Detection* - detetar pontos chave por um método semelhante ao SURF.
2. *Feature Description* - comparar níveis de intensidade e atribuir o valor 0 e 1 consoante o valor da comparação e guarda-se este valor. No final o valor composto de zeros e uns é chave booleana.

Nesta secção foram abordados alguns algoritmos que possuem características que podem ter interesse no contexto deste trabalho. Após a deteção de características é necessário a sua extração. Neste sentido, a próxima secção abordará este tema.

ORB Este algoritmo surge da necessidade de reduzir a complexidade temporal para a aplicação deste tipo de estratégias a sistemas de tempo real. O seu desempenho no contexto de correlação entre pontos é semelhante ao SIFT, é menos afetado por ruído de imagem e pode ser utilizado em sistemas de tempo real [1]. A sua estratégia consiste nos seguintes pontos:

1. *Detection* - detetar pontos chave por um método semelhante ao FAST, adicionando-lhe uma componente de orientação mais rápida e precisa .

2. *Feature Description* - usar os descritores do BRIEF, tornando-os mais eficientes.

A figura 2.4 mostra um exemplo de aplicação deste algoritmo:

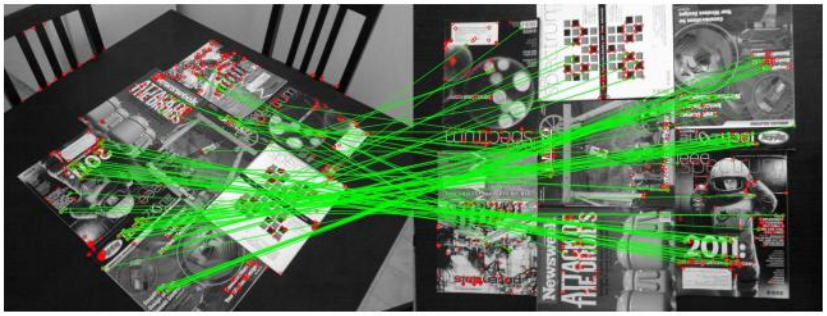


Figura 2.4: Resultado da correspondência resultante usando ORB em imagens reais com uma mudança no ponto de vista. As linhas verdes são correspondências válidas; os círculos vermelhos indicam pontos não correspondentes [1].

2.2.1.3 Extração de características

A extração de características visa encontrar objetos em imagens, o que implica descobrir a sua forma, tamanho e orientação. Usualmente, o objetivo é encontrar propriedades invariantes para que o processo de extração não varie de acordo com as condições admitidas. Sendo assim, as técnicas aplicadas para encontrar formas devem ser fiáveis e robustas qualquer que seja o valor de algum parâmetro que possua influência sobre a aparência da forma que se pretenda detetar [7].

A primeira propriedade invariante de interesse é a iluminação. O objetivo de detetar formas deve ser possível estando num ambiente escuro ou claro. Em teoria, desde que exista um contraste suficiente entre a forma a ser detetada e o seu fundo, a forma existe e pode ser detetada [7]. Na prática, qualquer sistema baseado em visão pode falhar se as condições de iluminação extremas (demasiado escuras ou claras).

A seguir à iluminação, outra característica importante é a posição da forma que se pretende detetar. A deteção de formas deve ser possível qualquer que seja a posição da forma na imagem. Esta característica é geralmente chamada de invariância de posição, localização ou translação [7].

Outra propriedade que se pretende atingir é a invariância de rotação ou orientação. A deteção deve ser possível qualquer que seja a rotação de uma forma na imagem [7].

Por fim, a invariância ao tamanho da forma é também um objetivo que se pretende atingir. Esta propriedade é importante, pois por vezes torna-se difícil garantir as mesmas condições de distância entre o objeto e a câmara [7].

As propriedades acima descritas são as mais importantes quando se aplicam técnicas de extração de características. Contudo, é preciso ter noção que o ruído está sempre presente nas imagens, pelo que por vezes torna-se difícil a extração. Outro ponto a ter em consideração é a possibilidade de existir outras formas na imagem que não são objetos de interesse e podem interferir na deteção de outros.

Neste documento, as técnicas que serão exploradas para extração de características estão representadas na Tabela 2.4.

Tabela 2.4: Organização das técnicas exploradas, baseado em [7]

Técnica de Extração	**Pontos abordados**
Operações ao nível do pixel	*Thresholding*. Subtração de fundo. Vantagens e desvantagens.
Template Matching	Extração de formas por *matching*. Vantagens e desvantagens.
Transformada de *Hough*	Extração de características por *matching*. Vantagens e desvantages.

Para a extração de uma forma numa imagem é necessário identificá-la. Esta identificação pode ser feita considerando a informação de intensidade de brilho ou comparando a imagem com um *template* predefinido. Na primeira abordagem se for conhecido o brilho do objeto, então os pixeis que o compõem podem ser extraídos definindo um nível fixo de *threshold*. Em alternativa, se o fundo da imagem for conhecido, este pode ser subtraído à imagem original de maneira a obter forma desejada [7]. A extração por *template matching* é feita procurando a melhor correlação entre a imagem e um modelo predefinido. A transformada de *Hough* implementa um método de *template matching* de uma forma eficiente para modelos binários. Esta técnica é capaz de extrair formas simples como linhas, quadrados, e circunferências como também formas aleatórias [7]. Em qualquer um dos casos, a complexidade computacional pode ser reduzida admitindo algumas assunções, tais como, algum tipo de invariância.

***Thresholding* e Subtração de fundo** *Thresholding* é um método de extração de formas que funciona sob a premissa que o nível de *threshold* da forma a ser extraída pode ser obtido através do brilho da forma na imagem [7]. Sendo assim, este método torna-se dependente das condições de iluminação. Este problema pode ser resolvido usando uma estratégia de calculo do nível de *threshold* automaticamente (p.ex.: histograma de equalização). Outro problema que pode surgir é o facto de existirem várias formas na mesma imagem. Se estas se sobrepuserem torna-se difícil a separação. Este método torna-se interessante e atrativo devido à sua facilidade de implementação e ao seu custo computacional ser reduzido quando comparado com outros. Um método alternativo é subtrair a uma imagem um fundo conhecido (imagem de fundo) antes de a binarizar. Este método é conhecido como subtração de fundo [7]. É assumido que o ambiente de fundo é conhecido e estático, o que permite a identificação de uma forma no mesmo. Um grande problema deste

método é a possibilidade da existência de ruído. Como este método aplica a subtração, se existir ruído este aparecerá na imagem à qual foi subtraído o fundo.

Para concluir, estes dois métodos apesar de serem atrativos, de simples implementação e rápidos computacionalmente, são altamente dependentes da iluminação e ruído.

Template Matching A estratégia de *template matching* consiste em percorrer uma imagem procurando a melhor correlação entre a imagem em estudo e uma subimagem (*template*). Este método torna-se muito dispendioso computacionalmente, pelo que é evitado o seu uso. Tem como grandes vantagens a insensibilidade ao ruído e à oclusão de objetos. A sua formulação matemática bem como exemplos são estudados em [7].

Transformada de *Hough* A transformada de *Hough* é uma técnica que identifica formas em imagens [7]. Geralmente, é usada para deteção de circunferências, linhas e elipses. Tem como grande vantagem a possibilidade de resolver um problema de *template matching* mas de uma forma mais eficiente. Esta vantagem, é alcançada pela reformulação do processo de *template matching*. A transformada de *Hough* cria um mapeamento dos pontos duma imagem para um espaço acumulador (espaço de *Hough*). Este mapeamento é alcançado de uma forma computacional eficiente [7].

Para concluir esta análise, foi elaborada a Tabela 2.5 com as vantagens e desvantagens destes métodos.

Tabela 2.5: Vantagens e Desvantagens das técnicas de extração

Técnica de Extração	**Vantagens**	**Desvantagens**
Thresholding	Baixo custo computacional. Fácil implementação.	Sensível para variações da iluminação. Apenas aplicável em cenários controlados.
Subtração de fundo	Baixo custo computacional. Fácil implementação.	Sensível para ambientes ruidosos.
Template matching	Insensível ao ruído e à oclusão de objetos.	Muito alto custo computacional.
Transformada de *Hough*	Resolve problemas de *template matching* de uma forma mais rápida.	Alto custo computacional e de memória.

2.2.2 *Tracking*

2.2.2.1 Video *Tracking*

Vídeo *tracking* é o problema de seguir elementos ou objetos que se movem ao longo de um vídeo. Os sistemas de *tracking* dividem-se em dois grandes problemas [42]:

- *Motion problem*: prever local de um objeto numa imagem no frame seguinte, isto é, definir uma região de interesse onde se espera que o objeto estará com uma elevada probabilidade.

- *Matching problem*: identificar o objeto no frame seguinte dentro da região definida acima.

A forma mais simples de resolver o problema de movimento é definir uma região fixa no frame seguinte à custa da posição do objeto na imagem no frame anterior. O tamanho da janela da região de interesse depende das características do problema. Este tipo de abordagem poderá funcionar se, à partida, se conhecer bem as características e restrições do problema o que nem sempre acontece. Outra solução recorrente é o uso dos filtros de Kalman. Este filtros têm várias aplicações no campo do vídeo *tracking* [43], [44], [45] e [46].

O problema de *matching* consiste em comparar e identificar um dado objeto de entre vários objetos candidatos entre a frame anterior e a atual. Alguns exemplos de algumas estratégias de *matching* são [42]:

- *Window Tracking*: este é o método mais simples de *tracking*. É definida uma janela de pequena dimensão para a qual se pretende fazer o *tracking*. Estas janelas podem ser seguidas de frame para frame através de uma correlação;

- *Feature Tracking*: a ideia deste algoritmo é definir e detetar características de uma ou mais partes de um objeto, por exemplo, cantos, linhas, contornos ou regiões bem definidas. O *tracking* a partir de características primeiro localiza-as em dois frames consecutivos e em seguida tenta relacionar cada característica do frame anterior com as do frame atual;

- *Planar Rigid Shapes*: a forma de um objeto numa imagem num plano depende da sua posição e orientação em relação à câmara, e do tipo de projeção. Formas simples como retângulos ou círculos podem ser capturados usando modelos de forma fechada como as equações para prever a sua aparência na imagem. Por outro lado, podem-se usar as características da imagem para detetar a forma o local e a orientação. Este método assume que características da imagem se procura e qual é o fundo da imagem.

- *Solid Rigid Shapes*: Tal como o anterior, se existir um modelo do objeto 3D do objeto este pode ser detetado. Em primeiro lugar os contornos que são detetados na imagem são relacionados com os contornos do modelo. Em seguida, usam-se os pontos que correspondentes para descobrir a rotação e a translação gerando a melhor aproximação da imagem observada. Este passo é repetido para cada frame tirado usando a estimativa anterior como estimativa inicial.

- *Visual Learning*: a ideia deste método é fazer com que o sistema aprenda a forma/aparência e o modelo dos objetos ao contrário de ter esta informação previamente guardada. Este é um método atrativo em particular em sistemas em que é muito difícil saber o modelo e a forma como é o caso dos sistemas que operam debaixo de água.

2.2.3 *Inverse Perspective Maping*

Ao longo dos anos, IPM tem sido aplicado em vários problemas no campo dos sistemas de transporte inteligentes [3]. IPM é uma transformação geométrica que projeta cada pixel da vista

em perspetiva 2D de objetos 3D e mapeia-os para uma nova posição construindo uma nova imagem num plano 2D inverso. Matematicamente, esta transformação pode ser vista como uma projeção de um espaço 3D num plano 2D [2]. Esta estratégia pode ser promissora no contexto da condução autónoma para o seguimento de linha pois remove o efeito da perspetiva da imagem também conhecido como *bird's eye view*. No entanto, esta transformação tem uma limitação grande que é a presença de obstáculos que interrompe a eficácia deste mapeamento. Uma solução a este problema é estudada em [3].

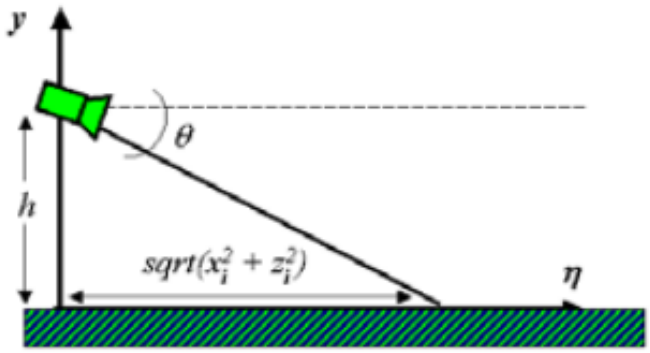


Figura 2.5: Modelo IPM [2].

As equações que regem esta transformação são [2]:

$$u(x,0,z) = \frac{\gamma(x,0,z) - (Y - \alpha)}{\frac{2\alpha}{n-1}} \tag{2.1}$$

$$v(x,0,z) = \frac{\theta(x,0,z) - (\beta - \alpha)}{\frac{2\alpha}{m-1}} \tag{2.2}$$

Onde,

$$\gamma = \arctan(\frac{z}{x}) \tag{2.3}$$

$$\theta = \arctan(\frac{h}{\sqrt{x^2 + z^2}}) \tag{2.4}$$

E Y é o angulo entre a projeção do eixo ótico no plano liso, β é o ângulo entre o eixo ótico e o horizonte, α é o ângulo de abertura da câmara (FOV - *Field of View*) e *nxm* é a resolução da imagem.

(a) (b)

Figura 2.6: Imagem de entrada *(a)* e respetiva transformação IPM *(b)* adaptado de [3].

A figura 2.6 mostra um exemplo prático de uma transformação IPM. Este método aparenta ser bastante promissor para aplicação no robô de condução autónoma.

2.2.4 Calibração de câmaras

A calibração é utilizada para tentar corrigir pequenos erros produzidos pelas câmaras. Estes erros são internos (advêm do processo de fabrico) e da qualidade das lentes. Algumas áreas que requerem uma calibração precisa são [5]:

- interpretação 3D de imagens;
- reconstrução de modelos no mundo;
- interação de robôs no mundo.

A calibração de câmaras visa encontrar os parâmetros internos que afetam as seguintes propriedades:

- centro da imagem;
- distância focal;
- parâmetros da distorção provocada pelas lentes.

Um exemplo de aplicação de uma imagem distorcida e a sua resultante após a calibração da câmara e removendo a distorção pode ser vista na figura 2.7.

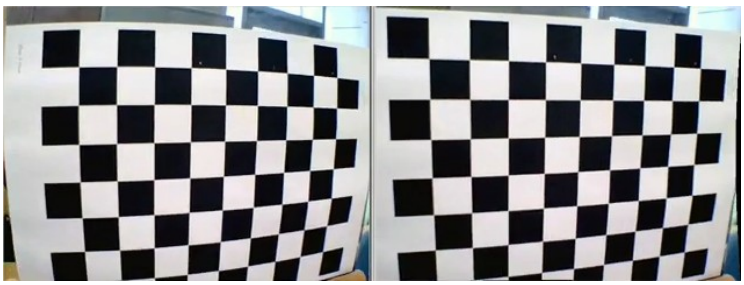


Figura 2.7: Exemplo de uma imagem distorcida (à esquerda) e remoção da distorção (à direita) [4].

Estes parâmetros são normalmente conhecidos como parâmetros intrínsecos da câmara. Para além destes, existem os parâmetros extrínsecos. Estes relacionam o referencial da câmara com um ponto no referencial do mundo. Matematicamente, esta mudança de referencial (figura 2.8) é vista como uma matriz de transformação homogénea 2.5.

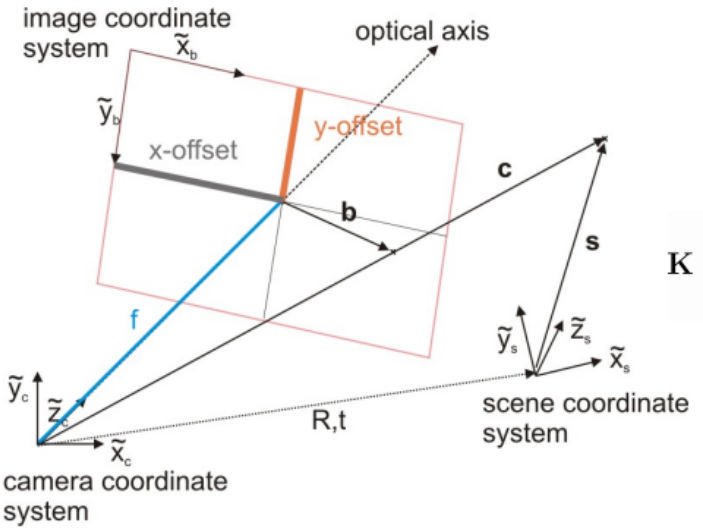


Figura 2.8: Modelo *pinhole* de uma câmara [5].

$$b = \begin{bmatrix} u \\ u \\ 1 \end{bmatrix} \sim \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_1 \\ r_{21} & r_{22} & r_{23} & t_2 \\ r_{31} & r_{32} & r_{33} & t_3 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \tag{2.5}$$

Em que f_x e f_y representam a distância focal, c_x e c_y representam o ponto central (x-offset e y-offset na figura 2.8), r_{ij} a matriz de rotação, t_k o vetor de translação e x, y, z as coordenadas no mundo.

Para a calibração dos parâmetros intrínsecos e extrínsecos de uma câmara é usando um padrão de xadrez que é capturado pela mesma em diferentes ângulos e distâncias garantido sempre que o padrão está visível na foto.

2.2.5 Considerações de Tempo Real

A velocidade dos processadores *standard* chegou ao ponto onde problemas interessantes que recorrem a análise de imagem em tempo real podem ser executados por qualquer computador normal. No entanto pouca atenção tem sido dedicada para encontrar algoritmos apropriados para aplicar em aplicações tempo real. A visão em tempo-real é uma excelente fonte de *feedback* para sistemas cuja interação com o ambiente seja dinâmica [47, 48] e com o avanço tecnológico das câmaras (maior ângulo de visão) problemas tradicionais como localização autónoma, reconhecimento de gestos ou outras interfaces tornam-se interessantes de explorar.

Um dos problemas essenciais de visão tempo-real é encontrar a pose de objetos ao longo de várias imagens.

Duas técnicas são exploradas na literatura:

- *Full-frame image processing*: *optical flow*, *region segmentation*;
- *Localized feature detection*.

O primeiro caso gera processamento de dados intensivo que geralmente é feito *offline*. O segundo concentra-se em áreas localizadas da imagem tentando recolher o máximo de dados possível num número reduzido de passagens pela imagem. Esta abordagem é bastante usada em robótica [47, 49, 50, 51, 52, 53].

Embora, a lista de exemplos de aplicação seja extensa, o conjunto de características são apenas variações de pequenos conjuntos de primitivas (chamados os *edgels*) [54].

Geralmente, estamos interessados não so na hipótese que produz resultados mais precisos, mas também nas que não comprometam o estado do sistema pela sua lentidão. "Em muitos casos, a abstração natural e uma *framework* multi-nível onde são impostas restrições geométricas *"top-down"* e a informação do mundo e tratada *"bottom-up""*[48], ou seja, restringe-se objetos com maior quantidade de dados e trata-se dos objetos com menor quantidade de dados primeiro para obter melhor performance.

Em aplicações de reconhecimento de objetos, classificadores são usados para aprendizagem *offline*. Esta abordagem torna-se difícil em *run-time* e perde a sua eficiência em casos onde adição ou remoção de conjuntos seja precisa [54].

Quando existe informação previa disponível passível de ser retirada de uma imagem, podemos restringir o estado do mundo e consequentemente extrair informação detalhada do ambiente.

Esta informação explicita o esforço computacional requerido para a extração das características envolvidas e o quão úteis serão para o entendimento do estado do sistema.

Perceber o que se pode retirar da imagem em cada *frame* e de extrema importância em aplicações de visão tempo real, onde os recursos são extremamente limitados e dispendiosos (por exemplo em aplicações de *tracking* com elevado *frame rate* a informação recolhida previamente e maior).

Em sistemas modernos, o processamento de imagem é tratado como uma operação *bottom-up* aplicada uniformemente a todos os inputs, de forma a detetar diversas características.

Os resultados da identificação são depois levados para a estimação dos parâmetros, onde diversas quantidades de interesse são identificadas (por exemplo em estimadores modernos como SIFT [38] que consegue identificar características com relativamente pouca informação anterior). Novas imagens são depois tratadas pelos mesmo detetores e a imagem original não precisa de ser examinada outra vez. Isto leva a que as aplicações de tempo real mais recentes tenham uma filosofia de "zero copy - one pass"[47], ou seja, nunca copiar a imagem original e extrair a maior quantidade de informação possível num só varrimento da imagem.

Andrew J. Davison [55] discute que faz mais sentido em aplicações tempo-real manter a operação como *top-down* e adaptativa, se e só se o poder de processamento disponível poder ser usado

mais efetivamente por esta abordagem do que pela abordagem *bottom-up*. A abordagem *top-down* é chamada de visão ativa [56].

"Active vision systems have mechanisms that can actively control camera parameters such as position, orientation, focus, zoom, aperture, and vergence (in a two-camera system) in response to the requirements of the task and external stimuli"[57].

Em aplicações de *real-time tracking* a abordagem do "zero copy - one pass"pode ser usada em conjunto com conhecimento da fluidez de movimento, melhorando a área de procura da localização, ou ate usando modelos mais complicados da dinâmica do robô.

Isard and Blake [58] conseguem atingir níveis satisfatórios de velocidade de execução realizando procuras locais na vizinhança das partículas de distribuição da localização do objeto depois da predição de movimento. As partes da imagem onde a probabilidade e baixa são ignoradas.

No entanto, nenhuma separação e feita do facto de alguns objetos serem mais discriminativos que outros ou que alguns fornecem informação redundante, e mais importante ainda nenhum calculo e feito para saber quantas características são necessárias identificar para obter uma performance. O facto de haver informação compartilhada e importante, por exemplo, em aplicações de *tracking* porque o objeto pode ser definido por um vetor finito de parâmetros que descrevem a sua configuração, os seus graus de liberdade como também outras variáveis de interesse.

Usando como exemplo o frame rate ser elevado, usar a informação compartilhada não escala o processamento necessário (em proporção) como seria o caso da abordagem *bottom-up*.

De qualquer forma, em sistemas de visão práticos não e definida qual a melhor abordagem porque a procura ativa tem problemas com possíveis incertezas ou incompatibilidades.

2.3 Ferramentas de Desenvolvimento

Esta secção tem como objetivo fazer um levantamento das ferramentas de *software* que podem ser usadas para a aquisição e processamento de imagem e integração de *software* no contexto de robótica.

Para a programação do LEGO®mindstorms EV3 será usado o *software* [22] fornecido pela marca. Para a programação do Pololu 3Pi Robot [59] será usado o *sofwtare* disponibilizado pelo Arduino e/ou em alternativa o FEUPAutom [60].

Para o processamento de imagem a linguagem de programação usada será C/C++ usando *OpenCV*. Apesar de ser uma ferramenta livre o seu suporte na comunidade de programadores é vasta quer em fóruns quer na documentação oficial.

Para a programação do sistema baseado em visão de tempo real (robô de condução autónoma) será ainda usado ROS - *Robot Operating System*. ROS é uma coleção de *software frameworks* para o desenvolvimento de programas e aplicações para robôs. Os conceitos fundamentais de ROS são [61]:

Nós são processos que correm uma dada aplicação. Como ROS foi desenhado para ser modular, um sistema que o use normalmente é constituído por vários nós.

Mensagens a forma pela qual os nós comunicam é por mensagens que estão bem definas em ROS. Estas mensagens podem ser compostas por inteiros, reais, caracteres ou vetores.

Tópicos a forma como as mensagens passam entre nós é publicando em tópicos. Exemplificando, quando um nó A precisa de informação do nó B, este deverá subscrever o tópico que está a ser publicado pelo nó B.

Serviços são usados quando se pretendem passar mensagens de forma síncrona, ao invés de usar o modelo de *broadcasting* de publicador-subscritor.

2.4 Considerações Finais

Este capítulo contemplou uma pesquisa das soluções de mercado para os casos de estudo propostos neste projeto. Para o caso de estudo mais relevante, robô de condução autónoma, foi feito um levantamento de algumas técnicas usadas nos sistemas baseado em visão de tempo real. Em particular, existem algumas estratégias como o ORB que aparenta ser bastante promissora para a deteção de semáforos, o IPM para a remoção de perspetiva na imagem da pista e a utilização da filosofia "zero copy, one pass"para o sistema baseado em visão que será desenvolvido.

Quanto aos restantes casos de estudo, foi feita uma pesquisa de algumas soluções já existentes que são usadas para a introdução ao ensino de robótica a alunos de secundário como meio para o ensino de outras áreas como as STEM.

Para finalizar, foi realizado um levantamento de algumas ferramentas de *software* que podem ser usadas no decorrer deste projeto.

Capítulo 3

Casos de estudo em Robótica Educativa

Nesta secção apresentar-se-á diversos casos de estudo e as motivações do projeto. Como caso de estudo particularmente relevante será apresentado o projeto "Conde", que participou na competição de condução autónoma do Festival Nacional de Robótica (FNR). Serão ainda apresentadas as motivações de projeto e as soluções de *software* para o funcionamento.

3.1 Introdução

Este trabalho procura estudar casos de estudo com complexidade variada:

1. **REDI** - Robô projetado na FEUP similar a outro da Universidade de Aveiro [62], para demonstrações curtas em ambientes não estruturados, programável por cabos. Existe um simulador para este robô;

2. **EV3** - Robô fabricado pela LEGO®, de mecânica fácil de construir, programável por blocos simples mas que foi programado com a lógica associada a máquinas de estado;

3. **3pi** - Robô comprado à empresa Pololu pronto a ser usado. Este robô é baseado num micro controlador AtmelAVR, programável em linguagem imperativa (linguagem C ou similar).

4. **Robô Telemóvel** - Robô elaborado no âmbito deste trabalho que utiliza um telemóvel (*smartphone*) e um Arduino para dar resposta a alguns desafios como o seguimento de linhas coloridas;

5. **Conde** - Robô projetado na FEUP no contexto da condução autónoma, a remodelar totalmente o *software*.

3.1.1 Ponto de partida

O ponto de partida de cada projeto é o seguinte:

1. **REDI** - Desenvolvido dentro da FEUP e pronto a testar;

2. **EV3** - No âmbito da cooperação com a Escola Secundária Aurélia de Sousa ajudou-se estudantes do secundário a construir uma configuração mecânica adequada;
3. **3pi** - Foi acrescentada uma capota e eletrónica adicional (incluindo comunicações);
4. **Robô Telemóvel** - Robô que foi feito no decorrer deste trabalho.
5. **Conde** - Robô para concorrer ao Festival Nacional de Robótica na prova de condução autónoma.

3.1.2 Trabalho realizado

O trabalho realizado foi o seguinte:

1. **REDI** - Foram feitas as adaptações para o seguimento automático de linhas e programadas as lógicas de controlo de histerese e linear;
2. **EV3** - Fez-se as adaptações mecânicas para o seguimento e implementou-se as lógicas de controlo com o *software* de blocos da LEGO® mas segundo uma lógica de máquinas de estados; foi criado o tutorial para programar por intermédio de "FEUPAutom Grafcet"+ "ST"; foram criados os módulos necessários para a aplicação da programação transparente; foi ainda desenvolvida uma biblioteca de funções para a possibilidade de programação deste robô usando C/C++;
3. **3pi** - Foi adaptado para seguimento melhorado tal como nos robôs supramencionados; foram criados tutoriais para utilização na Universidade Júnior (UJr) para programação via AVR GCC, via Arduino IDE e via "FEUPAutom Grafcet"+ "ST", consideradas mais amigáveis que a linguagem C;
4. **Robô Telemóvel** - Foi elaborado um protótipo e uma aplicação móvel para o seguimento de linhas de diferentes cores; foram elaborados tutoriais para a montagem e utilização deste robô.
5. **Conde** - programado em linguagem C++ tirando proveito de ROS; os sensores de visão foram remodelados e foram introduzidas duas câmaras PSEye; foi atualizada a programação para a prova de condução autónoma e foi otimizado o sistema de visão e controlo;

Em entrevista ao responsável pela Universidade Júnior da FEUP em Robótica Móvel e supervisor desta tese, Armando Sousa afirma que é benéfica a utilização estruturante do Grafcet para transformar um longo bloco declarativo em diversos blocos mais pequenos de código onde a transição entre os blocos está particularmente bem definida (numa transição do Grafcet), mesmo que nem todas as outras possibilidades do Grafcet sejam exploradas. Aliás, fica assim clara a natureza de controlo híbrido associado aos sistemas robóticos recentes. A mesma fonte assegura ainda que a programação ST reduz as possibilidades de acidentes com uma linguagem densa e cheia de carateres com significados privilegiados tal como o C ou C++ e, assim sendo, a verbosidade do Pascal é benéfica.

3.2 Sistema de *Tracking* de Robôs

O sistema de *tracking* de robôs já existente e utilizado neste trabalho é composto por uma câmara ligada a um PC que está montada num suporte. Como é possível observar na figura 3.1, a câmara aponta para o chão sendo possível capturar a área da pista e os robôs que por ela circulam.

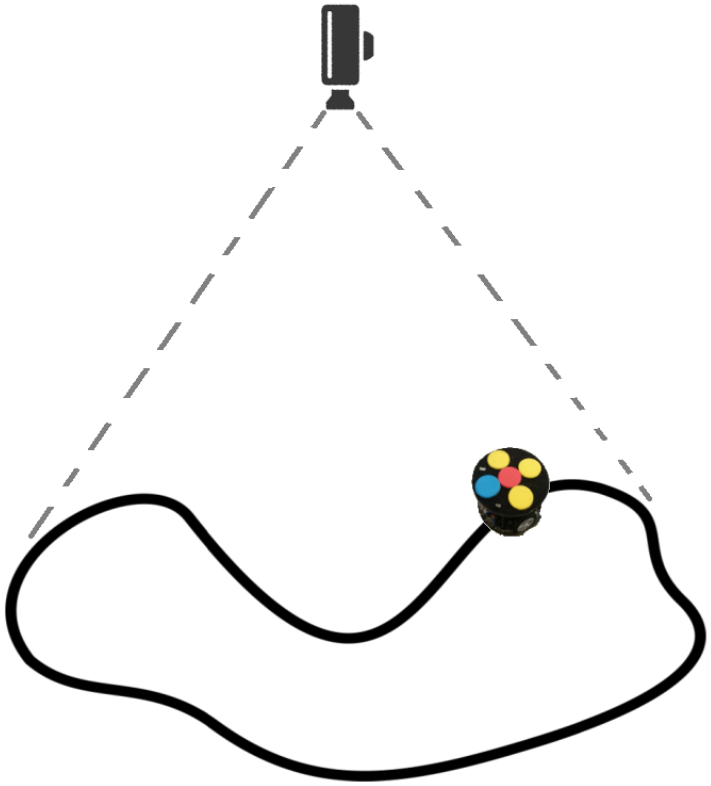


Figura 3.1: Diagrama representativo do sistema de *tracking* de robôs.

A câmara captura a imagem do robô na pista e identifica-o através do padrão de cores único que este possui. A partir desta identificação, é possível seguir o movimento deste robô na pista de forma a que seja possível à posteriori ver o percurso que este percorreu. O *software* que foi desenvolvido para este sistema, figura 3.2, possui uma interface gráfica que possibilita ao utilizador definir por exemplo o número de voltas que cada robô dá, definir *checkpoints* ao longo da pista, etc. Possui ainda uma janela de configuração que permite calibrar as cores dos robôs, calibrar o ponto central da câmara e vetorizar a pista.

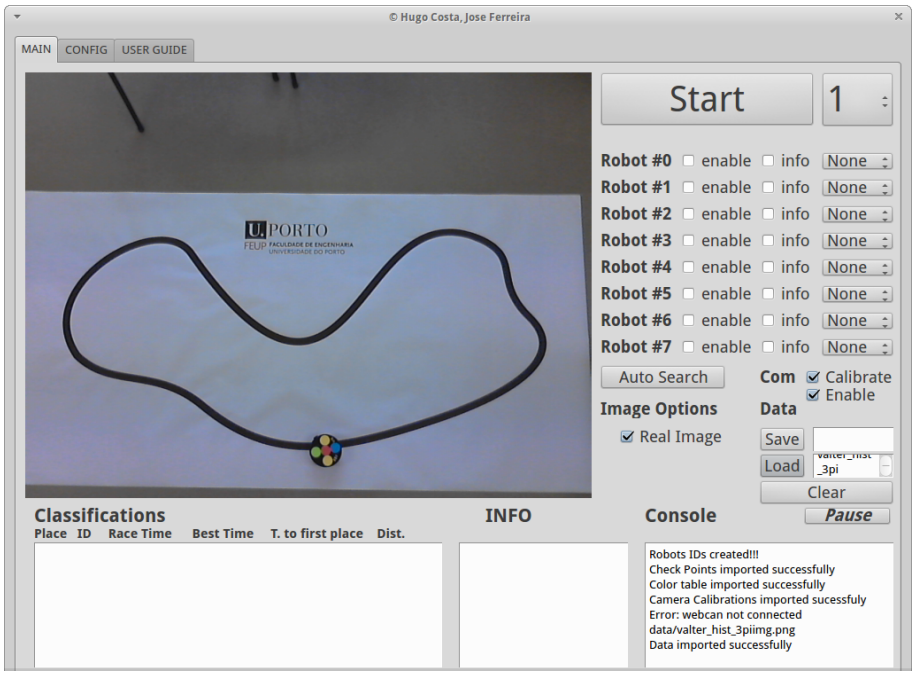


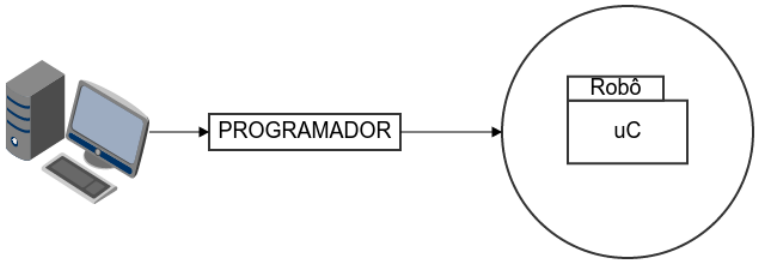
Figura 3.2: *Software* utilizado para o *tracking* de robôs.

Este sistema foi usado para a avaliação dos resultados de controlo por histerese e linear para os casos em estudo. Para este sistema foi calculado o *lag* da câmara. Este resultado é mostrado no capítulo 4.

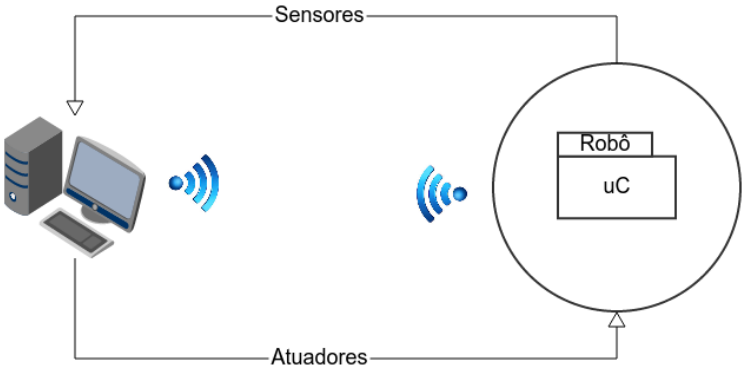
3.3 Programação Transparente

Nesta secção será apresentado o conceito de programação transparente. O método atual para a programação de micro controladores é utilizar um programador externo, como é o caso do robô 3pi, ou embutido na placa de programação, como é o caso das placas Arduino. Esta abordagem tem algumas limitações inerentes como a necessidade de utilizar um programador externo, o número finito de vezes que é possível programar um micro-controlador e a própria logística de ser necessária uma ligação física entre o micro-controlador e o PC.

No contexto deste trabalho foi desenvolvida uma abordagem que permita contornar as limitações acima mencionadas. Esta estratégia apresentada tira proveito das características dos desafios colocados nas demonstrações de robótica ou na Universidade Júnior, por exemplo o seguimento de linha por parte de um robô ou a resolução de um labirinto, para o desenho da mesma. A ideia desta estratégia passa pela troca de mensagens entre o robô e o PC 3.3b.



(a)



(b)

Figura 3.3: Estratégia de programação atual de micro-controladores *(a)* e estratégia proposta para a programação transparente *(b)*.

O robô envia o estado atual dos sensores para o PC que faz o processamento e toma uma decisão. Esta decisão é enviada para o robô que reage de acordo o que recebe. O conceito de programação transparente foi aplicado em dois casos de estudo - LEGO®EV3 e Pololu 3pi e são apresentados nas secções 3.5.4 e 3.6.3 respetivamente.

3.4 REDI - Robô de Ensino Didático

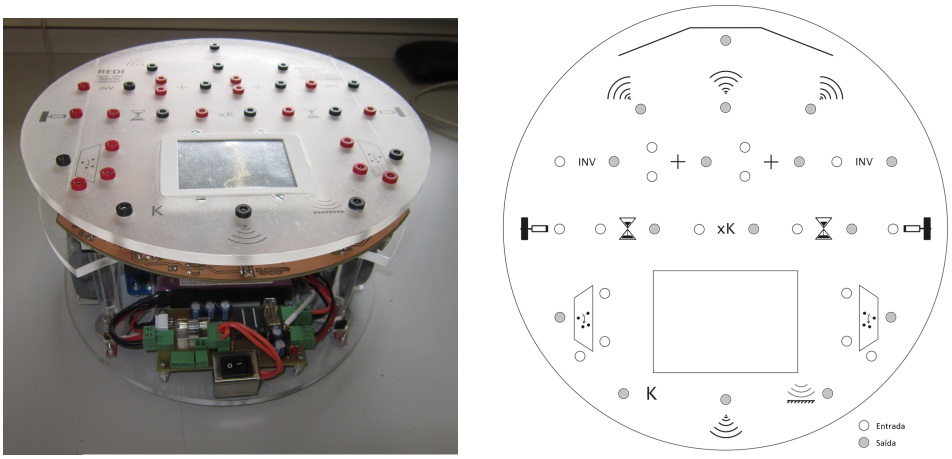
O REDI, figura 3.4a, é um robô dedicado ao ensino de conceitos básicos de robôs móveis autónomos. Este robô é programado usando fios para fazer ligações entre os vários sensores e motores e alguns blocos com funções matemáticas que este possui. Ao nível do *hardware* os principais componentes deste robô são:

- 4 sonares, 3 na frente do robô e um atrás;
- 1 *array* 8 de sensores de infravermelhos para a deteção de linha;
- 4 sensores de infravermelhos para a deteção do chão;
- 2 motores;
- 1 Arduino Mega + ecrã tátil;

- 1 bateria de lítio.

Para além dos sensores e motores disponíveis para a programação deste robô existe um conjunto de blocos que contemplam algumas operações básicas. São estes:

- 2 inversores (multiplicação por -1);
- 2 somadores;
- 2 multiplexadores;
- 2 temporizadores;
- 1 multiplicador;
- 1 *offset* (valor constante).



(a) (b)

Figura 3.4: Robô REDI *(a)* e representação dos seus blocos *(b)*.

A interface com o utilizador é feita através do ecrã tátil. Este ecrã disponibiliza funções como a definição de parâmetros dos blocos, ver estado da bateria, calibração de sensores, *vu-metter* e visualização de gráficos. A verificação e validação das ligações efetuadas é feita de forma a garantir que não é possível a realimentação de blocos, a ligação entre duas saídas e a ligação entre duas entradas.

A fim de testar comparar a performance deste robô no sistema de *tracking* de robôs, foi acrescentada uma capota, figura 3.5, e foram efetuadas duas montagens para o seguimento de linha.

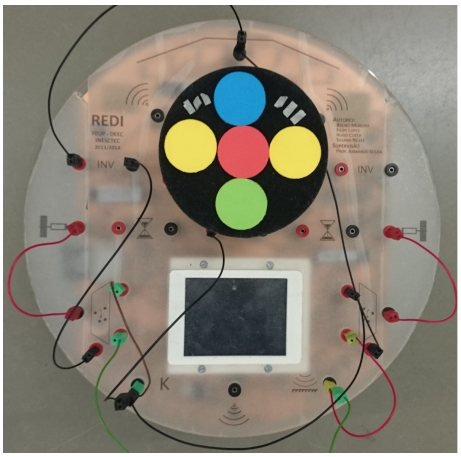
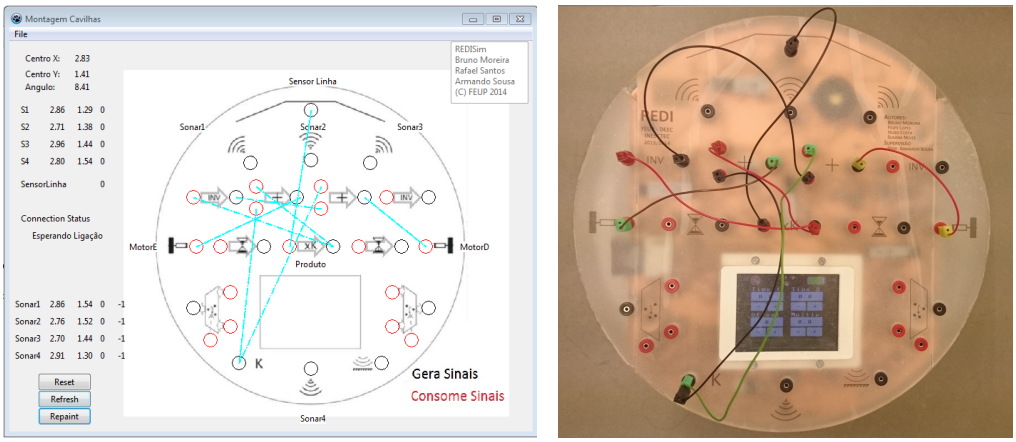


Figura 3.5: REDI com capota.

A capota que foi colocada permite a identificação inequívoca por parte do sistema de *tracking* de robôs. Desta forma é possível avaliar o desempenho das duas montagens testadas.

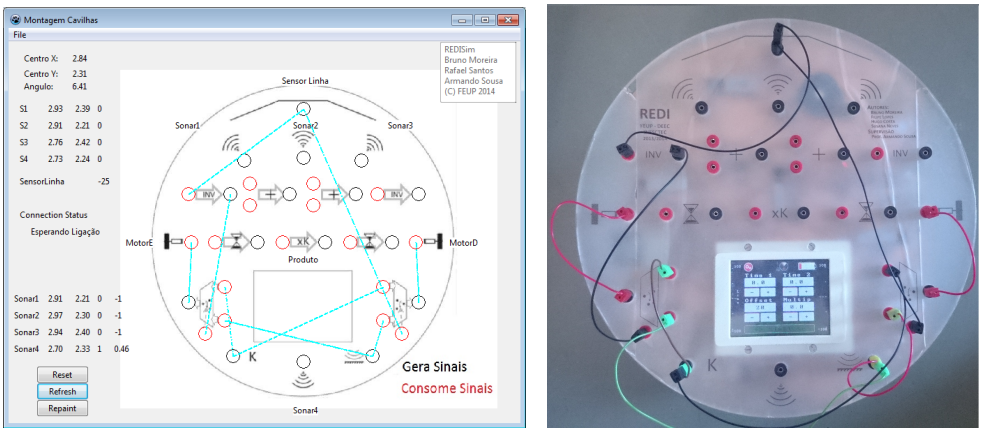
A primeira montagem implementa um controlador linear para o seguimento de linha. A ideia deste controlador é colocar em cada motor uma velocidade proporcional à distância à linha. Desta forma, verifica-se uma trajetória e movimento suaves no seguimento da linha.



(a) (b)

Figura 3.6: Esquema de ligações do controlo linear para o seguimento de linha. Simulador do REDI *(a)* e robô real *(b)*.

A segunda montagem implementa um controlador de histerese para o seguimento de linha. Este controlo o que faz é corrigir a sua posição à linha consoante a o valor das extremidades dos sensores. Visualmente observa-se o robô apresenta uma trajetória com correções à linha mais "bruscas"e um seguimento não tão suave quanto o linear.



(a) (b)

Figura 3.7: Esquema de ligações do controlo de histerese para o seguimento de linha. Simulador do REDI *(a)* e robô real *(b)*.

Os resultados do sistema de *tracking* de robôs é apresentado no capítulo 4.

3.5 LEGO®Mindstorms EV3

Nesta secção é apresentado um projeto de cooperação entre a FEUP e a ESAS - Escola Secundária Aurélia de Sousa. É esperado incentivar os estudantes de secundário na sua introdução ao mundo da robótica transformando atividades extra-curriculares num projeto que promova o crescimento e conhecimento destes noutras áreas. Outro objetivo é a participação destes estudantes no Festival Nacional de Robótica na competição de Rescue-B. Esta competição consiste num pequeno robô que deverá ser capaz de:

1. seguir uma linha preta em fundo branco;
2. detetar e desviar-se de obstáculos retomando em seguida o percurso;
3. detetar cruzamentos verdes;
4. procurar e transportar uma vítima (representada por uma garrafa de refrigerante);

Este projeto envolveu a presença semanal de um monitor da FEUP na ESAS no sentido de acompanhar, aconselhar e retirar dúvidas relativamente à construção e desenvolvimento dos algoritmos/programas desenvolvidos pelos estudantes.

Esta experiência envolveu o robô da LEGO®Mindstorms EV3 mostrado na figura 3.8, que a partir de agora será chamado apenas por EV3.

Figura 3.8: Kit LEGO®Mindstorms e montagem exemplo do robô EV3.

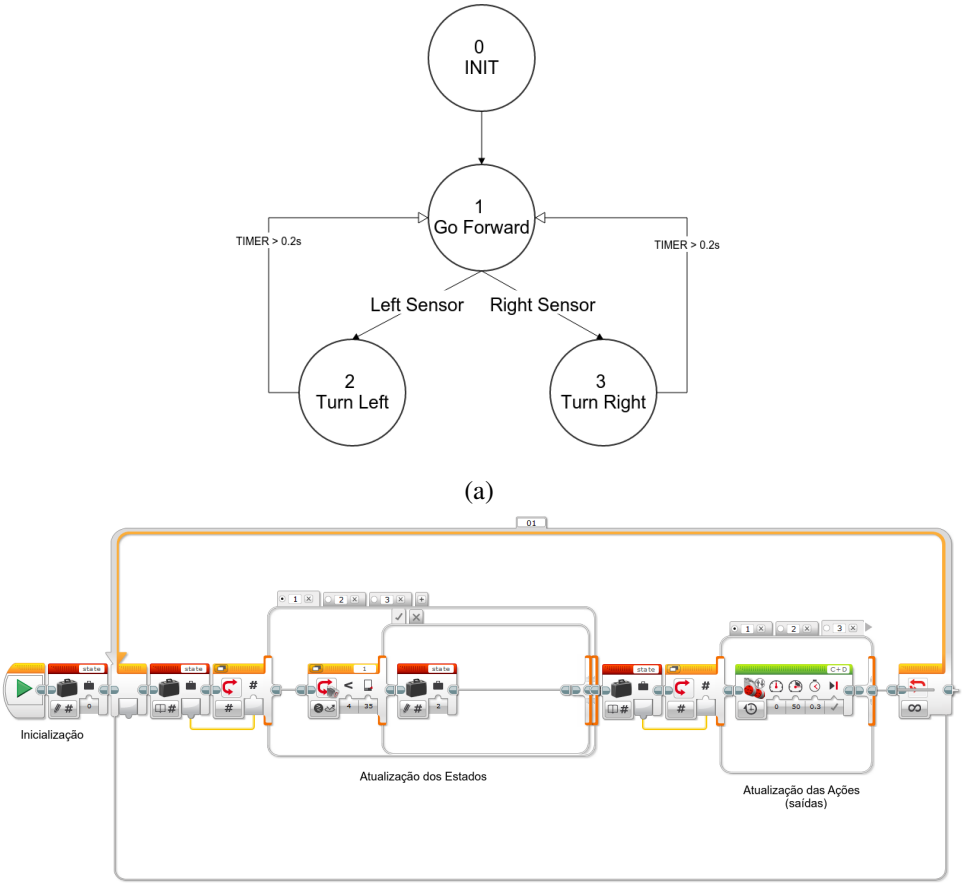
O kit de montagem LEGO®EV3 para além das peças LEGO®Technic usuais neste tipo de kits, disponibiliza um conjunto de componentes tais como:

- 1 sensor de cor;
- 2 sensores de toque;
- 1 sensor de giroscópio;
- 1 sensor de ultra-sons;
- 2 servo motores;
- 7 cabos de ligação para sensores e motores;
- 1 bateria recarregável;
- 1 *intelligent EV3 Brick*, é a unidade central de processamento composto por um processador ARM e corre um sistema operativo baseado em linux.

Tendo em conta o contexto deste projeto, este robô oferece algumas vantagens relativamente a outros pois permite a quem o usa o despreocupação de questões relacionadas com eletrónica e mecânica. Para além disso, a LEGO®oferece um *software* bastante apelativo para a programação deste kit que é a programação visual (por blocos) numa estratégia de *drag and drop*.

Para o arranque deste projeto foram desenvolvidos alguns guiões de trabalho com desafios adequados ao ensino secundário. Estes, foram pensados como sendo um ponto de arranque para a introdução ao kit e *software*, mas também com objetivos e desafios que demonstrem que a resolução de problemas complexos, podem ser resolvidos à custa da divisão e resolução de problemas mais pequenos, como é o caso da competição na prova Rescue-B.

Apesar das características de *user friendly* do *software* da LEGO®, com o decorrer do trabalho e o crescimento da dificuldade dos desafios os programas que eram desenvolvidos ficavam demasiado extensos o que dificultava a sua visualização e o seu *debug*. A solução encontrada para este problema foi a introdução de máquinas de estado. Esta solução permitiu uma redução significativa da extensão dos programas produzidos, a introdução de pensamento estruturado e uma



(a)

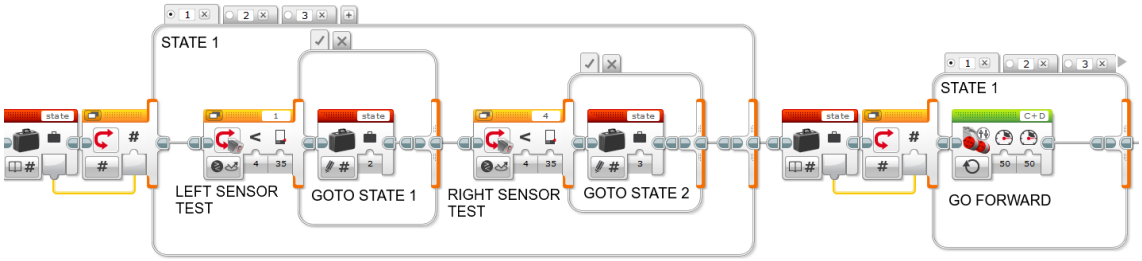
(b)

Figura 3.9: Máquina de estados para um seguidor de linha *(a)* e sua implementação em LEGO®*software (b)*.

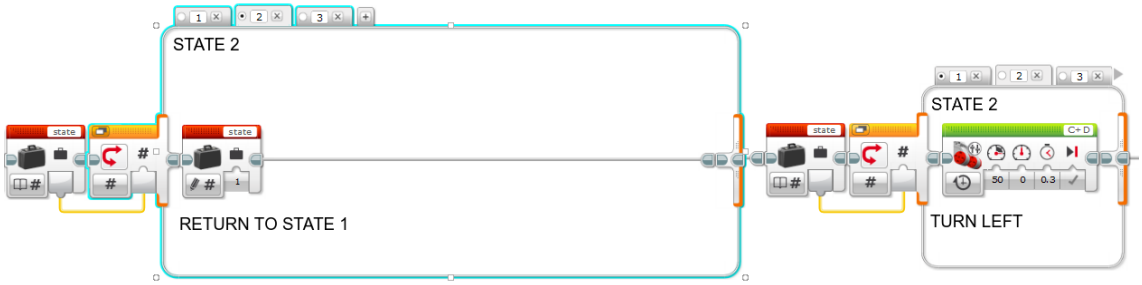
passagem rápida da abordagem de máquinas de estado para a sua implementação no *software*. Um exemplo de aplicação é mostrado na figura 3.9.

A implementação da máquina de estados pode ser dividida em três passos. O primeiro é a inicialização, que neste caso não possui nenhuma, mas poderá ter, por exemplo a calibração dos sensores de linha. O segundo é a atualização dos estados consoante as leituras dos sensores. A terceira é a atualização das saídas com base no estado atual.

Para a instalação do *software* deste num sistema Windows apenas é necessário fazer o *download* do ficheiro de instalação na página oficial da LEGO®[22]. Nessa mesma página, existem vários tutoriais que ensinam a programar e a dar os primeiros passos. No entanto, com este *software* estamos sempre limitados aos blocos que existem e às funcionalidades que estes possuem. Na secção 3.5.2 é apresentada uma possível solução por forma a retirar maior proveito das capacidades da unidade central de processamento.



(a)



(b)

(c)

Figura 3.10: Explicação da máquina de estados implementada em LEGO®. Estado 1 *(a)*, estado 2 *(b)* e estado 3 *(c)*.

3.5.1 *Tracking* do EV3

De forma análoga ao REDI, foi feito o *tracking* do EV3 usando para o efeito o mesmo sistema. Foi colocada uma capota, para que pudesse ser reconhecido pelo sistema e foram retirados os resultados do controlo linear e por histerese. Os gráficos das suas trajetórias são apresentados no capítulo 4. Na figura 3.11 é apresentada o *setup* do robô para o *tracking*.

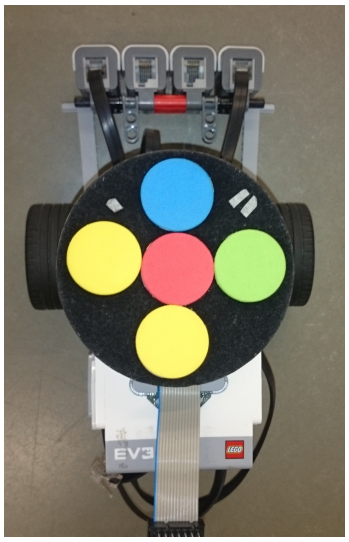


Figura 3.11: Robô EV3 com capota.

3.5.2 Guia para Instalação e Uso de Linux na Unidade de Processamento LEGO®

Nesta secção é apresentado um tutorial para a instalação de um *kernel* de linux na unidade central de processamento LEGO®. Este *kernel* será instalado num cartão micro SD pelo que não alterará o *firmware* original. Sendo assim, esta unidade passará a funcionar num sistema semelhante ao de *dual boot*. Este tutorial é baseado em [63] e reúne os passos básicos para instalação e uso do *kernel* ev3dev.

Considerações iniciais:

1. É necessário um cartão micro SD com capacidade não superior a 32GB;
2. Um computador com leitor de cartões SD;
3. Um cabo mini-USB para a ligação ao *brick* EV3;

Este tutorial foi feito num pc que tem como sistema operativo uma distribuição de linux pelo que todos os passos a seguir são válidos apenas para este SO. As instruções para outros sistemas tais como Windows e MAC OS X estão na página [63].

Passos:

1. Passo 1: *Download* da última versão do *kernel* ev3dev.
   Fazer *download* do ficheiro .xz seguindo o *link* GitHub [64].
2. Passo 2: Copiar a imagem para o cartão SD.
   Abrir um terminal e sem o cartão SD no pc correr o comando df. Deverá aparecer algo como isto:

```
df -h
```

```
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1       119G   79G   34G  70% /
none            4.0K     0  4.0K   0% /sys/fs/cgroup
udev            7.8G   12K  7.8G   1% /dev
tmpfs           1.6G  1.1M  1.6G   1% /run
none            5.0M     0  5.0M   0% /run/lock
none            7.9G  1.5M  7.9G   1% /run/shm
none            100M  3.7M   97M   4% /run/user
```

Em seguida colocar o cartão SD no pc e correr o novamente o comando df. Aparecerá uma nova entrada, neste caso /dev/sdb1. sdb é o nome do device e 1 é o nome da partição.

```
df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1       119G   79G   34G  70% /
none            4.0K     0  4.0K   0% /sys/fs/cgroup
udev            7.8G   12K  7.8G   1% /dev
tmpfs           1.6G  1.1M  1.6G   1% /run
none            5.0M     0  5.0M   0% /run/lock
none            7.9G  1.5M  7.9G   1% /run/shm
none            100M  3.7M   97M   4% /run/user
/dev/sdb1       2.0G  0.0G  2.0G   0% /media/user/LABEL
```

Nota: o nome do cartão pode ser diferente.
Sabendo o nome do cartão, o passo seguinte é desmonta-lo:

```
sudo umount /dev/sdb1
```

O último passo é escrever para o cartão:

```
sudo xzcat ~/path-to-ev3_image/ev3dev.img.xz |
sudo dd bs=4M of=/dev/sdb
```

3. Passo 3: *Boot* ev3dev
   Colocar o cartão micro SD no EV3 e em seguida liga-lo. De início, os LEDs estarão de cor vermelha passando depois para laranja e é mostrada a imagem de um pinguim no ecrã. Os LEDs mostram a atividade do cartão SD. Passados cerca de 2 minutos, o ecrã ficará branco, isto acontece apenas no primeiro arranque. Quando terminar, os LEDs ficarão de cor verde e ecrã mostrará uma imagem similar à da figura 3.12.

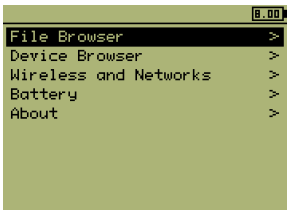


Figura 3.12: EV3 - janela de arranque do *kernel* ev3dev.

4. Passo 4: Criar a ligação ao EV3. A ligação que será apresentada é por cabo USB e permitirá quer a ligação por ssh ao EV3 como a ligação o EV3 à *internet*. Outras formas de ligação são apresentadas em [63].

   Em primeiro lugar, é necessário verificar se a *driver* CDC está ligada no EV3. Para tal, é necessário ir ao menu *Wireless an Networks* e selecionar *USB*. Caso a opção CDC esteja desativada é necessário ativa-la.

   No computador é necessário criar a rede para a ligação ao EV3 usando nas definições de IPv4 o método *Shared to other computers*.

   O passo seguinte é atribuir um endereço de IP ao EV3. No menu inicia selecionar *Wireless and Networks -> All Network Connections* e selecionar a ligação *Wired* que contém o símbolo USB. Na aba de IPv4 carregar na opção *Change...* e escolher *Load Linux defaults*. Para terminar, voltar à aba de *Conn.* e selecionar a opção *Connect automatically*. A partir deste momento a ligação ficará ativa e na aba IPv4 aparecerá o endereço IP do EV3 (neste exemplo foi 10.42.0.3).

5. Passo 5: Ligar ao EV3 por ssh.
   Abrir um terminal e correr o seguinte comando:

   ```
   ssh root@10.42.0.3
   ```

   Caso seja a primeira ligação aparecerá uma mensagem de autenticação. Para que a ligação seja validada, deverá ser aceite pelo utilizador escrevendo no terminal *yes*. Em seguida será pedida a *password* que é "r00tme". Após a autenticação será mostrado o seguinte ecrã de início:

   ```
   Linux ev3dev 3.3.0-0-ev3dev

                  _____      _
      _____    _|___ /  __| | _____    __
     / _ \ \ / / |_ \ / _` |/ _ \ \ / /
    |  __/\ V / ___) | (_| |  __/\ V /
     \___| \_/ |____/ \__,_|\___| \_/
   ```

```
Debian jessie on LEGO MINDSTORMS EV3!

The programs included with the Debian GNU/Linux system are
free software;the exact distribution terms for each program
are described in the individual files
in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the
extent permitted by applicable law.
root@ev3dev:~#
```

6. Passo 6: Atualizar a distribuição de ev3dev.
   No terminal de ligação ssh correr os comandos:

```
root@ev3dev:~# apt-get update
root@ev3dev:~# apt-get upgrade
root@ev3dev:~# reboot
```

A partir deste ponto já se pode programar usando uma linguagem imperativa como C ou C++;

A forma como este este *kernel* trata do sensores e atuadores da LEGO®é através do acesso (leitura e escrita) de ficheiros em áreas reservadas. Para facilitar o acesso e a programação usando este *kernel*, foi desenvolvida uma biblioteca de funções em C que trata dos sensores e motores de uma forma mais abstrata permitindo ao utilizador o desconhecimento do comportamento interno dos mesmos.

3.5.3 Programar EV3 usando FEUPAutom

Para além da biblioteca de funções desenvolvida, foi adaptado código para que seja possível programar o EV3 usando FEUPAutom. Desta forma, deixa de ser necessário possuir conhecimento de linguagem C para programar o EV3 passando a ser possível programar em Grafcet e ST. A estratégia aqui foi a de utilizar uma ferramenta do FEUPAutom que é capaz de converter Grafcet e ST em código C que é depois exportado automaticamente para o ficheiro "feupautom.c".

Os passos necessários para que esta passagem seja possível são:

1. Passo 1: fazer o *download* da última versão do FEUPAutom e do avrgcc disponível em [60].
   Descomprimir estes ficheiros e mover a pasta do avrgcc para a pasta do FEUPAutom.

2. Passo 2: correr o ficheiro FEUPAutom.exe.
   Deverá aparecer uma janela como a mostrada na figura 3.13.

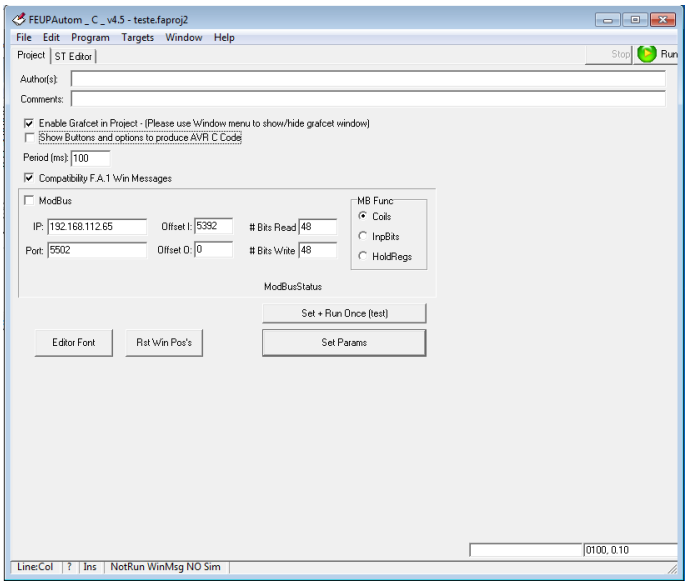


Figura 3.13: FEUPAutom - janela de arranque.

Em seguida carregar na opção *Show Buttons and options to produce AVR C Code*. Esta opção mostrará outras abas para além do *Project* e *ST Editor*.

3. Passo 3: Escolher o ambiente de programação.
   Para este o exemplo a ser mostrado foi usado Grafcet, mas para ST o processo é o mesmo. Para escolher o ambiente selecionar *Window -> Grafcet*. Neste momento, uma janela em branco para a edição em Grafcet será mostrada. Na figura é apresentada esta janela, já com um pequeno Grafcet exemplo.

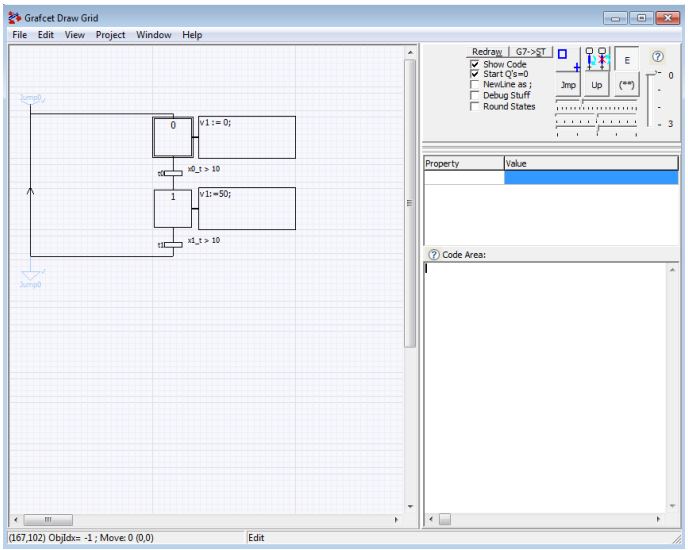


Figura 3.14: FEUPAutom - janela de edição de Grafcet.

4. Passo 4: Compilar o Grafcet.
   Para compilar o código selecionar *Project -> G7->ST->C*. Se não existirem erros de compilação, no diretório "avrgcc"dentro da pasta do FEUPAutom irá aparecer o ficheiro "feupautom.c".

5. Passo 5: Compilar e executar no EV3.
   Para executar o código produzido em Grafcet e exportado para C, "feupautom.c", é preciso copiar este ficheiro para o diretório do projeto em C criado previamente que está preparado para receber este ficheiro. Tendo este ficheiro basta correr o comando make para compilar e ./test para executar:

```
root@ev3dev:/home/EV3_bib# ls
c_header.c  ev3bib.c  ev3bib.h  feupautom.c  main.c  Makefile
root@ev3dev:/home/EV3_bib# make
gcc -c ev3bib.c
gcc main.c ev3bib.o -pthread -o test
root@ev3dev:/home/EV3_bib# ./test
```

O exemplo mostrado em Grafcet apenas liga e desliga um motor à cadência de 1 segundo. Foram efetuados outros testes em Grafcet como o seguidor de linha.

3.5.4 Programação Transparente Usando EV3

Para a aplicação do conceito da programação transparente no LEGO®EV3 foi necessário desenvolver os módulos de comunicação PC - Robô. A comunicação é feita com recurso a *sockets* TCP. Para isso, dotou-se o EV3 com uma *dongle WiFi*, figura 3.15, e implementou-se uma biblioteca de comunicação para este efeito.

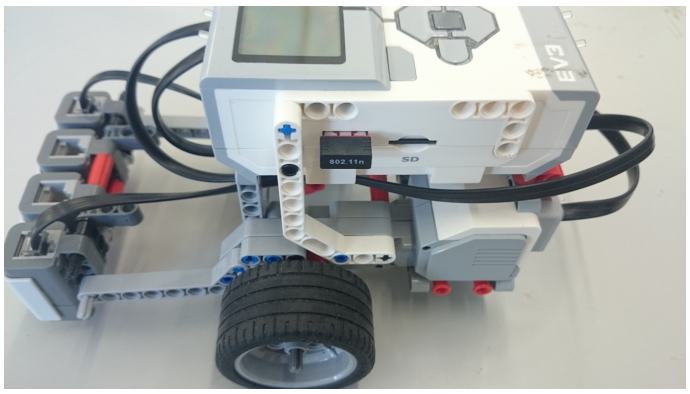


Figura 3.15: Módulo de comunicação usado no EV3.

Do lado do PC, elaborou-se uma pequena aplicação usando Lazarus IDE - Pascal para que a comunicação seja possível. Esta aplicação é apresentada na figura 3.16.

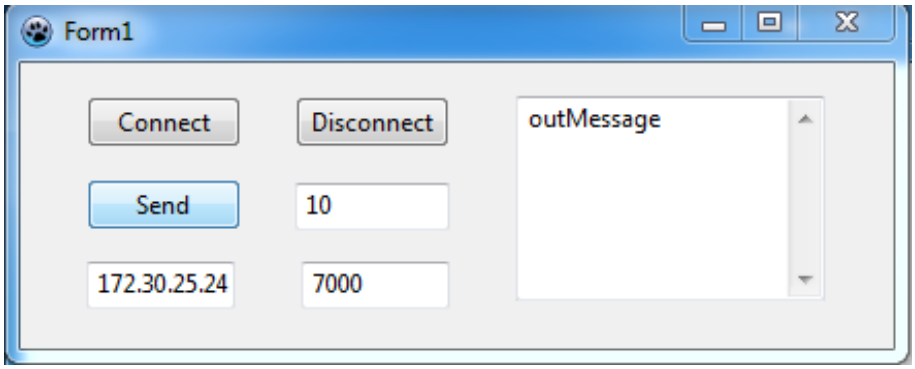


Figura 3.16: Programação Transparente - Aplicação Lazarus para comunicação EV3.

A arquitetura de comunicação usada é mostrada na figura 3.17.

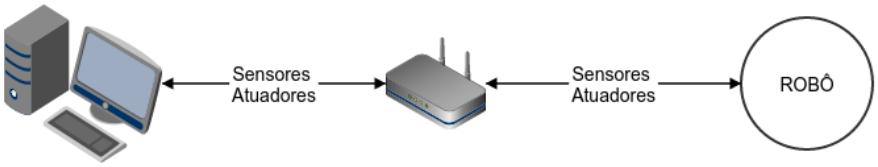


Figura 3.17: Programação Transparente - Diagrama de comunicação usado no EV3.

A comunicação entre PC - Robô via *socket* TCP é realizada com recurso a um *router*. As mensagens trocadas têm um formato estabelecido que é mostrado na tabela 3.1.

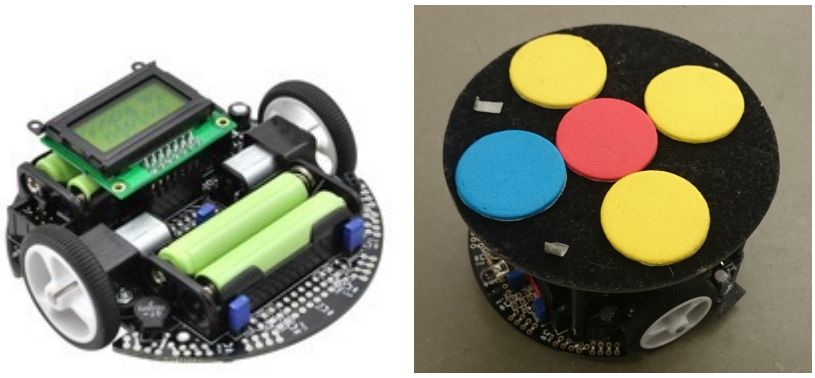
Tabela 3.1: Programação Transparente - Mensagens EV3.

Mensagem enviada pelo EV3	**Mensagem recebida pelo EV3**
<sensor_value_esq;sensor_value_dir>	<motor_value_esq;motor_value_dir>

Os valores enviados para o motor variam entre 0% e 100% enquanto que os valores dos sensores de infravermelhos entre 0 e 500 consoante o valor de intensidade refletido. Foram elaborados testes utilizando o mesmo controlo para um seguidor de linha usando o programação local e programação transparente usando o sistema de *tracking* de robôs. Os resultados são apresentados no capítulo 4.

3.6 Pololu 3pi

O 3pi é um robô da Pololu figura 3.18 de diâmetro 3π composto por dois micro motores, cinco sensores de infravermelhos para o seguimento de linha, um LCD de 8x2 caracteres, um *buzzer* e três botões de pressão. Todos estes componentes estão ligados a micro-controlador, ATmega328, programável em C.



(a) (b)

Figura 3.18: Robô Pololu 3pi original *(a)* e com capota *(b)*.

Este robô pode ser programado no software Arduino quer em Windows quer em Linux. Na secção 3.6.1 são mostrados os passos de como proceder para a instalação neste software nos dois sistemas operativos. Em alternativa, este robô também pode ser programado em FEUPAutom usando Grafcet ou ST como será mostrado na secção 3.6.2.

A avaliação de desempenho usando um controlador de histerese e linear com o 3pi com a capota, figura 3.18b, no sistema de *tracking* de robôs também foi feita estando os resultados no capítulo 4.

3.6.1 Programar 3pi usando Arduino (Windows + Linux)

Nesta secção é apresentado um micro tutorial de como instalar e usar o Arduino para programar o 3pi em Windows e Linux. Este tutorial é válido para programar este robô com um programador ISP.

1. Passo 1: Fazer *download* e instalar Arduino IDE.
   Em Windows fazer *download* do instalador do Arduino IDE na página [65]. Depois de o ter basta corre-lo e este instalará automaticamente.
   Para instalar este IDE em Linux, abre-se um terminal e corre-se o seguinte comando:

   ```
   sudo apt-get install arduino
   ```

2. Passo 2: Adicionar a placa 3pi e o programador ao Arduino IDE.
   Em Windows ir ao diretório de instalação do Arduino e procurar os ficheiros "boards.txt"e "programmers.txt". Estes ficheiros deverão estar no diretório "C:\Program Files\Arduino\hardware\arduino\avr". No ficheiro "boards.txt"acrescentar no final do ficheiro o seguinte código:

   ```
   ##############################################################
   orangutan48pgm.name=Pololu Baby Orangutan B-48 via Programmer
   orangutan48pgm.upload.using=avrispv2
   ```

```
orangutan48pgm.upload.maximum_size=4096
orangutan48pgm.build.mcu=atmega48
orangutan48pgm.build.f_cpu=20000000L
orangutan48pgm.build.core=arduino
orangutan48pgm.build.variant=standard
##############################################################
orangutan168pgm.name=Pololu 3pi robot w/ ATmega168 via Programmer
orangutan168pgm.upload.using=avrispv2
orangutan168pgm.upload.maximum_size=16384
orangutan168pgm.build.mcu=atmega168
orangutan168pgm.build.f_cpu=20000000L
orangutan168pgm.build.core=arduino
orangutan168pgm.build.variant=standard
##############################################################
orangutan328pgm.name=Pololu 3pi robot w/ ATmega328P via Programmer
orangutan328pgm.upload.using=avrispv2
orangutan328pgm.upload.maximum_size=32768
orangutan328pgm.build.mcu=atmega328p
orangutan328pgm.build.f_cpu=20000000L
orangutan328pgm.build.core=arduino
orangutan328pgm.build.variant=standard
```

No ficheiro "programmers.txt"acrescentar no final:

```
avrispv2.name=AVR ISP v2
avrispv2.communication=serial
avrispv2.protocol=avrispv2
```

Em Linux, o mesmo procedimento é aplicado. Estes ficheiros deverão estar do diretório "/usr/share/arduino/libraries/".

3. Passo 3 (opcional): Copiar os projetos exemplo para o Arduino IDE.
   Existem alguns projetos exemplo que podem ser usados. Estes projetos encontram-se disponíveis em [66]. Depois de feito o *download* destes projetos, é necessário copia-los para "C:\Program Files\Arduino\libraries\"em Windows, e em linux para "/usr/share/arduino/libraries/".

4. Passo 4: Programar e fazer *upload* usando Arduino IDE.
   No menu *Tools -> Board* seleciona-se *Pololu 3pi robot w/ ATmega328P via Programmer* para selecionar a placa. Seleciona-se o programador *AVR ISP v2* no menu *Tools -> Programmer*. Para finalizar seleciona-se a porta para programar que em Windows é normalmente designada por "COMxx"e em Linux por "/dev/ttyACMx". Para fazer upload do programa ir ao menu *File -> Upload Using Programmer*.

3.6.2 Programar 3pi usando FEUPAutom

É possível programar o 3pi usando o FEUPAutom. Para além dos passos 1,2,3 e 4 seguidos na secção 3.5.3, é necessário a instalação de alguns ficheiros adicionais. Os passos necessários são:

1. Passo 1: *Download* feupautomavrgccinstallers.zip na página [60] e extrair os ficheiros.

2. Passo 2: Instalar o programador ISP.
   Correr o ficheiro __1__pololu-usb-avr-programmer-win-121114.exe

3. Passo 3: Ligar o programador a uma porta USB.
   No menu Iniciar, navegar até ao "Device Manager"e mudar a porta "COMxx"para a porta COM13.

4. Passo 4: Instalar AVRStudio.
   Correr o ficheiro __2__AvrStudio4Setup.exe.

5. Passo 4: Instalar o compilador AVR GCC.
   Correr o ficheiro __3__avr-toolchain-installer-3.3.0.710-win32.win32.x86.exe

6. Passo 5: Instalar as *drivers* da Pololu.
   Correr o ficheiro __4__pololu-avr-bundle-140513.exe.

7. Passo 6: Copiar as bibliotecas e *includes* para os diretórios corretos.
   Correr o ficheiro __5__ajs_installer.bat como administrador.

Neste momento é possível construir um Grafcet no FEUPAutom, compilar e executar o código no 3pi.

3.6.3 Programação Transparente Usando 3pi

De forma análoga ao LEGO®EV3, implementou-se a programação transparente. Neste caso, as mensagens são passadas por comunicação RF recorrendo ao módulo de comunicação XBee que se encontra na capota, figura 3.19.

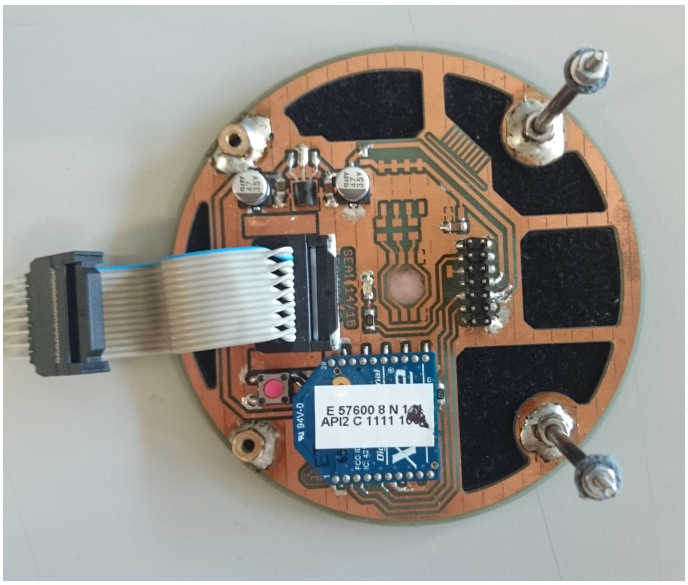


Figura 3.19: 3pi - Vista de baixo da capota.

A aplicação de comunicação do lado do PC foi desenhada, uma vez mais, com recurso usando Lazarus IDE e programada em Pascal. O aspeto da aplicação de teste para esta aplicação é apresentada na figura 3.20.

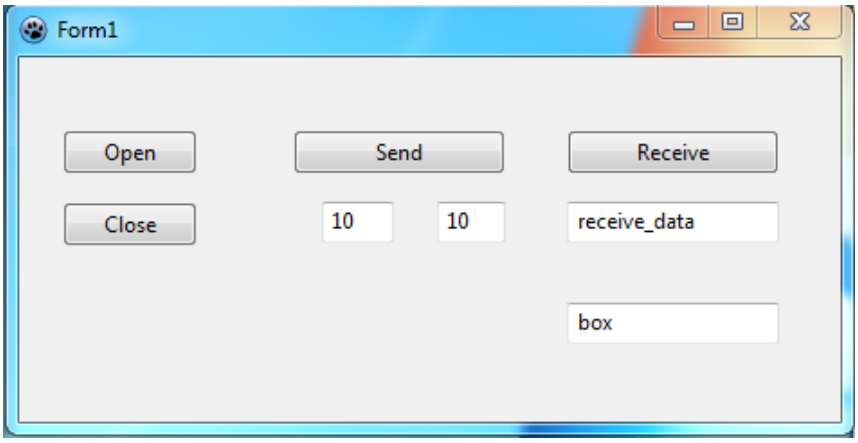


Figura 3.20: Programação Transparente - Aplicação Lazarus para comunicação 3pi.

Do lado do 3pi foi feito o código com recurso ao *software* do Arduino para a receção e envio das mensagens.

A escolha de Pascal para o envio e receção de mensagens possibilitou a integração deste modo de programação no FEUPAutom.

Quanto à arquitetura do sistema de comunicação por rádio frequência é apresentada na figura 3.21.

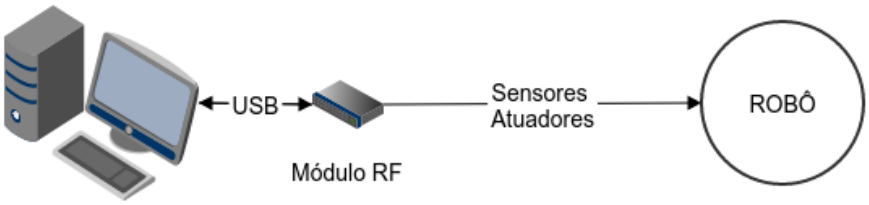


Figura 3.21: Programação Transparente - Diagrama de comunicações 3pi.

O envio e receção de mensagens é feito com recurso a dois módulos RF. Um dos módulos está acoplado no robô e o outro ligado a um PC por USB. Por este meio circulam as mensagens com informação dos sensores e motores. Note-se ainda que a comunicação é bidirecional. A tabela 3.2 especifica o formato das mensagens que são enviadas e recebidas.

Tabela 3.2: Programação Transparente - Mensagens 3pi.

Mensagem enviada pelo 3Pi	**Mensagem recebida pelo EV3**
<position_value>	<motor_value_esq;motor_value_dir>

O *position_value* varia entre 0 e 4000 sendo que o valor 2000 representa o robô alinhado com a linha. Os valores dos motores variam entre 0% e 100%.

3.7 Robô Telemóvel

O acesso aos telemóveis (*smartphones*) por parte da comunidade jovem é cada vez mais uma realidade. Neste momento, o telemóvel é quase um adereço indispensável no quotidiano desta comunidade. A ideia do robô telemóvel é tirar proveito das capacidades de processamento e periféricos que são inerentes a este tipo de equipamentos e fazer um *kit* que proporcione um mergulho no mundo da robótica. Este *kit* deverá ser fácil de montar, usar e programar, barato e entusiasmante para quem o usar. Neste sentido, esta secção mostra o processo de desenvolvimento e decisões tomadas para a construção deste robô.

3.7.1 *Hardware* e *Software*

Foi projetado um robô de tração diferencial que contém os seguintes componentes:

1. 2 servo motores (*Hobby Servos*) de rotação contínua + 2 rodas para a locomoção do robô;
2. 1 Arduino Uno para o controlo dos motores;
3. 1 módulo para comunicação *bluetooth* com o telemóvel;
4. 1 chassi para a montagem do robô.

Quanto à locomoção do robô, foi escolhida a tração diferencial devido à simplicidade do modelo cinemático associado à mesma que num contexto de aprendizagem direcionado à robótica é adequado.

Para o controlo dos motores, foi escolhido o Arduino Uno. Esta escolha foi mais uma vez direcionada ao contexto de aprendizagem. Uma vez que é possível programar uma placa deste tipo recorrendo a blocos numa estratégia de *drag and drop* utilizando o Visuino (*Visual Programming for Arduino*) ou o Ardublock. Nas figuras 3.23 e 3.22 são apresentados exemplos do aspeto do ambiente de programação de cada um destes *softwares*.

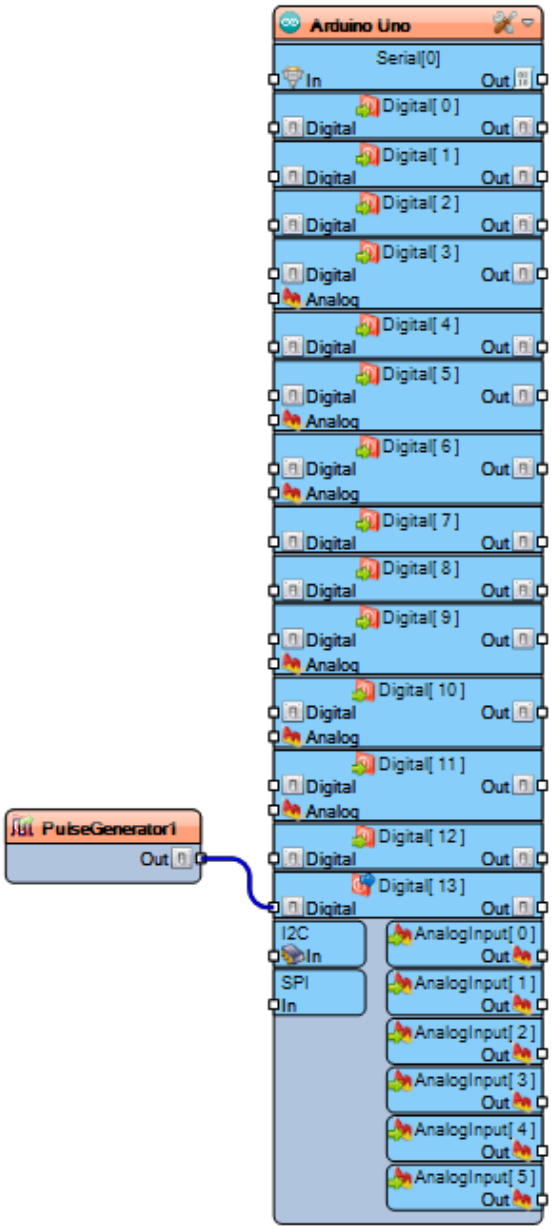


Figura 3.22: Visuino - exemplo de um LED a piscar.

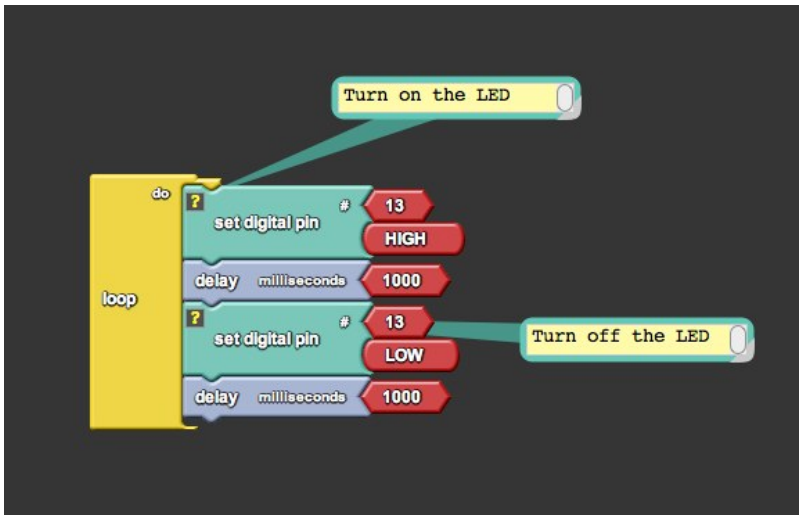


Figura 3.23: Ardublock - exemplo de um LED a piscar.

Por forma a ser possível a tomada de decisão na hora de colocar os estudantes mais novos a programar, foi elaborada a tabela 3.3 que compara estas duas abordagens numa série de pontos relevantes.

Tabela 3.3: Robô Telemóvel - Comparação entre Ardublock e Visuino.

	Ardublock	**Visuino**
Complexidade Ambiente Gráfico	+ Simples	+ Complexa
Facilidade de Utilização	+ Simples	+ Complexa
Curva de Aprendizagem	Menor	Maior
Simulação de Hardware	Não	Sim
Geração de Código	Sim	Sim

Para além dos pontos apresentados na tabela acima, existem outros dois que devem ser levados em conta como o público alvo e o contexto em que se insere o uso deste tipo de programação. Apesar da curva de aprendizagem ser maior no Visuino uma grande vantagem deste em relação ao Ardublock é a simulação de *hardware*. O Visuino oferece uma série de blocos como a geração de sinais analógicos e equipamentos de medida como pontas de osciloscópio que permite simular a utilização de um Arduino permitindo que otimizar o processo de desenvolvimento de *software*.

Existe outra possibilidade de programação do robô telemóvel sem recurso a um Arduino. Este método será apresentado na secção 3.7.4.

Em seguida foi decidido o método de comunicação entre a aplicação de *smartphone* e o Arduino. De entre os métodos de comunicação com e sem fios que poderiam ser escolhidos, foi escolhido o *bluetooth*. Este tipo comunicação está disponível na grande maioria dos *smartphones*.

Para finalizar, foi decido o sistema operativo sob o qual será feita a aplicação. Tendo em conta o mercado dos *smartphones*, Android é o que detém maior cota de mercado. Sendo assim, e tentando fazer chegar a aplicação a um público maior, foi decidido que nesta fase a aplicação será feita para o sistema operativo Android.

3.7.2 Arquitetura do sistema

Na figura 3.24 é apresentada a arquitetura do sistema que compõe o robô telemóvel.

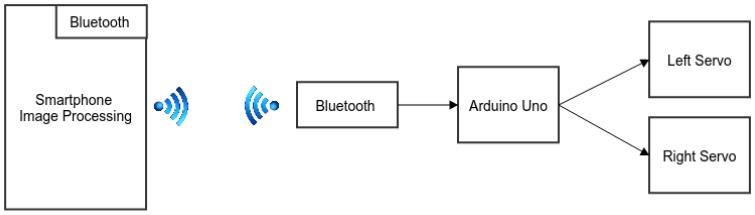


Figura 3.24: Robô Telemóvel - Arquitetura do sistema.

O *smartphone* adquire a imagem e deteta a linha que está a ver. Em seguida, calcula a distância a que está da linha e envia a via comunicação *bluetooth* para o Arduino. Recebendo esta informação, é calculada a velocidade para cada motor e o robô toma o movimento respetivo à velocidade calculada.

3.7.3 Esquema de Ligações e Montagem

O esquema de ligações entre o módulo *bluetooth*, Arduino Uno e servos é apresentado na figura 3.25. A alimentação deste sistema pode ser feita usando o *jack* ou o pino de vin disponíveis no Arduino.

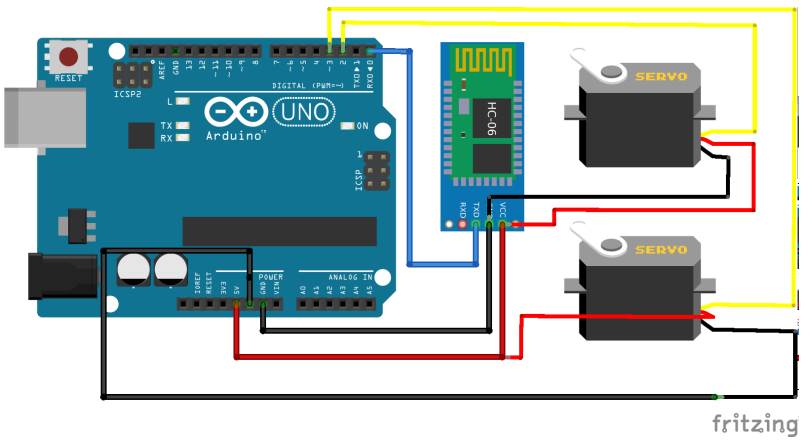
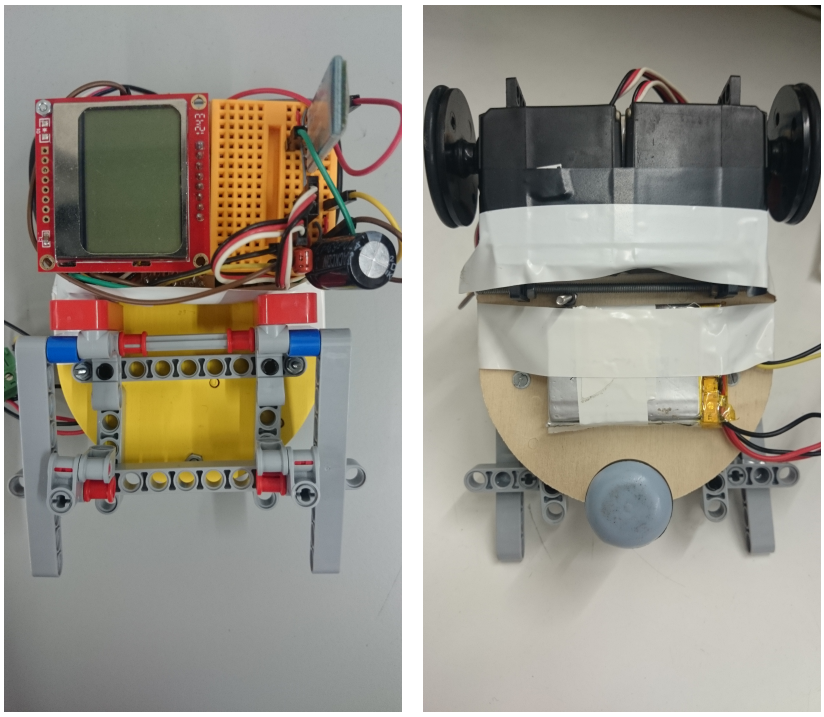


Figura 3.25: Robô Telemóvel - Esquema de ligações.

O protótipo elaborado do decorrer deste projeto é apresentado na figura 3.26. Neste caso foram usadas duas células de lítio-polímero 3.7V@850mA dispostas em série para a alimentação deste robô.



(a) (b)

Figura 3.26: Robô Telemóvel - vista de cima *(a)* e de baixo *(b)*.

Para além da placa do micro controlador, os dois servos e do módulo *bluetooth* foi acrescentado um LCD com o intuito de se fazer *debug*, caso necessário, e uma pequena construção em LEGO® que serve de suporte para o telemóvel.

O protótipo final montado com o telemóvel pode ser observado na figura 3.27.

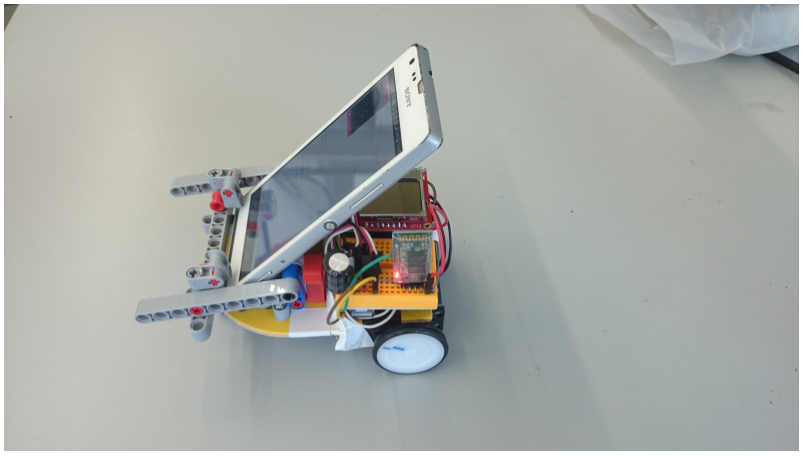
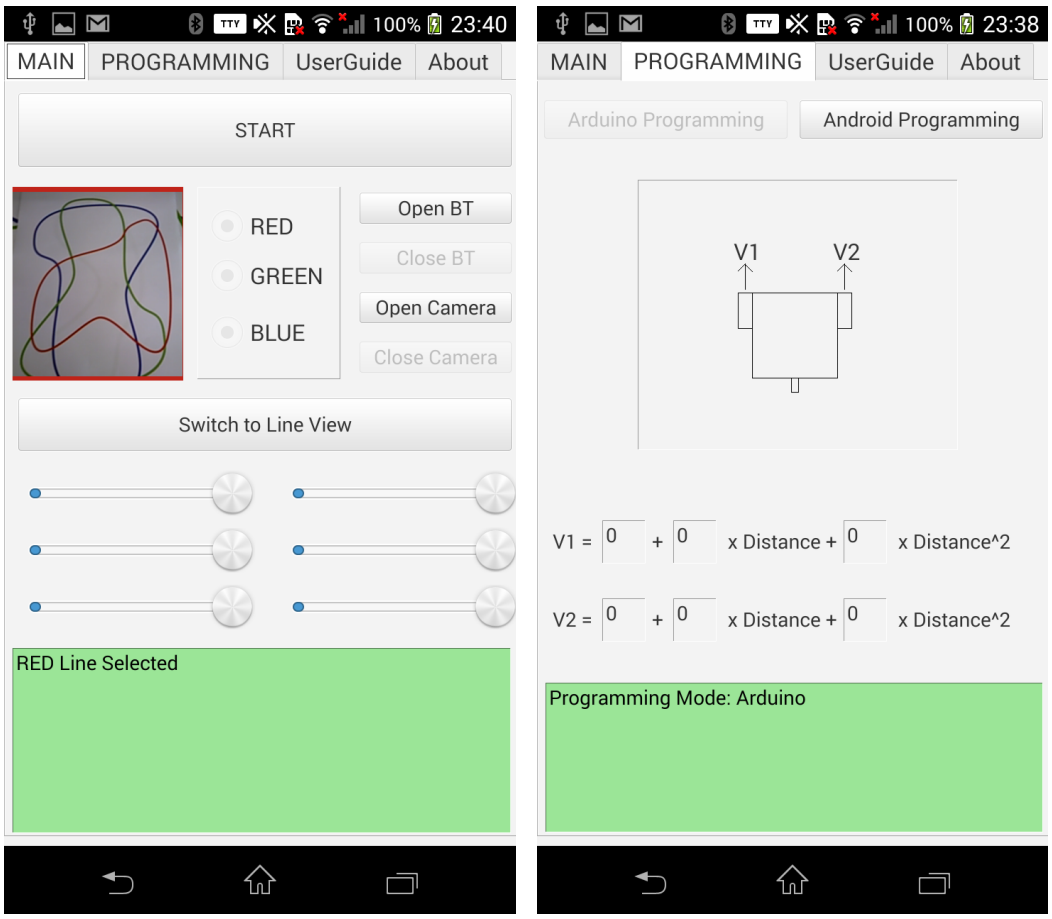


Figura 3.27: Robô Telemóvel - Protótipo final.

3.7.4 Aplicação Móvel

A aplicação móvel para a deteção de linhas de várias cores foi desenvolvida para o sistema operativo Android. A ferramenta de *software* usada para a criação desta aplicação foi o QtCreator. Esta ferramenta possibilita a criação de aplicações gráficas em C++. Para o processamento de imagem foi usada a biblioteca *opensource* de processamento digital de imagem OpenCV.

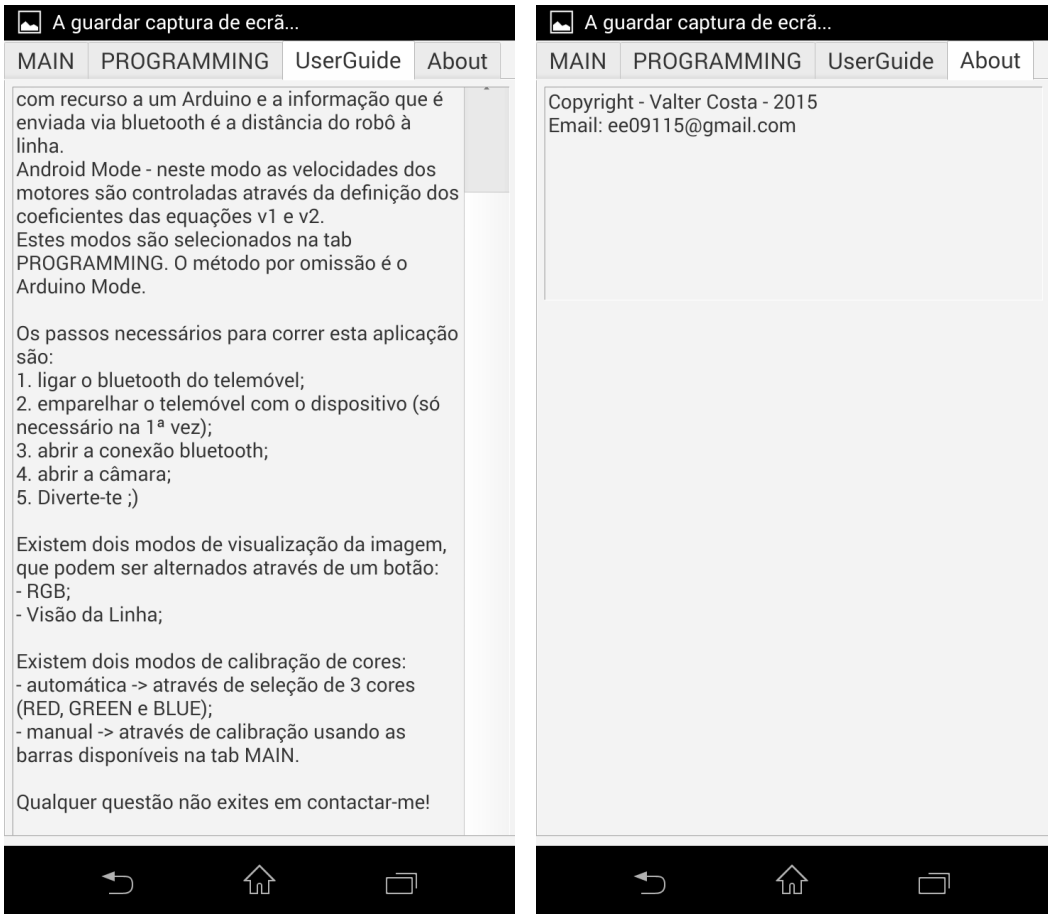
O aspeto final da aplicação é apresentado nas figuras 3.28.



(a) (b)

Figura 3.28: Robô Telemóvel/Aplicação - janela principal *(a)* e de programação *(b)*.

Esta aplicação permite a ligação ao módulo *bluetooth* conectado ao Arduino (depois de emparelhado), a abertura e fecho da câmara e a calibração automática e manual para a cor da linha que se pretende seguir. Contém ainda uma caixa de texto que informa o utilizador do estado atual da aplicação, e se existe algum erro. Na janela de *programming* é possível selecionar o método de programação, por Arduino ou no próprio telemóvel através da definição dos coeficientes das equações v1 e v2.



(a) (b)

Figura 3.29: Robô Telemóvel/Aplicação - janela do guia de utilização *(a)* e sobre *(b)*.

Para além da janela *MAIN* e *PROGRAMMING* existem mais duas janelas. A primeira, figura 3.29a, *USER GUIDE* que é um pequeno manual com instruções para utilização da aplicação. A segunda, 3.29b, com informações sobre o autor e o seu contacto. Os resultados do robô telemóvel são apresentados no capítulo 4.

Na próxima secção, será apresentado o orçamento do robô telemóvel.

3.7.5 Orçamento

Foi elaborado um orçamento de todo o equipamento utilizado na elaboração do robô telemóvel. A tabela 3.4 apresenta o orçamento obtido utilizando três vias para adquirir os componentes necessários.

Tabela 3.4: Robô Telemóvel - Orçamento.

	InMotion (euros)	**PtRobotics (euros)**	**Ebay (euros)**
Arduino Uno	20	23.75	5.38
Servo Motor c/ rodas	15.93 + 7.32	14.15 + 7.75	8.9
Battery Holder	1.11	2.03	1.08
Mini Breadboard	3.69	3.69	0.68
Bluetooth	12.92	12.92	3.55
Chassi	9.78	7.69	3.17
Total	92.89	91.85	31.66

3.8 Conde - Condução Autónoma

Este capítulo aborda vários aspetos relativamente ao sistema baseado em visão que foi desenvolvido para o robô de condução autónoma. Em primeiro lugar, feita uma introdução ao problema e é mostrada a arquitetura do sistema de alto nível com especial ênfase no campo da visão. Em seguida, são abordados alguns aspetos relativos a calibração de câmaras, *inverse perspective mapping*, como fazer o *tracking de linhas*, deteção de semáforos, algumas considerações de tempo real que otimizam as aplicações necessárias e um pequeno algoritmo de controlo para o movimento do robô.

3.8.1 Descrição do problema

A competição de condução autónoma consiste num robô completamente autónomo que percorre uma pista 2.1, deteta semáforos (que são projetados em dois monitores), sinais e desvia-se de obstáculos. O robô no qual foi implementado o sistema de visão desenvolvido está representado na figura 3.30.

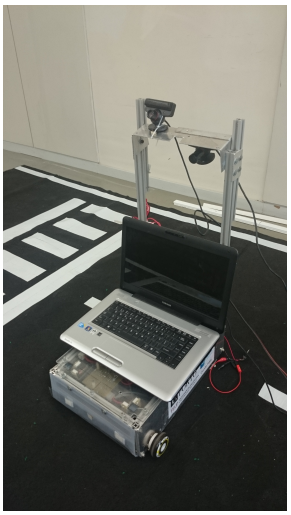


Figura 3.30: Conde - robô de condução autónoma.

O sistema baseado em visão desenvolvido terá que ser capaz de:

1. detetar e calcular a distância e ângulo à linha no referencial do robô;
2. enviar a informação de distância por mensagens UDP para a aplicação de controlo do robô;
3. identificar detetar semáforos;
4. enviar a informação dos semáforos por mensagens UDP para a aplicação de controlo do robô.

A distância e ângulo à linha pretendidos para a aplicação de controlo do robô estão ilustrados na figura 3.31.

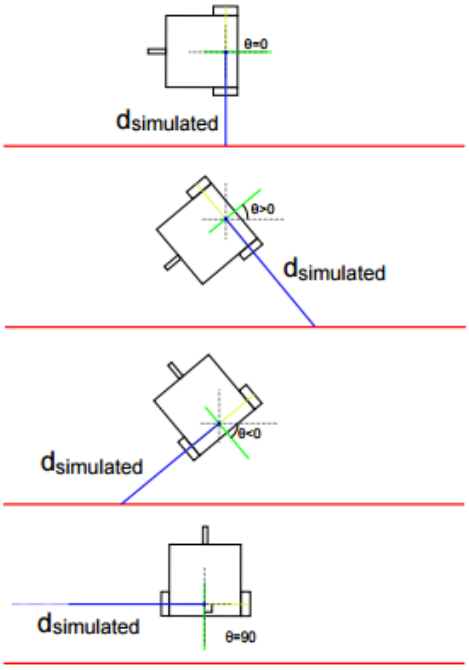


Figura 3.31: Cenários possíveis da distância e ângulo [6].

As medidas distância e ângulo que são enviadas para a aplicação de controlo têm de estar em unidades SI.

3.8.2 Arquitetura do Sistema

Quanto à arquitetura de baixo nível do robô, este é constituído por dois motores e respetivas *drivers* de potência e um arduino que envia as velocidades para cada motor. A tração deste robô é diferencial e a aplicação que faz as correções de posição na pista e toma as decisões que o robô deve seguir já está feita.

A arquitetura de alto nível do sistema está representado na figura 3.32.

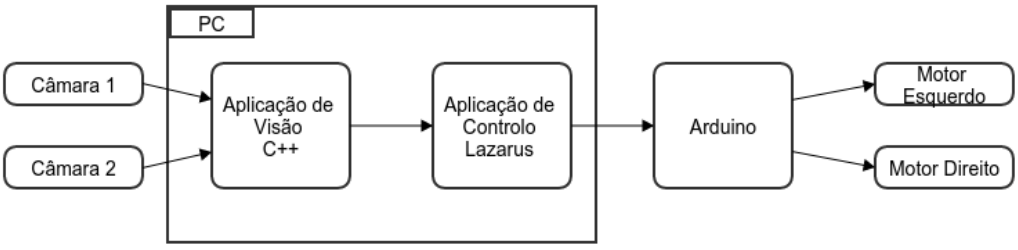


Figura 3.32: Arquitetura de alto nível do robô de condução autónoma.

O uso das duas câmaras advém do facto apenas com uma não se conseguir ver a pista e os semáforos ao mesmo tempo, especialmente na zona de início e fim das provas. Sendo assim, a

estratégia utilizada foi utilizar duas câmaras, uma para a deteção da pista e outra para a deteção de semáforos. Os algoritmos de *tracking* de linhas e deteção de semáforos bem como as suas implementações são exploradas nas próximas secções.

3.8.3 Algoritmo de *tracking*

Nesta secção é apresentado o algoritmo final de *tracking* de linhas do robô de condução autónoma.

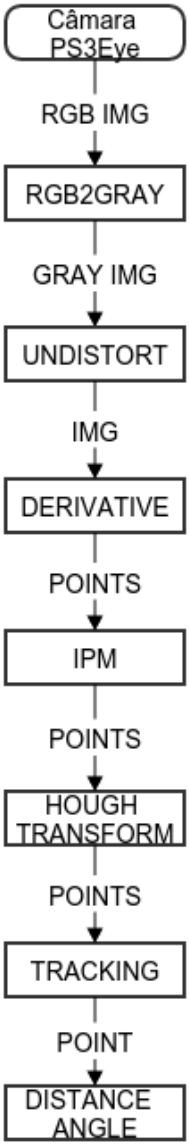


Figura 3.33: Diagrama de blocos representativo do algoritmo de *tracking* de linhas.

A figura 3.33 mostra o diagrama de blocos representativo do algoritmo de *tracking*. O primeiro passo é a captura de imagem feita pela câmara PS3Eye. Esta imagem vem numa resolução de

160*120 em RGB. Em seguida, esta imagem é convertida para escala de cinzentos. O terceiro passo é remover a distorção causada pela lente que foi acrescentada à câmara. Até este ponto todas as operações foram realizadas à imagem completa. Seguidamente, é aplicada a derivada horizontal à imagem por forma a realçar apenas os contornos das linhas. Todos os pontos que fazem parte da linha são guardados num vetor. A este vetor de pontos é feito o cálculo da remoção da perspetiva mapeando-os para uma imagem através do IPM. Nesta altura é aplicada a transformada de *Hough* para a deteção de linhas na imagem resultante do IPM. De entre as linhas retornadas é escolhida aquela que é a "melhor", isto é, aquela que não apresenta uma distância superior a um dado limite entre a anterior e a atual. Com base nessa linha, é calculada a distância e ângulo à linha no referencial do robô. Para finalizar, estes valores são enviados por UDP para a aplicação de controlo que fará o cálculo para a correção segundo a informação recebida.

3.8.4 Calibração de câmaras

Para a deteção de linhas, fazendo uma análise e testes preliminares surge a necessidade de aumentar o campo de visão da câmara de forma a que se consiga ter mais informação da pista. Uma solução possível é usar uma lente de forma a aumentar o campo de visão da câmara.

(a) (b)

Figura 3.34: Foto tirada sem lente *(a)* e com lente *(b)*.

Como é possível de observar na figura 3.34b consegue-se ter mais informação na imagem do que sem a lente 3.34a. Note-se que estas duas fotos foram tiradas com o robô e câmara na mesma posição. A desvantagem de acrescentar uma lente é a distorção provocada pela mesma. Esta distorção pode ser retirada fazendo uma calibração, obtendo os parâmetros intrínsecos da câmara. A calibração foi feita capturando várias fotos de um xadrez em várias posições e perspetivas diferentes e usando a *toolbox* de calibração de câmaras do MATLAB®. Esta ferramenta é bastante interessante pois dá ao utilizador a possibilidade retirar os parâmetros intrínsecos e extrínsecos da câmara e calcula o erro associado às fotos tiradas para calibração. Após a calibração da câmara a distorção provocada pela lente pode ser removida como é apresentado na figura 3.35.

(a) (b)

Figura 3.35: Foto tirada com lente *(a)* e remoção da distorção *(b)*.

As fotos foram tiradas com uma resolução de 160*120 pixeis. O erro de reprojeção associado em cada foto usada na calibração nunca é superior a 1 pixel. Este erro é a diferença entre um ponto chave detetado no padrão de calibração e o ponto correspondente no mundo projetado na mesma imagem.

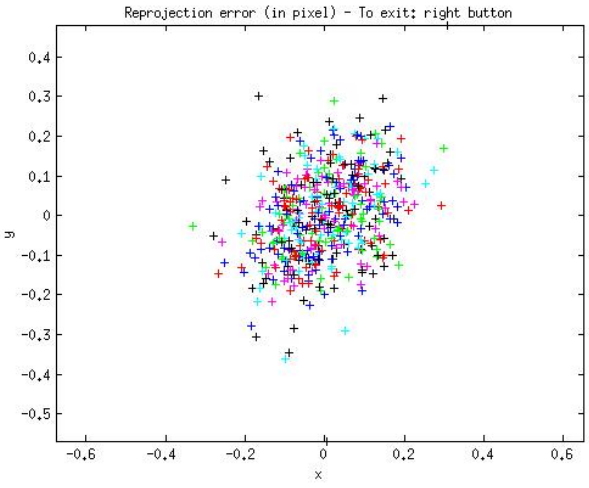


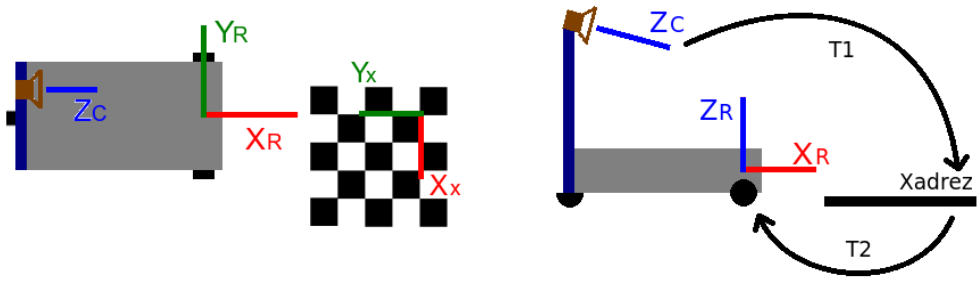
Figura 3.36: Erro associado a cada foto usada na calibração da câmara.

Este erro fornece uma medida qualitativa da precisão dos parâmetros obtidos. No caso de se obter um erro grande em alguma das fotos de calibração, pode-se remover essa foto e fazer a calibração novamente. Neste caso não passou de 0.5 pixeis em nenhuma das fotos, pelo que não foi necessário remover nenhuma das imagens de calibração.

Após a calibração dos parâmetros intrínsecos, o próximo passo é arranjar um método a partir do qual seja possível determinar com imagem de entrada a posição (x,y) em metros de um objeto no referencial do robô. Esta estratégia é abordada na próxima secção.

3.8.5 *Inverse Perspective Mapping*

IPM é uma transformação que projeta um espaço 3D num plano 2D. A estratégia adotada foi calcular a transformação do referencial da câmara para o referencial do xadrez representada por T1 da figura 3.37b e, em seguida a calcular a transformação do xadrez para o referencial do robô representada por T2 3.37b.



(a) (b)

Figura 3.37: Estratégia de Vista de topo *(a)* e de lado do robô *(b)*.

T1 foi feita com recurso ao cálculo dos parâmetros extrínsecos usando uma vez mais a *toolbox* de calibração de câmaras do MATLAB®. A transformação T2 foi calculada medindo as distâncias entre os referenciais xadrez/robô e retirando a matriz de rotação entre os mesmos. Uma vez tiradas estas duas transformações, para fazer a passagem pixel a pixel da imagem de entrada não distorcida para a imagem de saída (remoção da perspetiva no referencial do robô) é necessário fazer a transformação descrita em 3.1.

$$\begin{bmatrix} X \\ Y \\ Z \\ w \end{bmatrix} = \begin{bmatrix} p_{11} & p_{12} & p_{13} & -x \\ p_{21} & p_{12} & p_{23} & -y \\ p_{31} & p_{32} & p_{33} & -1 \\ a & b & c & -d \end{bmatrix}^{-1} \times \begin{bmatrix} -t_1 \\ -t_2 \\ -t_3 \\ -d \end{bmatrix} \tag{3.1}$$

$$P = K * T1_R * T2_R = \begin{bmatrix} p_{11} & p_{12} & p_{13} \\ p_{21} & p_{12} & p_{23} \\ p_{31} & p_{32} & p_{33} \end{bmatrix} \tag{3.2}$$

$$t = K * T_t = \begin{bmatrix} t1 \\ t2 \\ t3 \end{bmatrix} \tag{3.3}$$

$$K = \begin{bmatrix} \alpha_x & \beta & x_0 \\ 0 & \alpha_y & y_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.4}$$

$$a * X + b * Y + c * Z + d = 0 \tag{3.5}$$

Em que α_x e α_y são os fatores de escala da lente nos dois sentidos, x_0 e y_0 as coordenadas do ponto principal em pixeis e β o fator de assimetria na matriz K.

$T1_R$ e $T2_R$ são as matrizes de rotação das transformações $T1$ e $T2$.

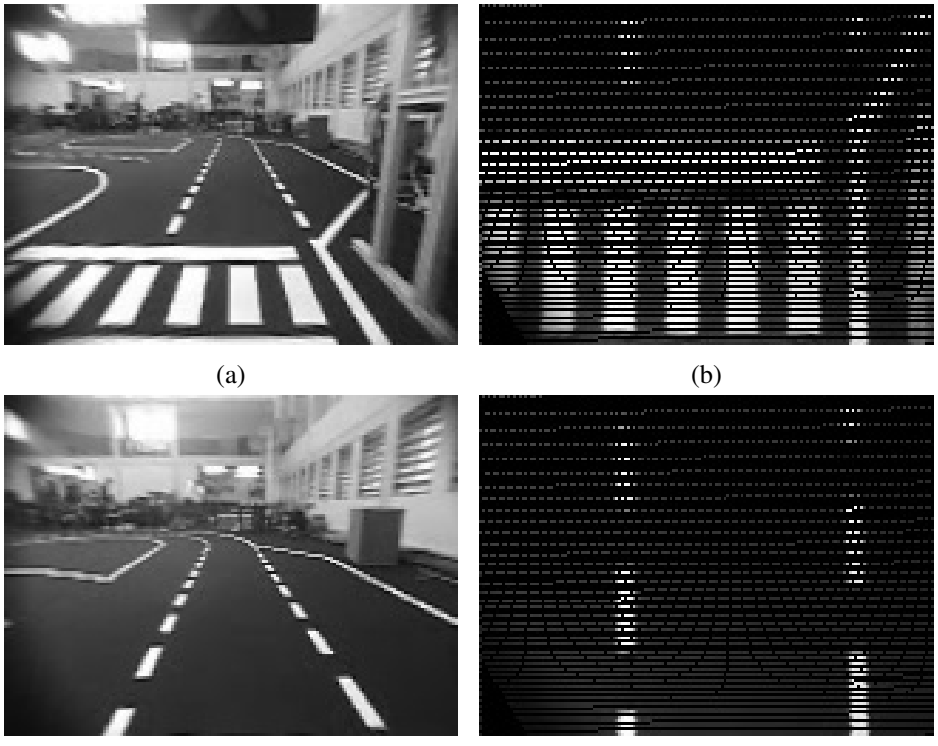
x e y são as coordenadas do pixel da imagem de entrada.

T_t é o vetor de translação resultante da multiplicação de $T1$ e $T2$.

a, b, c, d definem um plano genérico para a aplicação da transformação inversa, equação 3.5.

X e Y são as coordenadas métricas no referencial do robô.

Após a obtenção de X e Y, o ultimo passo é mapear estes pontos para uma imagem de forma a ter uma representação visual da aplicação desta transformação. Um exemplo do resultado obtido pode ser visto em 3.38.



(a) (b)

(c) (d)

Figura 3.38: Dois exemplos de aplicação do IPM. Imagens *(a)* e *(c)* tiradas da câmara removendo a distorção provocada pela lente e imagens *(b) e d)* obtidas pelo transformação IPM. Estas imagens foram tiradas durante a competição de condução autónoma 2015.

Esta abordagem é interessante pois permite a partir da imagem de entrada o cálculo direto de uma posição (x,y) em metros no referencial do robô. Note-se ainda que como a posição é dada em metros é possível delimitar uma janela de interesse no referencial do robô de forma a definir a que distância, frontal e lateral, se pretende ver.

3.8.6 *Tracking* de linhas

Após a transformação IPM, a passo seguinte é detetar as linhas obtidas na imagem resultante. Para tal, uma estratégia que pode ser seguida é a transformada de Hough para linhas. Esta transformada está implementada em *OpenCV* tendo duas vertentes:

- HoughLines() percorre todos os pontos da imagem de entrada e retorna todas as linhas possíveis tendo em conta os parâmetros de entrada. As linhas são retornadas por ordem decrescente de importância;

- HoughLinesP() a diferença para a implementação acima é a redução dos pontos da imagem de entrada. Em vez de visitar todos os pontos, visita apenas alguns pontos que são calculados aleatoriamente. Este método reduz assim o tempo de computação usado para o cálculo de linhas.

Tendo em conta o contexto deste trabalho e as preocupações de tempos associados às tarefas, a função utilizada para o deteção de linhas foi HoughLinesP().

Um exemplo de aplicação desta função é mostrado na figura 3.39.

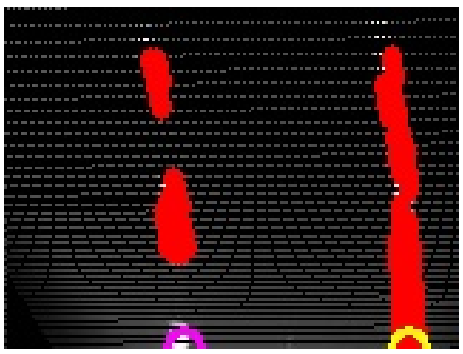


Figura 3.39: Deteção de linhas na imagem IPM. Linhas a vermelho são obtidas a partir da transformada de *Hough*. Os círculos amarelo e roxo mostram o ponto de *tracking* de linhas (central e da direita).

Após a deteção de linhas, é necessário detetar o ponto da linha a partir do qual será calculada a distância e ângulo à linha. A estratégia adotada é escolher de entre as linhas retornadas pela transformada de *Hough* aquela que pode será a melhor, isto é, a mais próxima da atual. Caso nenhuma linha cumpra esta condição, a anterior é mantida. Esta estratégia é aplicada quer para a linha do centro quer para a linha da direita. Em seguida, é calculado a distância e ângulo do ponto da linha direita (círculo amarelo na figura 3.39) ao referencial do robô. Para finalizar, estas medidas são enviadas por mensagens UDP para a aplicação de controlo do robô.

3.8.7 Deteção de Semáforos

Para além da câmara que está apontada para a pista, existe uma outra apontada para cima (figura 3.30) que é usada para a deteção de semáforos. Relembrando os semáforos que necessitam de ser detetados (figura 2.2) estes podem ser divididos em 3 classes consoante a cor que possuem.

3.8.7.1 Deteção do Semáforo de Paragem

Uma vez que o único semáforo vermelho que existe é representado na figura 3.40 a estratégia utilizada foi a de detetar uma grande mancha vermelha.

Figura 3.40: Semáforo de paragem do robô.

O algoritmo que deteta este semáforo é composto pelos seguintes passos:

1. Passar a imagem de RGB para HSV;
2. Calibrar o vermelho em HSV;
3. Calcular a área de vermelho detetado;
4. Se área for maior que um limite previamente definido, então é porque estamos na presença do semáforo de paragem.

A figura 3.41 mostra um exemplo de aplicação deste algoritmo.

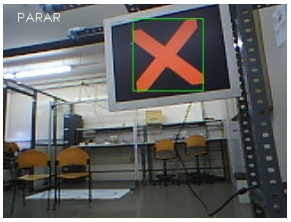


Figura 3.41: Exemplo de aplicação do algoritmo de deteção do semáforo de paragem.

3.8.7.2 Deteção do Semáforo de Parque

A estratégia aplicada ao sinal de estacionamento, figura 3.42 é a mesma que a aplicada no semáforo de paragem.

Figura 3.42: Semáforo de estacionamento do robô.

O algoritmo que deteta este semáforo é composto pelos seguintes passos:

1. Passar a imagem de RGB para HSV;
2. Calibrar o amarelo em HSV;
3. Calcular a área de amarelo detetado;
4. Se área for maior que um limite previamente definido, então é porque estamos na presença do semáforo de estacionamento.

A figura 3.43 mostra um exemplo de aplicação deste algoritmo.

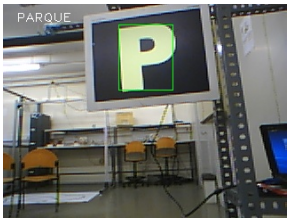
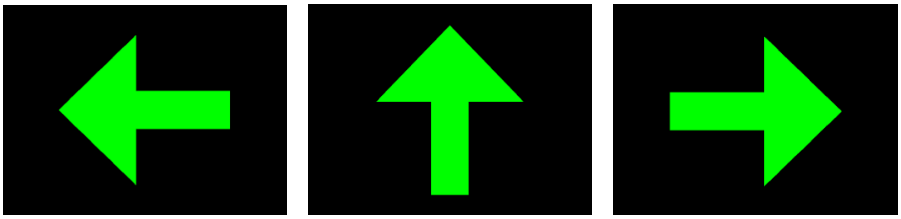


Figura 3.43: Exemplo de aplicação do algoritmo de deteção do semáforo de estacionamento.

3.8.7.3 Deteção dos Semáforos de Seguir em Frente, Esquerda e Direita

Para a deteção destes semáforos é preciso um pouco mais de processamento para a decisão de qual é o sinal que está a ser projetado.



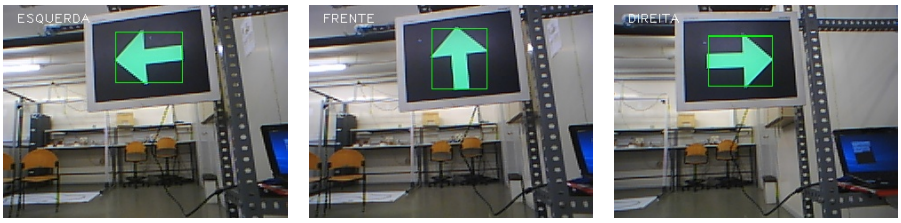
(a) (b) (c)

Figura 3.44: Semáforo indicativo de virar à esquerda *(a)*, direita *(c)* e semáforo indicativo de seguir em frente *(b)*.

O algoritmo que deteta estes semáforos é composto pelos seguintes passos:

1. Passar a imagem de RGB para HSV;
2. Calibrar o verde em HSV;
3. Calcular a área de verde detetado;
4. Dividir a área da região verde na vertical e horizontal e testar a relação entre as áreas das regiões para decidir qual o semáforo verde está a ser visto.

As figura 3.45 mostram a o resultado da aplicação para deteção dos semáforos verdes.



(a) (b) (c)

Figura 3.45: Exemplos de aplicação dos semáforo indicativos de virar à esquerda *(a)*, direita *(c)* e semáforo indicativo de seguir em frente *(b)*.

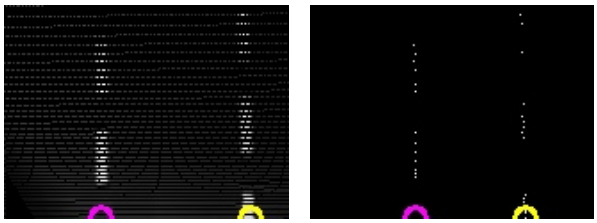
3.8.8 Visão Tempo Real

Uma das preocupações deste projeto, é que todas as aplicações de *software* desenvolvidas cumpram os requisitos temporais impostos. Neste caso, significa que aa informações de distância e ângulo à linha e dos semáforos passadas têm de ter utilidade quando chegam à aplicação de controlo do robô. Neste sentido, foi necessário reduzir a complexidade temporal da aplicação de *tracking* de linhas. A estratégia aqui adotada, foi a de passar uma única vez pela imagem principal de forma a retirar os pontos de interesse e ir trabalhando cada vez com menos pontos até chegar ao resultado final. Esta técnica é conhecida na área de visão em tempo real como *zero copy, one pass*.

No algoritmo de *tracking* de linhas, a remoção de perspetiva, IPM, é feita visitando cada pixel da imagem de entrada e calculando a transformação inversa. Este passo, apesar de necessário, é computacionalmente caro. Sendo assim, para tentar reduzir este tempo de execução, é necessário reduzir a quantidade de informação de entrada. Uma vez que é a ideia é detetar as linhas brancas em fundo preto, em vez de passar a imagem não distorcida toda para a remoção de perspetiva, a estratégia adotada foi:

- percorrer a imagem não distorcida detetando as transições de branco para preto e guardar esses pontos num vetor;;
- calcular a transformação inversa, IPM, para esses pontos;
- detetar as linhas usando *HoughLinesP()*;

A figura 3.46 mostra a transformação inversa de perspetiva completa da imagem não distorcida e usando a técnica acima descrita.



(a) (b)

Figura 3.46: Resultado do *tracking* de linhas sem otimização *(a)* e otimizado *(b)*

Esta otimização resultou numa redução de tempos bastante interessante que é apresentada no capítulo 4.

Para além do cálculo dos tempos de execução do algoritmo de *tracking*, foi calculado o tempo de *lag* das câmaras usadas. O método usado foi gravar um vídeo a alta resolução, neste caso 320*240 @ 187 FPS, em que os objetos da cena são uma luz ao lado de um PC que contém outra câmara a mostrar a imagem da luz. O *setup* montado para medir este tempo está representado na figura 3.47.

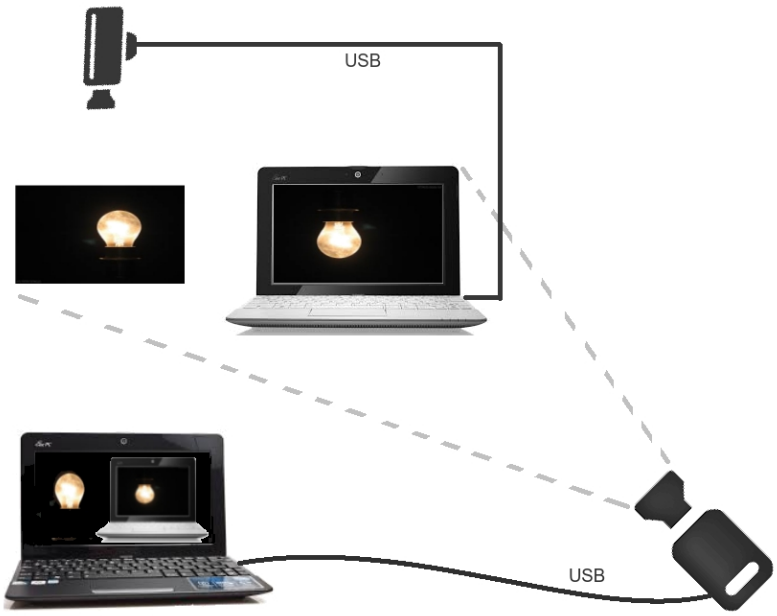


Figura 3.47: *Setup* utilizado para a medição do tempo de *lag*.

A ideia é ligar e desligar lâmpada e contar o número de *frames* que demora até que a imagem da lâmpada atualize consoante a mudança de estado. Os resultados da medição destes tempos estão no 4.

3.8.9 Algoritmo de Controlo do Robô de Condução Autónoma

Com o decorrer da competição de condução autónoma de 2015, e devido a algumas dificuldades surgidas, foi necessário fazer a aplicação de controlo do robô para que este andasse. Nesta secção, é explicado controlador usado e a sua integração no sistema. A aplicação desenvolvida consiste num nó de ROS que contém um controlador do tipo Proporcional-Integral que gera uma referência de velocidade para cada um dos motores. O diagrama de blocos que representa este sistema realimentado está na figura 3.48.

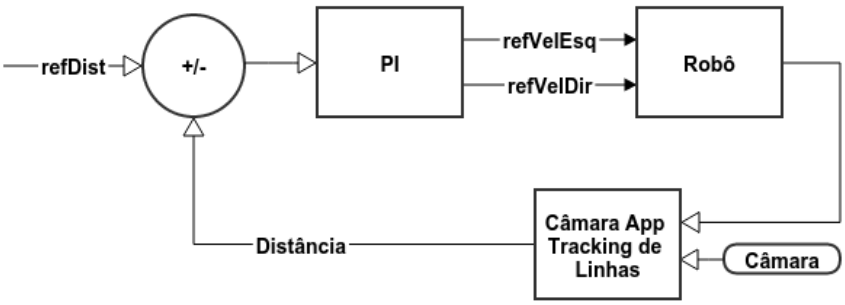


Figura 3.48: Diagrama de blocos do sistema, robô, realimentado.

Com esta alteração, o diagrama de blocos de alto nível do sistema é o seguinte:

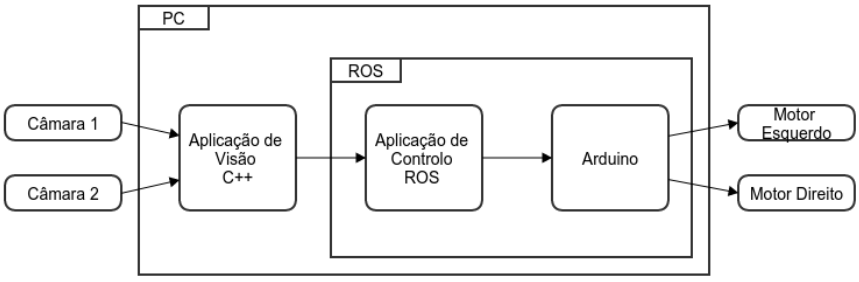


Figura 3.49: Arquitetura final de alto nível do robô de condução autónoma.

As câmaras adquirem as imagens, correspondentes à linha e semáforos, estas são processadas e como resultado é enviada a informação do semáforo caso exista e a distância à linha para o nó de ROS que trata do controlo do robô. Esta informação é enviada com recurso a mensagens UDP. Em seguida, baseada nesta informação este nó calcula uma referência de velocidade para cada motor e publica esta referência noutro nó de ROS que coloca cada motor do robô a rodar à velocidade desejada.

3.9 Inquérito

Nesta secção é apresentado um inquérito que foi realizado aos participantes da Universidade Júnior 2015. Foi-lhes apresentados os cinco casos de estudo desta dissertação e pedido que respondessem às seguintes questões:

1. Assinala com um 'x' de 1 a 5 o nível de interesse de cada demonstração/robô apresentado.
2. Dos cinco projetos mostrados, qual é a demonstração/robô que achaste mais interessante? E porquê?
3. Qual é o robô mais complicado?
4. Qual é o robô mais acessível?
5. Qual é o robô que eu gostava de ter em casa para programar? Porquê?

Os resultados deste inquérito estão no capítulo 4.

3.10 Considerações Finais

Neste capítulo foi apresentado o trabalho desenvolvido e processo para o alcançar.

Em primeiro lugar foi apresentado o REDI, robô de ensino didático, que é especialmente interessante para demonstrações rápidas em ambiente não estruturado.

Em seguida, o LEGO®Mindstorms EV3 que tem a vantagem de despreocupar a quem o usa das questões de eletrónica e possibilita a montagem fácil. Este robô enquadra-se no contexto de trabalho prolongado. A programação é feita recorrendo a blocos e foi mostrada como é possível programar este robô usando uma abordagem de máquina de estados. Foi ainda mostrado como instalar um sistema operativo baseado em Linux e como programa-lo usando uma linguagem imperativa e não imperativa usando o Grafcet no FEUPAutom. Foi ainda apresentada a estratégia de programação transparente. Para terminar foi apresentado um projeto de cooperação entre FEUP-ESAS para incentivo, divulgação e captação de estudantes para entrar mundo da robótica.

Seguidamente apresentou-se o 3pi, robô da Pololu. Este robô sendo fácil de programar em linguagem imperativa é também usado num contexto de trabalho prolongado. Foram apresentados dois métodos alternativos para a programação deste robô, seja por linguagem imperativa em Arduino ou por Grafcet usando o FEUPAutom. Foi também aplicada a estratégia de programação transparente.

Para estes três casos de estudo foi acrescentada uma capota identificativa para usar o sistema de *tracking* de robôs já desenvolvido no seguimento de uma pista implementando dois tipos de controlo (linear e histerese) para posterior análise de performance.

O quarto caso de estudo apresentado foi o robô telemóvel. Este robô tira proveito das capacidades de um telemóvel, a partir da aplicação que foi desenvolvida, e de um protótipo simples para

a introdução à Robótica. A programação deste protótipo pode ser feita através do Arduino ou no próprio telemóvel.

Para finalizar os casos de estudo foi apresentado o processo de desenvolvimento da aplicação de visão para o robô de condução autónoma - Conde. Sendo este robô de complexidade superior aos anteriores, pode ser usado como ferramenta para cativar os estudantes de ensino secundário. Adicionalmente, foi apresentada o método para calcular o *lag* dos dois sistemas de *tracking*.

Por fim, realizou-se um inquérito com a finalidade de retirar algumas conclusões sobre os casos de estudo analisados nesta dissertação.

Capítulo 4

Resultados

Neste capítulo são apresentados os resultados sobre os casos de estudo deste projeto. Em primeiro lugar, são comparados os dois tipos de controlo para os casos de estudo apresentados. Por forma a caracterizar o sistema de *tracking* utilizado são apresentados os erros do seguimento dos robôs calculado o *lag* deste sistema. São ainda apresentados os resultados do sistema baseado em visão de tempo real feito para o robô de condução autónoma bem como o erro do sistema de *tracking* de linhas desenvolvido.

4.1 *Tracking* de robôs - Erros

Esta secção apresenta a comparação entre os métodos de controlo, linear e de histerese, para os casos de estudo deste projeto. Esta comparação é feita com recurso à representação das trajetórias efetuadas pelos robôs na pista.

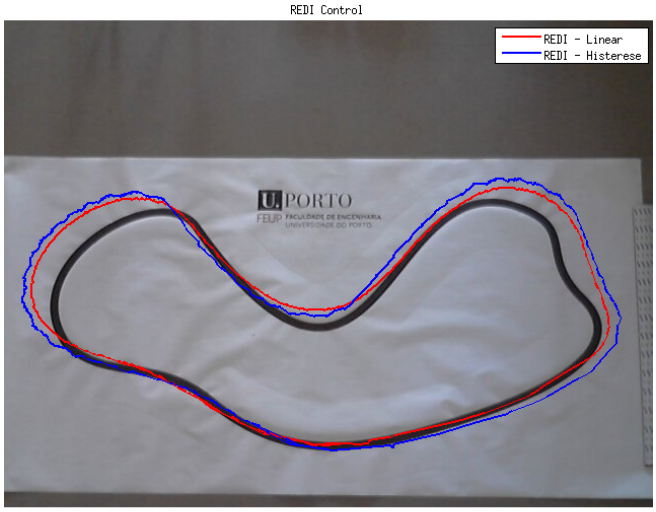


Figura 4.1: REDI - controlo linear *vs* controlo histerese.

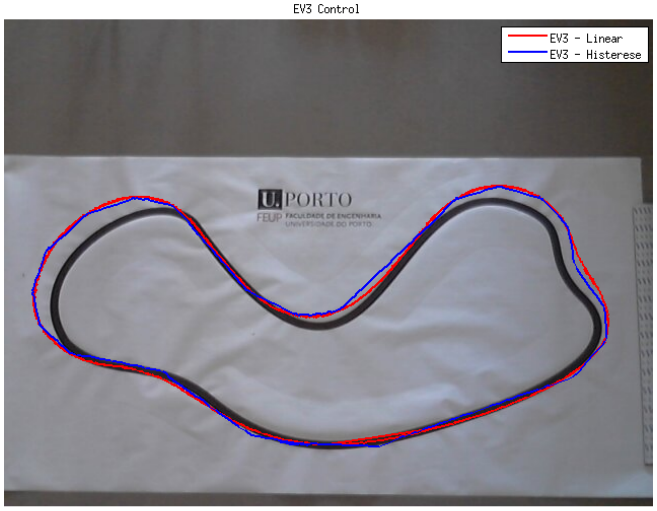


Figura 4.2: EV3 - controlo linear *vs* controlo histerese.

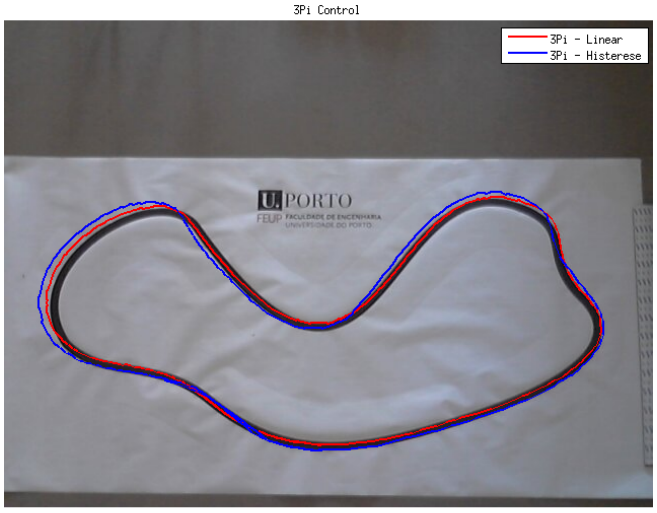


Figura 4.3: 3Pi - controlo linear *vs* controlo histerese.

Nas figuras 4.1, 4.2 e 4.3 estão representadas as trajetórias dos diversos robôs utilizando controladores lineares a vermelho nas figuras, e controladores de histerese a azul. É possível observar que em cada um deles a trajetória linear é mais suave nas correções efetuadas por cada robô do que nos controladores de histerese. Por forma a poder comparar os controladores nas duas abordagens, optou-se por representar para os três robôs os dois métodos de controlo, figuras 4.4 e 4.5.

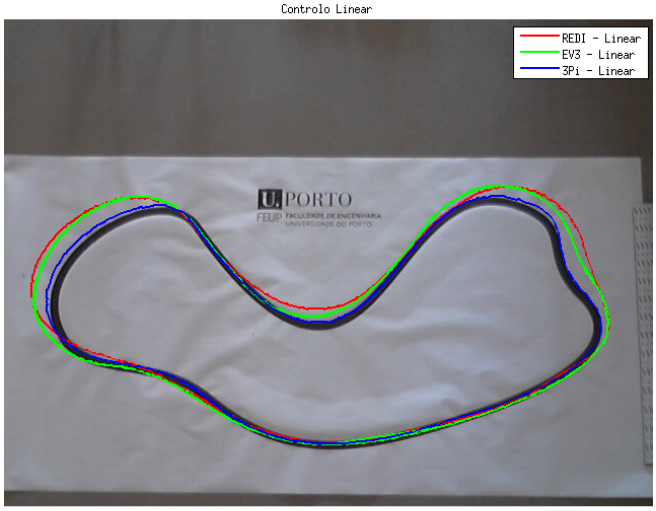


Figura 4.4: REDI, EV3 e 3Pi - controlo linear.

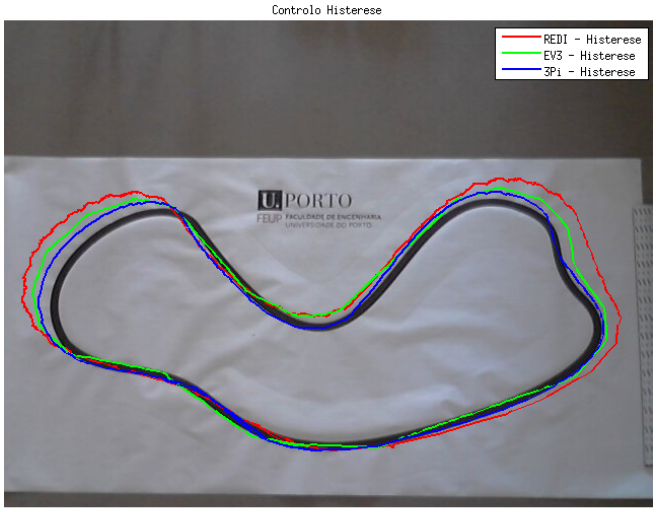


Figura 4.5: REDI, EV3 e 3Pi - controlo de histerese.

Nestas duas figuras é possível comparar o desempenho de cada robô. É observável que aquele que apresenta maior erro é o REDI, seguido pelo EV3 sendo o 3pi aquele que apresenta menor erro.

Esta informação é complementada pelo cálculo do integral do erro quadrático das duas abordagens apresentado nas tabelas 4.1 e 4.2.

Tabela 4.1: Resultados do integral do erro quadrático para o controlo por histerese.

	Integral Erro Quadrático
REDI	306868
EV3	109806
3pi	104458

Tabela 4.2: Resultados do integral do erro quadrático para o controlo linear.

	Integral Erro Quadrático
REDI	162585
EV3	107111
3pi	18817

4.1.1 Programação Local - Programação Transparente

Por forma a poder comparar o desempenho do conceito de programação transparente, submeteu-se o robô EV3 ao seguinte teste: utilizando o *setup* do sistema de *tracking* de robôs e a mesma pista colocar o mesmo código de controlo a executar localmente e no PC. O resultado deste teste pode ser observado na figura 4.6.

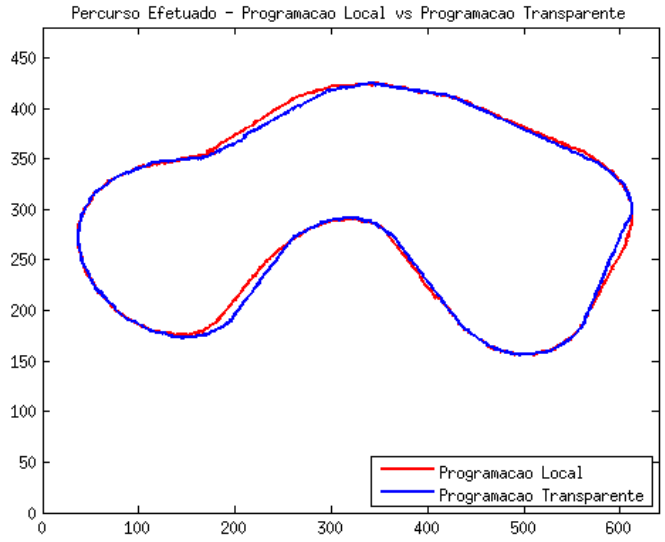


Figura 4.6: Percurso efetuado pelo EV3 executando o mesmo código localmente e no PC.

De forma a poder comparar de uma forma mais rigorosa as abordagens de programação em causa, calculou-se o erro relativo destas duas abordagens e a diferença entre elas. Este resultado pode ser observado na figura 4.7.

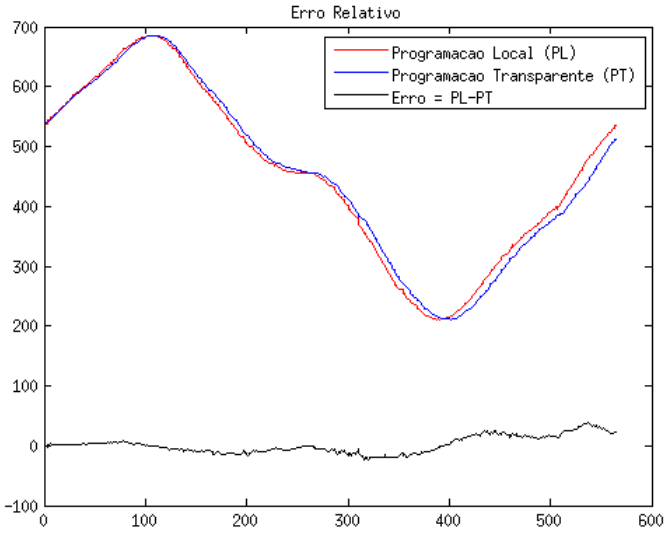


Figura 4.7: Erro Relativo - Comparação programação local e transparente.

Foi ainda calculado o erro médio e erro médio absoluto que são apresentados na tabela 4.3.

Tabela 4.3: Programação Transparente - Erro médio e erro absoluto médio.

Erro médio (pixeis)	**Erro absoluto médio (pixeis)**
1.5690	11.675

De forma análoga, foram registados os percursos efetuados pelo 3pi na versão local e transparente. Os resultados deste percurso são mostrados na figura 4.8.

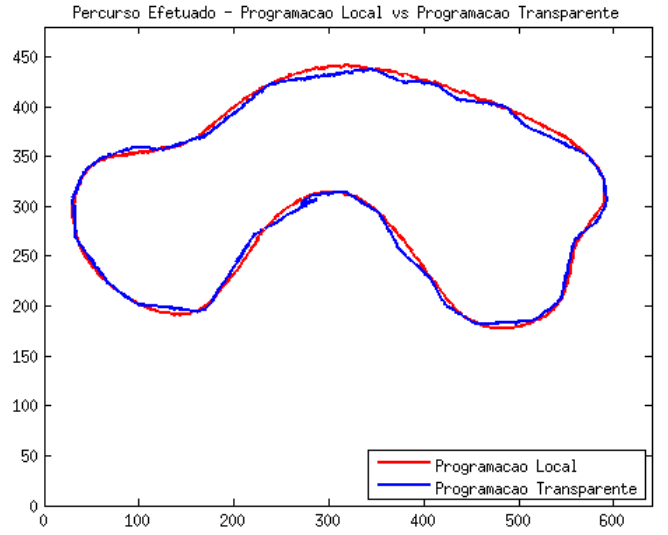


Figura 4.8: Percurso efetuado pelo 3pi executando o mesmo código localmente e no PC.

Foi também calculado o erro relativo no robô 3pi, figura 4.9.

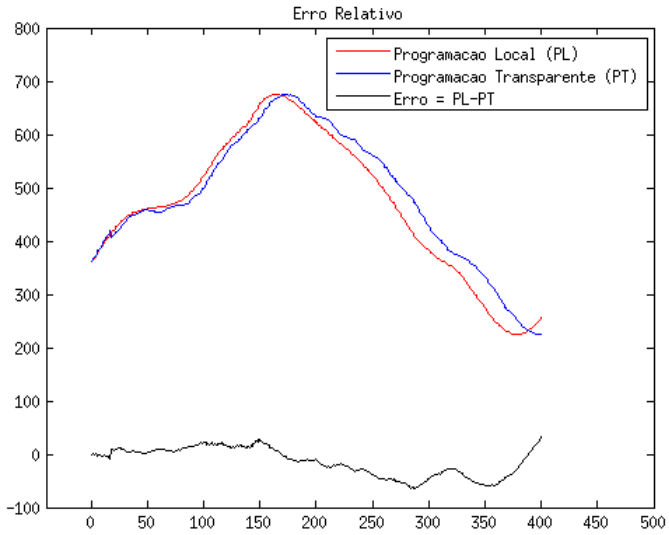


Figura 4.9: Erro Relativo - Comparação programação local e transparente.

Por fim, foi calculado o erro médio e erro médio absoluto do robô 3pi. Estes resultados são apresentados na tabela 4.4.

Tabela 4.4: Programação Transparente - Erro médio e erro absoluto médio.

Erro médio (pixeis)	**Erro absoluto médio (pixeis)**
-14.1430	24.4084

Observando os resultados aqui apresentados, é possível concluir que a abordagem da programação transparente pode ser aplicada, pois não coloca em causa o normal funcionamento dos robôs.

4.2 Visão Tempo Real

Esta secção dedica-se à apresentação dos resultados obtidos da otimização feita para o algoritmo de *tracking* de linhas. O mesmo algoritmo foi testado em quatro processadores diferentes. Os resultados são apresentados na tabela 4.5.

Tabela 4.5: Tempo médio de execução da aplicação da transformação inversa de perspetiva não otimizado e depois de otimizado.

PC	***Slow IPM Time* (ms)**	***Fast IPM Time* (ms)**	**Rácio (%)**
ROG	80.74	0.4854	16634
EeePC	587.7	3.100	18960
RaspberryPi	2970	15.52	19131
RaspberryPi2	1344	7.174	18741

A tabela 4.6 apresenta o tempo médio de execução para a aplicação de *tracking* de linhas.

Tabela 4.6: Tempo médio de execução da aplicação de *tracking* de linhas com IPM não otimizado e depois de otimizado.

PC	***Slow Tracking Time* (ms)**	***Fast Tracking Time* (ms)**	**Rácio (%)**
ROG	84.56	1.424	5936
EeePC	604.2	8.599	7025
RaspberryPi	3209	45.60	7037
RaspberryPi2	1366	22.98	5946

Observando estas tabelas, verifica-se uma redução significativa no tempo de execução da aplicação de *tracking* de linhas. No caso do ROG, o tempo de execução ficou 59 vezes mais rápido, no EeePC e RaspberryPi a melhoria foi de 70 vezes. Tal melhoria, deve-se à aplicação da "filosofia"de processamento de imagem em tempo real: *zero copy one pass*. Este tempo não contabiliza o tempo de passagem da informação da câmara por USB. As características dos PCs usados estão no anexo A.

Por forma a caracterizar o sistema de *tracking* de robôs e de linhas, calcularam-se o *lag* das câmaras que estes sistemas usam. Esta informação é apresentada na tabela 4.7.

Tabela 4.7: *Lag* do sistemas de *tracking*.

Câmara	***Average Lag* (frame)**	***Average Lag* (ms)**
PS3Eye	9.75	52.14
Logitech C270	13.75	73.53

4.3 *Tracking* de linhas - Erros

Nesta secção é mostrado o erro de distância e ângulo do sistema de *tracking* desenvolvido para o robô de condução autónoma. Os erros estão representados na forma de gráficos nas figuras 4.11 e 4.13 .

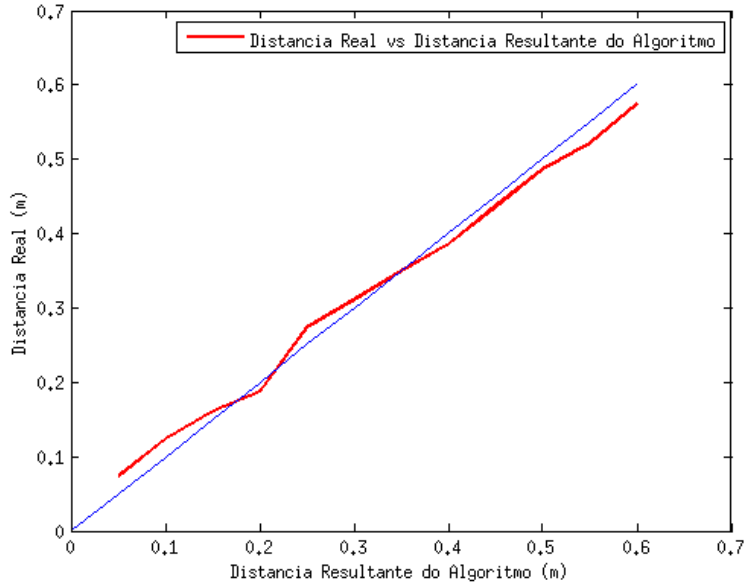


Figura 4.10: Gráfico que relaciona medidas reais de distância com as medidas obtidas pelo algoritmo.

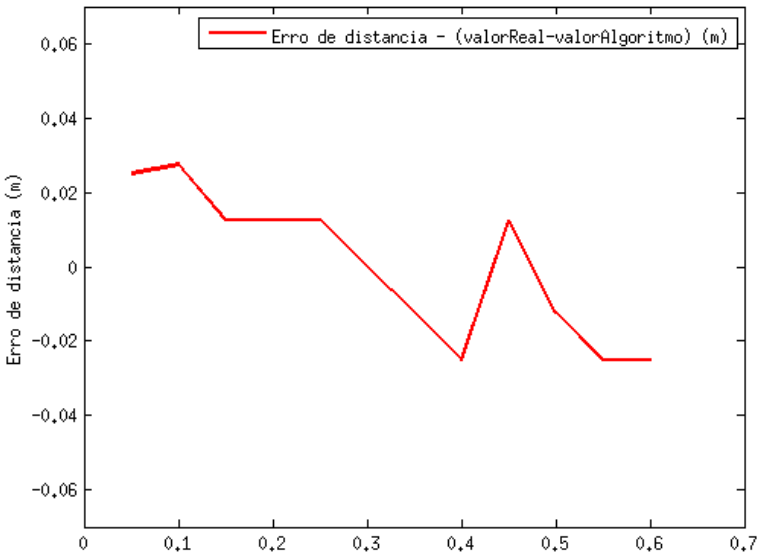


Figura 4.11: Erro de distância obtido entre as medidas reais e obtidas pelo algoritmo.

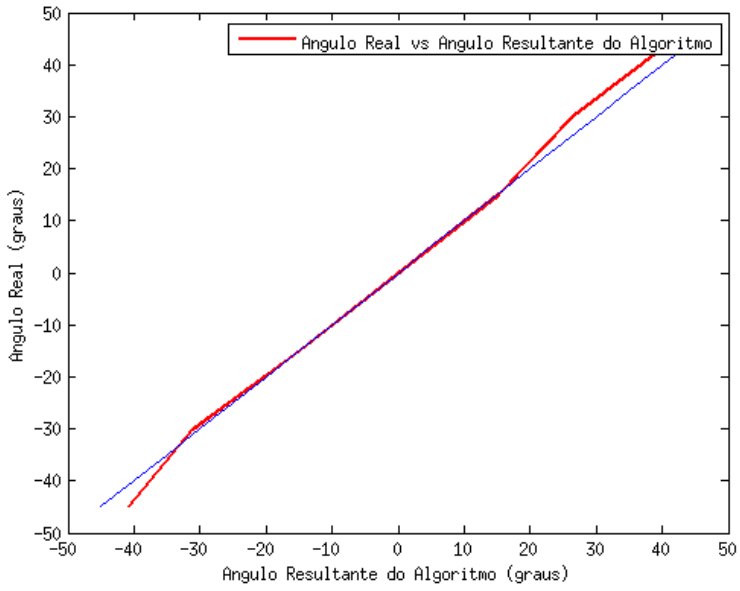


Figura 4.12: Gráfico que relaciona medidas reais de ângulo com as medidas obtidas pelo algoritmo.

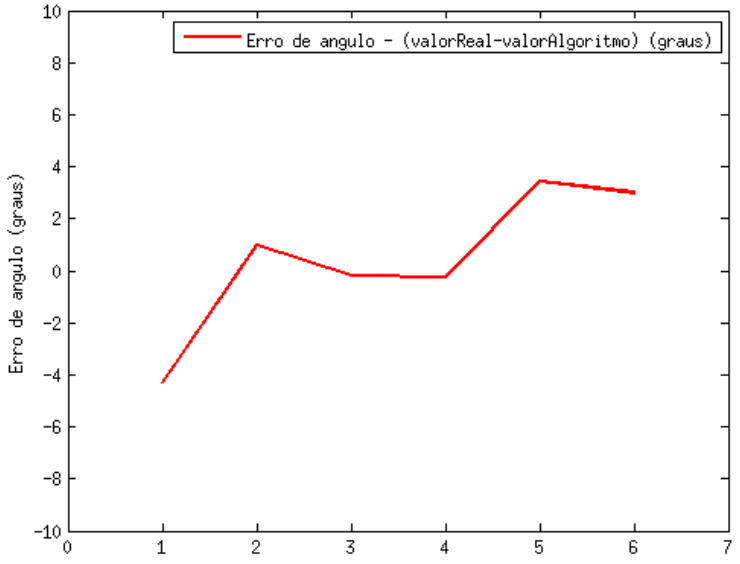


Figura 4.13: Erro de ângulo obtido entre as medidas reais e obtidas pelo algoritmo.

A vermelho na figura 4.10 está representado a reta que relaciona as medidas reais de distância com as resultantes do algoritmo em metros. A azul a reta de relação se não existisse erro. Como se pode observar existe um pequeno desvio. Este desvio, é o erro apresentado na figura 4.11 que

oscila em torno de zero. De forma análoga, é mostrado na figura 4.12 a relação entre as medidas reais angulares e retornadas pelo algoritmo. A reta em azul representa a reta de relação sem erro. O erro angular esta representado no na figura 4.13. Para uma melhor caracterização destes erros, calcularam-se os erros absoluto médio e quadrático médio de distância e angular apresentados nas tabelas 4.8 e 4.9 respetivamente.

Tabela 4.8: Erros de distância associados ao sistema de *tracking* desenvolvido.

Erro absoluto médio de distância (m)	**Erro quadrático médio de distância**(m^2)
0.0169	0.000349

Tabela 4.9: Erros de ângulo associados ao sistema de *tracking* desenvolvido.

Erro absoluto médio angular ($graus$)	**Erro quadrático médio angular** ($graus^2$)
2.0267	6.7172

4.4 Inquérito Universidade Júnior

As respostas ao inquérito realizado são apresentadas na forma de histograma. O número de alunos aos quais se realizou este inquérito foi 17. O inquérito realizado encontra-se no anexo A.

Pergunta 1 - Assinala com um 'x' de 1 a 5 o nível de interesse de cada demonstração/robô apresentado.

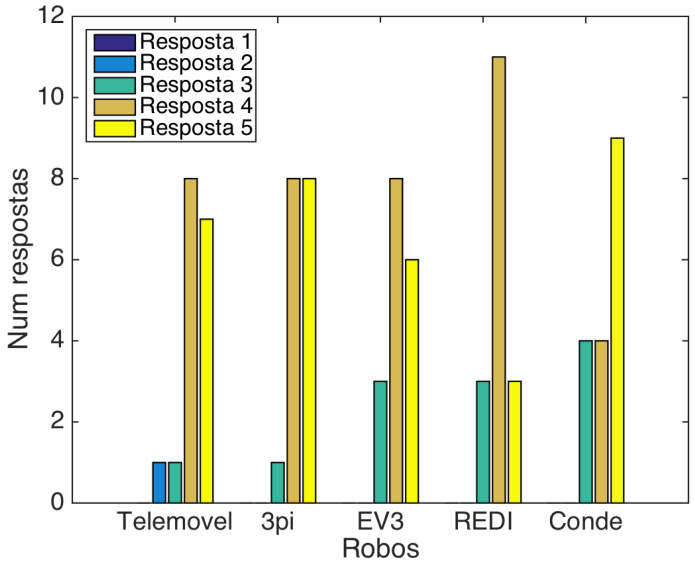


Figura 4.14: Inquérito - Histograma da distribuição de respostas relativo à questão 1 do inquérito.

Pergunta 2 - Dos cinco projetos mostrados, qual é a demonstração/robô que achaste mais interessante? E porquê?

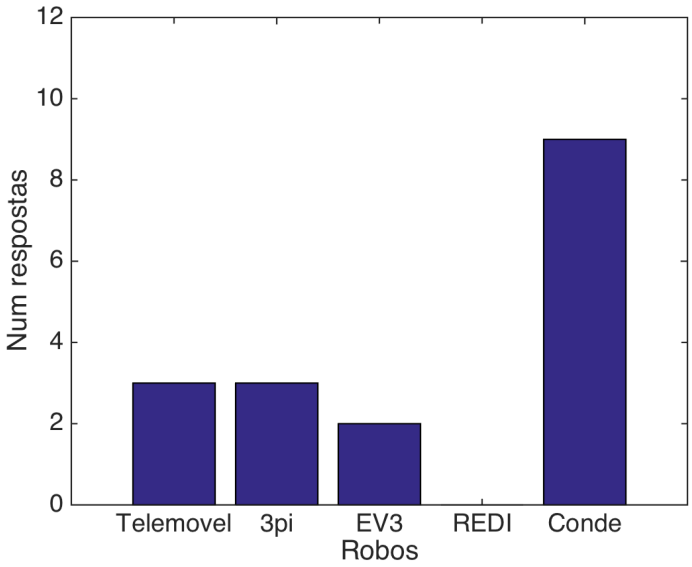


Figura 4.15: Inquérito - Histograma da distribuição de respostas relativo à questão 2 do inquérito.

Pergunta 3 - Qual é o robô mais complicado?

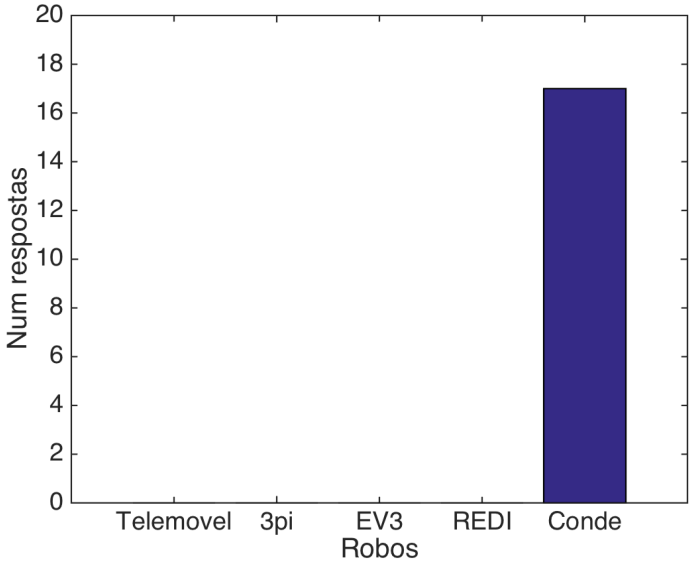


Figura 4.16: Inquérito - Histograma da distribuição de respostas relativo à questão 3 do inquérito.

Pergunta 4 - Qual é o robô mais acessível?

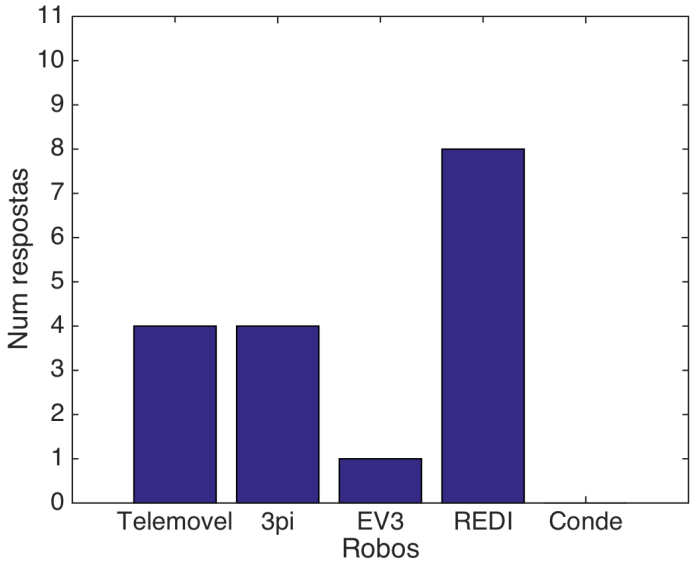


Figura 4.17: Inquérito - Histograma da distribuição de respostas relativo à questão 4 do inquérito.

Pergunta 5 - Qual é o robô que eu gostava de ter em casa para programar? Porquê?

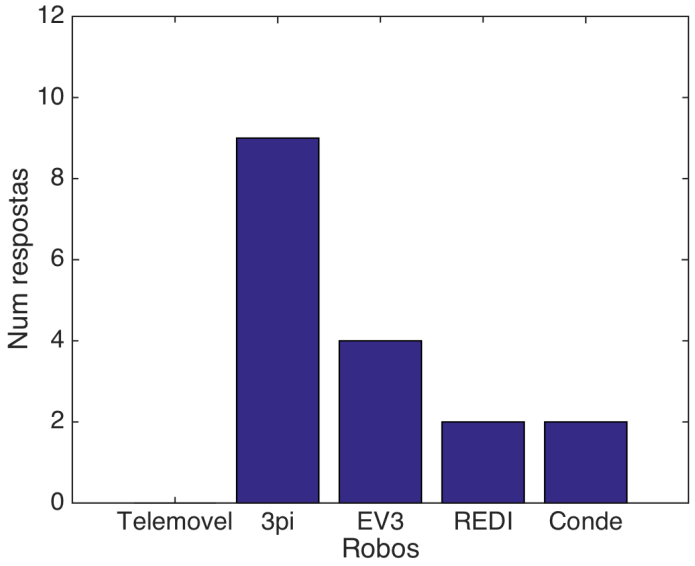


Figura 4.18: Inquérito - Histograma da distribuição de respostas relativo à questão 5 do inquérito.

4.5 Considerações Finais

Neste capítulo foram apresentados os resultados relativos às questões relevantes dos casos de estudo propostos. Em primeiro lugar, foram apresentados os resultados das duas abordagens do controlo de três robôs. Tendo em conta os testes efetuados, o robô que apresenta a melhor performance é o 3pi, seguido do EV3 e por último o REDI. Foram apresentados resultados que permitem concluir que a abordagem de programação transparente pode ser aplicada no contexto desta dissertação. Quanto ao robô de condução autónoma, foram apresentados os resultados relativos à complexidade temporal e às otimizações feitas para melhorar o algoritmo de *tracking*. Para a caracterização do sistema de desenvolvido para o Conde foram calculados os erros de distância e ângulo bem como o *lag* das câmaras que este sistema usa. Adicionalmente, foi também calculado o *lag* do sistema de *tracking* de robôs utilizado.

Capítulo 5

Conclusões

5.1 Satisfação dos Objetivos

Este trabalho procurou analisar em detalhe diversos casos de estudo em robótica utilizada em educação.

As demonstrações de robótica são visualmente apelativas e têm ganho importância aumentada pelo afastamento que se tem vindo a verificar recentemente entre os jovens e as áreas que os Anglo-saxónicos designam por STEM (Ciência Tecnologia Engenharia e Matemática).

A robótica pode ser utilizada como ferramenta para ensino pois é apelativa e vistosa. Porém, é também uma área de integração cheia de detalhes por vezes complexos que os mais jovens podem causar medo, considerar desmotivadores ou pelo menos causar distância acrescida - o que é o contrário do que pretende, aproximar os jovens da área STEM.

Pretende-se então desenhar demonstrações educativas e visualmente apelativas sem obrigar o público alvo a dominar todas as complexidades inerentes ao mundo da robótica real. Esta função deve também ser desempenhada pela FEUP, quer dentro quer fora das suas instalações. Este documento analisa e projeta um conjunto de casos de estudo que listam uma grande variedade de públicos alvo e demonstrações adequadas. São então analisadas caracterizadas e melhorados alguns sistemas demonstradores utilizados na FEUP, tabela 5.1.

Tabela 5.1: Caracterização e comparação dos casos de estudo propostos.

Nome Demo	**Alvo**	**Intenção**	**Controlo**	**Programação**	**Valor Acrescentado**	**Orçamento em Portugal €**
FEUP / REDI	• Escolas Secundárias • Mostra UP (Público) • Universidade Junior • Demo Rápida / Introdutória • Sem Estrutura	• Educação simples • Demonstraçã o simples • Lógica simples	• Linear • Histerese	• Lógica Cablada (fios)	• *Tracking*	• 800€ incluindo baterias LiPo e fios para ligações
LEGO / Ev3	• Escolas Secundárias • Casa Avançado • Construção Mecânica LEGO Programação na ES • Trabalho prolongado	• Educação mecânica • Programação por blocos	• Linear • Histerese	• C • FEUPAutom • Software LEGO	• *Tracking* • Biblioteca C + • Programação FEUPAutom • Programação Maq. Estado via software LEGO • Programação Transparente	• Só bloco 200€ • kit Educacional 500€
Pololu / 3Pi (+ FEUP)	• Visitantes à FEUP • Universidade Junior • Trabalho prolongado	• Robô simples • Fácil de programar em linguagem imperativa	• Linear • Histerese	• Arduino • FEUPAutom	• *Tracking* • Programação FEUPAutom • Arduino (Win + Linux) • Programação Transparente	• Robô inicial 100€ • “Capa” FEUP incl comunicações 100€
FEUP / Arduino/ Robô Tele	• Escolas Secundárias • Universidade Junior • Construção elétrica • Construçao mecanica • Demonstração Rápida • Trabalho Prolongado em diversas plataformas	• Tirar proveito ubiquidade do telemóvel • Partilha de plataforma móvel	• Tele → quadrático • Arduino → todo o tipo de controlo aplicável	Diversos tipos de Programação: • Tele → configurável • Arduino → todo o tipo de prog aplicável (blocos, imperativo,...)	• Demonstração global • Partilha de Plataforma • Varios tipos de programação em diversas plataformas	• 90 € + telemóvel Android
FEUP / Condução Autónoma “Conde”	• Demonstração: • Público técnico • Exigente • Estudantes Ensino Superior	• Promover a robótica nos estudantes Ensino Superior	• Linear	• C++ / ROS	• Arquitetura ROS controlo + visão RT (linha + semáforo)	• 1000€ incluindo baterias LiPo

Os inquéritos realizados mostram que todos os casos de estudo abordados são visualmente apelativos e interessantes.

O caso de estudo REDI é uma boa abordagem inicial à robótica e permite transporte fácil pois funciona em qualquer ambiente e sem necessidade de suporte externo.

O caso de estudo LEGO®EV3 é uma boa abordagem para estudos que incluam a construção mecânica. É programável através de uma variedade de soluções o que o torna bastante versátil para ter numa escola secundária com estudantes com interesses diversificados. É possível programar este robô sem fios e foi ainda desenhado um processo de controlo à distância que dispensa a programação do robô a que se chamou *programação transparente*. O trabalho anteriormente realizado e publicado demonstra ainda o interesse da abordagem Máquinas de Estado para a programação em robôs, mesmo utilizando o software original da LEGO®.

O caso de estudo 3Pi, com as alterações introduzidas na FEUP é aquele que os estudantes do secundário presentes na Universidade Júnior mais pretendem ter em suas casas para desenvolvimento pessoal, pode ser programado por intermédio de diversas ferramentas incluindo FEU-

PAutom Grafcet e com a ideia de programação transparente fica até disponível a visualização de variáveis durante o funcionamento do robô.

A programação transparente aplicada aos 2 robôs mencionados causa, claro, algum atraso de controlo difícil de quantificar devido à presença de comunicações sem fios bidirecionais. No entanto, foram feitos testes que validam que a degradação de desempenho para as condições testadas é aceitável e é possível obter um funcionamento que faça uma demonstração apelativa mesmo com programação transparente.

Outro desenvolvimento foi o robô telemóvel que pode vir a ser utilizado dentro e fora da FEUP para demonstrações e ou trabalho nas escolas secundárias. De facto, este protótipo pode ser programado por exemplo via FEUPAutom Grafcet produzindo código para o micro controlador Arduino. Para além desta utilização, é ainda possível configurar a aplicação do telemóvel para fazer o controlo propriamente dito do robô. Este tipo de plataformas permitirá que diversos estudantes do secundário (p. ex.) partilhem a plataforma móvel e que cada um faça a programação no seu telemóvel (que são muito frequentes nesta faixa etária).

A demonstração do robô "Conde"que participou no Festival Nacional de Robótica 2015 foi ainda considerada a mais complicada e interessante no inquérito realizado aos estudantes da Universidade Júnior (idade 16 a 17 anos). A nível deste projeto foram ainda abordadas uma série de questões técnico científicas a nível arquitetural associadas a portar todas as funcionalidades para o meta sistema operativo ROS. Para além disso, foi ainda apresentado o desenvolvimento de um sistema de visão em tempo real para aplicação neste robô de competição, incluindo a estimação dos tempos de atraso na aquisição da imagem.

5.2 Contributos para a Comunidade

- No decorrer do projeto de cooperação entre FEUP-ESAS foi elaborado um artigo já submetido e aceite na conferência indexada EDULEARN15, anexo A.

- Foi submetido outro artigo que está à espera de aprovação sobre a aprendizagem de ROS usando tutoriais de nível universitário.

- Programar 3pi usando FEUPAutom;

- Programar EV3 usando FEUPAutom;

- Programação Transparente no 3pi;

- Programação Transparente no EV3;

- Criação do robô telemóvel.

5.3 Trabalho Futuro

Tendo em conta os casos de estudo apresentados, existem algumas adições que podem ser implementadas.

Para o caso de estudo robô 3pi, pode ser adicionada a funcionalidade de programação sem fios do micro controlador (programação do micro controlador dentro do robô por intermédio de um sistema de programação sem fios). Com esta medida, é retirada a dependência do programador, ficando o sistema mais barato, tirando proveito do módulo de rápio para a programação sem fios.

O robô telemóvel pode ser também melhorado nomeadamente a sua estrutura. Pode ser desenhado um chassis para impressão 3D ao estilo do apresentado na figura 5.1.

A aplicação robô telemóvel poderá ainda ser brevemente melhorada para permitir programação aberta *run time* (*scripting*), permitindo assim que o conjunto se torne ainda mais flexível na sua utilização, mesmo sem necessitar de uma infra estrutura pouco frequente. A forma como foi feita a programação do telemóvel teve a preocupação de ser facilmente portável para outros dispositivos e Sistemas Operativos móveis.

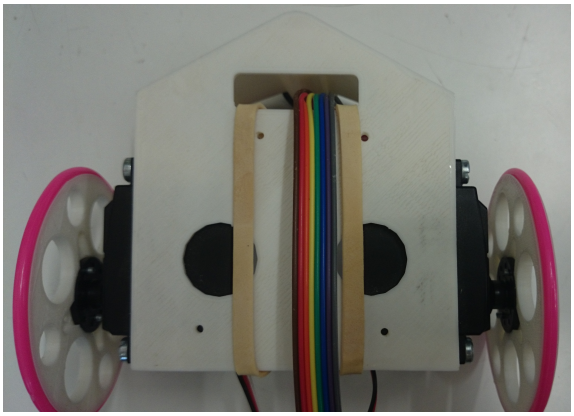


Figura 5.1: Impressão 3D - exemplo.

Quanto ao robô de condução autónoma existem algumas sugestões que devem ser registadas:

- integração de todos os sistemas em ROS;
- o uso de 2 câmaras com ou sem lentes que permita ver a qualquer momento a pista toda seja qual for a orientação do robô. Estas câmaras podem ser duas kinects;
- controlo da trajetória com mapa em ROS, ou sem recurso a mapa.

Referências

[1] E. Rublee and G. Bradski, "ORB: an efficient alternative to SIFT or SURF." [Online]. Available: http://www.willowgarage.com/sites/default/files/orb_final.pdf

[2] a.M. Muad, a. Hussain, S. Samad, M. Mustaffa, and B. Majlis, "Implementation of inverse perspective mapping algorithm for the development of an automatic lane tracking system," *2004 IEEE Region 10 Conference TENCON 2004.*, vol. A, pp. 207–210, 2004.

[3] M. Oliveira, V. Santos, and A. D. Sappa, "Multimodal inverse perspective mapping," *Information Fusion*, vol. 24, no. September, pp. 108–121, 2015. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1566253514001031

[4] "Lens Calibration." [Online]. Available: http://bersoft.com/bimagem/help/lens_calibration.htm

[5] B. Frank, "Robotics 2 Camera Calibration," *Camera*.

[6] E. Fernandes, "Autonomous Driving Robotic System for Demonstrative Purposes," 2013.

[7] M. Nixon and A. Aguado, *Feature extraction & image processing*, 2008. [Online]. Available: http://books.google.com/books?hl=en&lr=&id=97QebyNxyaYC&oi=fnd&pg=PP2&dq=Feature+Extraction+&+Image+Processing&ots=wThmg3NqI4&sig=H08w-8QSMiEwQZi1-NusLFgNRUQ

[8] M. U. Bers and M. Portsmore, "Teaching partnerships: Early childhood and engineering students teaching math and science through robotics," *Journal of Science Education and Technology*, vol. 14, no. 1, pp. 59–73, 2005.

[9] "RoboCupJunior." [Online]. Available: http://www.robocup2014.org/?page_id=60

[10] "RoboParty." [Online]. Available: http://www.roboparty.org/

[11] "CEABOT." [Online]. Available: http://www.ceautomatica.es/sites/default/files/upload/10/CEABOT/index.htm

[12] "RobotChallenge: Robotchallenge." [Online]. Available: http://www.robotchallenge.org/

[13] ":: ISTROBOT 2015 ::." [Online]. Available: http://www.robotika.sk/contest/2015/index.php

[14] "Micro-Rato." [Online]. Available: http://microrato.ua.pt/

[15] "Micromouse Portuguese Contest." [Online]. Available: http://www.micromouse.utad.pt/

[16] A. D. Alimisis and C. Kynigos, "Constructionism and robotics in education," *Teacher Education on Robotics-enhanced Costructivist Pedagogical Methods*, pp. 11–26, 2009.

[17] M. Portsmore, C. Rogers, P. Lau, and E. Danahy, "Remote sensing and tele-robotics for elementary and middle school via the Internet," *Computers in Education Journal*, vol. 14, pp. 72–77, 2004.

[18] P. Bianchetti, L. Giannini, D. Mazzei, and D. Merlo, "PROJECT ROB & IDE : The Story of Robot and android," pp. 6–8, 2006.

[19] M. Resnick, "Technologies for lifelong kindergarten," *Educational Technology Research and Development*, vol. 46, no. 4, pp. 43–55, 1998.

[20] "Scratch - Imagine, Programe, Partilhe." [Online]. Available: https://scratch.mit.edu/

[21] "VISUINO." [Online]. Available: http://www.visuino.com/

[22] "Home - LEGO.com." [Online]. Available: http://www.lego.com/en-us/

[23] J. Ruiz-del Solar and R. Avilés, "Robotics courses for children as a motivation tool: The Chilean experience," *IEEE Transactions on Education*, vol. 47, no. 4, pp. 474–480, 2004.

[24] A. Druin and J. Hendler, *Robots for Kids: Exploring New Technologies for Learning*, 2000.

[25] A. Pereira, "Robotic Rules," 2015. [Online]. Available: http://robotica2015.utad.pt/sites/all/themes/max/docs/Robotica2015_AutonomousDriving.pdf

[26] "DARPA CHALLENGE 2004." [Online]. Available: http://en.wikipedia.org/wiki/DARPA_Grand_Challenge_(2004)

[27] "DARPA CHELLENGE 2005." [Online]. Available: http://en.wikipedia.org/wiki/DARPA_Grand_Challenge_(2005)

[28] "DARPA CHALLENGE 2007." [Online]. Available: http://en.wikipedia.org/wiki/DARPA_Grand_Challenge_(2007)

[29] "DARPA CHELLENGE 2012." [Online]. Available: http://en.wikipedia.org/wiki/DARPA_Robotics_Challenge

[30] "DARPA CHALLENGE 2013." [Online]. Available: http://www.gizmag.com/fang-darpa-registrations-open/24393/

[31] C. A. Using, "Method for Reading Sensors and Controlling Actuators Using Audio Interfaces of Mobile Devices," pp. 1572–1593, 2012.

[32] "l Cellbots: Using Cellphones as Robotic Control Platforms." [Online]. Available: http://www.cellbots.com/

[33] R. V. Aroca, R. B. Gomes, D. M. Tavares, A. A. S. Souza, A. M. F. Burlamaqui, G. a. P. Caurin, and L. M. G. Gonçalves, "Increasing students' interest with low-cost cellbots," *IEEE Transactions on Education*, vol. 56, no. 1, pp. 3–8, 2013.

[34] "Necessitas/Ministro - KDE Community Wiki." [Online]. Available: https://community.kde.org/Necessitas/Ministro

[35] "pt - ROS Wiki." [Online]. Available: http://wiki.ros.org/pt

[36] A. M. Mendonça, "Sistemas baseados em Visão," pp. 1–47.

[37] D. Viswanathan, "Features from Accelerated Segment Test (FAST)," *Homepages.Inf.Ed.Ac.Uk*. [Online]. Available: http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/AV1011/AV1FeaturefromAcceleratedSegmentTest.pdf

[38] S. Invariant and F. Transform, "The SIFT (Scale Invariant Feature Transform) Detector and Descriptor Review : Matt Brown ' s Canonical Frames," 2004.

[39] L. Juan, "A comparison of SIFT , PCA-SIFT and SURF This paper compares three robust feature detection methods , they are , Scale Invariant Feature Transform (SIFT), Principal Component Analysis (PCA) -SIFT and Speeded Up Robust Features (SURF). Lowe presented," *Image (Rochester, N.Y.)*, no. 4, pp. 143–152.

[40] H. Bay, T. Tuytelaars, and L. Van Gool, "SURF: Speeded up robust features," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3951 LNCS, pp. 404–417, 2006.

[41] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief Binary Robust Independent Elementary Features778.pdf."

[42] E. Trucco and K. Plakas, "Video tracking: A concise survey," *IEEE Journal of Oceanic Engineering*, vol. 31, no. 2, pp. 520–529, 2006.

[43] G. S. Manku, P. Jain, a. Aggarwal, L. Kumar, and S. Banerjee, "Object tracking using affine structure for point correspondences," *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 704–709, 1997.

[44] L. Matthies, R. Szeliski, and T. Kanade, "Ad-a 195 818," *Notes*, 1988.

[45] L. S. Shapiro, H. Wang, and J. M. Brady, ": : r," 1992.

[46] Y.-s. Yao and R. Chellappa, "I ; [; :," pp. 654–657, 1994.

[47] A. Sousa, C. Santiago, P. Malheiros, P. Costa, and A. P. Moreira, "Using Barcodes for Robotic Landmarks."

[48] "xvision." [Online]. Available: http://www.xvision.com/

[49] M. E. Spetsakis and J. Y. Aloimonos, "Structure from motion using line correspondences," *International Journal of Computer Vision*, vol. 4, pp. 171–183, 1990.

[50] C. Taylor and D. Kriegman, "Structure and motion from line segments in multiple images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 1, 1995.

[51] S. Atiya and G. D. Hager, "Real-time vision-based robot localization," *IEEE Transactions on Robotics and Automation*, vol. 9, no. 6, pp. 785–800, 1993.

[52] Y. Liu and T. Huang, "A linear algorithm for motion estimation using straight line correspondences," *[1988 Proceedings] 9th International Conference on Pattern Recognition*, pp. 213–219, 1988.

[53] A. J. Davison, I. D. Reid, N. D. Molton, and O. Stasse, "MonoSLAM: Real-time single camera SLAM," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 29, no. 6, pp. 1052–1067, 2007.

[54] S. Hinterstoisser, V. Lepetit, S. Ilic, P. Fua, and N. Navab, "Dominant orientation templates for real-time detection of texture-less objects," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 2257–2264, 2010.

[55] A. J. Davison, "Active Search for Real-Time Vision," 2005.

[56] D. W. Murray, K. J. Bradshaw, P. F. McLauchlan, I. D. Reid, and P. M. Sharkey, "Driving saccade to pursuit using image motion," *International Journal of Computer Vision*, vol. 16, pp. 205–228, 1995.

[57] M. J. Swain and M. a. Stricker, "Promising directions in active vision," *International Journal of Computer Vision*, vol. 11, pp. 109–126, 1993.

[58] M. Isard and A. Blake, "Contour tracking by stochastic propagation of conditional density," *Engineering*, vol. 1, pp. 343–356, 1996. [Online]. Available: http://www.springerlink.com/index/y831557ht0601456.pdf

[59] "Pololu 3pi Robot." [Online]. Available: https://www.pololu.com/product/975

[60] "FEUPAutom - Armando Jorge Sousa." [Online]. Available: http://paginas.fe.up.pt/~asousa/wiki/doku.php?id=proj:feupautom

[61] M. Quigley, K. Conley, B. Gerkey, J. FAust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Mg, "ROS: an open-source Robot Operating System," *Icra*, vol. 3, no. Figure 1, p. 5, 2009. [Online]. Available: http://pub1.willowgarage.com/~konolige/cs225B/docs/quigley-icra2009-ros.pdf

[62] B. Oliveira, "Plataforma Robótica Multi-Funcional," 2008.

[63] “ev3dev.” [Online]. Available: http://www.ev3dev.org/

[64] “EV3 Releases.” [Online]. Available: https://github.com/ev3dev/ev3dev/releases

[65] “Arduino - Software.” [Online]. Available: http://www.arduino.cc/en/Main/Software

[66] “Pololu - 3pi.” [Online]. Available: https://www.pololu.com/docs/0J17/3

Anexo A

Anexos

A.1 Características dos PCs utilizados para testes

Tabela A.1: Características dos PCs usados para os testes de performance do algoritmo de *tracking*.

Nome	**Processador**	**Memória**
Raspberry Pi B 512MB	ARM1176JZF-S 700 MHz	512MB
Raspberry Pi 2 B	900MHz quad-core ARM Cortex-A7 CPU	1GB
Asus EeePC 1005HA	1.66GHz Intel Atom N280	2GB
Asus ROG GL550JK	Intel Core i7-4700HQ 2.5GHz	16GB

A.2 Artigo - ROBOTICS: A teaching tool for STEM education in high school

Nesta secção é apresentado o artigo produzido no decorrer do projeto conjunto FEUP-ESAS.

ROBOTICS: A TEACHING TOOL FOR STEM EDUCATION IN HIGH SCHOOL

Valter Costa, Armando Sousa, Tiago Cunha, Carlos Morais

FEUP - Faculty of Engineering, University of Porto, Portugal
INESC TEC - INESC Technology and Science (formerly INESC Porto) and FEUP - Faculty of Engineering, University of Porto, Portugal
FEUP - Faculty of Engineering, University of Porto, Portugal
ESAS – Escola Secundária Aurélia Sousa, Portugal

Abstract

This article describes an experience in university and high school cooperation. It is expected to foster knowledge and deep learning in secondary schools by turning extra-curricular activities into articulated subject. The robotics area is very useful and generates interest and enthusiasm, even more so when associated with competition.

The experience used Lego Ev3 robot and the students learned to program with a healthy technical approach called state machine programming and the easy to use Lego Software programming tool. The participation is enthusiastic because of the participation in the national robotics festival that leads into international RoboCup Federation robotics competitions.

The article proposes a set of sessions adequate for secondary school students that constitute the initial step to find a curriculum for robotics in order to simultaneously learn robotics and foster interconnections with the curricular courses in STEM areas, even extending into structured programming issues.

The test involved two participations in the national robotics competition that interestingly involved a team of 3 girls and another team of 3 boys although more students were involved during the year that the experience lasted.

Declarations from the involved stakeholders are mentioned, even allowing for a brief discussion for women in STEM areas and technology distance for young (wo)men. Some hints, issues and lessons learned are shown.

The advocacy of such informal learning strategy is made, advantages and limitations discussed.

Keywords-Edutainment; LEGO®; State-Machines; STEM ;

1. INTRODUCTION

This paper reports the findings of the joint education initiative between the Faculty of Engineering University of Porto (FEUP) and a local high school (ESAS, Escola Secundária Aurélia de Sousa, Portugal). This initiative consisted of teaching high-school students using Robotics as a tool in order to foster their interest in sciences and technology. Robotics is a highly motivating activity for children of all ages since it allows them to approach technology intuitively, while using and mastering underlying science principles. A range of activities were developed through practical robotics lessons using the LEGO® Mindstorms® EV3 kit and culminated with the participation of the six students in the national robotics competition.

It is also discussed in this text why should robotics be used to spark long-term interest in children before they choose their career.

We first start discussing related work and experiments performed, in section II. In section III a project contextualization is supplied. From section IV onwards, the project development is covered in detail. The last chapter presents the conclusions the authors collected from this education initiative, and expose some of the statements aforementioned by students involved.

2. RELATED WORK

Teaching Robotics to young high school students is very beneficial for them as a very good complementary feature to their studies in STEM areas. Recently, the number of students that select further education in those areas are decreasing [1]. Some initiatives like RoboCup Junior contest [2], RoboParty [3], CEABOT [4], RobotChallenge [5], IstRobot [6], “Micro-Rato” [7], “Micro-Mouse” [8] focus on improving upon this current trend.

Robotics can be used as teaching and learning tools in different contexts [9]. The first initiative includes educational activities aimed at creating a learning environment suitable for learners with Robotic related subjects. The latter,

proposes that Robotics can be a teaching and learning tool for other STEM areas. Since Robotics is a highly-disciplinary field lots of initiatives have been presented to the community. One for example is The Science, Engineering, NASA Site of Remote Sensing (SENSORS) project allowed high school and middle school students to control LEGO® RCX (pretending to be rovers) over the web, by uploading their own programs [10].

The network Robot@Scuola [11] is a virtual community for the knowledge of Robotic Science, coordinated by the School of robotics of Genoa, Italy. The community includes different schools ranging from kindergarten to high school. They share all their work and activities online. The Lifelong Kindergarten group (LLK) [12] is a group within the MIT Media Lab that focus on developing new technologies for children that manages to engage them into creative activities. They are responsible for project SCRATCH [13], a freeware software for basic visual programming. In this paper we're interested in activities that use LEGO® NXT or EV3 as a supporting teaching tool.

From a very early age, children fashion movement and moving things. The introduction of modern computers allowed them to build their own worlds and play with them (e.g. video games). But only in 1998, with the appearance of the first embedded computation system designed for children [14] were they able to create objects that can perform actions and move. Over the last fifteen years, LEGO® has invested a lot in education and in schools, spanning their kits worldwide. A lot of high schools like [15] already offer introductory courses in Robotics trying to spark interest in the STEM areas. High school students due to their advance knowledge over other grade school colleagues can avail the experience better than children, because parallel to these activities they are learning about math and physics. These two activities are complementary to each other, and as proposed by [16] it is beneficial for long term STEM skills development in both genders equally.

3. CONTEXT

In order to create a course in a contextual engineering environment it must provide two elementary propositions [17]:

1. "Build cumulative STEM competencies in students by building on the foundation of knowledge established at each level in education ..."
2. "Provide students with hands-on, open-ended, real-world problem solving experiences that are linked to the curriculum, using science, engineering, and technology modules, and grouping such experiences and modules by discipline and level of difficulty"

There are five strategies for teaching STEM expertise such as *Relating*, *Experiencing*, *Applying*, *Cooperating*, and *Transferring* [18]. The first two strategies are the most important because if the students don't relate to the learning experience then they won't benefit much from the activity. *Applying, Cooperating* and *Transferring* are strategies united by STEM learning and are applied to engineering professions.

Engineering is the profession in which knowledge of the mathematical and natural sciences are applied to develop solutions for benefitting mankind. In order to stimulate their interest and to make them relate to the thematic, because nowadays, it is common to see robots play a role in either video games or live films. By allowing students to design and program their own robots, they will get involved and will get to practice several other disciplines and will get to practice other soft-skills like team working and management.

It was with all the above in mind that Aurélia de Sousa, a local high school, teamed up with FEUP to create a joint teaching initiative for several high-school students. The course developed focused on teaching several key robotics concepts and applying them in the LEGO® Mindstorms® EV3 kit with the finality of competing in the national robotics competition (Search and Rescue B), leading to international participation in worldwide competition in the RoboCup meetings [2].

The school enlisted two teams Fig.1 of three students for the competition, with three male students in one team, and three female students in another. Other students were involved but these were the only teams enlisted. The next section pinpoints several of the project details.

Figure 1. The teams.

4. PROJECT DEVELOPMENT

4.1.1 Project Tools

Several tools were used in the project. As the participants weren't familiarized with any programming language nor with basic electronics, they were supplied with the EV3 and NXT kits and with the correspondent software.

- LEGO® Mindstorms® Kit + Extra Parts – Each EV3/NXT kit comes with programming brick, engines, sensors, and some building parts. In order to allow the involved parties some extra design room, an extra box of parts was acquired, Fig 2.
- LEGO® Mindstorms® Software – As it was the first time the students ever got in contact with programming languages, it was decided that a visual programming language would be the best suited for them. The target robot was a LEGO® Mindstorms® EV3, so the provided software was the safest choice to facilitate hands-on-learning, Fig. 3.
- School Website – A website was provisioned to the community documenting the entire development phases. The address can be found at [19].
- Testing Track – A testing track similar to the tournament was built in order to allow the students to test their prototypes in it. The track was intended to be as similar as the one in the 2014 rules of the robotics national competition.

Figure 2. LEGO® Mindstorms® kit.

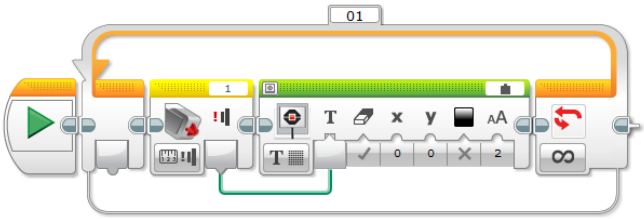


Figure 3. Design interface LEGO® software.

Figure 4. Arena constructed for the tests.

4.1.2 Budget

The following table presents a close estimative of the project costs. The reason LEGO® Mindstorms® was chosen in comparison to other alternatives is because high school students didn't have enough mechanical or electrical expertise and there wasn't enough time to teach them.

Table 1. Budget.

Items	Items Cost
	Euros
Human Resources (6 months Scholarship)	2310
Consumables	500
Dislocations	500
External Services	520
Equipments (LEGO® Mindstorms® EV3, LEGO® Expansion set, LEGO® software)	771
Registration	300
Teacher, Professor, Spaces	-
Total Cost	4901

4.1.3 Project Planning

The sessions presented by the authors to the school students were:

1. LEGO® basics: Introduction to software, EV3 and NXT;
2. Detect and avoid obstacles: basics;
3. Detect and avoid obstacles: part II;
4. Follow line: basics;
5. State-Machines basics;
6. State-Machines implementation;
7. Gradual acceleration and gradual deceleration using the state-machine approach;
8. Follow line and follow wall using the state-machine approach;
9. Concepts review;
10. Follow non-rectilinear paths using a floor marker for localization;
11. Explore remaining EV3 sensors;
12. Module integration with LEGO® software: basics;
13. Module integration with LEGO® software: part II;
14. Final concept Review;
15. Introduction to robotic competitions;
16. LEGO® construction: basics;
17. LEGO® construction: claws and other grappling methods;
18. Final prototype construction;
19. Follow line adjustments to the new prototype;
20. Slope conquer: basics;
21. Slope conquer: part II;
22. Claw integration and programming;
23. Final modules integration;
24. Competition preparation;
25. Final adjustments.

Teaching Phase: The first fourteen sessions were focused on teaching and accustom the students to the type of problems Robotics brings.*The first four introduced the subject and the LEGO® Mindstorms® EV3/NXT and how to program them in the software provided and made them develop algorithms for avoiding an obstacle or stopping before*

hitting and in session four a simple straight line follower. Starting from that, the liner programming style introduced wasn't enough to solve complex questions and step sequences. It was very difficult to debug and to proceed. It was at this stage that the addition of state-machines proved useful. State-Machines or Finite-State-Machine (FSM) can be described by [20]:

1) *A description of the stimuli that the machine will take into account and a description of the responses that the machine can generate;*
2) *States which are drawn as circles and there is one that must be considered as the initial stage;*
3) *The computational behavior of the machine which is described in terms of transitions (arrows) leading one state to another.*

An example of a FSM can be seen on Fig. 5. In it, two states are present (e.g. turning on and off the lights with two different switches). As an initial state is required, the diagram immediately progresses to state zero. Then, if T1 activates (e.g. hitting the on switch) the new state becomes state one and remains there until the T2 a stimulus happens.

Using this approach, programming with the LEGO® software becomes quite easy. Every state is in a different tab and as you can separate the left ring for transitions and the right one for actions, it becomes quite appropriate retaining its visual appeal and allowing for more complex interactions and programs which are required for the competition.

Finally, an interesting ability of the software was the ability to separate chunks of code into visual blocks, like function calls. This taught the students the concept of modularity, and helped them further when they were testing the competition program. One upsetting design fault of the software was still on the debug side. Even though, the State-Machines helped Fig. 5, a real debugger was lacking, and debug had to be made with the help of sounds.

For the final sessions of the teaching phase, a concepts review was made, and all the previous programs made were turned into visual blocks (functions) and saved for posterior use.

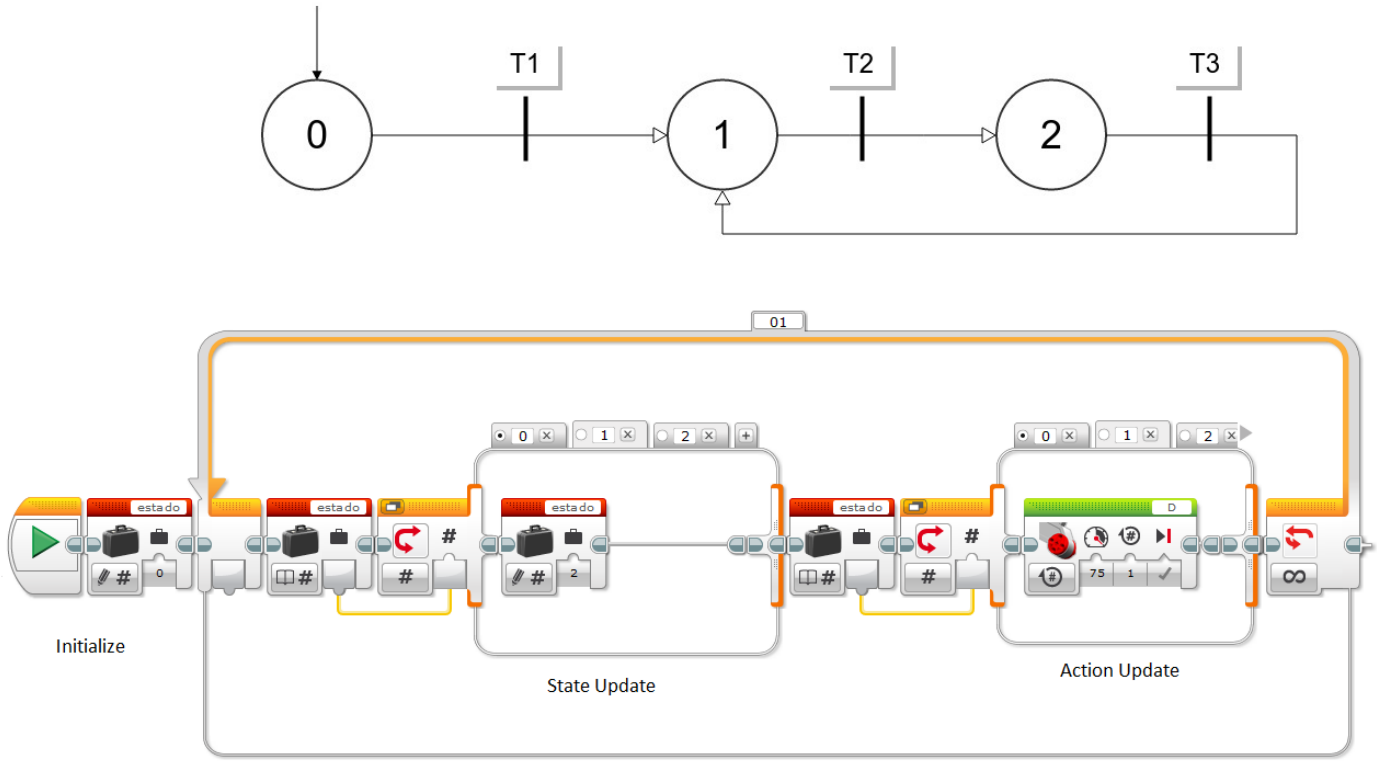


Figure 5. Example of a state-machine and its implementation on the LEGO® software.

Development Phase: Upon the start of the phase (session fifteen), the students were introduced to the search and rescue competition, with a mentionto the rules, state-of-the-art, and what to expect from their adversaries. This was followed by a discussion of what should the prototype have and what functions would it need to do. The students compiled the following list:

- Follow Line;
- Avoid Obstacle;
- Contour Obstacle;
- Detect Crossroads;
- Climb Slope;
- Search Victim;
- Descend Slope;
- Pick up Victim;
- Descend Slope;
- Put Victim in specific place.

Many of the functions had already been done in previous exercises, so the group focused on the construction on the prototype and the gripper. The faculty monitor helped with some ideas for the claw, but the entire design and building was left to the students to make. In the school website [19], some ideas for LEGO® building were gathered and made available for the students. Both teams built a differential drive robot, equipped with an ultrasound sensor, a color sensor and two other line sensors. This construction allows the robot to follow lines, detect crossroads, and pick up cans (victims). As only one motor slot was left in the core brick of the EV3/NXT, the claw could only use one motor and so had to be adapted to that restriction.

The most important constituents of the gripper are the cogwheels, one worm screw to connect the cogwheels, and one motor. With a design like Fig. 6 the restriction imposed was met.

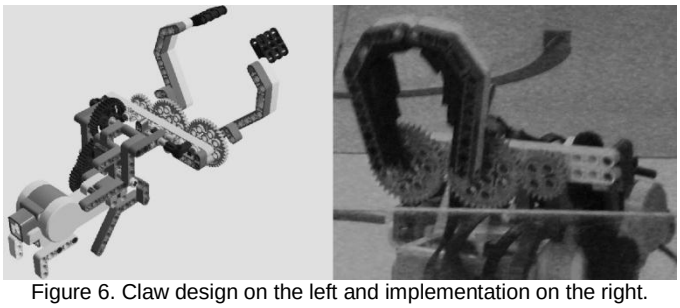


Figure 6. Claw design on the left and implementation on the right.

The remainder of the functions, even though they had been prepared before, still required adjustments for the prototype. The slope climbing proved to be complicated, because the robot was either too large, which made it hit the walls, or too heavy, which made it fall on his back midway through the slope. This adversity required one more session to solve than what was previously expected. The final sessions were spent turning the line followers and the claw for the victim picking.

4.1.4 Competition

The national robotics festival (FNR, Festival Nacional de Robótica) is a competition organized by the Portuguese robotics association (SPR, Sociedade Portuguesa de Robótica) since 2001. Every year, investigators, businessman, and other people connected to the areas of Robotics participate in diverse competitions. The high school students involved participated in the Search and Rescue B competition [21]. In this event, mobile robots are used to identify victims in catastrophe scenarios. The complexity of these scenarios increases from moving between walls (Rescue B) and through paths with obstacles, line interruptions and slopes until the robots reach an area where the victims can be placed randomly in an open field.

The Robótica2014 competition was held in Espinho, Aveiro, Portugal in the Nave Polivalente de Espinho. This competition has two stages. The first is an interview to assess that the students were in fact the ones that produced the prototype. General questions like physical architecture, design, how to solve specific competition problems (like conquering the slope) are asked. Later, the teams compete in the real course. It is to note that the track lines on the floor change with every round so that the students cannot hardcode the robot's movement.

5. CONCLUSIONS

The main purpose of the presented project was to (i) attract high school students to STEM areas (including programming); (ii) use robotics to visually illustrate the concepts at stake and to promote robotics area (iii) take advantage of the natural competitiveness of youngsters to promote technical knowledge and attract the best and the most motivated students into FEUP and (iv) to promote skills such as team work and communication.

Another of this project's aims was the development of a curriculum that is different than that of their peers with Learn-By-Doing as is common in many STEM areas.

We believe that all the supra-mentioned points were achieved with a high satisfaction level from those involved. Some testimonies were gathered to compare the final result with the initial expectations.

Even though results weren't as good as those expected in the competition, several key lessons were learned. For example, students reported to have difficulties in managing the Robotics' course with the demands of their main curriculum. Attempts of ameliorating this problem could pass through optimization of the project scheduling, so that its most challenging stages would coincide with low activity periods of the main curriculum, whenever possible. A second type of problems felt was of contextual and social nature. Despite being intrinsically motivated with the project, several students felt that the time and effort they invested on it was not appreciated or even understood by the people close to them, especially when their parents were professionals in non-STEM areas.

Events like the participation in the robotics competition could help improving recognition from the student's immediate social network, that is, the importance and the merits of the participation were recognized; a student shared that her father became much more supportive after accompanying her in the final competition and after seeing the work developed first-hand – before seeing the final product (the robot) family and friends became much more enthusiastic.

Overall, the group of students involved felt the project made them go out of their comfort zone and measured well against previous extracurricular experiences the students had. It greatly contributed to the improvement of their ability to tackle practical problems, in contrast with the mostly theoretical official curriculum the students were enrolled in. The classes stimulated teamwork, communication skills the autonomy, besides providing an additional mental framework for problem solving. The concept of state-machines was particularity popular among the students, which they were able to transfer to other subjects and share with their colleagues to tackle diverse questions and will be important if they engage in any kind of computer programming in future challenges. Ultimately, the project allowed the students to be more aware of the surrounding technology and non-intuitive challenges associated with it, such as ordering a robot to do a square.

Robotics, as a highly multi-disciplinary field, requires learners to use knowledge from several different areas, and allows for an easy connection with other areas of education. A course like the one presented in this paper, inexpensive to implement and allowing hands-on-learning, can help develop the student's ability to work on practical problems and develop other team work skills.

ACKNOWLEDGMENTS

We thank all the parties involved mainly, the students (Carolina Lobato,Catarina Lourenço, Carlos Vieira, Henrique Vasconcelos, Miguel, Rita Magalhães) and the homeroom teach (Carlos Morais). From the faculty we'd like to thank one of the authors Armando Sousa for the project management. Lastly, we'd like to thank our colleague Denise Gameiro for the qualitatively analysis of the testimonies gathered. This work is partially financed by ERDF – European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT – Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within project «FCOMP-01-0124-FEDER-037281».

REFERENCES

[1] M. U. Bers and M. Portsmore, "Teaching partnerships: Early childhood and engineering students teaching math and science through robotics," *J. Sci. Educ. Technol.*, vol. 14, no. 1, pp. 59–73, 2005.

[2] "RoboCupJunior." [Online]. Available: http://www.robocup2014.org/?page_id=60. [Accessed: 26-Feb-2015].

[3] "RoboParty." [Online]. Available: http://www.roboparty.org/. [Accessed: 26-Feb-2015].

[4] "CEABOT." [Online]. Available: http://www.ceautomatica.es/sites/default/files/upload/10/CEABOT/index.htm. [Accessed: 26-Feb-2015].

[5] “RobotChallenge: Robotchallenge.” [Online]. Available: http://www.robotchallenge.org/. [Accessed: 26-Feb-2015].

[6] “:: ISTROBOT 2015 ::” [Online]. Available: http://www.robotika.sk/contest/2015/index.php. [Accessed: 26-Feb-2015].

[7] “Micro-Rato.” [Online]. Available: http://microrato.ua.pt/. [Accessed: 10-Feb-2015].

[8] “Micromouse Portuguese Contest.” [Online]. Available: http://www.micromouse.utad.pt/. [Accessed: 05-Mar-2015].

[9] A. D. Alimisis and C. Kynigos, “Constructionism and robotics in education,” *Teach. Educ. Robot. Costr. Pedagog. Methods*, pp. 11–26, 2009.

[10] M. Portsmore, C. Rogers, P. Lau, and E. Danahy, “Remote sensing and tele-robotics for elementary and middle school via the Internet,” *Comput. Educ. J.*, vol. 14, pp. 72–77, 2004.

[11] P. Bianchetti, L. Giannini, D. Mazzei, and D. Merlo, “PROJECT ROB & IDE : The Story of Robot and android,” pp. 6–8, 2006.

[12] M. Resnick, “Technologies for lifelong kindergarten,” *Educ. Technol. Res. Dev.*, vol. 46, no. 4, pp. 43–55, 1998.

[13] “Scratch - Imagine, Programe, Partilhe.” [Online]. Available: https://scratch.mit.edu/. [Accessed: 26-Feb-2015].

[14] “Home - LEGO.com.” [Online]. Available: http://www.lego.com/en-us/. [Accessed: 27-Feb-2015].

[15] J. Ruiz-del-Solar and R. Avilés, “Robotics courses for children as a motivation tool: The Chilean experience,” *IEEE Trans. Educ.*, vol. 47, no. 4, pp. 474–480, 2004.

[16] A. Druin and J. Hendler, *Robots for Kids: Exploring New Technologies for Learning*. 2000.

[17] P. Reed, J. Ritz, C. Lin, S. Hsiung, W. Frazier, and R. Berry, “Stem initiatives stimulating students to improve science and mathematics achievement,” *Technol. Teach.*, vol. 1, no. January, pp. 23–29, 2005.

[18] J. D. Bransford, A. L. Brown, and R. R. Cocking, “Learning Transfer,” *How people Learn brain, mind, Exp. Sch.*, pp. 51–78, 2004.

[19] T. Cunha and V. Costa, “Robôs em Movimento,” 2013. [Online]. Available: https://sites.google.com/site/robosmovimento/home.

[20] M. Cryan, “Inf1A : Introduction to Finite State Machines,” pp. 1–6, 2004.

[21] “Junior Rescue (A e B).” [Online]. Available: http://www.robotica2014.espe.pt/index.php/en/competitions/robocup-junior-2/junior-rescue-a-e-b. [Accessed: 27-Feb-2015].

A.3 Robotics: Using a competition mindset as a tool for learning ROS

O artigo não se encontra neste anexo pelo facto de ainda não ter sido aceite na conferência. São apenas apresentados alguns resultados, nomeadamente, a análise de respostas a um inquérito realizado aos participantes no âmbito do curso de ROS.

Pergunta 1 - Já teve contacto com ROS antes?

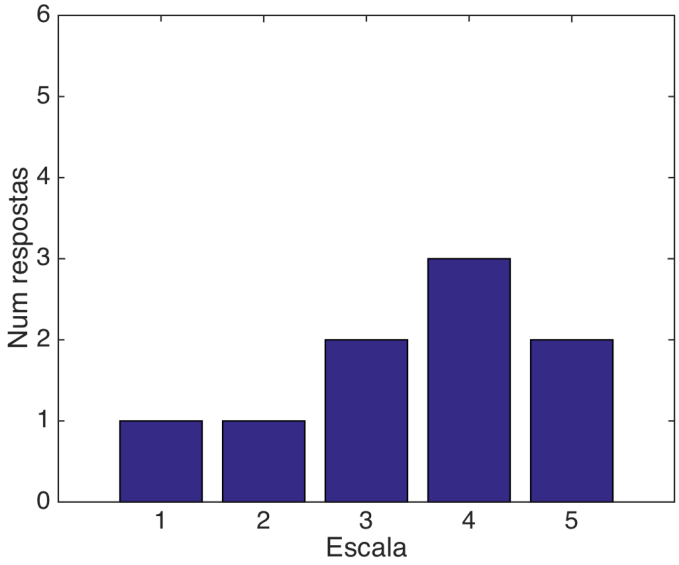


Figura A.1: Inquérito - Histograma da distribuição de respostas relativo à questão 1 do inquérito.

Pergunta 2 - A sequência de procedimentos pareceu adequada?

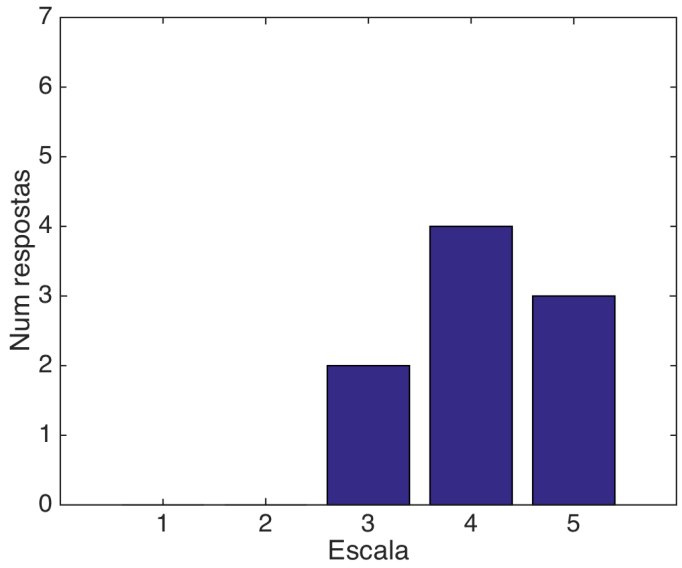


Figura A.2: Inquérito ROS - Histograma da distribuição de respostas relativo à questão 2 do inquérito.

Pergunta 3 - Seria necessário mais documentação escrita?

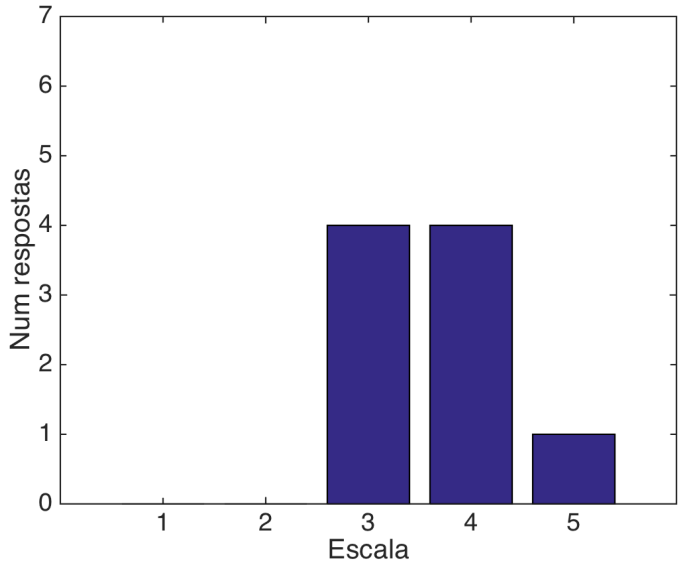


Figura A.3: Inquérito ROS - Histograma da distribuição de respostas relativo à questão 3 do inquérito.

Pergunta 4 - Concorda com a perspetiva de aprendizagem ser formulada em torno da competição?

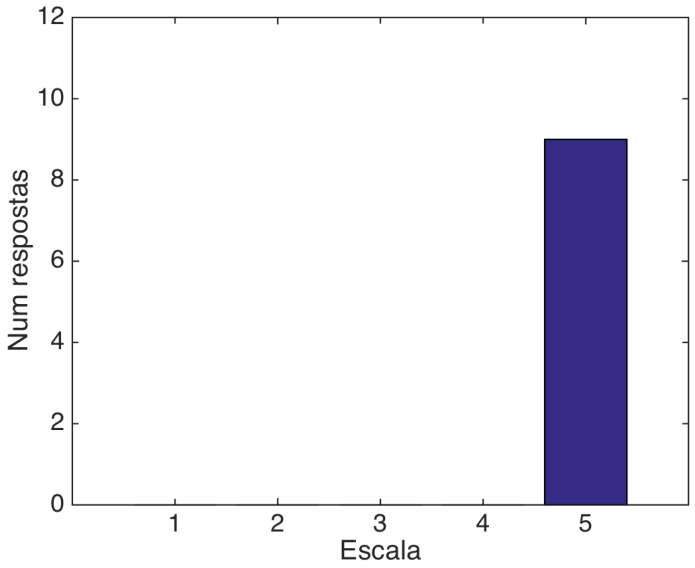


Figura A.4: Inquérito ROS - Histograma da distribuição de respostas relativo à questão 4 do inquérito.

Pergunta 5 - Consegui completar todas as tarefas?

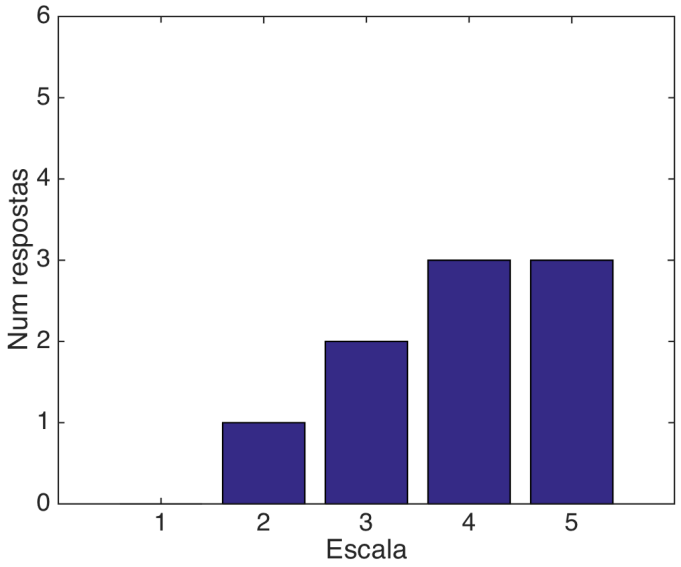


Figura A.5: Inquérito ROS - Histograma da distribuição de respostas relativo à questão 5 do inquérito.

A.4 Inquérito Universidade Júnior

Universidade Junior 15/07/2015

1. Assinala com um 'x' de 1 a 5 o nível de interesse de cada demonstração/robô apresentado.

	1 Pouco Interessante	2	3	4	5 Muito Interessante
a. Robô Tele					
b. 3pi					
c. LEGO EV3					
d. REDI					
e. Conde					

2. Dos cinco projetos mostrados, qual é a demonstração/robô que achaste mais interessante? E porquê?

3. Qual é o robô mais complicado?

4. Qual é o robô mais acessível?

5. Qual é o robô que eu gostava de ter em casa para programar? Porquê?

Obrigado!!!