

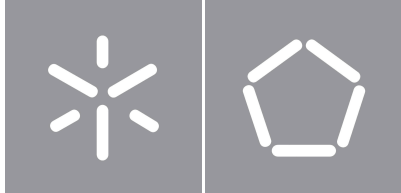
Universidade do Minho
Escola de Engenharia

Joel Alberto Abreu Sousa da Costa

Desenvolvimento de Robô Guarda-Redes Para a Equipa MinhoTeam

Dissertação de Mestrado
Engenharia Eletrónica Industrial e Computadores

Trabalho efetuado sob a orientação de
Professor Doutor António Fernando Macedo Ribeiro



Universidade do Minho
Escola de Engenharia

Joel Alberto Abreu Sousa da Costa

Desenvolvimento de Robô Guarda-Redes Para a Equipa MinhoTeam

Dissertação de Mestrado
Engenharia Eletrónica Industrial e Computadores

Trabalho efetuado sob a orientação de
Professor Doutor António Fernando Macedo Ribeiro

Direitos de Autor e Condições de Utilização do Trabalho por Terceiros

Este é um trabalho académico que pode ser utilizado por terceiros desde que respeitadas as regras e boas práticas internacionalmente aceites, no que concerne aos direitos de autor e direitos conexos.

Assim, o presente trabalho pode ser utilizado nos termos previstos na licença abaixo indicada.

Caso o utilizador necessite de permissão para poder fazer um uso do trabalho em condições não previstas no licenciamento indicado, deverá contactar o autor, através do RepositóriUM da Universidade do Minho.

Licença concedida aos utilizadores deste trabalho:



Atribuição

CC BY

<https://creativecommons.org/licenses/by/4.0/>

Agradecimentos

Dedico esta secção para transmitir os meus mais sinceros agradecimentos a todas as pessoas que contribuíram e me apoiaram ao longo destes anos de curso que são finalizados com esta dissertação.

Em primeiro lugar quero agradecer ao meu orientador Professor Dr. Fernando Ribeiro por todo o apoio e confiança que me foi transmitido assim como pela sua orientação que me fez crescer em todos os aspetos. Agradecer ao Professor Dr. Gil Lopes por toda a ajuda que deu a toda a equipa. Agradecer-lhes pela oportunidade de fazer parte do LAR, deste projeto de equipa que seria impossível sem as suas intervenções e ainda pela possibilidade de representar esta equipa em competições nacionais e internacionais.

Agradeço a todos os meus amigos, em especial aos membros da equipa por todos os dias e noites que passamos a trabalhar para alcançar o objetivo do projeto. Obrigado a todos por todas as experiências adquiridas que me permitiram evoluir a nível pessoal e profissional.

Agradecer à minha namorada Alisson Faria, que sem ela seria incapaz de finalizar o curso e que tornou esta minha etapa muito mais fácil. Obrigado por toda a paciência e trabalho que causei, e por toda a ajuda e incentivo que recebi.

À minha irmã e irmão, que sempre me apoiaram em todos os momentos e me permitiram focar em todo o trabalho da universidade. Obrigado por todos os conselhos que hoje me fazem uma pessoa melhor.

Por último, e mais importante, agradecer ao meu pai, Manuel Costa e à minha mãe, Anabela Costa, por esta oportunidade de continuar a estudar. Sei que tiveram de lutar muito para isso ser possível e sem vocês nada disto aconteceria. Por tudo, obrigado pai e obrigado mãe.

A todos, muito obrigado.

Declaração de Integridade

Declaro ter atuado com integridade na elaboração do presente trabalho académico e confirmo que não recorri à prática de plágio nem a qualquer forma de utilização indevida ou falsificação de informações ou resultados em nenhuma das etapas conducente à sua elaboração.

Mais declaro que conheço e que respeitei o Código de Conduta Ética da Universidade do Minho.

Universidade do Minho, Braga, janeiro 2024

Joel Costa

Joel Alberto Abreu Sousa da Costa

Resumo

Desde 1997 que a equipa MinhoTeam, equipa de futebol robótico do Laboratório de Automação e Robótica do Departamento de Eletrónica Industrial da Universidade do Minho, participa em vários torneios, sendo um deles a Middle Size League do RoboCup [1].

Este projeto consiste no desenvolvimento de uma nova geração de robôs capaz de fazer frente às equipas atuais participantes na MSL. Mais concretamente, este projeto foca-se no desenvolvimento do robô guarda-redes da equipa de forma que seja capaz de participar num jogo de futebol robótico, conseguindo defender qualquer oportunidade de golo da equipa adversária e que participe em jogadas com a equipa.

O desenvolvimento deste projeto passa então por módulos como visão por computador, controlo de movimento (robótica móvel) e alterações de *hardware* dos robôs da equipa anterior.

O módulo de visão tem de ser capaz de identificar remates ou oportunidades de golo da equipa adversária e calcular trajetórias da bola. Também tem de ser capaz de perceber onde a bola se encontra em campo e qual a melhor posição do guarda-redes na baliza de forma a reduzir as possibilidades de golo.

O módulo de controlo de movimento vai-se focar em mover o robô. Através de posições de referência, ou distâncias que tem de percorrer, este módulo deverá ser capaz de efetuar esses movimentos com a maior precisão possível.

As alterações de *hardware* são necessárias para atualizar a tecnologia usada por estes robôs. Uma função inovadora é a implementação de um sistema que permite ao guarda-redes aumentar temporariamente a sua área para cima, em dez centímetros num máximo de um segundo.

Palavras-Chave: Robótica móvel, RoboCup, MSL, Visão por Computador, Guarda-Redes

Abstract

Since 1997, the MinhoTeam robotic football team from the Laboratory of Automation and Robotics of the Department of Industrial Electronics at the University of Minho, has participated in several RoboCup leagues, one of them being the RoboCup Middle Size League [1].

The development of this project goes through modules such as computer vision, movement control and hardware updates on the robots.

The vision module must be able to identify shots or goal opportunities of the opposing team and calculate ball trajectories. The module also needs to understand where the ball is on the field and what is the best position for the goalkeeper in order to prevent as much goals as possible.

The movement control module focus on moving the robot. If the robot needs to go to reference positions, or go through certain distances, this module must be able to do these movements with the greatest possible precision.

Hardware changes happen for the necessity to update the technology used by these robots. An innovative function is the implementation of a system that allows the goalkeeper to increase his size upwards, by ten centimeters for a maximum of one second, permitted by the rules.

Keywords: Mobile Robotics, RoboCup, MSL, Computer Vision, Goalkeeper

Conteúdo

1	Introdução	1
1.1	MinhoTeam	1
1.2	RoboCup	2
1.2.1	RoboCupSoccer	2
1.2.2	RoboCup@Home	2
1.3	Middle Size League	4
1.4	Definição do problema	5
1.5	Objetivos	5
1.6	Motivação	6
1.7	Estrutura do documento	6
2	Estado da arte	7
2.1	Visão por computador	7
2.1.1	RGBD	8
2.1.2	HSV	9
2.1.3	Redes neuronais	9
2.2	Equipas participantes na MSL	11
2.2.1	CAMBADA	11
2.2.2	Tech United Eindhoven	13
2.2.3	Falcons	15
2.3	Conclusões	16
3	Robôs da equipa LAR@MSL	17
3.1	Estrutura mecânica	17
3.1.1	Sistema de manipulação de bola (dribblers)	19
3.1.2	Compartimento do computador	20

3.1.3	Estrutura do guarda-redes	21
3.1.4	Rodas omnidirecionais	22
3.2	Hardware	23
3.2.1	Arquitetura de hardware	23
3.2.2	Sistema de processamento	24
3.2.3	Sensores	28
3.2.4	Atuadores e sistema de controlo	29
3.2.5	Fonte de alimentação	33
3.3	Software	35
3.3.1	Código baixo nível	35
3.3.2	Comunicação entre hardware	37
3.3.3	Localização guarda-redes	38
3.3.4	Ferramentas utilizadas	40
3.3.5	Guarda-Redes como atacante	41
4	Implementação do Guarda-Redes	42
4.1	Sistema de visão	43
4.1.1	Drivers e bibliotecas	43
4.1.2	Deteção da bola	44
4.1.3	Posição da bola	47
4.1.4	Filtro da posição da bola	52
4.1.5	Deteção de remates	53
4.1.6	Defesa de remates	54
4.2	Posicionamento	58
4.3	Orientação	61
4.4	Controlo de movimento	63
4.5	Estrutura móvel	66
4.5.1	Tipo de material	67
4.5.2	Batentes	67
4.5.3	Suporte e veio do motor	68
4.5.4	Placa de controlo	70
4.5.5	Implementação	72
4.6	Defesa de penáltis	74

5	Testes e análise de resultados	76
5.1	Localização e orientação do robô	76
5.2	Sistema de visão	79
5.2.1	Deteção da bola e distância	79
5.2.2	Posição da bola	81
5.2.3	Deteção de remates	84
5.2.4	Defesa de remates	86
5.3	Posicionamento e orientação	88
5.4	Estrutura móvel	91
5.5	Defesa de penáltis	93
6	Competições	96
6.1	Festival Nacional de Robótica	96
6.2	RoboCup2023	97
7	Conclusão e trabalho futuro	99
7.1	Conclusão	99
7.2	Trabalho futuro	100

Lista de Figuras

1	RoboCup@Home - CHARMIE	3
2	Jogo da MSL [2]	4
3	Deteção de sinais de trânsito	8
4	Modelo de cores HSV [3]	9
5	Rede neuronal simples	10
6	Equipa CAMBADA [4]	11
7	Pipeline do sistema de visão do guarda-redes - CAMBADA [5]	12
8	Deteção de bola - CAMBADA [6]	13
9	Defesa Penálti - Tech United [7]	14
10	Protótipo Guarda-redes - Tech United [8]	14
11	Estrutura móvel - Falcons [9]	15
12	Chapas dos robôs	18
13	Robô futebolista	18
14	a) Peças dribblers. b) Motores dribblers.	19
15	Dribblers dos robôs	19
16	Caixa do computador	20
17	Esponja da caixa do computador	20
18	a) Estrutura do guarda-redes. b) Estrutura no guarda-redes.	21
19	Estrutura do guarda-redes alterada	21
20	Rodas omnidirecionais	22
21	a) Modelo da peça 3D do rolo. b) Peça 3D do rolo.	22
22	Arquitetura de hardware	24
23	Lenovo LEGION 5	25
24	Modelo 3D da placa baixo nível	26

25	Placa baixo nível sem componentes	26
26	Caixa da placa de baixo nível	27
27	Suportes da caixa da placa de baixo nível	27
28	a) Resultado final da placa de baixo nível. b) Suportes na caixa da placa de baixo nível. .	27
29	Unidade de processamento de baixo nível no robô	28
30	a) BlackFly PGE 13S2C. b) LIDAR URG-04LX-UG01.	28
31	Microsoft Kinect	29
32	Motores na base do robô	30
33	OMNI3D-MAX [10]	30
34	Componentes chuto eletromagnético	31
35	Ecrã robô	32
36	Caixa de potência	33
37	a) Máquina de solda por pontos. b) Células conectadas.	33
38	Packs baterias	34
39	Caixa bateria	34
40	Carril do robô	34
41	Fluxogramas código baixo nível	36
42	Diagrama de comunicação	37
43	Template localização	38
44	Localização robô	39
45	Habilidade dos robôs	41
46	Estrutura de processamento do Guarda-Redes	42
47	Deteção de Bola (HSV)	45
48	Segmentação de objetos	46
49	Ângulo da bola ao robô	47
50	Distância da bola ao centro	48
51	Posição da bola relativa ao robô	48
52	Rotação do referencial	49
53	Translação do referencial	50
54	Referenciais da BaseStation (vermelho) e do robô (verde)	51
55	Verificar Remate	53
56	Previsão da trajetória - Robô	55

57	Defesa Lateral	56
58	Defesa desnecessária	56
59	Previsão da trajetória - Baliza	57
60	Posicionamento na baliza	58
61	Elipse da baliza	59
62	Elipse com $B = 25$	60
63	Elipse com $B = 60$	60
64	Posicionamento final na baliza	61
65	Orientação por gama de valores	62
66	Equação 3° grau - Orientação	62
67	Fluxograma da Cinemática	63
68	Atrator velocidade linear	64
69	Direção do movimento linear	65
70	Modelo da estrutura móvel	66
71	Estrutura modelizada	67
72	a) Modelo dos batentes. b) Batentes na estrutura.	68
73	Modelo do suporte do motor	68
74	Modelo do veio do motor	69
75	Motor da estrutura móvel	69
76	Arquitetura de <i>hardware</i> da estrutura móvel	70
77	Ligação do microcontrolador (estrutura móvel)	71
78	Modelo do compartimento da placa (estrutura móvel)	72
79	Placa de controlo	72
80	Placa no robô	72
81	Máquina de estados (estrutura móvel)	73
82	Sequência de movimento no penáti	75
83	Estrutura de processamento com penáti	75
84	a) Imagem da <i>Microsoft Kinect</i> . b) Posição real do robô. c) Representação da localização do robô no campo. d) Coordenadas obtidas pelo robô em milímetros (x,y). (1° teste)	77
85	a) Imagem da <i>Microsoft Kinect</i> . b) Posição real do robô. c) Representação da localização do robô no campo. d) Coordenadas obtidas pelo robô em milímetros (x,y). (2° teste)	77
86	a) Imagem da <i>Microsoft Kinect</i> . b) Ângulo obtido pelo robô	78

87	Gráfico da orientação do Robô	78
88	a) Máscara para detecção de cor laranja. b) Detecção da bola	79
89	a) Máscara para detecção de cor laranja. b) Detecção de apenas uma bola (HSV)	79
90	Detecção da bola limitada pela distância (HSV)	80
91	Detecção e distância da bola (rede neuronal)	80
92	Detecção das bolas (rede neuronal)	81
93	a) Imagem da <i>Microsoft Kinect</i> . b) Coordenadas da bola obtidas em centímetros (x,y). c) Representação da posição da bola no campo. (Posição da bola - 1º teste)	82
94	a) Imagem da <i>Microsoft Kinect</i> . b) Coordenadas da bola obtidas em centímetros (x,y). c) Representação da posição da bola no campo. (Posição da bola - 2º teste)	82
95	a) Imagem da <i>Microsoft Kinect</i> . b) Coordenadas da bola obtidas em centímetros (x,y). c) Representação da posição da bola no campo. (Posição da bola - 3º teste)	83
96	a) Imagem da <i>Microsoft Kinect</i> . b) Coordenadas da bola obtidas em centímetros (x,y). c) Representação da posição da bola no campo. (Posição da bola - 4º teste)	84
97	a) Remate efetuado visto pela imagem da <i>Microsoft Kinect</i> . b) Posição da bola após o remate na baliza representada a preto. c) Remate efetuado noutro ângulo. (1º teste)	85
98	a) Remate efetuado visto pela imagem da <i>Microsoft Kinect</i> . b) Posição da bola após o remate na baliza representada a preto. c) Remate efetuado noutro ângulo. (2º teste)	85
99	a) Remate efetuado visto pela imagem da <i>Microsoft Kinect</i> . b) Posição da bola após o remate na baliza representada a preto. c) Remate efetuado noutro ângulo. (3º teste)	86
100	a) Remate efetuado visto pela imagem da <i>Microsoft Kinect</i> . b) Posição e orientação desejada do robô. c) Posição do robô na baliza. (Defesa de remate - 1º teste)	87
101	a) Imagem da <i>Microsoft Kinect</i> após remate. b) Posição e orientação desejada do remate (Defesa de remate - 2º teste)	87
102	a) Momento do remate. b) Posição e orientação final do robô (Defesa de remate - 2º teste)	87
103	Gráfico do erro da posição do robô em remates	88
104	a) Imagem da <i>Microsoft Kinect</i> . b) Posição desejada do robô c) Representação do cálculo da posição do robô no campo (elipse). (1º teste)	89
105	a) Imagem da <i>Microsoft Kinect</i> . b) Posição desejada do robô c) Representação do cálculo da posição do robô no campo (elipse). (2º teste)	89
106	a) Imagem da <i>Microsoft Kinect</i> . b) Posição e orientação desejada. c) Posição e orienta- ção real do robô na baliza. (Posicionamento e orientação - 1º teste)	90

107	a) Imagem da <i>Microsoft Kinect</i> . b) Posição e orientação desejada. c) Posição e orientação real do robô na baliza. (Posicionamento e orientação - 2º teste)	90
108	a) Imagem da <i>Microsoft Kinect</i> . b) Posição e orientação desejada. c) Posição e orientação real do robô na baliza. (Posicionamento e orientação - 3º teste)	91
109	Altura máxima da estrutura móvel	91
110	Estrutura móvel subida	92
111	Defesa com a estrutura móvel	92
112	Teste de trajetórias no penáلت	93
113	a) Imagem da <i>Microsoft Kinect</i> . b) Representação do robô (azul) na baliza (preto) e das coordenadas desejadas em centímetros. (Cálculo da trajetória do penáلت - 1º teste) . . .	93
114	a) Imagem da <i>Microsoft Kinect</i> . b) Representação do robô (azul) na baliza (preto) e das coordenadas desejadas em centímetros. (Cálculo da trajetória do penáلت - 2º teste) . . .	94
115	a) Imagem da <i>Microsoft Kinect</i> . b) Posição e orientação desejados para o robô. c) Posição e orientação real do robô na baliza. (Defesa do penáلت - 1º teste)	94
116	a) Imagem da <i>Microsoft Kinect</i> . b) Posição e orientação desejados para o robô. c) Posição e orientação real do robô na baliza. (Defesa do penáلت - 2º teste)	95
117	Festival Nacional de Robótica	97
118	RoboCup 2023	98

Lista de Acrónimos

3D Três Dimensões

ADC Analog-to-Digital Converter

DC Direct Current

GB GigaByte

GPIO General Purpose Input/Output

GPU Graphics Processing Unit

HSV Hue, Saturation, Value

I2C Inter-Integrated Circuit

IA Inteligência Artificial

LAR Laboratório de Automação e Robótica

LUT Look-Up Table

MSL Middle Size League

PC Portable Computer

PLA Poliacido Láctico

PPR Polipropileno Copolímero Aleatório

PWM Pulse Width Modulation

RAM Random Access Memory

RGB Red, Green, Blue

SSD Solid-State Drive

TCP Transmission Control Protocol

WLAN Wireless Local Area Network

Capítulo 1

Introdução

Esta dissertação tem como objetivo documentar o trabalho desenvolvido durante todo o projeto do robô guarda-redes para a equipa MinhoTeam.

Neste primeiro capítulo é apresentada essa equipa, o *RoboCup* e a liga em que este projeto se insere, a Middle Size League ([MSL](#)). É ainda apresentada a definição do problema, assim como os objetivos, motivação e estrutura do documento.

1.1 MinhoTeam

A *MinhoTeam* é uma equipa de robótica da Universidade do Minho em Portugal que compete em muitas competições, sendo uma delas a *RoboCup*. A equipa participou em várias ligas diferentes na competição, como *RoboCup@Home*, *RoboCupJunior Soccer* e ainda a [MSL](#) [11].

A *MinhoTeam* teve sucesso na competição RoboCup e sempre se classificou entre as melhores equipas nas suas ligas. Esta equipa dá a oportunidade de trabalhar em projetos de investigação, de desenvolvimento em robótica e participar em competições nacionais e internacionais [12].

A última vez que a *MinhoTeam* participou na [MSL](#) foi em 2016. Era utilizado um sistema de visão omnidirecional e uma câmara com canal de profundidade (apenas no guarda-redes). Uma das tecnologias a salientar, era o uso de três rodas omnidireccionais que permitiam ao robô obter uma movimentação omnidirecional [13].

A equipa desde que iniciou a sua atividade tem desenvolvido várias versões de robôs para competir na [MSL](#). Nessas versões foram implementadas algumas soluções inovadoras que ainda hoje são utilizadas por outras equipas, tais como, o sistema de chute eletromagnético [14] e os *dribblers* [15].

1.2 RoboCup

O RoboCup é uma competição internacional anual de robótica fundada em 1997 com o objetivo de promover a pesquisa e o desenvolvimento na área de robótica. O objetivo final do *RoboCup* é ter uma equipa de futebol constituída por robôs totalmente autónomos que sejam capazes de vencer, obedecendo às regras oficiais da *FIFA*, a equipa vencedora do Campeonato Mundial de humanos no ano de 2050 [1].

No entanto, devido ao crescimento e necessidade, o *RoboCup* expandiu para novos desafios, sendo estes:

- *RoboCupSoccer*
- *RoboCupRescue*
- *RoboCup@Home*
- *RoboCupIndustrial*
- *RoboCupJunior*

Nesta secção é abordada a liga *RoboCupSoccer* devido à participação da equipa, assim como a liga *RoboCup@Home* pois o Laboratório de Automação e Robótica (LAR) também tem uma equipa participante que se mostrou de grande ajuda no desenvolvimento do projeto.

1.2.1 RoboCupSoccer

Nesta competição o desafio é o futebol robótico. Todas as ligas que englobam esta competição têm como objetivo a realização de jogos de futebol de duas equipas com robôs completamente autónomos. Estes têm de ser cooperativos num ambiente dinâmico que é o futebol [16].

Algumas das ligas focam-se apenas no desenvolvimento de *software* (*Simulation* e *Standard Platform*), enquanto outras procuram também o desenvolvimento de *hardware* (*Humanoid*, *Middle Size*, *Small Size*).

Um requisito da liga de futebol robótico é que todos os robôs envolvidos na competição têm de ser completamente autónomos. A *MSL* é abordada individualmente devido à participação deste projeto na competição.

1.2.2 RoboCup@Home

Esta competição tem o objetivo de desenvolver robôs de serviços para uso profissional e doméstico [17]. De forma a avaliar as habilidades e o desempenho dos robôs é efetuado um conjunto de testes em

ambientes domésticos realistas não padronizados. O foco da liga passa pelos seguintes pontos:

- Interação e Cooperação Humano-Robô
- Navegação e Mapeamento em ambientes dinâmicos
- Visão Computacional e Reconhecimento de Objetos sob condições de luz natural
- Manipulação de Objetos
- Comportamentos Adaptativos
- Integração Comportamental
- Inteligência Ambiental
- Padronização e Integração de Sistemas

Esta competição conta com três ligas, *Open Platform*, *Domestic Standard Platform* e *Social Standard Platform*.

A liga *Open Platform* não tem nenhuma restrição de *hardware* e oferece maior liberdade, pois o robô pode ser desenvolvido pelas equipas. A liga *Domestic Standard Platform* concentra-se em tarefas domésticas, sendo obrigatório o uso do robô *TOYOTA*, enquanto a *Social Standard Platform* foca-se nos aspetos sociais da interação de robôs humanos, utilizando o robô *Pepper*.

A Figura 1 apresenta o robô desenvolvido pelo LAR da Universidade do Minho para participar nesta competição, o *CHARMIE* (*Collaborative Home Assistant Robot Minho Industrial Electronics*) [18].

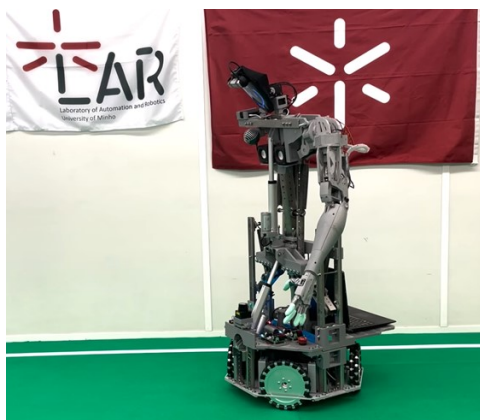


Figura 1: RoboCup@Home - CHARMIE

1.3 Middle Size League

Esta liga existe desde 1997 com o objetivo de derrotar a equipa campeã mundial de humanos no ano de 2050. Desde a sua existência foram recorrentes ao longo dos anos alterações nas regras de forma a se aproximar do objetivo final. Uma dessas alterações foi em 2008 onde houve uma remoção das cores das balizas, pois até então uma baliza era amarela e outra azul para facilitar o processo de localização dos robôs. Esta atualização aproxima as equipas do objetivo final, pois num campo de futebol as balizas são brancas e desta forma o robô precisa de se localizar com as linhas e dimensões do campo, sendo necessário conseguir identificar as balizas e objetos desse [19].

As dimensões do campo são de 12 por 18 metros. O interior da baliza tem um metro de altura, dois metros de largura e cinquenta centímetros de profundidade e os robôs têm que possuir dimensões iguais ou inferiores às medidas de uma caixa de 52x52x80 centímetros. O guarda-redes pode aumentar a área lateral temporariamente em dez centímetros em qualquer direção. O robô pode pesar até um máximo de 40kg.

Os jogos têm duas partes de quinze minutos cada, com um intervalo máximo de 10 minutos entre elas. As equipas são constituídas por cinco robôs, sendo que um é o guarda-redes. Como num jogo de futebol existem árbitros na partida que marcam faltas, foras, etc. As informações do árbitro são passadas aos robôs através de um computador externo denominado *BaseStation* que também pode calcular a estratégia da equipa. De forma a efetuar esta comunicação entre a *BaseStation* e os robôs é fornecida uma rede Wireless Local Area Network (WLAN).



Figura 2: Jogo da MSL [2]

1.4 Definição do problema

A última geração de robôs futebolistas desenvolvidos até à data do início deste projeto não dispunha de características ao nível das atuais exigências da [MSL](#), tanto pela evolução das equipas como pela evolução das regras da liga. Por isso, pretende-se construir uma nova geração de robôs, a qual deve ter em conta a atual dinâmica de jogo. Para o desenvolvimento desta equipa de robôs futebolistas com as atuais exigências, é necessário um robô guarda-redes que seja capaz de assegurar a equipa para a defesa da baliza.

O *hardware* e *software* necessários para este projeto são desenvolvidos por seis alunos do 5º ano do Mestrado Integrado em Engenharia Eletrónica Industrial e Computadores da Universidade do Minho.

1.5 Objetivos

Este trabalho está inserido no desenvolvimento duma nova geração de robôs, consistindo na especificação e desenvolvimento do robô guarda-redes, um robô capaz de defender qualquer oportunidade de golo num jogo de futebol e permitindo que o mesmo participe nas jogadas de equipa.

O desenvolvimento deste robô está inserido em módulos de visão, controlo de movimento, alterações da estrutura e também de *hardware*.

O objetivo do módulo de visão é que o robô seja capaz de identificar remates ou passes, e que em caso de remate ou situação eminente de golo, consiga detetar a trajetória da bola e defender a mesma. O sistema de visão também precisa de identificar onde a bola se localiza em campo de forma a ter o melhor posicionamento possível na baliza.

O módulo de controlo de movimento é responsável tanto por mover o robô, como por localizar o mesmo. O robô tem de ser capaz de se localizar no campo assim como identificar outros objetos no mesmo. A movimentação do robô tem de ser eficaz, onde este módulo tem de atualizar os atuadores de forma ao robô mexer-se para as posições desejadas da baliza.

As alterações da estrutura e de *hardware* devem-se ao facto dos robôs não disporem das características ao nível das atuais exigências. Uma grande alteração efetuada foi adicionar mecanismos que permitem ao guarda-redes aumentar a sua área em dez centímetros de altura, por no máximo, um segundo.

1.6 Motivação

Tendo como objetivo recriar a equipa *MinhoTeam* e voltar a participar nos campeonatos de futebol robótico, é necessário o desenvolvimento de um guarda-redes que seja capaz de atuar nas competições com as exigências atuais. A motivação para o desenvolvimento deste projeto para a equipa passa pela importância estratégica deste robô guarda-redes para a mesma, sendo necessário haver um foco individual para este robô devido às ações diferentes que este exerce em comparação com os restantes robôs.

Com o desenvolvimento deste projeto espera-se que a *MinhoTeam* volte às competições e também melhore a sua prestação nas competições em que venha a participar.

1.7 Estrutura do documento

Este primeiro capítulo faz uma introdução ao projeto, abordando a equipa em que este se insere assim como as ligas do *RoboCup* que essa participa. São também definidos o problema e objetivos assim como a motivação para este projeto. Além deste, existem mais 6 capítulos.

O capítulo 2 faz uma análise ao que é robótica, robótica móvel e visão por computador. Ainda neste capítulo, é feito um estudo de métodos utilizados por outras equipas na [MSL](#).

O capítulo 3 aborda os robôs da equipa com uma descrição das alterações na estrutura mecânica, *hardware* e *software*. É um capítulo que descreve o trabalho efetuado que não se enquadra nos objetivos desta dissertação, mas que necessita de ser referido devido à sua relevância e trabalho, pois sem essas alterações não seria possível de efetuar o desenvolvimento do guarda-redes da equipa.

No capítulo 4, denominado "Implementação do Guarda-Redes", são apresentados todos os métodos desenvolvidos para cumprir os objetivos impostos no guarda-redes da equipa, sendo este o tópico principal da dissertação.

O capítulo 5 contém os testes efetuados, assim como os resultados obtidos dos métodos desenvolvidos para os objetivos do guarda-redes, sendo feita uma análise a todos esses resultados.

No capítulo 6 é feito um resumo das participações da equipa no Festival Nacional de Robótica e no *RoboCup2023*.

O capítulo 7, como último capítulo, é constituído pelas conclusões retiradas no desenvolvimento deste projeto, assim como sugestões e ideias para trabalho futuro sobre o guarda-redes.

Capítulo 2

Estado da arte

Neste capítulo é feita uma descrição do que é visão por computador e de dois métodos diferentes para detecção de objetos, através de visão clássica (modelo de cores) ou através de redes neurais.

Por fim, são vistas outras equipes participantes na [MSL](#) de forma a estudar os melhores e mais recentes métodos utilizados na construção do robô guarda-redes.

2.1 Visão por computador

A visão por computador é um campo da Inteligência Artificial ([IA](#)) que permite que computadores e outros sistemas obtenham informações significativas de imagens digitais, vídeos e outras entradas visuais, tomando ações com base nessas informações. Se a [IA](#) permite que os computadores "pensem", a visão por computador permite que eles vejam, observem e "entendam".

A visão por computador funciona da mesma forma que a visão humana, no entanto, os humanos têm uma vantagem inicial. A visão humana tem a vantagem de várias ajudas externas, como outros sentidos e até ajuda de outras pessoas para compreensão do observado, assim como anos de "treino" que permitem identificar os objetos, a que distância estão, se eles se estão a mover, etc.

A visão por computador treina as máquinas para executar essas funções, mas precisa de fazer isso em muito menos tempo com câmaras, informações e algoritmos, em vez de retinas, nervos óticos e córtex visual. Por exemplo, um sistema treinado para inspecionar produtos, pode analisar milhares de produtos por minuto, detetando defeitos ou problemas impercetíveis desse produto, ultrapassando rapidamente as capacidades humanas [20]. A Figura 3, retirada de um projeto do [LAR](#), mostra um sistema de visão capaz de identificar sinais de trânsito.

O sistema de visão representado na Figura 3 utiliza redes neurais para a identificação de objetos, sendo essas explicadas num dos tópicos adiante.



Figura 3: Detecção de sinais de trânsito

Contudo existem sistemas de visão que não utilizam estas redes neurais para a detecção. Na [MSL](#) grande parte das equipas utiliza as cores dos objetos para detecção dos mesmos, assim como câmaras com canal de profundidade para auxílio desse sistema de visão. Desta forma é abordado a seguir este tipo de câmaras utilizadas (*RGBD*) para estes sistemas de visão, assim como o sistema de cor para a detecção de objetos (Hue, Saturation, Value (*HSV*)).

2.1.1 RGBD

RGBD é uma sigla referente à informação de uma câmara da combinação de Red, Green, Blue (*RGB*) com Depth (profundidade), analisando-se separadamente os dois tópicos.

RGB refere-se à informação de cor de uma imagem, sendo estas compostas por pixels, apresentando cada um valores de cores de diferentes combinações de vermelho, verde e azul. Desta forma, atribuindo valores a cada um dos parâmetros (vermelho, verde e azul) é possível obter qualquer cor [\[21\]](#). O modelo *RGB* usa 8 bits para cada parâmetro, tendo cada cor uma gama de valores entre 0 e 255. Desta forma, estes valores são traduzidos para uma gama de 16 777 216 cores.

O segundo tópico refere-se à informação de profundidade de uma imagem indicando a distância entre a câmara e os objetos da imagem. Num sensor *RGBD*, como a *Microsoft Kinect* (câmara utilizada), além das informações de cor é possível medir a distância entre a câmara e esses objetos. É frequentemente representado como um mapa de profundidade, onde os valores dos pixels representam a distância dos objetos que esses pixels correspondem em relação à câmara. [\[22\]](#)

Assim, *RGBD* é a combinação das informações de cor e profundidade em duas representações de imagem, que são frequentemente usadas em visão por computador para obter uma melhor precessão do mundo tridimensional.

2.1.2 HSV

HSV é um modelo de cor que as representa de forma diferente do RGB. Enquanto o modelo RGB descreve as cores em função dos parâmetros vermelho, verde e azul, o modelo HSV descreve as cores em função de três parâmetros, sendo estes a matiz (*hue*), a saturação (*saturation*) e o valor (*value*). [23]

A matiz representa a tonalidade da cor, como vermelho, azul, verde, etc. A matiz é representada em graus num círculo de cores, onde 0° representa o vermelho puro, 120° o verde e 240° o azul. Este parâmetro é o que permite distinguir cores diferentes.

A saturação refere-se à intensidade da cor, onde quanto maior for a saturação, mais "pura" é a cor, de forma que quando a saturação é zero, a cor tem uma tonalidade cinza.

O valor descreve a luminosidade variando entre 0 e 100%, onde 0% representa o preto (ausência de luz) e 100% representa a luminosidade máxima.

A Figura 4 representa o modelo HSV.

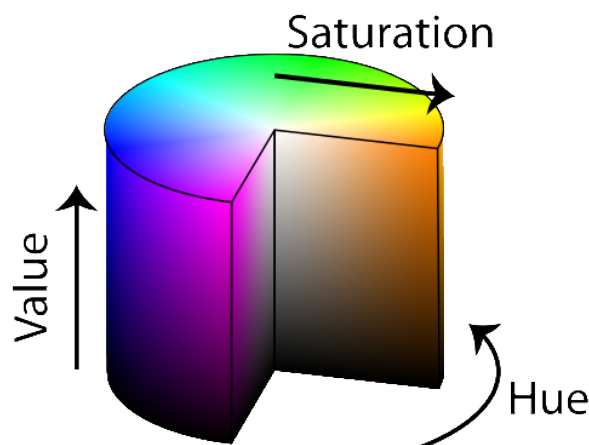


Figura 4: Modelo de cores HSV [3]

O espaço de cor HSV é frequentemente usado em processamento de imagem para realizar manipulações de cores e segmentação de objetos com base na matiz, tornando mais fácil isolar objetos de diferentes cores numa imagem. Este espaço de cor é o sistema mais utilizado na MSL para identificar a bola.

2.1.3 Redes neurais

O conceito de rede neuronal no âmbito da computação é baseado no conceito duma rede neuronal humana. O propósito duma rede neuronal é reconhecer uma relação entre dados. Esta é treinada de

forma a aprender a realizar um comportamento desejado, e para tal são fornecidos *inputs* e estímulos à rede, que por sua vez irá retornar um *output* após ser treinada, com a solução que pensa ser correta. Uma simples comparação com o funcionamento de um cérebro humano ajuda a perceber o funcionamento de uma rede neuronal, onde as dendrites podem ser consideradas os *inputs*, o corpo da célula pode ser considerado como os pesos da rede neuronal e a saída do neurónio como o *output* da rede neuronal. [24]

A Figura 5 demonstra o funcionamento de uma simples rede neuronal.

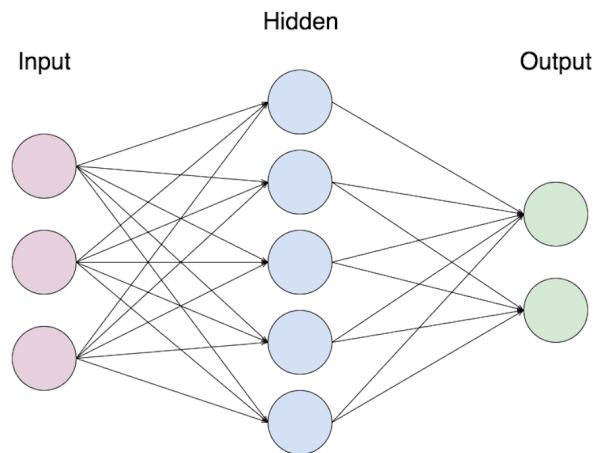


Figura 5: Rede neuronal simples

Como se pode perceber na Figura 5, existem 3 camadas numa rede neuronal simples: entrada, intermédia e saída. Na camada intermédia existem diversos neurónios, que calculam o valor à saída consoante os dados fornecidos pela entrada, consoante os pesos de cada neurónio e consoante uma tendência externa que possa existir. O valor à saída pode ser calculado seguindo a seguinte lógica:

$$combination = bias + weights * inputs \quad (2.1)$$

$$output = activation(combination) \quad (2.2)$$

Primeiro calcula-se um valor intermédio, somando o valor dos *bias* à multiplicação dos *inputs* com os pesos de cada neurónio. Uma vez obtido este valor intermédio, recorre-se a uma função de ativação, que tem em consideração um valor a partir do qual a saída é ativada, e analisando o valor intermédio, é obtido o valor à saída da rede.

Nos sistemas de visão estas redes neurais são utilizadas para deteção de objetos como exemplificado na Figura 3 com a identificação dos sinais de trânsito. Para o treino destas redes neurais, são dados como *inputs* imagens com a identificação dos objetos desejados. Após bastantes imagens com essas identificações, a rede neuronal será capaz de identificar esses objetos.

2.2 Equipas participantes na MSL

Nesta secção pretende-se analisar algumas equipas que já participaram na [MSL](#), e visto que uma parte importante desta competição é a partilha de conhecimento, estas equipas são uma grande referência na competição.

2.2.1 CAMBADA

CAMBADA (*Cooperative Autonomous Mobile Robots with Advanced Distributed Architecture*) é a equipa de futebol que participa no *RoboCup MSL* da Universidade de Aveiro, Portugal. O projeto começou oficialmente em outubro de 2003.

Neste tópico aborda-se o desenvolvimento tecnológico do robô guarda-redes da equipa, assim como os sistemas de localização e de visão.

De forma a obter um guarda-redes mais eficiente, produziram um quadro extensível que permitisse defender remates mais altos, utilizando a regra da competição onde o guarda-redes se pode estender no máximo até dez centímetros num espaço de tempo até um segundo [\[25\]](#).

Além disso, o guarda-redes foi equipado com a última geração da câmara *Kinect* da *Microsoft*, aumentando a resolução da imagem colorida de 640x480 para 1920x1080, assim como a resolução da câmara de profundidade de 320x240 para 512x424, mantendo a taxa de 30 frames por segundo.

Para lidar com os requisitos da câmara *Kinect* (*USB3* e Graphics Processing Unit ([GPU](#)) dedicado), foi incluída uma *NVidia Jetson TK2* para cuidar do processamento de imagem.

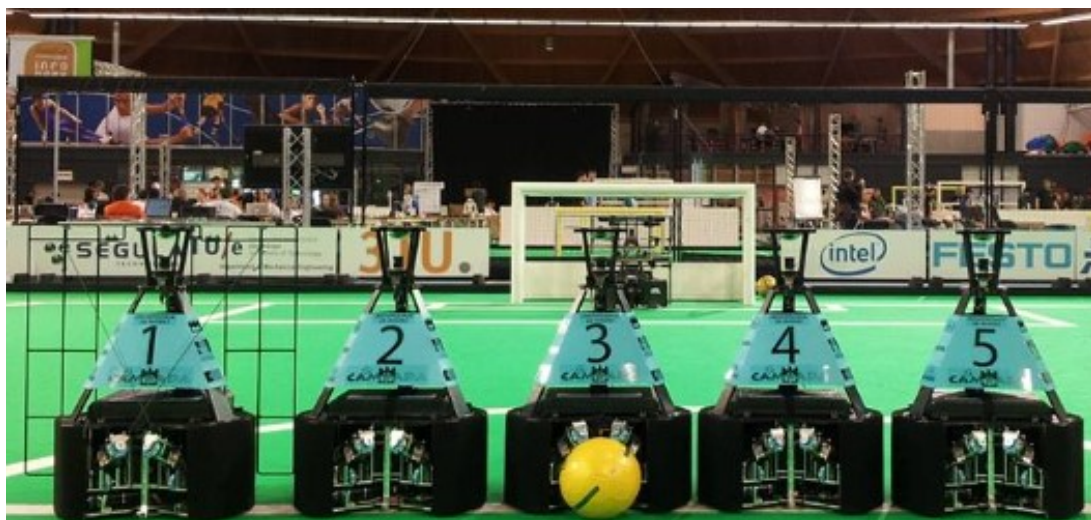


Figura 6: Equipa CAMBADA [\[4\]](#)

De forma ao guarda-redes se localizar é utilizada um módulo de visão omnidirecional que dá uso a uma câmara e um espelho. Posterior à visão é utilizado um sistema de localização complementar baseado em sensores de ultrassom [26]. A principal vantagem do uso deste sensor consiste no facto de o guarda-redes se encontrar sempre perto da baliza, o que faz com que o mesmo esteja sempre bem orientado na baliza fornecendo uma posição absoluta para o sistema de visão.

Para o guarda-redes saber a localização dos outros robôs de equipa, cada robô tem vários recetores de ultrassom, que respondem quando recebem um pulso do guarda-redes. Como cada robô usa um tempo de resposta diferente, o guarda-redes, através de dois sensores ultrassónicos, consegue obter as coordenadas dos robôs de equipa no campo.

Para a visão do guarda-redes, a equipa utiliza um filtro de cores de objetos de forma a identificar a bola. Utilizando a câmara *Kinect*, a equipa desenvolveu um algoritmo capaz de detetar bolas aéreas [5].

O pipeline do sistema de visão do guarda-redes é apresentado na Figura 7.

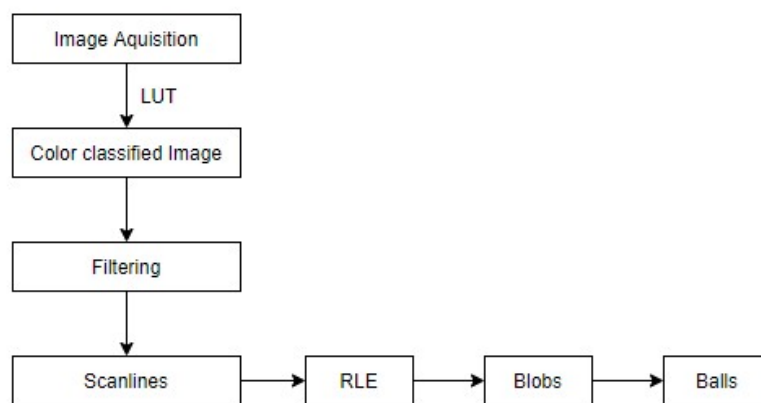


Figura 7: Pipeline do sistema de visão do guarda-redes - CAMBADA [5]

Depois de realizar uma segmentação de cores na imagem de entrada usando uma Look-Up Table (LUT), é aplicado um filtro com base nas informações de profundidade de forma a remover os objetos da classificação de cores que estão fora do campo de futebol. O *ScanLines* serve para pesquisar pixels da cor de interesse (a cor da bola de futebol) formando-se *blobs* dessa cor. Esses *blobs* passam por um processo de validação para estabelecer se um determinado *blob* é uma bola. O procedimento de validação é baseado no cálculo de recursos diferentes para cada um dos *blobs* encontrados, como a área da caixa delimitadora, a circularidade e relação largura-altura.

Na Figura 8 pode ver-se que na primeira imagem é feita uma segmentação de cores que depois é filtrada, retirando cores fora de campo (segunda imagem) com uso da distância, extraíndo-se todos os pontos que estejam mais longe que o tamanho do campo, detetando no fim a bola no ar [6].

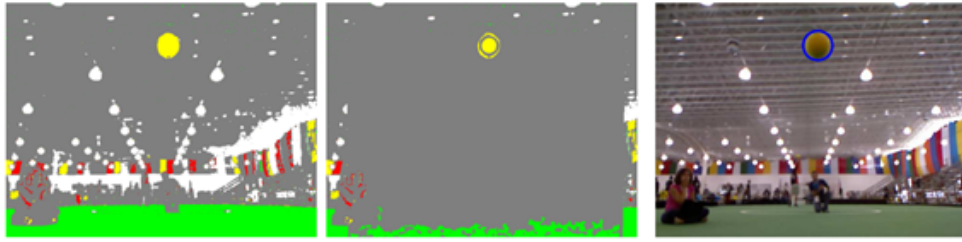


Figura 8: Detecção de bola - CAMBADA [6]

2.2.2 Tech United Eindhoven

A *Tech United* é uma equipa de (antigos) alunos, *PhDs* e funcionários da Universidade de Tecnologia de Eindhoven que estão envolvidos no desenvolvimento da robótica [27].

A nível de visão do guarda-redes é utilizada um sistema de visão omnidirecional, assim como uma câmara *Kinect* com canal de profundidade. Os algoritmos de deteção de bola são bastante semelhantes com os usados pela equipa *CAMBADA* (equipa mencionada anteriormente).

Assim sendo, são abordadas umas ideias diferentes aplicadas pelos *Tech United*, que são a defesa de penáltis e a defesa da bola para o seu sistema de manipulação da mesma (*Dribblers*).

2.2.2.1 Defesa de penáltis

Na MSL os penáltis são marcados a uma distância de três metros para a baliza que tem uma largura de dois metros. Um robô da MSL consegue chutar uma bola até doze metros por segundo o que faz com que o guarda-redes necessite de uma reação de 250 milissegundos, devido ao mesmo ter uma largura máxima de setenta centímetros com a possibilidade de se aumentar mais dez.

Devido a todo o processamento e ainda ao facto de o guarda-redes ter de se mexer quarenta centímetros para um dos lados, seria necessário uma aceleração 14 m/s^2 , tornando assim praticamente impossível a defesa de penáltis.

Desta forma, a equipa criou uma estratégia de defesa de penáltis. Como os robôs têm de rodar em volta da bola de forma a orientar-se para o lado que querem rematar, o guarda-redes da equipa leva esse movimento em consideração, movimentando-se para a zona da baliza mesmo antes do robô chutar.

Este movimento é explicado na Figura 9, onde o robô está de frente com a bola, e quando se começa a movimentar, o guarda-redes responde com o seu movimento, o que o faz conseguir defender a bola [7].

Esta estratégia permitiu à equipa ganhar o *RoboCup* em 2016 com cinco penáltis defendidos de cinco marcados.

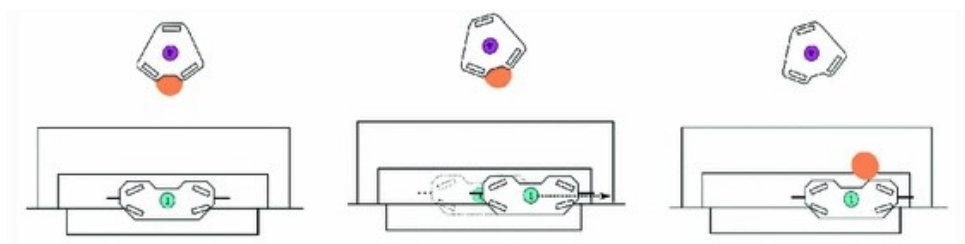


Figura 9: Defesa Penáلتi - Tech United [7]

2.2.2.2 Guarda-redes segurar a bola

A tecnologia seguinte, tem como objetivo combater o facto do guarda-redes nunca segurar a bola, pois sempre que o mesmo defende, a bola vai para uma zona aleatória, podendo ser jogadores, zonas vazias, ou até mesmo para fora de campo. De forma a evitar essas situações, a equipa criou um método para o guarda-redes segurar a bola quando defende a mesma.

Para o guarda-redes conseguir segurar a bola, a energia cinética da bola tem de ser dissipada num curto espaço de tempo. Utilizando um tecido com tensão folgada, existe mais tempo para dissipar essa energia, sendo utilizado um pano e uma rede na frente e nas costas do robô que dissipam a maior parte da energia cinética da bola antes de a apanhar para os seus *dribblers*. Esse pano dá à bola um movimento ascendente, devido ao fato da bola não ser capaz de se separar imediatamente do pano, reduzindo a velocidade da bola, sendo necessária a rede para levar a bola até ao buraco que conecta a bola aos *dribblers* (Figura 10).

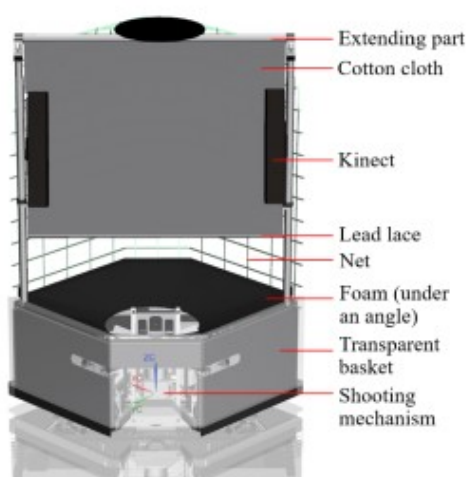


Figura 10: Protótipo Guarda-redes - Tech United [8]

Durante o primeiro conjunto de experiências que a equipa fez, o guarda-redes foi capaz de apanhar a bola 82% das vezes (26 remates). O que acontecia, era que a bola ao ir da rede em direção ao buraco,

esta falhava o mesmo e em seguida saltava para fora. A solução foi adicionar uma camada de espuma em baixo, impossibilitando a bola de saltar devido à redução de energia. No segundo teste a percentagem de bolas apanhadas foi de 100% (em 39 remates) [8].

Esta solução demonstra-se bastante eficaz para combater lances de jogo em que o guarda-redes defende a bola mas a posse da mesma é perdida, devido ao mesmo não conseguir ficar com a bola. Evitam-se assim possíveis lances de golo em que os adversários marcavam após o guarda-redes defender na sua direção.

2.2.3 Falcons

Falcons é uma equipa formada em novembro de 2013 que participa na [MSL](#), sendo que todos os seus membros são funcionários da empresa *ASML*, estando situados em Eindhoven como a equipa anterior.

A tecnologia referida desta equipa é a estrutura móvel. Essa é uma das tecnologias mais recentes instaladas nos guarda-redes atuais, permitindo ao guarda-redes estender a sua área num máximo de um segundo.

A estrutura que a equipa desenvolveu é feita em fibra de carbono para se obter a máxima força com o menor peso. Para o acionamento da estrutura foi escolhido um atuador elétrico por ser leve, robusto e rápido. Utilizado um design com um íman móvel e uma bobina, são evitados cabos na zona da estrutura móvel, tornando a mesma menos suscetível a danos em cabos e na própria estrutura. Numa situação normal a bobina retém o íman com uma força de retenção. Quando o comando para estender a estrutura é enviado, leva menos de 100 milissegundos para a estender completamente. São utilizados três motores, sendo dois para os lados e um para cima [9].

Na Figura 11 é possível observar a estrutura móvel desenvolvida, assim como os atuadores utilizados bem como a sua localização.

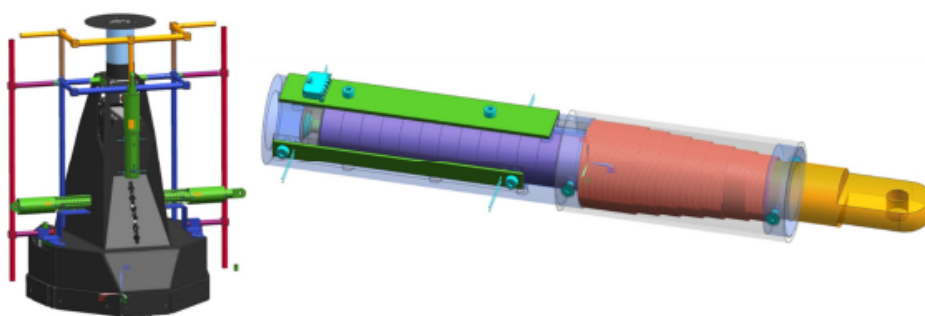


Figura 11: Estrutura móvel - Falcons [9]

2.3 Conclusões

Sendo o *RoboCup* uma competição de robótica, demonstra uma gama de equipas bastante desenvolvidas e com tecnologias inovadoras, desde o desenvolvimento de *hardware* até sistemas de *software* bastante robustos e com aplicações no dia a dia.

Sendo a competição que a equipa participou bastante relevante, as equipas apresentadas demonstram tecnologias muito interessantes. Uma dessas equipas, os *CAMBADA*, já não competiram nas últimas edições, sendo uma das equipas mais competitivas nos tempos em que participavam.

As outras duas equipas apresentadas, os *Tech United* e os *Falcons*, foram as equipas que conquistaram os dois primeiros lugares na última competição, sendo os campeões a equipa dos *Tech United*. Estas duas equipas vêm constantemente a disputar estes lugares nos últimos anos, sendo fruto de muitos anos de trabalho contínuo em investigação e desenvolvimento de novos algoritmos e *hardware*.

Analisando estas equipas é possível retirar algumas conclusões. A primeira conclusão é que grande parte das equipas utiliza o mesmo método para identificar a bola, utilizando a deteção da cor da mesma. Outra tecnologia que todas estas equipas demonstram no seu guarda-redes é a estrutura móvel, sendo ligeiramente diferente de equipa para equipa, demonstrando-se um desenvolvimento importante e necessário para suporte ao robô.

Uma atualização bastante interessante e que apenas uma equipa utiliza, é a defesa de penáltis desenvolvida pelos *Tech United*, sendo que pode ser uma mais valia.

Para concluir, as equipas utilizam métodos bastante semelhantes no desenvolvimento do guarda-redes sendo que têm umas pequenas alterações dentro de cada método, fazendo a diferença para melhores resultados.

Capítulo 3

Robôs da equipa LAR@MSL

LAR@MSL é o novo nome da equipa da *MinhoTeam* que participa na liga *MSL*. Conforme já mencionado, a equipa *MinhoTeam* compete em várias competições, tanto nacionais como internacionais, sendo uma delas o *RoboCup*. Como uma das primeiras equipas a surgir nesta prova, contribui ao longo dos anos para a evolução da mesma nas várias ligas que participa e participou. A equipa já participou várias vezes na liga *MSL* com o objetivo de acompanhar a evolução tecnológica e jogar ao mais alto nível da liga, visto que desde a primeira vez que participou, a equipa já desenvolveu várias versões de robôs.

A última alteração na estrutura dos robôs foi em 2016, sendo atualizada para uma estrutura mecânica mais estável. Mesmo assim devido à necessidade de evolução, os robôs tiveram de receber algumas alterações mecânicas. A nível de *hardware* foi necessário algumas reformulações, pois o existente era de difícil evolução, sendo que todos os processos de alterações mecânicas e de *hardware* foram desenvolvidos no *LAR* da Universidade do Minho. O *software* e a sua arquitetura também foram desenvolvidos do zero, de forma a estar mais atualizado com as especificações atuais, tornando assim mais fácil para a continuação do projeto.

Neste capítulo são apresentadas as alterações da estrutura mecânica e a nova arquitetura de *hardware*, assim como o que foi reutilizado da plataforma.

Por fim é feita uma análise da arquitetura de *software* implementada nos robôs, como o código de baixo nível, a comunicação e localização dos robôs, as ferramentas utilizadas e ainda o funcionamento do guarda-redes como atacante.

3.1 Estrutura mecânica

Na *MSL* existem várias regras sobre os robôs, desde dimensão, peso e até cor. No entanto, desde que esteja dentro dessas regras estipuladas, o robô pode ter qualquer formato para a estrutura mecânica. Desta forma, ao longo das competições é possível ver equipas com várias formas pelas suas desvantagens

e vantagens, assim como para estratégia de jogo. Esta equipa utiliza uma plataforma de base circular com 50 cm de diâmetro (constituída por 3 chapas), que desta forma cumprem o limite máximo de 52 cm × 52 cm x 80 cm determinado pelas regras. Na Figura 12 está representada a chapa superior, do meio e a inferior de cada robô respetivamente.

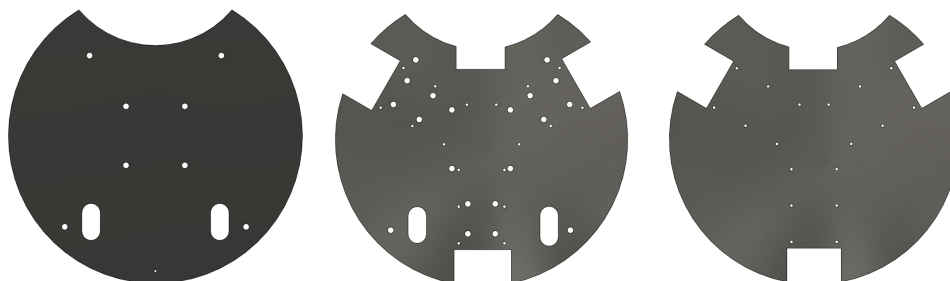


Figura 12: Chapas dos robôs

Na chapa do meio, na parte de baixo, ficam os motores dos robôs, sendo que cada robô é constituído por três motores, um em cada roda. Na parte de cima desta chapa encontra-se o chuto eletromagnético. Na chapa superior encontra-se o sistema de manipulação de bola ou *dribblers*.

O robô tem 4 ferros que funcionam como uma torre de forma a atingir o limite máximo de altura de 80 cm, onde é instalada a câmara omnidirecional. Para estabilidade desta torre, estão fixadas quatro barras de aço que unem a chapa superior com a torre. Esta torre, assim como as barras de estabilidade são possíveis de observar na Figura 13. Nesta Figura também se observa em cima o sistema de visão omnidirecional, com uma câmara virada para um espelho.



Figura 13: Robô futebolista

3.1.1 Sistema de manipulação de bola (dribblers)

O sistema de manipulação de bola tem como objetivo manter a bola na posse do robô, mesmo este estando em movimento, sendo obrigado a manter um movimento natural da bola. O mecanismo utilizado, é um mecanismo idêntico ao de outras equipas, consistindo em dois braços de controlo com um motor cada, com uma roda aparafusada de forma a receber a bola.

Para manter os motores fixos à plataforma foi utilizado um tubo assim como uma pequena chapa de ferro (Figura 14a), que foram cortados de forma a prender o motor ao tubo. Este tubo foi fixado na peça de ferro que está aparafusada na chapa, por meio de um ferro cilíndrico, permitindo a liberdade de movimento dos *dribblers* para cima e para baixo, facilitando a entrada da bola nos mesmos (Figura 14b).



(a)



(b)

Figura 14: a) Peças dribblers. b) Motores dribblers.

Por fim, nos *dribblers*, são adicionadas as rodas que prendem a bola nos mesmos, permitindo um movimento natural da bola, como se pode ver na Figura 15 (estas rodas foram desenvolvidas por outro projeto da equipa).



Figura 15: Dribblers dos robôs

3.1.2 Compartimento do computador

Devido a alterações de *hardware* o robô passou a ser controlado por um computador portátil, sendo necessária a criação de um compartimento para proteção do mesmo. Portanto, foi criada uma caixa em ferro no meio do robô, aparafusada às quatro barras de aço por meio de umas chapas de ferro dos dois lados em forma de triângulo, não permitindo que a caixa perca estabilidade e se comece a deformar (Figura 16).



Figura 16: Caixa do computador

De forma a que o computador não fique em contacto com o ferro, é usada uma esponja que protege o computador e não permite que o mesmo se arranhe ou movimente pela caixa. Para dissipar o calor gerado pelo computador, foram feitas aberturas na esponja, que permitem a libertação de ar (Figura 17).



Figura 17: Esponja da caixa do computador

3.1.3 Estrutura do guarda-redes

Devido a esta alteração do compartimento do motor a estrutura do guarda-redes, usada para aumentar a área de defesa, teve de ser alterada. Em primeiro lugar, como a estrutura era composta por peças separadas e aparafusadas, criando mais instabilidade, foi decidido soldar todas as peças para se tornar uma peça única como se vê na Figura 18a. Em seguida, a estrutura foi movida mais para a frente para não ficar sobreposta ao compartimento do computador, criando assim mais proteção para o mesmo, como se pode ver na Figura 18b.



(a)



(b)

Figura 18: a) Estrutura do guarda-redes. b) Estrutura no guarda-redes.

Devido a esta alteração da posição da estrutura, um novo problema surgiu. Como as dimensões máximas do robô são de 52cm x 52cm x 80cm, a diagonal da sua base tem cerca de 73,5cm, sendo esse o tamanho máximo da estrutura caso colocada ao meio. Como a mesma foi passada para a frente, o robô, com a estrutura desse tamanho, já não se encontrava dentro das medidas. Por esse motivo, foram feitos cortes ligeiros nas barras horizontais, perto das pontas, permitindo dobrar as laterais verticais um bocado para trás, colocando-se assim o robô dentro das regras (Figura 19).

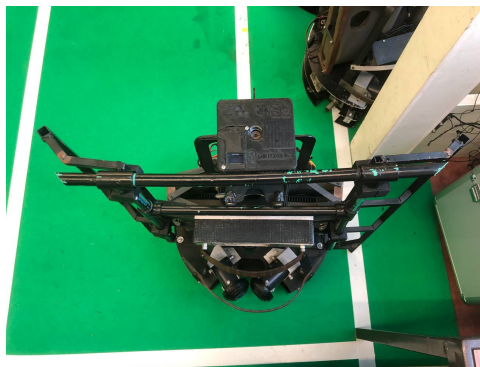


Figura 19: Estrutura do guarda-redes alterada

3.1.4 Rodas omnidirecionais

No eixo de cada caixa redutora está acoplado uma roda omnidirecional (Figura 20) que garante um movimento omnidirecional. Este tipo de movimento não apresenta qualquer restrição, podendo assim o robô mover-se em qualquer direção com uma qualquer orientação, tornando possível movimentos de translação e rotação em simultâneo.



Figura 20: Rodas omnidirecionais

Com o uso destas rodas, surgiu um problema, o desgaste dos rolos das rodas. Com o uso contínuo destas rodas, depois de algum tempo verificou-se um desgaste grave nos rolos, chegando até alguns a rasgar. Foi necessário desenvolver novos rolos em 3D (Figura 21a e 21b) para imprimir numa impressora de resina.



(a)



(b)

Figura 21: a) Modelo da peça 3D do rolo. b) Peça 3D do rolo.

A resina utilizada foi a *Resina lavável FlashForge*, devido ao seu preço/qualidade, com boa resistência ao calor e sem fissuras ou deformações. A resistência ao calor é bastante relevante devido aos robôs andarem a grandes velocidades sobre o campo.

3.2 Hardware

Um dos principais objetivos da equipa era tornar o *hardware* de todos os robôs bastante estável e fiável, de forma a permitir que em projetos futuros os elementos apenas se foquem no desenvolvimento de *software* e alterações inovadoras no *hardware* (como por exemplo chutos em parabólica).

Este capítulo descreve todo o *hardware* comum entre robôs (o guarda-redes tem umas adições que são descritas no capítulo seguinte), desde a sua arquitetura, até aos vários elementos individuais.

3.2.1 Arquitetura de hardware

No desenvolvimento desta nova arquitetura de *hardware* foi necessário tomar em conta alguns fatores, como o orçamento, estabilidade e eficácia.

Esta arquitetura é composta por cinco sistemas, sendo esses os sensores, os atuadores, a fonte de energia, o sistema de controlo e o sistema de processamento.

O sistema de processamento é composto pelas duas unidades de processamento, sendo essas o Portable Computer (PC) (unidade de processamento principal) e a placa de *hardware* com a ESP32 - SPARKFUN THING (unidade de processamento de baixo nível).

O sistema de controlo é composto por duas placas, uma responsável pelo controlo dos motores e a outra pelo controlo do chuto eletromagnético. Essas placas vão ser abordadas na secção dos atuadores no seu respetivo atuador.

Os sensores são constituídos por três *encoders* (um por cada motor), uma bússola e duas câmaras, a câmara *BlackFly PGE 13S2C* que é responsável pela localização dos robôs e a câmara da *Xbox 360, Microsoft Kinect*, câmara para deteções mais precisas no ambiente. No caso do guarda-redes, é ainda adicionado outro sensor para suporte da localização, o *LIDAR URG-04LX-UG01*.

Os atuadores são os motores dos *dribblers* já referenciados para manipulação da bola, os motores de movimentação dos robôs, o chuto eletromagnético e ainda um pequeno ecrã colocado no topo do robô para disposição de algumas informações.

A fonte de energia é uma *Power Box* desenvolvida por outra dissertação da equipa, onde a mesma é alimentada por umas baterias desenvolvidas no LAR.

A Figura 22 representa uma visão geral do sistema onde se pode perceber como é que todos se relacionam entre si.

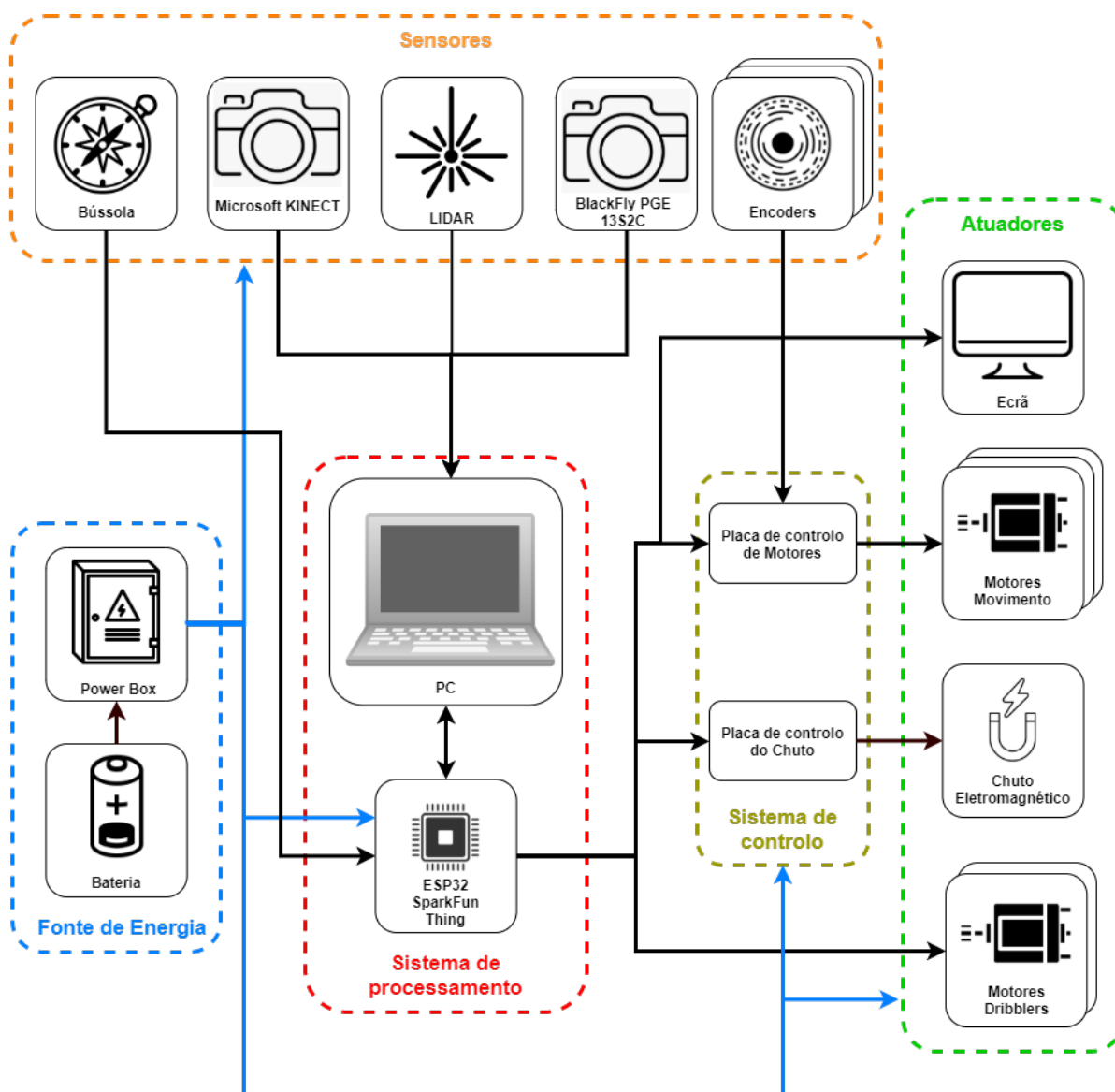


Figura 22: Arquitetura de hardware

Todos os sistemas são abordados mais aprofundadamente nas secções seguintes.

3.2.2 Sistema de processamento

Nesta secção são descritas as duas unidades de processamento, a unidade de processamento principal e a unidade de processamento de baixo nível.

3.2.2.1 Unidade de processamento principal

A principal alteração no *hardware* foi a unidade de processamento principal, pois esta passou a ser um PC da *Lenovo*, *LEGION 5* (Figura 23), para cada um dos robôs. Este PC tem um processador *AMD*

Ryzen 5 5600H com 8 GigaByte (GB) de Random Access Memory (RAM), 512 GB de memória Solid-State Drive (SSD), com uma GPU da NVIDIA, *GeForce RTX 3060*. Um aspecto importante deste PC é o seu peso (2,4 kg), não sendo desejado um peso elevado, pois existe um limite de peso nestes robôs. O sistema operativo instalado neste PC foi o *Linux*.



Figura 23: Lenovo LEGION 5

Este PC é responsável pelo processamento dos algoritmos de localização (visão por computador e fusão sensorial no caso do guarda-redes), dos algoritmos das habilidades dos robôs (mover, chutar, etc), assim como os algoritmos de previsão de trajetória e posicionamento do guarda-redes. As duas câmaras e o *LIDAR* estão diretamente ligados ao PC, sendo estes os sensores mais relevantes nestes algoritmos. Este também é responsável pela comunicação com a rede externa, de forma a enviar e a receber informação da *BaseStation* e de qualquer outro robô da equipa. É também no PC que todas as redes neuronais estão em execução, criando assim um peso acrescido nesta unidade.

O principal motivo desta alteração deve-se ao uso das redes neuronais, pois a unidade de processamento anterior (mini-PC *MSI Cubi* com o processador *Intel Pentium 3805U*) não tinha capacidade de aguentar tantos algoritmos ao mesmo tempo.

3.2.2.2 Unidade de processamento de baixo nível

Como descrito na arquitetura de *hardware* a unidade de processamento principal vai buscar informação diretamente aos sensores (exceto a bússola), mas não a envia diretamente aos atuadores. Estes comandos de ação de alto nível precisam de ser enviados para um sistema de baixo nível, sendo esse a unidade de processamento de baixo nível. Esta unidade está ligada diretamente à unidade de processamento principal e é responsável por receber os comandos de ação da unidade principal, sendo

necessário interpretá-los e atuar da forma desejada, controlando diretamente alguns dos atuadores ou enviando informação para o sistema de controlo.

Esta unidade obtém informações da percentagem das baterias através do Analog-to-Digital Converter (ADC), e envia essa informação, assim como a informação obtida da bússola, para a unidade de processamento principal.

O microcontrolador utilizado nesta unidade foi a *ESP32 - SPARKFUN THING*, tendo sido desenvolvida uma placa (Figura 24) de forma a permitir que este consiga comunicar com todos os sensores, atuadores e com o sistema de controlo.

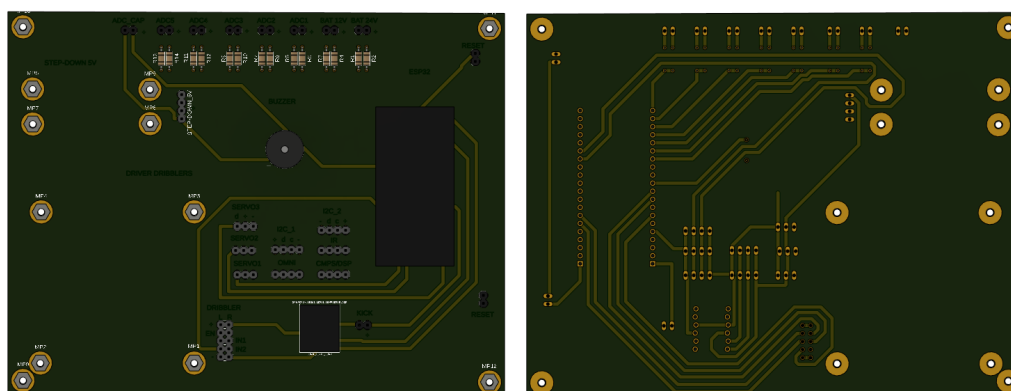


Figura 24: Modelo 3D da placa baixo nível

De forma a tornar o desenvolvimento da placa mais rápido e económico, os processos de pintura, furação e soldagem dos componentes foram feitos no LAR pela equipa. Na Figura 25 está representado o resultado final da placa sem componentes.

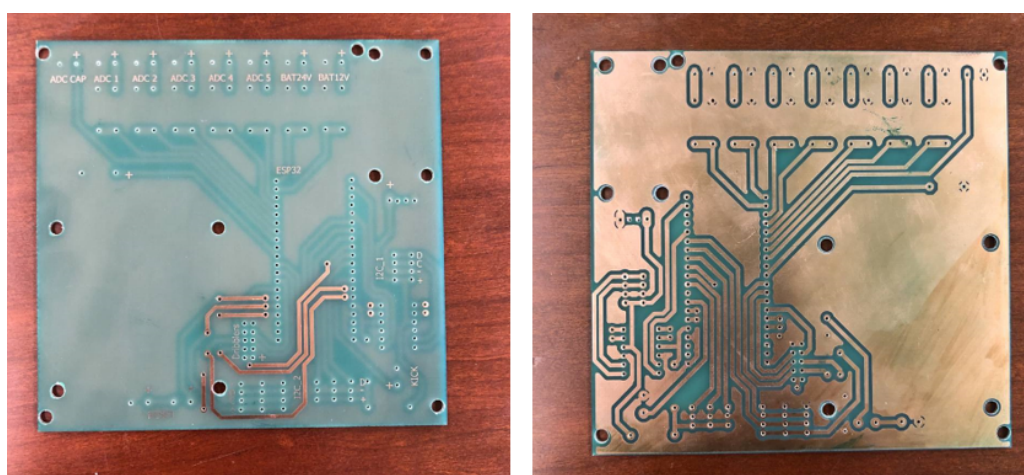


Figura 25: Placa baixo nível sem componentes

De forma a proteger a placa e arranjar um método de fixá-la no robô, foi desenvolvida uma caixa (Figura 26) e suportes (Figura 27).

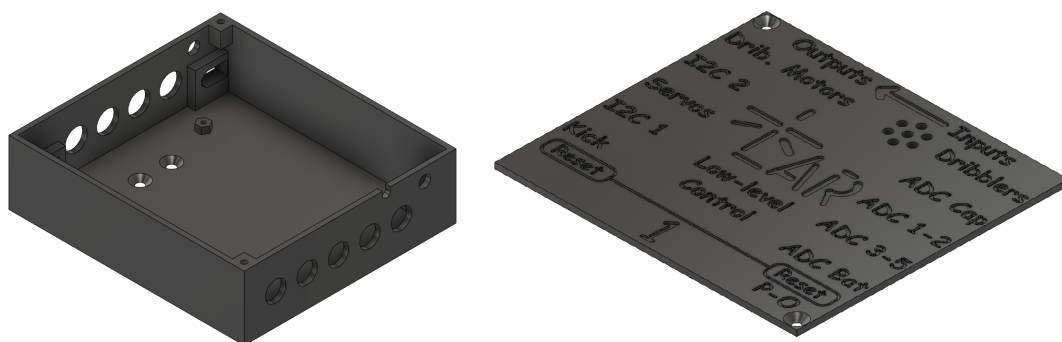


Figura 26: Caixa da placa de baixo nível

Os suportes (Figura 27) são aparafusados na parte de trás da caixa, e em seguida a caixa é colocada na torre do robô.



Figura 27: Suportes da caixa da placa de baixo nível

Na Figura 28a é possível ver o resultado final da placa com os componentes, assim como a disposição da mesma na caixa. Na Figura 28b é possível ver os suportes na caixa da placa.



(a)



(b)

Figura 28: a) Resultado final da placa de baixo nível. b) Suportes na caixa da placa de baixo nível.

Por fim, na Figura 29 está demonstrado o resultado final da unidade de processamento de baixo nível colocada no robô.

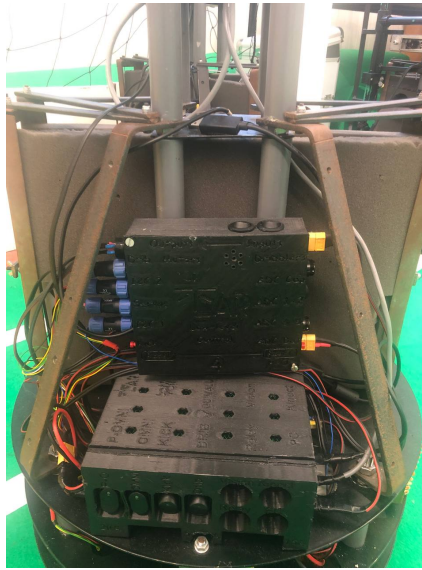


Figura 29: Unidade de processamento de baixo nível no robô

3.2.3 Sensores

Nestes robôs estão presentes em cada um três *encoders HEDL 5540 Maxon Series*, devido a este conter três motores.

Outro sensor utilizado é a câmera *BlackFly PGE 13S2C* (Figura 30a) utilizada no módulo de visão omnidirecional. A localização do robô que tem como base esse módulo, utiliza ainda outro sensor, sendo esse a bússola *CMPS14*. No guarda-redes é utilizado outro sensor de suporte à localização, sendo esse o *LIDAR URG-04LX-UG01* (Figura 30b). Os métodos de localização que utilizam estes sensores são abordados no subcapítulo de *software*.



(a)



(b)

Figura 30: a) BlackFly PGE 13S2C. b) LIDAR URG-04LX-UG01.

Um dos sensores mais importantes no desenvolvimento desta dissertação é a outra câmara utilizada, a *Microsoft Kinect* (Figura 31), pois é esta câmara que todos os algoritmos do guarda-redes usam, à parte da localização.



Figura 31: Microsoft Kinect

Esta é uma câmara *RGBD* que contém uma câmara com sensor no comprimento de onda infravermelho para se obter a profundidade e uma câmara *RGB*. A *Kinect*, na imagem colorida, tem uma resolução de 640x480 pixels e na imagem de profundidade, tem uma resolução de 320x240 pixels. O campo de visão desta câmara é de 57x43 graus.

3.2.4 Atuadores e sistema de controlo

Nesta secção são abordados todos os atuadores referidos exceto os motores dos *dribblers*, pois esses já foram referidos na secção 3.1.1. São também abordados os sistemas de controlo dos motores de movimento e do chuto eletromagnético no respetivo atuador.

3.2.4.1 Motores de locomoção

Cada robô tem na sua base três motores dedicados à locomoção. São motores *Maxon* de 150W e 24V Direct Current (DC), com os *encoders* acoplados a cada um. Os três motores estão afixados à base com um desfasamento de 120° entre si e a mesma distância ao centro da base (Figura 32).

No eixo de cada caixa redutora, está acoplada uma roda omnidirecional (Figura 20), fazendo com que o movimento do mesmo não apresente qualquer tipo de restrição. Deste modo o robô é capaz de se mover em qualquer direção em qualquer orientação, sendo mesmo possível movimentos de translação e rotação em simultâneo.

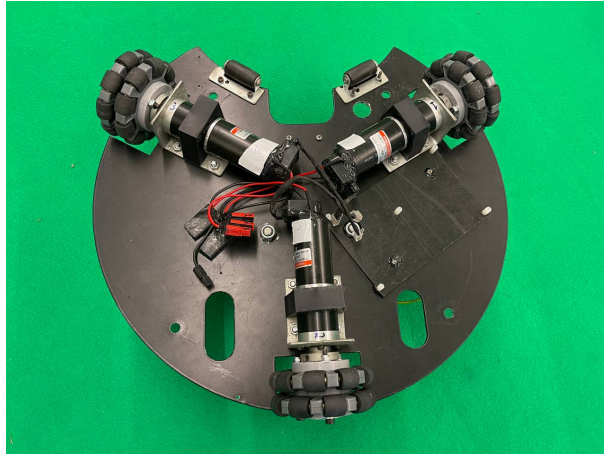


Figura 32: Motores na base do robô

De forma a controlar estes três motores, é utilizada a placa *OMNI3D-MAX* (Figura 33) desenvolvida pela empresa *SAR* (Soluções de Automação e Robótica). Esta placa abstrai o procedimento de controlo individual destes três motores à unidade de processamento de baixo nível, pois é capaz de controlar de forma independente três motores *DC* de 24V até 50A de corrente por motor. Assim, a unidade de processamento de baixo nível apenas precisa de indicar o tipo de movimento desejado para a *OMNI3D-MAX* calcular os valores a aplicar a cada motor individualmente.



Figura 33: OMNI3D-MAX [10]

Para simplificar a interação do utilizador com a placa *OMNI3D-MAX*, é disponibilizada uma biblioteca *open-source* para programação em *C* e direcionada para o *Arduino IDE*. Esta biblioteca contém funções de configuração, de leitura e de movimentação.

Esta placa contém vários modos de movimentação, sendo o mais relevante a movimentação omnidirecional, linear e posicional de três motores com ou sem controlo PID. Um modo que a placa também oferece é o modo de calibração. Este modo coloca os motores a rodar de forma a perceber qual a força

mínima necessária para os motores arrancarem com o peso a que estão sujeitos.

Nos robôs é implementado o modo de movimentação omnidirecional de três motores com controle PID. Deste modo o controle é em malha fechada com esse controlador, sendo necessária a leitura do valor da contagem incremental dos *encoders* dos três motores. Na biblioteca disponível existe uma função que recebe três parâmetros. O primeiro e segundo parâmetro são recebidos em percentagem e equivalem à velocidade linear e angular respetivamente. O terceiro parâmetro equivale à direção do movimento de translação que é recebido em graus, equivalente ao ângulo entre a frente do robô e a direção desejada.

Sendo assim, caso seja desejado que o robô vá com velocidade máxima para a sua esquerda enquanto o mesmo tem uma velocidade angular de 10%, é enviado para esta placa o comando de movimentação com os valores de 100, 10 e 90 respetivamente. Todo o processo de cálculo para obter os valores necessários a aplicar a cada motor é feito por esta placa de controle.

De forma a obter o movimento desejado existem dois componentes importantes a serem calibrados, o PID e a rampa de aceleração [28].

A rampa de aceleração é configurada através do parâmetro *slope* e do parâmetro *KI*, sendo este o "fator limiar", um fator dedicado à potência que é necessário aplicar aos motores para os colocar no limiar do movimento. Uma rampa de aceleração bem ajustada permite um arranque eficaz, suave e com deslizamento mínimo.

3.2.4.2 Chuto eletromagnético

O chuto eletromagnético, é constituído por uma bobina eletromagnética, uma bateria de condensadores com um total de $4,3\text{ mF}$ e 441 V de tensão máxima, uma placa controladora e duas bobinas com $3,8\text{ mH}$ de indutância conjunta, usadas para recarregar a bateria de condensadores (Figura 34).



Figura 34: Componentes chuto eletromagnético

À esquerda na Figura 34 é possível observar a bobina eletromagnética [29]. Como se verifica nessa figura, existe uma abertura no meio da mesma. Dentro desta abertura é colocada uma haste constituída por ferro e *nylon*, onde cada um ocupa exatamente metade da haste separadamente. A parte de trás dessa haste é apertada a um elástico que obriga a mesma a manter o *nylon* dentro da bobina. Percorrendo corrente elétrica por essa bobina, é gerado um campo eletromagnético que impulsiona a haste para o lado do *nylon*, o que faz impulsionar um objeto que esteja à frente da mesma, neste caso a bola. Como a haste está apertada ao elástico, quando o campo eletromagnético desaparece, o elástico coloca a mesma na posição inicial, pronta para outro remate.

A força aplicada pela haste na bola é proporcional à quantidade de energia que percorre a bobina. Esta quantidade de energia, armazenada na bateria de condensadores, é controlada por uma placa de controlo desenvolvida por um ex-elemento da equipa (placa à direita na Figura 34), que contém circuitos de controlo e de potência. Assim, com esta placa é possível controlar a duração da descarga de energia para a bobina.

3.2.4.3 Ecrã

Um dos atuadores adicionados foi um *Display OLED* com 0,98 polegadas com comunicação Inter-Integrated Circuit (I2C). Este ecrã situa-se na parte de cima do robô com o intuito de fornecer algumas informações à equipa, desde percentagem das baterias, carga dos condensadores do chuto, posição do robô no campo e até mesmo os parâmetros enviados para a *OMNI3D-MAX*. Existe ainda um quadrado no ecrã, que indica se a unidade de processamento de baixo nível está a funcionar corretamente (se o quadrado estiver a piscar é porque está a funcionar como desejado). Na Figura 35 é possível ver este ecrã com a posição dessas informações, sendo que algumas estão no valor por defeito devido ao robô não estar com tudo ligado.



Figura 35: Ecrã robô

3.2.5 Fonte de alimentação

A fonte de alimentação deste robô é uma caixa de potência que é alimentada pela bateria do robô. Todo o circuito de potência desta caixa, assim como a caixa em si, foram realizados por outra dissertação da equipa, sendo uma das peças fundamentais de todo o projeto.



Figura 36: Caixa de potência

Em relação às baterias, foram desenvolvidas 14 baterias, duas para cada robô. Cada bateria tem uma capacidade de 22,2V e 6700mAh. Nestas baterias são usadas pilhas *LI-ION 18650 3,7V 3350mAh*, o que faz um sistema de dois *packs* de pilhas em paralelo, sendo que cada *pack* é constituído por seis pilhas em série.

Para fazer estas baterias foi usada uma máquina de solda por ponto (Figura 37a) de forma a ligar as células (Figura 37b) para criar cada um dos *packs*.



(a)



(b)

Figura 37: a) Máquina de solda por pontos. b) Células conectadas.

Após ligar todas as células foi necessário soldar os cabos de alimentação e de balanceamento, assim como aplicar uma manga retrátil para proteção das mesmas (Figura 38).

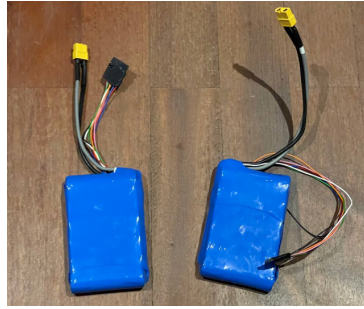


Figura 38: *Packs* baterias

Cada bateria necessita de dois *packs*, sendo necessário criar uma caixa, tanto para proteção, como para facilidade de transporte. Devido à necessidade da criação desta caixa, foi ainda instalado um *display* que mostra informação sobre a carga da bateria.

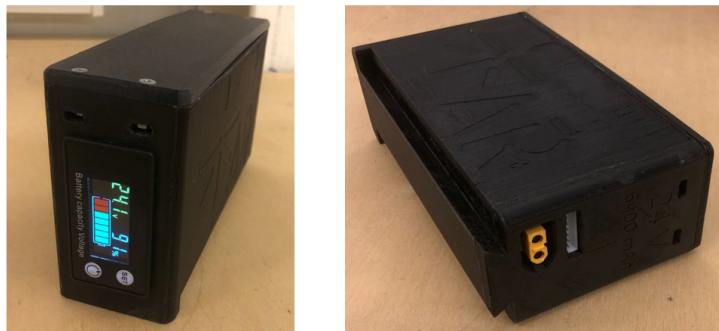


Figura 39: Caixa bateria

Esta caixa tem também uma característica bastante interessante, que é uma espécie de carril (como é possível ver na Figura 39 à direita). Assim, colocar e tirar baterias tornou-se um processo muito mais rápido, pois é só colocar a bateria no carril que está no robô (Figura 40), ligado à caixa de potência, fazendo o contacto entre a bateria e esse conector.

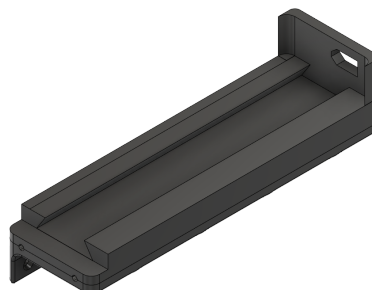


Figura 40: Carril do robô

3.3 Software

No [MSL](#) cada robô tem de jogar autonomamente e partilhar informações em tempo real.

De forma a tornar o robô autónomo, é necessário uma estratégia para os robôs saberem que funções executar em cada momento. Esta estratégia é calculada pela *BaseStation* (desenvolvida por outra dissertação da equipa), que funciona como uma espécie de treinador. Esta comunica com os robôs orientando-os para o local onde se devem deslocar e que habilidades usar (chutar, passar, receber a bola, mover para a bola, etc), sendo que estas habilidades foram desenvolvidas por outra dissertação da equipa. Estes projetos apenas se interligam com o guarda-redes quando se decide usar o mesmo como atacante, pois como guarda-redes, todo o processo de posicionamento e decisão é feito por ele mesmo como será visto mais à frente (exceto decisões de paragem de jogo em que a *BaseStation* envia a informação para o mesmo parar, como por exemplo, quando a bola vai ser repostada em jogo pós golo).

Um processo bastante importante que sem ele seria impossível de jogar e até mesmo o guarda-redes sendo independente necessita dele, é a localização dos robôs.

De forma a controlar os robôs, é também necessário controlar o *hardware* (código de baixo nível) assim como toda a comunicação entre o mesmo.

Assim sendo, este tópico explica o *software* geral dos robôs da equipa (código de baixo nível), o tipo de comunicação entre o *hardware* do robô, os métodos de localização e ainda as ferramentas utilizadas para o *software* do guarda-redes.

3.3.1 Código baixo nível

O *hardware* de baixo nível foi modificado sendo assim necessário o desenvolvimento de um novo código para possibilitar o controlo completo do *hardware*. Esse código foi desenvolvido para a *ESP32 - SPARKFUN THING* devido ao seu preço baixo, performance elevada e módulos de *Bluetooth* e *Wi-Fi* já integrados. O código foi desenvolvido em linguagem *C* no *IDE* do *Arduíno*.

Este código fica responsável por receber comandos por parte da unidade de processamento principal, decifrá-los e atuar respetivamente, atualizando os atuadores desejados com os valores calculados. Assim sendo, existe uma função dedicada a receber comandos e outra a decifrá-los, estando sempre em ciclo pronto a receber os mesmos. É também responsável por ler os valores do [ADC](#) sobre as baterias, assim como os valores da bússola e da *OMNI3D-MAX*, de forma a enviá-los para a unidade de processamento principal.

Existe uma função de *setup* para inicializar todos os módulos, como o *baudrate*, atribuição de pinos

para comunicação I2C, General Purpose Input/Output (GPIO), OMNI3D-MAX, etc. Para perceber melhor esta explicação, são apresentados dois fluxogramas deste programa (Figura 41), a função principal e a função responsável por receber os caracteres vindos da unidade de processamento principal.

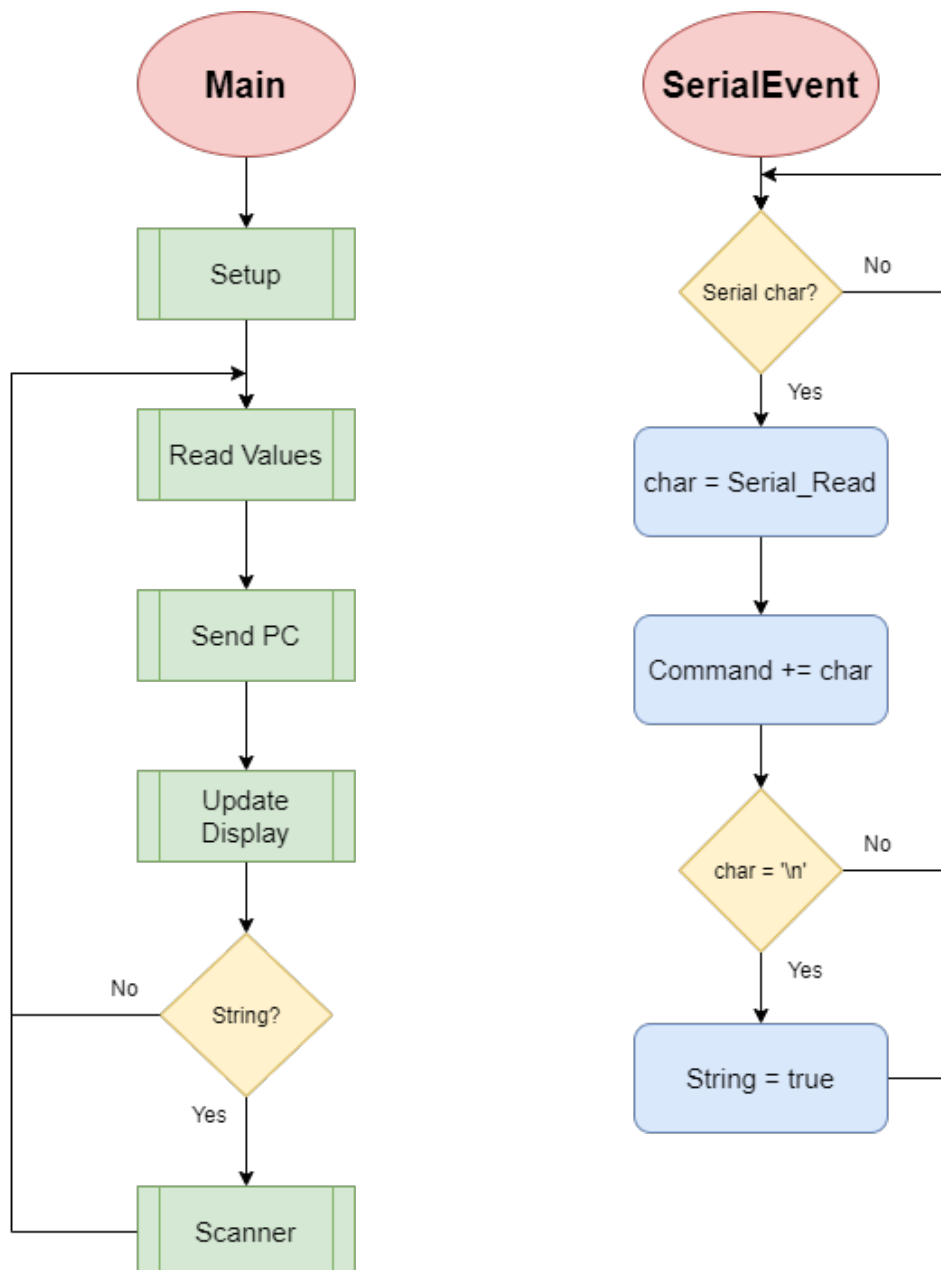


Figura 41: Fluxogramas código baixo nível

A função *Scanner* verifica o primeiro carácter do comando, e dependendo do valor desse carácter, age em conformidade. Por exemplo, para atribuir uma velocidade aos motores é utilizado o carácter *M*, em que é necessário três valores, velocidade linear, rotacional e direção separados por vírgulas. Então o comando recebido pode ser, por exemplo, "*M,80,10,270*", que resulta num movimento com velocidade linear de 80%, rotacional de 10% na direção dos 270° do robô. Caso o *Scanner* não receba esses três

valores, o comando é ignorado.

Para comunicar com a *OMNI3D-MAX* existem algumas funções da biblioteca que permitem atualizar os valores tanto da rampa como do PID, assim como enviar a velocidade linear, rotação e direção desejadas para o robô, sendo que é a *OMNI3D-MAX* que calcula através desses valores a velocidade desejada para cada motor.

3.3.2 Comunicação entre hardware

Uma informação bastante relevante do *software* é o meio de comunicação. No caso destes robôs existe a comunicação entre esses e a *BaseStation*, sendo esta desenvolvida por outro projeto, onde foi usado o protocolo de *WebSocket*, que é baseado no protocolo Transmission Control Protocol ([TCP](#)).

Outras comunicações bastante relevantes, sendo essas o foco principal desta secção, é a comunicação entre os módulos de *hardware* do robô (Figura 42).

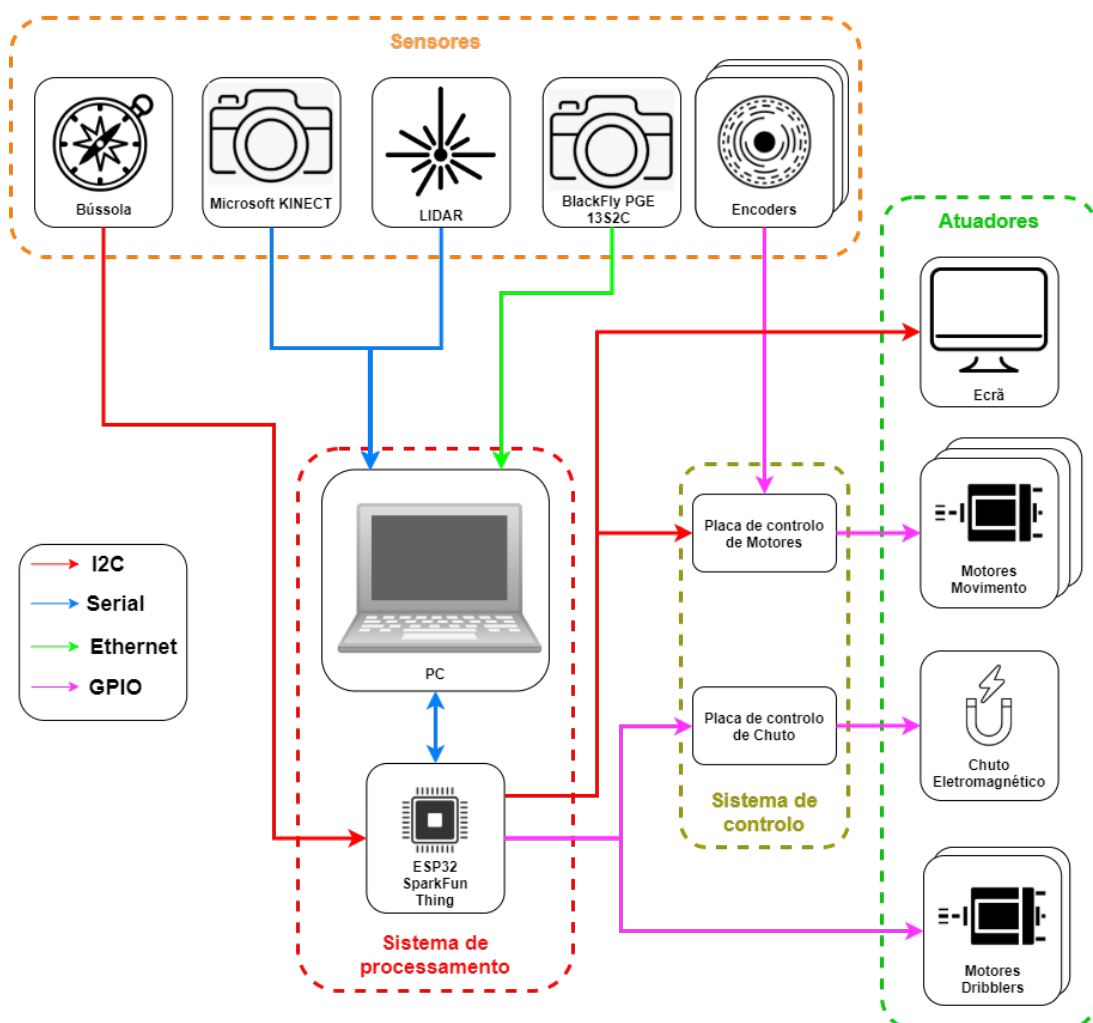


Figura 42: Diagrama de comunicação

Um protocolo de comunicação bastante usado nestes robôs é o protocolo [I2C](#), sendo um protocolo utilizado para ligar periféricos de baixa velocidade a uma placa principal. A principal vantagem deste protocolo é o facto de conter um barramento compartilhado, permitindo que se possa adicionar dispositivos de forma independente. Este protocolo foi usado para as comunicações entre a *ESP32 - SPARKFUN THING*, a bússola, o ecrã e ainda a placa de controlo dos motores.

A comunicação da *ESP32 - SPARKFUN THING* para a placa de controlo do chute e motores dos *dribblers* é feita através da ativação de certos pinos com uso de Pulse Width Modulation ([PWM](#)), sendo a comunicação através do [GPIO](#) da placa. A comunicação entre a *Microsoft Kinect*, o *LIDAR* e a *ESP32 - SPARKFUN THING* com o [PC](#), é feita por *Serial Communication*. Por fim a comunicação entre a câmara *BlackFly PGE 13S2C* e o [PC](#) é feita por *Ethernet (GigE Vision)*.

3.3.3 Localização guarda-redes

A localização de um robô no espaço de jogo é talvez uma das tarefas mais importantes, uma vez que a falha neste processo leva à atribuição de informação errada sobre todo o mundo que o rodeia [\[30\]](#). Para a localização destes robôs é utilizada a câmara *BlackFly PGE 13S2C* orientada para um espelho catadióptrico, permitindo obter uma imagem dos 360° do robô. Este método de visão é usado por quase todas as equipas da [MSL](#) e foi desenvolvido por um dos projetos da equipa.

Resumindo o método de localização, após a obtenção da imagem à volta do robô, são aplicados filtros para encontrar as linhas brancas do campo e com uso de um *Template Matching* conseguir perceber em que zona do campo se encontra este robô. O funcionamento do *Template Matching* pode ser resumido a sobreposição da imagem filtrada com uma imagem do campo de futebol (Figura [43](#)) com as dimensões corretas, verificando-se em que posição tem mais pixels coincidentes.

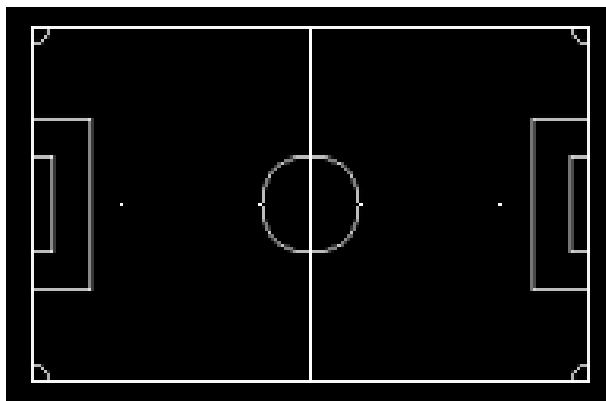


Figura 43: Template localização

Na Figura 44 é possível ver o resultado da localização de um robô e da bola no campo. Na esquerda da imagem também se podem ver as linhas brancas que a câmara deteta, assim como a localização antes e depois de ser filtrada.

Como se vê na imagem, a precisão não é muito elevada, sendo que também não é necessária para os robôs na frente. O mesmo já não se pode dizer em relação ao guarda-redes, pois este necessita da localização exata, pois uns centímetros de diferença podem significar um golo. Isso também se aplica para a bola, sendo esse um dos motivos de se usar a *Microsoft Kinect* para detetar a bola com mais precisão no espaço de coordenadas.



Figura 44: Localização robô

Desta forma, para complementar a localização, o guarda-redes usa um *LIDAR* que indica com precisão onde se encontra o guarda-redes na baliza, através da deteção da parte de trás da mesma e dos postes com o uso do *Template Matching*. O *LIDAR* não pode ser usado como único método de localização pois tem um processamento de informação lento, visto que a informação do robô é atualizada menos de 10 vezes por segundo.

Assim sendo, para a localização do guarda-redes é utilizado o melhor dos dois mundos, sendo a câmara o método de localização principal, pois tem um processamento de informação rápido, com o complemento do *LIDAR* que torna a localização muito mais precisa.

3.3.4 Ferramentas utilizadas

Nesta secção é abordada a linguagem de programação utilizada pela unidade de processamento principal (*Go*) e as razões da respetiva escolha. São descritas as bibliotecas utilizadas, sendo essas o *OpenCV* e o *GoCV*.

3.3.4.1 Go

Go é a linguagem de programação selecionada para o desenvolvimento deste projeto. *Go*, também chamado de *Golang*, é uma linguagem de programação de código aberto desenvolvido pela *Google*. Foi criada por Robert Griesemer, Rob Pike e Ken Thompson e lançada em 2009. *Go* foi projetada para ser eficiente, concorrente e fácil de usar, com o objetivo de abordar desafios comuns no desenvolvimento de *software* moderno [31].

A escolha desta linguagem deve-se ao facto de ter uma sintaxe simples, como o *Python*, com bibliarias de *opencv* (*GoCV*). A grande diferença entre *Go* e *Python*, sendo essa a principal razão da escolha do *Go*, é que o *Go* é uma linguagem compilada (como o C) e não interpretada (como o *Python*), resultando assim num desempenho muito mais rápido.

3.3.4.2 OpenCV e GoCV

OpenCV, que significa "Open Source Computer Vision Library", é uma biblioteca *Open Source* amplamente utilizada para processamento de imagem e visão por computador. Fornece um conjunto de funções e ferramentas que permite criar programas que podem analisar e processar imagens e vídeos [32].

O *OpenCV* é especialmente popular em áreas como visão por computador, *Machine Learning*, robótica e processamento de imagem, pois oferece suporte a uma ampla variedade de algoritmos e técnicas, incluindo deteção de objetos, reconhecimento de padrões, calibração de câmaras, filtragem de imagem e muito mais. Em suma, o *OpenCV* é uma ferramenta poderosa e versátil para lidar com tarefas de processamento de imagem e visão por computador numa variedade de campos e aplicações.

O *GoCV* é um projeto *Open Source* que fornece uma interface em *Go* (*Golang*) para a biblioteca *OpenCV*. O *GoCV* permite que se escreva código em *Go* para trabalhar com recursos avançados de processamento de imagem e visão por computador fornecidos pela biblioteca *OpenCV*. Isso significa que se podem criar aplicações de visão por computador e processamento de imagem eficientes em *Go*, aproveitando a funcionalidade e desempenho da biblioteca *OpenCV* [33].

3.3.5 Guarda-Redes como atacante

Esta característica do guarda-redes é apresentada neste capítulo por ser uma tarefa efetuada pela *BaseStation*, pois depende da estratégia utilizada.

Um dos objetivos desta dissertação é o guarda-redes ter a habilidade de funcionar como um robô atacante. Para atender a este requisito estes robôs foram desenvolvidos com o uso de certas *Habilidades* (Figura 45), sendo que as dos robôs atacantes foram desenvolvidas por outra dissertação. A Figura 45 representa as *Habilidades* existentes, sendo que a do guarda-redes corresponde a toda a implementação explicada no capítulo seguinte.

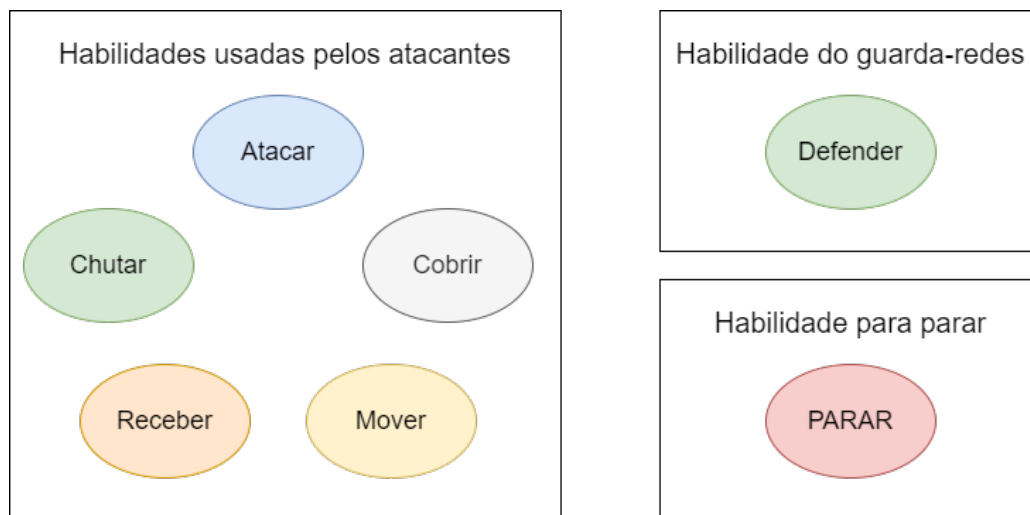


Figura 45: Habilidade dos robôs

As habilidades de cada robô são atualizadas pela *BaseStation*, sendo que para o guarda-redes funcionar como um guarda-redes é necessário colocar-lhe a habilidade *Defend*. Desta forma, caso a estratégia defina que o guarda-redes pode sair da baliza para atacar a bola, apenas tem de lhe atribuir a habilidade *Attack*. Se for atribuída a habilidade *Move*, o guarda-redes irá movimentar-se para o ponto desejado, assim como se lhe for atribuída a habilidade *Kick* o mesmo vai rematar. Da mesma forma que para o guarda-redes funcionar como atacante é apenas necessário atribuir-lhe a habilidade desejada, o mesmo funciona ao contrário, pois caso seja desejado um robô atacante funcionar como guarda-redes, apenas é necessário atribuir-lhe a habilidade *Defend*.

Capítulo 4

Implementação do Guarda-Redes

O principal objetivo desta dissertação é o desenvolvimento do guarda-redes. Este tópico descreve as implementações feitas no guarda-redes, desde o *software* desenvolvido, até à estrutura móvel com todo o seu processo de construção.

De forma a demonstrar o funcionamento do guarda-redes, está representado na Figura 46 um fluxograma que exemplifica a estrutura de processamento do guarda-redes.

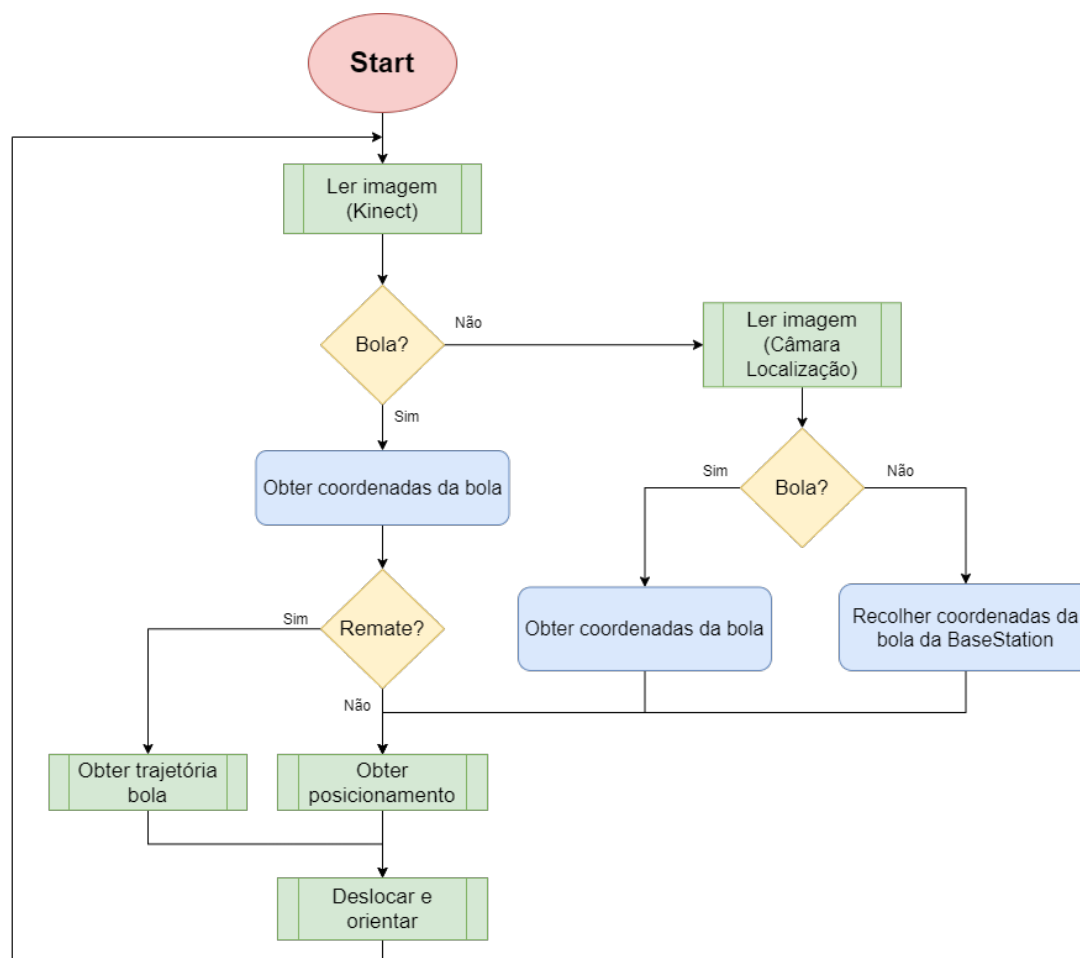


Figura 46: Estrutura de processamento do Guarda-Redes

A primeira tarefa deste robô é identificar onde se encontra a bola. Primeiro procura-se a bola com a *Microsoft Kinect* (o código continua como se não tivesse a ver nenhuma bola numa situação de erro), pois esta é a única com precisão suficiente para detetar remates. Caso esta não encontre nenhuma bola, será usada a câmara de localização. Se nesta situação a bola também não for encontrada, é pedida à *BaseStation* a localização da bola, pois essa recebe informação de todos os robôs, sendo que certamente algum deles conseguirá ver a bola. No caso de nenhuma bola estar a ser detetada por nenhum dos métodos, o guarda-redes coloca-se no centro da baliza.

Após detetar uma bola com a *Microsoft Kinect* o robô verifica se está a acontecer um remate, e nesse caso o guarda-redes irá calcular a trajetória da bola e definir a posição desejada como o ponto de interseção da bola na baliza.

Em caso de não haver remate ou de não ser a *Microsoft Kinect* a detetar a bola, o guarda-redes irá obter a melhor posição na baliza de forma a cobrir mais espaço em função da posição da bola.

Por fim, o guarda-redes desloca-se para a posição desejada com a orientação obtida e caso tenha sido identificado um remate, a estrutura móvel é ativada.

Esta estrutura de processamento é referente ao jogo corrido, não sendo aplicado nos penáltis. No caso dos penáltis, a *BaseStation* informa que está a decorrer um penálti e o guarda-redes entra nesse estado, que é explicado na secção [4.6](#).

Neste capítulo é abordado o sistema de visão, a deteção e defesa de remates, o posicionamento na baliza, o controlo de movimento, a estrutura móvel e ainda a defesa de penáltis.

4.1 Sistema de visão

Como já referido, o sistema de visão do guarda-redes usa a *Microsoft Kinect* como sensor principal. Neste sistema, a única ocasião em que este não é o sensor utilizado, é quando a mesma não consegue detetar nenhuma bola.

4.1.1 Drivers e bibliotecas

De forma a usar esta câmara, foi necessário instalar alguns *Drivers* e bibliotecas, tendo sido tomado como base o tutorial de instalação e biblioteca denominada *LibFreenect*. Este repositório da *OpenKinect* contém todos os *drivers* para a *Microsoft Kinect* e como instalá-los, tudo *Open Source* no *gitHub*, de forma a ser possível usar da melhor forma a câmara no *Linux*. A biblioteca disponibilizada foi tomada como base no projeto sendo feitas algumas alterações, como o método de aquisição do canal de distância da câmara.

Nesta biblioteca, de forma a obter a distância, era necessário usar uma função da mesma, para além de a complementar com algumas transformações e ciclos de programação, permitindo percorrer esse canal. Deste modo, foi criada uma função dentro da biblioteca permitindo que apenas seja necessário enviar a informação de qual o pixel desejado para saber a distância, retornando imediatamente esse valor.

4.1.2 Detecção da bola

Uma das tarefas fundamentais deste robô é identificar onde se localiza a bola. Neste projeto foram utilizados dois métodos para detecção da bola, sendo explicados em seguida bem como o porquê da escolha de um deles.

4.1.2.1 Detecção com uso do modelo de cores HSV

O primeiro método utilizado para detetar a bola foi através do uso do modelo de cores HSV de modo a encontrar o maior número de pixels juntos com a cor da bola. A Figura 47 representa o fluxograma do funcionamento deste método.

Em primeiro lugar a imagem obtida pela câmara (RGB) é transformada para o modelo de cores HSV, por ser mais fácil calibrar as cores. Em seguida, é aplicada uma máscara que contém os limites mínimos e máximos para os valores do HSV, de forma que quando aplicada à imagem, apenas os pixels com valores dentro desses limites prevaleçam. Esta máscara define as cores a detetar, sendo necessária uma calibração para as várias cores desejadas.

Uma das maiores dificuldades deste método é a calibração da máscara. Como se calibra o HSV de forma a detetar as cores desejadas, é necessário criar certos limites, sendo preciso calibrar de modo a detetar essas cores com mais ou menos luminosidade. No entanto, é necessário ter cuidado para que não sejam detetados falsos positivos da bola com cores idênticas. Imaginando que a bola é amarela, caso esta tenha muita luminosidade ficará perto do branco, por isso, o branco não pode ser uma das cores a detetar senão muitos pontos seriam vistos como uma bola. É necessário um compromisso da gama de valores escolhida de forma a detetar grande parte do objeto desejado com muito pouco ruído.

O passo seguinte utiliza a função *FindContours* do *OpenCV* que permite obter as várias áreas de pixels da cor desejada em forma de regiões, sendo obtidos os vários contornos desses pixels que se encontrem ligados entre si. Assim, verificando com uma área mínima de pixels, encontra-se a área correspondente à bola. Esta área mínima também implica uma calibração, não podendo ser demasiado grande, pois caso contrário a bola não é encontrada, nem demasiado pequena, pois nessa situação qualquer ruído seria considerado uma bola.

No caso de nenhuma das áreas encontradas ser maior que a área mínima, a função retorna que nenhuma bola foi detetada. Caso exista mais que uma área maior que a mínima, a primeira é considerada como bola, não sendo este um problema pois existe apenas uma bola em jogo e não é permitido o uso de cor idênticas a essa.

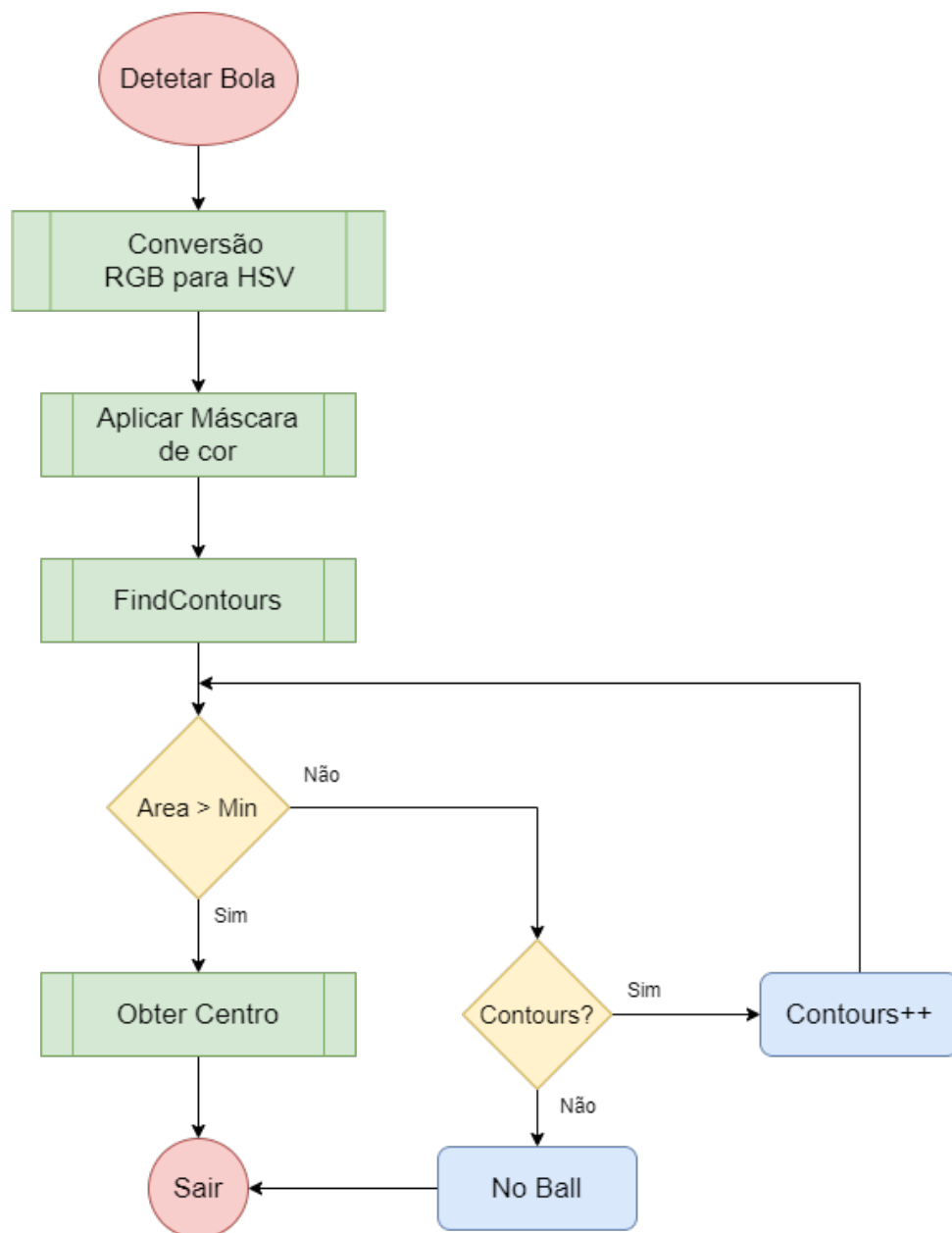


Figura 47: Detecção de Bola (HSV)

Uma desvantagem deste método prende-se com as distâncias, pois conforme a bola fica mais longe, menor será a área de pixels detetada, fazendo a bola deixar de ser detetada a partir de uma certa distância, o que se demonstrou bastante inconveniente no Festival Nacional de Robótica. Assim, a equipa decidiu partir para o uso de redes neuronais para a deteção de objetos.

4.1.2.2 Redes Neurais para detecção de objetos

A equipa decidiu utilizar redes neurais (algoritmos de *YOLO* com uso de segmentação), para detetar objetos no meio envolvente do robô, como a bola, a baliza, os robôs da equipa e os adversários.

O desenvolvimento da rede ficou a cargo de outra dissertação tendo a equipa ajudado na aquisição de dados (*dataset*). Para construir esta rede com vários objetos foram recolhidas várias fotos, onde de seguida era feita a classificação desses objetos.

Foi usado o *Label Image* para marcar os objetos com os respetivos nomes. Em cenários normais a legenda de imagens é feita com quadrados sobre os objetos, mas no âmbito deste projeto foi usada a segmentação, que se baseia no recorte exato do objeto para o identificar. Assim, os pixels identificados como objeto dizem respeito apenas a esse, enquanto que na legenda com quadrados, muitos pixels identificados no objeto dizem respeito ao cenário por trás do robô. Isto torna a rede mais robusta e permite que a identificação de objetos sobrepostos aconteça mais facilmente, pois a área de um objeto é menor que a área do quadrado envolvente do mesmo.

Na Figura 48 é possível ver essa segmentação sobre os vários objetos e à esquerda as classes (objetos) que aparecem nesta imagem.

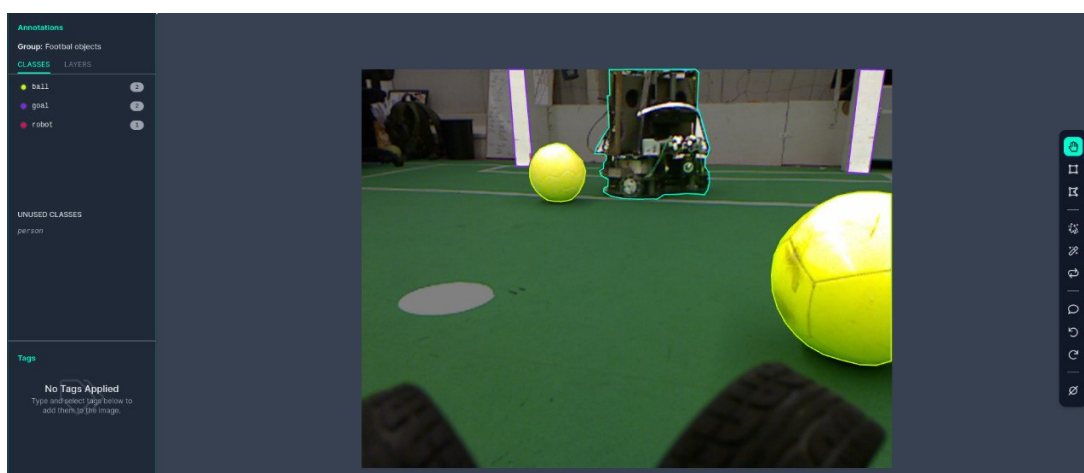


Figura 48: Segmentação de objetos

Após legendar todos os objetos em todas as imagens foi utilizado o *RoboFlow* [34] para dividir as imagens em imagens de treino, de teste e de validação, passando em seguida para o treino da rede com o novo *dataset*. Este foi o método final utilizado para deteção da bola, sendo também utilizado para deteção de robôs e da baliza.

4.1.3 Posição da bola

Após a bola ser detetada na imagem foi necessário calcular as coordenadas dessa. O cálculo da posição da bola no plano cartesiano passou por duas fases. Na primeira fase, o referencial do plano era o robô em si, sendo as coordenadas em relação ao guarda-redes. Numa segunda fase, essas coordenadas foram transformadas para ter o centro da baliza como referencial. Também se descrevem as transformações necessárias a realizar quando a informação da posição da bola vem do módulo da câmara de localização ou da *BaseStation*.

4.1.3.1 Referência no robô

Numa primeira fase é detetada a posição da bola relativamente ao robô. Para se obter esta posição, o primeiro passo é obter o ângulo da bola relativamente ao robô, como exemplificado na Figura 49.

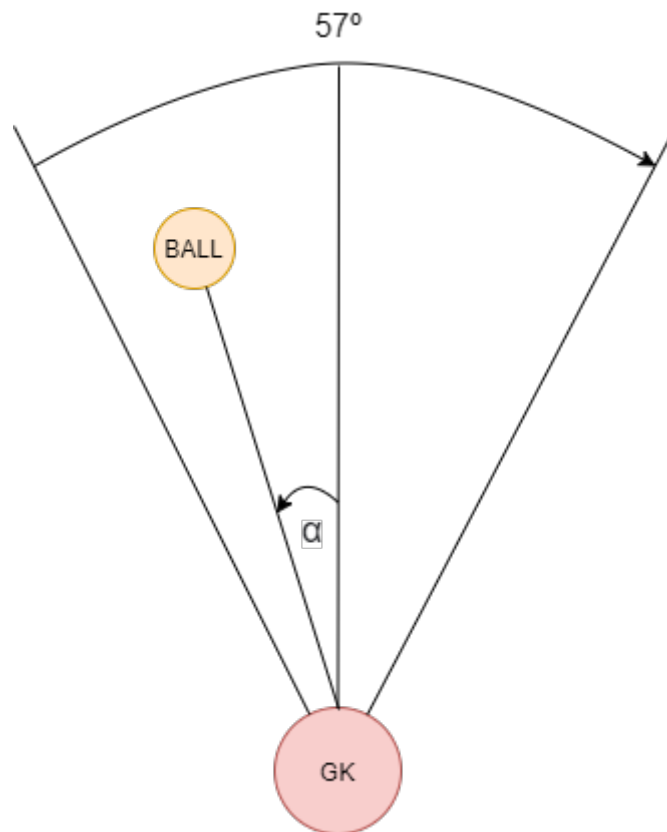


Figura 49: Ângulo da bola ao robô

Sabendo a resolução da imagem da *Microsoft Kinect* é calculada a distância da bola ao centro da imagem em pixels (Figura 50). Subtrai-se metade dessa resolução a essa posição, obtendo-se a distância representada na figura. Caso a bola se encontre na esquerda dessa metade, esse valor será negativo, definindo o lado em que a bola se encontra.

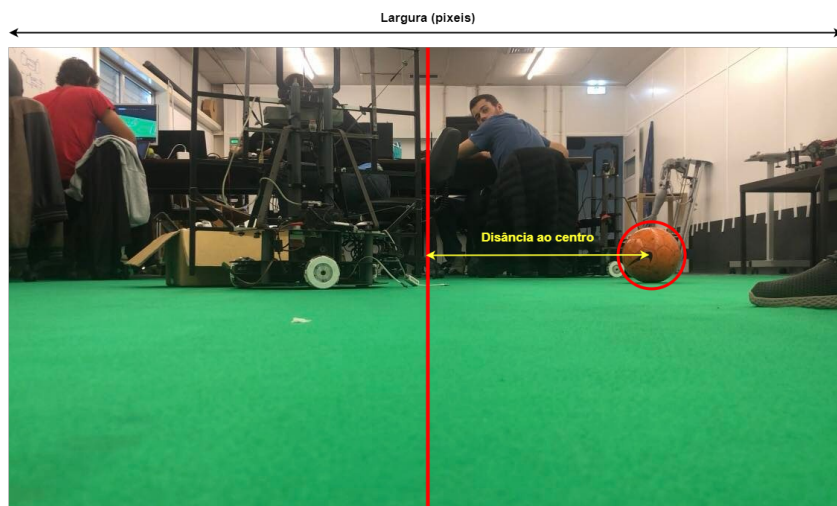


Figura 50: Distância da bola ao centro

Para se obter o ângulo é usada uma regra de três simples (equação 4.1), visto que o ângulo total ou campo de visão é conhecido (57° nesta câmara), assim como a resolução total e a distância ao meio em pixels (obtida na explicação anterior).

$$\alpha = dist_meio * \frac{57}{Largura} \quad (4.1)$$

Após calcular o ângulo é necessário utilizar a distância que a câmara indica. A obtenção desta distância trouxe um problema, pois em processo de desenvolvimento descobriu-se que a *Microsoft Kinect* tem duas versões, uma em que a distância recebida é entre a câmara e o objeto, e outra em que essa distância é entre o plano do objeto (bola) e a câmara, sendo a segunda versão a utilizada neste projeto. Essa distância está representada na Figura 51 com a letra D .

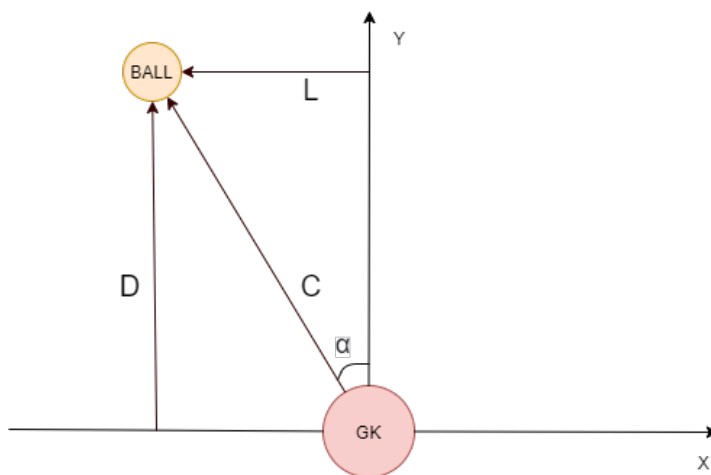


Figura 51: Posição da bola relativa ao robô

A ordenada da bola é dada pela distância retornada pela *Microsoft Kinect* e a abcissa é dada por essa distância a multiplicar pela tangente do ângulo calculado.

Como a outra versão da câmara também foi utilizada no início do projeto, a transformação também é explicada. A distância que a câmara fornecia está representada com a letra C na Figura 51, sendo a ordenada da bola calculada com a multiplicação de C com o cosseno do ângulo e a abcissa a multiplicar com o seno.

4.1.3.2 Referência no centro da baliza

Na segunda fase é calculada a posição da bola em relação ao centro da baliza. Foram necessárias algumas transformações, dando uso a partes do método anterior e reutilizando-se trabalho desenvolvido.

A troca de referencial consiste numa rotação e numa translação do robô para o centro da baliza, sendo esta mais simples uma vez que a localização já indica a posição e rotação do robô em relação ao centro da baliza.

Na Figura 52 está representado o processo de rotação do referencial do robô.

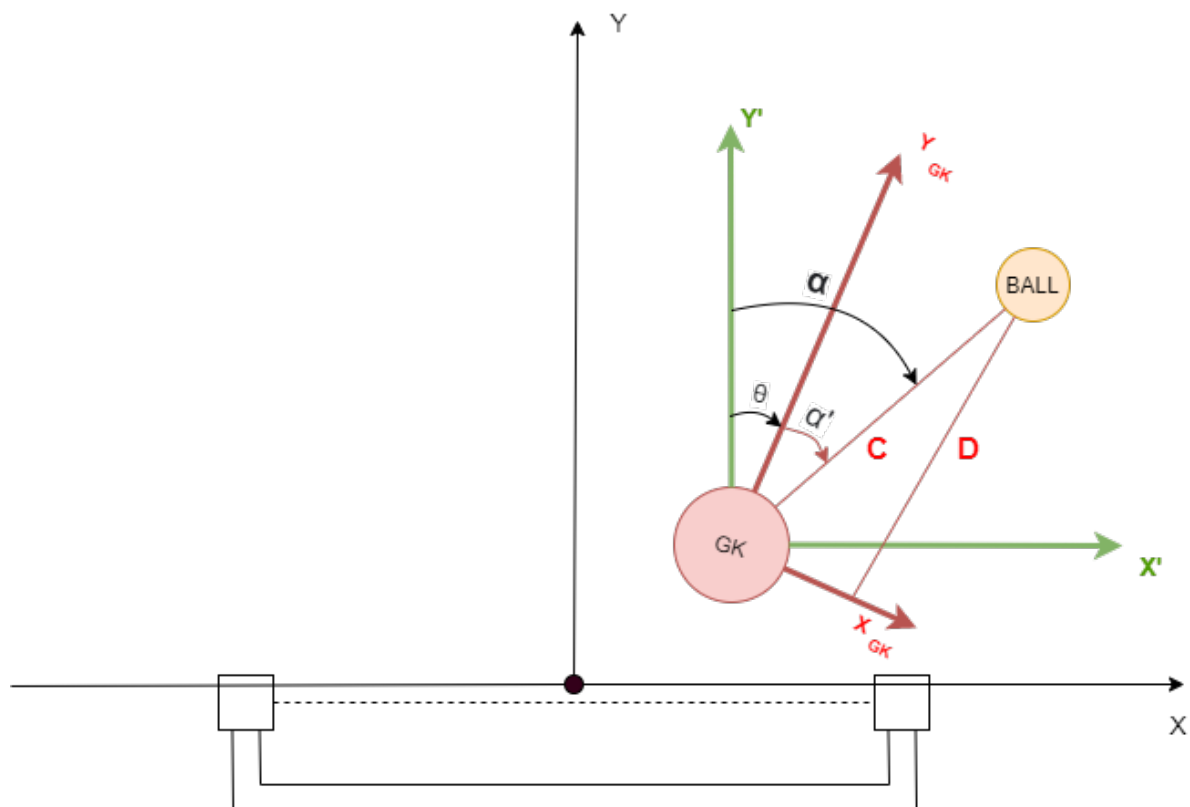


Figura 52: Rotação do referencial

Como se vê na Figura 52 a rotação tem influência no ângulo α , sendo este uma soma do ângulo do robô em relação ao centro da baliza (retornado pela localização), equivalente ao ângulo θ , com o ângulo α' , calculado como na secção 4.1.3.1 (equação 4.2).

$$\alpha = \theta + \alpha' \quad (4.2)$$

A rotação do referencial fica assim completa, sendo o novo ângulo entre a bola e o referencial no centro da baliza dado por α .

Devido ao uso desta versão da câmara a distância retornada equivale à letra D da Figura 52, sendo necessário calcular a distância representada com a letra C através da divisão de D com o cosseno de α' (equação 4.3).

$$C = \frac{D}{\cos(\alpha')} \quad (4.3)$$

Em seguida é necessário fazer uma translação do referencial obtido no passo anterior para o referencial do centro da baliza, estando este processo representado na Figura 53.

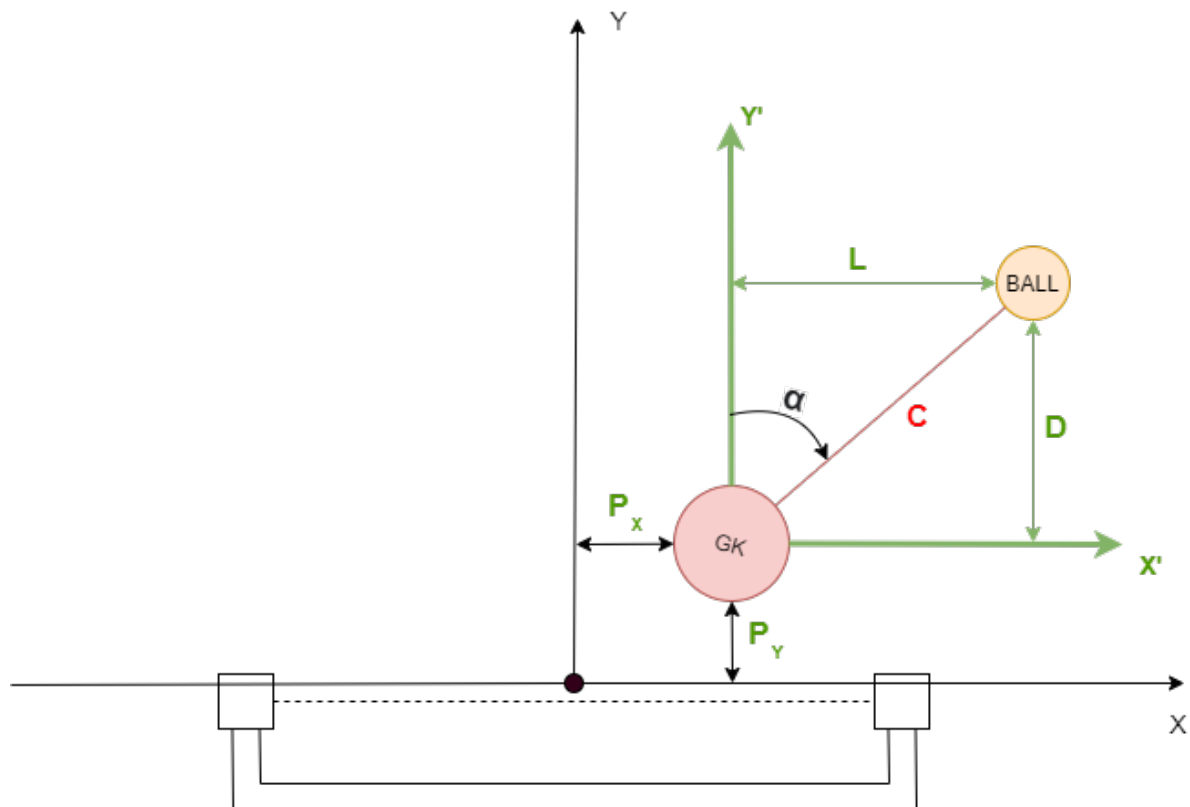


Figura 53: Translação do referencial

No primeiro passo são calculadas as coordenadas da bola no novo referencial após a rotação, sendo a abscissa (distância L) e a ordenada (distância D) do referencial $X'Y'$ representada na Figura 53. Tendo sido já calculado o ângulo α e a letra C , é utilizada a equação 4.4 de forma a obter a distância L .

$$L = C * \text{sen}(\alpha) \quad (4.4)$$

A distância D é obtida através da equação 4.5.

$$D = C * \text{cos}(\alpha) \quad (4.5)$$

Calculadas as coordenadas da bola neste novo referencial $X'Y'$, a translação equivale, como representado nas equações 4.6 e 4.7, à soma dessas coordenadas com a posição do guarda-redes em relação ao centro da baliza (P_x e P_y), sendo estas posições fornecidas pela localização do robô.

$$\text{Ball}_x = L + P_x \quad (4.6)$$

$$\text{Ball}_y = D + P_y \quad (4.7)$$

4.1.3.3 Posições externas

No caso da *Microsoft Kinect* não encontrar a bola a informação da mesma é obtida pela câmara de localização ou pela *BaseStation*. Estes dois processos utilizam um referencial com origem no centro do campo e uma orientação horizontal. Na Figura 54 está representado a vermelho o referencial utilizado pela câmara de localização e pela *BaseStation*, e a verde o referencial utilizado pelo guarda-redes.

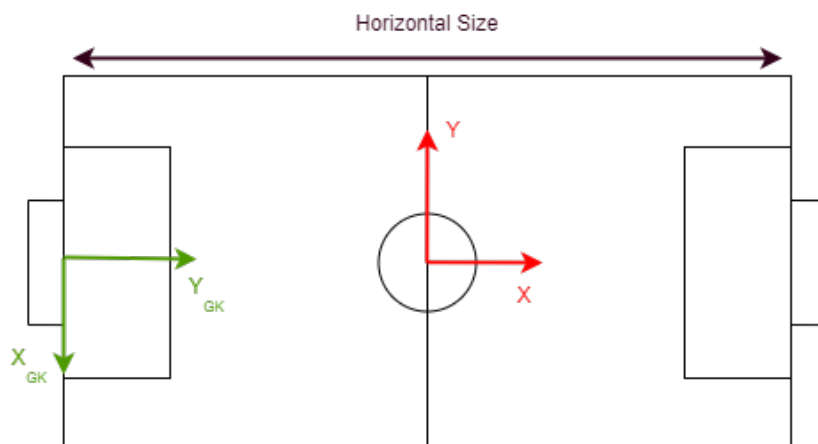


Figura 54: Referenciais da BaseStation (vermelho) e do robô (verde)

De forma a fazer uma transformação do referencial vermelho para o verde, é necessário rodar o referencial 90° e fazer uma translação de meio campo. A abscissa do referencial verde equivale ao negativo da ordenada do vermelho (equação 4.8) e a ordenada do referencial verde equivale à abscissa do vermelho somado com a distância de meio campo (equação 4.9).

$$X_{GK} = -Y \quad (4.8)$$

$$Y_{GK} = X + \frac{HorizontalSize}{2} \quad (4.9)$$

4.1.4 Filtro da posição da bola

Um problema detetado na posição da bola eram as variações da mesma, até mesmo com a bola parada. Devido à deteção do ponto central da bola não ser sempre exatamente o mesmo, havia oscilações na posição da bola, sendo que esta oscilação era proporcional à distância à bola.

Por esse motivo foi criado um filtro simples que procura combater estas pequenas oscilações, e que coloca um certo peso nas posições anteriores, fazendo depois uma média desses valores.

Considera-se que o filtro vai utilizar n valores anteriores, sendo que o índice 0 equivale ao valor mais recente e o índice n ao mais antigo sendo acrescentado o novo valor ao início do *array*. Sendo então o *array* composto por n elementos i , o peso de cada elemento será igual a $1/(1 + i)$, tornando assim o peso menor para posições mais antigas.

Como se pode ver na equação 4.10 todos os n valores são somados com o seu respetivo peso, faltando apenas fazer a média destes valores.

$$h = \sum_{i=0}^n \frac{x[i]}{1 + i} \quad (4.10)$$

De forma a fazer a média de todos estes valores é necessário somar todos os pesos que se usaram anteriormente (equação 4.11) para dividir aos valores somados (equação 4.10).

$$sum = \sum_{i=0}^n \frac{1}{1 + i} \quad (4.11)$$

O valor final filtrado com as n posições anteriores é dado pela equação 4.12.

$$filtered = \frac{h}{sum} \quad (4.12)$$

4.1.5 Detecção de remates

Após ser calculada a posição da bola o guarda-redes necessita de perceber quando existe algum remate em direção à baliza. Desta forma, o robô verifica em todos os momentos se há algum tipo de remate para atuar de acordo.

A única forma de identificar um remate é através da velocidade da bola. Visto que os robôs podem andar com a bola em direção à baliza, foi necessário levar em conta estas velocidades de forma a que o guarda-redes não identificasse remates quando existe algum jogador a deslocar-se com a bola em direção à baliza. Isto possibilita que bolas em direção à baliza, abaixo da velocidade mínima estipulada para remate, o guarda-redes não o considera, não se movimentando para fazer a defesa. Este problema foi resolvido com o posicionamento do guarda-redes na baliza.

Na Figura 55 está representado o fluxograma da função responsável por verificar remates, retornando *True* no caso positivo e *False* caso contrário.

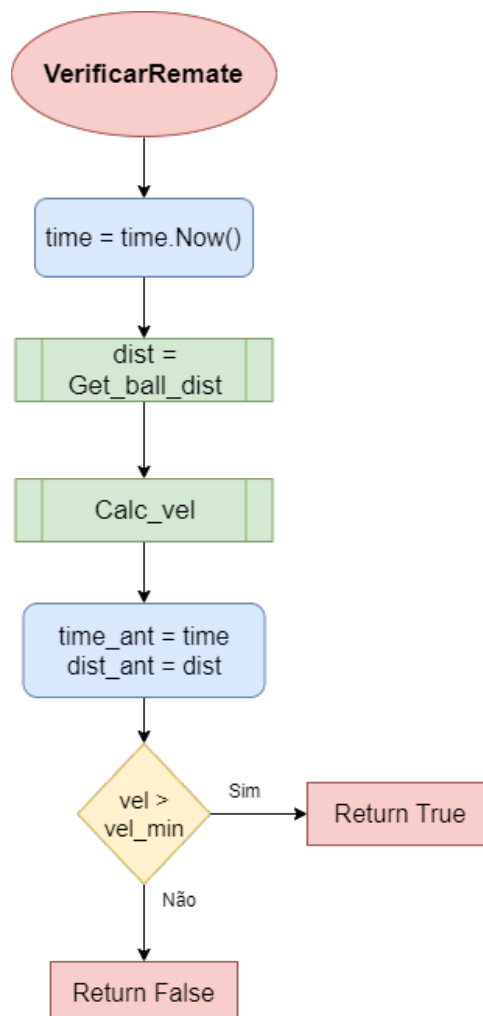


Figura 55: Verificar Remate

A velocidade da bola é calculada com base na distância a cada ciclo de processamento do código, sendo que a cada iteração é calculada a velocidade com base no tempo de ciclo e com a distância do robô à bola no ciclo atual e anterior (equação 4.13).

$$vel = \frac{dist - dist_{ant}}{time - time_{ant}} \quad (4.13)$$

Enquanto a velocidade da bola permanecer acima da velocidade mínima o robô estará sempre a detectar um remate, fazendo-o calcular a trajetória da bola repetidamente e solucionando assim um problema, bolas em arco, permitindo que o guarda-redes atualize sempre a posição em que a bola vai interceptar a linha da baliza.

4.1.6 Defesa de remates

Para o guarda-redes defender os remates foi necessário desenvolver uma previsão da trajetória da bola, sendo que sem este método a única solução seria o robô seguir a mesma, fazendo o guarda-redes raramente chegar a tempo de defender.

Foram desenvolvidos e testados dois métodos de previsão, um relativamente ao robô e outro relativamente à baliza, sendo analisada a razão do escolhido. O que todos estes métodos têm em comum é a necessidade de dois *frames* em que o remate é encontrado, pois para uma previsão são necessários no mínimo dois momentos da bola em movimento.

4.1.6.1 Previsão da trajetória - Referência ao robô

Este método usa o plano cartesiano da bola relativamente ao robô. Nos momentos em que um remate é detetado, com dois *frames*, a posição da bola desses dois é guardada, sendo possível traçar uma reta que permitirá perceber onde a bola vai interceptar o eixo do robô (Figura 56). Para determinar a função da reta são utilizados esses dois pontos.

De forma a descobrir o declive, é usada a equação 4.14:

$$m = \frac{Y_2 - Y_1}{X_2 - X_1} \quad (4.14)$$

É usada a equação 4.15 para descobrir a ordenada na origem.

$$b = Y_2 - (m * X_2) \quad (4.15)$$

A posição que a bola interceta o eixo do robô equivale ao valor das abcissas que tem como ordenada 0, dado pela equação 4.16.

$$P = -\frac{b}{m} \quad (4.16)$$

Se este valor for negativo significa que a bola vai intercetar o eixo do robô no seu lado esquerdo, definindo assim a direção que o mesmo tem de se mover.

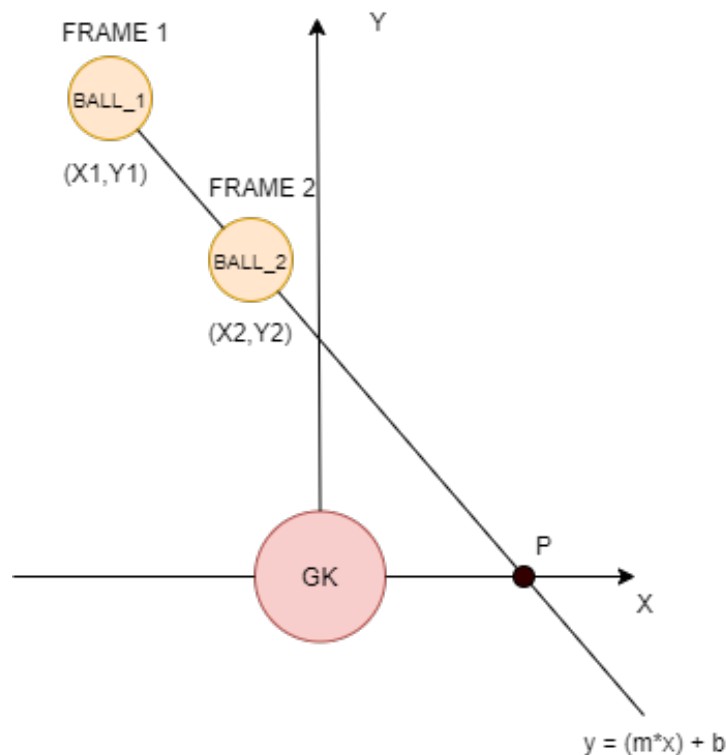


Figura 56: Previsão da trajetória - Robô

O principal foco deste método era o tempo de reação quando o robô se encontra na lateral da baliza, pois dessa forma em vez do robô se deslocar para trás para onde a bola vai intercetar a baliza (representado na Figura 57 com uma cruz (X)), andava lateralmente para impedir esse remate (representado na Figura 57 com um certo (✓)).

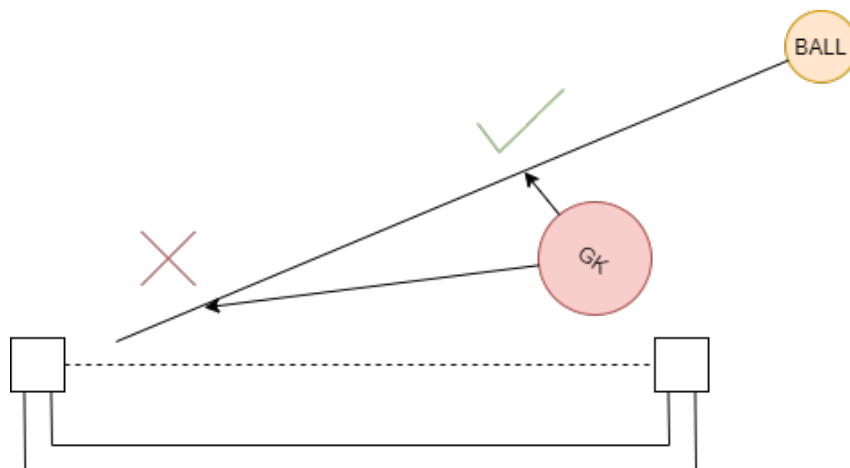


Figura 57: Defesa Lateral

Uma das desvantagens é que, considerando a Figura 57, caso o robô se desloque na trajetória representada com um certo (✓) nessa figura, necessita duma reação muito mais rápida para defender a bola devido a estar mais perto da mesma, sendo que a reação do robô não é boa e é compensada com a velocidade que ele consegue percorrer. A trajetória representada com a cruz (X) na Figura 57 utiliza a velocidade para compensar essa reação, pois apenas interceta a bola no seu ponto mais distante.

Outra desvantagem é no caso de um remate alto, havendo a possibilidade de ultrapassar por cima do guarda-redes e entrar na baliza, sendo que na trajetória representada pela cruz (X) na Figura 57 isso já não é possível.

A última desvantagem é nas situações em que a bola iria para fora da baliza passando perto do guarda-redes, visto que o mesmo iria fazer um movimento desnecessário (Figura 58) e até possibilitar meter a bola dentro da baliza.

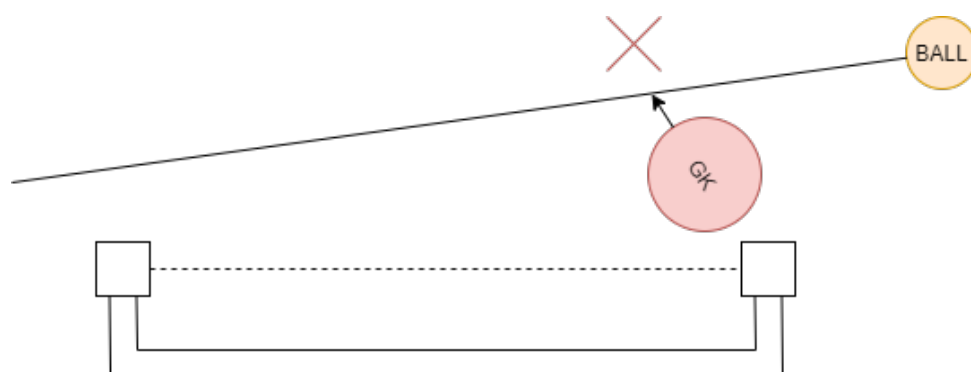


Figura 58: Defesa desnecessária

Esta implementação acabou por ser bastante útil para os robôs atacantes na habilidade de receção, pois dessa forma consegue-se deslocar para receber a bola quando o passe vai mais ao lado.

4.1.6.2 Previsão da trajetória - Referência à baliza

A diferença entre este método e o anterior está no referencial usado, sendo neste caso utilizado o referencial do centro da baliza. O resto do método é o mesmo, pois são usadas as mesmas equações para previsão da trajetória.

Através das coordenadas da bola no referencial da baliza é retirado o declive (equação 4.14) e a ordenada na origem (equação 4.15) da reta, obtendo-se depois o ponto de interseção da reta com o eixo das abscissas (equação 4.16) que equivale à linha da baliza (Figura 59).

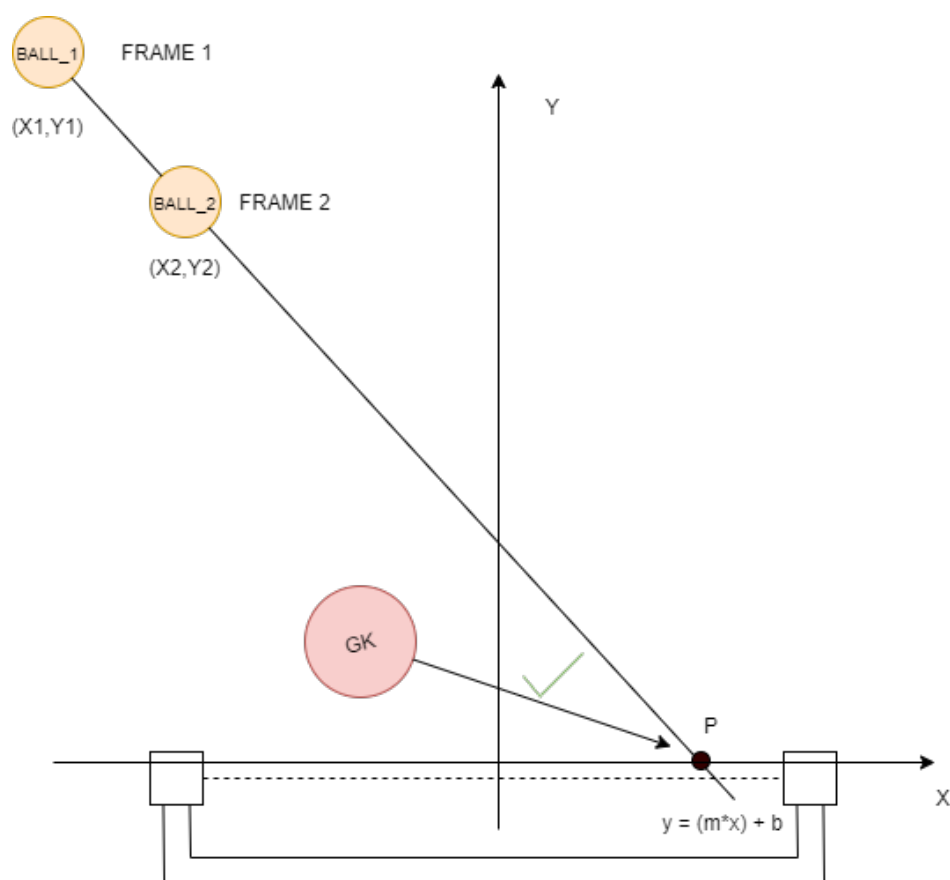


Figura 59: Previsão da trajetória - Baliza

Este método permite que o guarda-redes defenda os remates no ponto de interseção da bola com a baliza, para considerar as bolas altas, pois a estrutura móvel dificultará a entrada de bolas entre o topo do guarda-redes e a trave da baliza.

O robô também "percebe" se a bola vai intercepar a baliza ou se vai para fora do campo, fazendo o mesmo movimentar-se apenas nos casos necessários em que a bola vá na direção da baliza.

4.2 Posicionamento

Um problema de grande parte dos guarda-redes que participam na [MSL](#) vem da evolução do remate destes robôs, pois neste momento há equipas que rematam com muita precisão e velocidade, fazendo os guarda-redes não chegarem a tempo ao ponto de interseção da bola com a baliza. Com o uso da *Microsoft Kinect* esse tempo ainda é maior devido ao baixo *frame rate* (30 *frames* por segundo).

Foi necessário encontrar um método que permitisse ao robô cobrir o máximo de baliza possível de forma a impedir os remates com movimentos reduzidos. Assim, se a bola estiver do lado esquerdo do campo, o guarda-redes tem de se posicionar para esse lado e não se manter no meio da baliza à espera de um remate, sendo necessário ter cuidado para não cobrir demasiado esse poste, pois caso contrário, o outro fica muito exposto. Na Figura 60 está representada esta situação, em que o robô com o certo (✓) localiza-se na posição desejada ao contrário do que se encontra com a cruz (✗) (solução utilizada por muitos guarda-redes na liga [MSL](#)).

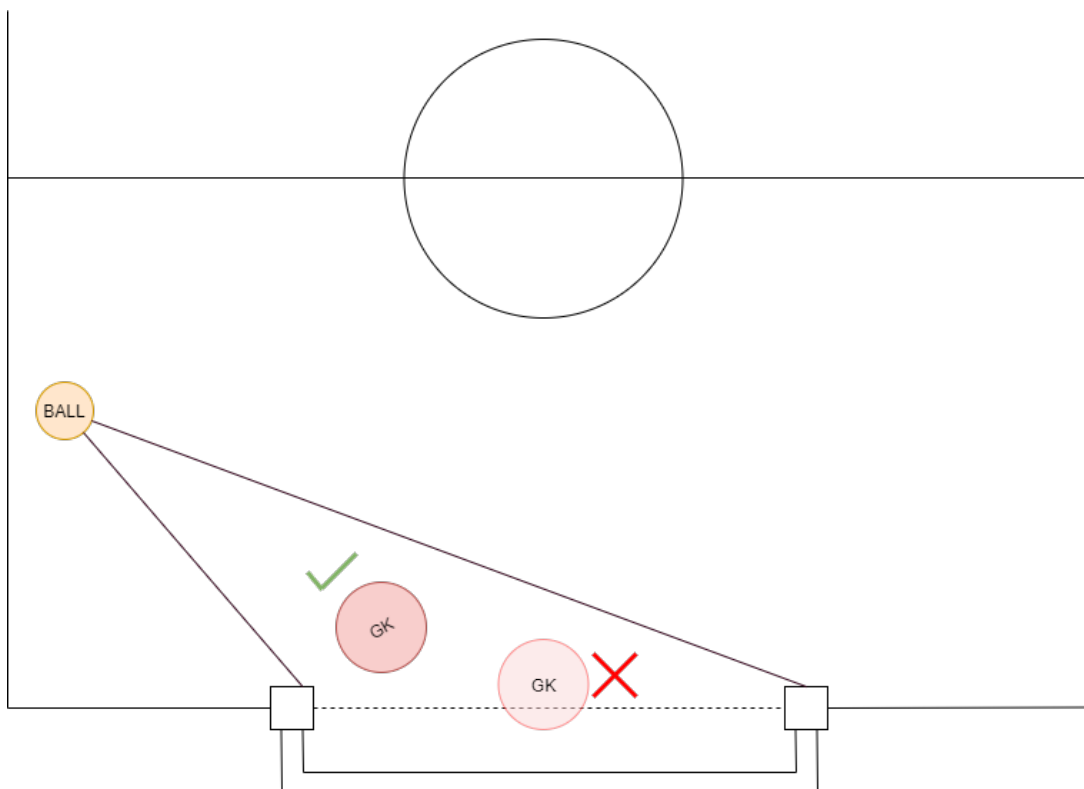


Figura 60: Posicionamento na baliza

Para produzir estes resultados foi utilizada uma elipse com os pontos incidentes perto dos postes da baliza (Figura 61). Estes pontos equivalem à posição lateral máxima que o robô se deve posicionar, não

sendo necessariamente no poste. As situações em que o robô se irá colocar nesses pontos incidentes é nos momentos em que a bola se encontra nas zonas do canto do campo. Visto que nessa situação o robô se encontrará na sua rotação máxima, a distância permitida ao guarda-redes entre o centro dele e o poste da baliza equivale ao diâmetro da bola, 20 centímetros. De forma a existir uma margem devido ao posicionamento não ser perfeito, é utilizada uma distância de apenas 10 centímetros.

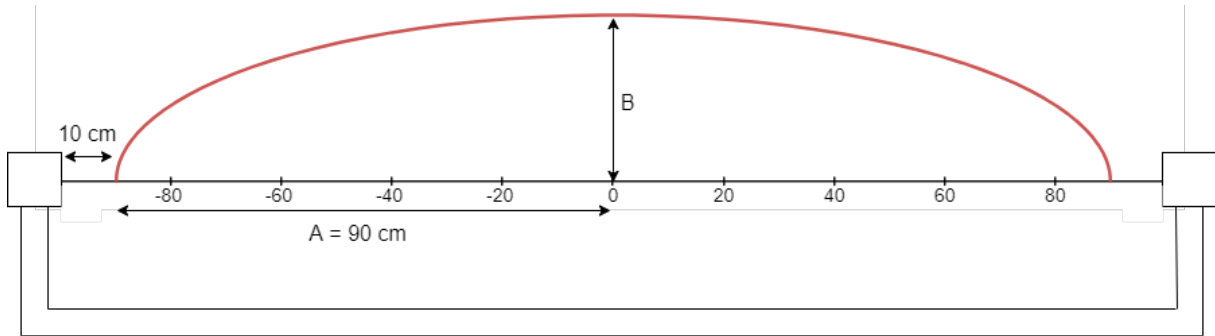


Figura 61: Elipse da baliza

Segundo a equação da elipse (equação 4.17), existem dois parâmetros ajustáveis. O parâmetro A , que é possível ver na Figura 61, indica o raio maior da elipse, e visto que a baliza tem 2 metros e o guarda-redes pode afastar-se dos postes 10 centímetros, o tamanho da elipse tem de ser de 180 centímetros entre esses pontos.

O ponto B indica a profundidade da elipse, sendo este um parâmetro importante pois define a agressividade com que o guarda-redes se direciona para os postes.

$$y = \sqrt{B^2 - \frac{B^2 * x^2}{A^2}} \quad (4.17)$$

O cálculo da posição do robô depende da posição da bola. É traçada uma reta entre a posição da bola e o centro da baliza, de forma a obter o ponto de interceção com a elipse (equação 4.18). Este ponto define a posição da abcissa onde o guarda-redes se deve posicionar. A agressividade que B provoca vem dessa interceção, pois quanto maior for B , mais perto das pontas ocorre a interceção.

$$x = \frac{B}{\sqrt{m^2 + \frac{B^2}{A^2}}} \quad (4.18)$$

Na Figura 62 está representada a interceção com um valor de 25 para B tendo uma reta a representar em que ponto esta interceção acontece no eixo das abcissas.

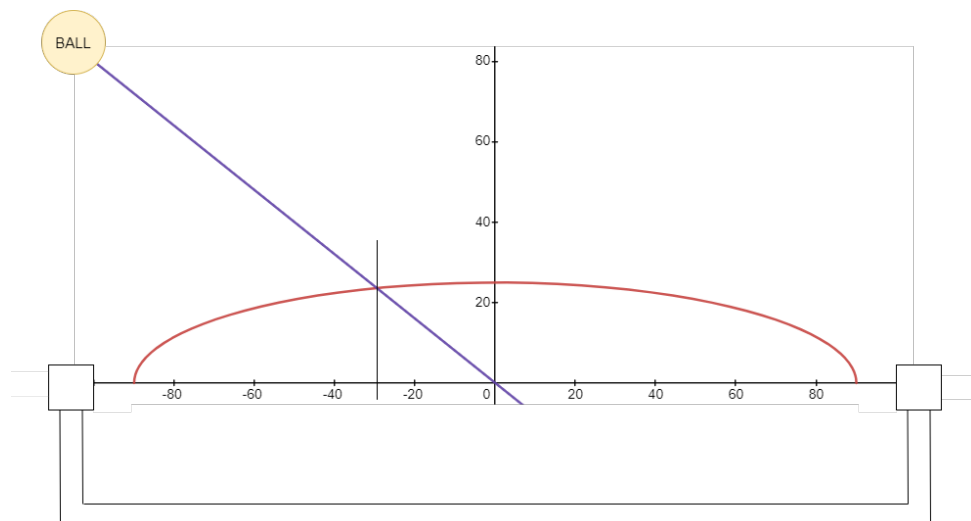


Figura 62: Elipse com $B = 25$

Na Figura 63 o valor de B é igual a 60, onde neste caso o robô teria de se deslocar mais para a esquerda estando a bola na mesma posição que na Figura 62.

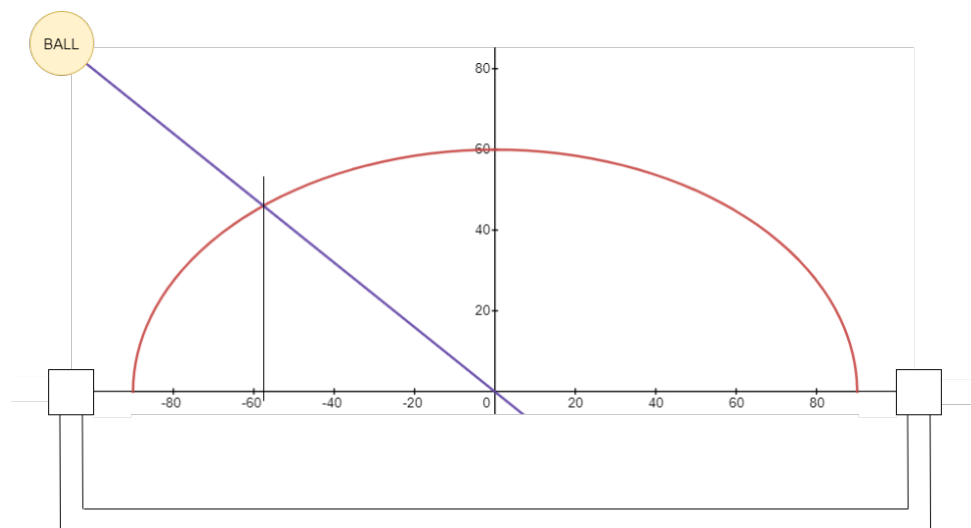


Figura 63: Elipse com $B = 60$

Foi tomada a decisão de fazer o robô andar numa reta fixa no eixo das ordenadas, pois é desejado que o robô tenha maior tendência para o poste mais próximo da bola mantendo uma pequena distância à baliza para não sofrer golos por cima. Desta forma com alterações nos valores de B da elipse é possível obter a tendência desejada para o poste mais próximo conforme as equipas rematem mais ou menos colocado, utilizando a movimentação pela reta que garante não sofrer golos por cima. Assim, após obter a interceção entre a reta e a elipse, a posição desejada do robô tomará o valor dessa intercessão em X e o valor da reta fixa em Y (Figura 64).

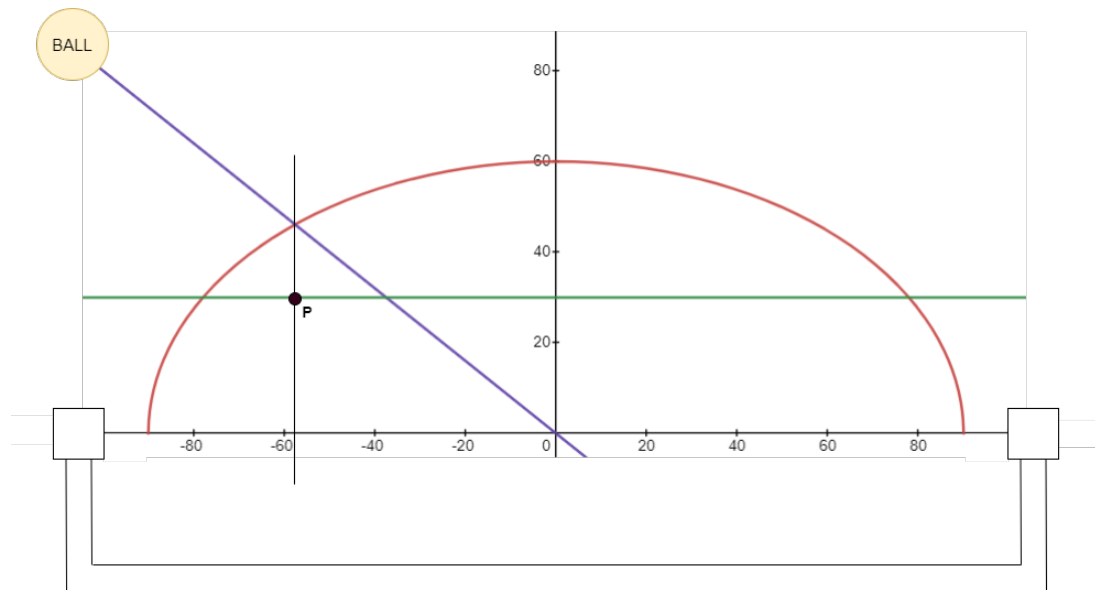


Figura 64: Posicionamento final na baliza

4.3 Orientação

O robô apenas detecta remates com o uso da *Microsoft Kinect*, mesmo esta tendo um campo de visão bastante reduzido, o que implica a necessidade do robô se orientar para a bola. Com isto em mente, foram desenvolvidos alguns algoritmos de orientação do robô.

O primeiro algoritmo faz o robô seguir sempre a bola utilizando um controlador PID, sendo o ângulo à bola a entrada do controlador. Este algoritmo foi testado, mas devido a oscilações da posição da bola e pequenos movimentos da mesma obrigarem o robô a estar em constante movimento, foi descontinuado, pois provocava maior consumo de bateria e instabilidade no robô.

A primeira alternativa foi limitar o movimento do robô utilizando gamas de valores dos pixels da imagem (Figura 65). Caso a bola se encontre entre uma certa gama de valores horizontais, entre o pixel 0 e 200 dos 640 horizontais, o robô teria de rodar para a esquerda até encontrar a bola na gama central da imagem (entre 200 e 440). O mesmo acontece se o robô estiver na gama da direita, rodando assim para esse lado.

Caso a *Microsoft Kinect* não detecte nenhuma bola é usada a câmara de localização para saber onde esta se localiza, orientando o robô com o ângulo obtido entre ele e a bola, fazendo a *Microsoft Kinect* voltar a vê-la novamente.

Este método também provou ser bastante instável, pois mesmo reduzindo o problema do método anterior, em situações que a bola estivesse muito próximo das gamas, a orientação tornava-se novamente instável.

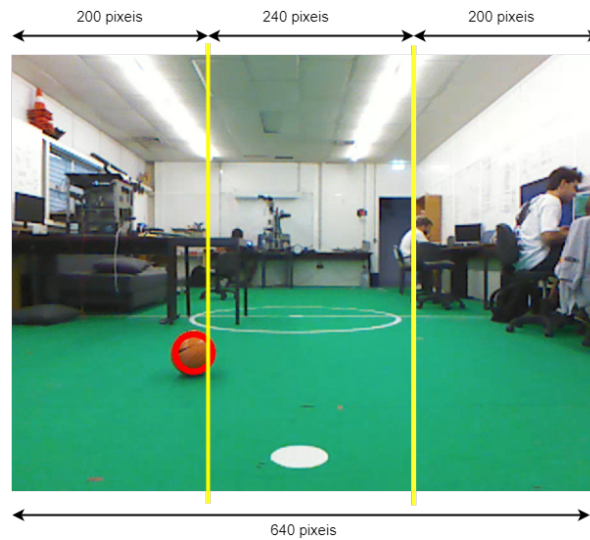


Figura 65: Orientação por gama de valores

Diferente dos métodos anteriores que usavam a posição da bola para se orientar, a segunda alternativa (método final implementado) utiliza como referência a posição do guarda-redes na baliza (demonstrado na secção 4.2), pois esse já depende da posição da bola. Assim, a orientação é calculada através de uma equação de terceiro grau em que a entrada x corresponde à localização do robô na baliza e a saída y corresponde à orientação do robô.

$$y = A * x^3 + B * x \quad (4.19)$$

Após alguns testes, os valores de A e B escolhidos foram 0.00014 e 0.1 respetivamente, obtendo o gráfico representado na Figura 66.

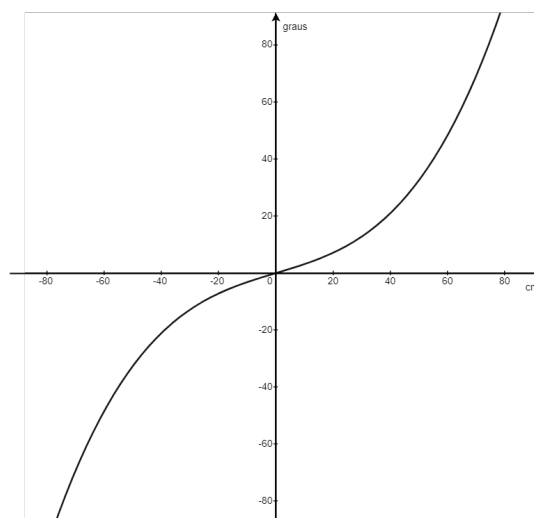


Figura 66: Equação 3º grau - Orientação

O eixo das abcissas corresponde à localização do robô na baliza em centímetros e o eixo das ordenadas corresponde à orientação em graus. A equação escolhida foi de terceiro grau de forma a obter valores negativos quando o robô se encontra no lado negativo da baliza, obrigando o guarda-redes a rodar para lados diferentes.

Este método mostrou-se bastante robusto, pois mesmo havendo oscilações na posição do guarda-redes, não eram acentuadas o suficiente para se notar no controlo da orientação.

4.4 Controlo de movimento

O controlo de movimento é a função responsável por calcular e enviar os comandos do PC para o baixo nível, sobre a velocidade linear, velocidade rotacional e direção de movimento. A Figura 67 representa o fluxograma desta função.

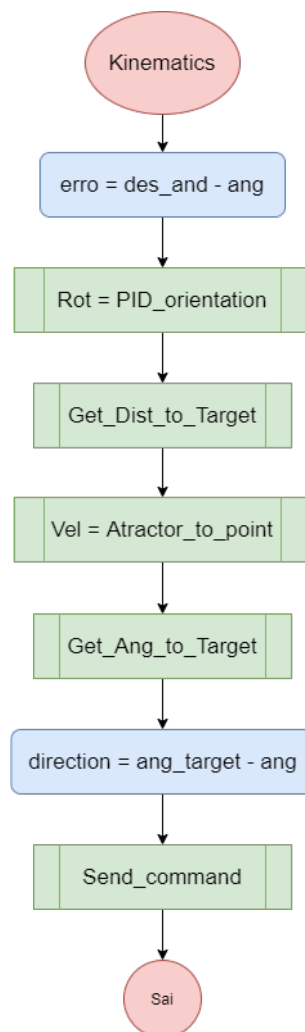


Figura 67: Fluxograma da Cinemática

Após calcular o erro da orientação através da subtração do ângulo desejado com o ângulo do robô, é utilizado um controlador PID para calcular a velocidade rotacional desejada. Em seguida é calculada a distância entre a posição do robô e a posição desejada para o mesmo segundo a equação 4.20.

$$dist = \sqrt{(P_d.Y - P_{GK}.Y)^2 + (P_d.X - P_{GK}.X)^2} \quad (4.20)$$

Esta distância é utilizada para entrada do atrator, uma função que calcula a velocidade linear desejada de forma ao robô convergir para um ponto desejado (ponto atrator). Este atrator é uma função exponencial com dois parâmetros de calibração, K_a e K_i , que permitem um ajuste para obter a saída desejada (equação 4.21).

$$vel = K_a * \exp x * K_i \quad (4.21)$$

Os valores finais destes parâmetros K_a e K_i são 6.6 e 0.04 respectivamente, sendo o gráfico final representado pela Figura 68. O eixo das abcissas corresponde à distância entre o robô e a posição desejada em centímetros e o eixo das ordenadas corresponde à velocidade linear em percentagem (100% equivale à velocidade linear máxima estabelecida pela placa controladora dos motores). Foram estabelecidos limites que não permitem a velocidade linear ultrapassar os 100% e percentagens abaixo dos 8% sejam equivalentes a 0%.

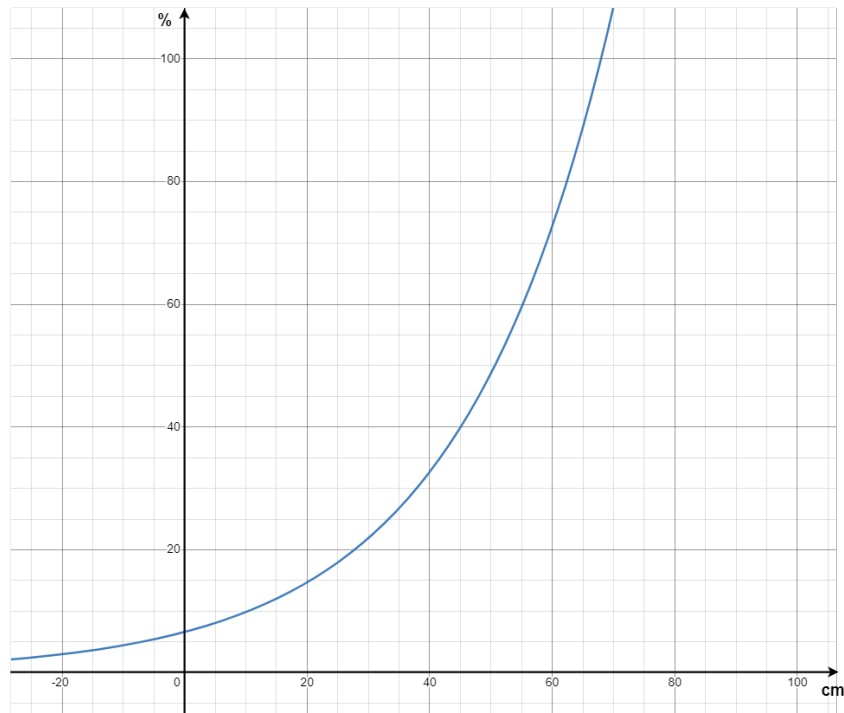


Figura 68: Atrator velocidade linear

Na situação de remate à baliza, as entradas (eixo das abcissas) não são representadas em centímetros mas sim em percentagem, pois numa situação de remate é desejado que o robô saia do ponto de repouso em velocidade máxima, independentemente da distância inicial para o objetivo. Assim sendo, no momento que um remate é detetado, essa distância ao alvo é considerada a distância máxima e, posteriormente, a distância em percentagem é dada pela divisão da distância atual por essa máxima multiplicada por 100 (equação 4.22).

$$dist(\%) = \frac{dist}{dist_{max}} * 100 \quad (4.22)$$

Para calcular a direção do movimento linear é necessário o ângulo entre o alvo e o robô, no referencial da baliza, obtido através da função *ATan2* entre esses dois pontos (equação 4.23).

$$ang_target = \arctan 2((P_d.Y - P_{GK}.Y), (P_d.X - P_{GK}.X)) \quad (4.23)$$

Esse ângulo é somado ao do robô em relação ao referencial para se obter a direção do movimento linear desejada (Figura 69).

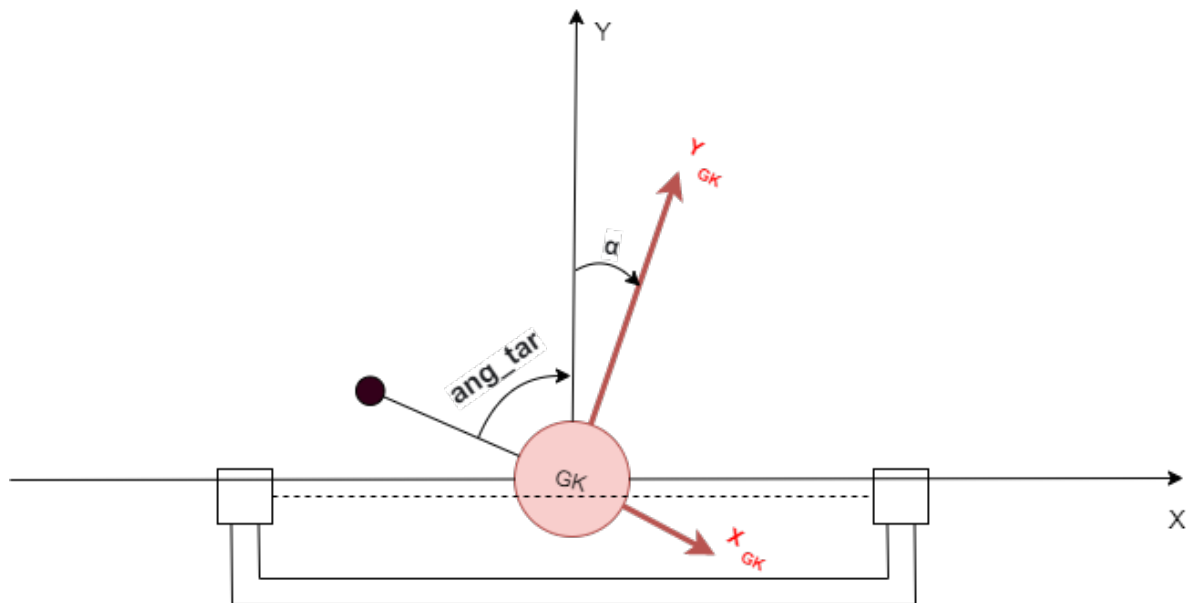


Figura 69: Direção do movimento linear

Os valores da velocidade linear, rotacional e direção são enviados para o baixo nível via porta série.

4.5 Estrutura móvel

Numa atualização recente dos regulamentos do *RoboCup*, foi permitido que o guarda-redes, dentro da pequena área, estendesse a sua área em 920cm^2 para qualquer lado durante um segundo, sendo necessário um intervalo de quatro segundos para o fazer novamente [35].

Foi decidido criar uma estrutura capaz de subir e proteger a parte superior do guarda-redes. Visto que a estrutura fixa do guarda-redes tem 70cm de comprimento, para criar uma área de 920cm^2 , a estrutura móvel poderia subir cerca de 12cm .

Para subir a estrutura e tomando como referência a equipa dos *Tech United*, foi utilizado um motor centrado com a estrutura que tem como função enrolar um fio ligado a parte inferior da mesma, fazendo com que esta suba até uma certa altura. Para garantir que a estrutura suba no máximo 12cm foram utilizados batentes. Para descer a estrutura o fio é desenrolado e a mesma cai com a força gravítica.

A estrutura móvel é colocada na estrutura fixa, sendo necessária uma distância de $38,5\text{cm}$ entre os eixos principais e um tamanho horizontal total de 70cm , sendo esse o tamanho máximo citado pelo regulamento. A Figura 70 representa um modelo desta estrutura móvel.

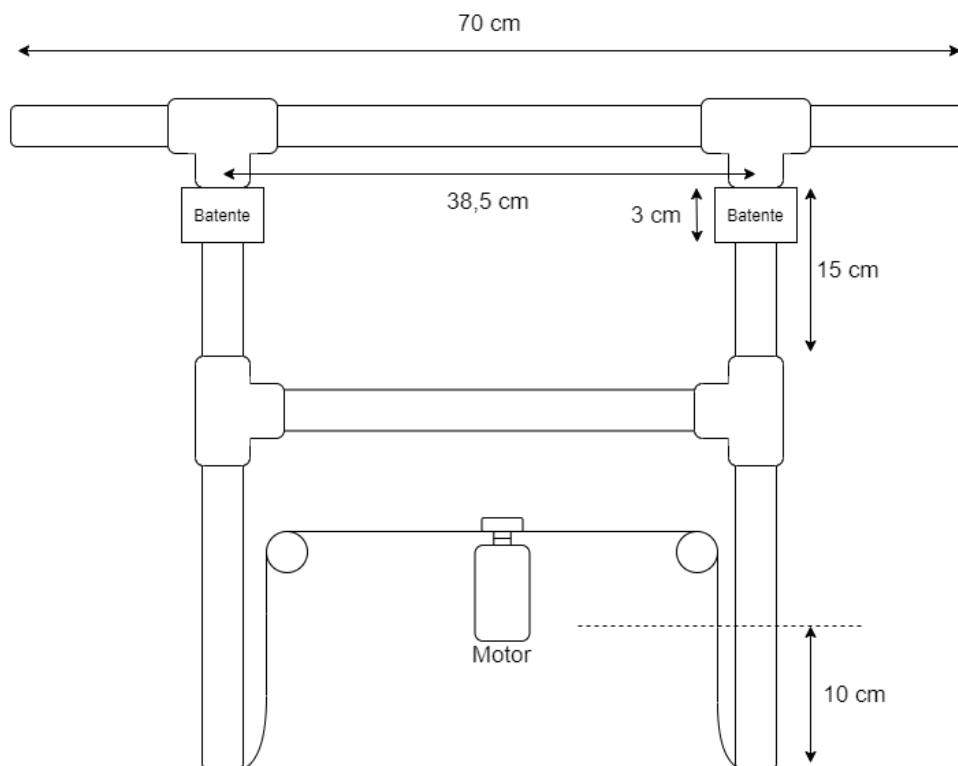


Figura 70: Modelo da estrutura móvel

4.5.1 Tipo de material

Grande parte das equipas que utilizam este tipo de estrutura usam fibra de carbono devido à sua rigidez e peso, pois é um material que não se deforma e é bastante leve. Porém, estes materiais de carbono são bastante caros, pretendendo esta equipa uma solução fiável e económica. O material escolhido acabou por ser o Polipropileno Copolímero Aleatório (PPR) devido ao seu baixo preço, alta rigidez e baixo peso. Os tubos de PPR mostraram-se rijos o suficiente para aguentar remates de humanos, mesmo sendo um material leve e barato.

Os tubos foram cortados às dimensões necessárias e ligados aos elos, sendo o resultado apresentado na Figura 71.



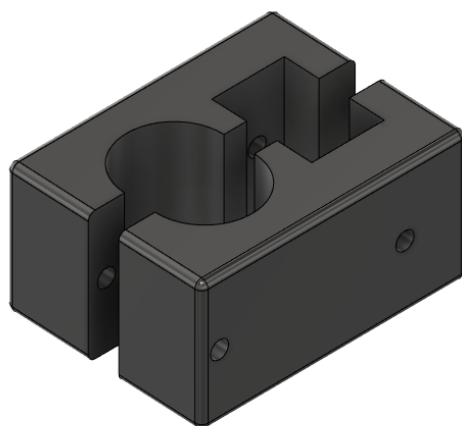
Figura 71: Estrutura modelizada

4.5.2 Batentes

Os batentes foram impressos a Três Dimensões (3D), com uso de filamento Poliacido Láctico (PLA) e foram desenhados de forma a servir de encaixe entre a estrutura móvel e fixa. Como a estrutura fixa é composta por um ferro quadrado e a estrutura móvel por um tubo redondo, foi desenvolvida uma peça que ligasse esses dois tipos.

A peça desenvolvida é aparafusada a outra igual pela estrutura fixa, englobando as duas estruturas e funcionando como batente (Figura 72a). Na Figura 72b é possível observar um dos batentes a ligar as duas estruturas no robô. Foram colocados quatro batentes no total, sendo que dois deles funcionam como conectores entre estruturas.

Os dois batentes estão situados na parte superior das duas barras e os conectores na parte inferior, de forma a aumentar a estabilidade desta estrutura.



(a)



(b)

Figura 72: a) Modelo dos batentes. b) Batentes na estrutura.

4.5.3 Suporte e veio do motor

Visto que o motor foi colocado no meio da estrutura, era necessário desenvolver uma peça que englobasse o motor e fizesse uma ligação entre esse e a estrutura fixa do guarda-redes, de forma a servir de suporte.

Também foi utilizada modelagem 3D para o desenvolvimento deste suporte. O motor é colocado e aparafusado dentro do suporte, criando assim uma camada de proteção do mesmo, tendo ainda sido desenvolvido um encaixe quadrado de forma a prender o motor aos ferros da estrutura fixa (Figura 73).

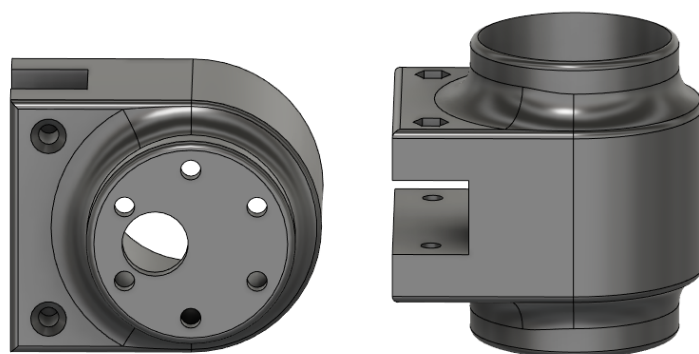


Figura 73: Modelo do suporte do motor

Como este motor tem o objetivo de enrolar um fio com a sua rotação foi desenhada uma peça para o veio com esse propósito. Esta peça entra no veio aparafusada ao mesmo, tendo um furo ao centro que possibilita a passagem do fio de um lado ao outro (Figura 74).

Visto que o motor utilizado é um motor DC de 12V e 150 rotações por minuto, o perímetro do círculo no meio da peça é de 24cm, de forma a garantir que a estrutura aumenta os 12cm com apenas meia rotação do motor. O tempo de subida da estrutura é dado pelo tempo que o motor demora a fazer meia rotação, o que equivale a 200 milissegundos.

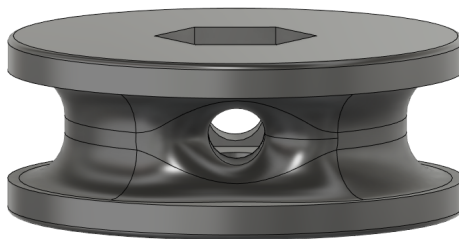


Figura 74: Modelo do veio do motor

O resultado final destas duas peças no robô é apresentado na Figura 75, onde é possível ver que o fio passa por duas roldanas em cada lado para transformar o movimento horizontal, gerado pelo motor e pela peça do veio, num movimento vertical responsável por subir a estrutura.

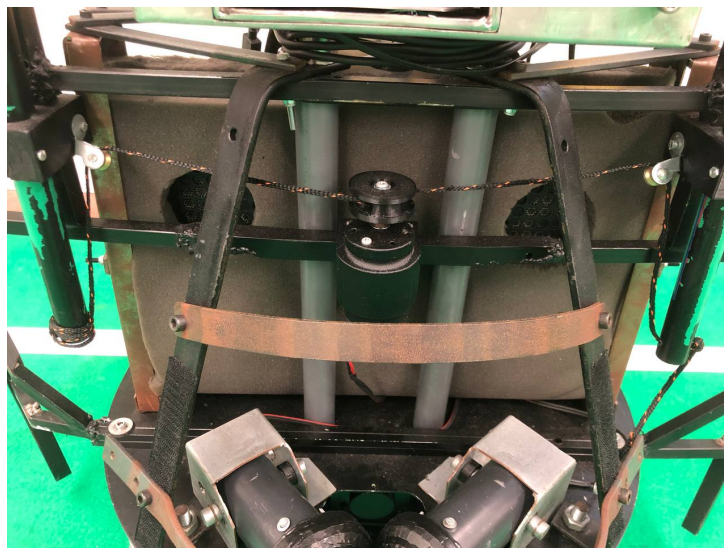


Figura 75: Motor da estrutura móvel

4.5.4 Placa de controlo

Após o desenvolvimento da estrutura foi necessário criar um sistema de controlo para dar ao motor a movimentação desejada em determinado tempo. Num cenário ideal, a placa responsável por controlar este motor seria a unidade de processamento de baixo nível, porém, devido à utilização desta placa para todo o projeto, esta ficou sem pinos de comunicação e *Input/Output*.

Foi necessário escolher outro microcontrolador de baixo custo pois apenas ia exercer uma função, a ativação de um motor num determinado tempo. Foi selecionada a *Raspberry Pi Pico* como o microcontrolador dedicado a controlar esse motor pelo seu baixo preço. Esta placa controla o motor com ajuda de uma *ponte H*, usada para controlar motores de corrente contínua.

A Figura 76 representa a arquitetura de *hardware* do sistema da estrutura que é apenas utilizada pelo guarda-redes, sendo esse o motivo de não ser representada na figura da arquitetura de *hardware* de todos os robôs (Figura 22). A comunicação entre a *Raspberry Pi Pico* e o *PC* é feita através de uma porta série.

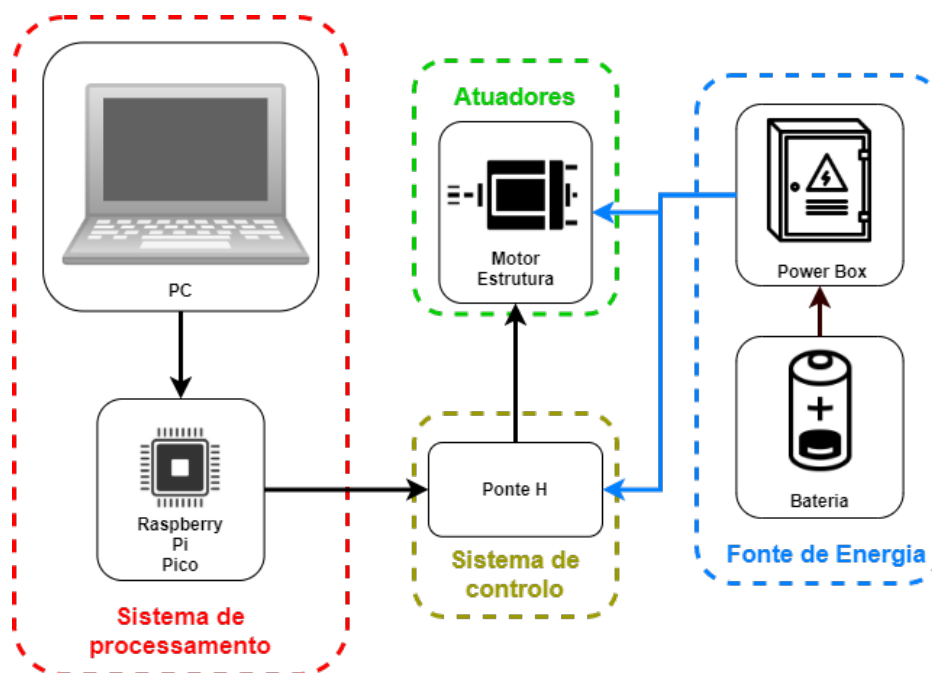


Figura 76: Arquitetura de *hardware* da estrutura móvel

Para a ligação entre a *Raspberry Pi Pico* e a *ponte H* são necessários três pinos de *Input/Output*. Um é dedicado ao pino de ativação da *ponte H*, sendo este um pino de *PWM* que dita a velocidade do motor. Os outros dois são pinos simples de *High/Low* dedicados a ativar o motor e sinalizar o sentido do mesmo, sendo ativados independentemente e respetivos a cada um dos sentidos.

Foi utilizado o *GPIO 20* da *Raspberry Pi Pico* como *PWM*, e o *GPIO 21* e *GPIO 22* como pinos simples de *High/Low*, correspondendo à subida e descida da estrutura respetivamente.

Esta ligação entre a *ponte H* e os pinos da *Raspberry Pi Pico* está representada na Figura 77, assim como a ligação entre a *ponte H*, o motor e a *Power Box*.

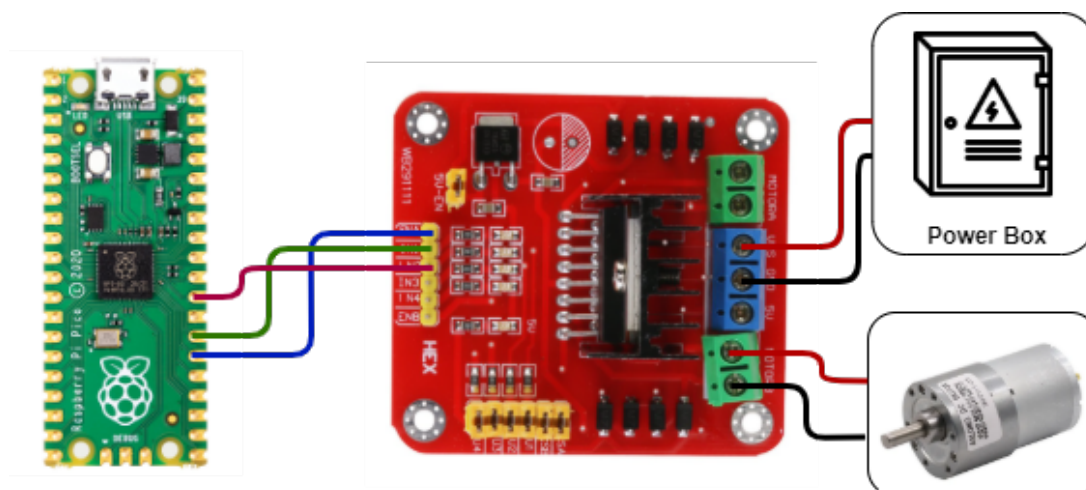


Figura 77: Ligação do microcontrolador (estrutura móvel)

A linguagem de programação utilizada neste microcontrolador foi o *MicroPython*, a mais utilizada no desenvolvimento de projetos com este microcontrolador. O *MicroPython* é uma implementação do *Python* projetada especificamente para operar em dispositivos com recursos limitados, como microcontroladores.

4.5.4.1 Compartimento da placa de controlo

Após ser desenvolvida a placa com a ligação do microcontrolador à *Ponte H*, foi necessário criar um compartimento que a protegesse e fixasse no robô. Foi desenvolvida uma caixa em modelização 3D e impressa em filamento *PLA*. A caixa desenvolvida tem por dentro uma zona mais elevada para a *ponte H* ficar mais subida que a placa e o microcontroaldor, pois apenas se liga a essa com uns *jumpers* na zona de ligação com os pinos. Assim essa camada foi adicionada para a mesma não ficar pendurada ou a forçar os *jumpers*. Tem também uma abertura na zona de ligação do cabo do microcontrolador com o *PC*, assim como na zona de ligação da *Power Box* e do motor com a *Ponte H*.

Nessa caixa foram feitos furos por dentro para aparafusar a placa e a *Ponte H*, assim como furos em cima das paredes para aparafusar a tampa.

Na tampa existem duas aberturas, uma circular para ser possível carregar no botão de *Reset* do microcontrolador e verificar o *LED* de ligação, e a abertura quadrada, que serve para o dissipador da

ponte H ficar fora da caixa.

Na Figura 78 é possível verificar à esquerda a caixa desenvolvida, e à direita a respetiva tampa.

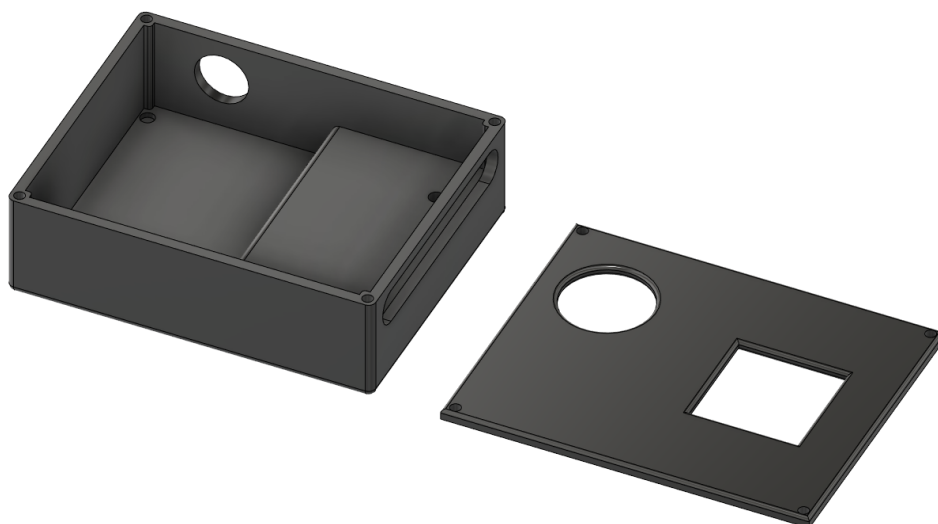


Figura 78: Modelo do compartimento da placa (estrutura móvel)

O resultado final após a soldagem e montagem da placa no compartimento é apresentado na Figura 79. Em seguida foi colocada no robô como se vê na Figura 80, onde é possível observar o *LED* aceso e os cabos ligados.



Figura 79: Placa de controlo



Figura 80: Placa no robô

4.5.5 Implementação

Quando um certo comando é recebido, a estrutura tem de subir e descer no tempo de um segundo e não o voltar a fazer durante quatro segundos, mesmo que continue a receber esses comandos. Para

isso foi desenvolvida uma máquina de estados representada na Figura 81, e uma *Thread* responsável por receber comandos do PC.

Uma *Thread* é uma função que se executa paralelamente com o programa principal. Assim, esta *Thread* funciona como uma interrupção que está sempre a ler a porta série à espera de um comando e quando recebe um carácter de finalização, neste caso '\n', esse comando é colocado numa variável global, para ser comparado na máquina de estados.

Como se pode ver na Figura 81 essa máquina contém cinco estados, sendo esses o *Idle*, *Up*, *Stay*, *Down* e *Wait*.

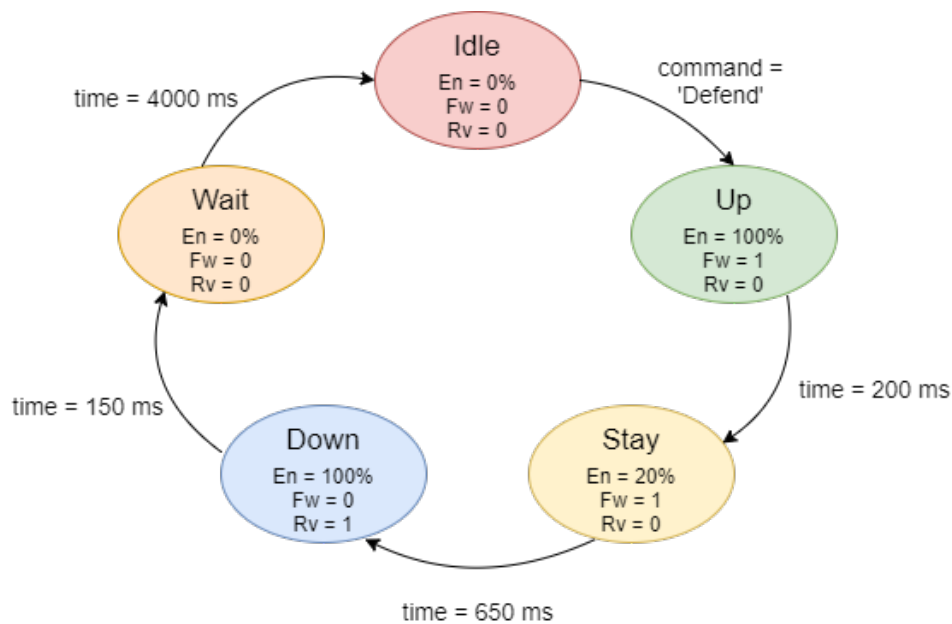


Figura 81: Máquina de estados (estrutura móvel)

O primeiro estado (*Idle*) é responsável por manter o motor desligado e esperar que o comando desejado seja recebido. Após receber um comando, é feita uma comparação desse com o comando desejado (*Defend*), e caso esteja correto, a máquina avança para o próximo estado.

O segundo estado (*Up*) é responsável por ativar o motor para levantar a estrutura. Este estado coloca o valor do PWM a 100% e o pino da direção correta do motor a *High* pelo tempo de 200ms. Este foi o tempo que se mostrou suficiente para a estrutura subir completamente, pois diz respeito a meia rotação do veio do motor. Após este tempo acabar, passa para o estado seguinte, o estado *Stay*.

Neste terceiro estado, a estrutura tem de permanecer em cima pelo tempo máximo possível, sendo esse tempo dado pela subtração de um segundo (tempo máximo) com os tempos de subida e descida da estrutura. De forma a permanecer a estrutura em cima, este estado coloca um valor de PWM de 20%, sendo o suficiente para a mesma não descer. Visto que a estrutura demora 350ms a subir e descer,

após passarem $650ms$, a máquina passa para o estado *Down*.

O estado *Down* tem como objetivo descer a estrutura, colocando o [PWM](#) aplicado ao motor a 100%, mas com o sentido contrário, de forma a dar folga à corda e fazer a estrutura descer com a força gravítica. Após alguns testes, verificou-se que o tempo necessário para a estrutura descer completamente era de $150ms$, sendo esse o tempo de passagem para o próximo estado.

O último estado, *Wait*, garante que são cumpridos os quatro segundos de espera até subir a estrutura novamente. Assim, apenas se passa para o primeiro estado para verificar o comando e voltar a subir a mesma quando são respeitados os quatro segundos.

4.6 Defesa de penáltis

Após a análise do estado da arte, houve uma solução que cativou a atenção da equipa, que foi a defesa de penáltis desenvolvida pela equipa dos *Tech United*. O funcionamento consiste num adianto ao remate do robô através do posicionamento dele com a bola.

Este adianto é possível devido aos robôs rematarem apenas em frente, sendo que ao traçar uma reta entre o robô e a bola, é obtida a posição que o robô irá rematar. Isto também só é possível devido aos robôs não se puderem mexer muito com a bola, precisando rematar imediatamente. Isto não é possível de aplicar em situações normais de jogo pois o jogador pode mover-se livremente com a bola.

A matemática aplicada é a mesma que a previsão de remates, sendo que neste caso, ao invés de serem retirados dois frames da bola, a reta é traçada entre dois pontos, o centro do robô e da bola. Os robôs adversários são identificados com a rede neuronal e a sua posição relativamente à baliza é obtida exatamente da mesma forma da bola, como referenciado na secção [4.1.3.2](#).

Para traçar a reta primeiramente é obtido o declive da mesma (equação [4.24](#))

$$m = \frac{Y_{bola} - Y_{robo}}{X_{bola} - X_{robo}} \quad (4.24)$$

Em seguida calcula-se a ordenada na origem segundo a equação [4.25](#)

$$b = Y_{robo} - (m * X_{robo}) \quad (4.25)$$

De forma a obter a posição desejada do guarda-redes, é obtido o ponto de interceção dessa reta com a baliza (equação [4.26](#)).

$$P = -\frac{b}{m} \quad (4.26)$$

Os resultados pretendidos são exemplificados na Figura 82 pela sequência de imagens.

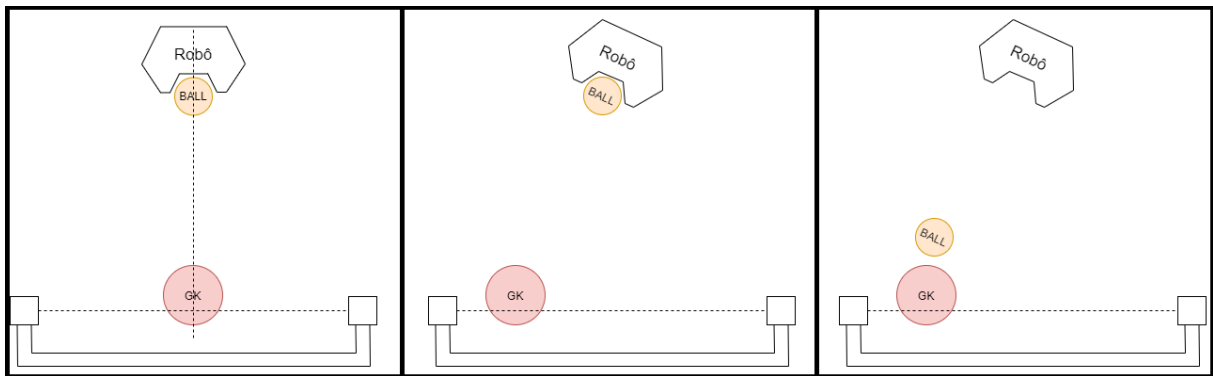


Figura 82: Sequência de movimento no penáلتي

Como se viu na estrutura de processamento do guarda-redes (Figura 46), a defesa de penáلتي não entrava nessa estrutura, pois esta implementação entra em vigor com o aviso da *BaseStation* de que é necessário defender penáلتي. O fluxograma da Figura 83 representa esta linha de processamento.

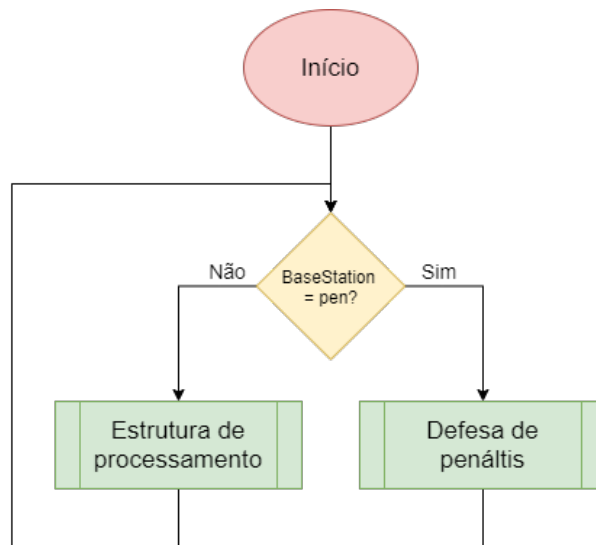


Figura 83: Estrutura de processamento com penáلتي

Capítulo 5

Testes e análise de resultados

Este capítulo é constituído por cinco subcapítulos que apresentam os resultados obtidos após a implementação de todos os processos apresentados. Estes resultados foram obtidos em três momentos/espços: no [LAR](#), no Festival Nacional de Robótica e no *RoboCup2023*.

No primeiro subcapítulo é feita uma análise à localização do robô, pois mesmo não sendo o trabalho desta dissertação, é um parâmetro que tem influência nos testes e resultados obtidos dos restantes processos. O segundo subcapítulo corresponde à análise dos resultados para o sistema de visão. Neste tópico são analisados os resultados da deteção e cálculo da posição da bola assim como a deteção e defesa de remates. No terceiro subcapítulo são analisados os testes e resultados obtidos para o posicionamento e orientação do guarda-redes na baliza. No subcapítulo seguinte é feita a análise à estrutura móvel desenvolvida para o guarda-redes. Por fim, no quinto subcapítulo é feita uma análise à solução implementada para a defesa dos penáltis.

5.1 Localização e orientação do robô

A localização do robô é um fator que influencia os testes de outros módulos desta dissertação. São demonstrados alguns testes realizados na localização do robô assim como a informação da orientação do robô segundo o referencial da baliza.

Para testar a localização do robô o mesmo foi colocado em pontos conhecidos da baliza para recolher os dados da sua localização.

O primeiro ponto tem como valor das coordenadas 0 milímetros para a abcissa e 250 milímetros para a ordenada segundo o referencial da baliza (ver secção [4.1.3.3](#)). Na Figura [84](#) é possível observar quatro imagens. Na imagem *a*) está representa a imagem da *Microsoft Kinect* com a distância da bola em milímetros e a percentagem de certeza sobre a bola. A imagem *b*) corresponde à posição real do robô na baliza vista por outro ângulo. Na imagem *c*) é demonstrada uma representação da localização do robô

no campo, e na imagem *d)* as coordenadas exatas obtidas pelo robô em milímetros pelo referencial da baliza.

Analisando estes resultados é possível observar que a localização do robô para este ponto tem um desvio médio de 54 milímetros para as abcissas e 47 milímetros para as ordenadas.



Figura 84: a) Imagem da *Microsoft Kinect*. b) Posição real do robô. c) Representação da localização do robô no campo. d) Coordenadas obtidas pelo robô em milímetros (x,y). (1º teste)

O segundo ponto tem como valor das coordenadas, 750 milímetros para a abcissa e 150 milímetros para a ordenada segundo o referencial da baliza. O resultado obtido está representado na Figura 85 e o desvio médio das abcissas é cerca de 106 milímetros, e no eixo das ordenadas é de 72 milímetros.

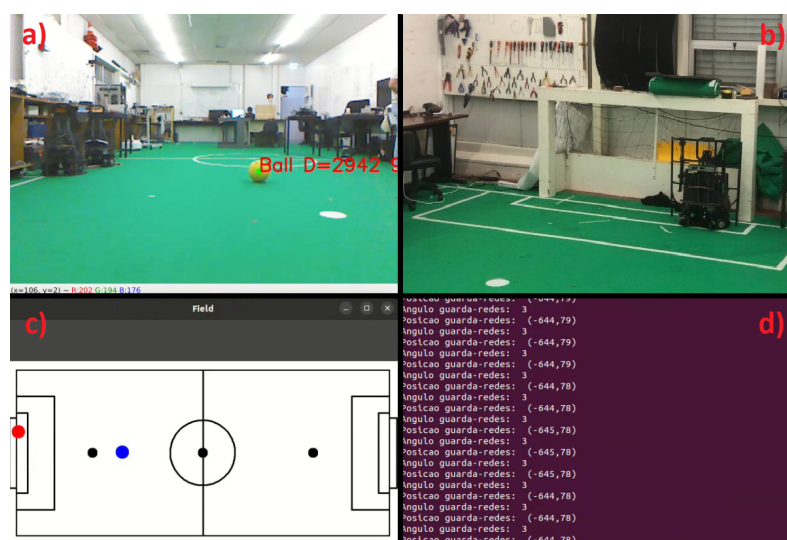


Figura 85: a) Imagem da *Microsoft Kinect*. b) Posição real do robô. c) Representação da localização do robô no campo. d) Coordenadas obtidas pelo robô em milímetros (x,y). (2º teste)

De acordo com estes dois testes é possível concluir que quanto mais longe do centro do referencial se encontra o robô, maior é o desvio da localização real, demonstrando também que a localização tem mais precisão que exatidão.

Para testar a orientação do robô, o mesmo foi colocado numa orientação de 45° segundo o referencial da baliza com uso de pontos colocados no chão, sendo obtidos os resultados apresentados na Figura 86.

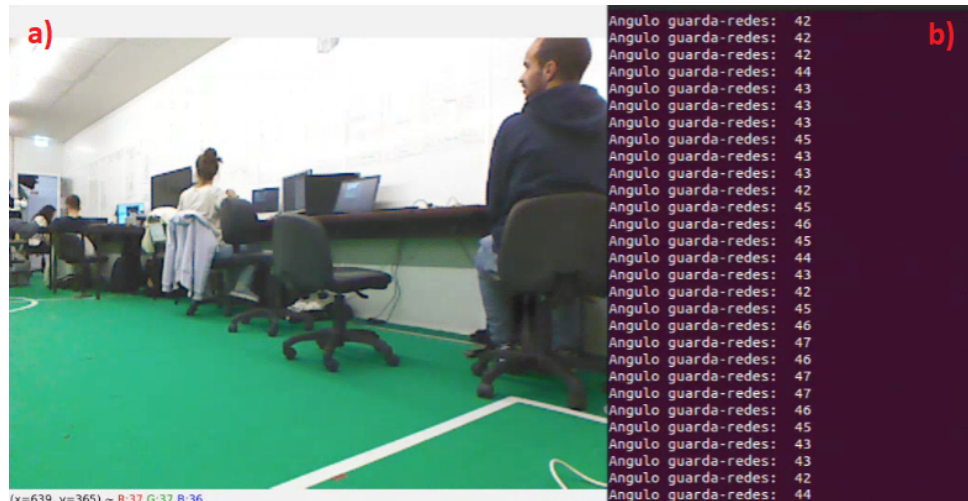


Figura 86: a) Imagem da *Microsoft Kinect*. b) Ângulo obtido pelo robô

Analisando estes resultados foi obtido o gráfico da Figura 87 onde se pode observar que a média dos valores retirados é de 44° , equivalente a um erro de 1° para a orientação real de 45° .

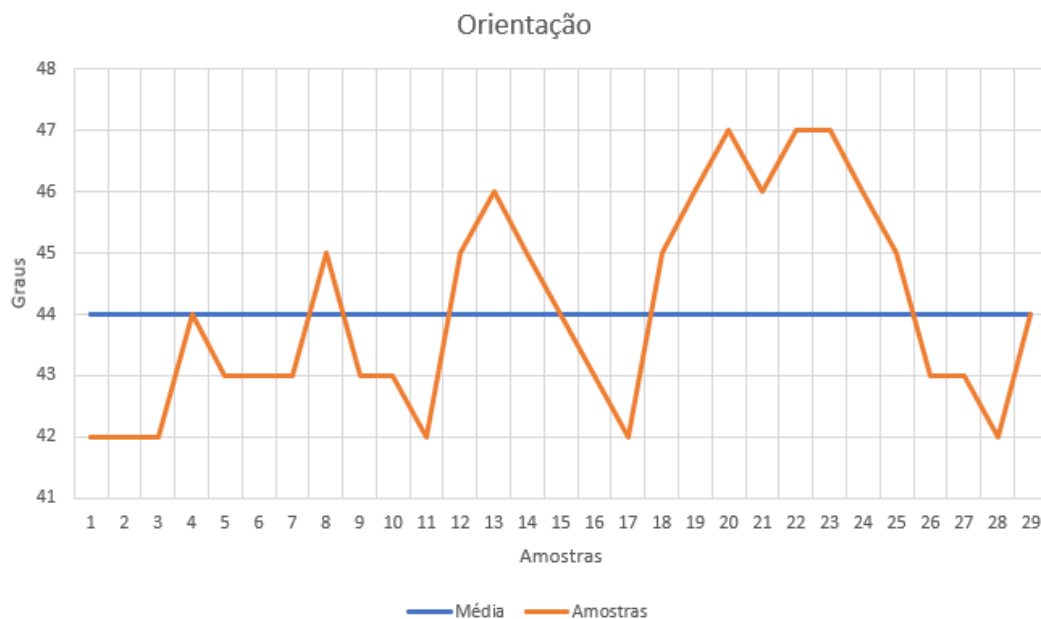


Figura 87: Gráfico da orientação do Robô

5.2 Sistema de visão

Em primeiro lugar, vão ser analisados os testes e resultados obtidos sobre a distância à bola lida pela *Microsoft Kinect* assim como a deteção da bola com o modelo de cores *HSV* e com a rede neuronal. Em seguida são analisados os testes do cálculo da posição da bola e é finalizado com a deteção e defesa de remates.

5.2.1 Deteção da bola e distância

Na primeira fase a deteção da bola era feita através do modelo de cores *HSV*. Na Figura 88 é possível observar na imagem *a)* uma máscara relativa à deteção da cor laranja da imagem obtida pela câmara, e na imagem *b)* a deteção da bola na câmara.

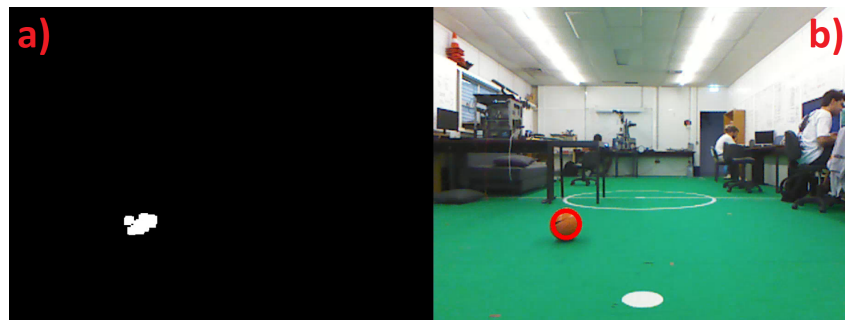


Figura 88: a) Máscara para deteção de cor laranja. b) Deteção da bola

Um inconveniente deste tipo de deteção provém do funcionamento da deteção se basear na cor e não na forma da bola, o que faz ser necessária uma nova calibração caso a bola seja de outra cor. Na Figura 89 está representado este problema em que apenas a bola laranja é detetada.

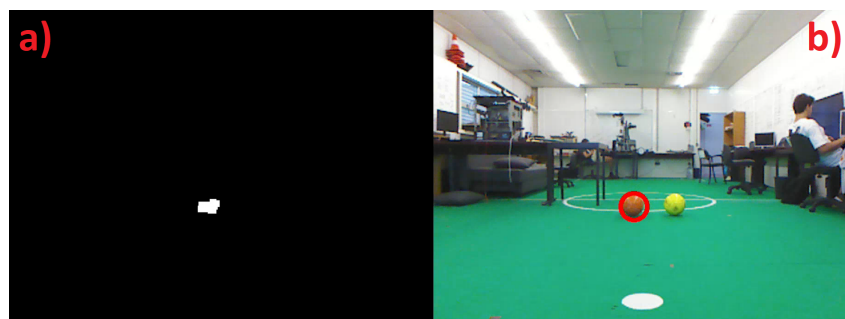


Figura 89: a) Máscara para deteção de cor laranja. b) Deteção de apenas uma bola (HSV)

Relativamente à deteção da bola pelo modelo de cores *HSV* existe outro grande problema, sendo que este se mostrou bastante inconveniente no Festival Nacional de Robótica, a primeira competição

que a equipa participou. No [LAR](#), este problema não foi muito relevante devido às pequenas dimensões do campo ali existente, já na competição, em que o campo apresenta umas dimensões maiores, foi bastante notório. Este problema deve-se à distância reduzida que este modelo consegue detetar a bola (até aproximadamente 4200 milímetros), sendo que na competição, devido ao tamanho do campo, as equipas rematavam de distâncias consideráveis fazendo o guarda-redes não conseguir defender devido à deteção tardia da bola.

Na Figura 90 é possível observar que o robô deixa de detetar a bola quando bastante afastada do guarda-redes devido à máscara utilizada. Esta distância representada na imagem (aproximadamente 5 metros) é a distância de onde grande parte dos remates são feitos nas competições.

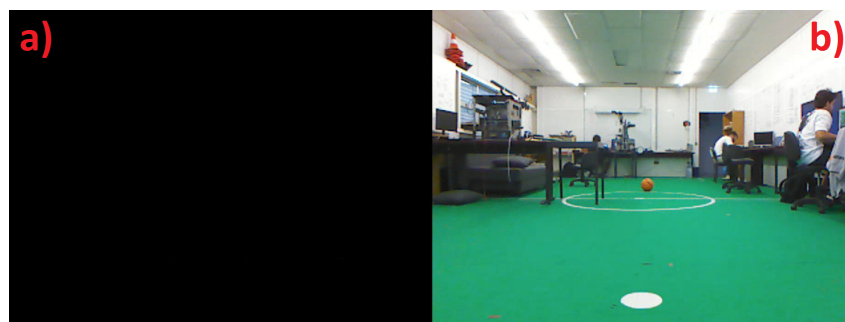


Figura 90: Deteção da bola limitada pela distância (HSV)

Para resolver este problema a equipa passou para a deteção da bola através de redes neurais desenvolvidas por outra dissertação tal como indicado na secção [4.1.2.2](#). Essa deteção é demonstrada na Figura 91, sendo também possível observar a distância entre o plano da câmara à bola em milímetros e a percentagem de fiabilidade. Nesta imagem a bola foi colocada a 2700 milímetros do plano da câmara, sendo o resultado retornado pela câmara 2690 milímetros.



Figura 91: Deteção e distância da bola (rede neuronal)

Os dois problemas que o modelo de cores HSV demonstrava foram descartados com o uso da rede. Como é possível observar na Figura 92, além de serem detetadas todo o tipo de bolas independentemente da cor, a bola também é detetada a distâncias bastante elevadas. Nesta imagem é possível analisar mais valores para as distâncias, sendo que a bola que se encontra a 10000 milímetros segundo a imagem está a 10500 milímetros, e a bola laranja foi colocada a 2500 milímetros, sendo retornada pela câmara 2478 milímetros.



Figura 92: Detecção das bolas (rede neuronal)

5.2.2 Posição da bola

Um tópico importante desta dissertação é a transformação da bola detetada na câmara para o plano cartesiano com a referência no centro da baliza. De forma a testar e perceber os resultados obtidos por parte do robô, foi criado um algoritmo que representa a posição da bola no campo.

Para testar a fiabilidade deste algoritmo a bola foi colocada em três posições diferentes com coordenadas recolhidas previamente. De forma a não ocorrerem oscilações nestes testes por causa do módulo da localização, visto que a posição da bola depende da posição do robô, o robô foi colocado e manipulado manualmente para o ponto com valor de coordenadas de 0 milímetros nas abcissas e 200 milímetros nas ordenadas segundo o referencial da baliza, com uma orientação de 0° com o eixo das ordenadas.

Para a representação da bola numa imagem do campo, foi feita uma translação do referencial da baliza para o canto do campo, como é representado na imagem c) da Figura 93. As coordenadas da bola nesta imagem são de 1300 milímetros nas abcissas e 2750 milímetros nas ordenadas. É possível observar a posição da bola na câmara (imagem a)), a posição da bola calculada representada no campo

(imagem c)) assim como as coordenadas calculadas da posição da bola em centímetros (imagem b)). Analisando os valores obtidos, o erro nas ordenadas é de 30 milímetros, enquanto que nas abcissas é de 50 milímetros.

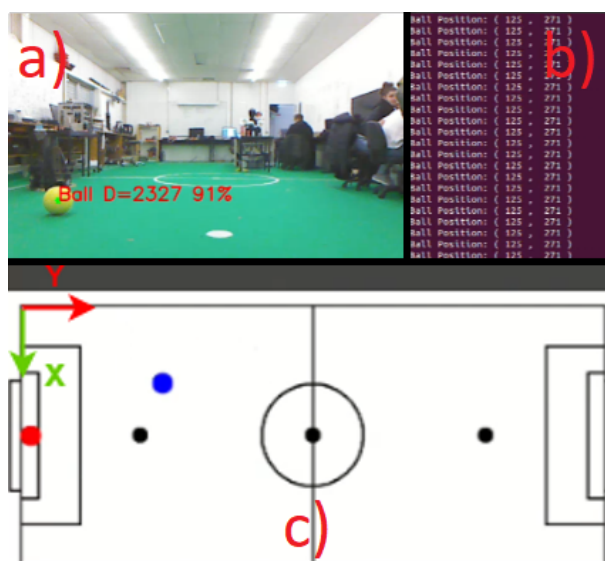


Figura 93: a) Imagem da *Microsoft Kinect*. b) Coordenadas da bola obtidas em centímetros (x,y). c) Representação da posição da bola no campo. (Posição da bola - 1º teste)

Num segundo teste, a bola foi colocada na posição (1900,4850) milímetros (Figura 94). Analisando os resultados obtidos, verifica-se que a posição nas abcissas tem um erro máximo de 10 milímetros, enquanto nas ordenadas é de 90 milímetros.

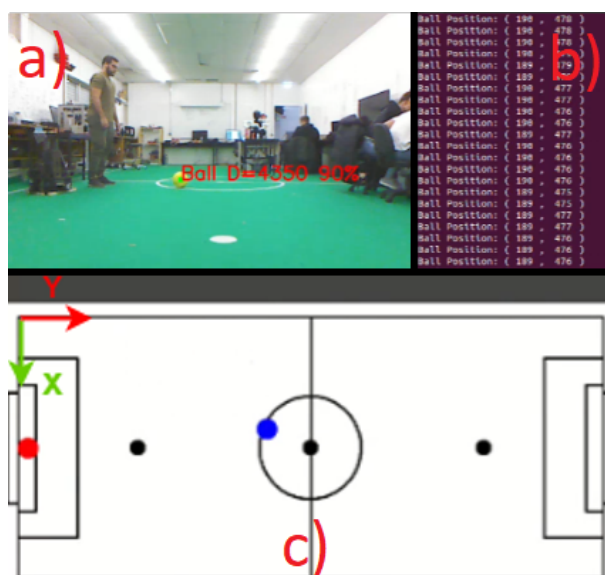


Figura 94: a) Imagem da *Microsoft Kinect*. b) Coordenadas da bola obtidas em centímetros (x,y). c) Representação da posição da bola no campo. (Posição da bola - 2º teste)

Foi feito um terceiro teste com a bola nas coordenadas (3700, 5850) milímetros (Figura 95). É possível observar que o erro nas abcissas é de 100 milímetros e nas ordenadas é de 170 milímetros.

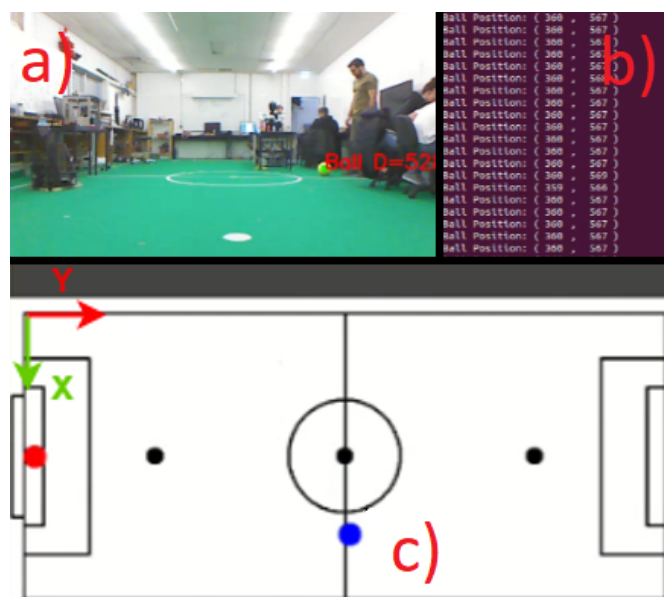


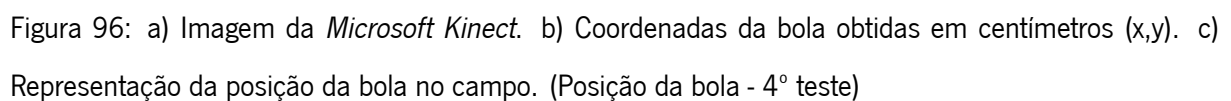
Figura 95: a) Imagem da *Microsoft Kinect*. b) Coordenadas da bola obtidas em centímetros (x,y). c) Representação da posição da bola no campo. (Posição da bola - 3º teste)

Com estes três testes é possível observar duas coisas: Primeiro, quanto a bola se encontra mais para os lados na imagem da câmara, maior é o erro nas abcissas. Segundo, quanto maior a distância à bola maior é o erro nas abcissas e ordenadas. Isto deve-se ao facto de haver duas dependências no cálculo da posição da bola.

A primeira dependência consiste no ângulo entre a bola e o robô, que irá conter mais erro conforme a bola se encontra mais nas laterais da imagem, pois o ângulo de visão da câmara não é proporcional aos pixels da mesma devido à deformação provocada pela lente. Tendo a *Microsoft Kinect* 57° de ângulo de visão horizontal e 640 pixels de resolução horizontal, caso esta fosse proporcional, os pixels que se encontrem na posição 160 horizontal seriam equivalentes a 14,25°. Devido a uma pequena deformação da imagem da *Microsoft Kinect* isto não se verifica, produzindo um pequeno erro no ângulo de no máximo 1° pelos testes efetuados na posição da bola. Este ângulo, junto com a segunda dependência, geraram um erro máximo de 100 milímetros nas abcissas.

A segunda dependência é a distância, que conforme a bola está mais distante, maior é o erro obtido dessa distância. Visto que o robô está orientado para a frente, a posição da bola no eixo das ordenadas equivale a posição do robô nas ordenadas (que foi manipulada manualmente) com a distância retornada pela câmara. Assim, os 170 milímetros de erro obtidos no terceiro teste nesse eixo, são devidos a essa distância.

Os resultados obtidos estão representados na Figura 96, sendo que o erro no eixo das abcissas é de 10 milímetros, e no eixo das ordenadas é de 20 milímetros.



5.2.3 Detecção de remates

Nos primeiros dois testes apresentados, o robô foi colocado no centro da baliza na posição (0,200) milímetros orientado para a frente. No primeiro teste, a bola foi rematada em direção ao poste que se encontra nas coordenadas (1000, 0) milímetros. Na Figura 97 é possível observar na imagem a) a imagem da *Microsoft Kinect*, na imagem c) está representada uma imagem retirada noutro ângulo sobre esse remate, e por fim na imagem b) está representada a previsão do robô no momento do remate sobre o ponto que a bola vai intercepar a baliza com essa informação por baixo em centímetros.

Neste teste, a detecção do remate calculou que a bola iria intercepar a baliza a 980 milímetros do centro da baliza para o lado direito, equivalente à posição (980, 0) milímetros, o que equivale a um erro de 20 milímetros.

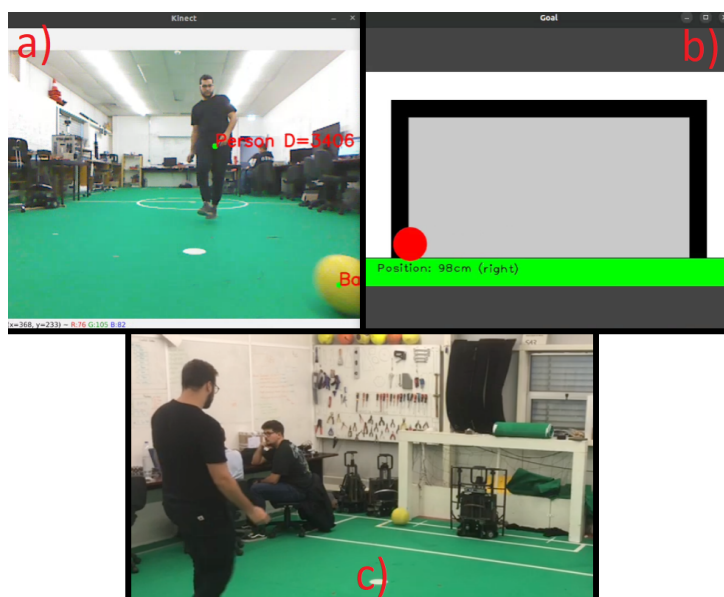


Figura 97: a) Remate efetuado visto pela imagem da *Microsoft Kinect*. b) Posição da bola após o remate na baliza representada a preto. c) Remate efetuado noutro ângulo. (1º teste)

No segundo teste a bola foi rematada para um ponto marcado no chão com uma posição de (-750,0) milímetros. Na Figura 98 é observado que a previsão calculou que a bola interceparia a baliza na posição (-710,0) milímetros, equivalente a um erro de 40 milímetros.

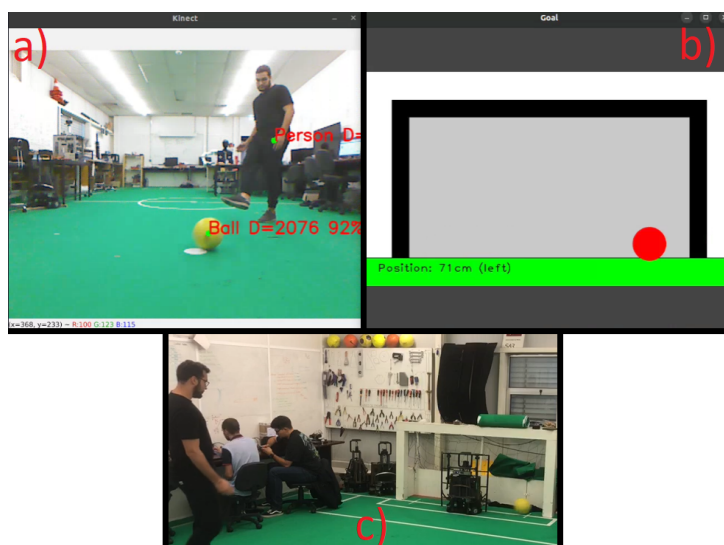


Figura 98: a) Remate efetuado visto pela imagem da *Microsoft Kinect*. b) Posição da bola após o remate na baliza representada a preto. c) Remate efetuado noutro ângulo. (2º teste)

No último teste efetuado, o robô foi colocado nas coordenadas (500,200) milímetros com uma orientação de -30° , de forma a perceber se os remates eram detetados corretamente mesmo com uma translação e rotação do robô. A bola foi rematada para o centro da baliza, coordenadas (0,0) milímetros.

O resultado obtido está representado na Figura 99 onde se observa um erro de 20 milímetros.

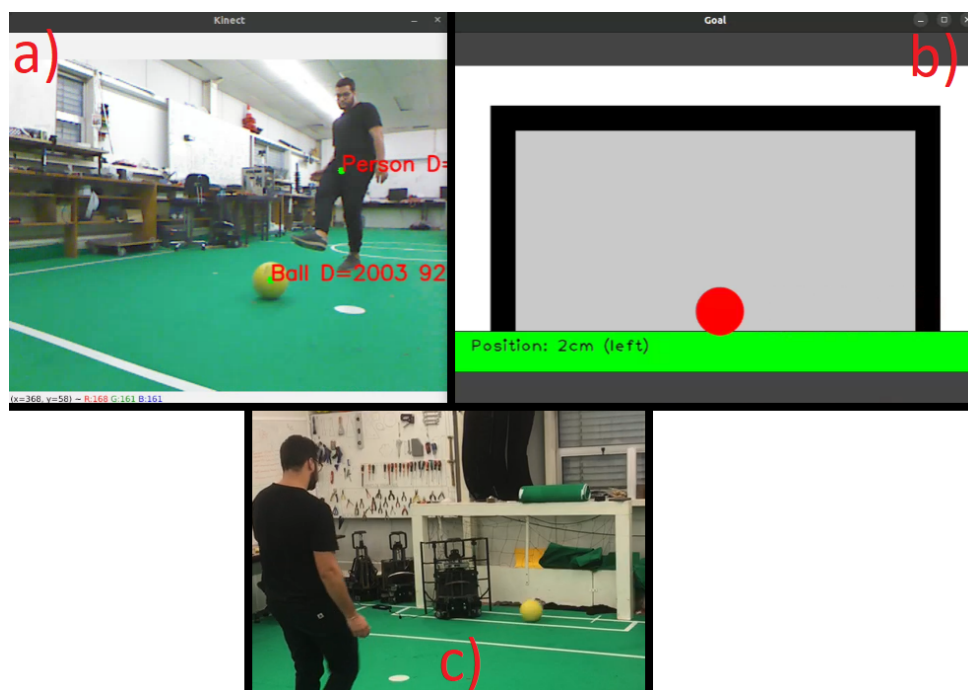


Figura 99: a) Remate efetuado visto pela imagem da *Microsoft Kinect*. b) Posição da bola após o remate na baliza representada a preto. c) Remate efetuado noutro ângulo. (3º teste)

Nestes testes é possível observar que a previsão da trajetória da bola tem uma grande eficácia pois depende da posição da bola, que segundo os testes da secção 5.2.2, não tem muito erro associado.

5.2.4 Defesa de remates

Os últimos testes deste sistema de visão estão associados à defesa dos remates que após ter uma posição desejada de deslocamento, depende apenas dos ganhos do atrator e da localização do robô.

No primeiro teste apresentado foi feito um remate para a zona central da baliza quando o robô se encontrava mais para a esquerda dessa. A posição e orientação desejada do robô é demonstrada na imagem b) da Figura 100 em milímetros, na imagem c) é apresentada a posição do robô visualmente e na imagem a) é a imagem da câmara da *Microsoft Kinect*. A posição final do robô foi (-50,200) milímetros com uma orientação de 3° , o que origina um erro de 122 milímetros na posição e de cerca de 1° na orientação.

Após vários testes e medições do erro da posição final do robô foi obtido o gráfico representado na Figura 103, demonstrando que em média o erro da posição do robô é de 172 milímetros com um desvio padrão de 28,03 milímetros. Este erro é derivado da localização pois como demonstrado na secção 5.1 esta tem algumas oscilações e erro associado.

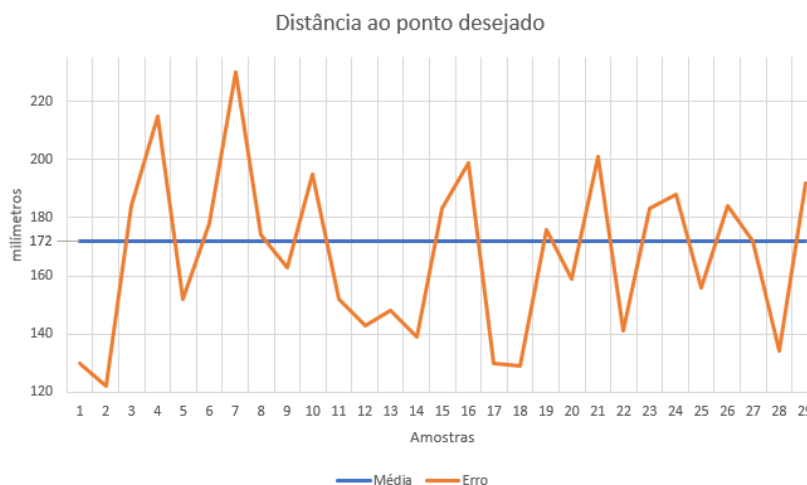


Figura 103: Gráfico do erro da posição do robô em remates

O tempo de resposta do robô aos remates no pior caso possível (guarda-redes está num poste e a bola é rematada para o outro), desde o momento em que é efetuado o remate até ao momento em que o robô chega à posição desejada é cerca de 2 segundos. O motivo deste tempo deve-se em grande parte à calibração efetuada na placa de controlo dos motores, pois a equipa optou por ter uma calibração igual entre todos os robôs, calibração essa que os fazia ter uma reação mais lenta de forma a terem um arranque mais suave. No mesmo momento que o robô deteta um remate, envia as velocidades desejadas para o baixo nível e o comando para subir a estrutura, mas nota-se que a estrutura sobe antes do arranque do robô.

5.3 Posicionamento e orientação

Os primeiros testes efetuados sobre o posicionamento do robô na baliza são sobre o cálculo das coordenadas do robô com o uso da elipse (secção 4.2). Desta forma, a bola foi colocada num ponto conhecido para se perceber quanto erro na posição desejada do robô era gerado pela elipse. A bola foi colocada nas coordenadas (-1100,2750) milímetros, e os valores de A e B da elipse são de 900 e 1400 respetivamente. Assim, segundo os cálculos matemáticos utilizados, a posição desejada que tem de ser calculada para o robô é (-482,200) milímetros.

Na Figura 104 observa-se na imagem c) a representação do campo com a elipse, a bola (círculo azul), a posição do robô (círculo vermelho), a posição desejada (círculo verde), a reta de interceção com a elipse (reta preta), a reta de movimentação do robô (reta verde) e ainda uma reta laranja que indica a interceção do ponto das abscissas obtido pela elipse com a reta fixa das ordenadas. Na imagem a) encontra-se a imagem recolhida da câmara, e na imagem b) a posição desejada calculada. Analisando essa posição desejada é possível observar que existe um erro no cálculo dessa posição de 18 milímetros para a coordenada das abscissas do robô, sendo que esse erro deriva da leitura da posição da bola.

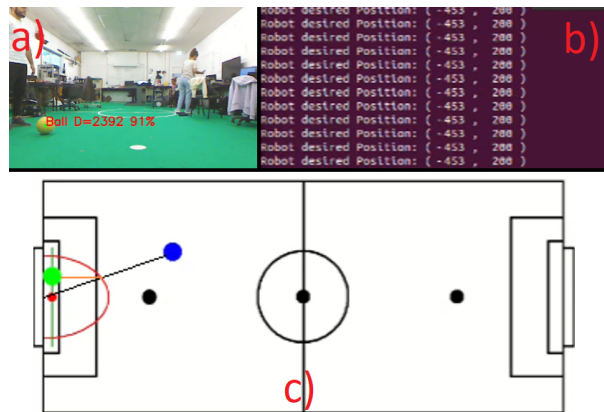


Figura 104: a) Imagem da *Microsoft Kinect*. b) Posição desejada do robô c) Representação do cálculo da posição do robô no campo (elipse). (1º teste)

No segundo teste a bola foi colocada na posição (720,2200) milímetros, onde seria obtida uma posição de (411, 200) milímetros para o robô. Na Figura 105 é observada uma variação dos valores vinda da movimentação da bola para a posição final, sendo o erro do cálculo de 10 milímetros para os últimos valores.

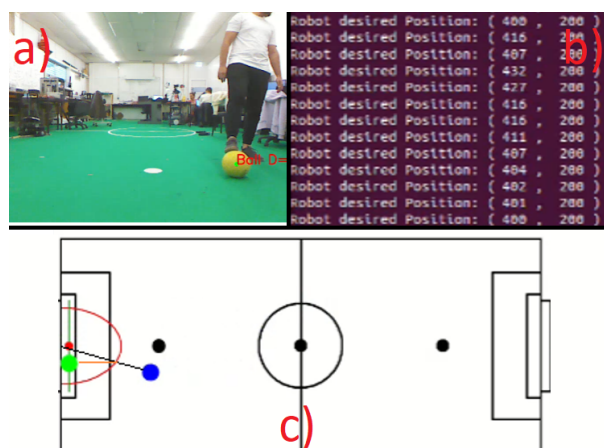


Figura 105: a) Imagem da *Microsoft Kinect*. b) Posição desejada do robô c) Representação do cálculo da posição do robô no campo (elipse). (2º teste)

O primeiro teste é apresentado na Figura 106, sendo que na imagem a) da figura está representada a imagem da *Microsoft Kinect* e na imagem b) a posição e orientação desejada para o robô. Na imagem c) está representada a posição e orientação final do robô neste posicionamento que toma os valores de (-500,250) milímetros para a posição e 23 graus para a orientação.

a)

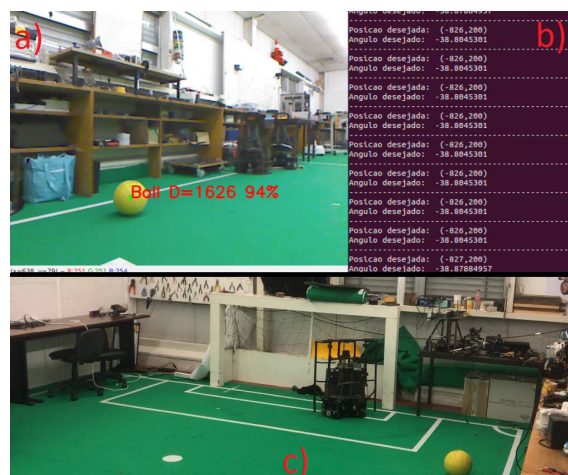
b)

Ball D=2799 92%

Posicao desejada:	(-568,200)
Angulo desejado:	-22.59834008
Posicao desejada:	(-569,200)
Angulo desejado:	-22.651064950000002
Posicao desejada:	(-570,200)
Angulo desejado:	-22.703858099999998
Posicao desejada:	(-571,200)
Angulo desejado:	-22.75671965
Posicao desejada:	(-572,200)
Angulo desejado:	-22.809649720000003
Posicao desejada:	(-573,200)
Angulo desejado:	-22.86264843
Posicao desejada:	(-574,200)
Angulo desejado:	-22.915715900000002
Posicao desejada:	(-575,200)
Angulo desejado:	-22.915715900000002

c)

No segundo teste a posição final do robô foi (-800,300) milímetros com uma orientação de -35° , o que forma um erro de 103 milímetros na posição e 3° na orientação (Figura 107).



90

No terceiro teste a posição final foi as coordenadas (400,250) milímetros com 20° de orientação, gerando um erro de 92 milímetros e 2° (Figura 108).



Figura 108: a) Imagem da *Microsoft Kinect*. b) Posição e orientação desejada. c) Posição e orientação real do robô na baliza. (Posicionamento e orientação - 3º teste)

De salientar que nestes testes foi possível observar que o robô tem momentos em que se efetuam pequenos movimentos na zona da posição desejada. Estas oscilações são devidas à localização do robô ter pequenas oscilações ao longo do tempo, o que faz variar o erro na posição desejada.

5.4 Estrutura móvel

O primeiro objetivo da estrutura móvel é garantir que a mesma não ultrapassa os 12 centímetros quando totalmente ascendida, pois caso contrário o guarda-redes não pode jogar. Na Figura 109 está representada a altura máxima que a estrutura sobe, cerca de 11 centímetros.



Figura 109: Altura máxima da estrutura móvel

Os dois testes necessários para perceber o correto funcionamento da estrutura móvel do guarda-redes são a correta ativação após receber um comando, e também que essa ativação, subida e descida da estrutura, seja efetuada dentro de um segundo. Na Figura 110 está representado o resultado da estrutura móvel a subir após o envio de um comando pela porta série do PC, sendo o tempo obtido desse teste, desde a subida até a descida da estrutura, de 0,95 segundos.

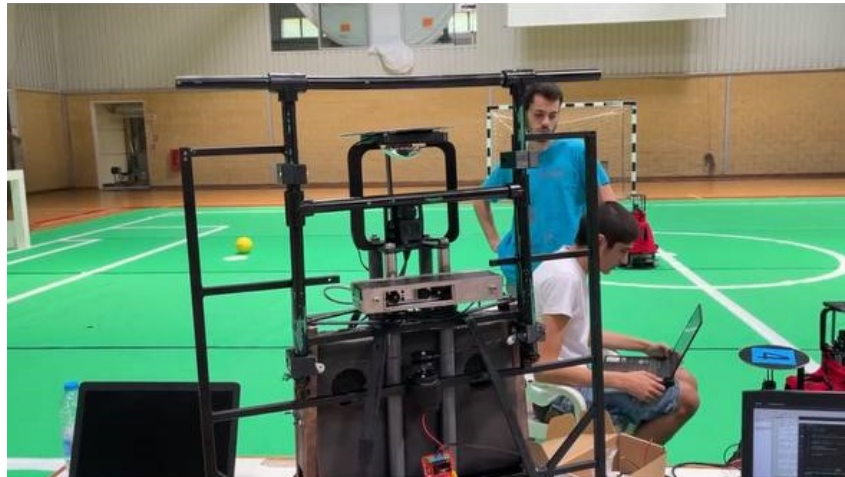


Figura 110: Estrutura móvel subida

De forma a testar a eficiência da estrutura móvel foram feitos 20 remates com o robô imóvel para a zona de ascensão da estrutura, não causando uma dependência da movimentação do robô nestes testes. Nesses 20 remates foram defendidos 19, visto que num deles a estrutura não subiu a tempo devido à proximidade do remate, demonstrando uma taxa de sucesso de 95%. Na Figura 111 está representado um desses remates.

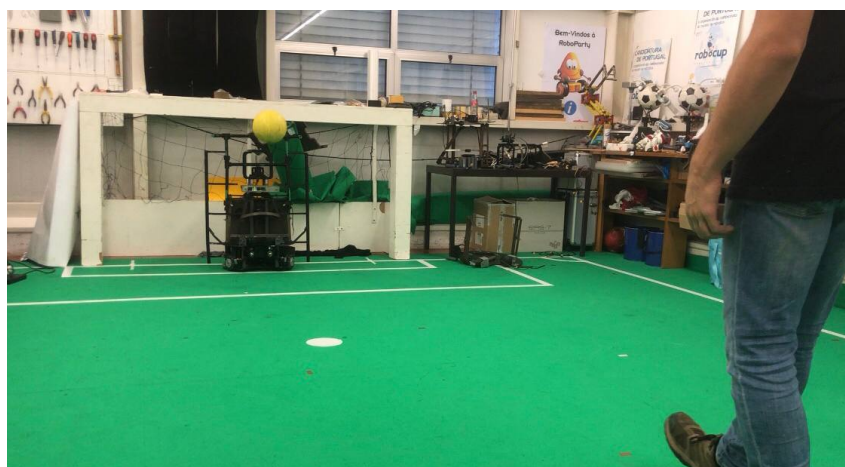


Figura 111: Defesa com a estrutura móvel

5.5 Defesa de penáltis

De forma a perceber se a estratégia da defesa dos penáltis funcionava conforme desejado, foram desenhadas no chão três trajetórias de remate que intercetam a baliza nos pontos de coordenadas (800,0), (0,0) e (-800,0) milímetros conforme demonstrado na Figura 112.



Figura 112: Teste de trajetórias no penálti

Com a bola colocada no ponto do penálti e movimentando o robô atacante para a linha da trajetória desejada, o guarda-redes tem de prever e de movimentar-se para o ponto dessa trajetória incidente com a baliza.

No primeiro teste o robô atacante foi colocado na trajetória do ponto (800,0) milímetros. Como o guarda-redes pode defender na pequena-área, a posição ideal é no centro dessa de forma a cobrir mais espaço, o que equivale a uma posição de 200 milímetros nas ordenadas. O declive da reta desta trajetória é -2,875 e a ordenada na origem equivale à distância do penálti que é 2300 milímetros ($y = -2,875x + 2300$). Assim, para uma ordenada de 200 milímetros a abcissa tem de ter o valor de 730 milímetros. O resultado obtido está representado na imagem b) da Figura 113, sendo possível observar um erro de 20 milímetros. A imagem encontra-se à escala, sendo possível observar a área que o guarda-redes irá cobrir.

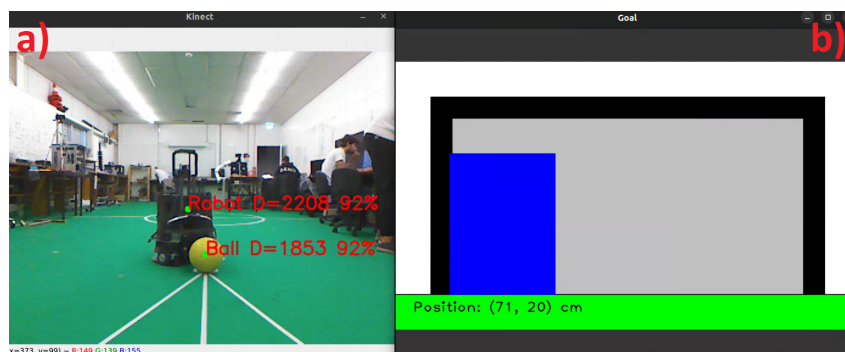


Figura 113: a) Imagem da *Microsoft Kinect*. b) Representação do robô (azul) na baliza (preto) e das coordenadas desejadas em centímetros. (Cálculo da trajetória do penálti - 1º teste)

No segundo teste o robô atacante foi movido em torno da bola onde foi verificado que a posição desejada do guarda-redes se moveu em correspondência pela baliza, sendo por fim colocado na trajetória do ponto $(-800,0)$ milímetros. Da mesma forma que na trajetória anterior, o ponto da abcissa dessa reta para um valor de ordenada de 200 milímetros equivale a -730 milímetros. O resultado obtido é representado na Figura 114 e analisando a imagem é possível observar um erro de 80 milímetros.

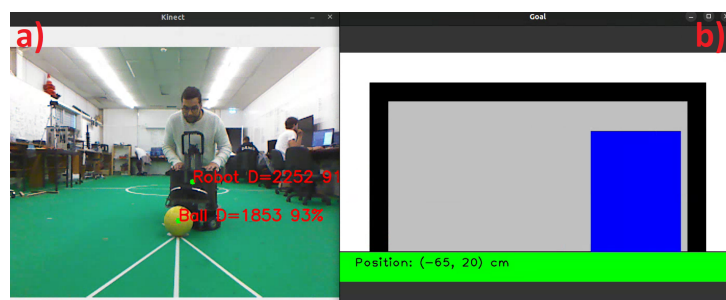


Figura 114: a) Imagem da *Microsoft Kinect*. b) Representação do robô (azul) na baliza (preto) e das coordenadas desejadas em centímetros. (Cálculo da trajetória do penáti - 2º teste)

Após o cálculo da trajetória entre a posição do robô e da bola, são demonstrados os testes da movimentação do robô para esses pontos desejados.

Na Figura 115 é possível observar o primeiro teste, sendo representado na imagem a) a imagem da *Microsoft Kinect*, na imagem b) a posição desejada calculada em milímetros e na imagem c) a posição final do robô na baliza que após ser medida tinha as coordenadas $(470,200)$ milímetros. Desta forma, a posição desejada teve um erro associado de 77 milímetros.



Figura 115: a) Imagem da *Microsoft Kinect*. b) Posição e orientação desejados para o robô. c) Posição e orientação real do robô na baliza. (Defesa do penáti - 1º teste)

Na Figura 116 está representado o segundo teste da defesa de penáltis, sendo a posição do guarda-redes representada na imagem c) equivalente à posição (-810,150), o que demonstra um erro de 64 milímetros.

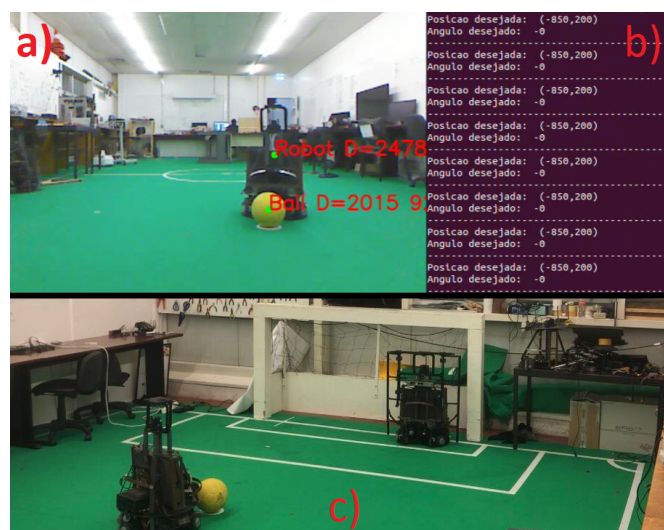


Figura 116: a) Imagem da *Microsoft Kinect*. b) Posição e orientação desejados para o robô. c) Posição e orientação real do robô na baliza. (Defesa do penálti - 2º teste)

Com estes testes é possível perceber que mesmo a posição tendo um erro associado, o robô coloca-se numa posição que torne possível interceptar a bola na baliza. A fiabilidade deste algoritmo não foi testada no LAR com os robôs atacantes a rematar, mas sim no mundial *RoboCup* no jogo de terceiro e quarto lugar da equipa, que de 10 remates o guarda-redes defendeu 5 (taxa de sucesso de 50%). Este algoritmo demonstrou-se útil mas menos efetivo que o esperado, pois após a análise dos resultados da outra equipa com este algoritmo, os *Tech United*, era expectável uma defesa de quase 100% dos penáltis. Assim como nos remates, esta percentagem de sucesso deve-se à calibração utilizada na placa de controlo dos motores como explicado na secção 5.2.4.

Capítulo 6

Competições

Este capítulo descreve a experiência das duas competições em que a equipa participou, o Festival Nacional de Robótica e o mundial de robótica, *RoboCup 2023*. Estas duas competições foram demonstradas bastante relevantes para o desenvolvimento da dissertação, tanto a nível de evolução como de testes.

6.1 Festival Nacional de Robótica

O Festival Nacional de Robótica 2023 que ocorreu em Tomar, contou com a inscrição de uma grande equipa que compete no *RoboCup*, sendo a equipa campeã nos últimos anos, os *Tech United*. A equipa *LAR@MSL* foi a única equipa portuguesa a participar neste evento.

Com esta participação foi possível observar uma das grandes complicações nestas competições, os problemas de rede. O acesso aos computadores instalados nos robôs era feito através da rede, e visto que nestes eventos centenas de pessoas utilizam a mesma, tornou-se uma tarefa complicada, atrasando o progresso da equipa.

Um dos grandes problemas encontrados nesta competição foi a deteção da bola por parte do guarda-redes, sendo que foi nesse momento que a equipa decidiu migrar da deteção da bola com o modelo de cores *HSV* para o uso de redes neuronais. Para além disto, este evento foi muito relevante para retirar fotos para o *DataSet* da rede neuronal que se iria desenvolver.

Outro problema detetado nesta competição foi o sobreaquecimento dos computadores devido ao calor e uso das camisolas de jogo. Para contornar esta situação foi decidido fazer aberturas em certas zonas das camisolas para permitir a dissipação do calor.

No que diz respeito ao trabalho desenvolvido nesta dissertação, foi possível neste evento validar todo o *software* desenvolvido até esse momento. Foi validada a deteção e defesa de remates por parte do guarda-redes, assim como o posicionamento do mesmo na baliza. Neste evento a ideia da orientação

ainda seria de que o guarda-redes estaria sempre orientado para a frente, o que se demonstrou impossível devido ao campo de visão reduzido da *Microsoft Kinect*, fazendo o mesmo nunca detetar remates pois não conseguia ver a bola. Assim surgiu a ideia de existir algum tipo de orientação para a zona da bola. A estrutura móvel e a defesa de penáltis ainda não tinham sido desenvolvidas, sendo que foi entre este evento e o mundial *RoboCup2023* que essas foram criadas.

Com a participação da outra equipa neste evento, foi possível discutir ideias e perceber evoluções que a equipa poderia efetuar, realçando que foi nesta competição e em conversa com a equipa campeã que se obteve a ideia utilizada para a estrutura móvel.



Figura 117: Festival Nacional de Robótica

6.2 RoboCup2023

O mundial de robótica *RoboCup 2023* ocorreu em Bordéus, contando com a participação de seis equipas na [MSL](#), sendo a única equipa portuguesa o *LAR@MSL*.

Esta competição iniciou-se com uma derrota contra a equipa francesa por 1 — 0. Conforme o evento foi passando e a equipa efetuando todos os seus jogos, foi alcançado um lugar nas meias finais. No jogo das meias finais foi defrontada a equipa campeã dos últimos anos, os *Tech United*, onde se saiu com a derrota. Posteriormente, essa equipa acabou por vencer a final e vencer o título novamente.

No último jogo, o *LAR@MSL* defrontou a equipa francesa para o jogo dos terceiros e quartos. Este jogo acabou por ir a penáltis sendo que o guarda-redes de 10 penáltis conseguiu defender 5 com o uso do algoritmo de defesa penáltis.

No que diz respeito a esta dissertação, todos os restantes módulos foram testados com sucesso. Os mais importantes, por não terem sido testados no evento anterior, eram a orientação, a estrutura móvel e a defesa de penáltis. A orientação demonstrou-se eficaz pois diferente do evento anterior, o guarda-redes apenas deixava de ver a bola caso um robô se coloca-se à frente. A estrutura móvel também demonstrou bons resultados pois não houve golos sofrido por cima do guarda-redes. A eficácia dos penáltis já foi referida, demonstrando uma taxa de sucesso abaixo do expectável.

Os resultados obtidos neste evento foram fruto de um ano inteiro de trabalho por parte de todos os membros da equipa.



Figura 118: RoboCup 2023

Capítulo 7

Conclusão e trabalho futuro

Este capítulo descreve as conclusões sobre o trabalho desenvolvido nesta dissertação e algumas ideias que poderiam ser implementadas no futuro.

7.1 Conclusão

Nesta dissertação foram cumpridos todos os objetivos iniciais, o desenvolvimento de um guarda-redes capaz de detetar a bola, detetar e defender remates, posicionar-se corretamente na baliza, participar em jogadas de equipa e desenvolver uma estrutura móvel dentro das regras da competição. Pode ser verificado nos capítulos 4 e 5 que todos estes objetivos foram cumpridos e testados, tendo sido tudo colocado em prática nas competições que a equipa participou.

Com todos os testes foram verificados algoritmos desenvolvidos que eram possíveis de melhorar. No desenvolvimento da deteção da bola, começou-se por um método pouco robusto que acabou por ser alterado para um que se demonstrou perfeito para o que era desejado.

A deteção e defesa de remates foi demonstrada capaz e com pouco erro associado, mas infelizmente, foi verificada uma dificuldade do guarda-redes no seu tempo de resposta devido à calibração da placa de controlo dos motores como explicado na secção 5.2.4, sendo que o mesmo passou a detetar remates relativamente cedo (dentro do possível da *Microsoft Kinect*) mas demorava a reagir com o seu movimento. Devido a esta reação, acabou por sofrer alguns golos nas competições.

O posicionamento e orientação do robô na baliza acabou por se tornar uma das implementações mais úteis e eficientes, pois demonstrou um erro associado bastante reduzido e graças a isso permitiu que o guarda-redes fosse capaz de defender bolas apenas com o seu posicionamento. Graças à orientação, o robô conseguia ter maior parte do tempo contacto visual com a bola usando a *Microsoft Kinect*.

A estrutura móvel foi uma das atualizações mais importantes pois demonstrou resultados acima do esperado com uma taxa de sucesso de 95%, sendo que grande parte dos robôs adversários remata por

cima do guarda-redes.

A possibilidade do guarda-redes participar como atacante ficou presente no robô, pois o robô tem a função de guarda-redes através da habilidade que lhe é atribuída pela *BaseStation*, sendo neste caso a habilidade de defesa. Caso a *BaseStation* defina que quer o guarda-redes como atacante, apenas precisa de lhe colocar uma habilidade diferente, desenvolvida por outra dissertação.

Um trabalho extra desenvolvido foi a defesa de penáltis que se demonstrou útil na última competição que o robô participou. Este algoritmo não é perfeito, pois o guarda-redes não defende todos os penáltis, mas foi bastante útil, visto que fez o mesmo defender penáltis que não seriam possíveis apenas com a deteção e defesa do remate.

Importa salientar que além deste trabalho de *software* cumprido, foi também efetuado bastante trabalho na estrutura dos robôs, *hardware* e *software* de baixo nível, sendo que desta forma se criou uma estrutura muito mais estável, robusta e preparada para competições futuras, estando perto do nível das equipas atuais.

Concluindo, foram feitas atualizações no guarda-redes para poder jogar ao nível atual da liga, e provou ser bastante robusto, capaz de participar nos jogos entre equipas de grande calibre e grande desenvolvimento. É de salientar que este trabalho de dissertação é apenas um módulo de um conjunto de *software* desenvolvido pela equipa *LAR@MSL*, sendo alguns desses módulos necessários para esta dissertação, pois sem eles seria impossível o guarda-redes jogar. Com o desenvolvimento de todos estes módulos, foi possível alcançar não só os objetivos do guarda-redes, mas sim de toda a equipa, obtendo resultados melhores que os expectáveis, pois foi a primeira equipa a jogar com apenas 10 meses de desenvolvimento.

7.2 Trabalho futuro

Durante o desenvolvimento deste projeto e com a experiência obtida nas competições que a equipa participou, surgiram algumas ideias para evolução do guarda-redes da equipa, não só a nível de *software*, mas da mecânica do robô também.

A nível de *software*, uma grande evolução para este robô é o mesmo efetuar em alguns momentos saídas da baliza para cortar a bola a jogadores, não como jogador atacante mas sim como guarda-redes. Já é possível, através da implementação desenvolvida, o guarda-redes ser um atacante, pois apenas é necessário atribuir uma habilidade ao guarda-redes diferente da de defesa (habilidade do guarda-redes) através da estratégia da *BaseStation*. A ideia para o futuro corresponde ao guarda-redes ser capaz de concretizar uma espécie de mancha como os guarda-redes de futebol, que em situações de um para um

com o atacante, atacam a bola e deixam a baliza. Isto poderá ser feito exatamente da mesma forma com o robô, que em situações de um para um com outro, é feito um cálculo de onde é o ponto que o guarda-redes precisa de se posicionar de forma a cobrir a baliza toda ao adversário e não ter espaço para colocar a bola por cima do guarda-redes.

A nível de mecânica e *hardware*, existem duas evoluções que são bastante recomendadas. Primeiramente, mover a zona do PC para um sítio diferente, permitindo que a estrutura do guarda-redes seja colocada novamente ao meio, obtendo o seu tamanho máximo.

A segunda evolução seria o desenvolvimento da estrutura móvel para os lados e não apenas para cima. A ideologia desta seria exatamente a mesma da já existente, sendo que a única alteração seria que a estrutura voltaria à sua posição inicial através do uso de elásticos e não com a força gravítica.

Bibliografia

- [1] Minoru Asada, Hiroaki Kitano, Itsuki Noda, and Manuela Veloso. Robocup: Today and tomorrow—what we have learned. *Artificial intelligence*, 110(2):193–214, 1999.
- [2] Robocupsoccer: Middle size league. [Online] <https://www.robocup.org/leagues/6>. (Accessed: Dec-2022).
- [3] Modelo de cores hsv. [Online] <https://jpsvsite.wordpress.com/2017/02/10/modelos-rgb-cmyk-hsv-yuv/>. (Accessed: Dec-2022).
- [4] Robôs cambada. [Online] <http://robotica.ua.pt/cambada/index.php?a=27f5e15b6af3223f1176293cd015771d>. (Accessed: Dec-2022).
- [5] C Bernardo, J António, R Neves, P Dias, J Azevedo, N Lau, R Dias, F Amaral, E Pedrosa, et al. *CAMBADA'2015: Team Description Paper*. CAMBADA, (2015).
- [6] António José Ribeiro Neves, Alina Trifan, Paulo Dias, and José Luís Azevedo. Detection of aerial balls in robotic soccer using a mixture of color and depth information. In *2015 IEEE International Conference on Autonomous Robot Systems and Competitions*, pages 227–232. IEEE, 2015.
- [7] Ferry Schoenmakers, Koen Meessen, Yanick Douven, Harrie van de Loo, Dennis Bruijnen, Wouter Aangenant, Bob van Nijhuijs, Matthias Briegel, Patrick van Brakel, Jordy Senden, et al. Tech united eindhoven middle size league winner 2016. In *RoboCup 2016: Robot World Cup XX 20*, pages 542–553. Springer, 2017.
- [8] ST Kempers, DMJ Hameeteman, RM Beumer, JP van der Stoel, JJ Olthuis, WHTM Aangenant, PEJ van Brakel, M Briegel, DJH Bruijnen, R van den Bogaert, et al. Tech united eindhoven middle size league winner 2022. In *Robot World Cup*, pages 337–348. Springer, 2022.
- [9] Jaap Vos. *Qualification for World Championship Montreal, Canada 2018*. Falcons, (2018).

- [10] Tiago João Fernandes Maia. Desenvolvimento de sistema de controlo de movimento para plataformas móveis autónomas. Master's thesis, Universidade do Minho (Portugal), 2019.
- [11] F Ribeiro, G Lopes, H Ribeiro, P Silva, T Maia, R Roriz, A Gomes, and N Ferreira. Minho team '2016: Team description paper. *University of Minho, Guimarães, Portugal*, 2016.
- [12] A Fernando Ribeiro, Ivo Moutinho, Nino Pereira, Fernando Oliveira, José Fernandes, Nuno Peixoto, and Antero Salgado. Cooperative behaviour of specific tasks in multi-agent systems and robot control using dynamic approach. 2006.
- [13] A Fernando Ribeiro, Ivo Moutinho, Pedro Silva, Carlos Fraga, and Nino Pereira. Three omni-directional wheels control on a mobile robot. *Control 2004*, 2004.
- [14] A Fernando Ribeiro, Gil Lopes, João Costa, João Pedro Rodrigues, Bruno Pereira, João Silva, Sérgio Silva, Paulo Ribeiro, and Paulo Trigueiros. Minho msl: a new generation of soccer robots. 2011.
- [15] A Fernando Ribeiro, Paulo Braga, Jorge Monteiro, Ivo Moutinho, Pedro Silva, and Victor Silva. New improvements of minho team for robocup middle size league in 2003. *University of Minho, Guimarães, Portugal*, 2004.
- [16] Robocupsoccer. [Online] <https://www.robocup.org/domains/1>. (Accessed: Dec-2022).
- [17] Robocup@home. [Online] <https://athome.robocup.org/>. (Accessed: Dec-2022).
- [18] Tiago Ribeiro, Fernando Gonçalves, Marco Luís, Eduardo Fernandes, Rui Silva, António Dias, Nuno Oliveira, Diogo Martins, Syed Fawad, Gil Lopes, et al. Lar@home 2023 team description paper. *University of Minho, Guimarães, Portugal*, 2023.
- [19] Robin Soetens, René van de Molengraft, and Bernardo Cunha. Robocup msl-history, accomplishments, current status and challenges ahead. *RoboCup 2014: Robot World Cup XVIII 18*, pages 624–635, 2015.
- [20] Nanning Zheng, George Loizou, Xiaoyi Jiang, Xuguang Lan, and Xuelong Li. Computer vision and pattern recognition, 2007.
- [21] Sabine Süsstrunk, Robert Buckley, and Steve Swen. Standard rgb color spaces. In *Proc. IS&T/SID 7th Color Imaging Conference*, volume 7, pages 127–134, 1999.

- [22] Radu Horaud, Miles Hansard, Georgios Evangelidis, and Clément Ménier. An overview of depth cameras and range scanners based on time-of-flight technologies. *Machine vision and applications*, 27(7):1005–1020, 2016.
- [23] Alvy Ray Smith. Color gamut transform pairs. *ACM Siggraph Computer Graphics*, 12(3):12–19, 1978.
- [24] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural networks*, 61:85–117, 2015.
- [25] R Dias, F Amaral, J Azevedo, M Azevedo, B Cunha, P Dias, N Lau, E Pedrosa, AJR Neves, et al. *CAMBADA'2017: Team Description Paper*. CAMBADA, (2017).
- [26] L Almeida, JL Azevedo, G Corrente, B Cunha, A Ferdowsi, JP Figueiredo, P Fonseca, S Lopes, R Marau, N Lau, et al. *CAMBADA'2016: Team Description Paper*. CAMBADA, (2016).
- [27] About tech united. [Online] https://www.techunited.nl/en/about_us. (Accessed: Dec-2022).
- [28] Omni-3md - manual de software. [Online] https://botnroll.com/omni3md/downloads/OMNI-3MD_Manual_Software_1v90.pdf. (Accessed: Feb-2023).
- [29] A Fernando Ribeiro, Gil Lopes, Bruno Martins, Pedro Silva, Ivo Moutinho, Paulo Braga, and Nino Pereira. Magnetically impelled kicker for robotic football in msl roboc. *Revista Robótica*, 2016.
- [30] Hélder Ribeiro, Pedro Silva, Ricardo Roriz, Tiago Maia, Rui Saraiva, Gil Lopes, and A Fernando Ribeiro. Fast computational processing for mobile robots' self-localization. In *2016 International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pages 168–173. IEEE, 2016.
- [31] Barney N. and Gillis A. Go programming language. [Online] <https://www.techtarget.com/searchitoperations/definition/Go-programming-language>. (Accessed: Jun-2023).
- [32] Ivan Culjak, David Abram, Tomislav Pribanic, Hrvoje Dzapo, and Mario Cifrek. A brief introduction to opencv. In *2012 proceedings of the 35th international convention MIPRO*, pages 1725–1730. IEEE, 2012.
- [33] What is gocv? [Online] <https://gocv.io/>. (Accessed: Jun-2023).

- [34] Roboflow. [Online] <https://roboflow.com/>. (Accessed: Mar-2023).
- [35] MSL Technical Committee. *Rules and Regulations for 2023*. Middle Size Robot League, (2023).