CONTROLO DIFUSO DE UM AGV

Dário Jorge dos Reis Osório

Mestrado em Engenharia Electrotécnica e de Computadores
Área de Especialização de Automação e Sistemas
Departamento de Engenharia Electrotécnica
Instituto Superior de Engenharia do Porto

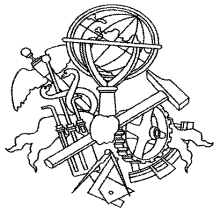
2010

Este relatório satisfaz, parcialmente, os requisitos que constam da Ficha da Unidade Curricular de Tese/Dissertação, do 2º ano, do Mestrado em Engenharia Electrotécnica e de Computadores

Candidato: Dário Jorge dos Reis Osório, Nº 1050356, 1050356@isep.ipp.pt

Orientação científica: Ramiro Barbosa, rsb@isep.ipp.pt

Co-orientação científica: Manuel Silva, mss@isep.ipp.pt



Mestrado em Engenharia Electrotécnica e de Computadores

Área de Especialização de Automação e Sistemas

Departamento de Engenharia Electrotécnica

Instituto Superior de Engenharia do Porto

12 de Novembro de 2010

Agradecimentos

Ao nível académico gostaria de agradecer ao Eng. Ramiro Barbosa e ao Eng. Manuel Silva por toda a compreensão, disponibilidade e ajuda que demonstraram durante a elaboração deste projecto. Gostaria também de agradecer ao Eng. Lino Figueiredo por me ter disponibilizado material para este trabalho.

A nível pessoal, gostaria de agradecer ao meu amigo Micael Cerqueira, por me ter elaborado os suportes dos motores do robô, e ao meu amigo Ricardo Brochado que me disponibilizou o seu programador para que eu pudesse elaborar o projecto trabalhando a qualquer hora.

Por último, mas não menos importante gostaria de agradecer à minha família, mais precisamente, aos meus pais, à minha esposa, à minha irmã, ao meu primo e aos meus sogros por me motivarem nos momentos em que tudo parece estar contra nós.

A admiração que possuo por todos aqueles que mencionei é grande, e de um modo simples mas bastante sincero, agradeço-lhes, pois sem eles seria impossível realizar este trabalho.

Dário Osório

Resumo

Neste trabalho pretende-se introduzir os conceitos associados à lógica difusa no controlo de sistemas, neste caso na área da robótica autónoma, onde é feito um enquadramento da utilização de controladores difusos na mesma. Foi desenvolvido de raiz um AGV (*Autonomous Guided Vehicle*) de modo a se implementar o controlador difuso, e testar o desempenho do mesmo. Uma vez que se pretende de futuro realizar melhorias e/ou evoluções optou-se por um sistema modular em que cada módulo é responsável por uma determinada tarefa. Neste trabalho existem três módulos que são responsáveis pelo controlo de velocidade, pela aquisição dos dados dos sensores e, por último, pelo controlador difuso do sistema.

Após a implementação do controlador difuso, procedeu-se a testes para validar o sistema onde foram recolhidos e registados os dados provenientes dos sensores durante o funcionamento normal do robô. Este dados permitiram uma melhor análise do desempenho do robô. Verifica-se que a lógica difusa permite obter uma maior suavidade na transição de decisões, e que com o aumento do número de regras é possível tornar o sistema ainda mais suave. Deste modo, verifica-se que a lógica difusa é uma ferramenta útil e funcional para o controlo de aplicações. Como desvantagem surge a quantidade de dados associados à implementação, tais como, os universos de discurso, as funções de pertença e as regras. Ao se aumentar o número de regras de controlo do sistema existe também um aumento das funções de pertença consideradas para cada variável linguística; este facto leva a um aumento da memória necessária e da complexidade na implementação pela quantidade de dados que têm de ser tratados. A maior dificuldade no projecto de um controlador difuso encontra-se na definição das variáveis linguísticas através dos seus universos de discurso e das suas funções de pertença, pois a definição destes pode não ser a mais adequada ao contexto de controlo e torna-se necessário efectuar testes e, consequentemente, modificações à definição das funções de pertença para melhorar o desempenho do sistema. Todos os aspectos referidos são endereçados no desenvolvimento do AGV e os respectivos resultados são apresentados e analisados.

Palavras-Chave

Lógica Difusa, AGV, Controlo Difuso, Controlo de Velocidade, Rede CAN.

Abstract

This work introduces the concepts associated with fuzzy logic control systems, in this case applied to autonomous robotics, where was made a contextualization of fuzzy controllers in this area. An AGV (Autonomous Guided Vehicle) was developed from scratch in order to implement the fuzzy controller, and test its performance. In order to incorporate future improvements, it was adopted a modular structure where each module is responsible for a particular task. In this work there are three modules that are responsible for velocity control, the acquisition of sensor data, and by the implementation of the fuzzy controller.

After implementing the fuzzy controller, the system was tested in order to validate the collected and recorded data from the sensors during normal operation of the robot. This data enabled the analysis of the performance of the robot. Fuzzy logic allows a smooth transition between the decisions, and increasing the number of rules the system becomes smoother. Thus, we conclude that fuzzy logic is a useful and functional tool for control applications. The disadvantage arises in the amount of data associated with the implementation, such as the universes of discourse, the membership functions and rules. When the number of control rules increases, there is also an increase of the membership functions for each linguistic variable considered, which leads to an increase in the required memory and complexity in the implementation because of the amount of data that must be addressed. The major difficulty in the design of a fuzzy controller is the definition of the linguistic variables through their universes of discourse and their membership functions, since the definition of these may not be the most appropriate to the context of control and it becomes necessary to make tests, and consequently, changes in the definition of the membership functions to improve the system performance. All these issues were addressed in the design of the AGV, and the obtained results are presented and analyzed.

Keywords

Fuzzy Logic, AGV, Fuzzy Control, Velocity Control, CAN Network.

Índice

Índice de Figuras

Índice de Tabelas

Acrónimos

ACK – *Acknowledge*

AGV – *Autonomous Guided Vehicle*

AuRA – *Autonomous Robot Architecture*

CAN – *Controller Area Network*

CAN_H – *Controller Area Network High*

CAN_L – *Controller Area Network Low*

CDB – *Context-dependent Blending*

CLA – *Centroid of Largest Area*

COA – *Centroid of Area*

CRC – *Cyclic Redundancy Check*

DAMN – *Distributed Architecture for Mobile Navigation*

DEE – Departamento de Engenharia Electrotécnica

DLC – *Data Lenght Code*

DOI – *Degree of Importance*

EOF – *End of Frame*

FLC – *Fuzzy Logic Controller*

IDE – *Identifier Extension*

IFS – *Inter Frame Space*

ISEP – Instituto Superior de Engenharia do Porto

MEEC – Mestrado em Engenharia Electrotécnica e de Computadores

NRZ – *Non Return Zero*

OSI – *Open Systems Interconnection*

PWM – *Pulse-width modulation*

REC – *Receiver Error Counter*

RISC – *Reduced Instruction Set Computer*

RTR – *Remote Transmition Request*

SAMBA – *Sensors, Actuators, Markers, Behaviors, and Arbiters*

SMPA – *Sense-Model-Plan-Act*

SOF – *Start of Frame*

SRR – *Substitute Remote Request*

TEC – *Transmiter Error Counter*

Tq – *Time Quantum*

1. INTRODUÇÃO

No contexto da Unidade Curricular de Tese/Dissertação, do 2º ano, do Mestrado em Engenharia Electrotécnica e de Computadores (MEEC), área de especialização de Automação e Sistemas, do Departamento de Engenharia Electrotécnica (DEE), do Instituto Superior de Engenharia do Porto, desenvolveu-se a Tese com o tema "Controlo difuso de um AGV" onde se abordou a temática da lógica difusa e dos controladores difusos aplicados à robótica.

O objectivo principal é o estudo dos conceitos associados à lógica difusa, de modo a desenvolver um robô com um controlador difuso que fosse responsável pela tomada de decisão do sistema. A construção do AGV (*Autonomous Guided Vehicle*) permitiu efectuar testes ao sistema de controlo difuso de modo retirarem-se algumas conclusões e validar o sistema de controlo proposto.

No decorrer deste documento será feito um enquadramento da utilização da lógica difusa em sistemas de tomada de decisão, na área da robótica autónoma, de forma a melhor compreender qual o lugar da mesma nesta área.

Devido à necessidade de desenvolver um AGV de raiz, existiu a necessidade de desenvolver outros módulos de *hardware* fundamentais para a integração e interacção de todos os módulos do sistema no AGV.

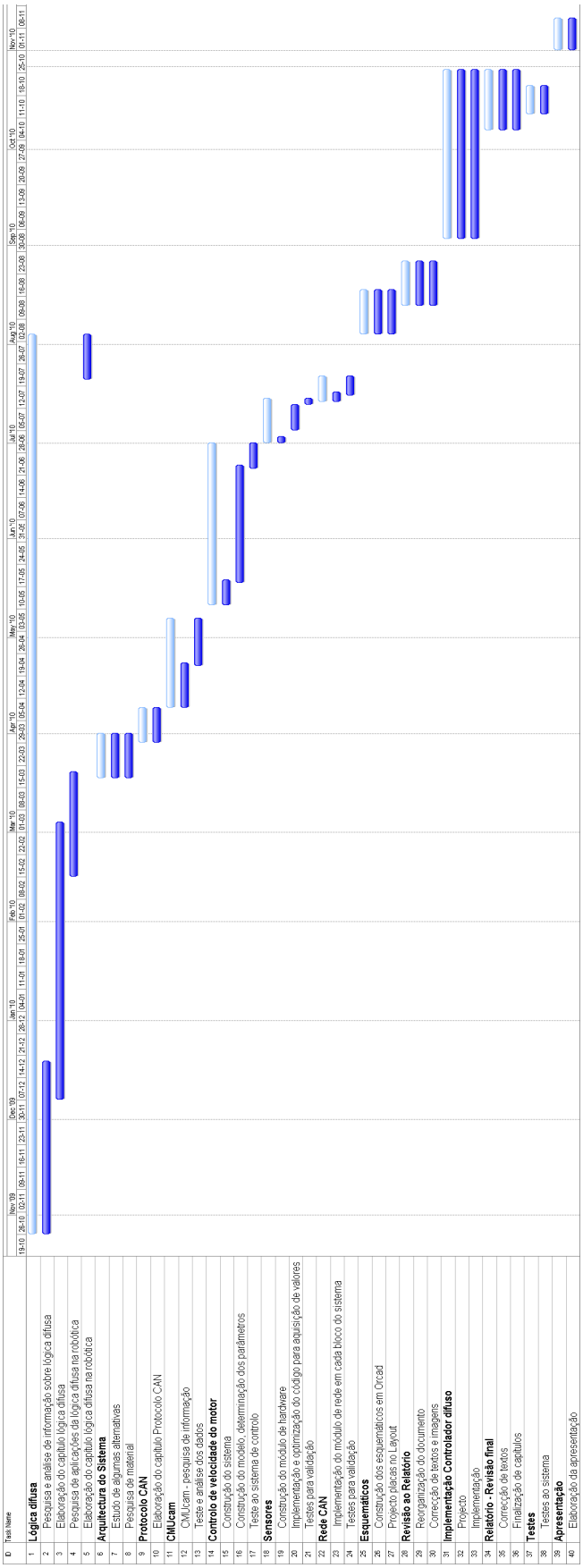
1.1. CALENDARIZAÇÃO

Para a execução deste trabalho foi necessário efectuar várias tarefas tais como: estudo dos conceitos associados à lógica difusa, e seu enquadramento na área da robótica; idealização do sistema e implementação do mesmo, entre outras que derivam do desenvolvimento do projecto. A sua prossecução conduziu à calendarização apresentada na Tabela 1.

1.2. ORGANIZAÇÃO DO RELATÓRIO

O relatório é constituído por seis capítulos sendo eles: Introdução, Lógica difusa, Lógica difusa na robótica, Arquitectura do sistema, Desenvolvimento do AGV e Conclusões e propostas de desenvolvimentos Futuros. No primeiro capítulo é feita uma introdução ao projecto e aos objectivos do mesmo. No Capítulo 2 são introduzidos os conceitos associados à lógica difusa e referidos os controladores Mamdani e Takagi-Sugeno. No capítulo seguinte, denominado de Lógica difusa na robótica, são introduzidos alguns conceitos relacionados com as arquitecturas de controlo, com o objectivo de fazer um enquadramento da utilização da lógica difusa na robótica autónoma, sendo aqui apresentados alguns exemplos. No quarto capítulo é apresentada a arquitectura do sistema desenvolvido e fundamentadas algumas das escolhas tomadas durante a idealização do mesmo. Faz-se também uma pequena introdução teórica ao módulo de controlo da velocidade. No Capítulo 5 apresenta-se a fase de desenvolvimento dos vários módulos do sistema, desde o módulo de aquisição dos sensores, passando pelo módulo de controlo de velocidade para os motores e pela implementação da rede CAN, até ao desenvolvimento do controlador difuso. No capítulo seguinte, o sexto, são apresentadas as análises dos resultados, as conclusões e a proposta de se efectuar o estudo referente à possibilidade de integração da CMUcam neste trabalho. No Anexo A são apresentadas informações mais detalhas sobre o protocolo CAN. No Anexo B são apresentadas características sobre a CMUcam, desde o *software* até as funcionalidades fornecidas pelo dispositivo. Por último, no Anexo C, são apresentados os esquemas das placas de circuito impresso desenvolvidas para este projecto.

Tabela 1 Calendarização do projecto.



2. LÓGICA DIFUSA

O controlo por lógica difusa é conseguido através de regras com um significado associado, que podem assumir graus de verdade diferentes. Tradicionalmente, as decisões na computação são rígidas e a decisão baseia-se em valores lógicos como verdadeiro ou falso, sim ou não, 1 ou 0. A lógica difusa permite efectuar uma graduação da verdade desde o valor zero até ao valor 1. Deste modo, na lógica difusa um acontecimento pode ser “mais ou menos verdade”.

A lógica difusa é utilizada em inúmeros trabalhos na área de sistemas autónomos para melhorar o desempenho dos robôs; por exemplo, esta pode ser usada em robôs futebolistas para melhorar aspectos como o posicionamento sem bola, tomada de decisão e remate. Uma vez que a lógica difusa permite abordar sistemas que lidam com incertezas, esta é cada vez mais uma ferramenta importante para a melhoria do desempenho de sistemas autónomos.

Actualmente podem-se também encontrar controladores difusos em produtos como máquinas de lavar, câmaras de vídeo e carros; são ainda utilizados na indústria, no controlo de comboios, etc.

Neste Capítulo será abordada a evolução histórica da lógica difusa, as características que lhe estão associadas e a arquitectura do controlador difuso.

2.1. INTRODUÇÃO

A lógica difusa (*Fuzzy logic*) é baseada na teoria dos conjuntos difusos. Tradicionalmente uma preposição lógica tem dois extremos: ou é "completamente verdadeira" ou é "completamente falsa". Na lógica difusa uma premissa varia em função do grau de verdade, podendo assumir assim valores entre zero e um, o que a leva a ser proporcionalmente verdadeira ou falsa.

No controlo por lógica difusa as heurísticas e as regras são a linguagem de controlo, em oposição ao controlo convencional em que se utilizam as equações diferenciais. Isto não significa que o controlo difuso não necessite das equações diferenciais, porque os conceitos do controlo clássico são muito úteis no controlo por lógica difusa. "O controlo por lógica difusa é uma alternativa prática a vários desafios no controlo de aplicações para construir controladores não lineares utilizando informações heurísticas" [1]. O objectivo da lógica difusa é gerar uma saída a partir de um conjunto de entradas imprecisas, com ruído, ou até mesmo originadas por falhas.

As principais características do controlo difuso em relação a outras técnicas são:

- Robustez, dado que não requer entradas precisas;
- Fácil alteração porque é baseada em regras;
- Controlo de sistemas não lineares sem modelo matemático;
- Solução mais rápida e barata em alguns casos;
- Implementação relativamente fácil em microprocessadores.

Devido à complexidade dos sistemas reais existe uma certa dificuldade de os modelar, simular e desenvolver o sistema de controlo. Este facto motivou a utilização da lógica difusa no controlo de sistemas. "O controlo por lógica difusa fornece uma metodologia formal para representar, manipular e implementar o conhecimento heurístico humano sobre como controlar um sistema" [1].

Quando se pretende controlar um processo físico complexo, geralmente são efectuados um conjunto de procedimentos, tais como, desenvolver o modelo dinâmico do sistema utilizando um modelo matemático, ou uma simplificação do mesmo. Deve-se considerar o modelo matemático do sistema para projectar o sistema de controlo e fazer uma análise da simulação para estudar o seu comportamento. Após a implementação do controlador, deve-se avaliar o desempenho em malha fechada do sistema de modo a validar o controlo do mesmo.

O controlo por lógica difusa permite uma outra abordagem ao problema, em que, em primeiro lugar, se deve adquirir conhecimento intuitivo de como se efectua o melhor controlo do processo, e essa informação irá fazer parte do controlador difuso.

O conhecimento humano permite assim definir um conjunto de regras, que irão ser integradas num controlador por lógica difusa, e este emulará as tomadas de decisão de um humano para controlar o processo. A informação proveniente das heurísticas pode surgir de pessoal especializado, como engenheiros da área de controlo com conhecimento no estudo, análise, projecto e implementação de algoritmos para o controlo de sistemas que advêm de modelos matemáticos, ou até, a partir do conhecimento que o operador adquiriu através de uma aprendizagem contínua do processo.

A lógica difusa permite adicionar alguma inteligência ao processo, uma vez que o computador infere uma acção a partir de conjuntos de regras "***Se…Então…***" [2]. A lógica difusa é computar com palavras.

A lógica difusa permite que pessoas não especializadas desenvolvam sistemas de controlo, devendo ser esta uma das principais razões para o seu sucesso [2].

2.2. HISTÓRIA

O conceito de lógica difusa surgiu no inicio da década de 60 do século XX, por Lofty Zadeh, mas só em 1965 é que este publicou o primeiro artigo sobre conjuntos difusos [2]. Foi na década de 70 que surgiram as primeiras aplicações na área de controlo automático desenvolvidas por Mamdani [3]. As primeiras experiências no controlo de processos industriais são datadas de 1975, onde foi demonstrado pelo Queen College, em Londres, um controlador difuso simples que era capaz de efectuar o controlo de uma máquina a

vapor de forma eficiente. Foi desenvolvida, na mesma altura, a primeira aplicação industrial implementada pela indústria Dinamarquesa de cimento F.L. Smidth Corp.

Em 1984, Sugeno desenvolveu um controlador difuso responsável pela tarefa de estacionar um automóvel. No seguimento deste trabalho, criou também um controlador difuso com auto-aprendizagem, capaz de construir as suas regras a partir da monitorização do processo. Mais tarde, em 1987, os controladores difusos foram utilizados no controlo da paragem e arranque das composições do metro de Tóquio e, a partir daí, foram utilizados em aplicações domésticas, como máquinas de lavar roupa, frigoríficos, máquinas fotográficas e câmaras de vídeo, ar condicionado, etc [2]. Apareceram também, cada vez mais, em aplicações industriais, como o controlo de elevadores dos grupos Hitachi e Toshiba, em veículos autónomos, robôs móveis da NASA e IBM, controlo de motores (Hitachi), ventilação de túneis urbanos (Toshiba), controlo de tráfego urbano, etc [3]. Na indústria automóvel também foram utilizados em transmissões automáticas de veículos da Nissan e Lexus, na injecção electrónica, suspensão activa e travões anti-bloqueantes.

Em 1989, deu-se início ao programa "*The LIFE –The Laboratory for International Fuzzy Engeneering*" no Japão, financiado pelo governo do Japão e por empresas de todo o mundo (cerca de 50 no total). Este programa foi dividido em cinco projectos principais, relacionados com processamento de imagem (*image understanding*), memória associativa difusa (*fuzzy associative memory*), computação difusa, interfaces inteligentes e robótica inteligente [2].

Pode-se então concluir que os controladores difusos tiveram aceitação por parte do mercado, existindo nos dias de hoje cerca de 1000 patentes que já foram colocadas em prática. Aparecem agora países, como a Alemanha e os EUA, na corrida com o Japão pelo desenvolvimento de sistemas controlados segundo a lógica difusa.

Para uma maior abrangência dos principais acontecimentos relacionados com a lógica difusa, apresentam-se alguns dos marcos históricos do seu desenvolvimento na Tabela 2.

2.3. CONCEITOS ASSOCIADOS À LÓGICA DIFUSA

A lógica difusa permite utilizar o conhecimento humano, com a incerteza associada ao mesmo, para gerar decisões, pois permite caracterizar uma variável linguística, quantificando o seu universo. As conclusões resultam de um mecanismo de inferência, que

relaciona os conceitos utilizados. A representação do conhecimento é feita através da teoria dos conjuntos difusos e das funções características associadas a estes conjuntos.

Tabela 2 História da lógica difusa [2].

Ano	Acontecimento	Referências
1965	Primeiro artigo sobre conjuntos difusos	Zadeh (1965)
1972	Fundamentos sobre controlo difuso	Zadeh (1972)
1973	Aproximação linguística	Zadeh (1973)
1974	Controlador de lógica difusa	Assilian and Mamdani (1974)
1976	Sistema de controlo de aquecimento de água por lógica difusa	Kickert and van Nauta Lemke (1976)
1977	Permutador de calor	Østergaard (1977)
1980	Controlo de processos de fabrico de cimento	Holmblad and Østergaard (1982)
1983	Controlo de comboios	Yasunobu *et al*. (1983)
1984	Controlo de estacionamento do modelo de um carro	Sugeno *et al*. (1989)
1985	*Chip* difuso	Togai and Watanabe (1985)
1987	Controlo de paragem e arranque de composições de Metros Urbanos (Sendai, Tóquio).	Yasunobu *et al*. (1983)
1989	Aplicações domésticas com lógica difusa vendidas no Japão	
1989	Início do projecto “The LIFE”, no Japão	
1990	Aprendizagem de regras por redes neuronais.	Kosko (1992)
1990	Controlador hierárquico	Østergaard (1990), (1996)

Se se considerar a teoria clássica dos conjuntos, um elemento do universo pode estar ou não contido num dado conjunto. Em relação aos conjuntos difusos existe um grau de pertença, que é obtido a partir de uma função característica real, designada por função de pertença, em que consoante o peso de cada elemento de um dado conjunto se lhe atribuí o valor entre 0 e 1.

A lógica difusa permite adquirir, de forma clara e precisa, os conhecimentos que irão ser utilizados para controlar o sistema, pois possibilita a manipulação simultânea de parâmetros numéricos e de informações de linguagem, ultrapassando assim o problema associado à linguagem natural, que contém expressões com significados imprecisos que podem conter em si próprias, definições diferentes.

2.3.1. VARIÁVEIS LINGUÍSTICAS

Na lógica difusa existem variáveis linguísticas que são definidas por funções de pertença, em que cada uma representa um valor ou conceito que a variável pode assumir, e às quais são atribuídos termos linguísticos apropriados.

Apenas a título de exemplo, considera-se uma variável linguística denominada por "*velocidade*": esta é definida por três funções de pertença, que definem três conjuntos difusos; são eles "*Baixa*", "*Moderada*" e "*Alta*". Pode-se ver, na Figura 1, o gráfico que apresenta as três funções de pertença que definem a variável linguística "*velocidade*".

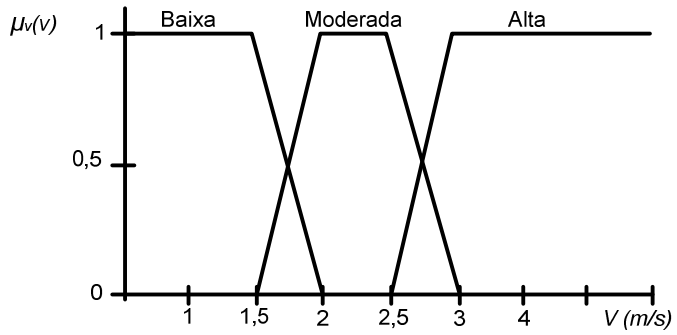


Figura 1 Funções de pertença para "Baixa", "Moderada" e "Alta" velocidade.

O conjunto de valores que uma variável linguística pode assumir designa-se de conjunto de termos, em que cada valor do conjunto de termos é um valor linguístico, ou termo, definido no universo de discurso, representado na Figura 1 pelo eixo das abcissas. Uma variável linguística assume um valor linguístico, associado a um conjunto difuso definido no universo de discurso.

Verifica-se que cada uma das funções de pertença define um intervalo de valores de velocidade, em que os graus de verdade ou de pertença (representados no eixo das ordenadas), de cada função de pertença, variam dentro do mesmo intervalo.

Existem modificadores (ou advérbios) que permitem alterar o valor destas variáveis, tais como, por exemplo, os termos "*muito*", "*pouco*", "*não muito*" e "*mais ou menos*". É complexo caracterizar precisamente o efeito de um modificador mas, por exemplo, no caso do modificador "*muito*", este tem um efeito intensificador enquanto o termo "*pouco*" tem um efeito oposto.

Quando se verifica a utilização destes modificadores pode ser necessário definir um outro conjunto difuso diferente para representar melhor a declaração modificada pelo termo modificador.

A par com os modificadores surgem algumas operações como: a Concentração, a Expansão, a Intensificação e a Potência.

As operações como a Concentração, quando aplicadas a uma função de pertença base que caracteriza um conceito, originam uma nova função de pertença obtida a partir da função de pertença base. Verifica-se que a operação de Concentração reduz ligeiramente o grau de verdade da nova função de pertença, quando o grau de verdade dos elementos é baixo, e reduz significativamente o grau de verdade da nova função de pertença, quando o grau de verdade dos elementos é alto. A expressão desta operação é:

$$\mu_{CON(A)}(x) = \left(\mu_A(x)\right)^2 \tag{1}$$

Se se considerar a existência de um conjunto difuso "*temperaturas altas*", pode-se usar esta operação de modo a criar um conjunto "*temperaturas muito altas*".

No caso da operação de Expansão, esta tende a produzir uma expansão grande quando o grau de verdade desses elementos é baixo e uma baixa expansão quando o grau de verdade é alto. Esta é representada pela equação:

$$\mu_{EXP(A)}(x) = \left(\mu_A(x)\right)^{0.5} \tag{2}$$

Esta operação pode ser utilizada quando é necessário criar um conjunto difuso "*temperaturas mais ou menos médias*" a partir do conjunto "*temperaturas médias*".

A Intensificação destina-se a aumentar o grau de verdade para valores maiores que 0,5 e a diminuir o mesmo para valores menores que 0,5. Esta operação pode ser útil quando se pretende criar o conjunto "*temperaturas realmente médias*". As expressões que definem esta operação são:

$$\mu_{INT(A)}(x) = 2\left(\mu_A(x)\right)^2 \text{ para } 0 \le \mu_a(x) \le 0{,}5 \tag{3}$$

$$\mu_{INT(A)}(x) = 1 - 2\left(1 - \mu_A(x)\right)^2 \text{ para } 0{,}5 < \mu_a(x) \le 1 \tag{4}$$

Por último, a operação Potência, utiliza-se quando existe a necessidade de criar um conjunto difuso "*temperaturas muito muito altas*", onde a expressão que a representa é:

$$\mu_{POT(A)}(x) = \left(\mu_A(x)\right)^n \text{, com } n = 3,4,5 \ldots \quad (5)$$

As operações de intersecção e união de conjuntos são representadas pelos conectivos "*e*" e "*ou*", respectivamente. Existem também outros conectivos como, por exemplo, "*Se...Então*" (ou "*implica*"), "*Se e só se*" ou "*Não*" [4]. As variáveis linguísticas são usadas nas regras difusas, onde deduzem informações sobre uma variável contida na sua conclusão como, por exemplo, "***Se** velocidade é baixa, **então** fazer aceleração alta*".

2.3.2. CONJUNTOS DIFUSOS

A teoria clássica dos conjuntos assenta numa lógica binária, representada pela álgebra de Boole, mas a teoria dos conjuntos difusos não aborda os conjuntos da mesma forma, uma vez que esta assume que o grau de verdade de uma afirmação pode variar entre os valores 0 e 1. Deste facto surge o conceito de função de pertença; este conceito é fundamental, uma vez que são as funções de pertença que definem os conjuntos difusos, mapeando os elementos no intervalo [0, 1]. Estes elementos assumem um valor que indica o grau de pertença desse elemento ao respectivo conjunto.

Como se sabe, a linguagem verbal é menos precisa que a "linguagem numérica". Então, utilizam-se as variáveis linguísticas na tentativa de descrever de forma aproximada fenómenos complexos ou mal definidos. Desta forma, as variáveis difusas, que não são mais que termos linguísticos, irão expressar os conceitos e conhecimentos utilizados na comunicação humana. Uma das principais razões do sucesso da lógica difusa em sistemas inteligentes é a possibilidade de combinar variáveis linguísticas (conceitos) e numéricas, permitindo uma boa articulação entre ambos os domínios.

As variáveis difusas são expressas dentro de um domínio. Geralmente, é definido por um especialista (duma dada área) que o define e realiza a fragmentação difusa da variável linguística que está a ser analisada.

Uma vez que já se abordaram nesta secção os conceitos de variável linguística, universo de discurso, grau de verdade e função de pertença, faz agora sentido abordar as operações que são possíveis de efectuar sobre os conjuntos difusos.

2.3.2.1. OPERAÇÕES COM CONJUNTOS DIFUSOS

As operações que se podem efectuar sobre os conjuntos difusos são semelhantes às operações que se efectuam sobre os conjuntos clássicos. Para se comparar os conjuntos difusos deve-se ter em mente que igualdade e inclusão são definidas pelo significado das funções de pertença.

Vamos considerar dois conjuntos difusos, A e B, definidos num universo de discurso U.

Os conjuntos difusos A e B são iguais (igualdade) se, e só se, tiverem a mesma função de pertença para todo o *x*; então:

$$A = B \equiv \mu_A(x) = \mu_B(x) \tag{6}$$

O conjunto difuso A é um subconjunto de B (inclusão), se, e só se, a função de pertença do conjunto A for menor ou igual à função de pertença de B, para todos os valores de *x*. Então, vem:

$$A \subseteq B \equiv \mu_A(x) \leq \mu_B(x) \tag{7}$$

Considerando as operações lógicas existentes para os conjuntos clássicos, como a intersecção, a união e complemento, a sua aplicação aos conjuntos difusos gera um significado específico, que será abordado em seguida para cada um dos casos [2].

No caso da união, e considerando os mesmos dois conjuntos difusos A e B, a união difusa de A e B é:

$$A \cup B \equiv \left\{ \left(x, \mu_{A \cup B}(x) \right) \mid x \in U \; and \; \mu_{A \cup B}(x) = \max \left\{ \mu_A(x), \mu_B(x) \right\} \right\} \tag{8}$$

Ao analisar a equação, verifica-se que união do conjunto difuso A com B, equivale a obter os valores máximos da união das duas funções de pertença.

Em termos de raciocínio, acontece algo semelhante no caso da intersecção, mas com resultado oposto, em que:

$$A \cap B \equiv \left\{ \left(x, \mu_{A \cap B}(x) \right) \mid x \in U \; and \; \mu_{A \cap B} = \min \left(\mu_A(x), \mu_B(x) \right) \right\} \tag{9}$$

Ao analisar a equação, verifica-se que a intersecção do conjunto difuso A com B, equivale a obter os valores mínimos da intersecção das duas funções de pertença.

Em relação ao complemento de um conjunto difuso, verifica-se também uma semelhança quando comparado com a lógica clássica, em que:

$$\overline{A} \equiv \left\{ \left(x, \mu_{\overline{A}}(x) \right) \mid x \in U \text{ and } \mu_{\overline{A}} = 1 - \mu_A(x) \right\} \tag{10}$$

Constata-se, à semelhança com a lógica clássica, que o complemento de um conjunto difuso A, equivale ao complemento da sua função de pertença (μ_A), ou seja, $1 - \mu_A(x)$. Analisando os gráficos apresentados, na Figura 2, pode-se constatar o que foi mencionado anteriormente.

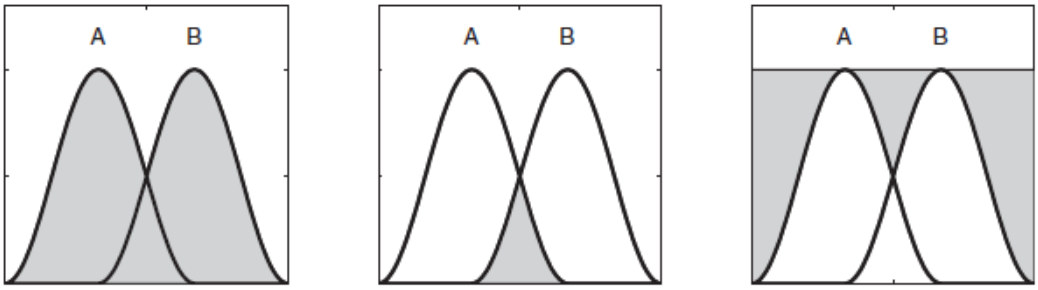


Figura 2 Reunião, intersecção e complemento dos conjuntos difusos A e B, respectivamente [2].

A criação dos conjuntos difusos vai depender da definição da sua função de pertença num universo de discurso. Esta deve representar o comportamento dos elementos, e as especificações do conjunto difuso, organizando-os com graus de verdade no intervalo [0, 1].

2.3.3. INFERÊNCIA DIFUSA

A lógica difusa considera os conjuntos difusos como preposições difusas, em que cada uma é uma declaração que define um valor para uma determinada variável linguística. Chama-se gama ou intervalo, aos possíveis valores que as variáveis linguísticas podem assumir no universo de discurso.

A título de exemplo, uma preposição difusa pode assumir a seguinte forma: "X *é A*", considerando que A é um conjunto difuso pertencente ao universo de discurso. Uma regra difusa que relacione duas preposições pode ser, por exemplo, "*Se X é A Então Y é B*". Verifica-se que a regra define uma ligação entre as duas preposições.

O processo de inferência difusa permite, na conclusão da regra, definir uma certeza que é consequência da avaliação da premissa da regra.

Este processo é representado na Figura 3, onde se identificam duas entradas que serão avaliadas segundo as premissas das regras que darão origem a conclusões que são uma consequência da avaliação. Deste modo, a inferência determina as reacções provenientes da aplicação das regras, em resposta a um determinado estímulo; em seguida, reúne os resultados para obter uma descrição global que permitirá posteriormente, na fase de desfuzificação ou colapsagem, encontrar o valor real a aplicar nas saídas [3].

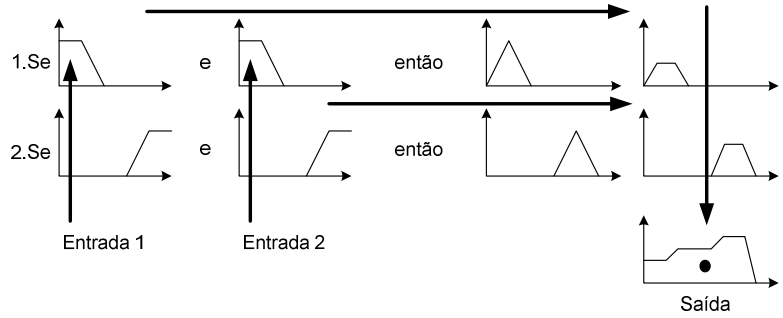


Figura 3 Diagrama do processo de inferência [3].

Na secção seguinte será feita uma abordagem mais detalhada de um controlador difuso, onde serão abordados os blocos constituintes do mesmo e as várias técnicas utilizadas nestes controladores.

2.4. CONTROLADOR DE LÓGICA DIFUSA

Um controlador difuso é, na sua essência, um controlador que incorpora o raciocínio humano através das regras, onde está implementado o conhecimento que indica como se consegue controlar o processo de modo a mantê-lo nas características desejadas.

O projecto de um controlador difuso baseia-se na formulação de regras estabelecendo uma relação entre as variáveis de entrada e as variáveis de controlo a partir das variáveis linguísticas.

A principal vantagem associada ao uso de um controlador difuso é a possibilidade de formular regras baseadas na experiência e na intuição do operador de um determinado processo. Já a principal desvantagem do controlador difuso é a dificuldade em especificar os parâmetros do controlador na forma de linguagem difusa, existindo dificuldade em

definir claramente o universo de discurso, o conjunto de condições da variável linguística e o mapa que relaciona o universo de discurso com a variável linguística.

Numa primeira fase, irá ser feita uma abordagem à constituição do controlador lógico difuso; em seguida, abordar-se-ão as várias teorias de controlo existentes para implementar um controlador lógico difuso.

O controlador difuso actua sobre o processo, através da aquisição das variáveis do processo, sua avaliação consoante as regras implementadas e determinação das variáveis de controlo. Assim, o controlador consegue actuar sobre o processo, realizando alterações na(s) sua(s) entrada(s), para um correcto e mais conveniente desempenho na saída do processo.

Como se observa na Figura 4, o controlador difuso é constituído por quatro blocos distintos; são eles:

- A interface de "*fuzificação*", ou difusão que, é responsável pela conversão dos valores reais em difusos, para que estes sejam interpretados e comparados com as regras;
- O "*mecanismo de inferência*", que avalia quais as regras que irão ser utilizadas para o controlo, avaliando a sua validade; e da aplicação das mesmas resultam conclusões difusas;
- A interface de "*desfuzificação*", ou colapsagem, que tem a responsabilidade de converter a resposta difusa obtida em valores numéricos de saída e a escala desses valores para o domínio dos sinais de controlo. A saída deste bloco é um valor de controlo real, deixando assim o domínio difuso;
- O bloco designado por "*Base de Conhecimento*" é constituído por dois blocos, o bloco da "*base de dados"* e o bloco "*base de regras*". Esta base de conhecimento é utilizada quando se faz o mapeamento das variáveis de entrada em variáveis de saída, recorrendo-se às regras. Este módulo contém as definições das funções de pertença para cada variável de estado e de controlo, bem como as regras difusas;

  i. A "*base de dados*" está incorporada no bloco "*base de conhecimento"* e contém a informação necessária ao funcionamento dos módulos de fuzificação e desfuzificação, na forma de funções de pertença e de factores de escala;
  ii. A "*base de regras*" é também parte integrante do bloco "*base de conhecimento"* e contém em forma de regras o conhecimento humano de controlo do processo.

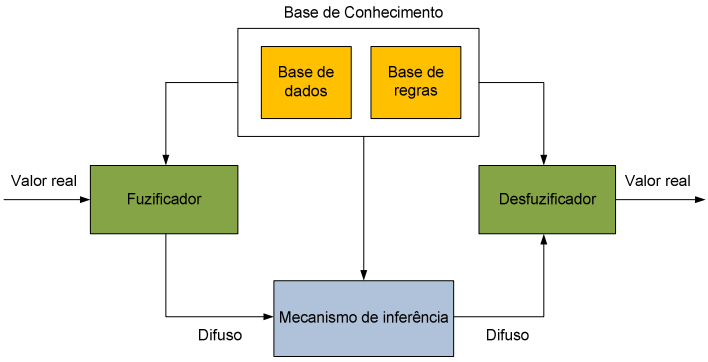


Figura 4 Blocos constituintes do controlador difuso [3].

Ao analisar os módulos de controlo difuso pode-se vê-lo como um sistema artificial de tomada de decisão, que compara a saída do processo com a entrada de referência e decide quais devem ser as entradas do processo de modo a atingir os objectivos de desempenho.

Para projectar um sistema de controlo por lógica difusa deve-se reunir informação de como o sistema artificial de tomada de decisão deve agir num sistema de malha fechada. Em muitos casos, esta informação pode ser adquirida a partir do conhecimento que um humano possui sobre como se deve efectuar o controlo das tarefas, enquanto noutras a engenharia de controlo surge para compreender a dinâmica do sistema e escrever um conjunto de regras que indicam como controlar o sistema.

Basicamente estas regras abordam o sistema do seguinte modo: "**Se** a saída do processo e a entrada de referência estão a ter determinado comportamento **então** a saída do controlador (ou entrada do processo) deve ter determinado valor".

Este conjunto de regras são do tipo "***Se (antecedentes) Então (consequentes)"*** e são colocadas no bloco *"Base de Conhecimento"*, mais propriamente na base de regras.

No controlo difuso utilizam-se as regras para definir como se controla o sistema, em vez de se utilizarem equações diferenciais como nos sistemas de controlo convencionais. A limitação fundamental está presente na capacidade humana de realizar um controlo com elevado desempenho [1].

No projecto de um sistema com controlo difuso devem-se, em primeiro lugar, escolher as entradas e as saídas e, em seguida, determinar o processamento mais adequado para as entradas e o pós-processamento para as saídas. Por último, deve-se proceder ao projecto de

cada um dos quatro componentes do controlador difuso. Existem vários métodos para efectuar a desfuzificação.

Depois de todas as entradas e saídas estarem definidas, pode-se especificar o sistema do controlo difuso. Verifica-se que a escolha das entradas e saídas do controlador impõe restrições no resto do projecto do controlo difuso do processo. Se a informação necessária para o controlador não for fornecida, haverá pouco esperança em ser possível desenvolver um bom conjunto de regras. Além disso, mesmo que exista informação suficiente para tomar decisões de controlo, deve-se garantir que o controlador é capaz de afectar convenientemente as variáveis de controlo do processo.

Pode-se concluir que a escolha das entradas e saídas do processo são uma parte fundamental do projecto do sistema de controlo.

2.4.1. FUZIFICAÇÃO

Ao processo de adquirir um valor numérico da entrada e enquadrá-lo numa gama de valores que são definidas por uma variável linguística, ou por um conjunto difuso, chama-se fuzificação ou difusão. De uma maneira simples, o processo de fuzificação é a acção de obter os valores de uma variável de entrada e encontrar os valores numéricos da função de pertença que são definidos para aquela variável. Pode-se encarar a função de pertença como uma codificação dos valores de entrada. Depois da codificação, esta informação será usada no mecanismo de inferência.

2.4.2. RELAÇÃO MECANISMO DE INFERÊNCIA/BASE DE CONHECIMENTO

Com as variáveis de entrada codificadas na lógica difusa, o processo de inferência começa com a tarefa de comparação e determinação das regras que irão ser usadas. O processo de inferência envolve geralmente dois passos [1]:

1. As premissas das regras são comparadas com as variáveis de entrada do controlador, que determina quais as regras que se devem aplicar para a actual situação. Este processo, designado de correspondência, envolve a determinação do grau de verdade que cada regra aplica, e tipicamente, leva-se mais em conta as recomendações provenientes de regras que implicam maior certeza à situação actual;

2. As conclusões (de que acções serão tomadas) são determinadas usando as regras que se aplicam à situação actual. Estas são caracterizadas com um conjunto difuso que representa o grau de verdade que a entrada do sistema deve ter nos vários valores.

A formulação das regras tenta imitar a tomada de decisão efectuada pelos humanos, de modo a incorporar nas mesmas as experiências de operários e peritos no controlo do processo.

2.4.2.1. CONHECIMENTO EM REGRAS

Supondo que um perito humano fornece uma descrição de como se pode controlar o sistema numa linguagem natural (idioma), então procura-se utilizar esta descrição linguística e transferi-la para o controlador difuso.

Esta descrição linguística fornecida pode, geralmente, ser dividida em várias partes, sendo que as variáveis linguísticas descreverão cada variação no tempo das entradas e saídas do controlador difuso. A descrição depende de cada projectista e pode ser mais ou menos profunda.

Usa-se uma quantificação linguística para especificar um conjunto de regras que contêm, em si mesmas, o conhecimento de como o controlo deve ser efectuado. As regras base definem o comportamento a adoptar considerando as variáveis do sistema. Convém ter presente que para um sistema com n entradas e x valores linguísticos, para se abranger todas as possibilidades, teriam de se considerar x^n regras possíveis, ou seja todas as premissas possíveis para as duas entradas. Um modo conveniente para listar todas as regras possíveis, para o caso onde não existem mais de 2 / 3 entradas, é a representação tabular. A representação tabular de um conjunto de regras, como a da Figura 5, é uma hipótese de representação. A coluna da esquerda e a linha superior indicam as premissas linguístico-númericas e o corpo da tabela lista as consequências linguístico-númericas das regras.

Saída		Entrada 2				
		-2	-1	0	1	2
Entrada 1	-2	2	2	2	1	0
	-1	2	2	1	0	-1
	0	2	1	0	-1	-2
	1	1	0	-1	-2	-2
	2	0	-1	-2	-2	-2

Figura 5 Representação tabular [1].

Cada valor da tabela (-2, -1, 0, 1, 2) representa, neste caso, um conjunto difuso; por exemplo, o valor "-2" representa o conjunto difuso "*muito baixo*" e o valor "2" representa o conjunto difuso "*muito alto*". Deste modo pode-se ver que "**Se** a entrada 1 "*muito baixa*" **e** entrada 2 "*baixa*" (valor -1), **então** a acção para a saída é "*muito alta*" (valor 2)".

2.4.2.2. QUANTIFICAÇÃO DIFUSA DO CONHECIMENTO

Deve-se também quantificar o significado dos termos linguísticos usando funções de pertença (por exemplo a da Figura 6) de modo a descrever o respectivo conjunto difuso. Esta quantificação é uma parte fundamental para a tomada de decisão.

A função de pertença μ_V quantifica o grau de verdade, ou de pertença, com que a velocidade pode ser classificada como "*Baixa*". A função de pertença quantifica de uma maneira contínua quais os valores da velocidade que pertencem ao conjunto difuso "*Baixa*".

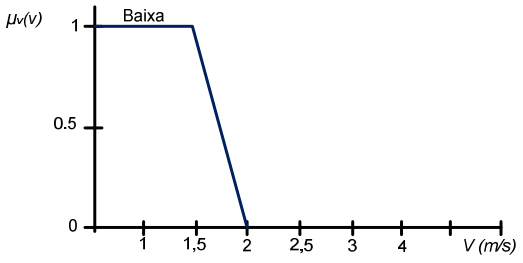


Figura 6 Função de pertença.

A função de pertença representada é apenas uma das possíveis definições, pois as funções de pertença podem assumir formas diferentes, tais como:

- Trapezoidal;
- Gaussiana;
- "Crista de tubarão";
- Triangular ou triangular enviesada;
- Rectangular;
- *Singleton.*

A partir das formas base podem surgir algumas variações, como o caso da forma triangular enviesada, representada na Figura 7.

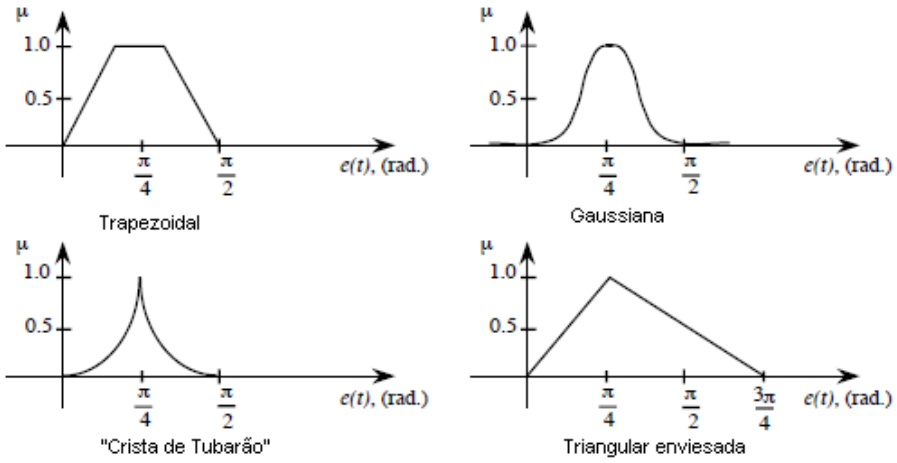


Figura 7 Algumas formas possíveis para a representação das funções de pertença [1].

Uma função de pertença que seja representada por um *Singleton* representará, em si própria, uma constante que assumirá o grau de verdade 1, como se pode ver na Figura 8.

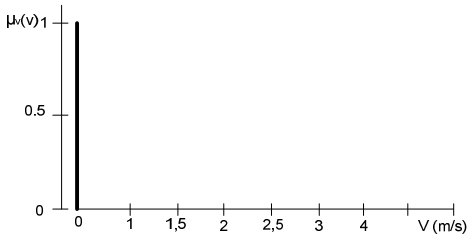


Figura 8 Representação de um conjunto difuso *Singleton*.

De modo a melhor descrever o contexto das variáveis no problema, cada aplicação pode requerer uma representação da sua função de pertença. Pode ser definida uma quantificação mais restrita para o conjunto difuso, em que a função de pertença representa um intervalo restrito com um grau de verdade, ou pertença, igual a 1 nesse intervalo, e igual a 0 fora do mesmo, como se pode ver na Figura 9.

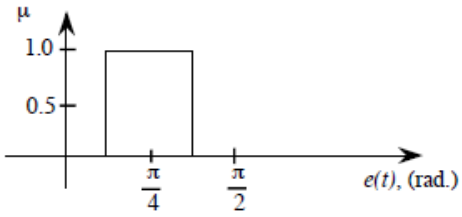


Figura 9 Representação de um conjunto restrito [1].

As funções de pertença são fundamentais para realizar a conversão dos valores reais em valores difusos, para realizar a inferência e até a desfuzificação, pois contêm em si mesmas a representação do conceito.

2.4.2.3. QUANTIFICAÇÃO DIFUSA DAS PREMISSAS

Para se efectuar a inferência deve-se quantificar cada uma das regras com lógica difusa. Assim, primeiro quantifica-se o significado das premissas das regras que são compostas por diversos termos, cada um dos quais envolve a entrada do controlador difuso, como por exemplo, "**IF** premissa1 **and** premissa2 **THEN** acção".

O operador lógico *and* combina o significado dos dois termos linguísticos, existindo várias operações possíveis de efectuar sobre os conjuntos difusos, como foi mencionado na secção 2.3.2.1.

2.4.2.4. DETERMINAÇÃO DA APLICABILIDADE DAS REGRAS

A determinação da aplicabilidade de cada regra é chamada de "*matching*" ou adequação. Para isso, o mecanismo de inferência procura quais as regras que são mais relevantes para a presente situação e, em seguida, procura combinar as recomendações de todas as regras para retornar uma única conclusão.

Ao determinar as conclusões, são efectuados passos de inferência que permitem saber que conclusões podem ser alcançadas quando as regras são aplicadas, para decidir que entrada

se deve aplicar ao sistema. Para isso considera-se cada regra independentemente, e só mais tarde é que se combinam todas as recomendações para determinar a entrada que se deve aplicar ao sistema. As conclusões obtidas por este bloco são uma descrição global da resposta do controlador e estão ainda no domínio difuso.

2.4.3. DESFUZIFICAÇÃO

A conversão das decisões, ou conclusões, em acções é da responsabilidade da interface de desfuzificação, que é o componente final do controlador difuso. A desfuzificação opera nos conjuntos difusos, produzidos pelo mecanismo de inferência, e combina os seus efeitos para fornecer a saída do controlador ou entrada do sistema mais adequada.

A desfuzificação descodifica a informação produzida pela inferência. Convém referir que os controladores do tipo Takagi-Sugeno não necessitam de desfuzificação. Assim estes métodos são destinados apenas aos controladores do tipo Mamdani. Estes dois tipos de controladores serão abordados nas secções 2.5 e 2.6.

Os métodos de desfuzificação, ou colapsagem, devem ter as seguintes propriedades [3]:

- Continuidade, em que uma pequena alteração na entrada não deve resultar numa grande alteração na saída;
- Desambiguidade, onde a resposta a uma entrada deve estar bem definida e ser única;
- Plausibilidade, em que o valor desfuzificado deve possuir um elevado valor de pertença no conjunto difuso de resposta;
- Complexidade computacional, onde o cálculo do valor desfuzificado deve ser obtido de forma eficiente e adequado ao contexto do sistema.

Existem vários métodos para efectuar a desfuzificação. Entre eles destacam-se os seguintes [5]:

1. Centro de área (CoA);
2. Média dos máximos (MoM);
3. Primeiro máximo/ Último máximo (FoM);
4. Altura ou "*Means of maxima*" (HM);
5. Centro da maior área (CLA).

Segundo Leonid Reznik [5], estes métodos apresentam as seguintes características apresentadas na Tabela 3:

Tabela 3 Propriedades de cada método de desfuzificação.

Métodos	CoA	MoM	FoM	HM	CLA
Continuidade	Sim	Não	Não	Sim	Não
Desambiguidade	Sim	Sim	Sim	Sim	Não
Plausibilidade	Sim	Não	Não	Sim	Sim
Complexidade computacional	Mau	Bom	Bom	Bom	Mau

Analisando a Tabela 3, verifica-se que os métodos MoM, FoM e HM são melhores ao nível computacional; deste modo, apenas estes serão abordados neste capítulo.

2.4.3.1. MÉDIA DOS MÁXIMOS

Este método considera a resposta na sua totalidade e calcula a média pesada dos máximos do conjunto resultante, visível a vermelho na Figura 10.

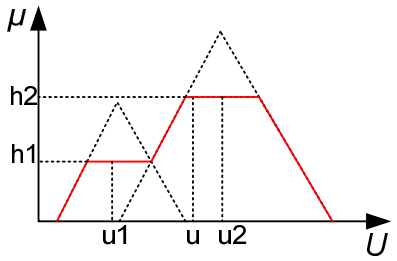


Figura 10 Método de desfuzificação Média dos máximos.

O valor de controlo será obtido considerando o somatório dos valores máximos verificados no universo de discurso. Esse somatório será dividido pelo número (n) de máximos considerados, como apresentado na equação (11).

$$u^* = \frac{\sum_{i=1}^{n} x_i}{n} \tag{11}$$

em que x_i, representa os máximos considerados.

2.4.3.2. PRIMEIRO MÁXIMO/ ÚLTIMO MÁXIMO

Este método considera apenas a resposta com maior grau de verdade e utiliza o valor mais baixo do universo (ou alto, para o método último máximo) com o maior grau de verdade do universo de discurso (Figura 11).

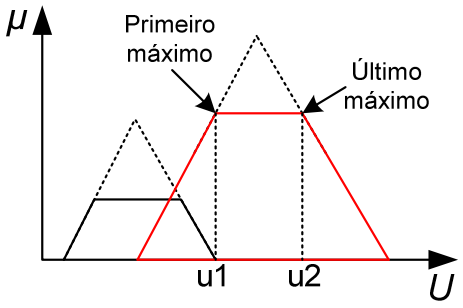


Figura 11 Método de desfuzificação Primeiro/Último máximo.

O valor a aplicar para determinar o primeiro máximo e último máximo, resultam respectivamente das equações (12) e (13).

Se se pretender utilizar o primeiro máximo, será analisada apenas a resposta com maior grau de verdade e será escolhido o valor onde se verifica o maior grau de verdade e menor valor no universo, como se pode ver na equação (12).

$$u* = inf\left\{u \in U \,|\, \mu_U(u) = hgt(u)\right\}, u \in U \tag{12}$$

Para a situação do último máximo, será analisada apenas a resposta com maior grau de verdade e será escolhido o valor onde se verifica o maior grau de verdade e maior valor no universo, como se pode ver na equação (13).

$$u* = sup\left\{u \in U \,|\, \mu_U(u) = hgt(u)\right\}, u \in U \tag{13}$$

Em que *hgt (u)* representa a a altura desse elemento do universo de discurso.

2.4.3.3. ALTURA

Neste método são consideradas separadamente cada resposta, mas o raciocínio é semelhante ao caso Média dos máximos, uma vez que vão ser considerados os máximos de cada resposta (Figura 12).

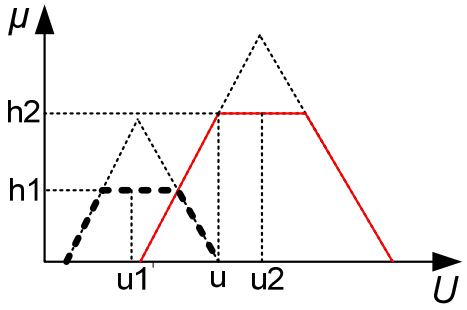


Figura 12 Método de desfuzificação Altura.

Este utilizará os máximos de cada uma das respostas (consideradas separadamente) e fará uma soma pesada dos mesmos para determinar o valor de controlo.

2.4.3.4. MEIO DOS MÁXIMOS

Este método é semelhante ao método Primeiro máximo/ Último máximo mas, em vez de determinar o valor a partir do primeiro ou último máximo, determina o valor intermédio entre o primeiro e último máximo (Figura 13) conforme mostra a equação (14).

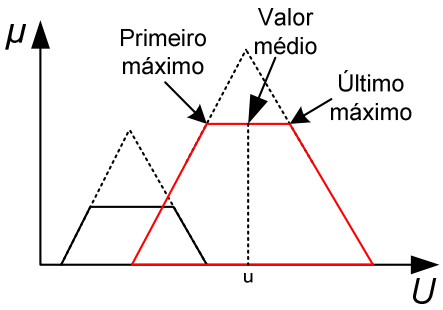


Figura 13 Método de desfuzificação Meio dos máximos.

$$u* = \frac{inf\left\{u \in U \,|\, \mu_U(u) = hgt(u)\right\} + sup\left\{u \in U \,|\, \mu_U(u) = hgt(u)\right\}}{2} \qquad (14)$$

2.5. CONTROLADORES MAMDANI

No controlador Mamdani descreve-se o estado do processo através de variáveis linguísticas que são utilizadas como entradas para as regras de controlo. A variável básica é uma variável de entrada de um sinal medido ou uma variável de saída de outro controlador.

O número de variáveis linguísticas, e a relação entre as mesmas, determina o número de possíveis regras, mas devido à existência de estados impossíveis ou de estados que não acrescentam vantagens no controlo, estes podem ser desprezados. Assim, apenas se torna necessário formular as regras que cubram as condições essenciais.

A arquitectura do controlador é apresentada na Figura 14. A tracejado são indicados os valores difusos e a traço contínuo os valores numéricos reais. Constata-se a existência do bloco de desfuzificação, após a agregação dos resultados de cada regra, de modo a obter-se o valor da variável de controlo do processo.

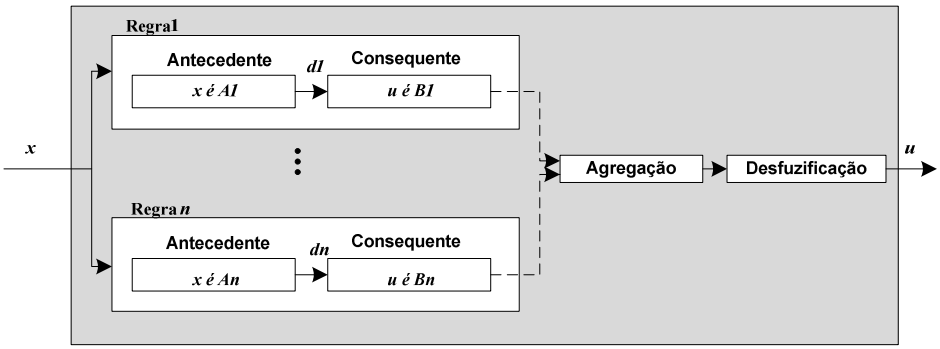


Figura 14 Arquitectura do controlador Mamdani [6].

Na Figura 15 apresenta-se um exemplo do processo de inferência, segundo Mamdani, de modo a melhor se compreender a abordagem efectuada.

Existem duas variáveis linguísticas designadas por "*Erro*" e "*Controlo*", sendo ambas definidas por três conjuntos difusos. A variável "*Erro*" representa a premissa da regra enquanto a variável "*Controlo*" representa a acção definida por cada regra.

Admitindo que a variável básica de entrada "e*rro_observado*" tem um valor de -50, e analisando a Figura 15, verifica-se que existe uma comparação desse valor com cada premissa de cada regra (cada linha representa uma regra) onde se determinam que regras são aplicáveis, ou seja, quais as funções de pertença que cobrem o valor da variável básica "*erro_observado*".

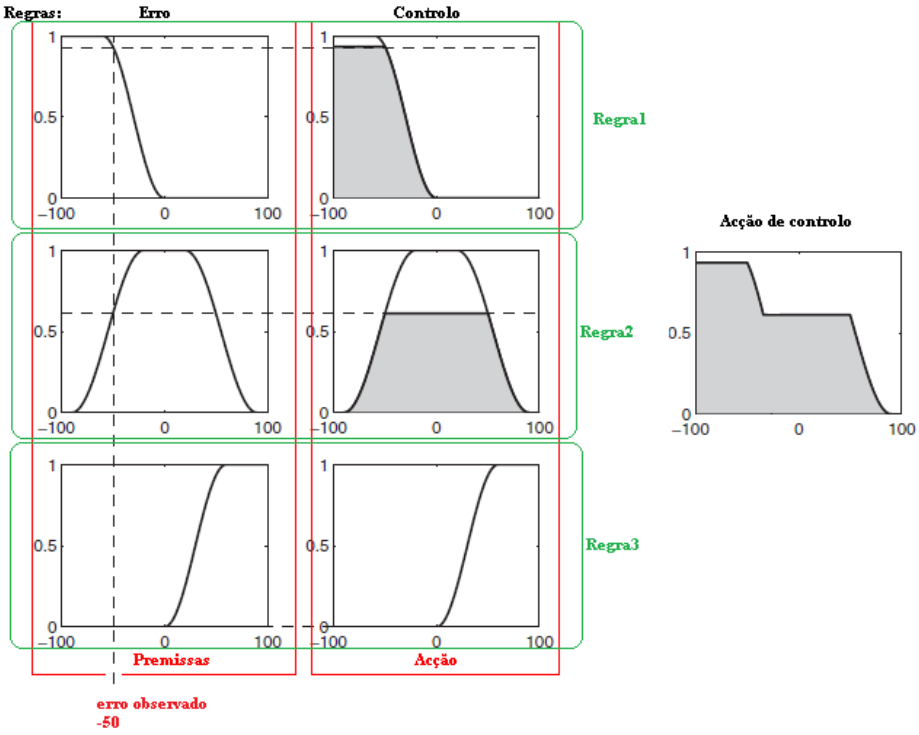


Figura 15 Processo de inferência de Mamdani [2].

Após a comparação e a determinação da aplicabilidade de cada regra, é necessário processar a informação, ou seja, cruzar a informação contida nas funções de pertença das regras consideradas válidas de modo a construir a acção de controlo.

Verifica-se que o valor máximo do grau de pertença observado na variável linguística "*Controlo*" é, no máximo, igual ao valor observado na comparação entre o valor da variável básica de entrada "*erro_observado*" e a função de pertença que define a variável linguística "*Erro*" da respectiva regra.

Como se pode ver na Figura 15, apenas as duas primeiras regras são aplicáveis; logo, apenas as acções dessas regras serão consideradas para a construção do conjunto difuso "*Acção de controlo*". Pode-se constatar que o conjunto difuso "*Acção de controlo*" resulta da reunião das acções resultantes das duas primeiras regras. Após serem obtidos os conjuntos que definem a variável linguística "*Controlo*", pode-se construir o conjunto difuso "*Acção de controlo*", representado também na Figura 15.

Quando o conjunto difuso "*Acção de controlo*" estiver totalmente definido, deve-se utilizar um método para efectuar a desfuzificação, por exemplo, um dos referidos na secção 2.4.3.

A aplicação de um dos métodos vai permitir obter uma variável real passível de ser aplicada ao processo.

2.6. CONTROLADORES TAKAGI-SUGENO

Este controlador usa o mesmo esquema de inferência que o controlador Mamdani, mas a principal diferença reside no método de activação e nas funções de pertença da resposta. Convém referir que os controladores Takagi-Sugeno não necessitam da aplicação dos métodos de desfuzificação [2].

Este controlador resulta de uma modificação do controlador Mamdani, onde as regras formuladas têm antecedentes, ou premissas, difusas, tal como acontece no controlador Mamdani. Os consequentes, ou acções, são definidos como uma função das variáveis linguísticas de entrada, sendo a regra do tipo [6]:

$$Se\ x\ é\ A\ ENTÃO\ u = f(x)$$

A função *f(x)* é um conjunto difuso que representa, em geral, uma combinação linear das entradas:

$$F(x) = b_0 + b_1 x_1 + \ldots + b_n x_n$$

Na Figura 16 apresenta-se o processo de inferência onde se compara o valor das variáveis básicas com as premissas das regras, tal como foi feito para o controlador Mamdani. Em seguida é feita a operação lógica "***E***" com os graus de verdade verificados e, como resposta a cada regra, obtém-se o valor ($\boldsymbol{u_n}$) em função das variáveis de entrada.

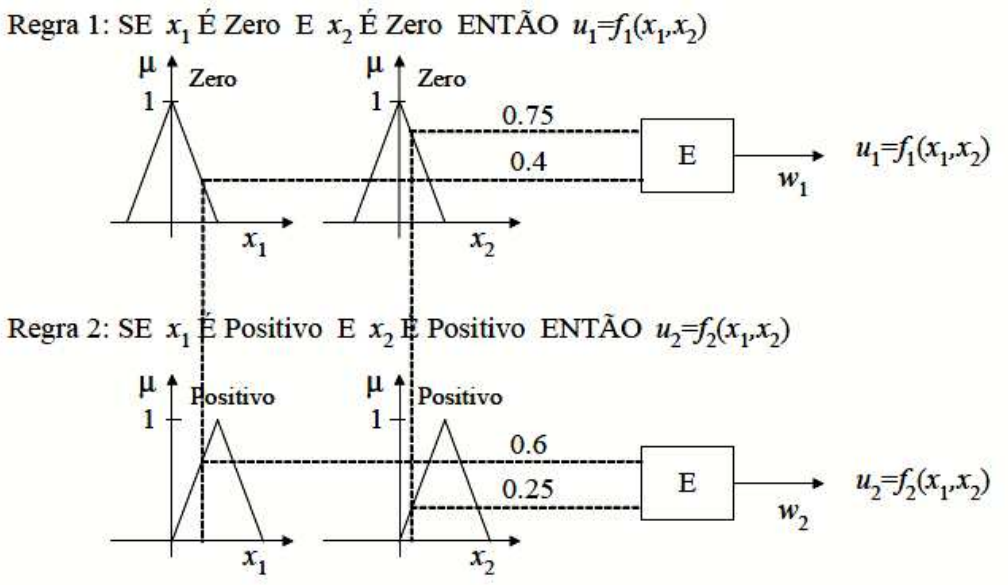


Figura 16 Processo de inferência [6].

O valor de controlo (u) a aplicar ao processo é obtido através da média ponderada das repostas de cada regra, como por exemplo [6]:

$$u = \frac{w_1 \cdot u_1 + w_2 \cdot u_2}{w_1 \cdot w_2} \tag{15}$$

A arquitectura deste controlador pode ser descrita pelo esquema apresentado na Figura 17.

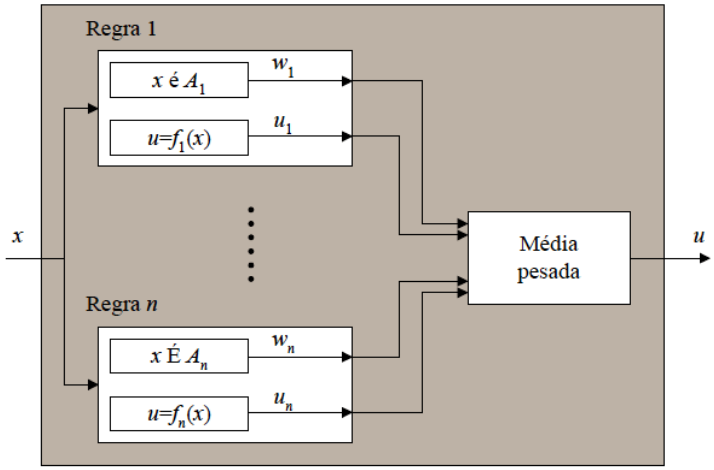


Figura 17 Arquitectura do controlador Takagi-Sugeno [6].

Como à saída de cada regra são fornecidos os valores que permitem calcular a média pesada, e esses valores não são difusos, não existe a necessidade de efectuar o processo de

desfuzificação. O valor obtido no cálculo da média pesada será o valor da variável de controlo que irá permitir controlar o processo.

Após a introdução dos conceitos principais associados à lógica difusa, a apresentação de alguns dos métodos de desfuzificação existentes e a análise dos controladores difusos de Mamdani e Takagi-Sugeno, serão abordados nos capítulos seguintes os requisitos para o projecto do controlo por lógica difusa de um AGV, bem como todas as alternativas e escolhas adoptadas para a implementação do mesmo.

3. LÓGICA DIFUSA NA ROBÓTICA

No Capítulo 2 abordaram-se as questões mais relevantes associados à temática “Lógica Difusa”, desde a evolução histórica, passando pelos conceitos teóricos, até aos controladores difusos.

Neste Capítulo pretende-se enquadrar o conceito “Lógica Difusa” na área da robótica, sendo feito um enquadramento da sua utilização neste domínio. Por último, serão mencionadas algumas aplicações onde a lógica difusa é utilizada na robótica.

A robótica é uma área que sofreu um grande desenvolvimento nos últimos anos principalmente ao nível industrial. Existem hoje inúmeras soluções robotizadas para aumentar a eficiência das indústrias e retirar tarefas desgastantes e stressantes aos operários, assim como reduzir os custos associados à produção dos seus produtos. Um dos exemplos é a utilização de braços robóticos nas linhas de produção; outro exemplo é o uso de robôs autónomos na indústria afectos a tarefas de transporte e armazenamento. Mas não é só na indústria que a robótica se tem vindo a impor. Actualmente existem, por exemplo, sistemas autónomos aéreos capazes de realizar tarefas como vigilância, detecção da área de fogos florestais, entre outras.

Na área da robótica autónoma, realizam-se todos os anos torneios e encontros internacionais para fomentar o desenvolvimento tecnológico e a permuta de conhecimento a nível mundial sobre o tema.

A origem dos robôs tem início nos autómatos; desde então surgiram robôs manipuladores, dispositivos móveis guiados à distância, robôs semi-autónomos e mesmo totalmente autónomos (como é o caso do robô *Opportunity*, desenvolvido pela NASA para missões espaciais). Mas mesmo este tipo de robôs tem ainda limitações na sua autonomia pois necessitam de comandos enviados por humanos que controlam a missão.

Existem diferentes perspectivas em relação ao conceito de controlo, e destas múltiplas visões surgem múltiplas arquitecturas de controlo. A arquitectura de controlo caracteriza a forma como está organizada a criação de acções a partir de percepções.

Associadas ao conceito de controlo surgem várias temáticas, tais como, Inteligência Artificial, Teoria de Controlo, Neurociências, Electrónica e Mecânica. Em cada uma destas temáticas o ponto de vista do conceito de controlo difere. Existem assim vários problemas de controlo, como por exemplo, controlo da velocidade das rodas de um veículo terrestre, controlo da trajectória e controlo da missão [7].

3.1. CONTROLO

A arquitectura de controlo define a estrutura e a organização do robô. Esta difere consoante a abordagem efectuada, podendo a mesma ser deliberativa, reactiva, hierárquica, centralizada, etc. A definição da arquitectura permite organizar a complexidade do problema de controlo.

Dependendo da abordagem, as arquitecturas podem ser Deliberativas, as quais se baseiam em planeamento e têm origem nos paradigmas de Inteligência Artificial convencional. Podem também ser Reactivas, baseando-se na relação directa entre sensorização e actuação (comportamentos), ou podem ainda ser Mistas, situação em que, incorporando características de ambas as arquitecturas anteriores, combinam em geral comportamentos reactivos de baixo nível com planeadores de alto nível [7].

3.1.1. ESQUEMA GENÉRICO DE CONTROLO

O esquema de controlo genérico, apresentado na Figura 18, reflecte a metodologia de controlo. Numa primeira fase dá-se a percepção do ambiente que rodeia o robô reunindo informações sobre o mesmo. Em seguida, através de conhecimento e/ou base de dados, é feita a localização e construção do mapa global. Conforme a missão, será planeada uma trajectória para o robô executar. O resultado é fornecido a outra camada que será responsável pelo controlo de movimento, onde é feita a execução da trajectória planeada.

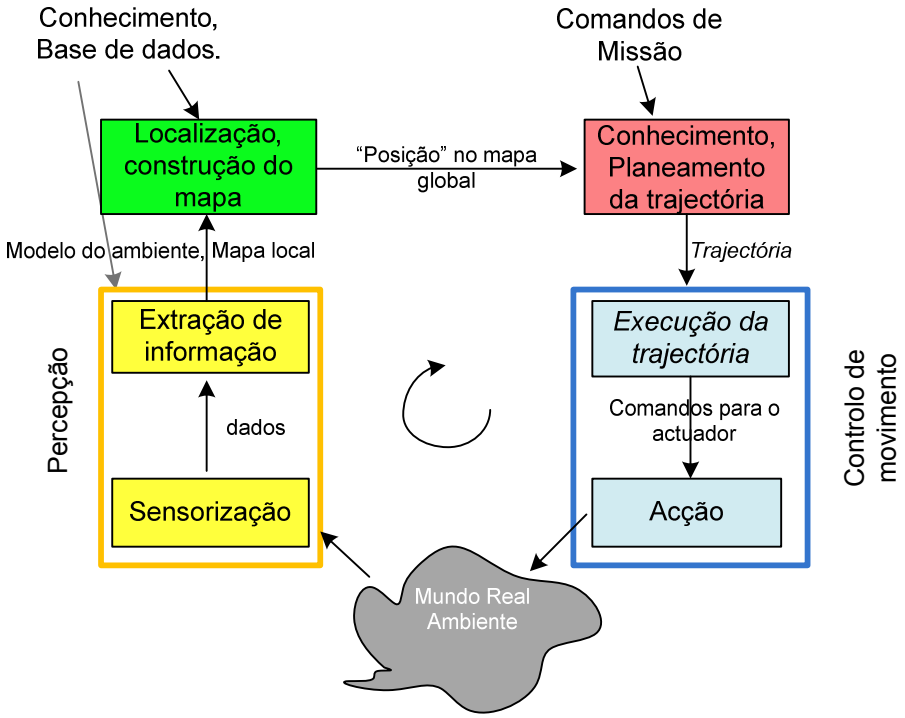


Figura 18 Esquema de controlo genérico [7].

Qualquer abordagem para controlar um sistema dinâmico necessita de conhecimento do sistema ou de um modelo do sistema a ser controlado. Se se considerar que os robôs são inseridos num ambiente real, tudo se torna mais complexo pois estes são caracterizados por possuírem incertezas impossíveis de quantificar [9].

3.2. ABORDAGENS AO CONTROLO DE ROBÔS

Uma das estratégias mais comuns para lidar com este problema, é a de equipar o robô com a capacidade de ser ele próprio a construir o modelo. Esta estratégia leva-nos às arquitecturas hierárquicas onde, como se pode ver na Figura 19, existem duas camadas: uma de baixo nível, responsável pela execução do plano, e outra de alto nível, responsável pela construção do modelo e geração do plano.

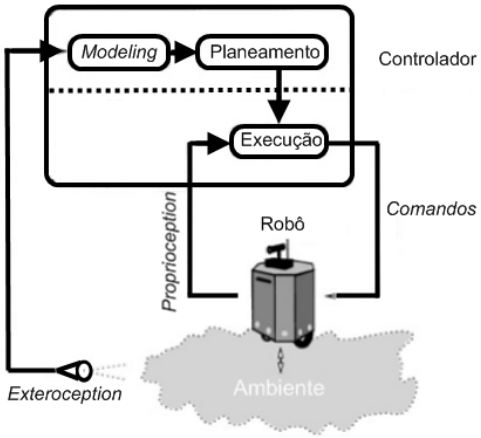


Figura 19 Arquitectura hierárquica [9].

O robô utiliza sensores para percepcionar o ambiente (*exteroception*), como sonares ou câmaras, e usa sensores, como codificadores nas rodas, para monitorizar as suas operações (*proprioception*). Através dos sensores que percepcionam o ambiente, o robô adquire o modelo do espaço de trabalho como ele é. A partir desse modelo, um programa de planeamento cria um plano da tarefa a executar num dado ambiente[9]. Em seguida, este plano é fornecido a um programa de controlo de nível inferior onde será feita a sua execução. Tipicamente, a execução prossegue de uma maneira cega, onde o controlador usa um modelo do robô e monitoriza o seu movimento, por exemplo, a partir dos dados obtidos pelos *encoders* (*proprioception*), mas não tenta percepcionar novamente o ambiente ou o modelo durante a execução. Esta é a limitação desta abordagem, pois nos ambientes reais a informação adquirida pelo robô é incompleta e inexacta, devido às incertezas na sua percepção. Desta forma, o modelo torna-se rapidamente desactualizado e inadequado para o ambiente realmente encontrado durante a execução [9]. A modelação e os processos de planeamento são normalmente computacionalmente complexos, e gastam bastante tempo de processamento, o que agrava este problema. Poder-se-ia adoptar como

solução a utilização de um *feedback* do ambiente por todas as camadas. Esta abordagem é conhecida como "*Sense-Model-Plan-Act*", ou SMPA [9]. No entanto, o aumento de complexidade leva a tempos de resposta do sistema robótico na ordem dos segundos, que seriam elevados para ambientes dinâmicos.

A ideia geral considera que o planeamento deve fazer o menor número possível de suposições sobre o ambiente actual, encontrado durante a execução, e que a execução dever ser sensível ao ambiente e adaptar-se às ocorrências encontradas. Para concretizar isso, os dados provenientes da percepção do ambiente têm de ser incluídos na camada de execução, como mostra a Figura 20. Esta nova derivação tem duas consequências: em primeiro lugar afecta a velocidade da camada de execução, que geralmente é alta, o que torna o tempo disponível para interacção do robô com o ambiente muito apertado; em segundo lugar, acarreta um aumento da complexidade da camada de execução, que precisa de considerar múltiplos objectivos, tais como, cumprir os objectivos tácticos provenientes do planeador, e reagir aos acontecimentos ambientais detectados pela percepção [9].

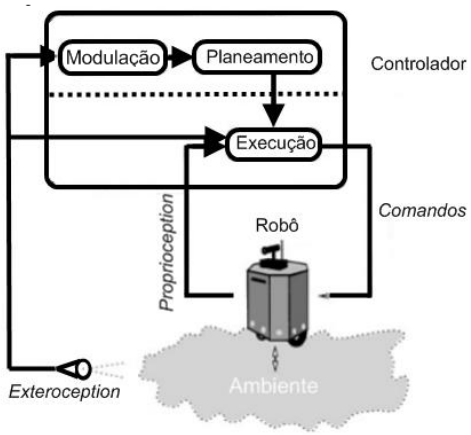


Figura 20 Arquitectura híbrida [9].

A maioria dos investigadores optou pela estratégia de "dividir para reinar", sendo a camada de execução decomposta em pequenos processos independentes de tomada de decisão, ou comportamentos (*behaviors*). Desta forma, surgiu o conceito de unidade comportamental que será abordada na secção seguinte.

3.3. UNIDADES COMPORTAMENTAIS

A organização da camada de execução, apresentada na Figura 21, é baseada em comportamentos, em que cada comportamento (*behavior*) implementa uma política de controlo para cada sub-tarefa específica como, por exemplo, seguir um caminho ou evitar um obstáculo.

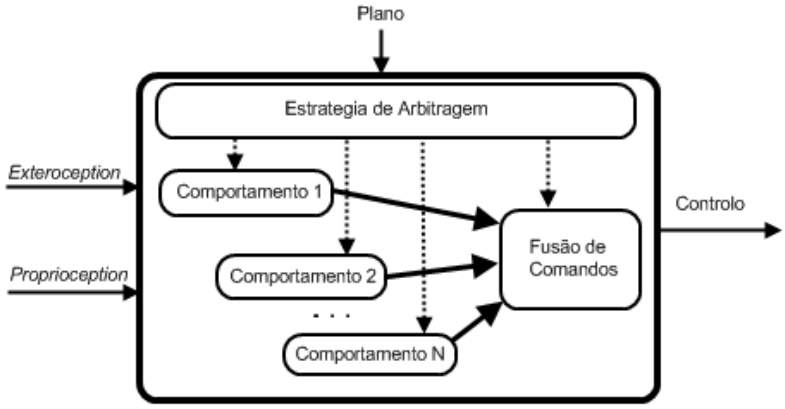


Figura 21 Organização do módulo de execução baseado em comportamentos.

A estratégia de arbitragem decide que comportamento deverá ser activado dependendo do objectivo actual e das ocorrências ambientais. Uma vez que vários comportamentos podem estar activos, é necessária uma fusão dos comandos, de forma a combinar os resultados desses comportamentos num comando de acção. As arquitecturas híbridas não resolvem o problema de navegação autónoma, mas proporcionam um ponto de partida para tratar e integrar os diferentes sub-problemas [9].

A principal e mais comum aplicação da lógica difusa é o uso de controladores difusos para implementar unidades individuais de comportamentos (*individual behavior units*), onde é incorporado conhecimento heurístico de controlo em regras, sendo a escolha conveniente quando não é possível obter um modelo preciso do sistema a ser controlado. A lógica difusa oferece um bom grau de robustez, face à grande variabilidade e incerteza dos parâmetros. Estas características tornam o controlo difuso adequado para as necessidades de navegação em robôs autónomos, devido à quantidade de informação imprecisa com que estes sistemas têm de lidar.

Existem três abordagens ao projecto de comportamentos designadas por *proprioceptive behaviors*, *sensor-based behaviors* e *complex behaviors.*

Os *proprioceptive bahaviors* surgem da abordagem onde o robô utiliza sensores, como *encoders* ou outros sensores internos, para inferir a sua posição. E uma vez que a posição é conhecida ou passível de ser conhecida, é dado um caminho na forma de sequência de coordenadas, que levará à geração de comandos para os módulos de controlo dos motores, para que estes executem esses comandos da forma mais precisa possível.

Os *sensor-based behaviors* são comportamentos baseados nos dados adquiridos pelos sensores; esta abordagem consiste na aquisição de informações sobre o ambiente, onde o robô "sente o exterior" e move-se em relação às características do ambiente, ao invés de se movimentar em relação a um caminho representado internamente.

Alguns exemplos típicos da aplicação deste método são, a movimentação ao longo de paredes, alcançar uma fonte de luz e evitar obstáculos. O primeiro uso registado da lógica difusa pertence a este método, tendo Sugeno e Nishida desenvolvido um controlador difuso capaz de movimentar um modelo de um carro por um caminho delimitado por duas paredes [9].

Os comportamentos anteriores podem ser considerados como básicos, pois preocupam-se apenas com um único objectivo, ou critério de desempenho. A lógica difusa tem sido usada para implementar comportamentos complexos, que levam em conta múltiplos objectivos como, por exemplo, seguir um dado caminho enquanto evita obstáculos.

Por último, um comportamento complexo tipo (*complex behaviors*) pode ser conseguido com uma implementação completa do módulo de execução, como o da Figura 20. Se se pretendem considerar dois objectivos distintos, pode-se escrever regras cujos antecedentes consideram os dois objectivos, ou podem-se escrever dois conjuntos de regras, uma para cada objectivo, e combinar de alguma forma as suas saídas.

O controlo difuso é considerado como sendo uma metodologia adequada para projectar controladores robustos, capazes de apresentar um desempenho satisfatório face aos ruídos e flutuações das entradas.

As três principais vantagens são: em primeiro lugar, devido ao formato das regras difusas, estas facilitam a escrita de comportamentos simples e eficazes para uma variedade de tarefas sem se ter de recorrer a modelos matemáticos complexos; em segundo lugar, devido à utilização de variáveis linguísticas que possuem conceitos associados, as regras podem

ser facilmente adaptadas a outras plataformas; por último, como o controlo difuso é menos sensível às variações das entradas, porque lida com conceitos, como resultado obtém-se um movimento suave do robô e uma pequena degradação da saída face aos erros e às flutuações nos dados dos sensores.

3.3.1. *BEHAVIOR COORDINATION MECHANISM*

O principal problema provém da questão de como se coordenam as actividades simultâneas das várias unidades produtoras de comportamentos (*behavior-producing units*) para obter um comportamento global que atinja os objectivos pretendidos.

O problema da coordenação de comportamentos é reconhecido como uma das maiores questões das abordagens baseadas em comportamentos na robótica. A coordenação de comportamentos divide-se em dois problemas diferentes: como decidir que comportamento deve ser activado em cada momento, e como combinar o resultado dos diferentes comportamentos num único comando final para os actuadores.

Muitos mecanismos foram propostos nas últimas décadas para resolver estes problemas, sendo uma classe particular, baseada na lógica difusa e no controlo difuso, ainda alvo de investigação. Os mecanismos de coordenação de comportamentos dividem-se em duas classes principais, são elas, a arbitragem (*behavior arbitration*) e a fusão de comandos (*command fusion*), de onde surgem diferentes metodologias como é visível na Figura 22.

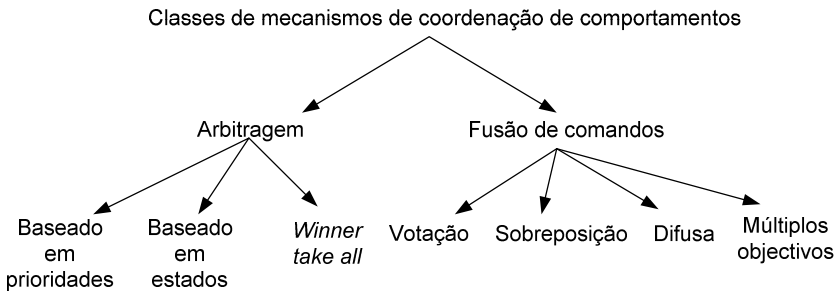


Figura 22 Classes de mecanismos de coordenação de comportamentos [12].

Numa primeira fase será abordada a política de arbitragem, que determina que comportamento deve influenciar a operação do robô em cada momento, e assim, em última instância, determinar a tarefa realmente executada pelo robô.
Como foi apresentado na Figura 22, a arbitragem divide-se em três metodologias, são elas:

- Através de prioridades, onde os comportamentos com maior prioridade podem suprimir as saídas dos comportamentos de menor prioridade;

- Através de estados, que se pode dividir em quatro estratégias diferentes, nomeadamente:

    1. Sistema de eventos discretos, onde a selecção do comportamento é feita através da transição de estados, e a detecção de certos eventos muda o sistema para um novo estado e para um novo comportamento;

    2. Sequência temporal, em que em cada estado um comportamento é activado e disparos perceptuais provocam a transição de estado;

    3. *Bayesian Decision Analysis*, onde é escolhida a acção que maximiza a utilidade esperada do agente segundo a relação custo/benefício;

    4. Aprendizagem com reforço (*Reinforcement Learning*), que pode ser de dois tipos, *Hierarchical Q-learning* e *W-learning* [12]*;*

- "*Winner-take-all*", que é um mecanismo de selecção de acções que resulta da iteração de um conjunto de comportamentos distribuídos que competem até que um ganha e assume o controlo do sistema [12].

A *subsumption architecture*, proposta por Brooks, é uma arquitectura reactiva, composta por camadas de comportamentos desejados, ou níveis de competência, com um método de conexão paralelo e distribuído, visível na Figura 23 [7]. Esta possui um mecanismo de arbitragem baseado em prioridades, onde as prioridades são pré-definidas de forma a resolverem conflitos. Surge através da ideia de que se deve começar com inteligência simples e, em seguida, expandir a inteligência para níveis superiores, ou seja, camadas superiores, dando por isso vantagem à simplicidade e onde o comportamento complexo "emerge" a partir de comportamentos base. Deste modo, se os níveis superiores de inteligência falharem os mais baixos continuam a funcionar [10]. Os comportamentos de

nível superior podem alterar o fluxo de dados nos níveis inferiores por supressão ou inibição.

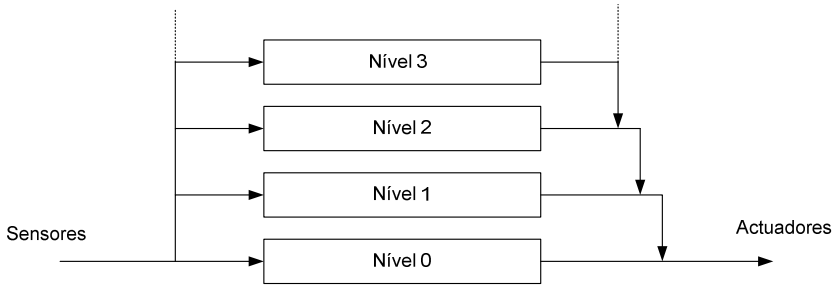


Figura 23 Estrutura de camadas de inteligência da *subsumption architecture* [10].

Para efectuar a supressão ou a inibição existem nós supressores ou inibidores, apresentados na Figura 24. O nó de supressão permite a substituição do fluxo normal de dados de um agente pelos dados de outro agente, sendo o fluxo normal de dados suprimido durante um período de tempo específico [11]. O nó inibidor permite a outro agente inibir a saída de um agente, também por um período de tempo específico.

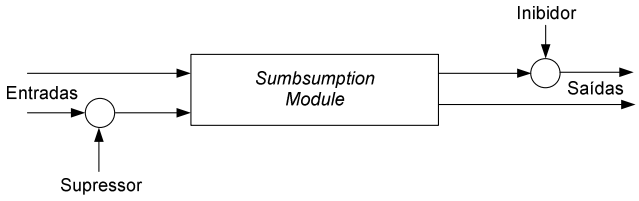


Figura 24 Agente com nó supressor e nó inibidor [11].

A maioria das arquitecturas assume uma arbitragem dinâmica, onde a decisão de qual comportamento se deve activar depende do plano actual e das contingências do ambiente. O plano é geralmente gerado por módulos de raciocínio de nível superior. Muitas destas arquitecturas não permitem a execução simultânea de comportamentos e, deste modo, não necessitam do bloco para a fusão de comandos.

Tanto a arbitragem dinâmica como a arbitragem fixa podem ser implementadas utilizando mecanismos da lógica difusa [9]. As duas principais vantagens em fazê-lo são: capacidade de expressar activações parciais e simultâneas de comportamentos, e transições suaves entre comportamentos.

Em relação à fusão de comandos, a maneira mais simples de efectuar a selecção de comandos de diferentes comportamentos é usar um esquema de *switching*, em que a saída

de um comportamento é seleccionada para execução e todas as restantes são ignoradas. A saída seleccionada dependerá da estratégia de arbitragem. Este esquema é muito utilizado mas é desadequado para situações onde vários critérios devem ser levados em conta simultaneamente. Por exemplo, consideremos um robô que encontra um obstáculo enquanto segue um caminho, e suponhamos que este tem a opção de contornar o obstáculo pela direita ou pela esquerda; esta escolha pode ser indiferente ao comportamento de "evitar obstáculo". Contudo, do ponto de vista do comportamento de "seguir caminho", uma das escolhas poderá ser dramaticamente melhor que a outra [9]. Por este facto surgiram diferentes abordagens para lidar com este problema.

Na Figura 25, são visíveis quatro metodologias distintas para a fusão de comandos; são elas a Votação, Sobreposição (*Superposition*), Difusa e de Múltiplos objectivos. O mecanismo de fusão de comandos combina as recomendações de múltiplos comportamentos para formar uma acção de controlo que representa o consenso. Assim, esta abordagem fornece um esquema de coordenação que permite que todos os comportamentos contribuam simultaneamente para o controlo do sistema de modo cooperativo, em vez de competitivo.

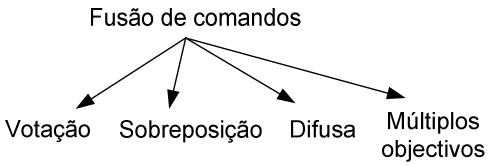


Figura 25 Metodologias para fusão de comandos [12].

Em relação às técnicas de votação, estas interpretam a saída de cada comportamento como um voto a favor ou contra as possíveis acções e a acção com o somatório máximo ponderado de votos é seleccionada [12]. Existem três exemplos principais destas técnicas, nomeadamente:

- *Distributed Architecture for Mobile Navigation* (DAMN);
- *Sensors, Actuators, Markers, Behaviors, and Arbiters* (SAMBA);
- *Action Voting.*

Na arquitectura DAMN, visível na Figura 26, não é imposta uma hierarquia de controlo e todos os comportamentos votam uma determinada acção, sendo esta semelhante à

arquitectura reactiva (como a *subsumption*, proposta por Brooks), mas permite a existência de módulos deliberativos com planeadores e modelos do mundo [7]. O votador selecciona a melhor acção.

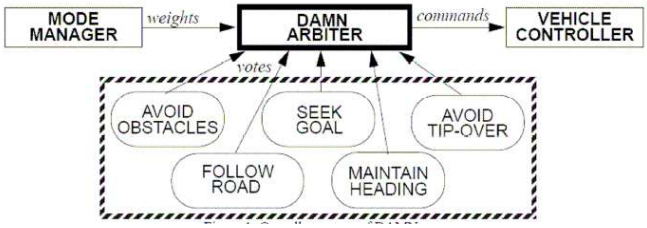


Figura 26 Estrutura global da DAMN [7].

Na Figura 27 é apresentado um exemplo da fusão de comandos nesta arquitectura.

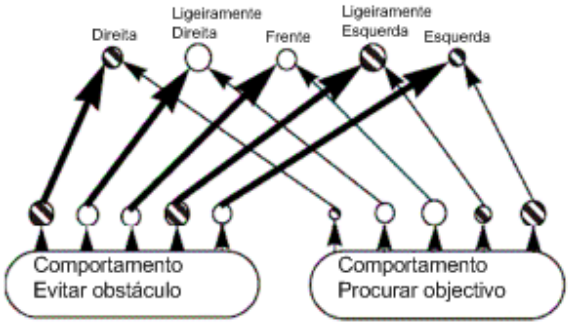


Figura 27 Fusão de comando na arquitectura DAMN [7].

Outro exemplo desta técnica é a arquitectura SAMBA (*Sensors, Actuators, Markers, Behaviors, and Arbiters*) onde os comportamentos primitivos produzem reacções em forma de mapas de acção primitivos, e as saídas dos comportamentos são geradas a partir de quatro mapas de acção primitivos. A arbitragem de comandos combina os mapas multiplicando cada um pelos ganhos e somando os resultados [13], como é visível na Figura 28.

O último exemplo é o *Action Voting*, em que cada comportamento vota por uma acção e contra acções indesejadas. Os votos são somados e a acção com maior valor é seleccionada [13].

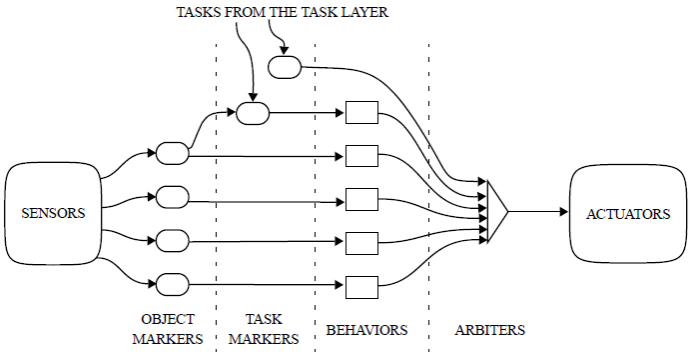


Figura 28 Arquitectura SAMBA.

No caso das técnicas de sobreposição, estas combinam as recomendações dos comportamentos usando uma combinação linear. Existem dois exemplos principais, nomeadamente:

- *Potential Fields;*
- *Motor Schemas.*

O primeiro, designado por *Potential Fields*, faz uma abordagem ao planeamento do movimento e o robô move-se sob a influência de um campo potencial artificial produzido por uma força de atracção na configuração do objectivo e forças repulsivas nos obstáculos [13].

O segundo exemplo, o *Motor Schemas*, gera um vector que codifica a direcção e a intensidade da acção do motor, calculando o campo potencial para a configuração actual do robô. Estes vectores são adicionados para gerar uma acção motora combinada e, por fim este é multiplicado por um ganho [13]. Esta abordagem é usada na *Autonomous Robot Architecture* (AuRA) [13].

Uma outra técnica é o mecanismo de fusão difuso, que tem vários pontos de contacto com as técnicas de votação, mas onde as noções da lógica difusa e da inferência difusa são usadas para formalizar o processo de selecção da acção.

No controlo difuso baseado em comportamentos, cada comportamento é sintetizado por uma base de regras e por um mecanismo de inferência para produzir um conjunto de valores de saídas, tal como o exemplo com o comportamento “Evitar obstáculo”, apresentado na Figura 29, que possui apenas duas regras.

Usando a inferência difusa clássica, as regras são combinadas num conjunto de valores das saídas, que codificam o grau de adequação de cada acção do ponto de vista do comportamento. Os comportamentos que competem pelo controlo têm de ser coordenados de modo a evitar conflitos. A coordenação difusa de comportamentos é desempenhada pela combinação das saídas difusas dos comportamentos usando um operador apropriado. Quando a saída de um comportamento é um conjunto difuso, pode-se ver o problema da fusão de comandos como uma parte do problema de combinação das preferências individuais. Pode-se ver cada unidade produtora de comportamento como um agente que expressa preferências sobre qual comando a aplicar, e os graus de preferência são representados por um conjunto difuso sobre o espaço de comando [9].

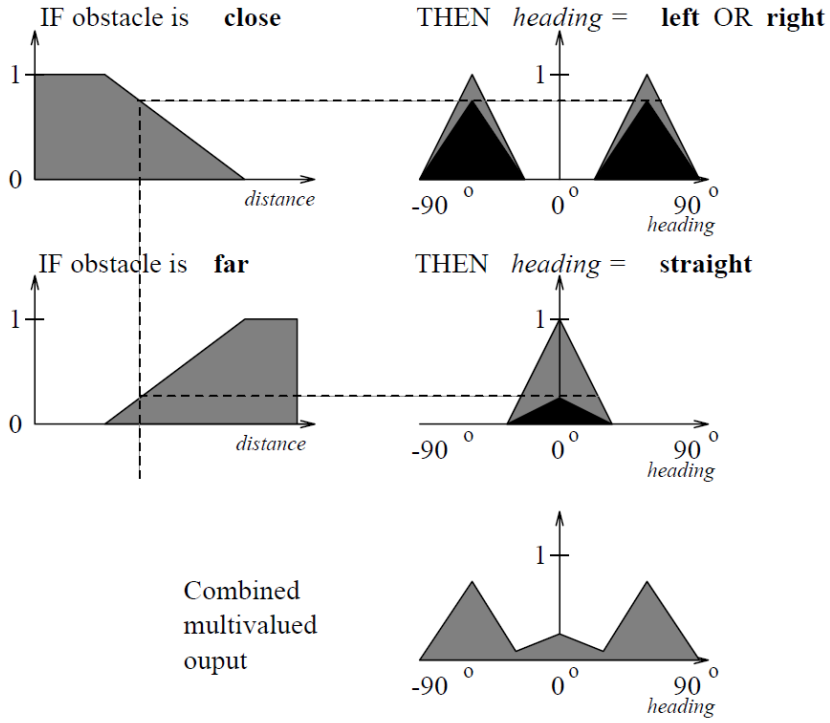


Figura 29 Exemplo da base de regras difusa para descrever o comportamento difuso "Evitar obstáculo" [9].

Podem-se usar os operadores difusos para combinar as preferências dos diferentes comportamentos numa preferência colectiva e, finalmente escolher um comando desta preferência colectiva, originando uma definição formal a partir da interpretação da lógica difusa como uma lógica de preferências. A fusão de comandos pode ser dividida em dois passos: combinação de preferências e decisão. A lógica difusa oferece diferentes operadores para desempenhar a combinação e muitos métodos de desfuzificação para desempenhar a decisão. É importante ter presente que a decisão de uma preferência global

pode ser diferente do resultado da combinação das decisões de todas as preferências individuais, como é visível no exemplo da Figura 30 [9]. Intuitivamente, cada decisão individual proveniente de um comportamento, indica qual é o comando preferido para o comportamento em questão mas não diz nada sobre a conveniência das alternativas. A desfuzificação é o processo para determinar um valor numérico final usado para o controlo, a partir dos conjuntos difusos resultantes, como se mostra no exemplo da Figura 30.

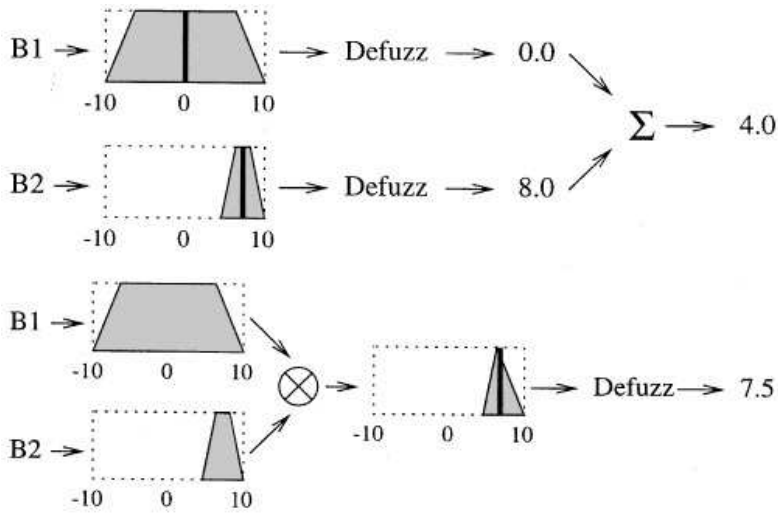


Figura 30 Duas abordagens para a fusão de comandos: combinação de decisões individuais (em cima) e combinação de preferências individuais (em baixo) [9].

As preferências contêm mais informações, pois dão uma medida da conveniência para o comando possível, e ao combinar as preferências usa-se mais informação do que combinar vectores e pode-se produzir uma decisão final diferente.

Foram propostas várias sugestões para melhorar a fusão de comandos, mas a mais popular utiliza a combinação arbitrária de conjuntos difusos, seguida por uma etapa de desfuzificação [9]. Têm sido apontados vários problemas provenientes de uma aplicação "cega" da desfuzificação a conjuntos difusos combinados, principalmente quando estes não resultam numa condição única, nomeadamente quando resulta desta selecção um valor de controlo indesejável, ou seja, um valor que se situa entre dois picos do conjunto difuso combinado e possui um grau de pertença baixo nesse conjunto. No caso do controlo de um robô, que considera a tarefa de evitar obstáculos pela direita ou pela esquerda, esta situação pode levar a que este decida ir em frente embatendo no obstáculo. Várias abordagens a este problema apontam para a sua resolução através da definição de diferentes esquemas de desfuzificação, ou com a utilização do método de desfuzificação "centro de gravidade",

apenas indicando que o projectista deve certificar-se que a saída do comportamento deve ser única. Uma outra abordagem exige que o projectista declare explicitamente as relações entre todos os comportamentos potencialmente conflituosos. Em ambas, fica claro que as incoerências e as ambiguidades no conjunto de regras devem ser prevenidas, através de um projecto cuidadoso, em vez de se efectuarem correcções na arbitrariedade das manipulações matemáticas. Interpretando a geração de várias saídas como um sinal de inconsistência na arbitragem das regras, permite um tratamento mais analítico dos conflitos entre os comportamentos.

Embora a lógica difusa seja uma ferramenta útil para escrever estratégias de coordenação, esta ainda não soluciona o problema geral da coordenação de comportamentos. Por exemplo, continua-se sem se saber como se deve descriminar uma situação em que os diferentes comandos, propostos por diferentes comportamentos, devem ser geridos ou mediados, pois surgem a partir de um caso que deve ser considerado como um conflito. Estes problemas resultam de uma qualquer forma de associação local, e podem ser vistos como exemplos do problema geral de computação local (ou acção) com os resultados globais (ou objectivos). Estes problemas só podem ser resolvidos por uma integração cautelosa do raciocínio local e global.

Este esquema deve levar à selecção de uma acção, que de algum modo, representa o consenso entre os comportamentos e, assim, compreender a acção que melhor satisfaz os objectivos da decisão que estes codificam. Contudo, existem situações que levam a conflitos entre comportamentos activos, gerando resultados desadequados. Para lidar com estes conflitos surgiu um método designado por "c*ontex-dependent blending*", que será abordado na secção seguinte.

3.3.2. MISTURA DE CONTEXTOS DEPENDENTES (*CONTEXT-DEPENDENT BLENDING*)

A mistura de contextos dependentes (CDB) é um mecanismo de coordenação para esquemas de controlo. Este é um mecanismo geral, que permite expressar diferentes padrões da combinação de comportamentos. A forma mais usual de combinação de comportamentos que se pode realizar através da lógica difusa é obtida usando as regras para expressar uma política de arbitragem, e combinações difusas para desempenhar a fusão de comandos. Podem-se usar condições perceptuais para decidir entre dois

comportamentos alternativos como, por exemplo, as seguintes regras para um robô se deslocar para um destino enquanto evita reactivamente os obstáculos no caminho [9].

IF Obstáculo-perto THEN Evitar-obstáculo

IF NOT Obstáculo-perto THEN Ir-para-destino

Podem-se também sequenciar os dois comportamentos B_1 e B_2, que visam dois objectivos G_1 e G_2, pela utilização das regras de contexto da seguinte forma [9]:

IF G_1 não alcançado THEN B_1

IF G_2 alcançado THEN B_2.

Esta é uma maneira de definir as prioridades dos conjuntos de regras difusas. Note-se que esta mistura é direccionada a objectivos, enquanto a anterior era direccionada a eventos [9].

Estas duas metodologias podem ser combinadas num conjunto complexo arbitrário de regras de contexto, que representam um plano completo para a acção, dizendo que comportamentos devem ser usados em cada situação. O método CDB pode ser implementado num controlador difuso hierárquico, como se mostra na Figura 31 [9].

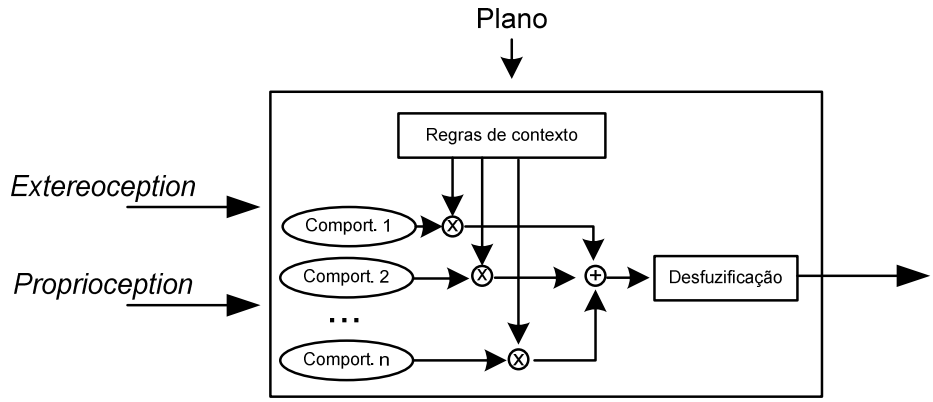


Figura 31 Implementação da mistura de contextos dependentes (CDB) de comportamentos implementada num controlador difuso hierárquico [9].

As condições contextuais podem ser implementadas por um conjunto de *meta-rules* que activam um esquema de controlo, conforme se mostra a seguir.

IF context = A THEN activate_control_scheme C

Onde A, representa o contexto actual. Esta regra activa o esquema de controlo C para um nível determinado pelo grau de verdade do antecedente da regra. O nível de activação pondera, ou pesa, a adequação do esquema de controlo.

Sobre esta metodologia devem ser efectuadas algumas observações. Por exemplo, a desfuzificação deve ser efectuada depois da combinação e, embora todas as regras de contexto estejam agrupadas no mesmo módulo, o mesmo efeito pode ser obtido incluindo cada regra de contexto dentro do comportamento correspondente. Pode-se também usar a estrutura da Figura 31, para implementar comportamentos individuais e combinar vários desses comportamentos (comportamentos complexos) usando uma segunda camada de regras de contexto, e assim por diante. A desfuzificação deverá sempre ser o último passo a realizar [9]. No entanto deve-se ter em atenção a abordagem a tomar para a fusão dos comandos, pois a escolha pode gerar resultados desadequados, ou seja, as regras de contexto podem activar os dois comportamentos no mesmo instante gerando resultados opostos. Deve-se assim garantir que não existem conflitos nas acções, de modo a gerar bons resultados. Uma abordagem possível consiste em fazer o *switching* de comportamentos onde apenas um se encontra activo num determinado instante. Outra abordagem possível, considera todos os comportamentos e selecciona apenas a acção do comportamento que possui maior grau de verdade, por último, pode-se considerar a intersecção entre as acções resultantes de todos os comportamentos pois a decisão tomada provém de uma decisão comum. O último passo a efectuar é a desfuzificação, mas convém referir que cada caso é singular e deve ser analisado convenientemente de modo a garantir que não sejam gerados resultados inapropriados.

As regras de contexto difusas proporcionam um meio flexível para codificar estratégias de arbitragem de comportamentos. Tal como as regras de controlo difuso, as regras de contexto permitem escrever estratégias complexas de modo modular, usando um formato lógico [13]. Uma vez que as regras de controlo e as regras de arbitragem possuem o mesmo formato, isto facilita o desenvolvimento de regras mais complexas de uma forma hierárquica. A fusão de comandos difusa pode ser usada para combinar as recomendações de comportamentos simultâneos, de forma a ser cumprido o objectivo geral.

Fruto do aparecimento da lógica difusa e do desenvolvimento de novas metodologias baseadas na mesma, surgiram arquitecturas que utilizam os seus conceitos. Em seguida, apresentam-se dois exemplos de arquitecturas difusas, são elas *Fuzzy/Multivalued Logic*

Approach e *Fuzzy DAMN*, onde se pretende apenas ter uma ideia das principais características de cada uma delas.

3.3.3. *FUZZY/MULTIVALUED LOGIC APPROACH*

Esta arquitectura tem grande influência dos conceitos da lógica difusa e muitos dos seus componentes são baseados no formalismo difuso ou *multivalued logic*. Esta arquitectura é organizada em três camadas, que consistem em esquemas de controlo, esquemas de comportamentos e esquemas de planeamento, que interagem com o agente assim como com o seu ambiente, em diferentes níveis de abstracção: sinais/estímulos, *intermediate* (símbolo-para-sinal) e símbolo, respectivamente [13].

Os esquemas de controlo descrevem tipos de movimentos baseados nos estados internos do agente e das entradas sensoriais. Os planeadores sintetizam planos que são baseados em descrições simbólicas. Os esquemas de comportamentos fazem a ligação entre os planeadores e os esquemas de controlo, contendo partes simbólicas e níveis de sinal de abstracção [13].

Os esquemas de controlo são conceptualizados como um mapeamento de um conjunto de estados em preferências de um conjunto de acções, a desempenhar. Na prática, cada esquema de controlo é implementado por um conjunto de regras difusas, usando um raciocínio difuso clássico, em que as saídas das regras são combinadas em funções de pertença que reflectem a preferência, ou a adequação, do esquema de controlo implementado. O conjunto difuso resultante é desfuzificado para um único valor de controlo. Os esquemas de controlo podem ser compostos usando operadores da lógica difusa. Como já foi mencionado, a utilização de operadores pode, em certas condições satisfazer os requisitos, mas em casos onde existam conflitos entre comportamentos pode gerar resultados indesejados [13]. Para resolver este problema surge o método CDB, abordado na secção anterior. Os esquemas de controlo utilizam variáveis internas, como a direcção do movimento ou a velocidade.

Os esquemas de comportamento são estruturas que ligam certos padrões de acção, implementados pelos esquemas de controlo, com certos estímulos num ambiente. Desta forma, os comportamentos respondem aos estímulos e podem ser formulados para atingir objectivos como “Evitar obstáculos” [13].

Um comportamento é descrito por três parâmetros:

$$B = \{C, D, O\}$$

onde *D* é o esquema de controlo para um tipo de movimento específico, *O* é o conjunto de descritores do objecto, em relação a qual movimento deve ser desempenhado e *C* é a condição contextual, que descreve o contexto da aplicabilidade do movimento [13].

Os descritores de objectos são modelos de objectos do mundo real, que provocam um comportamento para agir em relação ao mundo exterior, em oposição a um esquema de controlo que actua sobre as variáveis internas. A função dos descritores de objectos é similar aos marcadores perceptuais, onde estes são adaptados cada vez que o sistema perceptual os detecta no mundo real, formando uma malha fechada (Figura 32) [13].

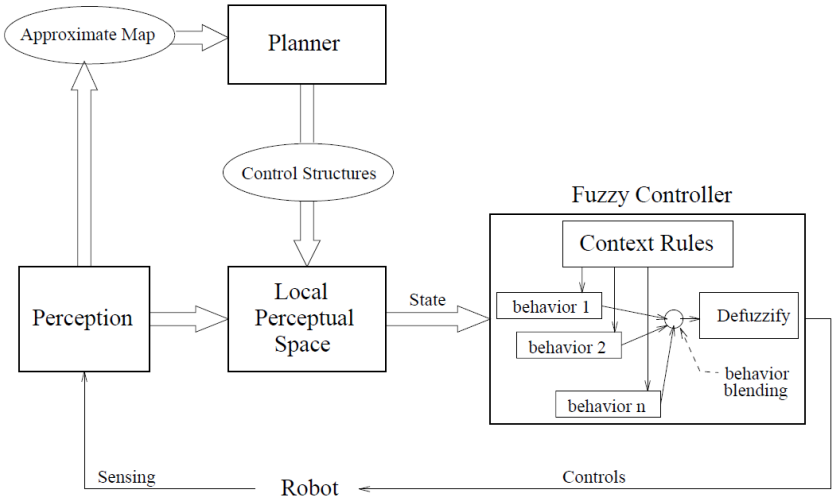


Figura 32 Arquitectura do *FLAKEY* [13].

Assim que os conjuntos de comportamentos primitivos estejam formulados, podem ser compostos em comportamentos complexos, usando a técnica de composição dos esquemas de controlo [13]. Os comportamentos têm especificações em termos das condições prévias, ou contexto de aplicabilidade, e condições posteriores, ou completar objectivos, que podem ser usados pelos planeadores tradicionais para construir planos para realizar tarefas.

A maior desvantagem das abordagens difusas é que não existem linhas claras para definir precisamente as funções de pertença. Contudo, e como já foi referido, existem técnicas de aprendizagem que estão em desenvolvimento e poderão ser uma boa ajuda [13].

Esta estrutura foi desenvolvida com sucesso para o controlo do sistema robótico móvel, FLAKEY (Figura 33).

Figura 33 FLAKEY.

3.3.4. *FUZZY* DAMN

Baseados no trabalho de Payton e Rosenblatt, Yen e Pfluger aperceberam-se de certas questões fundamentais na arquitectura DAMN [13]. Primeiro, as saídas dos comportamentos podem ser expressas como funções de pertença discretas sobre o conjunto de possíveis acções. Segundo, a fusão de comandos baseada no somatório dos pesos da arquitectura DAMN pode ser substituída pelos métodos clássicos da inferência difusa (*Max-min* e *Weighted-sum*) na sua formulação difusa [13]. Terceiro, a selecção da acção pelo máximo da votação pode ser substituído pelas técnicas de desfuzificação na lógica difusa (CoG) [13]. Assim, Yen *e* Pfluger propuseram implementar cada comportamento por um conjunto de regras difusas, conforme mostra a Figura 34.

A formulação lógica difusa da abordagem de Payton e Roseblatt introduziu conceitos úteis no projecto de sistemas de controlo baseados em comportamentos. Contudo, a fusão de comandos e a selecção da acção, usando a inferência difusa clássica e técnicas de desfuzificação, respectivamente, podem ter algumas desvantagens, como já foi referido neste capítulo. Foi então proposta a utilização de um novo método de desfuzificação, chamado *Centroid of Largest Area* (CLA), em que é seleccionado o comando que corresponde à maior área de um conjunto difuso, como se mostra na Figura 35.

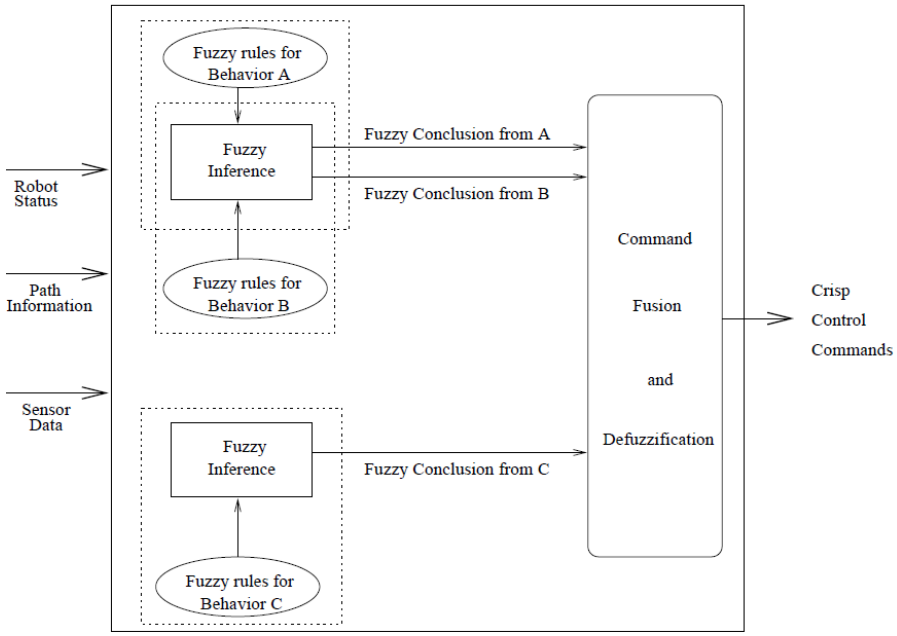
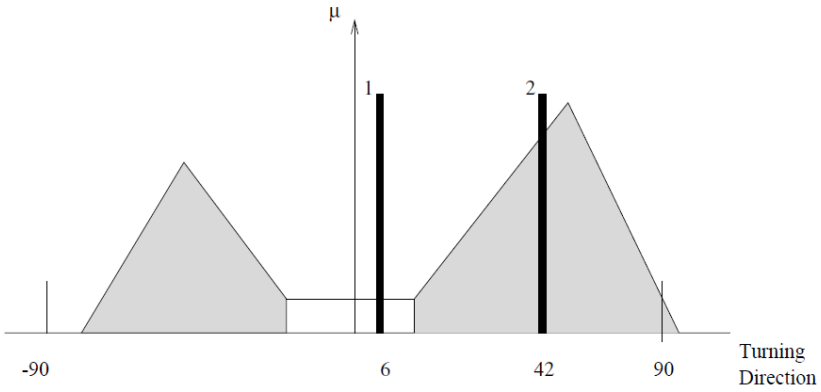


Figura 34 A arquitectura DAMN baseada na lógica difusa [13].

Mas, como foi referido na Tabela 3, este método acarreta alguma complexidade computacional, para além de uma pequena variação na entrada poder causar uma grande variação da saída. Adicionalmente esta técnica nem sempre produz resultados apropriados, como se pode verificar na Figura 36. Contudo, continua a ser recomendado um projecto cuidadoso dos controladores baseados em comportamentos.



1) Center of Area for the Entire Fuzzy Set
2) Centroid of the Largest Area

Figura 35 Técnica de desfuzificação CLA [13].

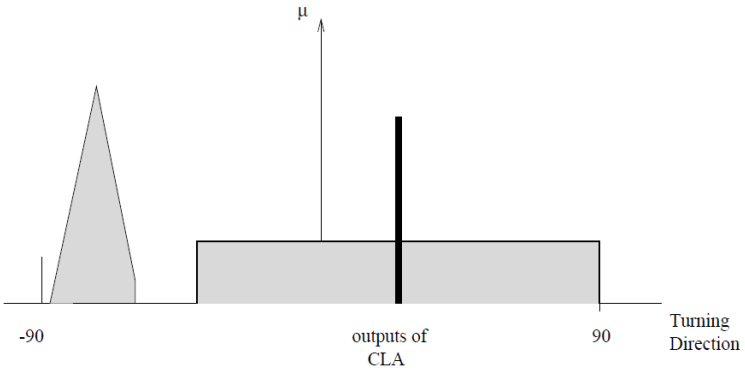


Figura 36 Exemplo de um resultado inapropriado na aplicação da técnica CLA [13].

3.3.5. *MULTIPLE OBJECTIVE BEHAVIOR COORDINATION*

Finalmente, a última metodologia para a fusão de comandos apresentada na Figura 23, a classe designada por *multiple objective behavior coordination*, é na sua essência muito semelhante ao coordenador de comportamentos difuso [12]. Cada comportamento calcula uma função objectivo ao longo de um conjunto de acções admissíveis. A acção que maximiza a função objectivo é considerada como o objectivo mais satisfatório. A selecção da acção inclui a geração, e depois a selecção, de um conjunto de soluções satisfatórias entre um conjunto de soluções eficientes, conhecidas como as soluções óptimas de Pareto.

Esta classe não será abordada, devido à complexidade da mesma e por se afastar da teoria difusa, bem como à extensão considerável deste trabalho.

A temática relacionada com sistemas autónomos é vasta e complexa, existindo muitas mais áreas de interesse que seria interessante abordar. Devido à sua complexidade e dimensão torna-se impossível inseri-las neste trabalho. Deste modo, de seguida, são apresentadas algumas aplicações que utilizam a lógica difusa.

Devido a existirem inúmeros trabalhos de investigação utilizando diferentes abordagens e teorias torna-se praticamente impossível abordar todas as possibilidades num único trabalho/documento.

Desta forma, pretendeu-se clarificar a posição da lógica difusa na robótica, e abordar algumas questões relevantes sobre a sua aplicação nesta área.

3.4. APLICAÇÕES

Nesta secção serão referidos três exemplos de aplicações relacionadas com a temática deste trabalho, ou seja, controladores difusos.

No primeiro exemplo será abordado o trabalho "*A Fuzzy Logic Based Navigation of a Mobile Robot*", de *Fatmini et al*, onde é apresentado um sistema de navegação baseado em comportamentos, a partir da utilização da lógica difusa [14].

No segundo exemplo será referido o trabalho "*Fuzzy Logic Based Behaviors Blending For Intelligent Reactive Navigation of Wlaking Robot*", de *Al-Jumaily* e *Amin*, onde é discutido um sistema de navegação baseado em comportamentos, a partir da utilização da lógica difusa, para um robô com quatro pernas [15].

No terceiro e último exemplo, é apresentado o trabalho "*Fuzzy Logic Controller for an Autonomous Mobile Robot*", de *Peri*, onde é vista a utilização de um controlador difuso para um robô autónomo móvel cuja tarefa principal é seguir paredes [17].

3.4.1. EXEMPLO 1

No exemplo apresentado em seguida utiliza-se um sistema de navegação baseado em comportamentos, onde se aplica a ideia "dividir para reinar", tornando o sistema modular. Isto simplifica a solução da navegação e oferece a possibilidade de adicionar novos comportamentos sem um aumento da complexidade [14].

O sistema de navegação considera vários comportamentos, tais como, *Goal Reaching, Emergency Situation, Obstacle Avoidance,* e *Wall following.* Os comportamentos foram representados usando regras difusas do tipo *IF…THEN,* como é apresentado a seguir [14].

$$R^{(l)}: \text{IF } x_1 \text{ is } A_1^l \text{ and } \ldots \text{and } x_n \text{ is } A_n^l, \text{ THEN } y \text{ is } B^l$$

Sendo *l=1,…m* e *m* é o numero de regras dado pela base de regras, $x_{1\ldots}\, x_n$ são as variáveis de entrada provenientes do sensores do robô, $A_1^l \ldots A_2^l$ são as conjuntos difusos de entrada. Por último, B^l é o conjunto difuso de saída e *y* é a variável de saída.

Verifica-se que vários comportamentos podem estar activos simultaneamente dependendo da situação ou do contexto. A técnica de coordenação deste trabalho resolve o problema de

activação de vários comportamentos simultaneamente e/ou independentemente. O que distingue este trabalho é a forma como a coordenação de comportamentos activos foi realizada. A camada de supervisão baseada em contextos toma a decisão de qual comportamento se deve processar (activar), em vez de processar todos os comportamentos e, em seguida, mistura (*blending*) os apropriados, poupando assim tempo e recursos computacionais.

A título de exemplo, o comportamento *Goal Reaching* tende a levar o robô para a esquerda, direita ou em frente, com baixa ou alta velocidade, dependendo da distância para o objectivo (*Distance to goal, D_{rg}*) e da diferença entre a posição desejada e a posição real (θ_{error}). Embora não exista restrição quanto à forma das funções de pertença, as funções apropriadas para D_{rg} e θ_{error} são as apresentadas na Figura 37 [14].

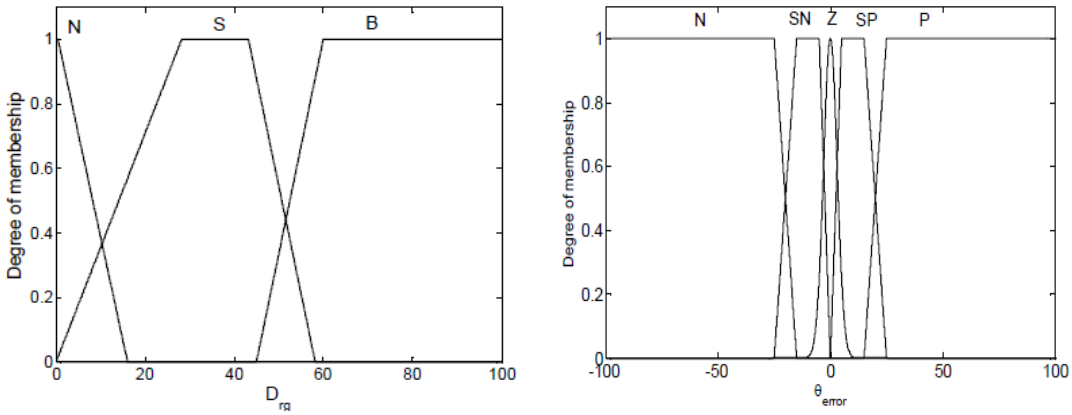


Figura 37 Conjuntos difusos para D_{rg} (Esquerda) e conjuntos difusos para θ_{error} (Direita) [14].

O comportamento *Goal Reaching* tende a alinhar a posição do robô com a direcção do objectivo, e as regras para a realização desta tarefa, que podem ser lidas da tabela, são do seguinte formato:

$$\text{If } \theta_{eror} \text{ is } P \text{ And } D_{rg} \text{ is } Big \text{ THEN } Velocity \text{ is SP}$$

$$\text{If } \theta_{eror} \text{ is } P \text{ And } D_{rg} \text{ is } Big \text{ THEN } Steering \text{ is } L$$

onde P significa *positive*, *Velocity é* uma variável linguística de saída, e SP significa *small positive.*

Neste trabalho existem duas variáveis linguísticas de saída, ou duas varáveis de saída; são elas, *Velocity* e *Steering*, que são definidas pelos conjuntos difusos apresentados na Figura 38.

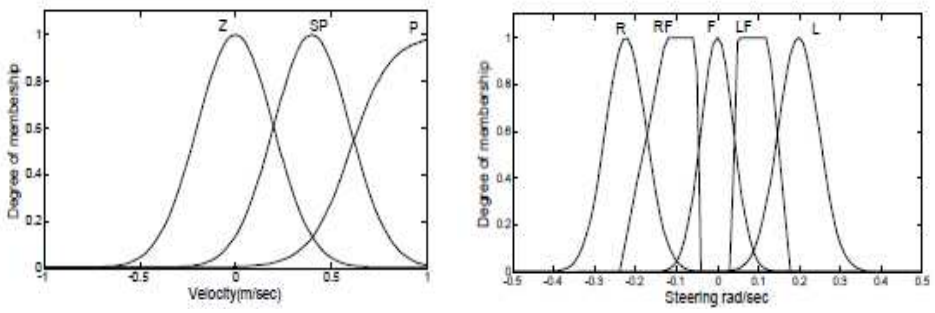


Figura 38 Conjuntos difusos para *Velocity* (Esquerda) e conjuntos difusos para *Steering* (Direita).

Para outros comportamentos o robô necessita de adquirir informação relativa ao ambiente, nomeadamente, a distância aos obstáculos. Para isso necessita de uma nova variável linguística, designada de "*Distance to obstacle"*, que usará conjuntos difusos para representar as leituras dos sensores. A definição desta variável linguística é apresentada na Figura 39.

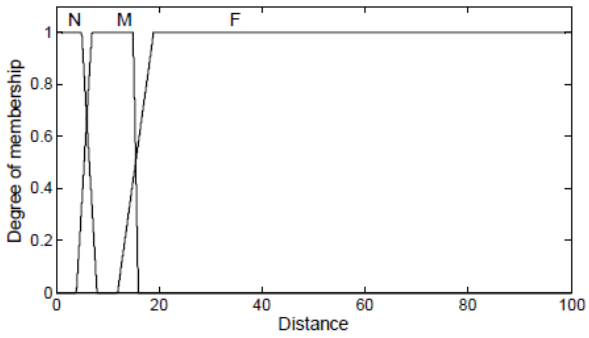


Figura 39 Variável linguística "*Distance to obstacle*" [14].

Na Figura 40 são apresentadas as tabelas, onde estão definidas as regras, e as acções a tomar, em consequência das variáveis de entrada θ_{error} e D_{rg}, para as variáveis de saída *Velocity* e *Steering.*

Os outros comportamentos implementados são: *Obstacle avoidance, Wall following* e *Emergency situation.* O comportamento *Obstacle avoidance* tende a direccionar o robô de modo a evitar colisões; o comportamento *Wall following* mantém o robô numa distância segura das paredes mantendo-se alinhado com estas; por último, o comportamento *Emergency situation* tende a guiar o robô para longe de obstáculos em forma de U, ou similares, indiciando caminhos sem saída.

A camada de supervisão baseia-se em contextos e decide qual (quais) o(s) comportamento(s) que deve(m) ser processado(s)/activo(s) e, em seguida mistura (*blending*) apenas aqueles que são apropriados, poupando recursos computacionais.

D_{rg} \ θ_{error}	*Z*	*SN*	*N*	*SP*	*P*
Near	*Z*	*Z*	*Z*	*Z*	*Z*
Small	*P*	*P*	*SP*	*SP*	*SP*
Big	*P*	*P*	*SP*	*P*	*SP*

D_{rg} \ θ_{error}	*Z*	*SN*	*N*	*SP*	*P*
Near	*F*	*RF*	*R*	*LF*	*L*
Small	*F*	*RF*	*R*	*LF*	*L*
Big	*F*	*RF*	*R*	*LF*	*L*

Figura 40 Tabelas com as regras e as acções a tomar para as variáveis de saída, *Velocity* (Superior) e *Steering* (Inferior) [14].[1]

Na Figura 41 é apresentada a estratégia de mistura de contextos, sendo S_i e V_i (i=1,…,4) as saídas *Steering* (direcção) e *Velocity* (velocidade) de cada comportamento.

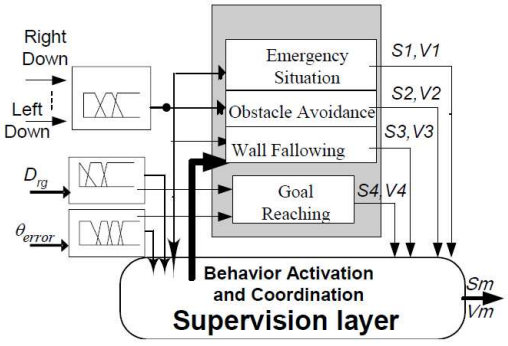


Figura 41 Estratégia de "mistura" de contextos (*Context blending strategy*) [14].

[1] Significado: N- *Negative*, SN- *Small Negative*, Z- *Zero*, P- *Positive*, SP- *Small Positive*, F- *Forward*, RF- *Right forward*, LF- *Left forward*, L-*Left*, R- *Right.*

As entradas de supervisão são as distâncias aos obstáculos, medidas por cada sensor, assim como D_{rg} e θ_{error}. A camada de supervisão é construída a partir de uma base de regras do tipo:

IF context THEN behavior.

A título de exemplo, uma das regras poderia ser:

$R^{(1)}$: IF RU is F and FR is F and FL is F and LU is F, THEN Goal Reaching.

Onde *RU*, *FR*, *FL* e *LU* designam *Right up*, *Front right*, *Front left* e *Left up* e são, respectivamente, leituras dos sensores, como se mostra na Figura 42 [14]. *F* significa *Far* (longe) e *Goal Reaching* corresponde ao comportamento "Alcançar objectivo" [14].

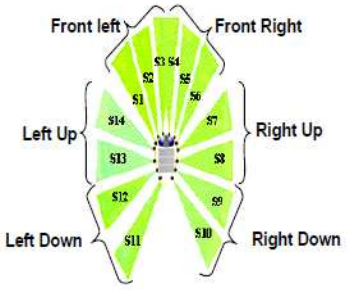


Figura 42 Disposição dos sensores no robô [14].

3.4.1.1. SIMULAÇÃO E RESULTADOS

Para validar o esquema proposto, foram testados alguns casos típicos, nos quais o robô move-se a partir duma dada posição inicial até uma posição final pretendida, em vários ambientes desconhecidos. Confirmou-se em todas as situações que o robô era capaz de atingir a posição final evitando obstáculos. Na Figura 43 é apresentada uma das simulações.

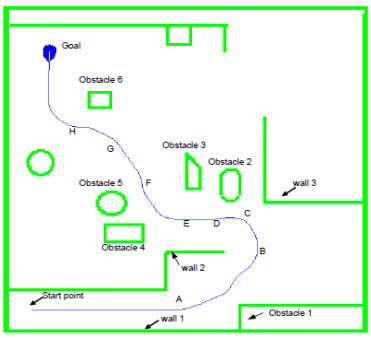


Figura 43 Navegação num ambiente superlotado (*crowded environment*) [14].

Os níveis de activação para cada comportamento são apresentados na Figura 44, onde se encontram marcados com letras os vários obstáculos referidos na Figura 43.

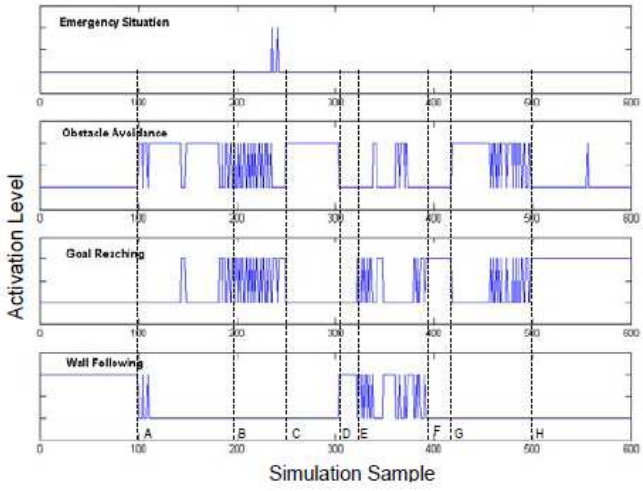


Figura 44 Níveis de activação de cada comportamento [14].

Esta imagem permite verificar a activação dos diferentes comportamentos consoante o contexto, ou seja, consoante a situação em que o robô se encontra. Este tanto tem de alcançar o objectivo como tem que reagir ao ambiente, de modo a evitar os obstáculos. É também perceptível na mesma figura, que o robô teve a maioria do percurso o comportamento "Evitar obstáculo" activo.

A eficácia da abordagem de navegação sugerida neste trabalho foi experimentalmente demonstrada numa plataforma robótica chamada Pekee (*kit* robótico, da *Wanny Robotics*), um robô com duas rodas de tracção e uma de suporte e 15 sensores infravermelhos.

3.4.1.2. CONCLUSÕES

Os autores desta plataforma concluíram ser esta uma boa maneira de estruturar a tarefa de navegação, sendo as questões de projecto de comportamentos individuais e das acções de coordenação de comportamentos implementados através da lógica difusa. A técnica de coordenação aplicada consiste em duas camadas: uma camada primitiva de comportamentos básicos e uma camada de supervisão que se baseia em contextos, em que estes decidem qual/quais o(s) comportamento(s) que deve(m) ser activo(s). Na Figura 45 são apresentadas algumas imagens do robô num ambiente com vários obstáculos.

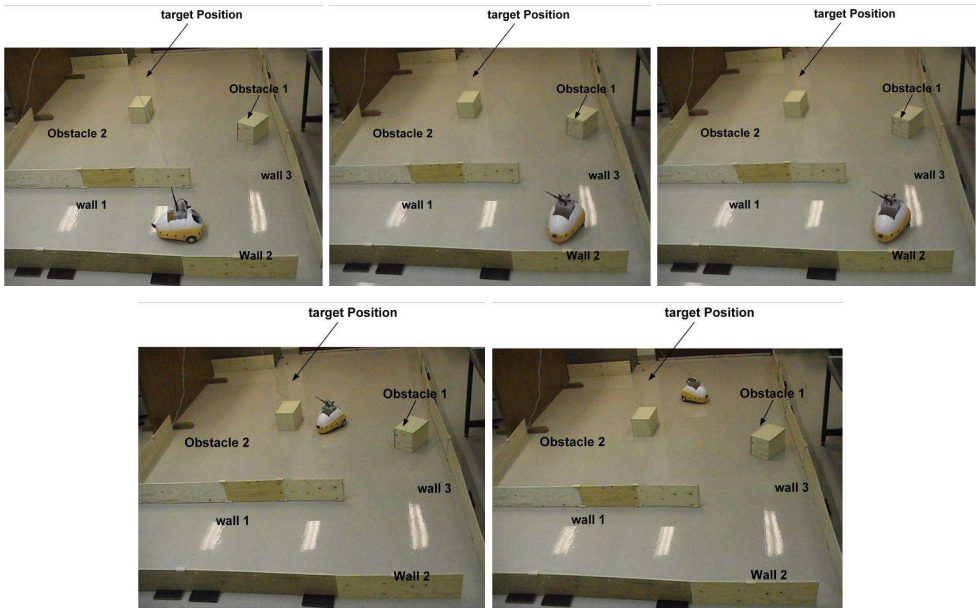


Figura 45 Robô em acção num ambiente complexo [14].

3.4.2. EXEMPLO 2

A reacção de um robô autónomo móvel a um ambiente dinâmico, repleto de incertezas, é uma das questões mais difíceis no controlo de movimentos de robôs autónomos inteligentes. A lógica difusa aparece como uma ferramenta útil para lidar com a navegação reactiva inteligente. O trabalho descrito nesta secção consiste na construção de um sistema de navegação reactivo baseado em comportamentos, a partir da lógica difusa, onde o robô terá uma meta a alcançar e deverá evitar obstáculos no seu trajecto [15].

Este trabalho propõe um novo método para misturar e coordenar os vários comportamentos ao mesmo tempo. O conceito apresentado é semelhante ao da mistura de contextos, mas os autores acrescentaram um parâmetro na regra, ficando esta com o formato o seguinte:

IF context THEN p behavior

Onde p, é o grau de importância (*Degree of inportance* - DOI) e assume um valor fixo entre 0 e 1, reflectindo a importância do comportamento. Por exemplo, as acções propostas pelo comportamento “Evitar obstáculo” deverão receber a prioridade mais alta quando existir o perigo de colisão.

O robô mencionado neste trabalho é um quadrúpede, e utiliza dois tipos de sensores: ultra-sónicos e de calor. Os primeiros são usados para detectar a distância aos objectos e os

segundos permitem detectar fontes de calor e, deste modo, detectar o objectivo. A estrutura do robô é visível na Figura 46.

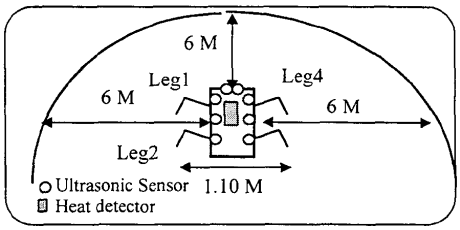


Figura 46 Estrutura do robô com pernas [15].

A arquitectura de controlo difuso adoptada é a apresentada na Figura 47.

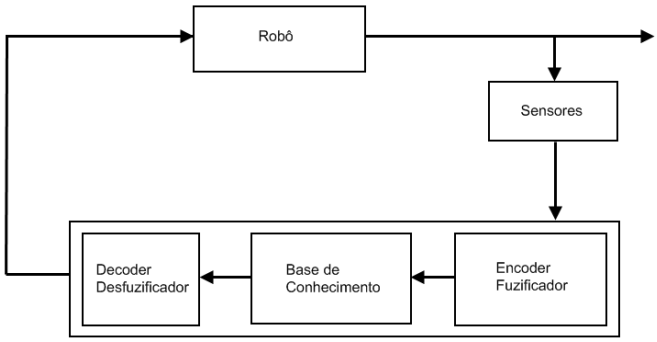


Figura 47 Arquitectura de controlo difusa [15].

Cada comportamento recebe dados provenientes dos sensores relevantes para a tomada de decisão e produz uma saída de controlo baseada na acção desejada. O propósito do trabalho não é construir um mapa do ambiente a partir dos dados dos sensores, mas sim obter algumas indicações imediatas sobre o ambiente do robô ,[15]. Os algoritmos difusos são capazes de activar comportamentos necessários para navegar em ambientes dinâmicos e com o propósito de cumprir um movimento do robô num ambiente desconhecido, foram construídos os seguintes tipos de comportamentos de navegação reactiva: Comportamento *Move, Stop, Reaching Goal* (e os seus próprios sub-comportamentos) e *Obstacle avoidance* e os seus sub-comportamentos. Para melhor percepção, mostram-se na Figura 48, os diversos comportamentos.

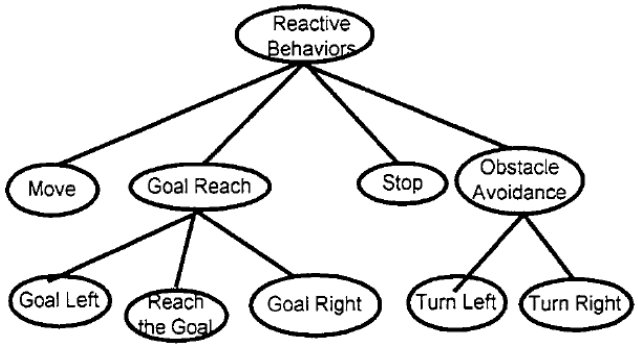


Figura 48 Comportamentos difusos [15].

Os DOI associados a cada comportamento foram definidos do seguinte modo: *Stop* - grau 1*; Obstacle avoidace* - grau 0,9; *Goal reaching* – 0,7; *Move* o grau 0,6.

Na Figura 49 é apresentado o sistema de navegação reactivo baseado na lógica difusa. As entradas são dados (sinais) fornecidos pelos sensores de detecção de obstáculos e de calor e as saídas são valores bem definidos (*crisp values*), ou comandos, de velocidade e orientação.

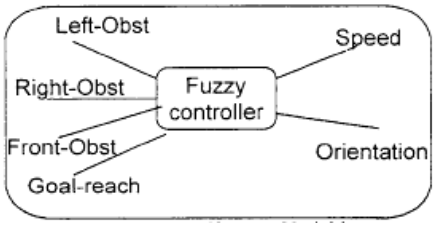


Figura 49 Variáveis de entrada e saída [15].

O robô desenvolvido é um robô bombeiro, cuja função é detectar uma fonte de calor e eliminá-la. Este robô foi dotado de sonares, que cobrem as zonas frontais e laterais do robô, de modo a detectar obstáculos. O sensor utilizado para detectar o objectivo, que neste caso é uma fonte de calor, é um sensor de temperatura. As funções de pertença definidas são trapezoidais e são usadas para representar os valores possíveis das entradas e saídas.

Existem as seguintes variáveis de entrada: *Right, Left, Front Obstacle* e *Goal Reach.* Para a saída existem as variáveis *Speed* (Velocidade) e *Orientation* (Orientação). As variáveis são difusas, sendo por isso definidas por termos que trazem consigo uma representação desse conceito, chamada de função de pertença. Neste trabalho não existem quaisquer referências às funções de pertença que definem as variáveis linguísticas.

No mecanismo de inferência foi utilizado o método Min-Max e para a desfuzificação o método da centróide.

3.4.2.1. SIMULAÇÃO DO SISTEMA

O modelo da simulação calcula a localização do objectivo e dos obstáculos, simula os dados dos sensores, determina a distância aos obstáculos e encaminha a informação para o controlador difuso que toma a decisão sobre o movimento (velocidade e orientação). O robô mover-se-á para a nova posição dependendo da decisão de controlo difusa.

Foram escolhidos diferentes ambientes durante a fase de teste, onde o robô começava num ponto inicial e tomava a decisão para se mover e gerar um caminho usando os comportamentos até que atingisse o objectivo, que neste caso era o calor. Foi testado o comportamento do robô para diferentes posições iniciais e no mesmo ponto. Os resultados são apresentados na Figura 50.

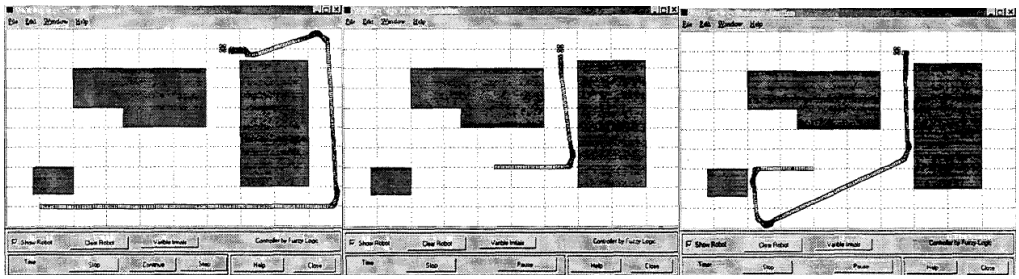


Figura 50 Trajectória descrita pelo robô, através de um *software* de simulação [15].

3.4.2.2. CONCLUSÕES

A navegação reactiva para robôs autónomos com pernas assemelha-se à lógica humana usada para andar por uma cidade repleta de obstáculos. Este sistema depende de uma simples unidade de comportamentos. No entanto é difícil de formular quantitativamente comportamentos reactivos e os projectistas são forçados a considerar todas as possibilidades. Para ultrapassar este problema, foi aplicada a lógica difusa a este problema e os autores apresentam o seu sistema reactivo, o seu esquema de comportamentos, o método para misturar e coordenar mais que um comportamento ao mesmo tempo, e a simulação e animação para testar o sistema desenvolvido, considerando que os resultados obtidos são promissores [15].

Embora os dados contidos neste trabalho sejam resultado de uma simulação, estes parecem apontar para a implementação deste sistema com sucesso. O robô, mesmo partindo de várias posições diferentes, é capaz de encontrar o objectivo.

3.4.3. EXEMPLO 3

No trabalho descrito por Peri [17] é apresentado o desenvolvimento de um robô autónomo que segue paredes e o projecto de um controlador difuso para o controlo do seu movimento. O controlador desta tarefa possui duas entradas e duas saídas. As duas entradas são provenientes das medições referentes à distância das paredes e as duas saídas são as velocidades das duas rodas de tracção. Como se pode ver na Figura 51, o pequeno robô móvel está equipado com uma PIC16F877, dois motores de tracção independentes e três sonares, dois para um lado e um frontal, para situações de emergência, de modo a evitar a colisão do mesmo com algum obstáculo. Para registar os valores obtidos durante os testes o robô necessita de estar ligado com um PC, através de uma ligação série (RS-232). No entanto, em funcionamento normal não necessita de qualquer dispositivo externo para efectuar as suas tarefas.

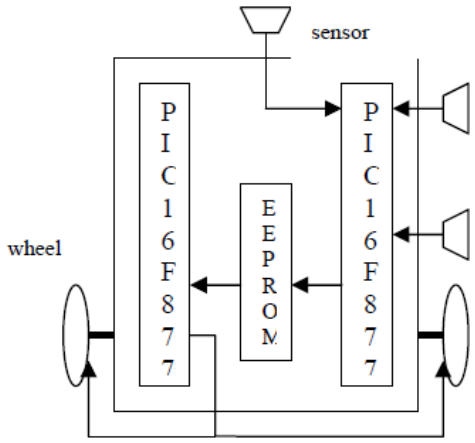


Figura 51 Diagrama de blocos do *hardware* usado [17].

A detecção de uma parede activa o controlador que, simplesmente, tenta alinhar o robô com a parede, mantendo uma distância de referência previamente especificada. O robô segue a parede e tenta manter-se alinhado durante o movimento. Existem vários comportamentos que podem ser modelados, tais como, seguir paredes, evitar colisões, seguir corredores, procurar objectivo (exemplo, fonte de calor), etc. Na Figura 52 pode-se ver uma decomposição hierárquica de alguns dos comportamentos possíveis.

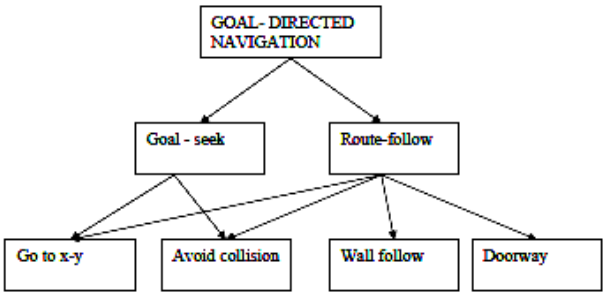


Figura 52 Decomposição hierárquica de comportamentos [17].

O controlador difuso usado tem duas entradas: o erro na posição e o erro no ângulo do robô. Para ajudar a projectar o sistema de controlo difuso foi utilizado o MATLAB. O diagrama de blocos do sistema é apresentado na Figura 53.

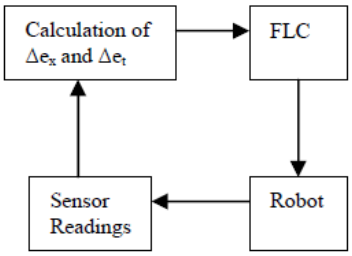


Figura 53 Diagrama de blocos do sistema [17].

Analisando a Figura 54, verifica-se que as duas entradas são o erro no ângulo de orientação (θ), e o erro na distância *x*. A saída do controlador, que é um sinal de controlo, é um sinal PWM de modo a controlar a velocidade angular dos motores.

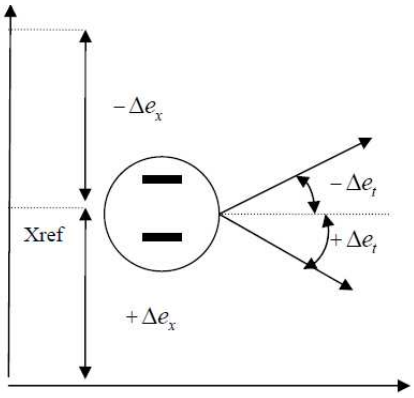


Figura 54 Robô no espaço cartesiano [17].

O universo de discurso do erro no ângulo de orientação Δe_t, varia normalmente entre mais e menos o ângulo máximo da orientação. Neste caso, para simplificar o modelo, a gama foi reduzida ao intervalo entre – 0,3 e + 0,3 rad.

O universo de discurso do erro na distância Δe_x varia entre mais e menos o valor da distância de referência a ser conservada. Para simplificar, a gama de variação neste trabalho situa-se entre – 4 e 4 polegadas, assumindo-se que a distância de referência não é definida para valores superiores a 4 polegadas da parede.

A saída do controlador difuso é a mudança na velocidade angular dos dois motores, Δw_r e Δw_l (sinais de controlo). As suas gamas dependem das restrições mecânicas, e situam-se entre os – 3 e 3 rad.s^{-1}.

As variáveis linguísticas são o erro no ângulo (Δe_t), o erro na distância (Δe_x) e a velocidade dos motores (Δw_r e Δw_l). O erro na distância e o erro no ângulo de orientação são arredondados, uma vez que as medições efectuadas pelos sonares também são arredondadas à polegada.

As três variáveis linguísticas podem assumir três valores linguísticos. As variáveis linguística erro no ângulo e erro na distância, Δe_t e Δe_x, respectivamente, podem assumir os valores linguísticos N (*Negative*), Z (*Zero*) e P (*Positive*). As variações na velocidade angular dos motores podem assumir os valores linguísticos S (*Slow*), M (*Medium*) e F (*Fast*). Cada valor linguístico é definido por funções de pertença triangulares, apresentadas nas Figuras 55 a 58. As funções de pertença triangulares são usadas pela sua simplicidade; por exemplo, a fuzificação das entradas e o cálculo da área é mais rápido e simples quando comparado com outras funções de pertença, como as de forma Gaussiana.

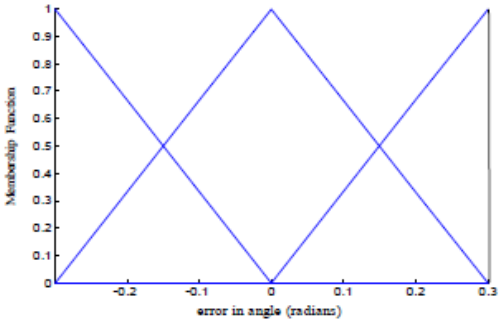


Figura 55 Erro no ângulo de orientação [17].

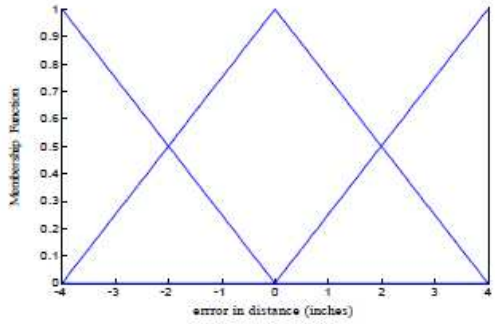


Figura 56 Erro na distância [17].

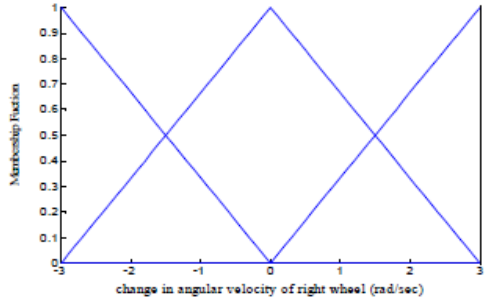


Figura 57 Mudança na velocidade angular da roda direita [17].

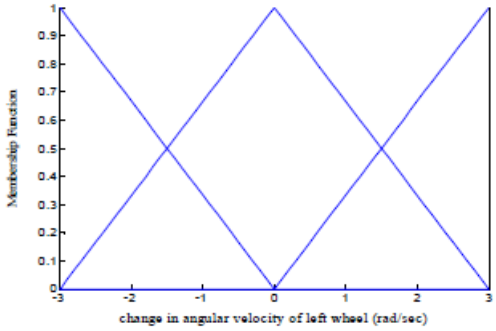


Figura 58 Mudança na velocidade angular da roda esquerda [17].

Gerar as regras para o sistema de inferência difusa é um dos passos mais difíceis no projecto e requer algum conhecimento sobre a dinâmica do processo. A base de regras do controlador difuso proposta neste trabalho é apresentada nas Tabelas 4 e 5. Das 18 regras que aqui se encontram (no total para os dois motores), apenas duas estão activas num dado instante de modo a facilitar a compreensão e o *debug*.

Tabela 4 Base de regras para a mudança na velocidade angular do motor direito [17].

$\Delta e_x / \Delta e_t$	N(Negative)	Z(Zero)	P(Positive)
N(Negative)	S(Slow)	S(Slow)	S(Slow)
Z(Zero)	S(Slow)	F(Fast)	M(Medium)
P(Positive)	S(Slow)	M(Medium)	F(Fast)

Tabela 5 Base de regras para a mudança na velocidade angular do motor esquerdo [17].

$\Delta e_x / \Delta e_t$	N(Negative)	Z(Zero)	P(Positive)
N(Negative)	F(Fast)	M(Medium)	S(Slow)
Z(Zero)	M(Medium)	F(Fast)	S(Slow)
P(Positive)	S(Slow)	S(Slow)	S(Slow)

Na Figura 59, é apresentado um exemplo de como se obtêm os valores de controlo para um dado erro na distância e no ângulo, sendo possível ver as regras activas e qual o valor obtido na desfuzificação.

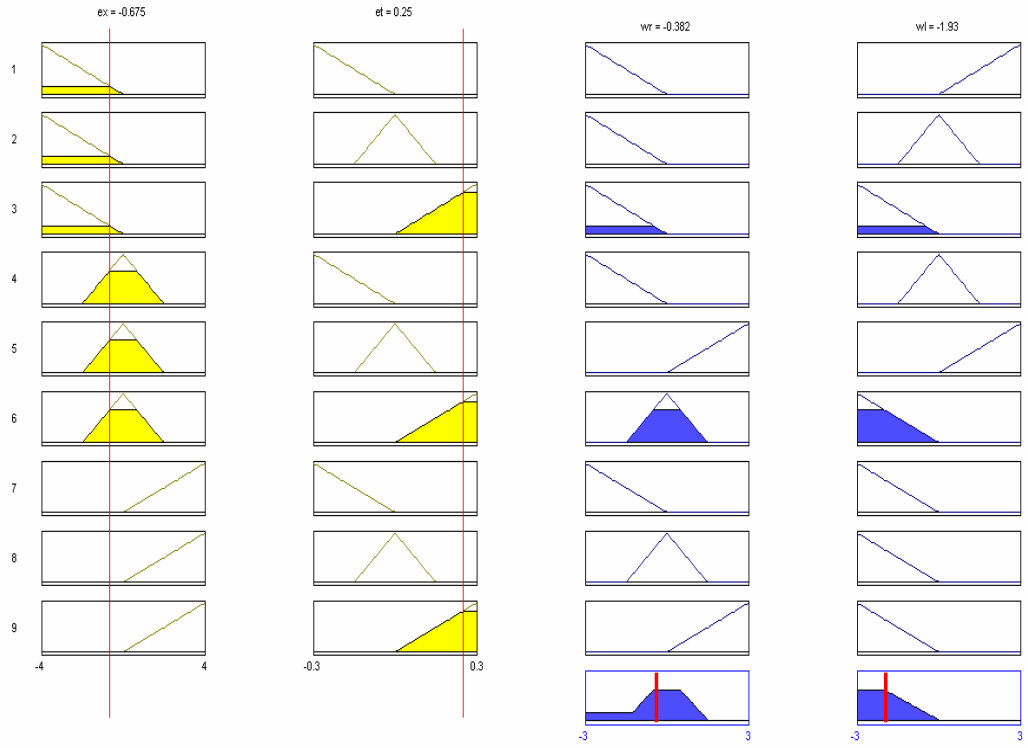


Figura 59 Exemplo da determinação dos valores de saída, para valores de entrada definidos [17].

As superfícies de controlo obtidas para este controlador difuso são apresentadas nas Figuras 60 e 61, para a variação da velocidade angular no motor direito e esquerdo, respectivamente.

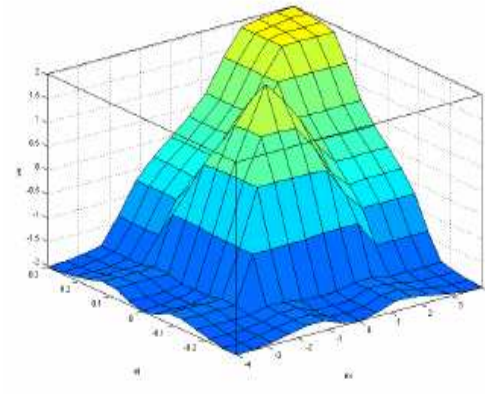


Figura 60 Superfícies de controlo para a variação da velocidade angular do motor direito [17].

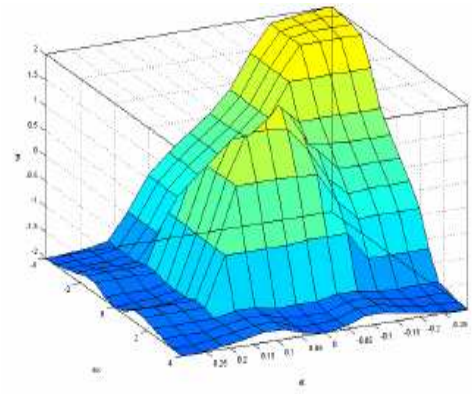


Figura 61 Superfícies de controlo para a variação da velocidade angular do motor esquerdo [17].

Neste caso, estas são representações das superfícies de controlo, ou seja, dos sinais de controlo (velocidades angulares de ambos os motores) em função do erro no ângulo e na distância.

Uma vez que as entradas estão fuzificadas, basta enquadrar o valor das entradas em relação ao valor linguístico da variável linguística que as define e o controlador difuso sabe o valor do grau de verdade das funções de pertença de cada uma das entradas. O antecedente de cada regra deve então ser resolvido por intermédio de um operador difuso. O operador utilizado neste trabalho é o AND, que faz com que seja considerado apenas o valor mais baixo dos graus de verdade que se verificam nas funções de pertença nas duas regras

activas. O mínimo valor é aplicado à função de pertença de saída da respectiva regra particular. A função de pertença é redimensionada para este valor. No final, as funções de pertença de saída são agregadas usando o operador difuso OR (máximo). A desfuzificação é feita com o método centro da área (CoA) de modo a obter os valores de controlo a aplicar. Este método foi escolhido porque produziu os melhores resultados de todos os métodos testados em MATLAB. Os outros métodos que foram usados para desfuzificação são os métodos bissetriz, média dos máximos, primeiro máximo e último máximo.

3.4.3.1. RESULTADOS EXPERIMENTAIS

O esquema Simulink utilizado para a simulação é apresentado na Figura 62.

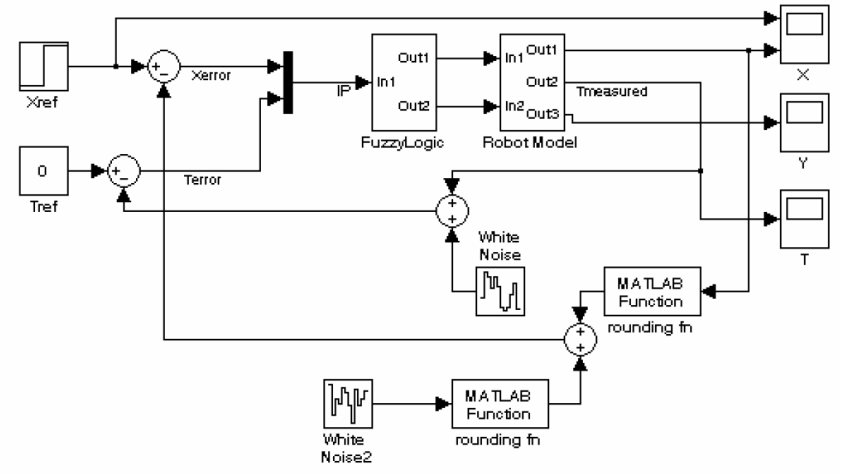


Figura 62 Modelo em Simulink do robô autónomo [17].

As Figuras 63 e 64, mostram os resultados das simulações obtidas para a resposta do robô utilizando um controlador difuso (FLC) e um controlador proporcional (P). O controlador proporcional foi afinado através de algoritmos genéticos. Pretende-se que o robô se mantenha a uma distância de 4 polegadas da parede, partindo de uma distância de 0 polegadas da parede, como se vê nas figuras referidas. As linhas sólidas representam a resposta do controlador difuso e as linhas a tracejado a resposta do controlador proporcional. Com base nos gráficos apresentados nas Figuras 63 e 64, o autor salienta que o controlador difuso atinge o regime permanente em menor tempo.

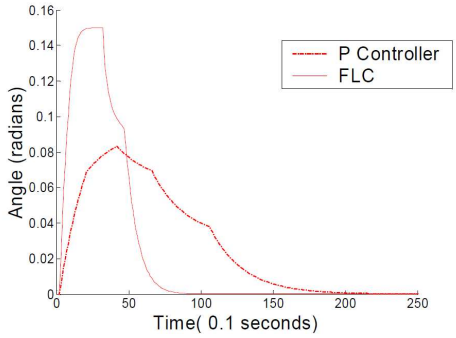


Figura 63 Ângulo de orientação em função do tempo [17].

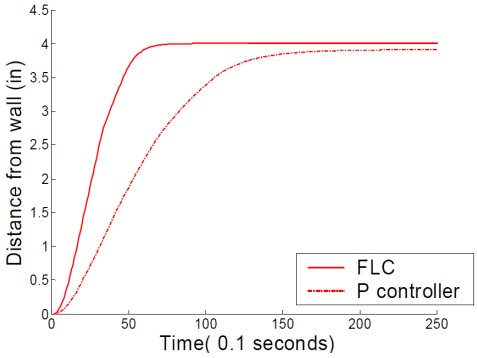


Figura 64 Distância da parede em função do tempo [17].

3.4.3.2. CONCLUSÕES

Segundo Peri [17], verifica-se que os controladores difusos são capazes de controlar o robô de maneira eficiente e de modo a este atingir qualquer posição final independentemente da sua posição inicial. Dos resultados apresentados verifica-se que este sistema teve melhor desempenho que o controlador proporcional e que foi capaz de atingir os objectivos com sucesso. Como sugestões para desenvolvimentos futuros, o autor espera melhorar o sistema usando um processador mais poderoso, de maneira a efectuar os cálculos difusos em vez de utilizar uma *lookup table*, adicionaria também mais sonares de modo a seguir corredores e, por fim, menciona a intenção de utilizar outras funções de pertença que não as triangulares, de forma a ter uma resposta de controlo mais suave.

4. ARQUITECTURA DO SISTEMA

Após a pesquisa efectuada sobre os assuntos abordados nesta Tese, cuja síntese se apresenta nos Capítulos 2 e 3, tornou-se mais clara a arquitectura que o sistema pode apresentar. Nesta idealização foi levado em conta o facto de se pretender desenvolver este projecto para futuras evoluções/melhorias.

As principais tarefas que o AGV terá de desempenhar são o seguimento de paredes e o evitar obstáculos, sendo necessária a utilização de sonares para a determinação das distâncias a que se encontra das paredes e/ou obstáculos. Para o robô ter a capacidade de evitar os obstáculos e se orientar optou-se pela tracção diferencial, que possui um modelo cinemático simples e um oferece um grau de mobilidade que satisfaz os requisitos do sistema.

4.1. ESTRUTURA IDEALIZADA

A estrutura idealizada para o AGV é a apresentada na Figura 65, na qual se podem identificar os constituintes do sistema. Este possuirá quatro sonares, dois deles frontais e

dois laterais de modo a permitirem efectuar as tarefas de detecção de obstáculos e de orientação a partir de paredes.

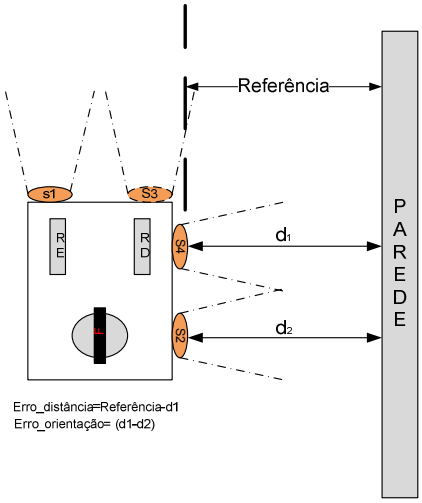


Figura 65 Estrutura idealizada para o AGV.

Para se deslocar, o robô irá possuir dois motores responsáveis pela tracção diferencial, o que possibilita direccionar o veículo em várias direcções. Possui ainda uma roda traseira livre apenas para equilibrar o peso do sistema.

A arquitectura de *hardware* adoptada é justificada com a ideia de tornar o sistema apto a receber futuras melhorias. Deste modo, para tornar mais fácil integrar futuros módulos neste trabalho, optou-se por uma arquitectura distribuída onde a comunicação entre os vários módulos é feita por uma rede CAN.

A arquitectura de *hardware* do sistema é apresentada na Figura 66. É possível verificar que é uma arquitectura distribuída, interligada por uma rede CAN, que será responsável pela interacção entre os vários nós do sistema. Este sistema é composto por três nós distintos. O primeiro é responsável pela aquisição dos valores das leituras dos sensores e envio dos mesmos através da rede CAN. Outro será responsável pela medição da velocidade em cada motor através de um *encoder*, por proceder ao controlo da velocidade de cada um deles e ainda pelo envio dos valores actuais da velocidade e orientação, assim como pela recepção do valor de referência para a velocidade de cada um dos motores. O terceiro, e último nó será responsável pela recepção dos valores das leituras dos sensores, da velocidade e da direcção de cada motor, pelo controlo difuso do sistema e envio das velocidades de referência para cada um dos motores. Este terceiro nó será o “cérebro” do sistema, onde é

implementado o controlador difuso, responsável pela tomada de decisão, que controla o AGV.

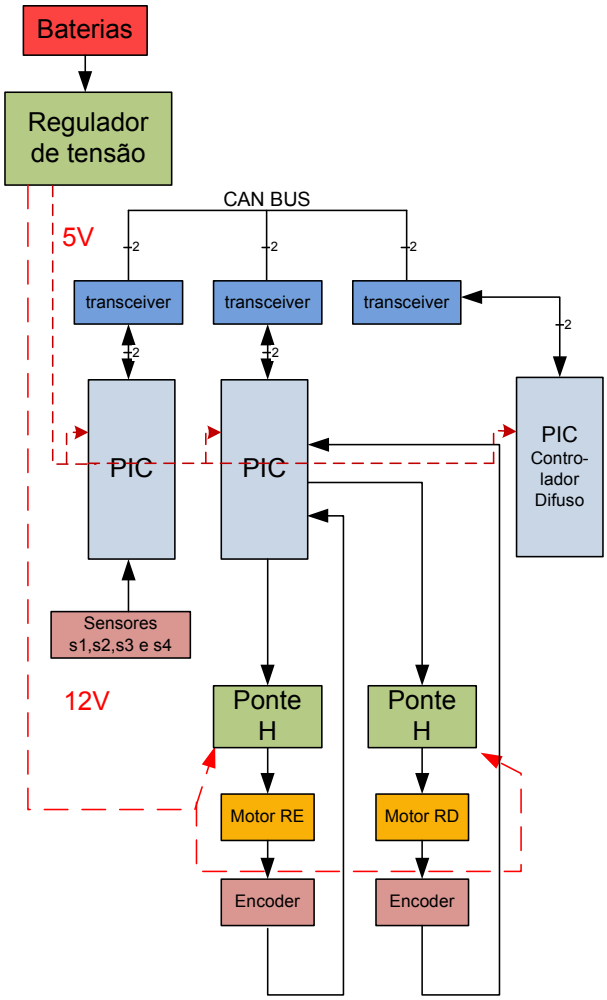


Figura 66 Arquitectura idealizada para o AGV.

4.1.1. MICROCONTROLADOR

Cada nó possui um PIC18F4585 para efectuar todas as tarefas necessárias nesse nó, bem como para o envio/recepção de mensagens através do seu módulo CAN. Este possui também dois canais para gerar PWM o que permite gerar o sinal de controlo para cada um dos motores. Este modelo de PIC foi escolhido pelo facto de possuir todas as funcionalidades necessárias para a realização deste trabalho e de estar disponível.

O microcontrolador PIC18F4585 utiliza instruções de 16 bits, dados de 8 bits e incorpora uma arquitectura RISC (*Reduced Instruction Set Computer*) com uma *stack* com 32 níveis de profundidade. Considerando a gama de processadores de 8 bits da MICROCHIP, estes

processadores apresentam melhor desempenho em relação às famílias anteriores, como se pode constatar através da relação funcionalidade/desempenho, disponibilizada pela MICROCHIP, e apresentada na Figura 67.

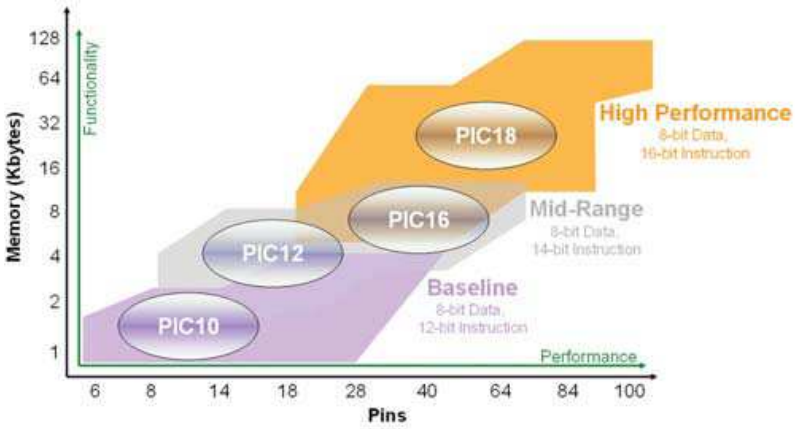


Figura 67 Distribuição das famílias de microcontroladores PIC da arquitectura de 8 *bits*.

Na Tabela 6 são apresentadas as características mais importantes deste microcontrolador.

Tabela 6 Características principais do microcontrolador PIC18F4585.

Descrição PIC 18F4585	
Frequência máxima de trabalho	40 MHz até 10 MIPS
Comunicação	SPI, CAN, RS-232
Controlo	2 Módulos de PWM

Este microcontrolador possui 20 fontes de interrupção, três das quais interrupções externas, com a possibilidade de definir níveis de prioridade (prioridade alta e prioridade baixa). Possui também 48 Kbytes de memória *flash* para memória de programa e, em relação à memória de dados, possui 3328 bytes de memória SRAM e 1024 bytes na EEPROM. No total possui 40 pinos de entrada e 44 de saída, 11 canais A/D, um canal *standard* de PWM e um melhorado que permite várias configurações. Este microcontrolador permite comunicação SPI e I^2C. Por último, possui um temporizador de 8 bits e três de 16 bits com a possibilidade de configuração dos mesmos para os dois modos (8 e 16 bits).

Estas são as principais características técnicas deste controlador, as quais fornecem as funcionalidades necessárias para o desenvolvimento deste projecto.

4.1.2. MOTORES

Em relação aos motores necessários para a construção do AGV, estes tiveram de ser adquiridos. Foram analisados três motores diferentes, comercializados pela Robot Italy, cujas características principais se apresentam de seguida.

O motor com a Ref. 420115, e apresentado na Figura 68, possui as seguintes características principais:

- Velocidade máxima de 66 rpm;
- Binário de 6,12 kg/cm;
- *Encoder* que fornece 3 impulsos por rotação;
- Relação de redução da caixa 94,37:1;
- Pequenas dimensões, com 6,6 cm de comprimento.

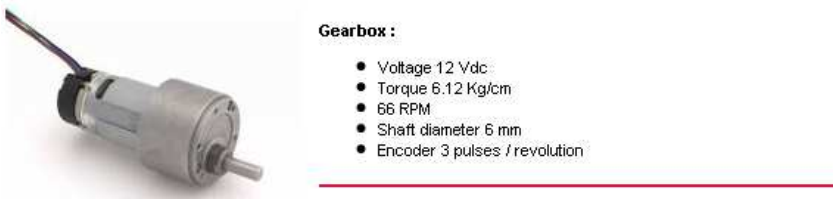


Figura 68 Motor com Ref. 420115 comercializado pela Robot Italy.

O motor apresentado na Figura 69, com a Ref. EMG30 12V, possui as seguintes características:

- Velocidade máxima de 170 rpm;
- Binário de 1,5 kg/cm;
- *Encoder* que fornece 360 impulsos por rotação do veio (exterior);
- Relação de redução da caixa 30:1;
- Ligeiramente superior aos restantes, com 8,6 cm de comprimento.

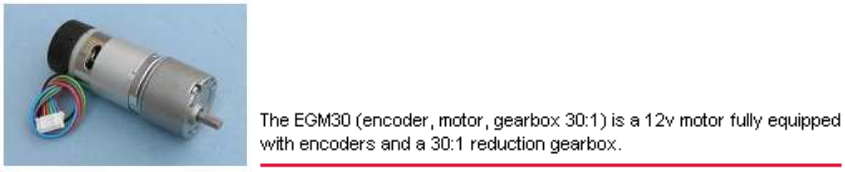


Figura 69 Motor com a Ref. EMG30 12V comercializado pela Robot Italy.

Por último, o motor apresentado na Figura 70, com a Ref. 420123, possui as seguintes características:

- Velocidade máxima de 75 rpm;
- Binário de 2,55 kg/cm;
- *Encoder* que fornece 3 impulsos por rotação;
- Relação de redução da caixa 90,3:1;
- Pequenas dimensões, com 7,6 cm de comprimento.

Figura 70 Motor com a Ref. 420123 comercializado pela Robot Italy.

Após analisar a informação relativa às características dos três motores, e tendo em conta a aplicação que se pretende desenvolver, o motor que melhor se enquadra nas características do projecto é o motor Robot Italy, Ref. 420115, uma vez que é o que apresenta menores dimensões, maiores binários e uma velocidade máxima considerada satisfatória para a aplicação. Se se considerar a utilização de rodas com 3 cm de raio e o motor a rodar à velocidade máxima de 66 rpm, o robô pode atingir a velocidade aproximada de 20 cm/s.

4.1.3. SONARES

Os sensores utilizados no projecto são os sonares SRF05, representados na Figura 71, e que se encontravam disponíveis no ISEP.

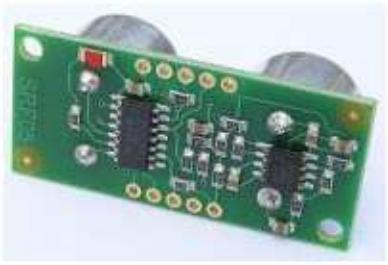


Figura 71 Sonar SRF05.

Este sonar é uma evolução do modelo anterior (SRF04), e que foi desenvolvido para aumentar a flexibilidade apresentando uma gama de funcionamento entre 3 cm e 4 m. A

gama de detecção, a fácil interacção e o baixo consumo fazem deste dispositivo uma boa solução para as aplicações robóticas.

O princípio de funcionamento subjacente a este dispositivo é o seguinte: o transmissor emite um impulso com 40 kHz, que se propaga à velocidade do som, espalhando-se pelo espaço na forma de cone até que seja reflectido num objecto. O controlador faz uma pequena pausa até receber o sinal reflectido. Assim que o eco é recebido, o controlador é informado e calcula a distância através do tempo de voo observado. Em seguida, a linha de *echo* assume o valor lógico 1 durante um intervalo de tempo proporcional à distância medida. Se nenhum objecto for detectado, a duração do intervalo será aproximadamente 36 ms. O diagrama temporal do funcionamento do sonar é apresentado na Figura 72.

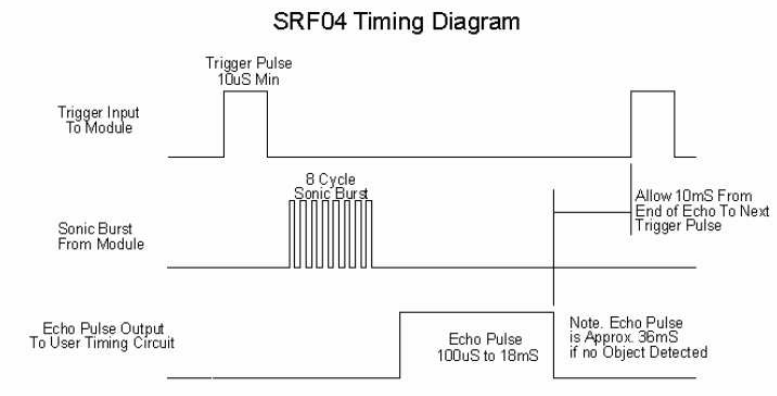


Figura 72 Diagrama temporal do funcionamento do sonar.

Neste trabalho, são utilizados quatro destes sonares com a finalidade de obter informação referente às distâncias existentes entre o AGV e os obstáculos existentes nas proximidades.

4.1.4. RODAS

As rodas utilizadas neste projecto são apresentadas na Figura 73, tendo sido adquiridas na empresa Multi-Borracha, em Alfena. As rodas têm cerca de 3,5 cm de raio; deste modo a força de tracção disponível por motor será de aproximadamente 1,75 N.

Figura 73 Roda escolhida para o projecto.

4.1.5. *HUBS* E SUPORTES PARA O MOTOR

Os suportes para as rodas foram feitos por pedido para este trabalho, podendo ser vistos na Figura 74.

Figura 74 Suporte do motor e *hub*.

4.1.6. ESTRUTURA

A estrutura base do robô foi construída a partir da reutilização de uma placa de acrílico, não sendo necessária a aquisição de material para a sua construção.

Numa primeira fase idealizou-se uma forma adequada para a estrutura (ver Figura 75), e procedeu-se ao corte e retoques para melhorar o acabamento. Numa fase posterior, aquando da montagem de todas as partes constituintes, foram efectuados alguns ajustes à forma da estrutura de modo a melhorar alguns aspectos.

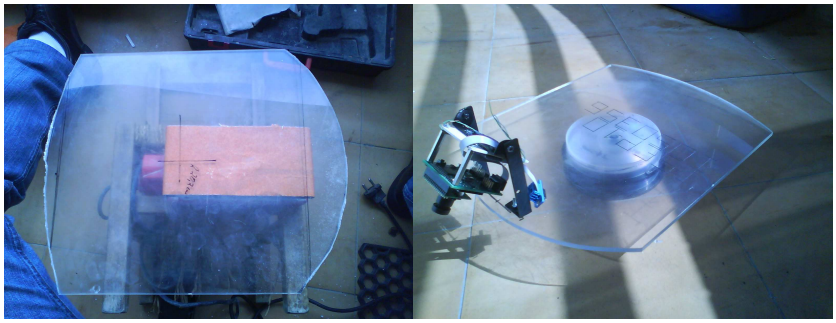


Figura 75 Construção da estrutura.

Na Figura 76 é apresentada a estrutura final do robô, onde se verifica a existência de uma segunda plataforma que foi adicionada de modo a permitir a aplicação de todo o *hardware*.

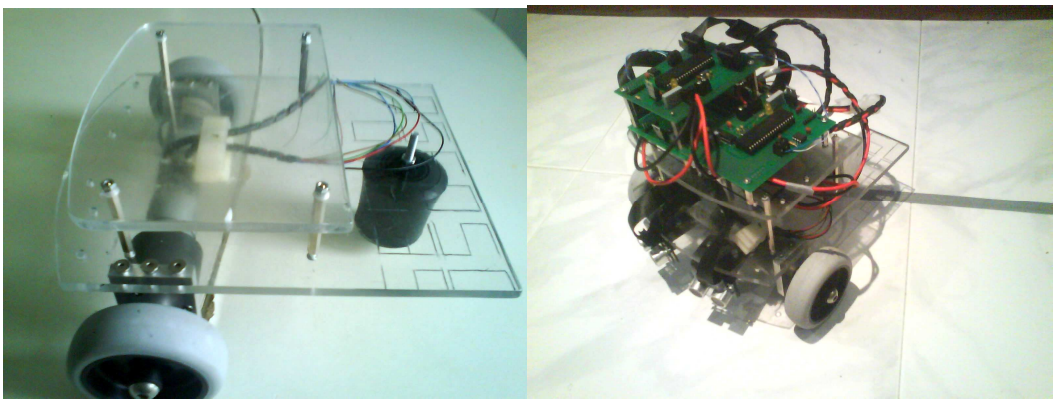


Figura 76 Estrutura final.

Nas secções seguintes deste capítulo será feita uma pequena abordagem teórica ao controlo PID, para que no capítulo 5 sejam mencionados todos os detalhes do projecto do controlo PI e a implementação da rede CAN no sistema. No Anexo A são abordadas, com maior detalhe, as características principais do protocolo CAN.

Na opinião do autor faz sentido abordar estas temáticas pois estas são utilizadas no decorrer do projecto dos vários módulos constituintes deste sistema.

4.2. CONTROLO PID

Os controladores PID são bastante utilizados no controlo de aplicações, principalmente na indústria, devido à facilidade em implementá-los e também aos resultados que proporcionam. Como neste trabalho o controlo da velocidade dos motores é feito recorrendo a estes controladores, efectua-se aqui uma pequena abordagem a esta temática.

O principal objectivo destes controladores é corrigir o erro existente entre a entrada de referência e a saída do processo, de modo a obter um controlo mais preciso sobre o mesmo.

Estes controladores são compostos por três componentes distintas (são elas a componente proporcional, componente derivativa e a componente integral) onde cada uma tem um efeito diferente na resposta do sistema.

De seguida, abordam-se os quatro controladores geralmente usados no controlo de processos.

4.2.1. CONTROLADOR PROPORCIONAL (P)

O modo proporcional ajusta o sinal de saída numa proporção directa da variável de entrada, que é o sinal de erro. O parâmetro de ajuste é designado como ganho do controlador, K_p. O controlador proporcional reduz o erro em regime permanente do sistema mas não o elimina [8]. A Figura 77 mostra o diagrama de blocos do sistema com o controlador P.

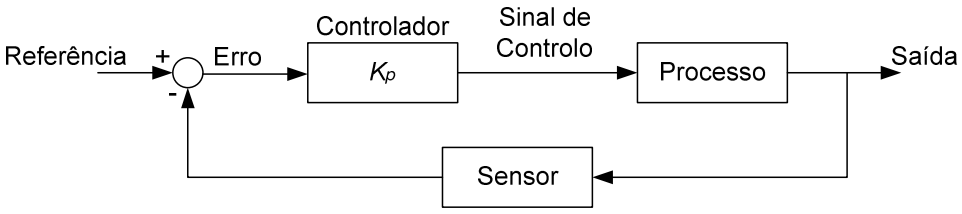


Figura 77 Diagrama de blocos do sistema com o controlador P [16].

As equações nos domínios dos tempos e de Laplace do controlador P são dadas respectivamente por:

$$u(t) = K_p\, e(t) \tag{16}$$

$$U(s) = K_p\, E(s) \tag{17}$$

O equivalente controlador digital possui a seguinte função de transferência discreta e equação às diferenças, dadas por:

$$\frac{U(z)}{E(z)} = K_p \tag{18}$$

$$u(k) = K_p\, e(k) \tag{19}$$

4.2.2. CONTROLADOR PROPORCIONAL E INTEGRAL (PI)

Como já foi referido, a componente proporcional não elimina o erro em regime permanente. Para isso tem de se introduzir a componente integrativa que permite eliminar o erro do sistema em regime permanente, isto é, anular a diferença entre a entrada de referência e o valor da variável de saída do processo. Se esta componente for aplicada isoladamente piora a estabilidade do sistema. Assim, e para evitar essa situação, a componente integrativa é implementada em conjunto com a proporcional [8]. Na Figura 78 é apresentado o diagrama de blocos do sistema com o controlador PI.

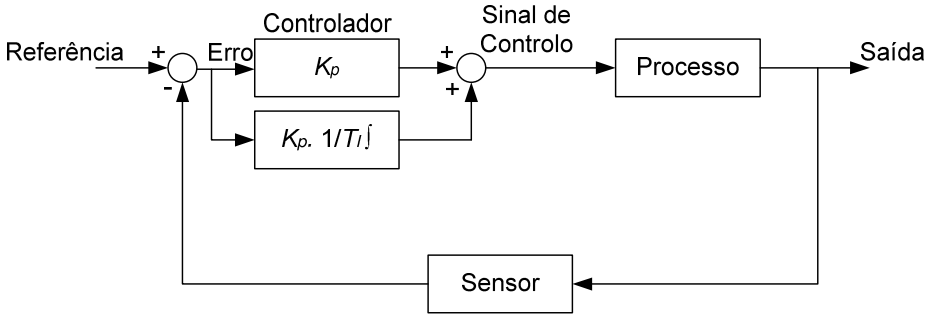


Figura 78 Diagrama de blocos do sistema com o controlador PI [16].

As equações nos domínios dos tempos e de Laplace do controlador PI são dadas respectivamente por:

$$u(t) = K_p \left[e(t) + \frac{1}{T_I} \int e(t)\, dt \right] \tag{20}$$

$$U(s) = K_p E(s) \left[1 + \frac{1}{T_I s} \right] \tag{21}$$

O equivalente controlador digital possui a seguinte função de transferência discreta e equação às diferenças, dadas por:

$$\frac{U(z)}{E(z)} = K \frac{z + \alpha}{z - 1} \tag{22}$$

$$u(k) = u(k-1) + K\, e(k) - K \cdot \alpha\, e(k-1)$$

$$\text{em que, } K_I = \frac{K_P}{T_I} \;;\; K = K_p + \frac{K_I T}{2} \;;\; \alpha = \frac{\frac{K_I T}{2} - k_p}{\frac{K_I T}{2} + k_p} \tag{23}$$

A componente integral resulta da discretização pelo método de Tustin (ou regra trapezoidal), sendo obtida utilizando a seguinte expressão:

$$D_I(z) = K_I \frac{T}{2} \frac{z+1}{z-1} \tag{24}$$

4.2.3. CONTROLADOR PROPORCIONAL E DERIVATIVO (PD)

A saída de um processo apresenta uma certa dinâmica quando ocorrem variações na entrada do mesmo, fazendo com que na saída demore um certo tempo até se verificar a reacção à mudança que ocorreu na entrada. Este facto gera oscilações que ocorrem até se atingir a estabilidade. Estas oscilações podem numa situação limite, levar o sistema para a instabilidade [8]. Para evitar esta situação, utiliza-se um termo derivativo no controlador. A Figura 79 mostra o diagrama de blocos do sistema com o controlador PD.

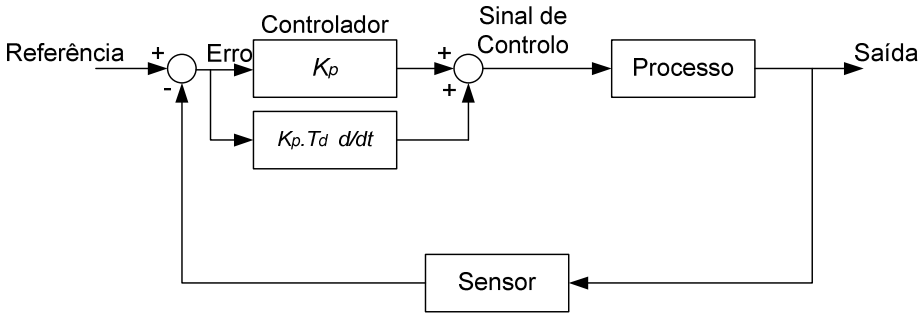


Figura 79 Diagrama de blocos do sistema com o controlador PD [16].

As equações nos domínios dos tempos e de Laplace do controlador PD são dadas respectivamente por:

$$u(t) = K_p\left[e(t) + T_d \frac{de(t)}{dt}\right] \tag{25}$$

$$U(s) = K_p E(s)\left[1 + T_d s\right] \tag{26}$$

O equivalente controlador digital possui a seguinte função de transferência discreta e equação às diferenças, dadas por:

$$\frac{U(z)}{E(z)} = K\frac{z+\alpha}{z} \tag{27}$$

$$u(k) = K\ e(k) - K\alpha\, e(k-1)$$

$$\text{em que, } K_d = K_p \times T_d\ ; K = K_d + \frac{K_d}{T}\ ; \alpha = \frac{K_d}{K_d + K_p T} \tag{28}$$

A componente derivativa resulta da discretização pelo método das diferenças atrasadas, de modo a aproximar a derivada, sendo obtida utilizando a seguinte expressão:

$$D_D(z) = K_d \frac{z-1}{Tz} \tag{29}$$

A componente derivativa permite uma antecipação do valor da saída possibilitando efectuar uma correcção no seu comportamento em tempo útil. Assim, esta componente tende a melhorar a estabilidade do sistema em malha fechada.

4.2.4. CONTROLADOR PROPORCIONAL, INTEGRAL E DERIVATIVO (PID)

O controlador PID reúne as vantagens dos controladores PI e PD, em que a acção integral se vai relacionar com a precisão do sistema, eliminando o erro em regime permanente. O efeito destabilizador provocado pela acção integrativa é compensado pela acção derivativa, que tende a aumentar a estabilidade relativa do sistema, uma vez que torna a resposta deste mais rápida devido à capacidade de antecipação que esta acção lhe proporciona [8]. Na Figura 80 é apresentado o diagrama de blocos do sistema com o controlador PID.

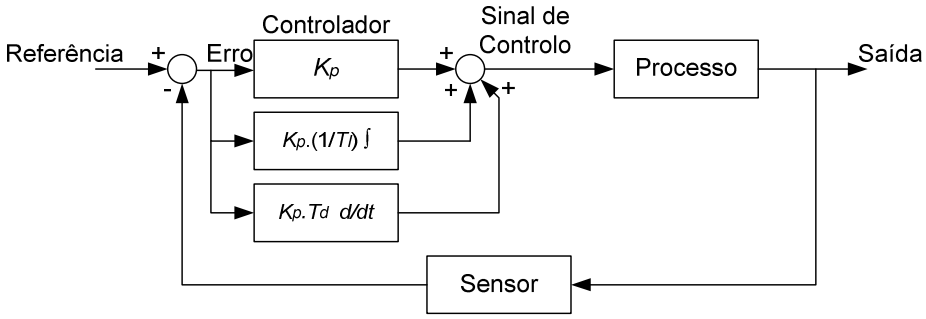


Figura 80 Diagrama de blocos do sistema com o controlador PID [16].

As equações nos domínios dos tempos e de Laplace do controlador PID são dadas respectivamente por:

$$u(t) = K_p \left[e(t) + \frac{1}{T_I} \int e(t)\,dt + T_d \frac{de(t)}{dt} \right] \tag{30}$$

$$U(s) = K_p E(s) \left[1 + \frac{1}{T_I s} + T_d s \right] \tag{31}$$

O equivalente controlador digital possui a seguinte função de transferência discreta e equação às diferenças, dadas por:

$$\frac{U(z)}{E(z)} = \frac{A\,z^2 + Bz + C}{z(z-1)} \tag{32}$$

$$u(k) = u(k-1) + A\,e(k) - KB\,e(k-1) + C\,e(k-2) \tag{33}$$

$$\text{em que, } A = \frac{K_I\,T}{2} + \frac{K_d}{T} + K_p\,; B = \frac{K_I\,T}{2} - 2\,\frac{K_d}{T} - K_p\,;\; C = \frac{K_d}{T}\;) \tag{34}$$

Como foi abordado anteriormente, a componente integral resulta da discretização segundo o método de Tustin e a componente derivativa resulta da aplicação do método das diferenças atrasadas.

Este tipo de controlador permite melhorar várias características do sistema, e existem muitos métodos para a determinação dos valores das componentes proporcional, integrativa e derivativa. Talvez por este facto, e pela simplicidade associada à sua implementação, seja um dos controladores mais usados na indústria.

No Capítulo 5, será abordado o projecto de um controlador do tipo PI para efectuar o controlo da velocidade de um motor de corrente contínua.

5. DESENVOLVIMENTO DO AGV

Neste capítulo são abordados todos os passos relevantes efectuados durante o projecto e desenvolvimento do AGV. São aqui mencionados os projectos do controlador PI (para controlo da velocidade), da rede CAN e do módulo de aquisição de dados provenientes dos sensores ultra-sónicos, assim como o projecto do controlador difuso para o AGV.

5.1. MEDIÇÃO DA VELOCIDADE

Tal como mencionado na secção 4.1.2, os motores escolhidos são da Robot Italy (Ref. 420115). Este motor possui um *encoder* interno que fornece 3 impulsos por rotação do motor. Apesar de, geralmente, se designar por *encoder*, na verdade este é um sensor por efeito de Hall, sendo gerado um sinal que varia de acordo com a variação do campo eléctrico. Neste caso, são gerados três impulsos por cada volta completa do veio do motor.

Associado ao veio interno está uma caixa redutora com a relação de 94,37:1, o que significa que, para cada volta do veio externo, são fornecidos pelo *encoder* 283 impulsos. Estes impulsos permitem medir a velocidade a que o motor está a rodar.

Existem dois métodos possíveis para efectuar a medição da velocidade a partir dos impulsos gerados pelo *encoder*. O primeiro método de medição é designado por medida da velocidade por contagem de impulsos e o segundo por medida da velocidade por contagem de tempo.

No primeiro método obtém-se a velocidade pela contagem de impulsos: através de um contador activado durante um tempo fixo são contados o número de impulsos gerados. Uma vez que é conhecido o número de graus que o veio roda por cada sinal gerado, e como o tempo da contagem é fixo, pode-se calcular a velocidade a que o motor está a rodar. A medição por este método é tanto mais precisa quanto maior for o número de impulsos contados. Por esta razão, este método assume erros relativos mais baixos para velocidades elevadas. Se se aumentar o tempo da contagem, perde-se o significado de velocidade instantânea e passa-se a ter a velocidade média para um período de tempo.

O segundo método baseia-se na activação de um contador para conhecer o valor de tempo que separa dois impulsos consecutivos. A precisão é directamente proporcional ao número de impulsos contabilizados pelo contador. Assim, para aumentar a precisão pode-se aumentar a frequência de contagem do contador. Contudo existem limitações ao nível dos circuitos. Este método tem um erro relativo mais baixo para velocidades reduzidas.

Para se escolher o método de medição da velocidade mais adequado foram calculados os erros relativos de cada método, tendo sido considerado o intervalo de velocidades admissíveis para o motor entre [8, 66] rpm, e uma frequência de contagem para o segundo método de 1,25 MHz. Os resultados são apresentados na Figura 81, onde é visível o erro relativo calculado para cada método na gama de velocidades considerada.

Como se pode verificar pela análise dos gráficos, o primeiro método apresenta um erro relativo superior ao segundo método. Logo, o método escolhido para efectuar a medição da velocidade foi o segundo, no qual a frequência de contagem do contador é de 1,25 MHz.

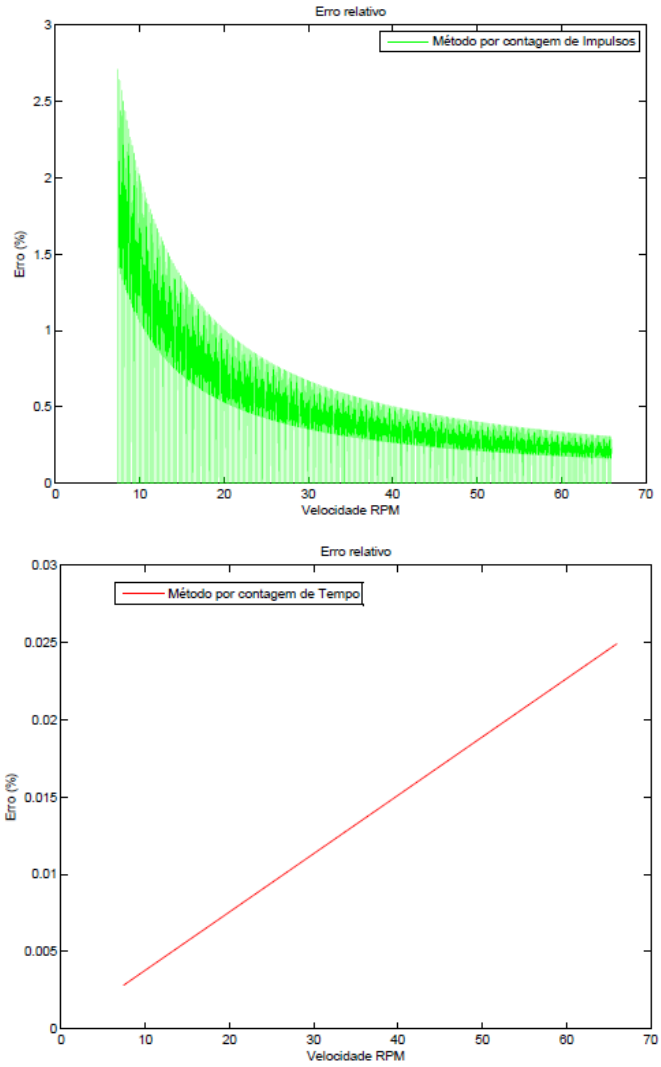


Figura 81 Gráfico do erro relativo calculado para o método de contagem de impulsos (em cima) e método de contagem de tempo (em baixo).

Assim, por cada impulso do *encoder* o veio do motor (veio externo, a seguir à caixa redutora com a relação 94,37) roda aproximadamente 0,0222 radianos; se se dividir este valor pelo número presente no contador multiplicado pelo período de contagem, resulta a velocidade instantânea do motor, tal como se mostra na equação (35).

$$w = \frac{2\frac{\pi}{3}\times\frac{1}{94{,}37}}{n_{impulsos\ contador}\times\left(\frac{1}{1{,}25}\times 10^{6}\right)} = \frac{0{,}0222}{n_{impulsos\ contador}\times 0{,}0000008} \qquad (35)$$

Como se pode ver na Figura 82, a cada sinal gerado pelo *encoder* é gerada uma interrupção, sendo lido o valor da contagem e reiniciado o contador. Em seguida, é calculado o valor da velocidade.

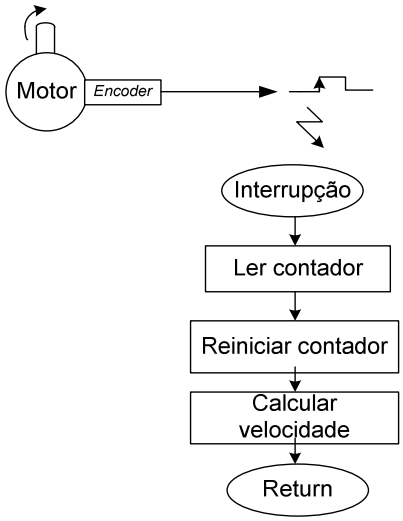


Figura 82 Fluxograma do processo para cálculo da velocidade.

Uma vez implementado o processo de medição da velocidade, procede-se ao projecto do controlador PI, cujo objectivo é controlar a velocidade do motor. Este aspecto do trabalho é abordado em seguida. Este é uma parte importante do projecto uma vez que se deve garantir a estabilidade e precisão dos actuadores do sistema.

5.2. CONTROLADOR PI

Antes de se começar o projecto do controlador PI, é necessário verificar a resposta do motor. Para isso foram realizados testes experimentais ao motor. Numa primeira fase, foi realizado um teste onde foi variada a tensão de alimentação do motor desde 0 V até 12,47 V e foram registados os valores de velocidade que este atingiu. Estes valores são apresentados na Tabela 7, e o diagrama de blocos do sistema com controlador é apresentado na Figura 83.

Tabela 7 Valores registados nos testes: Tensão (V) *vs*. Velocidade (rad/s).

Tensão (V)	Velocidade (rad/s)
0,03	0,00000
0,51	0,00000
1,04	0,00000
1,53	0,00000
2,1	0,00000
2,57	0,00000
3,1	0,00000
3,55	0,78493
4,13	1,10922
5,16	1,81416
5,63	2,08542
6,07	2,36419
6,51	2,76530
7,17	3,18486
7,52	3,39285
8,13	3,78874
8,48	4,07875
9,06	4,45471
9,59	4,78830
10,04	5,13751
10,44	5,40474
11,06	5,78865
11,56	6,14829
11,97	6,43382
12,47	6,76913

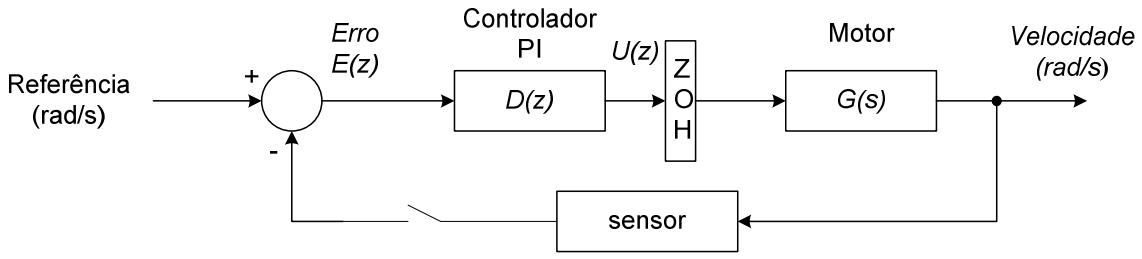


Figura 83 Diagrama de blocos do sistema.

Na Figura 84 representam-se estes resultados sob a forma de um gráfico Tensão (V) *vs.* Velocidade (rad.s^{-1}).

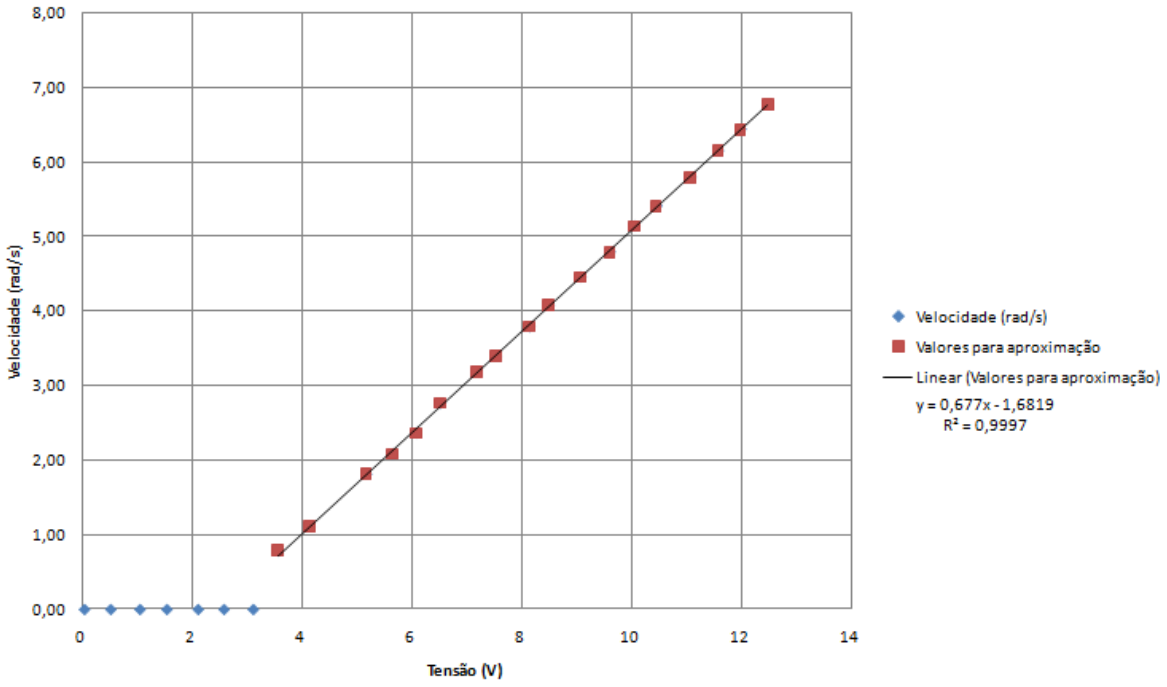


Figura 84 Gráfico dos dados experimentais: Tensão *vs*. Velocidade.

Da análise deste gráfico, pode-se verificar uma zona morta, onde a tensão aplicada ao motor não é suficiente para o pôr a rodar; só a partir dos 3,55 V é que o motor começa a mover-se. Teoricamente, seria de esperar que com aproximadamente 12 V fosse atingida a velocidade máxima de 66 rpm, ou seja, aproximadamente 6,91 rad/s. No entanto, o motor não se aproximou desse valor, possivelmente devido a algumas perdas na ponte H, porque não foram utilizados díodos de resposta rápida.

A função de transferência, entre a posição e a tensão de alimentação do motor de corrente contínua é:

$$\frac{\theta(s)}{U(s)} = \frac{K}{s \cdot (s\tau + 1)} \tag{36}$$

Da equação (36), chega-se à relação entre a velocidade e a tensão, que é apresentada de seguida:

$$\frac{\omega(s)}{U(s)} = \frac{K}{(s\tau + 1)} \tag{37}$$

Para se definir o sistema é necessário a obtenção dos parâmetros K e τ sendo, para isso, necessário obter a resposta do sistema a uma entrada em degrau unitário. Considera-se que o degrau unitário corresponde a aplicar à entrada do sistema 12 V. Após se registar a resposta deste, recorreu-se ao método dos mínimos quadrados para se determinar os parâmetros K e τ. No teste experimental, foram registados 12,1 V na entrada e os valores obtidos na saída são os apresentados a azul no gráfico da Figura 85. A aproximação obtida é apresentada no mesmo gráfico (a vermelho), para a qual foi utilizado o método dos mínimos quadrados implementado no MATLAB.

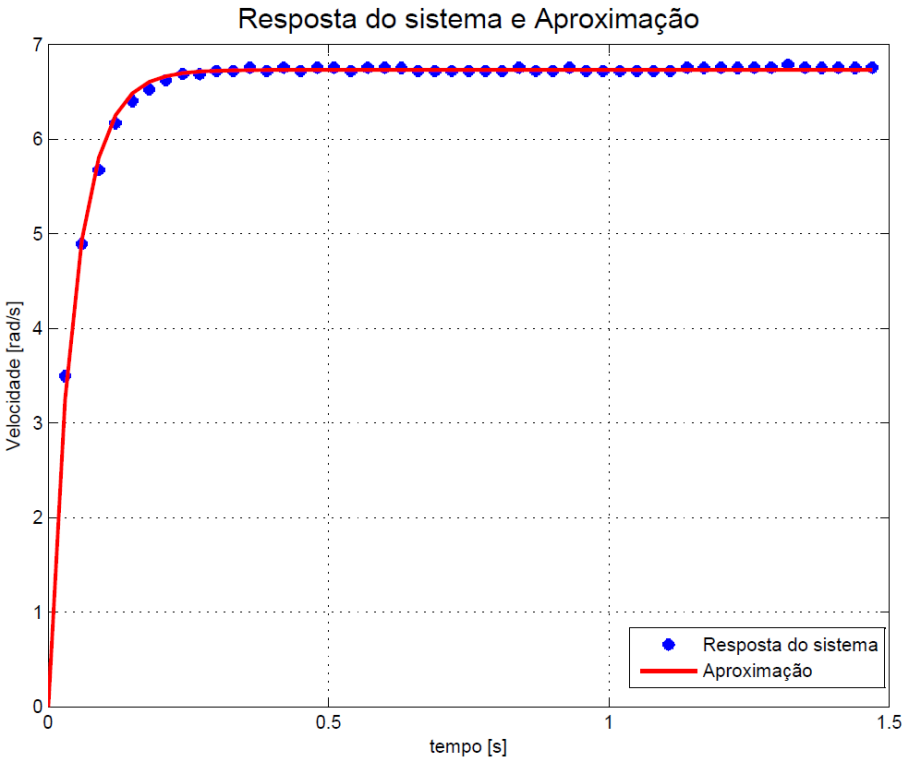


Figura 85 Resposta do sistema (azul) e aproximação obtida (vermelho).

Os valores calculados de K e τ são, respectivamente 6,72 e 0,0454, resultando por isso a seguinte função de transferência para o processo:

$$G(s) = \frac{\omega(s)}{U(s)} = \frac{6,72}{0,0454s + 1} \tag{38}$$

Uma vez definido o processo pode-se começar o projecto do controlador PI. Foi escolhido este tipo de controlador porque o sistema iria apresentar um erro em regime permanente, dado que este não possui nenhum pólo na origem, como se pode constatar analisando a equação (38).

Para o projecto do controlador PI foi utilizado o MATLAB, tendo sido feitos os cálculos e a simulação do controlador nesta aplicação. Foi utilizado o método de projecto através do lugar de raízes no Plano *z* [16]. Este projecto é efectuado inteiramente no domínio digital. Para aplicar este método é necessário converter as especificações do plano *s* para o plano *z*, através da expressão $z = e^{sT}$.

Como requisitos foram definidos um *overshoot* inferior a 10%, um tempo de subida de 0,08 segundos e período de amostragem *T*=0,01 segundos. A partir das equações seguintes calculam-se os pólos desejados no plano *z*. Assim, para as especificações definidas é possível obter o valor do coeficiente de amortecimento (ζ) e da frequência natural (ω_n).

$$\begin{cases} \zeta = 0{,}6\left(1 - M_p\right) \\ w_n = \dfrac{1{,}8}{tr} \end{cases} \Rightarrow \begin{cases} \zeta = 0{,}054 \\ w_n = 22{,}5\ rad \cdot s^{-1} \end{cases} \tag{39}$$

Após a obtenção destes parâmetros, é possível determinar a localização dos pólos no plano *z*, através da resolução do seguinte sistema de equações:

$$z_{1,2} = re^{\pm j\theta}, \begin{cases} r = e^{-\sigma T} = e^{-\zeta w_n T} \\ \theta = w_d = w_n\, T\sqrt{1-\zeta^2} \end{cases} \tag{40}$$

Obteve-se um valor de r = 0,8856 e um θ = 0,18494. Daqui surgem os pólos no plano z:

$$z_{1,2} = 0{,}8698 \pm j0{,}1667 \cdot \tag{41}$$

Em seguida, é necessário discretizar a função de transferência do processo, *G*(*s*) antecedida de um conversor D/A (*Zero-order Hold*), resultando a função de transferência discreta, *G*(*z*), dada por:

$$G(z) = \frac{z-1}{z}\left\{\frac{G(s)}{s}\right\} \tag{42}$$

$$G(s) = 6{,}72 \cdot \frac{22{,}03}{s + 22{,}03} \tag{43}$$

em que *G*(*z*) resulta em:

$$G(z) = 6{,}72\,\frac{1 - e^{-22{,}03T}}{z - e^{-22{,}03T}} \tag{44}$$

Em seguida, considera-se a função de transferência em malha aberta do sistema, ou seja $D(z)G(z)$, e aplica-se a condição de fase e de módulo para se determinar os restantes parâmetros do projecto. A função de transferência discreta do controlador PI, $D(z)$, aplica-se:

$$D(z) = K_c \frac{z-\alpha}{z-1} \tag{45}$$

Após a realização destes passos, efectuados no MATLAB, obtêm-se os valores de α=0,2869 (pólo do controlador) e K_c=0,0472. O diagrama de blocos final para simular o sistema de controlo é apresentado na Figura 86.

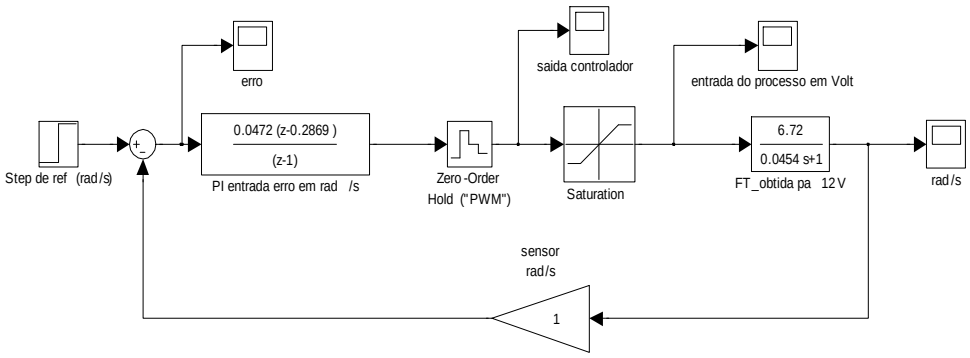


Figura 86 Diagrama de blocos (Simulink) do sistema em malha fechada.

A resposta simulada por este sistema para um sinal de referência de 6,72 rad/s é apresentada na Figura 87.

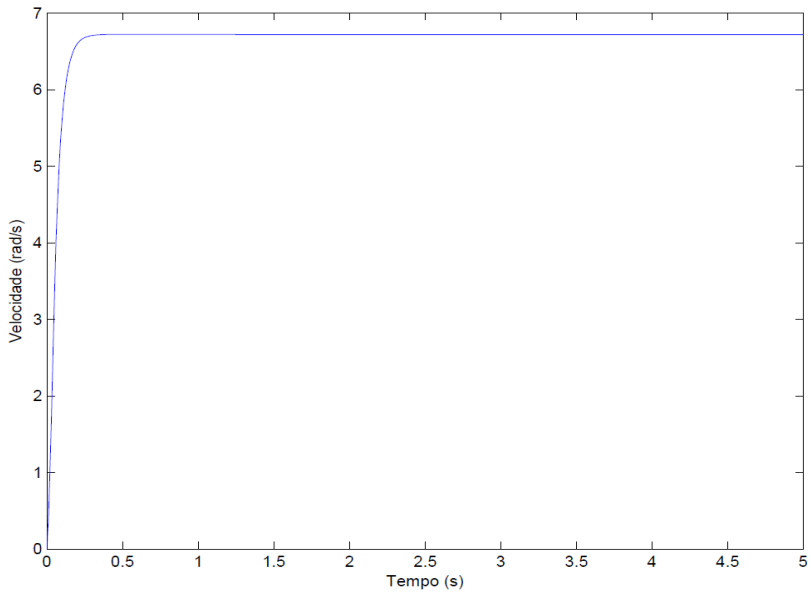


Figura 87 Resposta do sistema a um sinal de referência de 6,72 rad/s.

Para a referência de 1 rad/s a resposta do sistema apresenta um *overshoot* de 13,4 % (ligeiramente acima do especificado) e, tal como no caso anterior, não possui erro em regime permanente, que é uma das características pretendidas para este sistema. O tempo de subida ronda os 0,08 s, tal como era pretendido. A resposta para este caso pode ser vista na Figura 88.

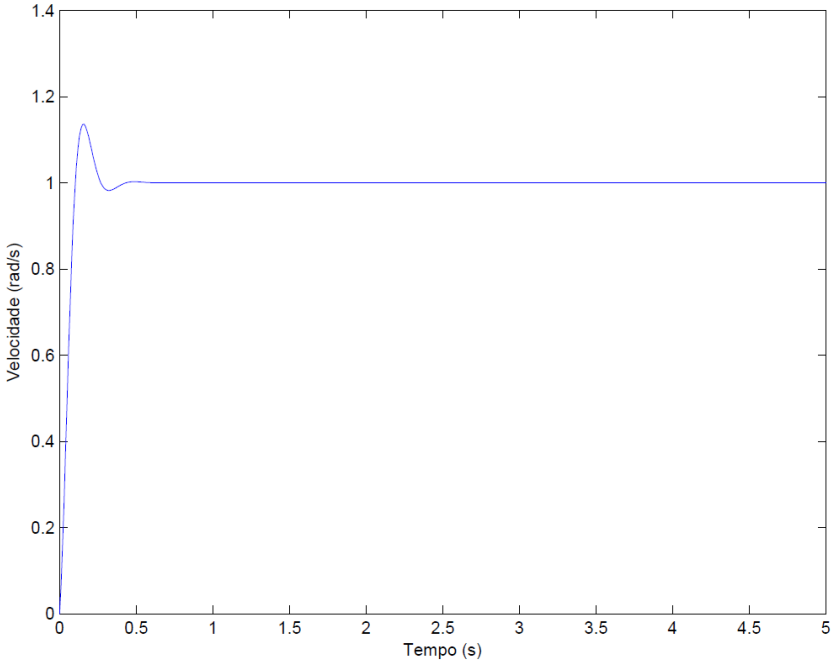


Figura 88 Resposta do sistema a um sinal de referência de 1rad/s.

Numa fase seguinte procedeu-se à obtenção das equações de controlo a implementar no microcontrolador, em que K=0,0472, α=0,2869 *e* $K.\alpha$=0,135, aproximadamente.

$$c(k) = K\,e(k) \underbrace{-K\alpha e(k-1) + c(k-1)}_{x_1} \tag{46}$$

O sinal de controlo será fornecido por um módulo PWM, e será limitado ao intervalo [0,1], tal como foi feito no modelo utilizado para a simulação. Apresenta-se, de seguida, o código com a implementação do controlador PI.

```
…
#define k 0.0472
#define alfa 0.2869
#define k_alfa 0.0135
float ref=1.0,erro=0,ctr=0,x1=0; //para teste
float vel=0;
…
void PID()
{
erro=ref-vel;
```

```
ctr=x1+k*erro;
if(ctr>=1.0)
        {
        ctr=1.0;
        }
if(ctr<=0.0)
        {
        ctr=0.0;
        }
dc=(int)(1023*ctr);
SetDCPWM1(dc);

x1=ctr-(k_alfa*erro);
}
```

Após a implementação da equação de controlo, visível na equação (46), foram efectuados alguns testes experimentais para verificar se o comportamento obtido na prática estava próximo do comportamento desejado. Os resultados obtidos (Figura 89) resultam da aplicação de várias velocidades de referência.

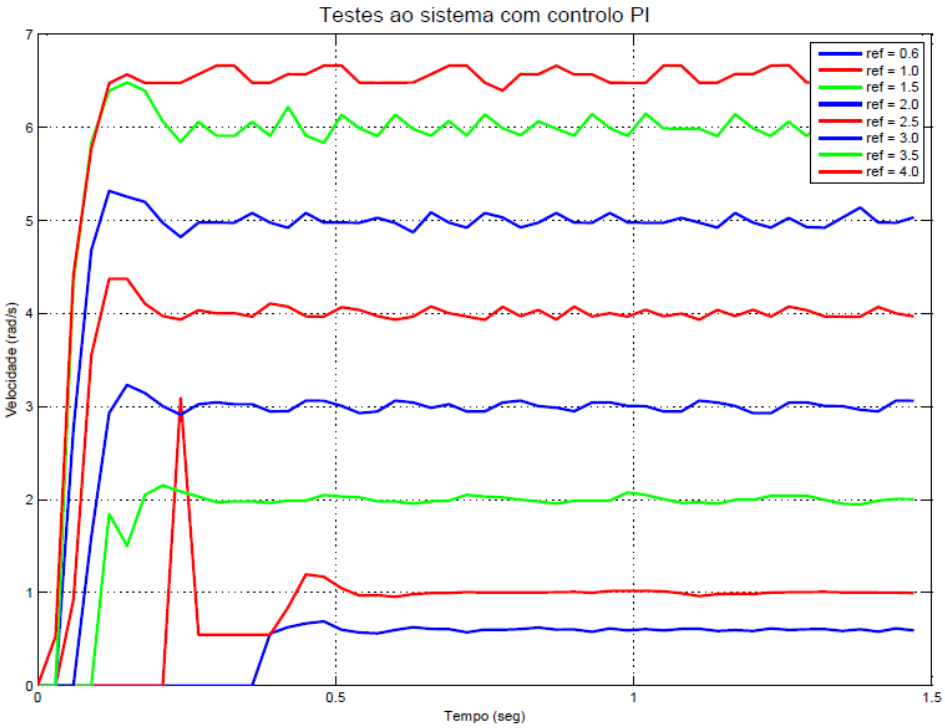


Figura 89 Resultados experimentais obtidos, sem sintonia fina do controlador PI e motor sem carga.

É importante salientar que estes resultados foram obtidos através de medições efectuadas no motor sem qualquer carga acoplada ao veio. Para valores de referência inferiores a 0,6 rad/s o motor não entra em funcionamento; por este facto não foram registados valores

inferiores a essa referência. Quando se adicionou carga ao sistema este sistema funcionou convenientemente.

Como foi mencionado na secção 4.1 (Figura 66), um dos microcontroladores PIC controla dois motores, recorrendo a dois módulos para gerar o PWM, designados de CCP e ECCP. Este nó do sistema é responsável, também, por calcular a velocidade de cada motor, por receber pela rede CAN a referência para cada motor e enviar as velocidades medidas e a direcção do motor para a rede. No entanto, nesta secção foca-se apenas a implementação relacionada com o módulo de controlo PI. O esquemático do circuito deste nó do sistema é o apresentado na Figura 90.

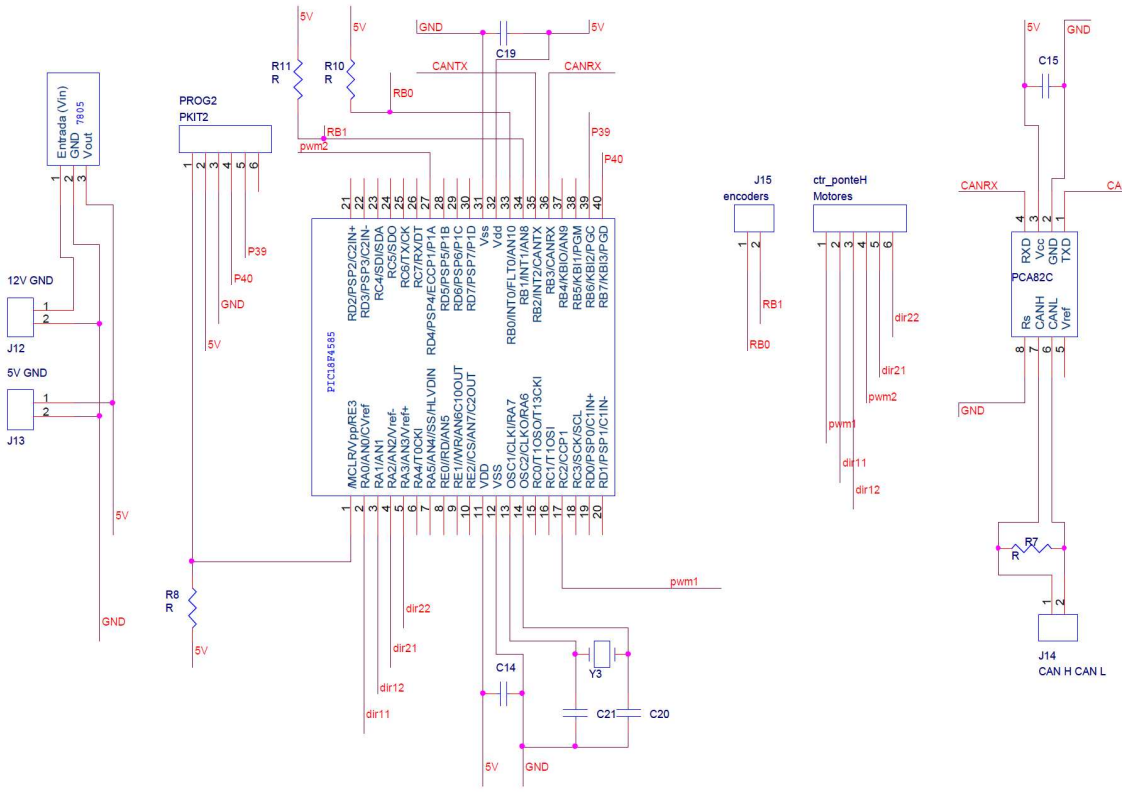


Figura 90 Esquemático do circuito que implementa o controlador de velocidade para os motores.

Como se pode ver na Figura 90, para além da PIC18F4585 existe também um *transceiver* CAN para permitir a interligação à rede CAN. É também visível a existência de dois conectores, um de dois contactos e outro com seis contactos. O primeiro refere-se às entradas dos sinais provenientes dos *encoders* de cada motor. O segundo possui os sinais de controlo para a ponte H, que é responsável pela entrega de energia aos motores. Este conector possui os contactos para os sinais que indicam a direcção (*dir11*, *dir12*, *dir21*,

dir22) e para os sinais de PWM que irão controlar a energia entregue a cada motor (*pwm1, pwm2*).

Para o esquemático apresentado na Figura 90, foi desenvolvido um esquema para a elaboração da placa de circuito impresso, que se encontra no Anexo C, secção C.1.

5.3. LEITURA DOS SONARES

Como se pode ver na Figura 66, referente à arquitectura de *hardware* do sistema, um dos microcontroladores é responsável por efectuar as leituras dos sensores e enviar os dados para a rede CAN.

Os sonares possuem o diagrama temporal apresentado na Figura 91, onde é visível o sinal de controlo a gerar (*trigger pulse*) e o sinal que indicará as distâncias medidas (*echo pulse*).

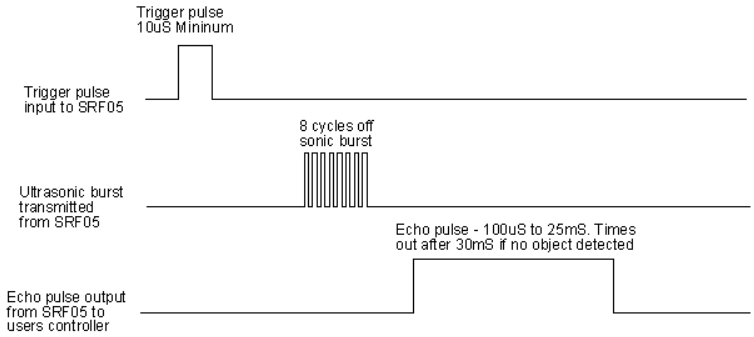


Figura 91 Sinais para interacção com o sonar SRF05.

Para o sonar efectuar uma medição é necessário fornecer um sinal de disparo com uma largura mínima de 10 μs. Depois de o sonar emitir a sequência de pulsos, deve-se esperar que a linha de eco vá ao nível lógico "1". O tempo que este sinal permanecer em "1" vai ser proporcional a uma distância; neste caso, a relação distância = tempo (μs) /58 indica directamente a distância em centímetros. Os passos para a aquisição dos valores dos sensores estão representados na Figura 92.

Como existem quatro sensores, decidiu-se dispará-los dois a dois. Os sensores disparados simultaneamente encontram-se em locais opostos para não existirem interferências entre eles. Deste modo, pode-se diminuir o tempo de aquisição dos valores em comparação à situação em que se disparavam os sensores individualmente. Nessa situação, cada

aquisição poderia ultrapassar a duração de 30 ms; com esta solução o tempo é reduzido sensivelmente para metade.

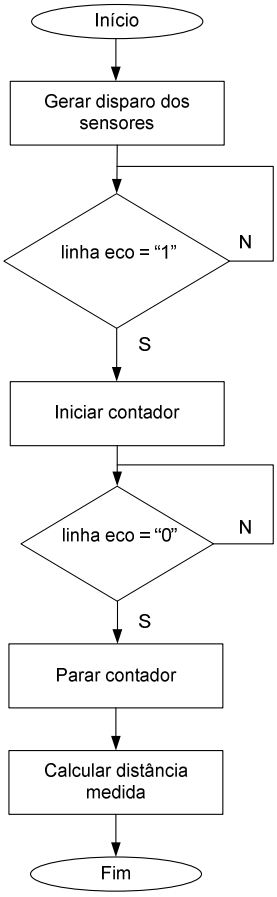


Figura 92 Fluxograma representativo da aquisição dos dados de um sonar.

A implementação da aquisição dos valores das leituras dos quatro sensores passa por uma reprodução dos passos apresentados no fluxograma da Figura 92.

O esquemático deste módulo é apresentado na Figura 93, onde se identifica o microcontrolador PIC 18F4585 e o *transceiver* CAN (à direita do mesmo); entre ambos estão os conectores referentes às ligações dos sinais que permitem a interacção (disparo e leitura) com os sonares. Nesses conectores estão os sinais que farão o disparo dos quatro sonares (*trigger12* fará o disparo dos sonares 1 e 2 enquanto o *trigger34* fará o disparo dos sonares 3 e 4). Os restantes sinais permitem a obtenção da distância entre o AGV e os obstáculos ou as paredes.

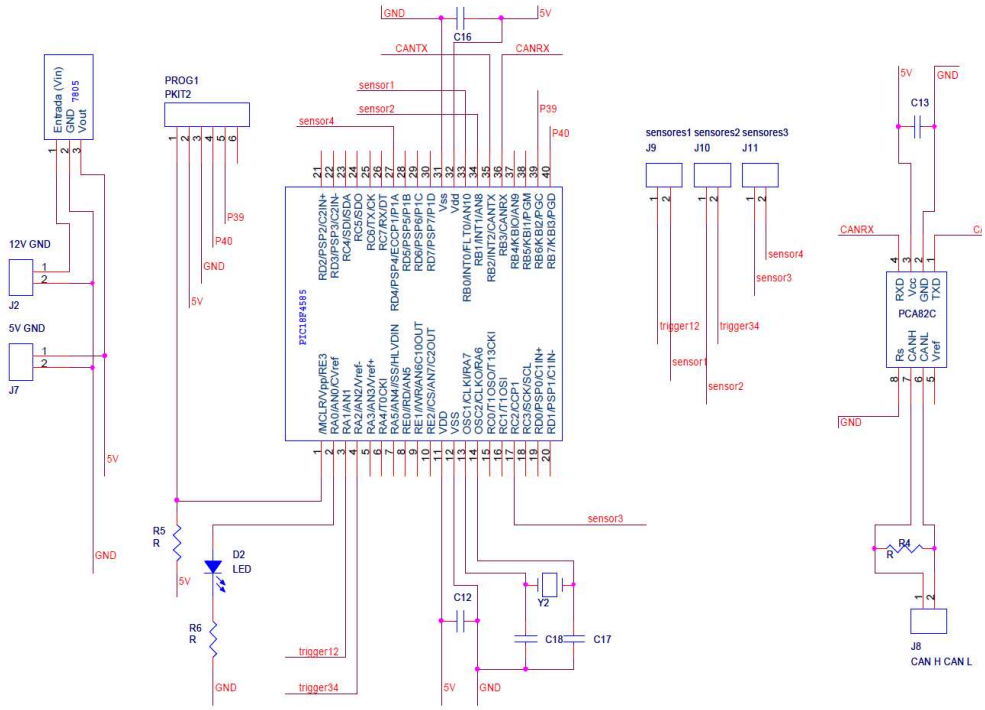


Figura 93 Esquemático do circuito que implementa a leitura dos sonares.

Para o esquemático apresentado na Figura 93, foi desenvolvido um esquema para a elaboração da placa de circuito impresso, que se encontra no Anexo C, secção C.2.

5.4. REDE CAN

Neste trabalho, implementou-se a rede CAN com o intuito de distribuir tarefas pelos vários nós, mas também pela flexibilidade que esta rede traz ao sistema, pois facilita a introdução de futuros módulos no sistema. A rede implementada tem como função promover a interacção entre os vários nós, sendo responsável pela difusão das mensagens.

Uma vez que não existe um número elevado de identificadores, ou seja de mensagens, não será uma grande preocupação, até este ponto, a taxa de comunicação. De qualquer modo optou-se pela taxa de 444 kbps.

As configurações que permitem esta velocidade são apresentadas na Figura 94, sendo apresentados os parâmetros mais importantes para a configuração da taxa de comunicação.

```
//Baud rate   444kBITS/S
Tq -> 1 - SINCRO 1- PROP SEG 4- PHASE SEG1 4- PHASE
SEG 2
   //***********************************************
   //BRGCON1
```

```
//-----------------------------------------------
//7|6|5|4|3|2|1|0|
//-----------------------------------------------
//x x-> SJW
//    x x x x x x->BRP
//***********************************************

BRGCON1=0b11000011;

//***********************************************
//BRGCON2
//-----------------------------------------------
//7|6|5|4|3|2|1|0|
//-----------------------------------------------
//x -> 0 INDICA PHASE SEG2 É NO MAX == PHASE SEG 1
//  x -> 1 -> amostra três vezes
//    x x x ->Phase seg 1 até 8Tq
//          x x x ->PROPAGATION seg até 8Tq
//***********************************************

BRGCON2=0b01011000;
BRGCON3=0b10000101;
CIOCON=0b00100000;
```

Figura 94 Extracto do código de configuração da comunicação CAN.

Cada *bit time* é constituído por um *Tq* para sincronização, um para o segmento de propagação, quatro para o segmento *Phase1* e quatro para o segmento *Phase2*. O *Sincronization Jump Width* pode ter um valor máximo de dois *Tq*, para permitir uma resincronização.

Cada nó envia e recebe apenas as variáveis que lhe interessam, podendo-se utilizar os 8 bytes disponíveis na mensagem CAN para o envio das variáveis que pretendermos, desde que se tenha em atenção o formato de cada uma delas.

De seguida são referidos os formatos das mensagens que cada nó envia, ficando assim definido o formato das mesmas como informação para futuras evoluções do sistema.

O nó dos motores envia mensagens com o identificador número 3 (ID3), em que os 8 bytes da mensagem CAN estão divididos da forma apresentada na Figura 95. Esta mensagem está prevista mas o seu envio para o controlador difuso não acontece, pela simples razão que o controlador difuso não necessita desta informação na sua entrada; por este facto, o envio desta mensagem para a rede CAN foi suspenso.

ID: 3

n bytes	Tamanho: 1byte
0	Byte menos significativo t0
1	Byte mais significativo t0
2	Byte menos significativo t1
3	Byte mais significativo t1
4	Direcção Motor 0
5	Direcção Motor 1
6	NA
7	NA

Figura 95 Mensagem enviada pelo nó de controlo dos motores.

O nó dos sensores envia mensagens com o identificador número 5 (ID5), em que os 8 bytes da mensagem CAN estão divididos da forma representada na Figura 96.

ID: 5

n bytes	Tamanho: 1byte
0	Byte menos significativo sensor1
1	Byte mais significativo sensor 1
2	Byte menos significativo sensor 2
3	Byte mais significativo sensor 2
4	Byte menos significativo sensor 3
5	Byte mais significativo sensor 3
6	Byte menos significativo sensor 4
7	Byte mais significativo sensor 4

Figura 96 Mensagem enviada pelo nó que efectua a leitura dos sonares.

O nó do controlador difuso envia mensagens com o identificador número 2 (ID2), em que os 8 bytes da mensagem CAN estão divididos da forma representada na Figura 97.

O controlador difuso irá receber as mensagens provenientes dos outros módulos, utilizando esses dados para calcular os valores de controlo para o sistema. De seguida, envia uma mensagem para o módulo de controlo dos motores com os valores de referência das velocidades para cada motor.

ID: 2

n bytes	Tamanho: 1byte
0	Referência Motor 0 Byte menos significativo
1	Referência Motor 0 2 Byte
2	Referência Motor 0 3 Byte
3	Referência Motor 0 Byte mais significativo
4	Referência Motor 1 Byte menos significativo
5	Referência Motor 1 2 Byte
6	Referência Motor 1 3 Byte
7	Referência Motor 1 Byte mais significativo

Figura 97 Mensagem enviada pelo nó do controlador difuso para controlo dos motores.

Antes de se abordar o projecto e a implementação do controlador difuso, faz sentido clarificar a interacção entre os vários módulos do sistema. Esta abordagem é feita na secção seguinte.

5.5. SISTEMA GLOBAL

O sistema é composto por três módulos distintos responsáveis por efectuar o controlo de velocidade dos motores, aquisição de dados provenientes dos sensores e efectuar o controlo por lógica difusa do sistema onde estão implementados os vários comportamentos (os quais são apresentados na secção 5.6).

É apresentado na Figura 98 um esquema representativo da distribuição modular e da interacção entre os vários módulos do sistema. Analisando o esquema, notam-se as várias dependências entre os módulos; por exemplo, o módulo de controlo de velocidade do sistema necessita das saídas do módulo que implementa o controlador difuso, ou seja, as velocidades de referência dos motores direito e esquerdo. Por sua vez, o controlador difuso necessita na sua entrada dos valores medidos pelos sensores, sendo estes obtidos do módulo de aquisição de dados dos sensores e enviados para o módulo do controlador difuso. O envio das variáveis, acontece por intermédio de uma rede CAN que liga os vários módulos.

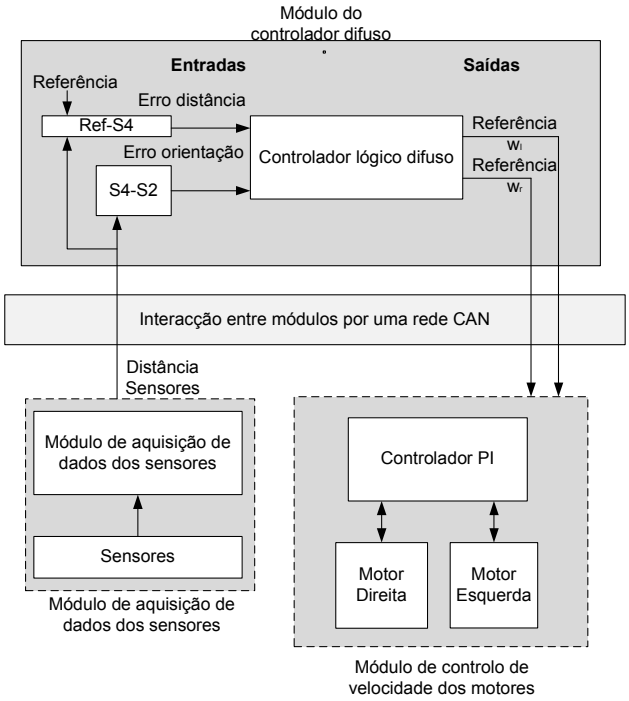


Figura 98 Esquema representativo da interacção entre os vários módulos do sistema.

A decisão de desenvolver um sistema modular prende-se com o facto de permitir uma maior flexibilidade do sistema, permitindo implementar futuras evoluções e melhorias com relativa facilidade. A divisão das tarefas por diferentes módulos permitiu também simplificar a implementação e desenvolvimento de código para o sistema, visto que cada módulo executa uma tarefa específica.

O módulo responsável pela aquisição de dados provenientes dos sensores, irá enviar os dados referentes a cada sensor assim que terminar a leitura dos quatro sensores. Em seguida, coloca-os numa mensagem com o formato previsto no protocolo CAN, e envia-a para a rede. O módulo do controlador difuso, ao verificar que existe uma mensagem com o ID 5, sabe que esta é proveniente do módulo dos sensores e que a informação contida na mensagem é necessária e por isso recebe-a.

Após efectuar todas as tarefas previstas no controlador difuso, são obtidos os valores de velocidades angulares para cada motor. Estas velocidades são os valores de referência utilizados pelos controladores PI; deste modo é necessário pegar nesta informação e colocá-la numa mensagem CAN com o ID2, e enviá-la para a rede. O módulo do controlo de velocidade reconhece o identificador e recebe a mensagem e, em seguida, actualiza as variáveis de modo a serem utilizadas no próximo ciclo de controlo.

Após a descrição do funcionamento global do sistema é abordado com maior detalhe, na secção seguinte, o módulo do controlador difuso.

5.6. CONTROLADOR DIFUSO

Em primeiro lugar, faz sentido contextualizar o problema de modo a compreender melhor as decisões que foram tomadas durante o projecto e implementação do controlador difuso.

Na Figura 99 é apresentado um esquema que permite mostrar de uma forma mais clara o que se pretende. O controlador deve ser capaz de guiar o robô ao longo de uma parede (à sua direita) mantendo-se a uma distância pré-definida. Para isso, o robô tem de possuir dois sensores laterais que fornecem informações relativas à distância a que se encontra a parede e sobre a orientação do robô segundo a mesma. Este deve também ser capaz de evitar obstáculos que encontre no seu caminho. Assim deste modo foram introduzidos no sistema dois sensores frontais para que detecte a presença de algum obstáculo.

Numa primeira fase foi apenas abordado o comportamento designado por "seguir parede", onde as suas variáveis de entrada são as seguintes: "Erro na distância" (em relação à parede) e "Orientação". Como é apresentado na Figura 99, o erro na distância é obtido pela diferença entre uma distância de referência pré-definida e a distância d_1, enquanto a orientação é determinada pela diferença entre as duas distâncias, d_1 e d_2, que são obtidas pelos sensores S_4 e S_2, respectivamente.

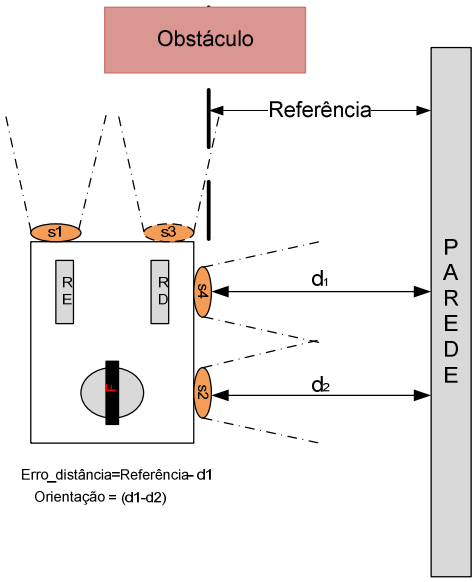


Figura 99 Contexto do problema de controlo.

Em seguida, será abordado o comportamento evitar obstáculo, que utiliza a informação proveniente dos sensores frontais S_1 e S_3. Esta informação permite decidir que acção se adequa mais consoante a posição do obstáculo face ao robô, e permite definir um novo comportamento designado de "emergência" em que se a distância do obstáculo em relação ao robô for reduzida este suspende a execução deixando de se mover.

Será ainda abordado o método escolhido para seleccionar o comportamento mais adequado conforme a situação que é encontrada pelo robô durante a execução.

Na Figura 100 é apresentado um esquema que pretende clarificar, de uma maneira simples, as variáveis e os comportamentos para este sistema.

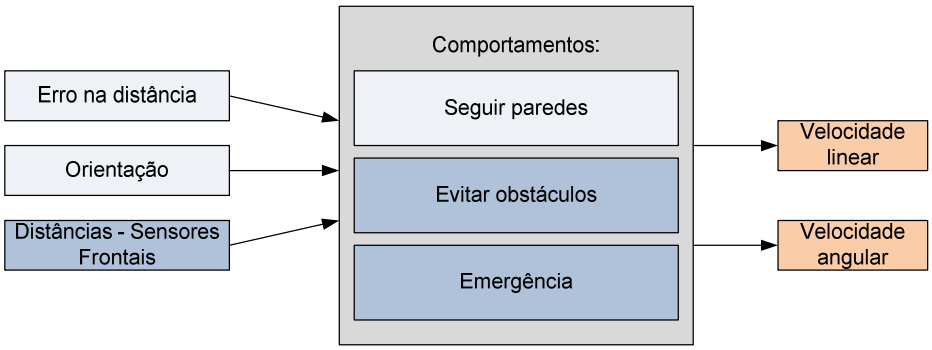


Figura 100 Comportamentos e variáveis utilizadas no controlador difuso.

O projecto do controlador difuso é realizado com o auxílio do MATLAB, e será apresentado nas sub-secções seguintes, assim como as variáveis de entrada e de saída, os universos de discursos, as funções de pertença e as regras geradas para cada comportamento.

5.6.1. COMPORTAMENTO "SEGUIR PAREDE" – 9 REGRAS

As variáveis de entrada do controlador difuso para implementar o comportamento "seguir parede" são o erro na distância do robô em relação a uma distância de referência e a orientação do mesmo face à parede.

5.6.1.1. PROJECTO

Estas variáveis linguísticas "Erro na distância" e "Orientação" são apresentadas nas Figuras 101 e 102. A variável linguística "Erro na distância" está definida no universo de discurso compreendido no intervalo de [-40; 40] centímetros. O valor de entrada é

encontrado pela diferença entre uma referência pré-definida e o valor medido pelo sensor 4, como é apresentado na equação (47).

$$ErroDistância = Referência - Valor_{sensor4} \quad (47)$$

Esta variável linguística é definida por três funções de pertença que representam três termos linguísticos, são eles, “Ref.àDireita”, “Zero” e “Ref.àEsquerda”. A função de pertença “Ref.àDireita” define os valores do universo de discurso em que a referência pode ser interpretada como estando à direita do robô, a “Zero” como estando na referência e, por último, “Ref.àEsquerda” como estando à esquerda do robô.

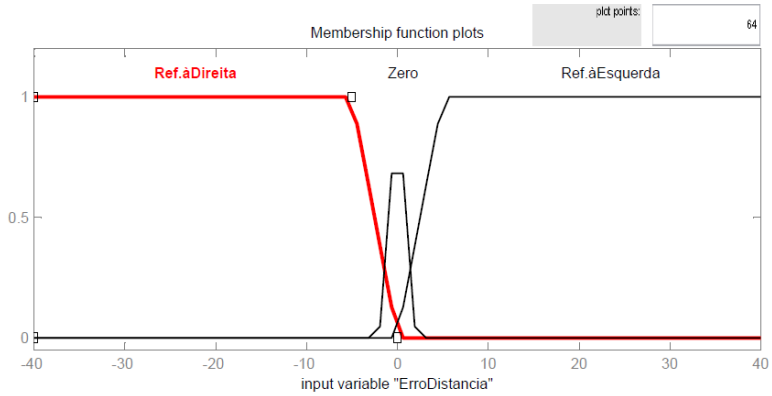


Figura 101 Variável linguística “ErroDistância”.

Convém referir que estas funções foram amostradas para 64 elementos, tal como aconteceu na implementação no microcontrolador. Por este motivo é que algumas formas se encontram “cortadas”.

A variável linguística “Orientação” está definida para o mesmo universo de discurso da anterior, ou seja, no intervalo de [-40; 40] centímetros. Esta variável representa a variação do ângulo do robô com a parede, e os valores de entradas são obtidos pela diferença das distâncias medida pelos sensores S_4 e S_2, como se pode ver na equação (48).

$$ErroOrientação = Valor_{sensor4} - Valor_{sensor2} \quad (48)$$

Se o valor medido pelo sensor 4 for maior que o medido pelo sensor 2, o valor resultante da equação será positivo, significa por isso que o robô está orientado para a esquerda. Se acontecer o contrário, ou seja, o valor do sensor 4 for menor que o valor do sensor 2, o resultado da equação será negativo e o robô estará orientado para a direita. Se for zero, o robô está paralelo à parede.

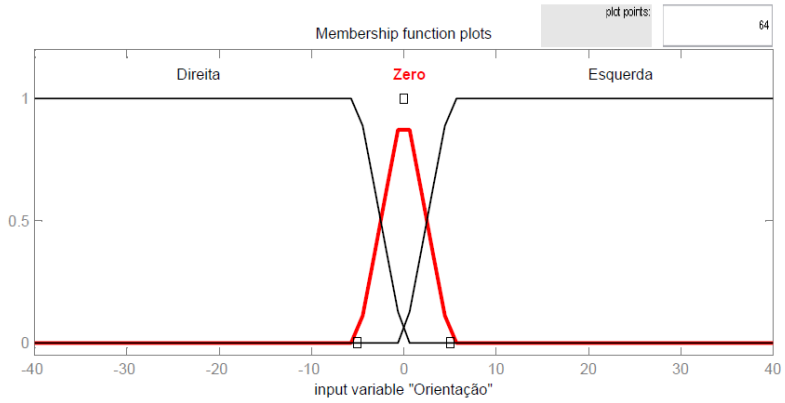


Figura 102 Variável linguística "Orientação".

A variável linguística "Orientação", é definida por três funções de pertença designadas por "Direita", "Zero" e "Esquerda".

Para uma maior simplicidade inicial, apenas foram consideradas três funções de pertença para cada variável de entrada de modo a não aumentar o número de regras.

As variáveis linguísticas para as acções de controlo são a "Velocidade linear" e a "Velocidade angular" estando apresentadas nas Figuras 103 e 104.

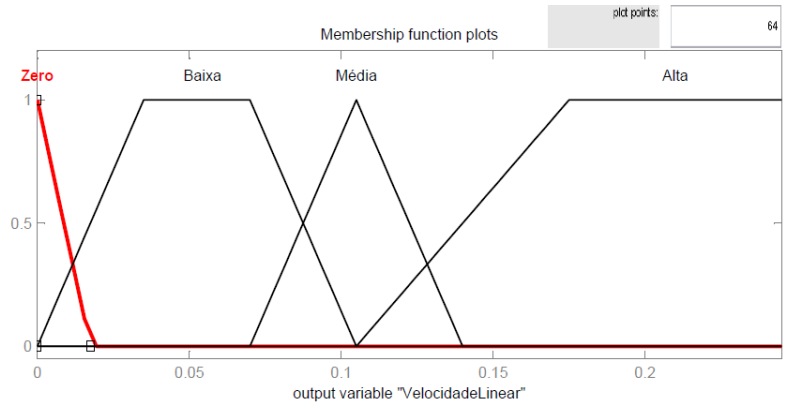


Figura 103 Variável linguística "VelocidadeLinear".

O universo de discurso desta variável está definido no intervalo [0; 0,245] $m.s^{-1}$. A variável linguística é caracterizada por quatro funções de pertença, são elas, "Zero", "Baixa", "Média" e "Alta". Estes conceitos serão fundamentais pois permitem a construção do conjunto difuso de saída para a "VelocidadeLinear" e, após a desfuzificação desses conjuntos, resulta um valor numérico passível de ser aplicado. O mesmo acontece com a variável linguística "Velocidade Angular".

Esta variável é definida por três funções de pertença, designadas por “Direita”, “Zero” e “Esquerda”.

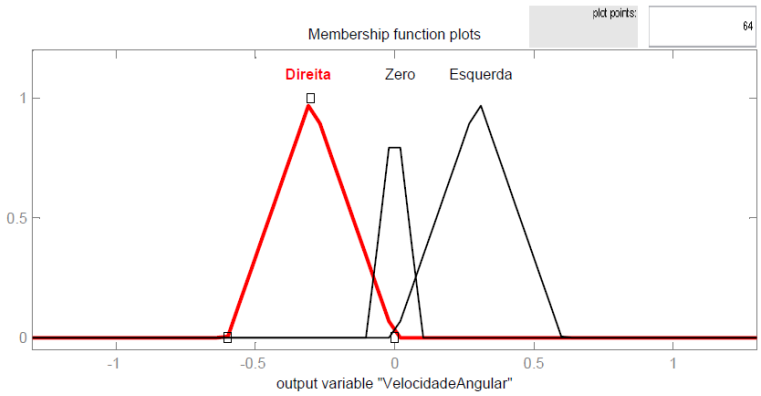


Figura 104 Variável linguística “VelocidadeAngular”.

As variáveis linguísticas “Velocidade Linear” e “Velocidade Angular”, através das suas funções de pertença, estão presentes nas regras do controlador, e após a análise da aplicabilidade das regras, do processo de implicação e agregação dão origem a dois conjuntos difusos de saída que representam o peso de cada acção verificada em cada uma das regras. Esses dois conjuntos, após a desfuzificação, dão origem a dois valores numéricos, um para a velocidade linear e outro para a velocidade angular, podendo agora estes valores ser utilizados para o controlo do sistema. Mais à frente, após a apresentação tabular das regras, será descrito todo o processo desde a fuzificação até à desfuzificação.

Nas Tabelas 8 e 9, é apresentada a representação tabular das regras para cada uma das variáveis linguísticas que darão origem aos conjuntos difusos de saída “Velocidade Linear” e “Velocidade Angular”.

Tabela 8 Representação tabular das regras para a “VelocidadeLinear”.

Velocidade Linear	Erro na Distância			
Orientação		Ref.àEsquerda	Zero	Ref.àDireita
	Direita	Baixa	Baixa	Baixa
	Zero	Média	Média	Média
	Esquerda	Baixa	Baixa	Baixa

Tabela 9 Representação tabular das regras para a "VelocidadeAngular".

Velocidade Angular	Erro na Distância			
Orientação		Ref.àEsquerda	Zero	Ref.àDireita
	Direita	Esquerda	Zero	Esquerda
	Zero	Zero	Zero	Zero
	Esquerda	Direita	Zero	Direita

A aplicação das regras, através da utilização do MATLAB, deu origem a uma ilustração do sistema apresentada na Figura 105, onde cada linha representa uma regra na base de regras a ser implementada pelo controlador difuso.

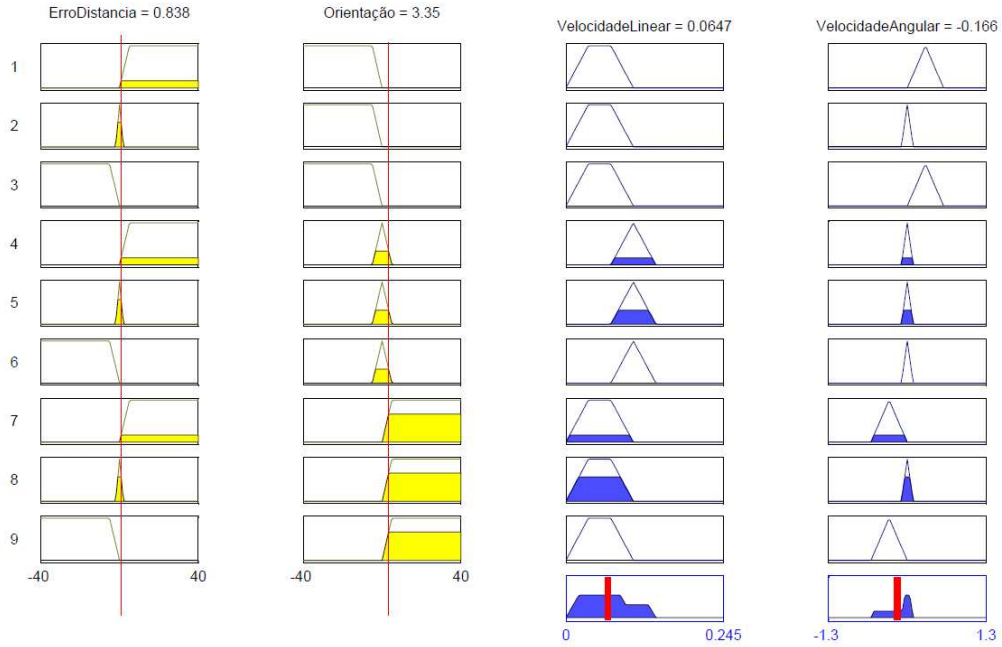


Figura 105 Regras implementadas no controlador.

Uma vez que cada variável linguística de entrada é caracterizada por três funções de pertença, pode-se ter no máximo 9 regras. As variáveis linguísticas de entrada são representadas na imagem a amarelo, e as variáveis linguísticas de saída são apresentadas a azul. As barras vermelhas verticais, do lado esquerdo, representam os valores numéricos de entrada do controlador; estes são comparados com os termos linguísticos onde se verifica o enquadramento dos mesmos (a este processo chama-se fuzificação). Em seguida, é verificada através dos antecedentes das regras a aplicabilidade de cada uma consoante, neste caso, as duas variáveis linguísticas de entrada.

Após a verificação da aplicabilidade de cada regra, é feita a implicação, que consiste na utilização do menor valor do grau de verdade verificado nos antecedentes de uma regra, isto para o operador lógico difuso *and*, de modo a limitar a esse valor os graus de verdade das variáveis linguísticas de saída.

Após este processo, estão definidas as acções resultantes de cada uma das regras, sendo necessário efectuar a agregação, que neste trabalho é realizada com o operador lógico difuso *or*, sendo por isso agregadas todas as acções activas, sobrepondo todos os conjuntos difusos de cada acção. O conjunto difuso final representa a envolvente da sobreposição de todas as acções, ou seja, deve-se considerar para a construção do conjunto difuso final o valor máximo verificado em cada uma das acções para cada valor do universo de discurso.

Uma vez definidos os conjuntos difusos de saída, é necessário aplicar um método de desfuzificação de modo a se obter um valor numérico em vez de um conjunto difuso. Neste projecto foi utilizado o método "centróide da área", que é descrito pela equação (49).

$$u^* = \frac{\sum_{i=1}^{n} u_i * \mu_c(u_i)}{\sum_{i=1}^{n} \mu_c(u_i)} \tag{49}$$

em que o valor final u, será igual ao somatório do produto do valor do elemento do universo de discurso (u_i) pelo seu grau de verdade, isto para todo o universo de discurso, divido pelo somatório de todos os graus de verdade verificados no universo para todos os elementos (u_i) do mesmo. E n representa o número total de pontos discretos do universo de discurso considerados.

As superfícies de controlo do controlador difuso projectado para o comportamento "Seguir Parede" são apresentadas na Figura 106.

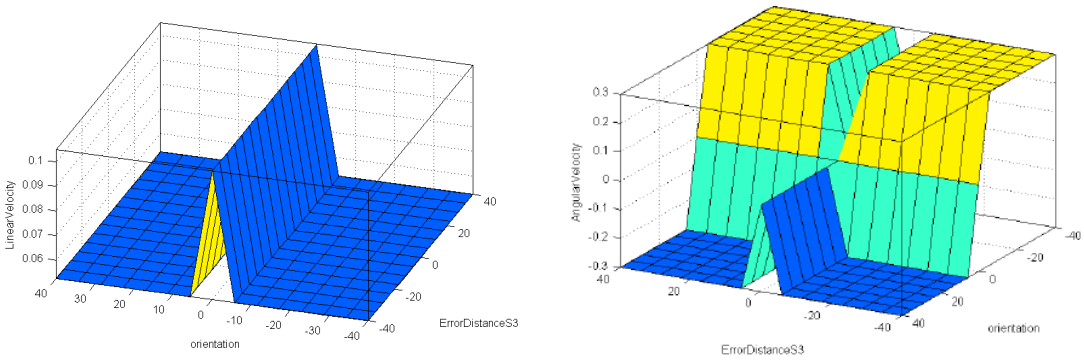


Figura 106 **Superfície de controlo da Velocidade Linear (à direita) e da Velocidade Angular (à esquerda).**

Após a desfuzificação dos conjuntos difusos de saída têm-se dois valores numéricos, um para a velocidade linear e outro para a velocidade angular, mas neste trabalho actua-se no sistema através das velocidades angulares de cada motor. No entanto, através das velocidades lineares e angulares podem-se obter as velocidades angulares a aplicar a cada motor.

Neste trabalho, como já foi referido na sub-secção 3.1, após a desfuzificação obtêm-se dois valores numéricos, um para cada variável de saída, ou seja, um para a velocidade linear e outro para a velocidade angular do sistema. Estes valores são utilizados para determinar as velocidades que se devem aplicar a cada motor, que são obtidos da forma apresentada de seguida.

Como,

$$w_{sistema} = \frac{V_{right} - V_{left}}{L} \tag{50}$$

a diferença de velocidades entre os motores será:

$$\Delta v = w_{sistema}\, L \tag{51}$$

em que L representa a distância entre os eixos das rodas.

Uma vez que a velocidade linear pretendida já é conhecida, pois resulta da saída do controlador difuso “VelocidadeLinear”, considera-se que as velocidades a aplicar a cada motor serão:

$$v_{right} = v_{linearsistema} - \frac{\Delta v}{2} \quad (52)$$

$$v_{left} = v_{linearsistema} + \frac{\Delta v}{2} \quad (53)$$

Após a obtenção das velocidades lineares a aplicar a cada motor, apenas é necessário dividir o valor obtido pelo raio da roda, de modo a obter-se uma velocidade angular (em $rad.s^{-1}$) de referência para os motores.

Em seguida as referências de controlo são enviadas para o módulo que efectua o controlo de velocidade de motor, o qual as usa como variáveis de entrada do sistema, ou referências de controlo.

Uma das dificuldades encontradas durante o projecto, reside na maneira como se deve definir as funções de pertença. Uma das hipóteses é após a implementação analisar-se os resultados obtidos e proceder a ajustes nas funções de pertença. A outra hipótese refere-se à utilização de redes neuronais que efectuam os ajustes durante a execução. Neste trabalho foi considerada a primeira hipótese, onde serão analisados os resultados finais e ajustadas as funções de pertença de cada variável linguística.

A definição das regras requer por parte do projectista a capacidade de, a partir das funcionalidades desejadas para o sistema, construir um conjunto de regras que relacionam uma ou mais entradas e geram uma ou mais saídas para cada combinação de variáveis na entrada

5.6.1.2. IMPLEMENTAÇÃO DO COMPORTAMENTO

A implementação do controlador difuso foi efectuada num microcontrolador PIC 18F4585 e o fluxograma da implementação é apresentado na Figura 107. Este diagrama representa apenas a estrutura geral do programa que implementa o controlador difuso, não contendo ainda a estratégia adoptada para a arbitragem de comportamentos.

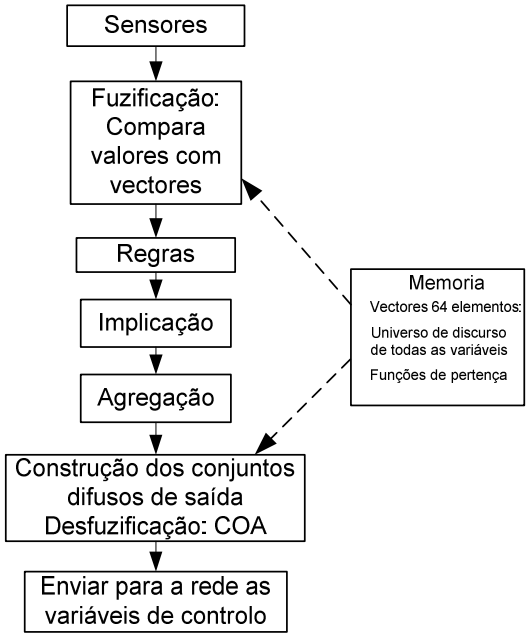


Figura 107 Fluxograma do programa implementado.

Os vectores em memória têm um comprimento de 64 elementos e possuem a definição de todos os universos de discurso e de todas as funções de pertença, permitindo a fuzificação das variáveis de entrada, pois é feito um enquadramento das mesmas no universo de discurso. Em seguida obtêm-se os graus de verdade verificados em cada uma das funções de pertença para esse valor do universo de discurso.

As regras de controlo são implementadas em código por funções condicionais IF, onde se verificam quais as regras que se aplicam à situação actual. Após esta verificação procede-se à implicação com o operador lógico difuso *and*, que não é nada mais do que limitar o valor do consequente da regra ao menor valor do grau de verdade verificado nos antecedentes das mesmas. Em seguida, consideram-se todos os consequentes de cada regra activa e é feita a agregação dos mesmos utilizando o operador lógico difuso *or*, que considera apenas os máximos dos consequentes verificados entre todos, para um determinado valor do universo de discurso. Daqui surgem, neste caso, dois conjuntos difusos de saída, um para cada variável de saída do controlador.

O último passo é efectuar a desfuzificação dos conjuntos difusos de saída de modo a se obter apenas um valor numérico por conjunto difuso, passível de ser aplicado ao sistema de modo a controlá-lo.

Como já foi mencionado nesta secção, as variáveis de controlo do sistema são as velocidades angulares dos motores; por isso deve-se efectuar o raciocínio apresentado nas equações (50) a (53), de modo a se obter a velocidade angular a aplicar a cada motor. Em seguida, envia-se essa informação para o módulo de controlo de velocidade.

5.6.1.3. CONCLUSÕES E RESULTADOS EXPERIMENTAIS

Para se validar o controlador difuso, adquiriam-se os dados provenientes dos dois sensores laterais S_4 e S_2 pois os dados permitem obter informações sobre a distância do robô à parede e sobre o ângulo do robô com a parede. A aquisição dos dados foi efectuada durante 35 segundos, ao fim dos quais o robô envia os dados recolhidos para a porta série. Os dados são recolhidos a cada 1,2 segundos. Convém também referir que a parede utilizada neste teste era uma parede lisa sem rodapé.

Na Figura 108 são apresentados os resultados experimentais obtidos durante aproximadamente 35 segundos, a partir do instante inicial. Neste gráfico são apresentadas as medições do sensor 4 e da variação do ângulo do robô com a parede.

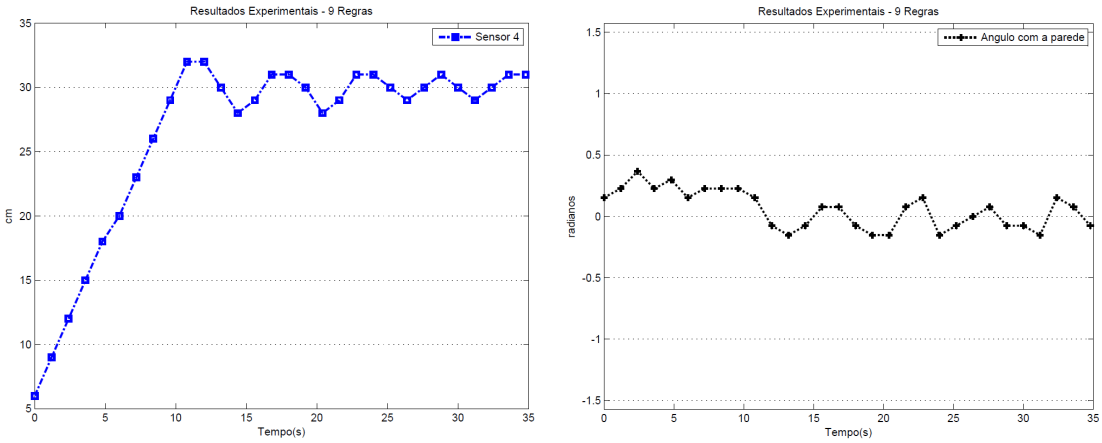


Figura 108 Resultados experimentais obtidos para o controlador de 9 regras.

Com a implementação deste comportamento verificou-se que o robô era capaz de seguir paredes mas verifica-se que a partir dos 10 segundos existem oscilações na sua trajectória em torno da referência, que para este trabalho é de 30 centímetros. No entanto, considera-se satisfatório o desempenho verificado.

5.6.2. COMPORTAMENTO "SEGUIR PAREDE" – 25 REGRAS

Foi também desenvolvido um controlador difuso onde eram consideradas mais funções de pertença para cada uma das variáveis de entrada, do qual resultaram as seguintes variáveis linguísticas, visíveis nas Figuras 109 a 112. Este controlador surge do anterior, embora com algumas alterações nas suas funções de pertença.

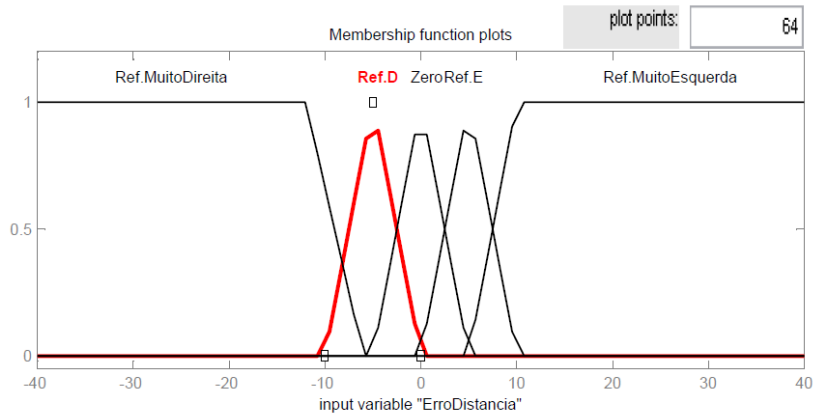


Figura 109 Variável linguística "ErroDistância".

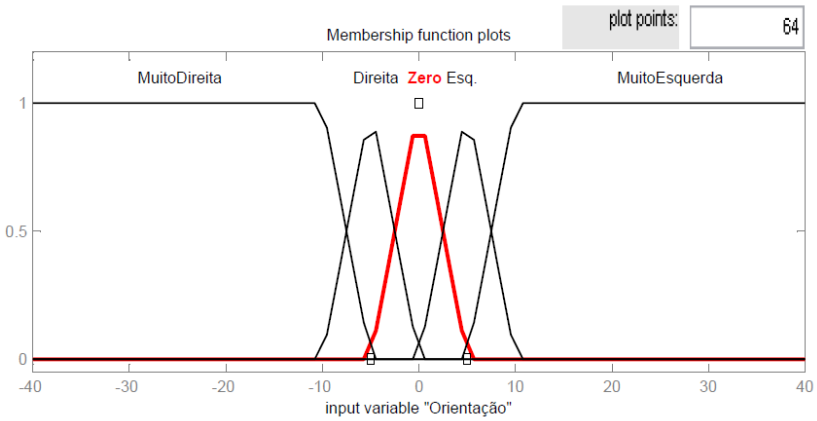


Figura 110 Variável linguística "Orientação".

Nas variáveis linguísticas de entrada apresentadas, foram adicionadas duas funções de pertença em cada uma delas, o que resulta num aumento do número de regras de controlo para 25 regras. Na variável linguística "ErroDistância" surgiram as funções de pertença "Ref.MuitoDireita" e "Ref.MuitoEsquerda"; na variável linguística "Orientação" surgiram as funções de pertença "MuitoDireita" e "MuitoEsquerda".

Nas variáveis de saída apenas a variável linguística “Velocidade Angular” teve um aumento no número das funções de pertença. Com este aumento surgiram as funções de pertença designadas por “MuitoDireita” e “MuitoEsquerda”, sendo assim possível obter maiores rotações do sistema, aumentando a capacidade de reagir a situações extremas.

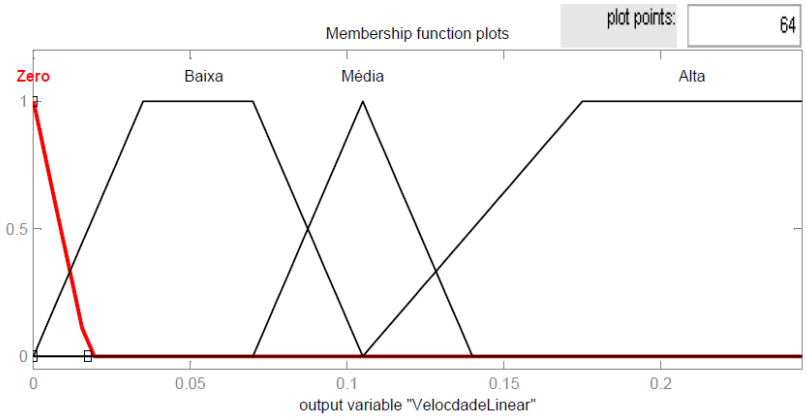


Figura 111 Variável linguística “VelocidadeLinear”.

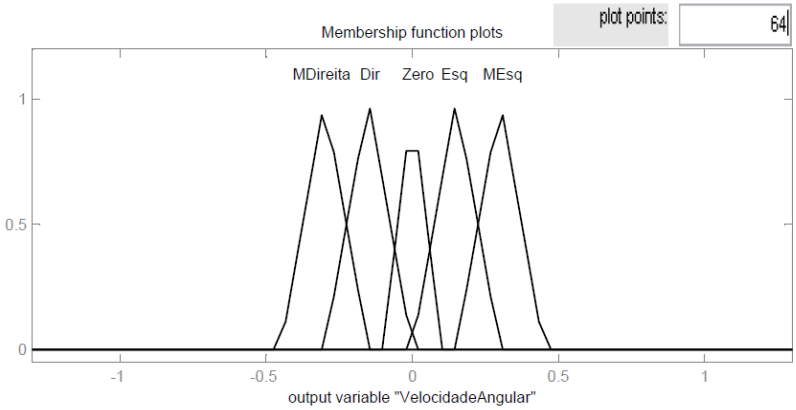


Figura 112 Variável linguística “VelocidadeAngular”.

Nas Tabelas 10 e 11, estão as representações tabulares das regras de controlo para o caso das 25 regras, onde a parte dos valores das tabelas que resultam das regras apresentadas no controlador difuso com 9 regras da secção anterior, estão apresentadas a negrito nas mesmas.

Para este controlador definiu-se que a velocidade linear do robô seria média para todos os casos, eliminando as transições bruscas verificadas em algumas situações.

Tabela 10 Representação tabular das regras para a "VelocidadeLinear".

Velocidade Linear	Erro na Distância					
Orientação		Ref.Muito Esquerda	Ref.àEsquerda	Zero	Ref.àDireita	Ref.Muito Direita
	Muito Direita	Média	Média	Média	Média	Média
	Direita	Média	Média	Média	Média	Média
	Zero	Média	Média	Média	Média	Média
	Esquerda	Média	Média	Média	Média	Média
	Muito Esquerda	Média	Média	Média	Média	Média

Tabela 11 Representação tabular das regras para a "VelocidadeAngular".

Velocidade Angular	Erro na Distância					
Orientação		Ref.Muito Esquerda	Ref.àEsquerda	Zero	Ref.àDireita	Ref.Muito Direita
	Muito Direita	Muito Esquerda	Esquerda	Esquerda	Zero	Zero
	Direita	Esquerda	**Esquerda**	**Zero**	**Esquerda**	Zero
	Zero	Esquerda	**Zero**	**Zero**	**Zero**	Direita
	Esquerda	Zero	**Direita**	**Zero**	**Direita**	Direita
	Muito Esquerda	Zero	Zero	Direita	Direita	Muito Direita

Em seguida, na Figura 113 são apresentadas as 25 regras de controlo desenvolvidas para o sistema. O método de implicação foi o mesmo que o utilizado para as nove regras, assim como o método de agregação, ou seja, através dos operadores lógicos difusos *and* (mínimo) e *or* (máximo), respectivamente. O método de desfuzificação, em semelhança aos anteriores, foi o centróide da área.

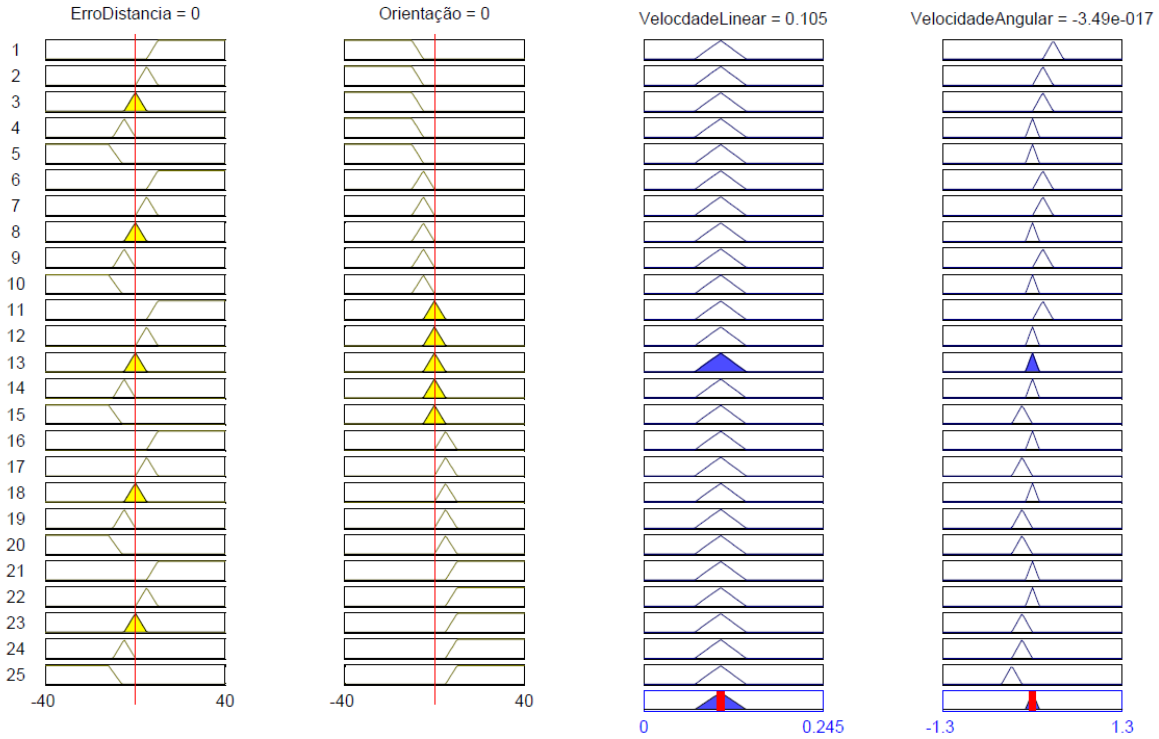


Figura 113 Visualização das regras de controlo.

Na Figura 114 são apresentadas as superfícies de controlo para este sistema. Note-se que apenas três das regras activam acções em que a velocidade é média.

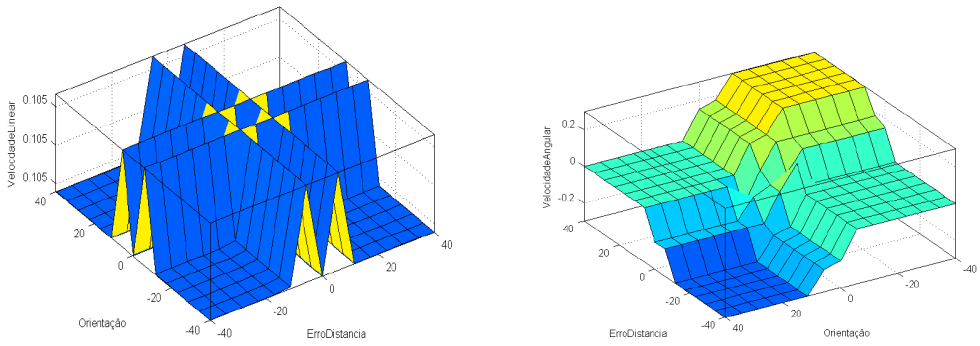


Figura 114 Velocidade Linear (à esquerda) e Velocidade Angular (à direita).

Em relação à implementação do controlador, esta é exactamente igual à anterior sendo que a única diferença reside na quantidade de funções de pertença consideradas e no número de regras a construir (ver secção 5.6.1.2).

5.6.2.1. CONCLUSÕES E RESULTADOS EXPERIMENTAIS

À semelhança com o caso anterior, ou seja, do controlador difuso com 9 regras, procedeu-se à aquisição dos valores medidos pelos sensores laterais S_4 e S_2, mas desta vez utilizou-se um período de amostragem superior, mais precisamente 1,5 segundos. A partir dos dados dos sensores determina-se a distância à parede e o ângulo de orientação com a mesma.

Com a implementação deste comportamento, com 25 regras, verificou-se que o robô era capaz de seguir paredes, e quando comparado com o gráfico da experiência presente na Figura 108, verifica-se que não existem tantas oscilações e o robô tem um comportamento mais suave durante a trajectória. O controlador implementado, com 25 regras, leva a um aumento de complexidade mas verifica-se que o robô atinge a referência aproximadamente no mesmo tempo que o caso anterior, ou seja, aos dez segundos. Mas após este ponto é muito mais suave, como se pode ver na Figura 115, onde são apresentados os resultados experimentais para o controlador de 25 regras.

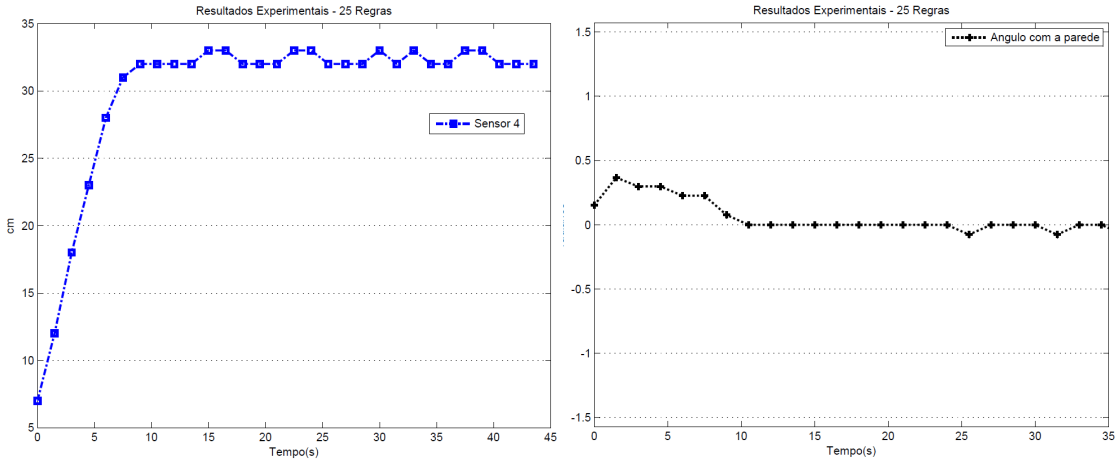


Figura 115 Resultados experimentais obtidos para o controlador de 25 regras.

É de salientar que este teste foi efectuado numa parede lisa com rodapé. Esta diferença na parede, considerando o teste efectuado para o caso das nove regras, aponta para que esta tenha sido responsável pelo erro de dois centímetros em relação ao valor de referência.

Concluiu-se que o comportamento obtido com o controlador de 25 regras, é mais suave que o controlador de 9 regras. Mas deve-se ter em consideração o aumento de complexidade que o aumento de regras acarreta e ao aumento de espaço necessário em memória. Em qualquer um dos dois casos, pode-se ajustar as funções de pertença para um

melhor desempenho, mas esta tarefa é bastante difícil de desempenhar porque não existe uma linha de orientação para a efectuar. Verifica-se também que existe um erro de aproximadamente dois centímetros na distância do robô em relação à referência de 30 centímetros.

Para se perceber se as características da parede influenciam os resultados, procedeu-se ao mesmo teste experimental, mas desta vez numa parede lisa. Os resultados experimentais são apresentados na Figura 116.

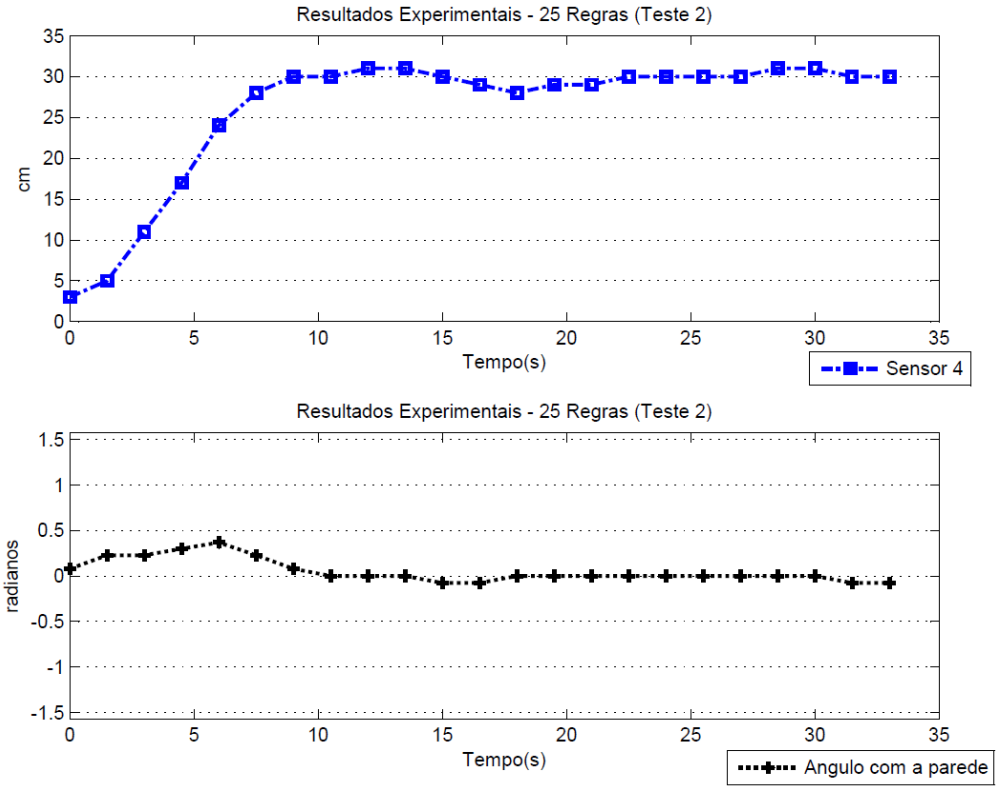


Figura 116 **Resultados experimentais obtidos para o segundo teste.**

Ao analisar os gráficos da Figura 116, verifica-se que o robô tende para o valor de referência pré-definido de 30 centímetros, o que leva a concluir que as características da parede influenciam o resultado final, ou seja, se se colocar o robô a seguir paredes que possuem rodapé o resultado final pode não ser exactamente o esperado. Deste modo conclui-se que a existência de um rodapé pode influenciar o comportamento do robô.

5.6.3. COMPORTAMENTO "EVITAR OBSTÁCULO"

A implementação deste comportamento levou a uma alteração do posicionamento dos sensores frontais de modo a diminuir os "ângulos mortos" do sistema na zona frontal direita. Convém salientar que o robô apenas se desviará dos obstáculos pela esquerda, e se estes estiverem numa zona central de um corredor, o robô ao desviar-se do obstáculo, poderá detectar também a parede da esquerda interpretando-a como sendo um obstáculo e continuando a rodar à esquerda. Se este se aproximar demasiado da parede será activado o comportamento "Emergência" e o robô pára. O novo posicionamento dos sensores frontais está representado na Figura 117.

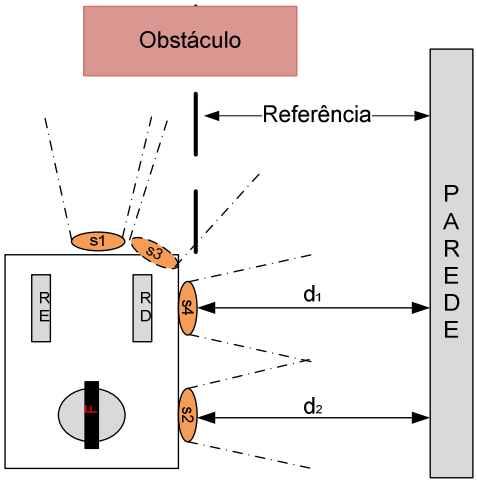


Figura 117 Novo posicionamento dos sensores frontais.

Este comportamento, utiliza como variáveis de entrada as distâncias dos sensores frontais S_1 e S_3. As variáveis linguísticas que foram adicionadas ou modificadas são apresentadas nas Figuras 118 a 120. As variáveis de entrada "DistânciaS1" e "DistânciaS3" foram criadas para se desenvolver o comportamento "Evitar obstáculo".

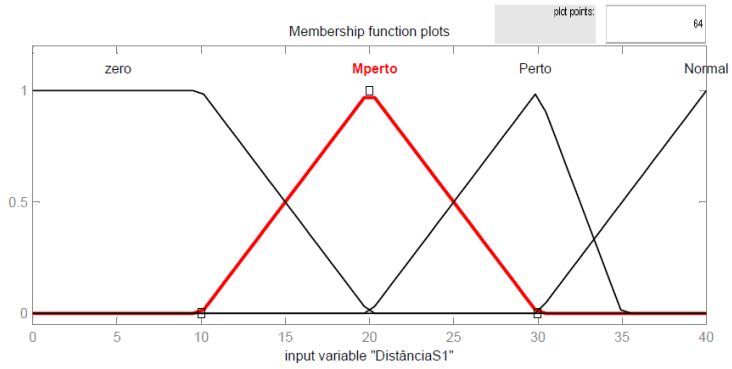


Figura 118 Variável linguística “DistânciaS1”.

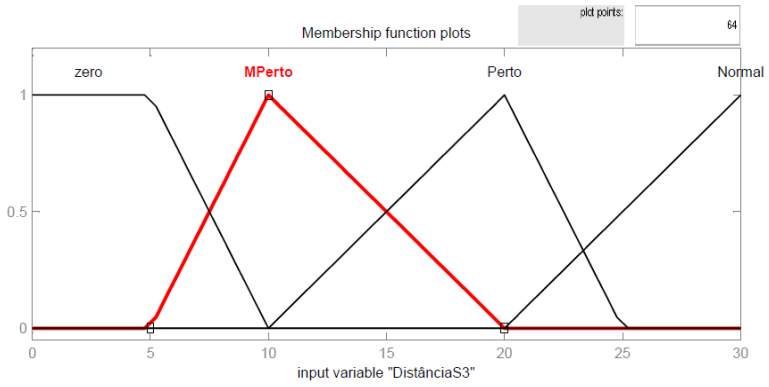


Figura 119 Variável linguística “DistânciaS3”.

As definições apresentadas em seguida, representam a solução final, pois estas foram alteradas de modo a diminuir a área da intersecção entre o conceito “Perto” e ”Normal” para tornar mais suave a transição entre o comportamento “Evitar obstáculo” e “Seguir parede”, pois este último apenas está activo quando as variáveis linguísticas “DistânciaS1” e “DistânciaS3” assumem o termo linguístico “Normal”.

De modo a aumentar a reactividade do sistema aumentou-se a capacidade de rotação do mesmo, para isso foram criados dois novos conceitos na variável linguística de saída “VelocidadeAngular”, sendo que os restantes conceitos foram mantidos, tais como foram apresentados no projecto do comportamento “Seguir Parede”.

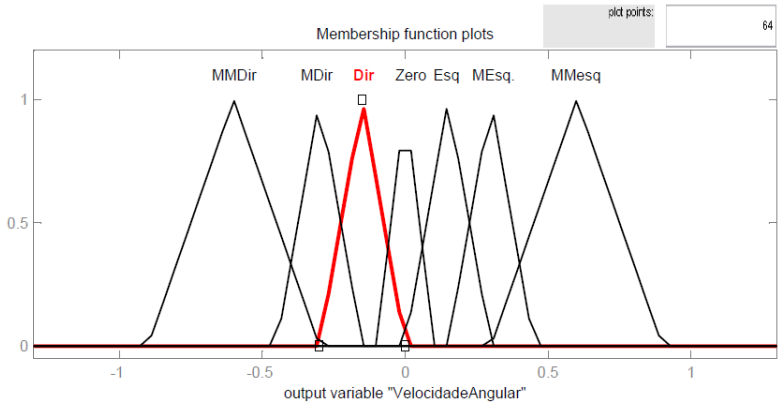


Figura 120 Variável linguística "VelocidadeAngular".

Os novos conceitos criados são "MMDir" e "MMEsq", que representam virar "Muito Muito Direita" e virar "Muito Muito Esquerda".

Considerando todas as possibilidades, e relacionando as variáveis linguísticas "DistânciaS1" e "DistânciaS3", surgiram as seguintes regras apresentadas nas Tabelas 12 e 13. Ao analisar as tabelas verifica-se que quando as variáveis "DistânciaS1" e "DistânciaS3" assumem o valor linguístico "Zero", existe uma referência para o comportamento "Emergência", que é abordado na secção seguinte.

Tabela 12 Representação tabular das regras para a "VelocidadeLinear".

Velocidade Linear		Sensor S_1 - DistânciaS1			
Sensor S_3 – DistânciaS3		Zero	MuitoPerto	Perto	Normal
	Zero	Comportamento Emergência	Comportamento Emergência	Comportamento Emergência	Comportamento Emergência
	MuitoPerto	Comportamento Emergência	Média	Média	Média
	Perto	Comportamento Emergência	Média	Média	Média
	Normal	Comportamento Emergência	Média	Média	Comportamento Seguir Paredes

Tabela 13 Representação tabular das regras para a "VelocidadeAngular".

Velocidade Angular	Sensor S_1				
Sensor S_3		Zero	MuitoPerto	Perto	Normal
	Zero	Comportamento Emergência	Comportamento Emergência	Comportamento Emergência	Comportamento Emergência
	MuitoPerto	Comportamento Emergência	MMEsquerda	MMEsquerda	MMEsuerda
	Perto	Comportamento Emergência	MMEsquerda	MMEsquerda	MEsquerda
	Normal	Comportamento Emergência	MMEsquerda	MEsquerda	Comportamento Seguir Paredes

Este comportamento dota o robô com a capacidade de evitar obstáculos, em que quando as distâncias medidas pelos sensores S_1 e S_3 deixam de estar contidas no conceito "Normal" o robô começa a considerar as acções previstas nas tabelas anteriores.

Na prática, as condições "DistânciaS1" igual a "Zero" ou "DistânciaS3" igual a "Zero" e, "DistânciaS1" e "DistânciaS3" iguais a "Normal" funcionam como regras de contexto fazendo assim a arbitragem de comportamentos. A arbitragem de comportamentos será abordada com maior detalhe na secção 5.6.5, aquando da abordagem da implementação do controlador difuso final, onde é explicada com maior detalhe a integração dos vários comportamentos.

5.6.4. COMPORTAMENTO "EMERGÊNCIA"

Como foi referido na secção anterior, o comportamento "Emergência" será activo quando "DistânciaS1" igual a "Zero" ou "DistânciaS3" igual a "Zero".

As regras para este comportamento são apresentadas nas Tabelas 14 e 15, isto para cada variável de saída, ou seja, "VelocidadeLinear" e "VelocidadeAngular".

Este comportamento depende das mesmas variáveis que o comportamento "Evitar obstáculo" onde as variáveis de entrada e de saída possuem as mesmas definições, ou seja, as mesma funções de pertença que já foram apresentadas na secção 5.6.3.

Tabela 14 Representação tabular das regras para a "VelocidadeLinear".

Velocidade Linear	Sensor S_1 – DistânciaS1				
Sensor S_3 – DistânciaS3		Zero	MuitoPerto	Perto	Normal
	Zero	Zero	Zero	Zero	Zero
	MuitoPerto	Zero	Comportamento Evitar obstáculos	Comportamento Evitar obstáculos	Comportamento Evitar obstáculos
	Perto	Zero	Comportamento Evitar obstáculos	Comportamento Evitar obstáculos	Comportamento Evitar obstáculos
	Normal	Zero	Comportamento Evitar obstáculos	Comportamento Evitar obstáculos	Comportamento Seguir Paredes

Tabela 15 Representação tabular das regras para a "VelocidadeAngular".

Velocidade Angular	Sensor S_1 – DistânciaS1				
Sensor S_3 – DistânciaS3		Zero	MuitoPerto	Perto	Normal
	Zero	Zero	Zero	Zero	Zero
	MuitoPerto	Zero	Comportamento Evitar obstáculos	Comportamento Evitar obstáculos	Comportamento Evitar obstáculos
	Perto	Zero	Comportamento Evitar obstáculos	Comportamento Evitar obstáculos	Comportamento Evitar obstáculos
	Normal	Zero	Comportamento Evitar obstáculos	Comportamento Evitar obstáculos	Comportamento Seguir Paredes

A ideia principal deste comportamento é garantir que quando o robô se encontra muito perto de um obstáculo, este pára de modo a garantir que não existem colisões.

5.6.5. CONTROLADOR DIFUSO FINAL

O controlador difuso final, resulta da integração das regras referidas nos comportamentos abordados nas secções anteriores. As variáveis de entrada e de saída deste controlador foram mencionadas no decorrer do projecto dos vários comportamentos e não sofreram qualquer alteração em relação aos dados já apresentados.

Por isso nesta fase pretende-se apenas apresentar a solução implementada para a arbitragem e fusão de comportamentos. A integração de comportamentos pressupõe um

mecanismo que seja capaz de arbitrar ou seleccionar comportamentos, por esta razão é apresentada na subsecção seguinte a arbitragem e fusão dos comandos do controlador difuso implementado.

5.6.5.1. ARBITRAGEM E FUSÃO DE COMANDOS

No caso da arbitragem de comportamentos utilizou-se o método de contextos dependentes onde se utilizam regras de contexto para activar/seleccionar um ou vários comportamentos. Durante a implementação dos vários comportamentos verificou-se a existência de um conflito entre o comportamento “Seguir Parede” e o comportamento “Evitar Obstáculo”. Para resolver este conflito recorreu-se às regras de contexto onde se activa o comportamento “Seguir Parede” quando não é detectado um obstáculo no universo de discurso definido para as variáveis “DistânciaS1” e “DistânciaS3”. Estas variáveis linguísticas caracterizam a existência de um obstáculo na trajectória do robô e, deste modo, definiu-se que o comportamento “Seguir Parede” seria activo quando as variáveis “DistânciaS1” e “DistânciaS3” assumissem o valor linguístico de “Normal”.

Deste contexto surgiu a seguinte regra contextual:

IF DistânciaS1=Normal **And** DistânciaS3= Normal **Then** “Seguir Parede”

onde, os graus de verdade verificados em cada uma das duas premissas será utilizado na implicação, segundo o operador lógico difuso *and*, ou seja, assumindo o valor mínimo verificado na regra para limitar as acções provenientes de cada uma das regras.

Existe a possibilidade de dois comportamentos estarem activos num dado instante e para valores específicos do universo, mais precisamente quando uma das variáveis, ou as duas variáveis linguísticas “DistânciaS1” e “DistânciaS3”, são caracterizadas simultaneamente pelos termos linguísticos “Normal” e “Perto”, ou seja, as funções de pertença que representam o termo “Normal” e “Perto” possuem um grau de verdade diferente de zero.

É fundamental garantir que a activação de vários comportamentos não resulta num conjunto difuso de saída que aponte para duas soluções opostas, pois assim o resultado na saída seria desadequado, ou mesmo inapropriado à situação actual. Deste modo o projecto de um controlador com vários comportamentos requer um cuidado extra para se garantir a inexistência de conflitos, para que após a desfuzificação surja um valor aplicável e adequado ao controlo do sistema.

Durante o projecto, houve a necessidade de ajustar as funções de pertença “Perto” das variáveis “DistânciaS1” e “DistânciaS3”, onde foi reduzida a área de intersecção com o conceito “Normal” e, deste modo, garantiu-se uma transição mais suave entre os comportamentos “Seguir Parede” e “Evitar obstáculo”.

Como se pode perceber na secção anterior, o comportamento “Emergência” encontra-se bem definido, e possui implicitamente uma regra de contexto na sua definição que limita a sua activação.

Se se considerar novamente as Tabelas 14 e 15, verifica-se a seguinte regra de contexto implícita:

IF DistânciaS1=Zero **Or** DistânciaS3= Zero **Then** “Emergência”

Resumindo quando as variáveis linguísticas “DistânciaS1” e “DistânciaS3” assumem simultaneamente o valor linguístico “Normal”, o comportamento activo é o comportamento “Seguir Parede”. À medida que o robô se aproxima de um obstáculo, fazendo com que uma das duas variáveis altere o seu conceito de “Normal” para “Perto”, vai existir uma mudança gradual do comportamento “Seguir Parede” para o comportamento “Evitar obstáculo”. O mesmo acontece na comutação do comportamento “Evitar obstáculo” para o comportamento “Emergência”, ou seja, à medida que as variáveis “DistânciaS1” ou “DistânciaS3” alteram do conceito de “MuitoPerto” para “Zero”, existirá uma mudança gradual do comportamento “Evitar obstáculo” para o comportamento “Emergência”. Estas mudanças graduais permitem uma maior suavidade na mudança de comportamentos e, consequentemente, nas acções de controlo resultantes. Estas transições suaves resultam num movimento mais suave do robô.

Este tipo de abordagem permite gerir os conflitos devido à co-existência de vários comportamentos no mesmo controlador. Uma vez resolvidos, ou geridos os conflitos, pode-se considerar a próxima etapa, a fusão de comandos.

A fusão de comandos utilizada neste trabalho, baseia-se nos conceitos base do controlador difuso, em que através do operador lógico difuso *or*, consideram-se os valores máximos de cada regra activa para a construção do conjunto difuso de saída. Como foram resolvidos os conflitos entre comportamentos apenas as regras que estão activas geram acções, logo não existirão conflitos nas acções a serem consideradas. Em seguida, e uma vez que as acções já foram agregadas num conjunto difuso de saída (ou mais, neste caso existem dois

conjuntos difusos de saída, um referente à velocidade linear e outro referente à velocidade angular), procede-se então à desfuzificação através de um dos vários métodos possíveis que foram abordados na secção 2.4.3. Para este trabalho foi utilizado o método do centróide da área, que também é disponibilizado na Fuzzy Logic Toolbox do MATLAB, o que permitiu de uma forma rápida comparar os valores obtidos pelo controlador implementado com os valores do projecto do controlador difuso efectuado nesta aplicação. A expressão que este método utiliza foi referida na secção 5.6.1.1, equação (49).

O projecto do controlador através do MATLAB permitiu de uma forma mais clara verificar a existência ou inexistência de conflitos entre as regras que implementam os vários comportamentos, e como consequência permitiu solucionar o problema através de uma análise dos conflitos verificados. Esta análise levou à geração das regras contextuais já referidas, que anularam os conflitos e permitiram a construção de conjuntos difusos de saída sem inconsistências de forma a se obter, após a desfuzificação, um valor adequado e passível de ser aplicado para o controlo do sistema.

5.6.6. CONCLUSÕES E RESULTADOS EXPERIMENTAIS

Com o intuito de testar o robô durante a execução efectuou-se o seguinte teste, o robô foi colocado num ponto inicial do percurso, o qual possuía um obstáculo, como se pode ver na Figura 121. Em seguida, o robô irá guardar em memória os dados, obtidos à cadência de 1,5 segundos durante aproximadamente 45 segundos, ao fim dos quais o robô envia pela porta série os dados recolhidos.

Figura 121 Percurso para o teste experimental.

Os dados obtidos durante este teste foram registados e são apresentados na Figura 122.

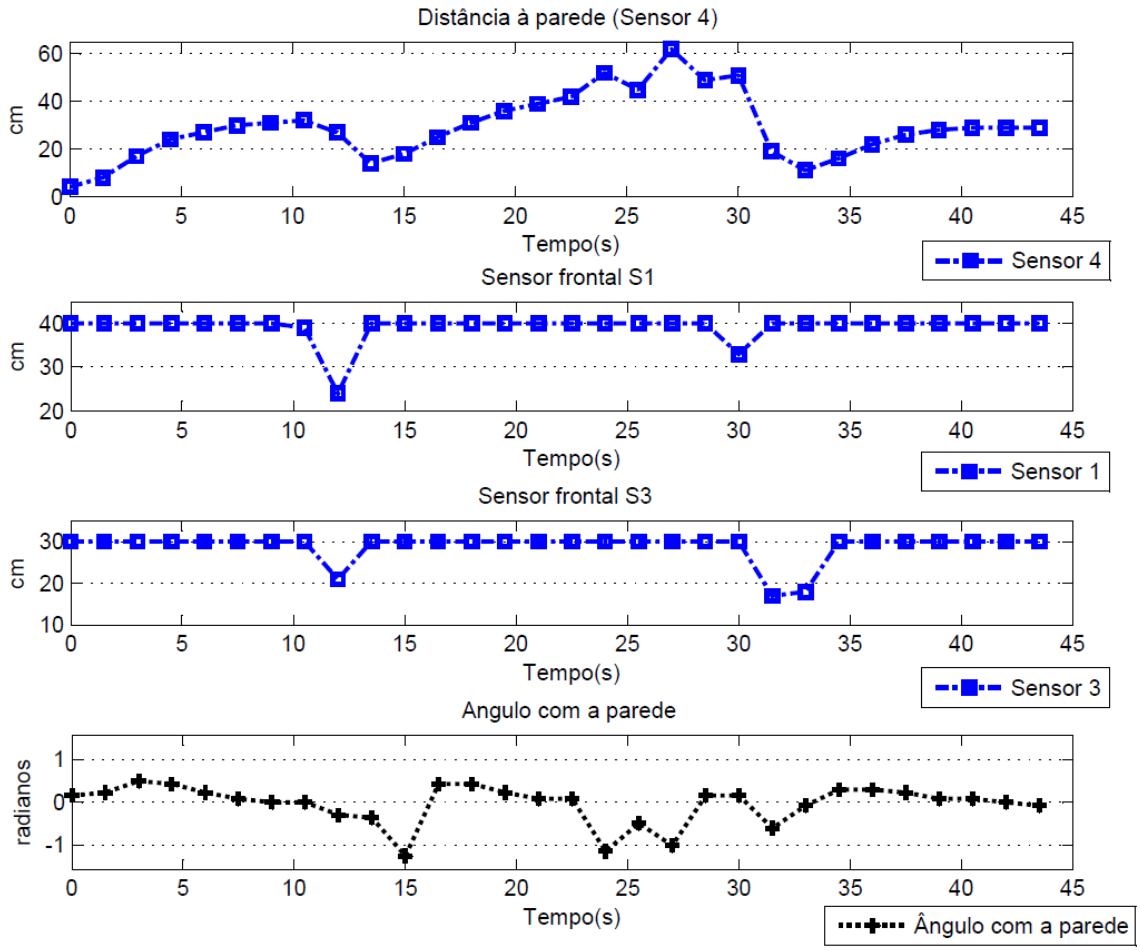


Figura 122 Resultados experimentais referentes ao trajecto definido.

Analisando os gráficos apresentados na Figura 122, pode-se efectuar a seguinte interpretação: nos primeiros dez segundos o robô vai seguir a parede, com tendência para o valor de referência pré-definido de 30 centímetros, como se pode ver no primeiro gráfico com o título "Distância à parede (Sensor 4)". Em seguida é detectado um objecto, aproximadamente aos 13 segundos, como se pode ver nos gráficos referentes ao sensor S_1 e ao sensor S_3, e o robô começa a rodar à esquerda. Este acontecimento remete para o último gráfico referente ao ângulo do robô com a "parede/obstáculo". Se se analisar o gráfico em torno dos 15 segundos, verifica-se uma oscilação que acontece porque o sensor S_4 já está na lateral do obstáculo e o sensor S_2 (lateral traseiro), ainda regista os valores da parede, e por este motivo o ângulo tornou-se mais negativo. Em seguida, verifica-se uma grande variação, que esta ligada à rotação à direita do robô de modo a este se orientar com o objecto, como se este fosse uma parede. Voltando ao primeiro gráfico, em torno dos 25

segundos, verifica-se um aumento da distância do robô à parede, isto significa que o obstáculo já ficou para trás, por isso o robô tenta aproximar-se rodando para a direita seguindo agora a lateral do obstáculo.

Entretanto a parede começa a aproximar-se, como se pode ver nos gráficos referentes ao sensor S_1 e ao sensor S_3, no intervalo [30; 35] segundos. Com esta aproximação o robô começa a rodar novamente à esquerda até ficar paralelo à parede, em seguida, este irá tentar seguir novamente a parede a uma distância pré-definida de 30 centímetros.

Com este teste verificou-se que o robô foi capaz de seguir a parede à distância pré-definida e ultrapassar o obstáculo. Caso o robô não consiga desviar-se do obstáculo é activo o comportamento de emergência e o robô pára.

Constatou-se a suavidade com que o robô se desloca quando está no comportamento "Seguir Parede" e verifica-se um aumento de reactividade quando é detectado um obstáculo e o comportamento "Evitar obstáculo" é activo.

Conclui-se que é possível utilizar a lógica difusa, ou os controladores difusos com sucesso para a implementação de unidades comportamentais. Os controladores difusos permitem respostas mais suaves pois esta é uma característica destes controladores dado que grandes variações na entrada não provocam grandes variações na saída.

As principais dificuldades no projecto de controladores difusos são: a dificuldade que existe na definição das funções de pertença, uma vez que não existe uma linha de guia para a definição das mesmas. Para a sua definição torna-se essencial um bom conhecimento do sistema a controlar, pois este conhecimento permite chegar mais rapidamente à solução mais adequada. Uma outra dificuldade está associada à arbitragem e fusão de comportamentos, pois por vezes torna-se difícil garantir uma transição suave entre comportamentos, e a resolução de alguns conflitos nas acções devido à activação simultânea de comportamentos pode ser uma tarefa difícil. Deve-se garantir que não existem conflitos nas acções pois, se se garantir isso será mais fácil a fusão dos comandos, pois não resultam acções opostas. Deste modo, a fusão será simplesmente a agregação de todas as acções resultantes das regras activas. Por fim apenas é necessário desfuzificar o conjunto de saída de modo a obter um único valor de controlo para o sistema.

5.6.7. SOFTWARE

Na Figura 136 é apresentado um fluxograma onde é apresentado o programa desenvolvido para implementar o controlador difuso no microcontrolador. Numa primeira fase são inicializadas algumas variáveis, o módulo CAN, as interrupções para a recepção das mensagens existentes no módulo CAN, o Timer e a porta série.

Em seguida, aguarda-se pela recepção de uma mensagem CAN proveniente do módulo de aquisição de dados dos sensores que actualize os dados dos sensores. Após esta actualização inicial o programa ficará em ciclo infinito.

Neste ciclo é implementado o controlador difuso. Numa primeira fase tratam-se os dados a serem utilizados no controlador. Todas as variáveis, universos de discurso e funções de pertença utilizadas no controlador são convertidas para o intervalo de valores [0; 255] de modo a ocupar menos espaço em memória. Só após a desfuzificação é que os valores voltam a ser reconvertidos.

Após o tratamento e conversão dos dados, procuram-se os valores das variáveis de entrada do controlador nos respectivos universos de discurso. Quando o valor dessas variáveis for encontrado é retornado o índice do vector onde ele se situa, e desta forma, apenas obtemos os valores das funções de pertença dessa variável linguística para o índice determinado na procura. Depois de determinados os graus de verdade de cada função de pertença, de cada uma das variáveis de entrada do controlador, analisa-se a aplicabilidade de cada regra e determina-se o valor mínimo verificado nas premissas de cada uma das regras (implicação com o operador *and*). Este valor mínimo limitará o grau de verdade das acções impostas por essa regra.

Uma vez concluída a análise e a implicação, agregam-se as acções impostas por cada regra segundo o operador lógico difuso *or*, o que equivale a considerar os máximos para a construção dos conjuntos difusos de saída.

Para a construção dos conjuntos difusos de saída necessita-se da informação que está em memória, mais precisamente das funções de pertença das variáveis linguísticas de saída.

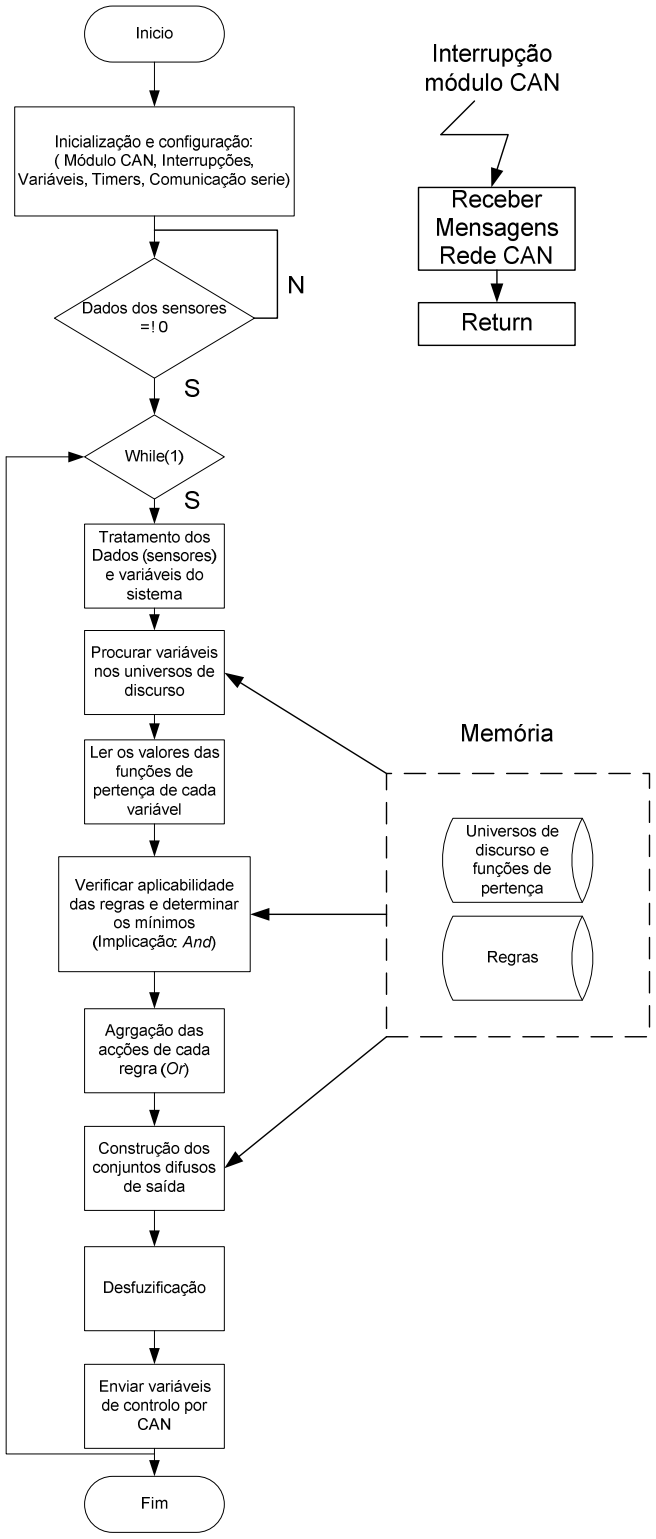


Figura 123 Fluxograma do programa que implementa o controlador difuso.

Para isso, lê-se o valor de cada função de pertença para um valor do universo e limita-se o valor lido ao valor do máximo verificado nas regras para esse conceito. Após esta consideração guarda-se o valor no conjunto difuso de saída para o valor do universo considerado.

Quando estiver concluída esta acção, possui-se dois conjuntos difusos de saída, em que cada um deles possui a consideração final dos valores a aplicar ao sistema em termos de velocidade linear e velocidade angular. Na fase seguinte é necessário efectuar a desfuzificação dos conjuntos difusos aplicando o método do centróide da área (ver equação (49)) onde é feito o somatório do produto do valor do universo que está a ser considerado pelo grau de verdade verificado para esse valor e, divide-se pelo somatório de todos os graus de verdade verificados no universo. O resultado será o valor de controlo a aplicar ao sistema. Após se possuir um valor de controlo para a velocidade linear e angular faz-se a reconversão e determina-se através destas duas variáveis a velocidade que se deve aplicar a cada motor, tal como foi explicado na secção 5.6.1.1.

Por fim, envia-se os valores de velocidade determinados para cada motor através da rede CAN, para o módulo de controlo de velocidade dos motores que irá utilizar esse valor como referência para o controlador PI.

5.6.8. HARDWARE

A implementação do controlador difuso foi efectuada num microcontrolador PIC 18F4585, sendo o esquema do *hardware* deste módulo apresentado na Figura 124, onde numa primeira fase criou-se em memória vectores onde são definidos todos os universos de discurso e funções de pertença necessárias para definir todas as variáveis do sistema de controlo, com um tamanho de 64 elementos, ou seja, para o mesmo número de pontos considerado no projecto efectuado no MATLAB.

Para o esquemático apresentado na Figura 124, foi desenvolvido um esquema para a elaboração da placa de circuito impresso, que se encontra no Anexo C, secção C.3.

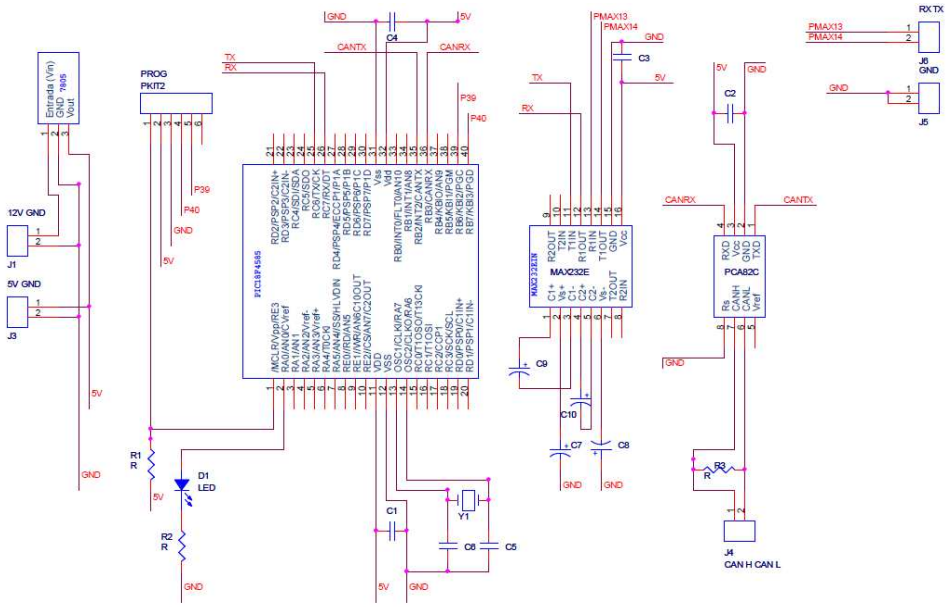


Figura 124 Esquemático do circuito que implementa o controlador difuso.

6. CONCLUSÕES E PROPOSTAS DE DESENVOLVIMENTOS FUTUROS

Neste capítulo são apresentadas as conclusões obtidas a partir dos resultados e da implementação do controlador difuso, assim como abordadas algumas propostas de melhoria e evolução para o sistema.

6.1. ANÁLISE E CONCLUSÕES DOS RESULTADOS

No capítulo anterior foram apresentados os resultados experimentais obtidos no teste do comportamento “Seguir Parede” com o controlador difuso de 9 e 25 regras, respectivamente nas secções 5.6.1.3 e 5.6.2.1.

Da análise dos resultados obtidos nos dois testes, verificou-se que o aumento do número de regras resulta num comportamento mais suave do robô durante o percurso; como

contrapartida leva a um aumento de complexidade no projecto e implementação do controlador.

Numa fase posterior foram adicionados os comportamentos “Evitar obstáculo” e “Emergência”. Após a implementação procedeu-se ao teste descrito na secção 5.6.6, de modo a testar o controlador difuso final. Neste teste fez-se uma análise dos dados de modo a perceber qual era o comportamento do robô. Na prática, verificou-se que o robô foi capaz de se deslocar ao longo do trajecto mantendo a distância de 30 centímetros à parede e, quando detectou um obstáculo, foi capaz de o ultrapassar. Convém referir que existiu a necessidade de reposicionar os sensores frontais, de modo a diminuir os “ângulos mortos” do sistema para diminuir as probabilidades de colisão com objectos que entrem nessas zonas sem detecção, tal como foi referido na secção 5.6.3.

Uma das maiores dificuldades foi a definição das funções de pertença de modo a obter o melhor desempenho do sistema, assim como foi necessário ter um cuidado extra na integração de todos os comportamentos de modo a gerir ou anular conflitos nas acções propostas por cada um dos comportamentos. Para gerir os comportamentos foram utilizadas as regras de contexto, e foi redefinida a função de pertença “Perto” associada às variáveis linguísticas “DistânciaS1” e “DistânciaS3” de modo a diminuir a área de intersecção desta função de pertença com a função de pertença “Normal” das duas variáveis. Esta redefinição resultou numa transição mais suave entre os comportamentos “Evitar obstáculo” e “Seguir Parede”.

Após a gestão dos conflitos, ou eliminação dos mesmos, pode-se fazer a agregação das acções activas e proceder à desfuzificação dos conjuntos difusos de saída, pois existe a garantia que não existem conflitos.

Os objectivos desta tese foram atingidos, uma vez que foi construído de raiz um AGV e foi desenvolvido um controlador difuso capaz de habilitar o robô com a capacidade de reagir ao ambiente.

6.2. CONTRIBUIÇÃO DA TESE

A elaboração desta teste permitiu adquirir conhecimentos associados à lógica difusa e, consequentemente, no projecto e implementação de controladores difusos. O

enquadramento da lógica difusa na robótica permitiu obter algum conhecimento básico sobre arquitecturas de controlo, uma vez que este não era a temática nuclear deste projecto.

Com a construção de um AGV de raiz, várias tarefas foram necessárias, tais como: projecto de um controlador PI, implementação de uma rede CAN e aquisição de dados dos sensores. Estas tarefas permitiram aplicar conhecimentos adquiridos durante o curso, com vista a atingir o objectivo final da elaboração de um AGV e o projecto e implementação de um controlador difuso.

A variedade de temáticas abordadas na elaboração deste projecto permitiu também solidificar alguns conhecimentos na área de controlo de sistemas.

Este trabalho contribui também para o estudo de sistemas difusos e sua utilização na robótica móvel, onde foi desenvolvido uma plataforma de hardware flexível que permite uma fácil integração de futuros módulos para a melhoria do sistema. A utilização de uma rede CAN para permitir uma interacção entre os vários módulos do sistema é um importante factor que corrobora a flexibilidade do mesmo. Esta rede permitiu distribuir as tarefas por vários módulos surgindo assim uma arquitectura distribuída

Efectuaram-se testes experimentais ao sistema e validaram-se os resultados obtidos. Em termos de implementação foram abordadas as questões referentes à implementação de um controlador difuso que previa três comportamentos distintos, são eles: “Seguir Parede”, “Evitar obstáculo” e “Emergência”. A existência de três comportamentos levou à implementação de técnicas difusas para a arbitragem de comportamentos através de regras de contexto e à análise do controlador para se efectuar a fusão de comandos.

6.3. PROPOSTAS DE DESENVOLVIMENTOS FUTUROS

A construção de um AGV pressupõe a capacidade do mesmo para se mover num dado ambiente. Deste modo, surgiu a ideia de propor a integração de um sistema de visão artificial para a detecção de um traçado, como melhoria ao sistema actual.

Propõe-se o estudo da utilização da CMUcam no sistema actual, devido às funcionalidades da mesma e à existência de documentação e software de apoio existente. O software existente e as funcionalidades desta câmara encontram-se descritas no Anexo B, onde as suas características são abordadas com maior detalhe.

Propõe-se também a junção de um novo módulo para a adição de sensores que será abordada na secção 6.3.2.

6.3.1. DETECÇÃO DE LINHAS

A utilização da CMUcam tem como principal objectivo permitir obter características do ambiente para a orientação do AGV, quer através da identificação de linhas brancas paralelas ou apenas de uma única linha.

De seguida são abordadas algumas soluções idealizadas, testes realizados e resultados preliminares, de modo a determinar qual a solução que oferece mais vantagens para a detecção de linhas e determinação da sua orientação. As principais observações e conclusões serão apresentadas no final desta subsecção.

De modo a não aumentar demasiado a complexidade da solução estudada, optou-se por fazer apenas a detecção de uma linha, cuja função é marcar o caminho que o AGV deve seguir.

De seguida, na secção 6.3.1.1, apresenta-se uma análise aos diferentes testes realizados na aplicação CMUcam2GUI, a partir do interface da mesma. Esta aplicação é abordada com maior detalhe no Anexo B.

6.3.1.1. CENTRO DE MASSA

Neste método a linha é identificada através da sua cor, sendo o centro de massa e uma aproximação da área do objecto identificado na imagem, neste caso a linha. Nas Figuras 125 a 131, são apresentadas algumas imagens que mostram os resultados obtidos para algumas orientações da linha a ser detectada. É possível identificar nas imagens duas características importantes; são elas, a vermelho o ponto que indica o centro de massa e a azul a área aproximada do objecto detectado.

Na Figura 125 verifica-se que o centro de massa se situa numa zona central da imagem uma vez que a linha em questão é representada por uma recta e está numa zona central da imagem.

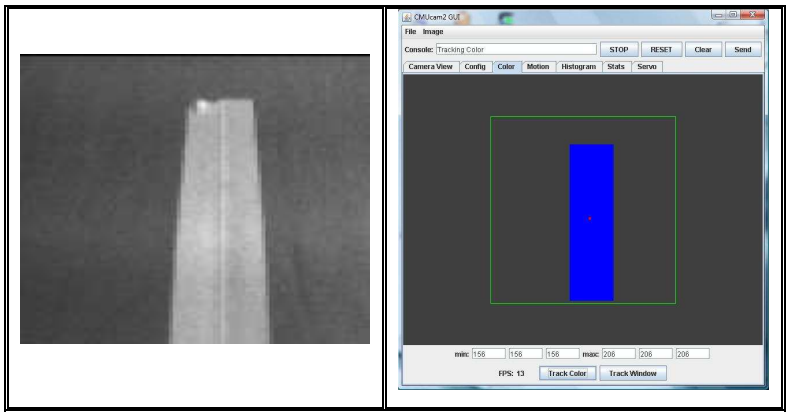


Figura 125 Recta centrada na imagem.

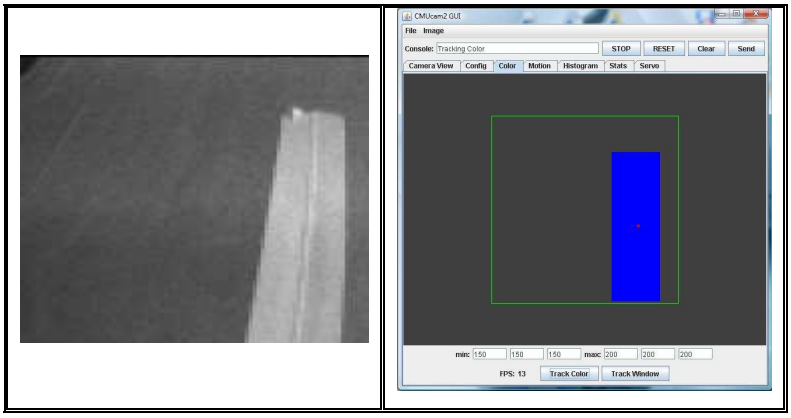


Figura 126 Recta situada à direita na imagem.

Algo semelhante acontece nas Figuras 126 e 127 em que se podem ver duas rectas mas desta vez localizadas à direita e à esquerda da imagem, respectivamente. O centro de massa também se desloca para a zona onde a linha se encontra e situa-se a cerca de metade da altura total da imagem.

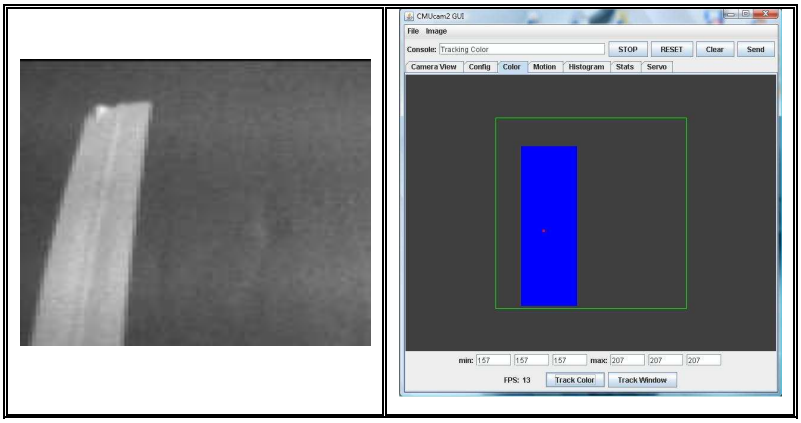


Figura 127 Recta situada à esquerda na imagem.

No caso de se procurar identificar uma curva, neste caso uma curva à direita centrada na imagem (ver Figura 128), verifica-se que a área a azul ocupa a zona inferior direita e o centro de massa também se localiza nessa área.

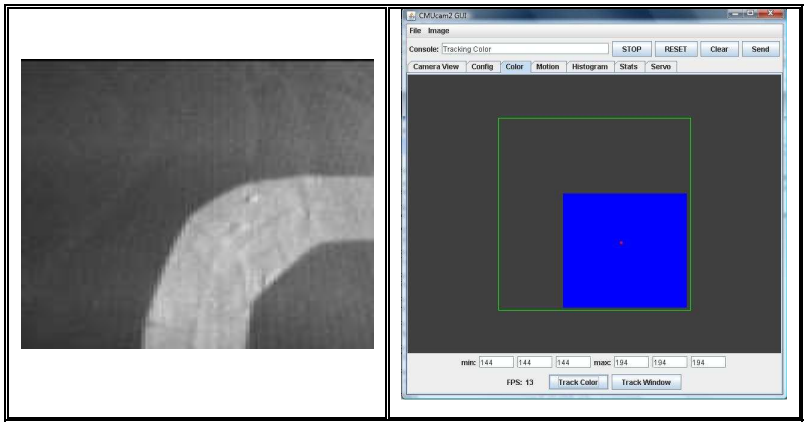


Figura 128 Curva à direita centrada na imagem.

Se se procurar identificar uma curva à direita mas descentrada, que como se pode ver na Figura 129 se localiza agora à esquerda na imagem, constata-se que a área ocupada pelo objecto detectado ocupa toda a zona inferior da imagem e o seu centro de massa está ligeiramente deslocado para a esquerda, fruto da maior quantidade de pixéis brancos presentes à esquerda.

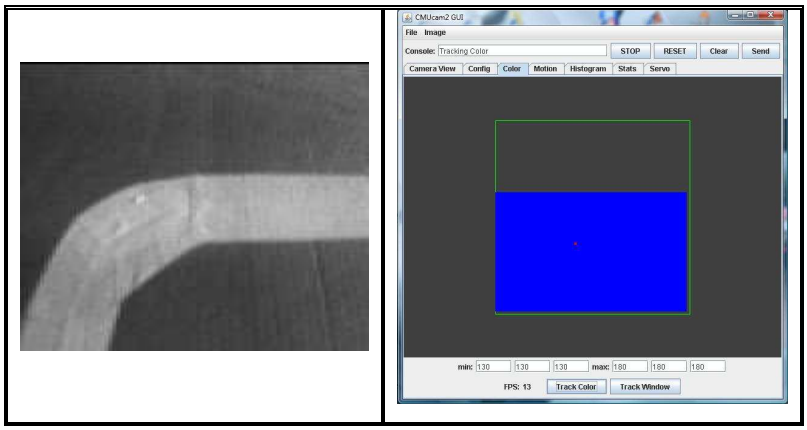


Figura 129 Curva à direita situada à esquerda na imagem.

Se a curva à direita estiver agora descentrada para a direita na imagem, verifica-se (ver Figura 130) que a zona ocupada pelo objecto a azul ocupa a zona inferior direita e o seu centro de massa está aproximadamente no centro dessa área.

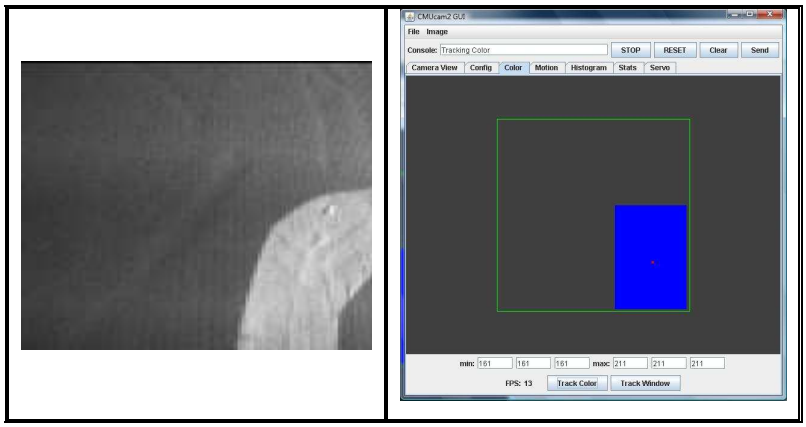


Figura 130 Curva à direita situada à direita na imagem.

Na Figura 131 apresenta-se uma situação extrema em que se supõe que o AGV está completamente perdido, apenas se identificando uma linha na imagem que a atravessa na horizontal. Nesta imagem, a área que se obtém ocupa uma faixa horizontal central que representa a área ocupada pelo objecto. Como seria de esperar, o seu centro de massa está situado no centro dessa mesma área.

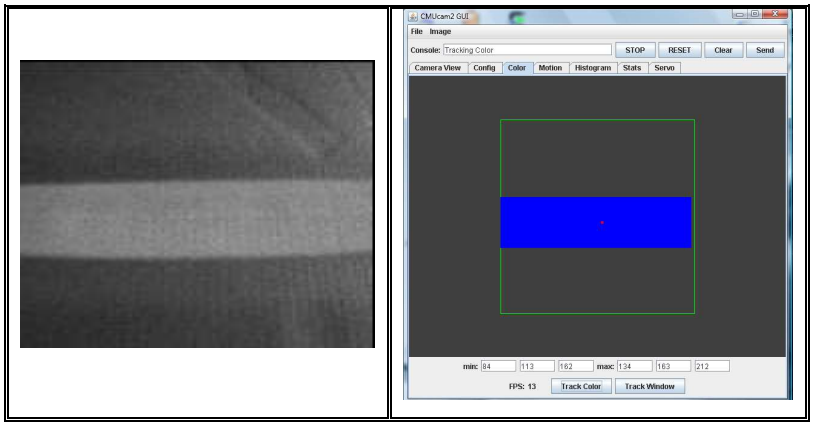


Figura 131 Caso extremo, recta paralela na imagem (AGV perdido no traçado).

Analisando estas imagens, conclui-se que existem situações em que este método de identificação da linha, através do centro de massa calculado, fornece informação que aponta para a presença de uma recta ou curva, mas não existe a garantia que o resultado indique, com toda a certeza, a solução correcta. Basta, para isso, pensar-se na situação em que a curva à direita está descentrada para a esquerda da imagem. Nessa situação representada na Figura 129, apesar de ser uma curva à direita, verifica-se o centro de massa está deslocado para a esquerda devido ao maior número de pixéis brancos presentes à esquerda da imagem.

6.3.1.2. MÉTODO "LINHA A LINHA"

Através de uma outra funcionalidade fornecida pela câmara, é possível fazer uma análise da imagem linha a linha, sendo possível identificar o número de pixéis com a cor pretendida e o ponto médio da linha que contém essa mesma cor. Esta funcionalidade é apresentada nesta secção

Neste método a imagem é analisada linha a linha, sendo detectada a existência da cor pretendida, o ponto médio dessa mesma nessa linha e, numa análise final, é indicado também o centro de massa. Este teste foi também realizado a partir da interface da aplicação CMUcam2GUI.

Como se pode ver nas Figuras 132 e 133, para além da área ocupada pelo objecto detectado que é representada pela cor azul, é também visível a amarelo a área detectada numa análise linha a linha da cor do objecto pretendido. Embora não seja facilmente perceptível nas imagens, é também indicado nas mesmas o centro de massa dos objectos detectados através de um ponto vermelho.

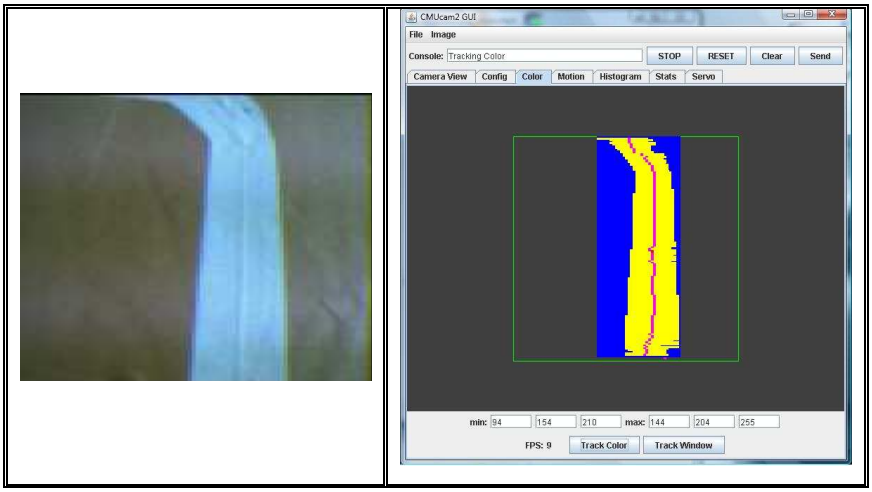


Figura 132 Recta e resultado obtido.

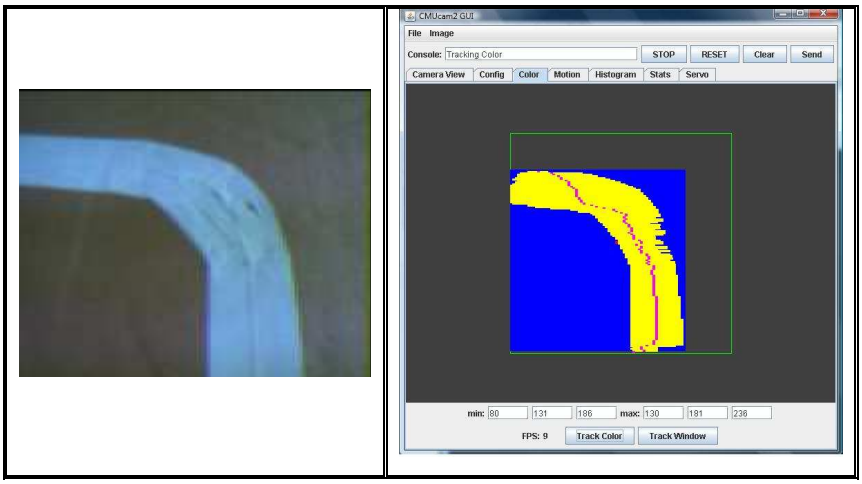


Figura 133 Curva e resultado obtido.

Este método fornece mais informação que o anterior, pelo que pode possibilitar a detecção de rectas ou curvas. No entanto, reparando na imagem da direita da Figura 133, verifica-se que quando na imagem se começa a desenhar uma zona em que a linha está na horizontal todos os pontos calculados (linha a linha) ficaram centrados no centro dessa área formando um conjunto de pontos verticais. Isto é uma consequência de a informação ser tratada linha a linha.

O que se propõe é estudar a possibilidade de utilizar a informação fornecida pela CMUcam (tal como, por exemplo, o centro de massa e a área ocupada na imagem), para a determinação de um parâmetro que possibilite ao robô seguir um traçado, com um comportamento satisfatório, ou seja, sem que se perca irremediavelmente do mesmo.

6.3.2. AUMENTO DO NÚMERO DE SENSORES

Por último, propõe-se também o aumento do número de sensores no AGV, pela adição de um módulo de sensores igual ao desenvolvido, onde apenas será necessário modificar o identificador da mensagem CAN a enviar. O aumento do número de sensores diminuirá os ângulos “mortos” do sistema.

Com a adição deste módulo podem-se desenvolver novos comportamentos, como o seguir paredes à esquerda do robô e evitar obstáculos à direita ou esquerda do mesmo.

Isto pressupõe uma análise do controlador existente, para se saber se este possui poder de processamento suficiente para a implementação dos comportamentos referidos.

Com o aumento do número de sensores o comportamento de emergência poderá prever que o robô recue e tente de novo evitar o obstáculo que não conseguiu transpor numa primeira tentativa.

Referências Documentais

[1] *Fuzzy Control*, Kevin M. Passino, Stephan Yurkovich, Addison-Wesley, 1998.

[2] *Foundations of Fuzzy Control*, Jan Jantzen, Wiley, 2007.

[3] *Controlo Lógico Difuso*, Apresentação, Isabel S. Jesus, Instituto Superior de Engenharia do Porto, 2009.

[4] *Design of Fuzzy Controllers*, Jan Jantzen, Technical University of Denmark of Automation, 1998.

[5] *Fuzzy Controllers*, Leonid Reznik, Newnesm, 1997.

[6] *Introdução à lógica difusa e ao controlo difuso*, Apresentação, A. Pina Martins, Faculdade de Engenharia da Universidade do Porto, 2004.

[7] *Controlo de sistemas autónomos*, Disciplina de Controlo de sistemas autónomos, Alfredo Martins, Instituto Superior de Engenharia do Porto, 2009.

[8] *Controlo de servomotores CC*, Oliveira, Marco e Zucatelli, Fernando, 2007, visitado em 20/07/2010, disponível em http://www.scribd.com/doc/16300774/Controle-de-Servomotores-CC

[9] *The uses of fuzzy logic in autonomous robot navigation*, A. Saffiotti, Center for Applied Autonomous Sensor System, Orebro University - Suiça, 1997.

[10] *Brooks' Subsumption Architecture*, T. Ryan Fitz, Gibbon,2004, visitado em 24/07/2010, disponível em http://www.cs.ucf.edu/~lboloni/Teaching/EEL6938_2005/slides/Presentation_RyanFitzGibbon_SubsumptionArchitecture.ppt

[11] *Notes: The Subsumption Architecture*, Jason P. Garforth, 1997, visitado em 24/07/2010, disponível em http://www.janus.demon.co.uk/alife/notes/subsump.html

[12] *Multiple Objective vs. Fuzzy Behavior Coordination*, Paolo Pirjanian, Maja Matarić, University of Southern California, 1999.

[13] *Behavior Coordination Mechanisms – State-of-the-Art*, Paolo Pirjinian, University of Southern California, 1999.

[14] *A Fuzzy Logic Based Navigation of a Mobile Robot*, Anis Fatmini, Amur Al Yahmadi, Lazhar Khriji e Nouri Masmoudi, World Academy of Science, 2006.

[15] *Fuzzy logic based behaviors blending for intelligent reactive navigation of walking robot*, Ali S. Al-Jumaily, Shamsudi H.M. Amin, Center for Artificial Intelligence and Robotics, 1999.

[16] *Projecto de controladores usando técnicas clássicas – Parte 2*, Disciplina de Sistemas controlados por computador, Ramiro Barbosa, Instituto Superior de Engenharia do Porto, 2008.

[17] *Fuzzy Logic Controller for an Autonomous Mobile Robot*, Vamsi Mohan Peri, Cleveland State University, 2005.

[18] *CANPRES Version 2.0*, Siemens Microelectronics, 1998.

[19] *Comunicação de tempo-real em barramentos CAN baseados no controlador SJA1000*, António Júlio Morais Pires, 2005.

[20] *CMUcam3 Datasheet*, Carnegie Mellon University, 2002.

[21] *CMUcam3 SDK Installation Guide*, Carnegie Mellon University, 2006.

[22] *CMUcam2 Vision Sensor User Guide*, Carnegie Mellon University, 2003.

Anexo A. Protocolo CAN

Como se referiu, o AGV apresenta uma arquitectura distribuída em que a comunicação entre os vários blocos constituintes é feita segundo o protocolo CAN. Deste modo, faz sentido abordar algumas das questões mais relevantes em relação a este protocolo.

O protocolo CAN surgiu na década de 80 e foi especialmente desenvolvido para a indústria automóvel, tendo sido apresentado em 1986 em Detroit pela Bosh.

Este protocolo foi desenvolvido de modo a apresentar características como a difusão de mensagens com identificação pela mensagem e não pelo nó, e a prioridade nas mensagens é definida pela própria mensagem através do identificador, em que a mensagem que tem um identificador menor possui maior prioridade de acesso ao barramento. Este protocolo contém mecanismos de detecção de erros e recuperação de falhas e ainda a retransmissão automática de mensagens corrompidas.

Uma rede CAN pode funcionar com velocidades até 1 Mbps. Este facto, agregado às suas características, torna a rede CAN uma alternativa viável para incorporar aplicações críticas com requisitos como fiabilidade, tempo-real e confinamento de faltas.

O protocolo CAN possui uma arquitectura de comunicação, face ao modelo OSI (Figura 134), com apenas duas camadas implementadas, a camada física e a camada de ligação de dados, ambas implementadas por *hardware*.

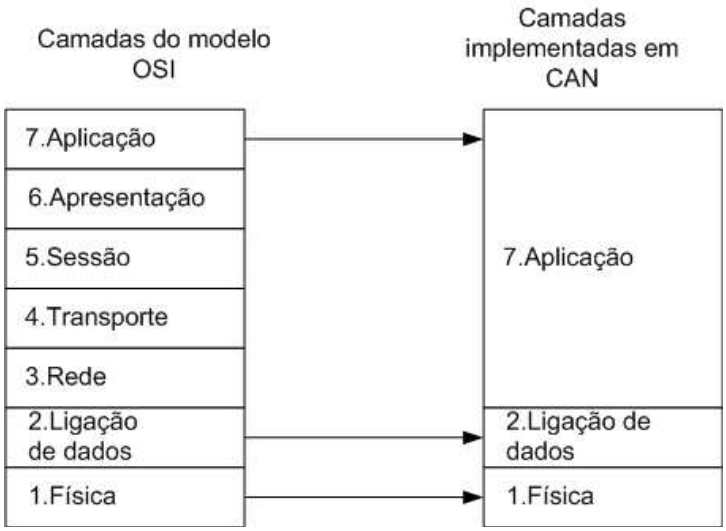


Figura 134 Modelo de camadas OSI e CAN.

A.1. CAMADA FÍSICA

Na camada física estão presentes as características da transmissão física dos dados entre os nós da rede, tais como a codificação/descodificação dos bits, *Timing* dos bits e a sincronização entre os nós.

O barramento CAN é constituído por um par entrançado com dois terminadores nas suas extremidades, de modo a eliminar reflexões, como é visível na Figura 135. Este pode assumir três estados distintos, nomeadamente, *Idle*, bit recessivo e bit dominante.

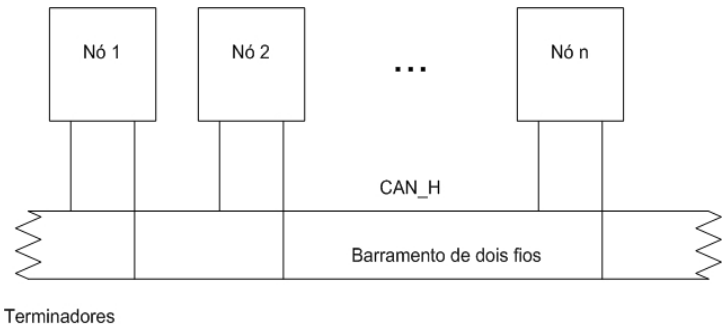


Figura 135 Topologia de uma rede CAN.

Os estados são definidos por tensões diferenciais introduzidas/medidas nas linhas pelos nós. Uma vez que a medida é de natureza diferencial, consegue-se diminuir o efeito de

interferências electromagnéticas, pois os efeitos serão semelhantes em ambos os fios. A medida diferencial permite anular o efeito indesejado.

A.1.1. CARACTERÍSTICAS DO MEIO

O barramento de uma rede CAN pode atingir uma distância de 40 metros, à velocidade máxima de 1Mbps, através de um par entrançado. Com o aumento da distância do barramento a velocidade diminui, sendo para um barramento de 1000 metros cerca de 50 kbps. A Figura 136 mostra um gráfico ilustrativo da velocidade *versus* comprimento do barramento.

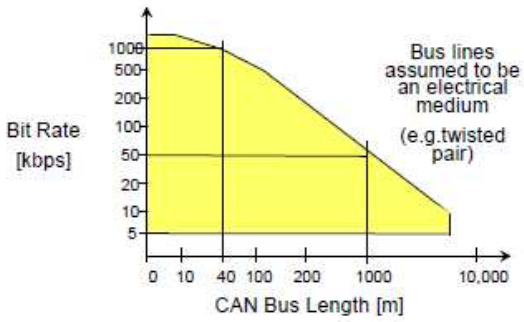


Figura 136 Velocidade *versus* comprimento do barramento [18].

A.1.2. CODIFICAÇÃO

A codificação dos bits é feita por bits Dominantes ou Recessivos, e não através dos níveis lógicos “0” e “1”. Os bits Dominantes ou Recessivos surgem da condição presente nos fios do barramento, CAN_H e CAN_L. Os níveis de tensão previstos no protocolo CAN são apresentados na Figura 137.

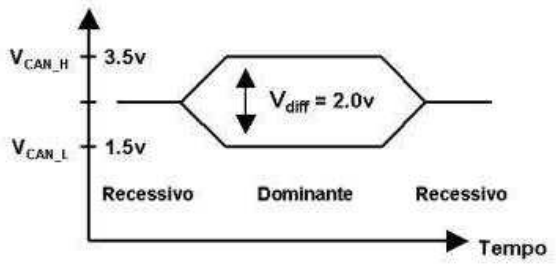


Figura 137 Níveis de tensão presentes no protocolo CAN.

Uma sequência de bits no barramento é codificada em *Non Return Zero* (NRZ). O bit a ser transmitido vai definir o estado do barramento, uma vez que quando nenhum nó está a transmitir o barramento está no estado de *Idle.*

Se não ocorrerem transições no barramento na transmissão de vários bits, a sincronização entre os nós torna-se difícil, podendo fazer com que os nós não identifiquem os limites dos bits. Para resolver este problema de falta de sincronização, este protocolo utiliza a técnica de *bit stuffing*, onde apenas é permitido enviar cinco bits consecutivos da mesma polaridade. Quando um nó pretende transmitir mais do que é permitido, introduz *stuff-bits* de polaridade oposta, de modo a garantir que não existem mais de cinco bits de igual polaridade no barramento, como se pode ver na Figura 138.

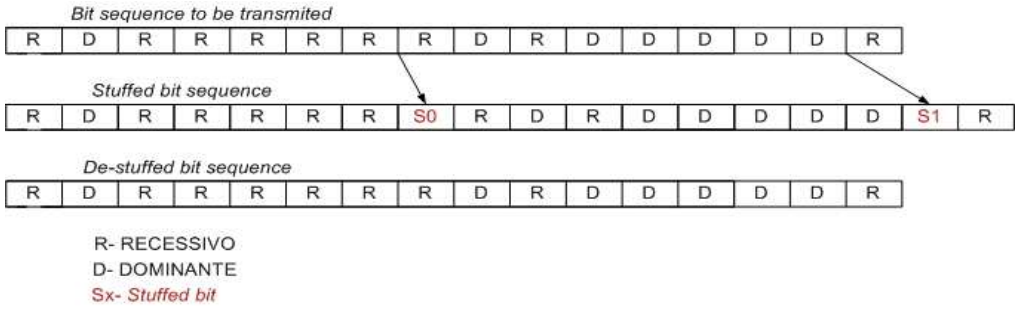


Figura 138 Exemplo da utilização da técnica de *bit-stuffing.*

A.1.3. TIMING

Um nó está em sincronismo com um oscilador local a uma frequência pré-definida, e a cada *n* períodos dá-se o nome de *Time Quantum* (*Tq*), sendo o tempo de transmissão de cada bit múltiplo de *Tq.* De modo a configurar-se a taxa de transmissão, deve-se efectuar o cálculo de *Tq*, ou seja, o número de subdivisões do tempo de um bit, sendo que este valor é por definição superior a 8 e inferior a 25 *Tq* [18].

O tempo de um bit é constituído por quatro segmentos, em que cada um é construído a partir de um múltiplo inteiro do *Time Quantum*, como se pode ver na Figura 139.

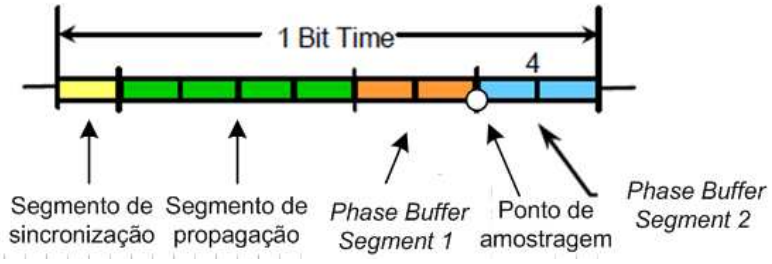


Figura 139 Constituição de um *bit time.*

O segmento de sincronização permite a sincronização dos vários nós, e tem o comprimento fixo de um *Tq.* O segmento de propagação serve para compensar atrasos na propagação do sinal por toda a rede, incluindo os interfaces dos seus nós. O seu comprimento é variável e pode, no máximo, atingir o valor de oito *Tq.*

O segmento *Phase Buffer Segment* 1 possibilita compensar erros de fase no flanco, e permite o alongamento do bit durante a re-sincronização; este deve ter entre um e oito *Tq.* O ponto de amostragem é o ponto onde o nível do barramento é lido e interpretado o valor do respectivo bit. Este ponto localiza-se entre os dois *Phase Buffer Segment*, podendo ser alterado se o *Phase Buffer Segment* 1alterar o seu comprimento.

O *Phase Buffer Segment* 2 também é usado para compensar erros de fase no flanco, e pode ser encurtado durante a re-sincronização; o seu comprimento varia entre um e oito *Tq*, mas deve ter, no mínimo, o mesmo tamanho que o tempo de processamento da informação e não deve ser maior que o comprimento do segmento *Phase Buffer* 1. O tempo necessário para processamento da informação começa logo após o ponto de amostragem, e é utilizado para o cálculo do nível do bit seguinte, onde o seu comprimento deve ser menor ou igual a dois *Tq*.

O segmento *Phase Buffer* 1 pode ser alongado e o segmento *Phase Buffer* 2 pode ser encurtado, como resultado da re-sincronização. A quantidade pela qual o comprimento do bit pode ser ajustado é definida como o comprimento do salto de sincronização (*Syncronization Jump Width*) que representa o número de unidades *Tq* que os segmentos *Phase Buffer* 1 e 2 podem introduzir ou excluir. Este pode atingir um máximo de quatro *Tq*.

Através da configuração destes parâmetros pode-se ajustar o desempenho da comunicação em relação às características do barramento e do ambiente onde este está inserido.

A.1.4. SINCRONIZAÇÃO DE NÓS

Devido aos atrasos na propagação do sinal no barramento, um nó que esteja sincronizado com o mesmo não estará sincronizado com o emissor. Para que um emissor que envia um bit recessivo consiga receber um bit dominante, enviado por um nó sincronizado com o barramento, é necessário estender o tempo do bit ao longo dos quatro segmentos apresentados na secção A.1.3, de modo a garantir a coerência entre todos os nós da rede CAN.

Se se considerarmos o tempo de propagação, este será tanto maior quanto maior for o comprimento do barramento, reduzindo assim a taxa de transmissão.

A.2. CAMADA DE LIGAÇÃO DE DADOS

Na camada de ligação de dados é definido o formato das mensagens que circulam no barramento, sendo que esta possui mecanismos de detecção de faltas e prevenção de falhas.

A camada de ligação de dados pode ser dividida em dois níveis:

1. Controlo da ligação lógica:
    a) filtro de aceitação de mensagens;
    b) notificação de sobrecarga;
    c) gestão e recuperação de mensagens ou situações de erro.
2. Controlo de acesso ao meio:
    a) encapsulamento/desencapsulamento de mensagens;
    b) codificação das mensagens;
    c) detecção e sinalização de erros;
    d) confirmação da integridade das mensagens (*Acknowledge*);
    e) tratamento da informação.

Nesta secção apenas serão abordadas as características do protocolo CAN mais relevantes para a realização deste projecto.

A.2.1. DIFUSÃO E ACEITAÇÃO DAS MENSAGENS

Qualquer um dos nós da rede consegue "ver" as mensagens que circulam no barramento, mas cabe aos nós decidirem se procedem à recepção das mensagens e consequente colocação das mesmas em *buffers*. A sua aceitação ou rejeição é feita através do identificador da mensagem, ou seja, o nó é configurado para receber um conjunto de identificadores que podem ser definidos individualmente ou por conjuntos através do uso de máscaras. Na Figura 140 é apresentado um exemplo de uma rede CAN com quatro nós.

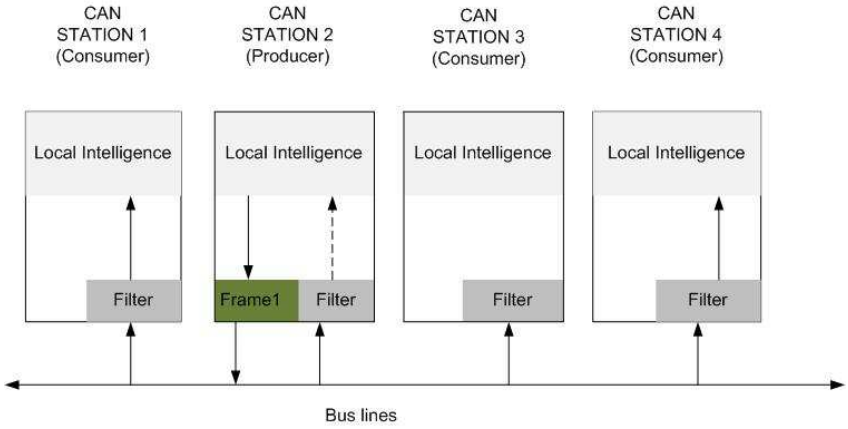


Figura 140 Difusão de mensagens numa rede CAN.

O identificador presente na mensagem permite saber qual a variável presente na mesma. Deste modo, o nó verifica se a mensagem lhe é útil; se sim procede à sua recepção, caso contrário descarta-a. Este método permite que o microcontrolador não seja constantemente interrompido aquando da chegada de uma mensagem, mas apenas quando a mensagem lhe interessa.

A utilização das máscaras permite definir a recepção de conjuntos alargados de identificadores através de um identificador base.

A.2.2. TRAMAS DE DADOS

As tramas de dados são as unidades de comunicação entre as camadas de ligação de dados de cada nó da rede. Sempre que um dos nós enviar dados para o barramento, estes dados são encapsulados em uma ou mais tramas, em que cada uma pode transportar de 1 a 8 bytes de informação [19].

Existem dois tipos de formatos para a trama de mensagens suportadas pelo protocolo CAN, nomeadamente as tramas *standard* e as *extended*, sendo a única diferença entre elas o tamanho do identificador.

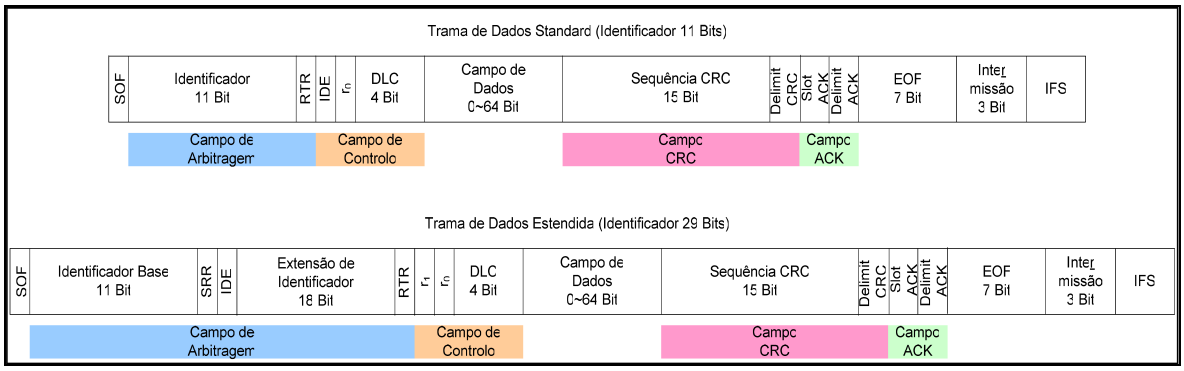


Figura 141 Trama de dados CAN: s*tandard* (em cima) e e*xtended* (em baixo) [19].

Como se pode verificar na Figura 141, uma mensagem do formato *standard* possui um indicador no início da mensagem que sinaliza o início da trama (*Start of Frame* – SOF). Em seguida existe o campo de arbitragem, que é constituído pelo identificador de 11 bits e pelo bit *Remote Transmition Request* (RTR). É através deste que é feita a distinção entre uma trama de pedido remoto e uma trama de dados. Em seguida, como se pode ver na Figura 141, vem o campo de controlo composto pelo bit IDE, que identifica o formato da trama e o campo *Data Lenght Code* (DLC) onde está contida a informação da quantidade de dados que o campo de dados transporta. No caso da mensagem ser um pedido remoto de dados, o campo DLC indica a quantidade de dados pedida. Existe um bit reservado no formato *standard*, que se situa entre o bit IDE e o campo DLC.

O campo de dados pode possuir até 8 bytes de dados. A sua integridade é assegurada pelo *Cyclic Redundancy Check* (CRC). O *ACKnowledge Field* é composto pelo Delimitador *ACK* e pelo bit *Slot ACK*. Este bit é enviado como recessivo, sendo sobreposto pelos receptores como dominante, desde que a recepção seja feita da forma correcta.

A mensagem termina como a indicação *End of Frame* (EOF), que corresponde a 7 bits recessivos. O último campo corresponde ao *Inter Frame Space* (IFS) que indica o espaço entre duas tramas consecutivas. Este campo é composto por 3 bits recessivos, designados de *Intermission*, durante os quais não pode haver nenhuma trama, excepto se a trama estiver sinalizada como trama de erro.

Considerando agora a trama de formato *extended*, presente na Figura 141**,** a principal diferença face ao formato anterior é o tamanho do identificador que neste caso atinge 29 bits distribuídos da seguinte forma: 11 bits na mesma posição do identificador *standard* mais o bit IDE e mais 18 bits de extensão. Esta disposição vai permitir uma compatibilidade com o formato anterior. Neste formato de trama o RTR aparece depois da extensão do identificador, sendo substituído na sua posição normal, que seria a seguir aos 11 bits do identificador, pelo *Substitute Remote Request* (SRR).

Como ambos os formatos existem em simultâneo no mesmo barramento, em caso de conflito no acesso ao meio, está convencionado que o formato *standard* tem sempre prioridade sobre o formato *extended,* devido ao bit IDE que também é contido na arbitragem, para além do bit RTR.

Existem ainda outros tipos de tramas, tais como a trama de pedido remoto de dados, a trama de erro, a trama de sobrecarga, mas não serão abordadas no âmbito deste trabalho.

Convém também referir que existem vários tipos de erros, tais como erros de bit, CRC, ACK e de forma, mas também não serão abordados neste trabalho.

Nesta secção abordaram-se as principais características de uma rede baseada no protocolo CAN, sendo os conceitos mais importantes a reter o formato das tramas *standard e extended*, a constituição do *bit time*, a topologia da rede e a difusão numa rede CAN.

Anexo B. CMUcam3

Este anexo pretende apresentar as funcionalidades e comandos disponibilizados pelo dispositivo para interacção com o mesmo, assim como fazer uma introdução ao *software* existente para a CMUcam.

A versão abordada neste trabalho foi a CMUcam3 (Figura 142), para a qual existem algumas aplicações que permitem fazer interacção com o dispositivo. No entanto, ainda não existe disponível para esta câmara um programa que ofereça funcionalidades para a detecção de obstáculos, tal como existe para a CMUcam2. Para a CMUcam3, o que existe são alguns códigos exemplo para realizar algumas tarefas individualmente como, por exemplo, utilização do cartão de memória e detecção da cor, entre outras. Mas estas aplicações não estão integradas num só programa, o que exige um esforço adicional para a integração e desenvolvimento de código para a aglomeração de diferentes funcionalidades num só programa. Caso se optasse pelo desenvolvimento de código existe a possibilidade de recorrer a bibliotecas abertas para o processamento de imagem, tal como a OpenCV, que está disponível online. Existe também documentação de apoio para auxiliar a utilização destas bibliotecas.

Figura 142 CMUcam3.

Como alternativa ao desenvolvimento de código de raiz, existe a possibilidade de emular na CMUcam3 o código da CMUcam2, que oferece algumas funcionalidades de processamento de imagem. Este software, o CMUcam2GUI, possui uma interface amigável, que permite a aprendizagem das funcionalidades que são fornecidas pela

aplicação que está carregada na CMUcam2, não sendo necessário conhecer todos os comandos para a interacção com a mesma.

Em relação aos blocos constituintes da CMUcam3, esta é composta por um processador ARM7TDMI, integrado num controlador da Philips, um *buffer* e uma câmara, como se pode ver na Figura 143 [20]. A comunicação com o exterior pode ser feita através dos pinos da porta série ou através de pinos de SPI.

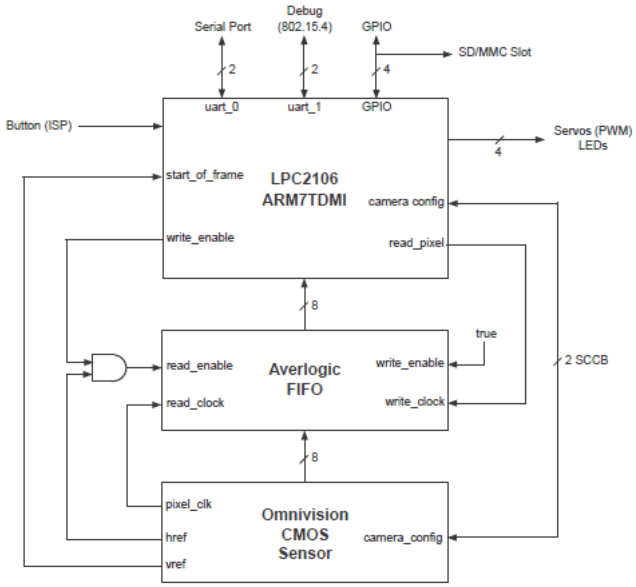


Figura 143 Diagrama de blocos da CMUcam3.

Através do *software* da Philips, *LPC210x FLASH Utility*, pode-se descarregar para o controlador da câmara as aplicações desenvolvidas para a mesma, quer pelo utilizador quer descarregadas do site da CMUcam.org. Neste site encontra-se uma aplicação para a CMUcam3 que emula nesta, as funcionalidades já desenvolvidas para a CMUcam2.

Existem versões desta aplicação para vários sistemas operativos mas, uma vez que todas as ferramentas de desenvolvimento utilizadas neste projecto correm em Windows, optou-se pela instalação das mesmas neste sistema operativo.

Se se desenvolver ou modificar código é necessário compilar o mesmo. Para isso é necessário recorrer ao GCC, ou a outro programa que compile código para o processador alvo.

Após a compilação do código este pode ser transferido para o controlador da CMUcam a partir do programa, *LPC210x FLASH Utility* (já mencionado), através da porta série.

A sugestão indicada no *CMUcam3 SDK Installation Guide*, aponta para a instalação do compilador GCC para o ARM. Para isso é necessário um *software* adicional, o *Cygwin*. Este permite que os sistemas Windows possam, de certa forma, agir como um sistema Unix, sendo que a principal função é a de permitir correr aplicações Unix em sistemas Windows. As instruções para a instalação dos vários programas mencionados, encontram-se no documento *CMUcam3 SDK Installation Guide*, onde existe informação não só sobre a instalação mas também sobre como se deve proceder para carregar os programas no controlador da câmara [21]. Pode ser utilizado outro tipo de compilador, desde que seja adequado ao processador alvo.

Em seguida, na Secção B.1, é abordado o *software* CMUcam2GUI, sendo feita uma pequena abordagem a algumas das funcionalidades e potencialidades desta aplicação que emula as funcionalidades da CMUcam2 na CMUcam3.

B.1. CMUCAM2GUI

Em primeiro lugar, é importante referir que é necessário carregar no controlador da CMUcam3 o código referente às funcionalidades existentes para a CMUcam2. Em seguida, pode-se iniciar a aplicação CMUcam2GUI, desenvolvida em JAVA, que está disponível *online*, através do *links*, http://www.cs.cmu.edu/~cmucam2/downloads.html. Encontra-se também aqui o manual de utilização da aplicação, que contém a informação essencial à sua utilização e compreensão das funcionalidades disponibilizadas.

A aplicação interage com a câmara através da porta série, sendo as acções transformadas em comandos que são enviados para a câmara de uma forma transparente. Os comandos estão descritos no manual da aplicação, de modo a se poder recorrer a este suporte para melhor se compreender o funcionamento do sistema.

O primeiro passo a executar no início da aplicação é escolher a porta série correspondente à comunicação com a câmara (Figura 144).

Figura 144 Interface para selecção da porta série.

Após a selecção da porta, a interface é iniciada e estão reunidas as condições para a interacção com a câmara. Numa primeira fase deve-se focar a imagem, ajustando a lente da câmara, de modo a obter uma imagem, o mais nítida possível que é visível através da aplicação.

B.1.1. INTERFACE

Em seguida irá ser feita uma introdução básica às funcionalidades disponíveis na aplicação.

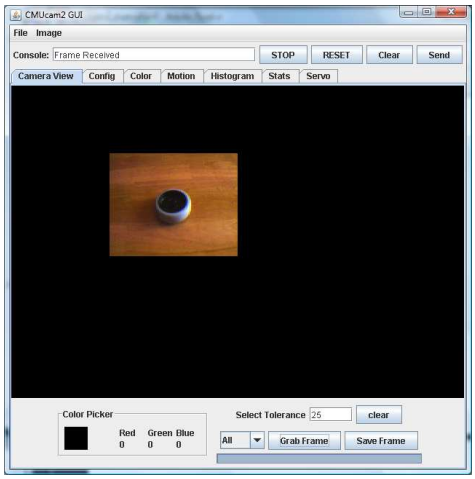


Figura 145 Aplicação CMUCam2GUI.

Como se pode ver na Figura 146, a aplicação possui uma caixa de diálogo, para a comunicação com a câmara através de comandos, várias abas para a transição entre funcionalidades e configurações. Nesta imagem é possível observar para além da consola, as funcionalidades fornecidas na aba de vista da câmara.

Em seguida é apresentada a legenda da Figura 146, onde são descritos os vários elementos enumerados na Figura.

1. Interface de comunicação com a câmara através de comandos e botões de atalho para parar a acção actual, reiniciar a câmara, limpar a caixa de diálogo e envio do comando;

2. Aba de selecção das várias funcionalidades existentes, tais como, vista da câmara, configuração, detecção da cor, histograma, estatísticas e controlo dos servomotores;

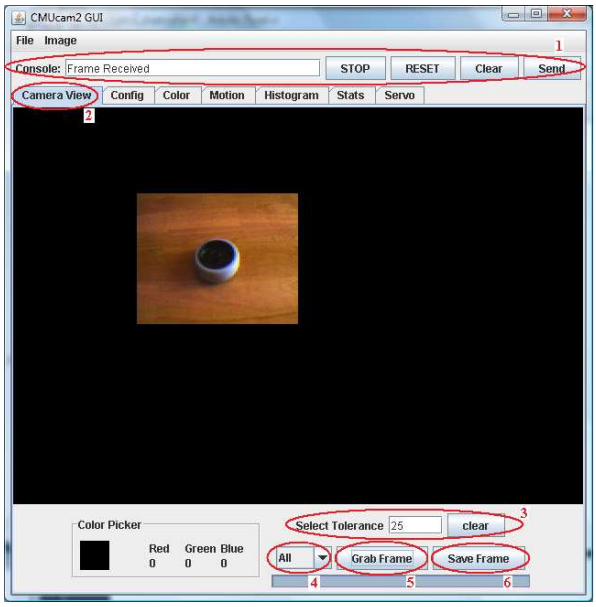


Figura 146 Descrição da interface.

3. Configuração da tolerância na selecção da cor a detectar;

4. Selecção das matrizes que compõem a imagem (RGB, R, G, B);

5. Adquirir imagem da câmara;

6. Gravar imagem.

Na aba de configuração de parâmetros da câmara, visível na Figura 147, podem-se ajustar parâmetros como a matriz a utilizar, resolução, método de detecção, filtro de ruído, janela virtual, entre outros.

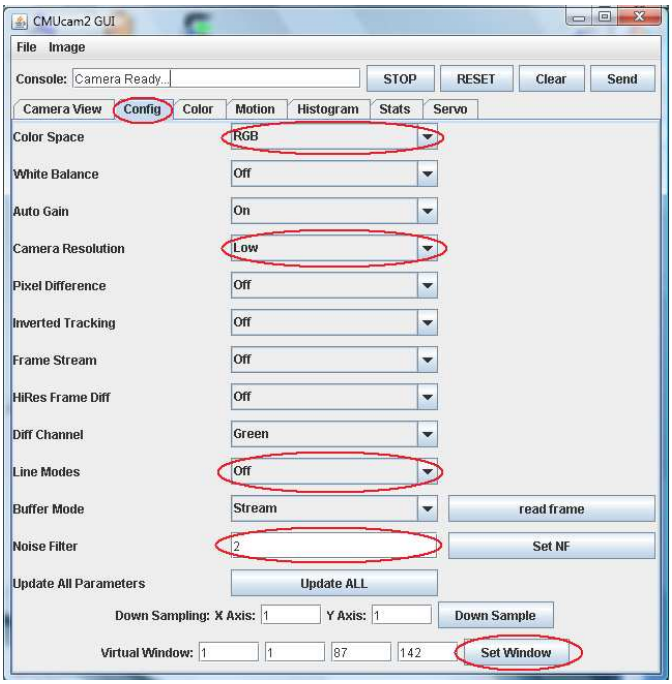


Figura 147 Descrição da interface, aba de configurações.

A aba *Color*, visível na Figura 148, permite visualizar um mapa resultante da detecção da cor pretendida onde, dependendo do método seleccionado nas configurações, se pode visualizar quer o centro de massa da cor pretendida com uma tolerância associada, quer a área detectada com essa mesma cor. Num dos métodos a imagem é analisada linha a linha, sendo identificada a cor pretendida na linha e indicado o centro de massa nessa mesma linha. Esta acção é efectuada para todas as linhas da imagem e, no final, é indicado também o centro de massa da cor pretendida mas considerando a imagem total.

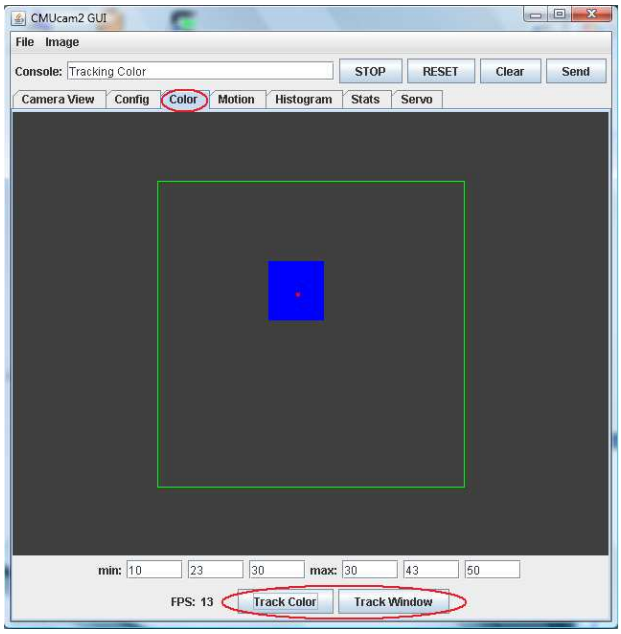


Figura 148 Interface, aba de detecção por cor.

Na aba *Motion* é possível detectar o movimento através do método de diferenciação entre *frames* numa serie de imagens. Como se pode ver na Figura 149, é guardada numa matriz de 8x8 bytes uma imagem abstracta.

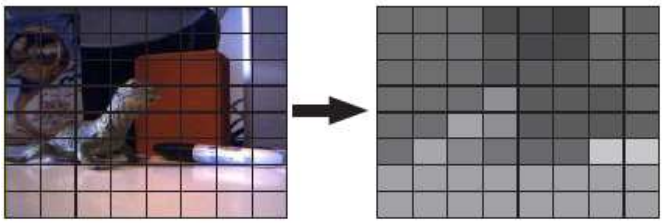


Figura 149 Matriz abstracta resultante [22].

É através da matriz abstracta de referência que a diferenciação irá ser feita. Quando uma nova imagem é recebida, é transformada numa matriz abstracta que irá ser subtraída à matriz de referência. Se for detectado um valor maior que o definido no campo *threshold*, é assinalada uma mudança [22].

Um exemplo de uma matriz resultante da detecção de movimento é apresentado na Figura 150; a azul é assinalada a área que sofreu alteração e a verde o centro de massa.

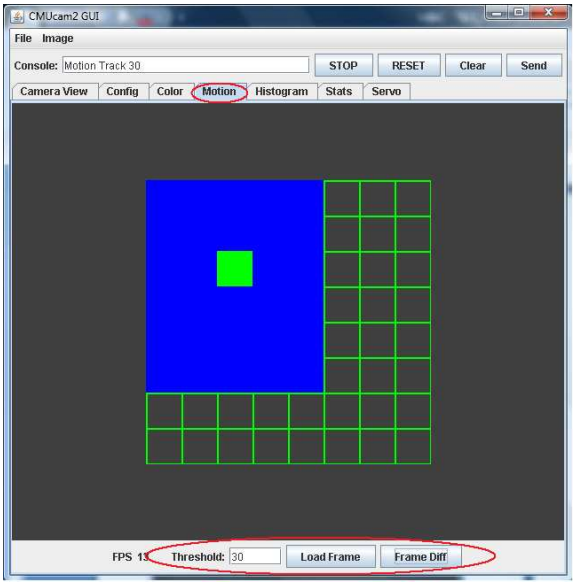


Figura 150 Interface, aba de detecção de movimento.

Em relação à aba *Histogram*, esta permite visualizar a intensidade em cada uma das componentes RGB que compõem a imagem. Na Figura 151 pode-se ver um exemplo das intensidades encontradas na componente *Green* de uma imagem. Esta é uma ferramenta que permite analisar o conteúdo da imagem, e pode ser usada para o reconhecimento de objectos, para o ajuste da cor ou para distinguir diferentes texturas [22].

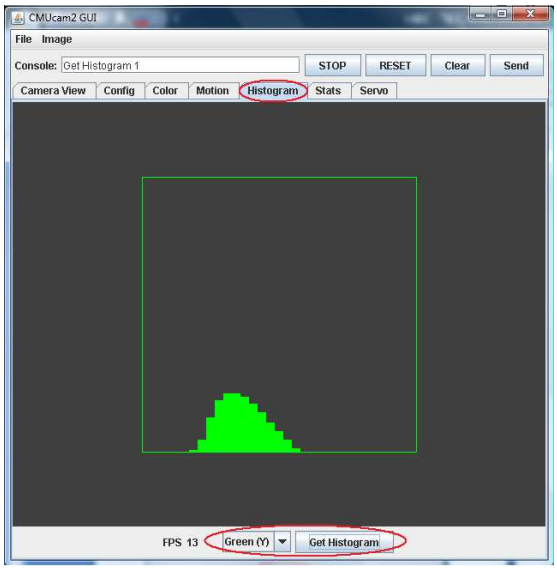


Figura 151 Interface, aba de visualização de histogramas.

Na Figura 152 é visível o conteúdo disponibilizado através da aba *Stats*. Estão reprsentadas três rectângulos verticais que indicam a média da intensidade verificada em cada uma das componentes RGB.

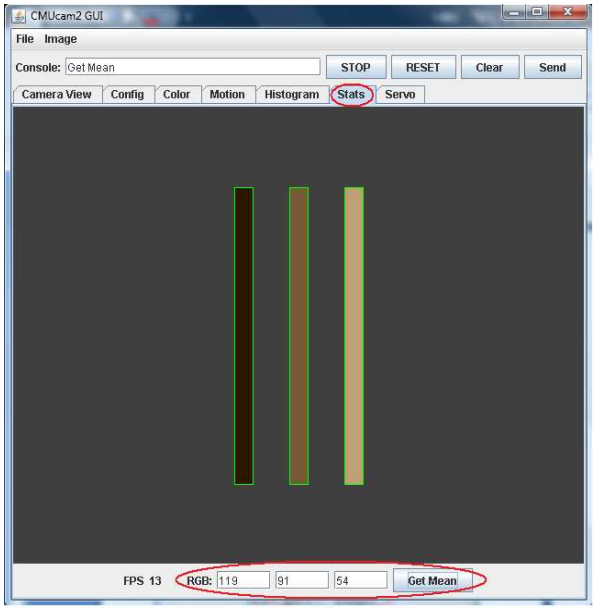


Figura 152 Interface, aba de *Stats*.

A última aba disponível na aplicação diz respeito ao controlo dos servomotores. Esta funcionalidade não é abordada neste trabalho; de qualquer maneira, apresentam-se na Figura 153, as opções disponíveis nesta aba.

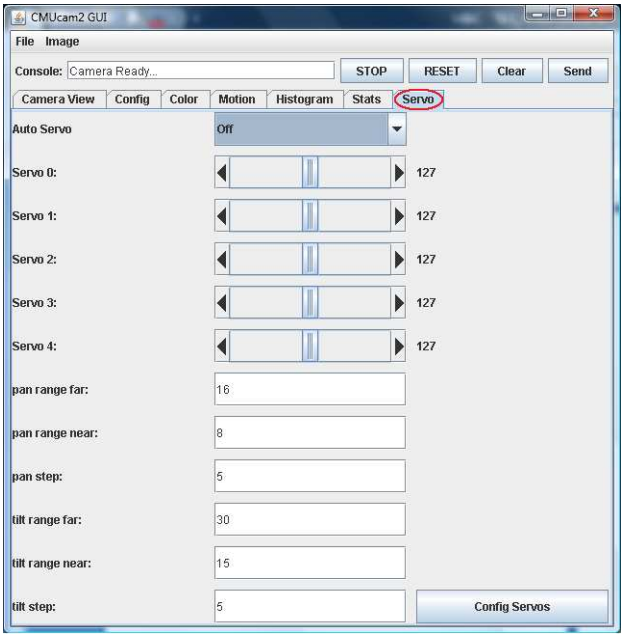


Figura 153 Interface, aba referente ao controlo dos servomotores.

A aplicação CMUcamGUI2 permite utilizar, de uma maneira transparente, as funcionalidades oferecidas pelo programa que está carregado no controlador da câmara. É

essencial conhecer os comandos para a interacção com a câmara, para uma possível futura integração da mesma no sistema.

Na secção B.2, apresentam-se os comandos que permitem interagir com a câmara e obter os dados pretendidos.

B.2. COMANDOS

Nesta secção é feita uma descrição dos comandos utilizados na interacção com a câmara que se relacionam com o processamento de imagem e transmissão dos dados; os comandos referentes ao controlo dos servomotores não serão aqui abordados.

Os comandos são agrupados em secções consoante a sua funcionalidade. Para mais informações o documento completo está disponível *online* através do *link* www.cs.cmu.edu/~cmucam/cmucam2/CMUcam2_manual.pdf [22].

Todos os comandos são enviados usando caracteres ASCII, e a confirmação da resposta é feita por ACK, ou NCK no caso de detectar um erro. No fim de cada ACK, ou NCK, é enviado um **\r** seguido de **":";** isto significa que a câmara está à espera de um novo comando. O **\r** é também utilizado para terminar cada linha e activar cada comando, para além de ser usado para colocar a câmara em modo *Idle*. São usados espaços para separar argumentos.

B.2.1. COMANDOS PARA O *BUFFER*

Existem dois comandos nesta categoria; são eles o *Buffer Mode* e o *Read Frame*, apresentados na Tabela 16.

Tabela 16 Comandos relacionados com o *Buffer*

BM active \r	Buffer Mode
RF \r	Read Frame

BUFFER MODE

Este comando, é representado por "BM", tem como função activar a utilização do *buffer*. O seu valor por omissão é "0", o que significa que novos *frames* são carregados no *buffer* constantemente. O valor "1" significa que apenas um *frame* permanece no *buffer*. Isto

permite vários processamentos sobre a mesma imagem, ou seja, todos os comandos são aplicados ao *frame* actual que está no *buffer* em vez de capturar um novo *frame*. Deste modo podem-se obter os histogramas dos três canais e, em seguida, detectar a cor ou chamar o comando "*get mean"*. Se se introduzir o comando "RF" irá ser capturado e carregado para o *buffer* um novo *frame*. Quando o "BM" está desactivo, o comando RF não é necessário para a aquisição de novos *frames* [22].

READ FRAME

Este comando, representado pela sigla "RF", permite ler um novo *frame* para o *buffer*. Não deve ser usado para capturar novos dados quando o *Buffer Mode* está activo, caso contrário será carregado um novo *frame*, mesmo após a chamada da função de processamento.

B.2.2. COMANDOS PARA O MÓDULO DA CÂMARA

Os comandos englobados para esta categoria são os apresentados na Tabela 17 [22].

Tabela 17 Comandos relacionados com o módulo da câmara.

CR [reg1 value1 [reg2 value2 ... reg16 value16]]\r	Camera Register
CP boolean \r	Camera Power
CT boolean \r	Camera Type

CAMERA REGISTER

Este comando, representado por "CR", permite configurar os registos internos. Toda a informação enviada por este comando deve ser em caracteres decimais, a não ser que o "*raw mode*" seja activado previamente. É possível enviar até 16 combinações de valores dos registos. As configurações anteriores não serão repostas nos valores por omissão entre dois comandos "CR", no entanto, o envio deste comando sem argumentos repõe as configurações de origem.

CAMERA POWER

Este comando desliga e liga novamente a câmara, com os argumentos "0". Este deve ser usado em situações em que a carga da bateria tem de ser preservada. De referir que as imagens contidas no *buffer* podem ser corrompidas quando a câmara é desligada.

CAMERA TYPE

Este comando permite alternar entre tipos de câmaras e só deve ser utilizado quando a câmara estiver no modo *slave* [22]. Com o valor "0" activa o modo "ov6620" e com o valor "1" activa o modo "ov7620". O modo *slave*, por omissão é o modo "ov6620".

B.2.3. COMANDOS PARA CONTROLO DAS TAXAS DE DADOS

Os comandos referentes a esta categoria encontram-se na Tabela 18 [22].

Tabela 18 Comandos para controlo das taxas de dados.

DM value \r	Delay Mode
PM mode \r	Poll Mode
PS number \r	Packet Skip
RM bit_flags \r	Raw Mode
PF boolean \r	Packet Filter
OM packet mask \r	Output Packet Mask

DELAY MODE

Como se pode ver na Tabela 18, este comando é representado por "DM", e activa o *Delay Mode* que controla o atraso entre caracteres transmitidos pela porta série, deste modo, permitindo a interacção com processadores mais lentos. O seu valor pode ser definido entre "0" e "255", em que o valor "0" (por omissão) representa a inexistência de atraso e o valor "255" representa o atraso máximo possível. Cada unidade de atraso é igual ao tempo de transferência de um bit, à taxa de comunicação actual.

POLL MODE

O comando *Poll Mode*, "PM", quando está definido como "0", permite retornar um *stream* contínuo da função de processamento. Quando o valor está definido como "1", apenas é retornado um pacote quando a função de processamento é chamada. Se estiver definido com o valor "2", então o *Poll Mode* vai esperar que um objecto seja detectado e só depois retorna. Este comando é útil se for necessário mudar rapidamente os parâmetros ou se o sistema tiver um processador lento que não consiga aguentar a *frame rate*.

PACKET SKIP

O comando Packet Skip, “PS”, controla se os pacotes devem ser ignorados ou não. O seu valor por omissão é “0”, o que significa que todos os pacotes serão transmitidos. O valor “1” significa que todos os outros pacotes serão ignorados e o valor “2” significa que apenas será mostrado um pacote a cada segundo. Este comando é útil se se necessitar de diminuir a taxa de dados, para que o processador consiga manter uma *stream* de dados quando o *poll mode* está activo.

RAW MODE

Este comando, é representado por “RM”, permite ler os três primeiros valores dos *bits* menos significativos para configurar as definições. Todos os bits limpos activam por omissão o modo visível ASCII. Se o *bit* 0 estiver activo, então todas as saídas da câmara estão em pacotes de bytes não processados. O formato dos dados nos pacotes será alterado, de modo a não incluírem espaços ou estarem formatados como texto ASCII legível. Em vez disso retorna um byte com o valor “255” ao início de cada pacote, o caracter de identificação, e finalmente o pacote de dados. Não será enviado um “\r” após cada pacote, deste modo deverá usar “255” para sincronizar a recepção de dados. Qualquer *byte* com o valor “255” que possa ser enviado como parte do pacote será redefinido para o valor “254” para evitar confusões. Se o bit 1 estiver activo, não serão enviadas as confirmações “ACK\r” e “NCK\r”. Se o bit 2 estiver activo, a entrada será lida como valores de um byte sem processamento. Neste modo, após dois comandos dos valores dos *bytes* serem enviados, envia um *byte* a indicar quantos argumentos se irão seguir.

PACKET FILTER

Este comando é representado pela sigla “PF” e activa o modo de filtro dos pacotes, inactivo por omissão. O valor “1” faz com que apenas o primeiro pacote vazio seja mostrado quando um objecto não seja detectado, não sendo transmitido mais nenhum pacote até que o objecto volte a ser visível.

OUTPUT PACKET MASK

Este comando, representado por “OM”, permite definir a máscara de saída para os vários pacotes. O tipo de pacotes está referenciado na Tabela 19.

Tabela 19 Elemento identificador do método e tipo de pacote.

#	Tipo de detecção	Pacote
0	Detecção de cor	T
1	Adquirir média (Get mean)	S
2	Diferença entre frames	T
3	Non-Tracked Packets[2]	T
4	Additional Count info[3]	T, H
5	Detecção da cor da linha Modo 2	T
6	Detecção da cor da linha Modo 1 e 2	S

O primeiro argumento selecciona o tipo de pacote e o segundo a máscara. A máscara deve ser um único *byte* que representa o tamanho da máscara do pacote de detecção. Um valor de "255" permitirá que todos os parâmetros sejam mostrados, enquanto o valor de "3" apenas permitirá que os dois parâmetros sejam mostrados. Cada máscara, para cada tipo de pacote, será guardada separadamente e permanecerá activa até que a câmara seja reiniciada.

[2] *Non-Tracked Packets* - pacotes que são mostrados quando o objecto a ser identificado não é detectado. Se este campo está definido com o valor "0", então o pacote é mostrado quando o objecto não é encontrado; se estiver definido com "1" então apenas é enviado "T 0" quando não for detectado o objecto. Por último, se o valor é definido para "2" (por omissão) então o pacote será idêntico a um pacote de detecção daquele tipo [22].

[3] *Additional Count information flag* – permite o acesso aos 16 *bits* para valores do contador para detecção da cor ou para elaboração do histograma. O valor "0" (por omissão) desactiva os 16 *bits* para valores, e o valor "1" adiciona os 16 *bits* de contagem dos pixéis detectados em dois bytes separados, o primeiro para *o byte* menos significativo e o segundo para o mais significativo. O valor "2" adiciona 16 *bits* para a contagem de todos os pixéis usados para gerar o histograma com os dois primeiros *bytes* após o "H" nos pacotes de histograma. O valor "3" activa ambos os modos simultaneamente [22].

B.2.4. COMANDOS PARA A JANELA DE IMAGEM

Na Tabela 20 são apresentados os comandos pertencentes a esta categoria [22].

Tabela 20 Comandos para a janela de imagem.

SF [channel] \r	Send Frame
DS x_factor y_factor \r	Down Sample
VW [x y x2 y2] \r	Virtual Window
FS boolean \r	Frame Stream
HR state \r	HiRes Mode
GW \r	Get Window
PD boolean \r	Pixel Difference

SEND FRAME

O comando *Send Frame*, "SF", permite enviar um *frame* para o computador (exterior) pela porta série. Este é o único que, por omissão, enviará pacotes de caracteres ASCII não visíveis. Este envia dados de vídeo não processados, linha por linha, com um *byte* sincronização por *frame* e um *byte* de sincronização por coluna. Estes dados podem ser lidos e visualizados pela aplicação CMUcam2GUI. Opcionalmente, pode-se adicionar um canal (0-2) ao comando que provoca o envio do *frame*, apenas enviando o canal desejado, assim apenas um terço de informação é enviado. Pode-se ver na Figura 154, um exemplo que permite perceber melhor o funcionamento deste comando e o seu formato de resposta.

Type F data packet format flags:
1 - new frame followed by X size and Y size
2- new col
3 - end of frame
RGB (CrYCb) ranges from 16-240
1 xSize ySize 2 r g b r g b ... r g b r g b 2 r g b r g b ... r g b r g b 3

Figura 154 Exemplo demonstrativo do comando "SF".

DOWN SAMPLE

O comando *Down Sample*, representado por "DS", permite a aquisição da imagem que está a ser processada. Um *x_factor* de "1" (valor por omissão) significa que não existiu uma mudança na resolução horizontal. Um *x_factor* de "2", indica que a resolução horizontal

foi efectivamente dividida. Todos os comandos, como *Send Frame* e *Track Color*, funcionam com uma resolução baixa na aquisição. Este comando permite um aumento da velocidade, uma diminuição da quantidade de dados a serem enviados no *Send Frame* e o modo de linhas *bitmap* (*bitmap linemodes*) não fará o corte da imagem como a janela virtual faria. O factor *y_factor* controlará de forma independente a resolução vertical, à semelhança do factor *x_factor*.

VIRTUAL WINDOW

Este comando, representado por "VW" configura o tamanho da janela virtual da câmara, através das coordenadas do canto superior esquerdo, seguidas das coordenadas do canto inferior direito. A origem está localizada na zona superior esquerda da imagem. Este comando pode ser chamado antes dos comandos de processamento de imagem para restringir o campo de visão. Ao reduzir o tamanho vertical da janela, pode-se melhorar o tempo de processamento para realizar tarefas a uma taxa mais elevada, com os comandos de detecção.

FRAME STREAM

Este comando, é representado por "FS", configura o modo *Frame Streaming* da câmara. O valor "1" habilita o *Frame Streaming*, enquanto o valor "0" (valor por omissão) o desactiva. Quando este modo está activo, um comando *Send Frame* irá continuamente retornar *frames* através da porta série.

HIRES MODE

O comando que configura a câmara num modo de alta resolução é representado por "HR", e apenas está disponível usando o módulo da câmara "OV6620". Com o valor "0" (valor por omissão) disponibiliza a resolução de 88x143 pixéis, enquanto o valor "1" fornece 176x287 pixéis. O modo de Alta resolução trunca a imagem a 176x255 pixéis, por isso o valor não ultrapassa os 8 *bits*.

GET WINDOW

Este comando é representado por "GW" e permite conhecer o tamanho actual da janela virtual através do retorno das coordenadas que indicam o limite da janela actual.

PIXEL DIFFERENCE

O comando *Pixel Diference*, "PD", permite activar o modo de diferença entre pixéis, que está desligado por omissão. O valor "1" permite diferenciar entre o pixel actual e o anterior para se utilizar em todos os comandos de processamento em vez do valor original do pixel. Essencialmente, isto corresponde a fazer uma detecção por flanco horizontal na imagem.

B.2.5. COMANDOS PARA A DETECÇÃO DE COR

Esta categoria engloba os comandos apresentados na Tabela 21.

Tabela 21 Comandos para a detecção da cor.

TC [Rmin Rmax Gmin Gmax Bmin Bmax] \r	Track Color
TI boolean \r	Track Inverted
TW \r	Track Window
NF threshold \r	Noise Filter
LM type mode \r	Line Mode
GT \r	Get Tracking Parameters
ST Rmin Rmax Gmin Gmax Bmin Bmax \r	Set Tracking Parameters

TRACK COLOR

Este comando permite iniciar a detecção da cor, adquirindo num intervalo de valores das componentes RGB e retornando pacotes do tipo T. Este pacote (por omissão) retorna, as coordenadas do centro de massa e pode ser mascarado com o comando "OM". Convém referir que os valores das cores vão variam entre 16 e 240.

TRACK INVERTED

Este comando, "TI", activa o modo de detecção invertida. Quando está activo, a câmara detectará as cores que estão fora da gama de cores definidas pelo utilizador.

TRACK WINDOW

O comando *Track Window*, "TW", irá detectar a cor encontrada na região central da janela actual. Depois da a cor ter sido capturada, as funções para a sua detecção são chamadas com aqueles parâmetros e com o tamanho total da janela.

NOISE FILTER

Este comando, "NF", permite configurar o filtro de ruído. Este aceita valores que determinam quantos pixéis activos consecutivos devem existir, antes do pixel actual, para que o pixel seja detectado. Os valores variam entre "0" e "255".

LINE MODE

Este comando habilita o *Line Mode* que transmite dados mais detalhados sobre a imagem, adicionando dados como prefixos em pacotes do tipo T ou S. Este modo destina-se a utilizadores que pretendam fazer um processamento de imagem mais complexo. Devido ao aumento da taxa de dados, este modo pode não ser adequado para processadores mais lentos. Os diferentes tipos e modos que o *Line Mode* aplica às diferentes funções de processamento, são apresentados na Tabela 22.

Para informação mais detalhada sobre os tipos e modos de detecção fornecidos, pode-se consultar o *User Guide*, disponível através da hiperligação www.cs.cmu.edu/~cmucam/cmucam2/CMUcam2_manual.pdf.

Tabela 22 Tipos e modos associados ao comando "LM".

Tipo	Modo	Comando efectivo	Descrição
0	0	TC TW	Por omissão, onde o *line mode* está inactivo.
0	1	TC TW	Envia uma imagem binária dos pixéis a serem detectados.
0	2	TC TW	Envia a média, o mínimo, o máximo, a confiança e a contagem para cada linha horizontal da imagem processada.
1	0	GM	Por omissão, o *line mode* não está activo.
1	1	GM	Envia os valores médios para cada linha da imagem.
1	2	GM	Envia os valores médios e os desvios de cada uma das linhas que estão a ser processadas.
2	0	FD	Por omissão, o *line mode* não está habilitado.
2	1	FD	Retorna o *bitmap* dos pixéis detectados mais parecidos com a detecção da cor do tipo 0, modo 0.
2	2	FD	Envia a diferença entre os valores dos pixéis da imagem actual e da imagem guardada. Este fornece um *frame* diferencial das imagens analisadas.
2	3	LF FD	Fornece o actual valor médio para cada elemento no *frame* diferencial, e retorna estes valores quando for carregado um novo *frame*.

GET TRACKING PARAMETERS

Este comando é representado por "GT" e permite saber quais são os valores actuais para a detecção da cor. Este comando é útil porque permite saber quais os parâmetros que estão a ser usados.

SET TRACKING PARAMETERS

O comando "ST" permite configurar os parâmetros de detecção sem chamar o comando *Track Color*. Estes valores podem ser guardados até se poder chamar o comando "TC", sem acrescentar argumentos ao mesmo.

B.2.6. COMANDOS PARA HISTOGRAMAS

Na Tabela 23 são apresentados os comandos referentes a esta categoria.

Tabela 23 Comandos para os Histogramas.

GH <channel> \r	**Get Histogram**
HC #_of_bins scale \r	Histogram Config
HT boolean \r	Histogram Track

GET HISTOGRAM

O comando *Get Histogram,* "GH", adquire o histograma de um canal especificado pelo utilizador. O histograma contém 28 barras que indicam o número de pixéis que ocorreram dentro da gama delimitada por cada barra. Por exemplo, a barra 0 do canal 0 contém o número de pixéis do canal vermelho que se situam entre os valores 16 e 23. Se não forem adicionados argumentos, este comando utilizará o último canal seleccionado. Se for a primeira vez que este comando é chamado, e não possuir nenhum argumento, será usado o canal verde.

HISTOGRAM CONFIG

O comando *Histogram Config,* "HC", permite configurar os parâmetros do histograma. O primeiro parâmetro permite três valores distintos, em que cada um indicará o número de barras utilizadas. O valor "0" (valor de omissão), fará com que, em resposta ao comando "GH", surjam 28 barras de saída. O valor "1" selecciona a saída para 14 barras e o valor "2" para 7 barras. O segundo argumento do comando é a escala.

HISTOGRAM TRACK

Este comando, é identificado por "HT", e activa ou desactiva a detecção por histograma. Se a detecção por histograma estiver activa, apenas os valores que se situam dentro dos limites de detecção serão mostrados nos histogramas. Isto permite seleccionar as gamas de cor exactas, fornecendo mais detalhes, e ignorando outras influências externas. O valor "0" (valor por omissão), desactiva a detecção por histograma, enquanto o valor "1"a activa.

B.2.7. COMANDO PARA A DIFERENCIAÇÃO DE FRAMES

Na Tabela 24 são apresentados os comandos que estão associados a esta categoria. De referir que o comando *Line Mode* já foi abordado na secção B.2.5.

Tabela 24 Comandos para a diferenciação de *Frames.*

FD threshold \r	Frame Difference
DC value \r	Difference channel
LF \r	Load Frame
MD threshold \r	Mask Difference
UD <64 raw bytes> \r	Upload Difference
HD boolean \r	HiRes Difference
LM type mode \r	Line Mode

FRAME DIFFERENCE

O comando *Frame Difference*,"FD", faz a diferenciação do *frame* com o último *frame* lido usando o comando "LF". Este retorna um pacote do tipo T que contém o centro de massa, os limites da caixa, contagem de pixéis e a confiança de qualquer mudança desde a última leitura de um *frame*. Este comando faz isto através do cálculo da média da intensidade da cor de uma grelha de 8x8, com 64 regiões na imagem, e compara estas com o valor definido como *threshold*.

DIFFERENCE CHANNEL

Este comando, "DC", é usado para comados de diferenciação de *frames* onde se selecciona o canal pretendido. O valor "0" configura os comandos "LF" e "FD" para usar o canal vermelho. O valor "1" (por omissão), selecciona o canal verde e o valor "2" o canal azul.

LOAD FRAME

Este comando é representado por “LF” e permite carregar um novo *frame* para a memória, para se efectuar a diferenciação. Este simplesmente carrega uma imagem base para diferenciação e detecção de movimento, não estando relacionado com o *buffer* da câmara.

MASK DIFFERENCE

Este comando é representado por “MD” e é idêntico ao FD excepto que este mascara o primeiro *frame* que diferencia. Qualquer movimento detectado no primeiro *frame* é mascarado, logo as áreas com maior quantidade de ruído são ignoradas. Basicamente, se se chamar a diferenciação de *frames* existe sempre uma área do *frame* que se move; o comando “MD” vai mascarar essa porção da imagem de modo a que futuras chamadas do comando “FD” irão também ignorar essa porção da imagem. Chamando o comando “LF” limpa-se o mascaramento de pixéis.

UPLOAD DIFFERENCE

O comando *Upload Difference*, “UD”, permite carregar o *buffer* para diferenciação de *frames.* O comando espera por valores, no total de 64 bytes de dados não processados, que preenche numa diferenciação interna de *frames* de 8 x 8. Um “\r” cancela a transferência. O valor “0” indica que região deve ser mascarada e não detectar movimento. Com este comando, em combinação como o *Line Mode type 2*, é possível fazer o *download* e o *upload* de diferentes *frames* de referência para a diferenciação de *frames*.

HIRES DIFFERENCE

Este comando, é representado por “HD”, activa ou desactiva o modo de diferenciação com alta resolução. O valor de omissão é o “0”, que desactiva este modo, enquanto o valor “1” o activa. Quando o modo de configuração com alta resolução está activo funciona com uma grelha de 16x16, em vez da grelha 8x8 quando está inactivo. A resolução extra é conseguida através de quatro pequenas comparações de cada pixel guardado internamente.

B.2.8. COMANDOS DE ESTATÍSTICAS DE COR

Os comandos englobados nesta categoria são apresentados na Tabela 25. De referir que o comando *Line Mode* já foi abordado na secção B.2.5.

Tabela 25 Comandos para as estatísticas de cor.

GM \r	Get Mean
LM type mode \r	Line Mode

GET MEAN

O comando *Get Mean,* "GM", calcula o valor médio da imagem actual. Se, opcionalmente, uma sub-região da imagem é seleccionada por via da janela virtual, esta função irá apenas operar na região seleccionada. Os valores médios encontram-se entre 16 e 240 assim como os limites de cada canal de cor. Este comando retorna também uma medida do desvio médio absoluto da cor encontrada nessa região.

B.2.9. COMANDOS AO NÍVEL DO SISTEMA

Na Tabela 26 são apresentados os comandos ao nível do sistema.

Tabela 26 Comandos ao nível do sistema.

SD \r	Sleep Deeply
SL active \r	Sleep
RS \r	Reset
GV \r	Get Version

SLEEP DEEPLY

Este comando, representado por "SD", possibilita diminuir o consumo de energia do sistema, pondo o processador no estado de *sleep,* tal como o comando "SL" (*sleep*). Pode-se acordar o sistema enviando qualquer caracter para o módulo.

SLEEP

Este comando, “SL”, activa o modo *sleep* pondo o processador no modo *sleep.* Para “acordar” o sistema basta o envio para o mesmo de um qualquer caracter, usualmente “\r”.

RESET

Este comando é representado por “RS” e reinicializa o sistema. Numa reinicialização o primeiro caracter é “\r”.

GET VERSION

Este comando, representado por “GV”, permite saber qual a versão actual do *Firmware* e a versão do módulo da câmara, retornando um ACK (*Acknowledge*) seguido da *string* da versão do *Firmware.* Nessa string, c6 significa que reconhece o modelo “OV6620” e o código c7 significa que o modelo detectado é “OV7620”.

Nesta secção foram abordadas apenas as questões que se consideram mais importantes para facilitar a aprendizagem das funcionalidades deste dispositivo.

Anexo C. *Layout* das Placas Desenvolvidas

C.1. PLACA – CONTROLO DE VELOCIDADE (*TOP* E *BOTTOM*)

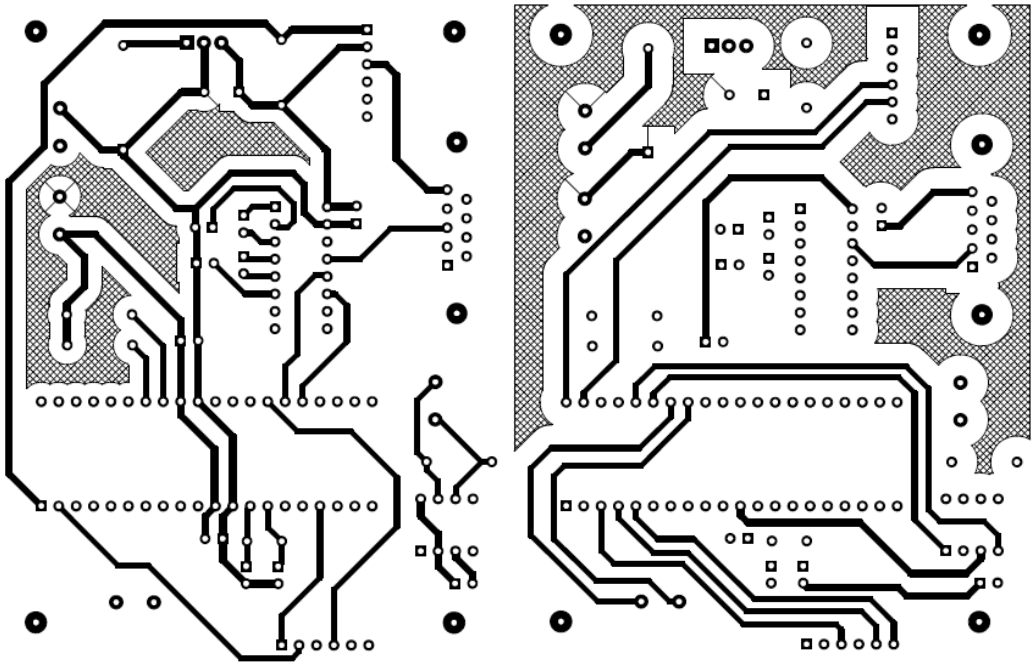


Figura 155 ***Layout*** **das placas,** ***Top*** **à esquerda e** ***Bottom*** **à direita.**

C.2. PLACA – AQUISIÇÃO DE DADOS DOS SENSORES (*TOP* E *BOTTOM*)

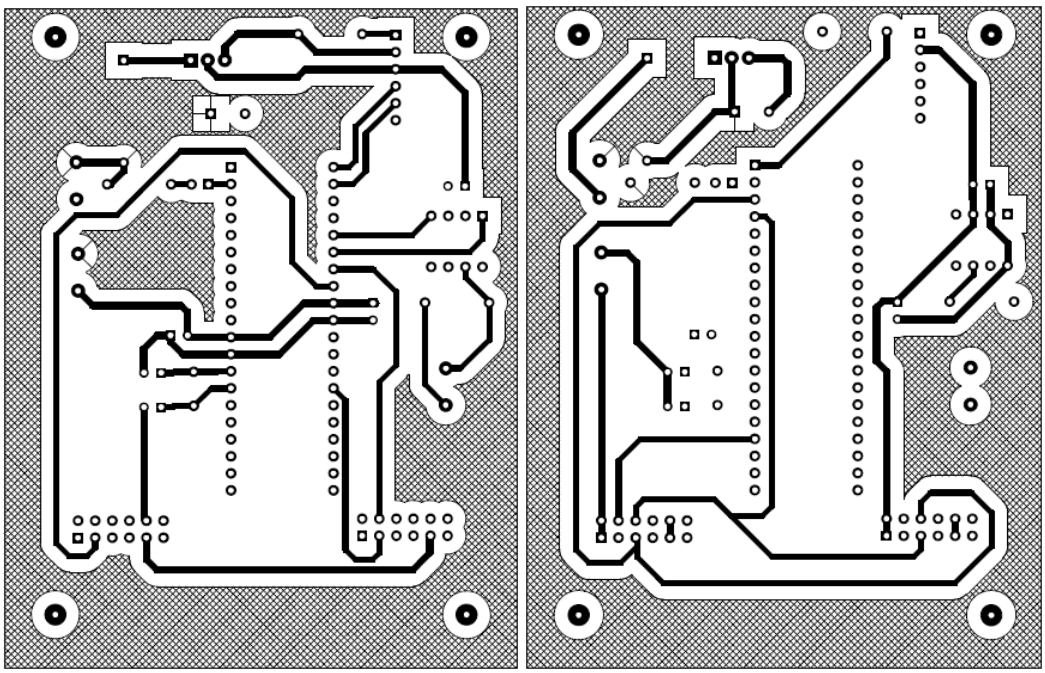


Figura 156 ***Layout*** **das placas,** ***Top*** **à esquerda e** ***Bottom*** **à direita.**

C.3. PLACA – CONTROLADOR DIFUSO (*TOP* E *BOTTOM*)

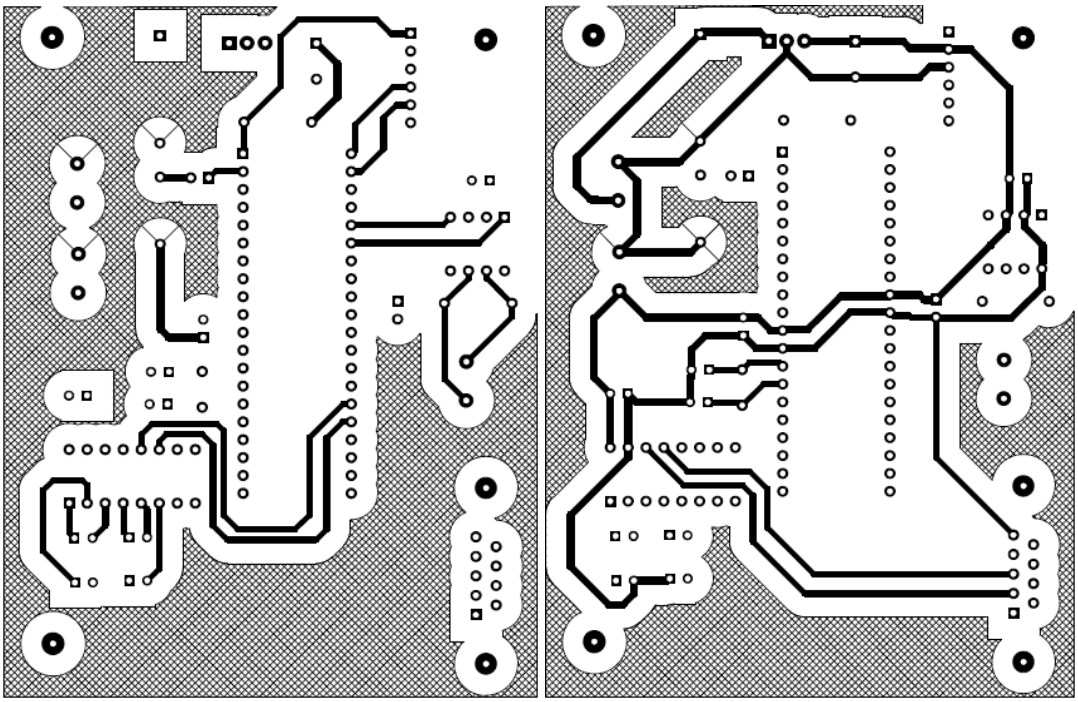


Figura 157 ***Layout*** **das placas,** ***Top*** **à esquerda e** ***Bottom*** **à direita.**