DESENVOLVIMENTO DE FUNÇÕES DE LÓGICA DIFUSA PARA PLC

Maurício Armada Martins

Mestrado em Engenharia Electrotécnica e de Computadores
Área de Especialização de Automação e Sistemas
Departamento de Engenharia Electrotécnica
Instituto Superior de Engenharia do Porto
2014

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

Candidato: Maurício Armada Martins, Nº 1020118, 1020118@isep.ipp.pt
Orientação científica: Ramiro de Sousa Barbosa, rsb@isep.ipp.pt
Coorientação científica: Lino B. Figueiredo, lbf@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
21 de novembro de 2014

Aos meus Pais…

Agradecimentos

A elaboração de um trabalho como este exige sempre dedicação e entrega, o que impossibilita de vivermos a nossa vida do dia-a-dia como se gostaria. Por isso gostaria de agradecer, em primeiro lugar à minha esposa que sempre me apoiou em todos os momentos e em segundo ao meu filho, de dois anos de idade, que me apoia, mesmo sem saber, com o seu sorriso e boa disposição todos os dias!

Ao Professor Ramiro Barbosa pela ajuda e orientação ao longo destes meses, principalmente nesta fase final com um impulso decisivo para concluir a tese e o mestrado.

À INESE que colaborou fornecendo os materiais necessários para que fosse possível efetuar a validação dos resultados

Resumo

Este trabalho pretende preencher uma lacuna no controlo difuso em todos os autómatos/controladores que usam a plataforma Codesys para a sua programação.

Começando por um levantamento histórico e a respetiva compreensão do que é a lógica difusa até à análise do que existe no mercado nos dias de hoje.

Será realizada uma abordagem do que é a plataforma Codesys e a utilização da norma IEC61131-3. Sendo efetuada também uma análise da norma IEC61131-7 que explica como deve ser realizado o controlo difuso em PLC.

Para colmatar a lacuna existente foi desenvolvido um *software* tendo por base a plataforma Codesys e validado e testado com o *software* SoMachine da Schneider Electric. Esse *software* será devidamente descrito para ser entendido de uma forma fácil.

Palavras-Chave

Desenvolvimento, Autómatos, PLC, Fuzzy Logic, Lógica Difusa, Omron, Siemens, Allen Bradley, Schneider Electric, SoMachine, Codesys.

Abstract

This thesis aims to fill a diffuse control gap in all PLC/controllers that use the CoDeSys programming platform.

Beginning with a historical survey and the respective understanding what is fuzzy logic to analyze what exists in the market today.

An approach of what is the CoDeSys platform and the use of standard IEC61131-3 is performed. Will be made an analysis of the IEC61131-7 standard that explains how programming should be done in the fuzzy control PLCs.

To fill the gap, a software was developed based on the CoDeSys platform and validated and tested with SoMachine software of Schneider Electric. This software will be fully described to be understood in an easy way.

Keywords

Desenvelopment, PLC, Fuzzy Logic, Omron, Siemens, Allen Bradley, Schneider Electric, SoMachine, Codesys.

Índice

Índice de Figuras

Índice de Tabelas

Acrónimos

FB – *Blocos de Função (Function Blocks)*

IEC – *Comissão Eletrotécnica Internacional (International Electrotechnical Commission)*

CNC – *Controladores de Controlo Numérico*

SFC – *Controlo sequencial de funções (Sequencial Function Control)*

FBD – *Diagrama de blocos de função (Function Block Diagram)*

LD – *Diagrama Ladder (Ladder Diagrams)*

I/O – *Entradas/Saídas (Inputs/Ouputs)*

CFC – *Gráfico de sequência de funções (Continuous Function Chart)*

FCL – *Linguagem de controlo difuso (Fuzzy Control Language)*

IL – *Lista de Instruções (Instrution List)*

NL – *Negativo Grande (Negative Large)*

NS – *Negativo Pequeno (Negative Small)*

PL – *Positivo Grande (Positive Large)*

PS – *Positivo Pequeno (Positive Small)*

PDO – *Process data object (PDO)*

PLC – *Programmable Logic Controller*

P – *Proporcional*

PD – *Proporcional e Derivativo*

PI – *Proporcional e Integrativo*

PID – *Proporcional, Integrativo e Derivativo*

SE – *Schneider Electric*

ST – *Texto estruturado (Structured Text)*

VNA – *Velocidade Negativa Alta*

VNB – *Velocidade Negativa Baixa*

VN – *Velocidade Nula*

VPA – *Velocidade Positiva Alta*

VPB – *Velocidade Positiva Baixa*

ZR – *Zero*

1. INTRODUÇÃO

O controlo difuso é cada vez mais usado para resolver situações em que os métodos de controlo clássico não conseguem resultados satisfatórios. Na indústria a maior parte das máquinas são controladas por Controladores lógicos programáveis (PLC) (*Programmable Logic Controller*). A maioria dos PLC que estão no mercado atualmente apenas permitem usar os métodos clássicos de controlo, como por exemplo P, PI, PD, PID. Estes controladores usam as componentes P (Proporcional), I (Integrativa) e D (Derivativa). Com base nesses pressupostos surgiu a necessidade de perceber o grau de evolução do controlo com lógica difusa neste tipo de controladores.

CoDeSys é uma plataforma de *software* especialmente desenvolvida para atender às mais diversas necessidades dos projetos modernos de automação industrial. O desenvolvimento de sistemas com a norma IEC 61131-3 é o coração do CoDeSys. Oferece soluções integradas e fáceis de usar, para apoiar o utilizador no desenvolvimento das suas tarefas.

A plataforma CoDeSys apresenta uma lacuna que consiste em não ter nenhuma biblioteca de controlo difuso desenvolvida de raiz, ou desenvolvida pela comunidade que o utiliza. A gama de produtos que nos dias de hoje usam a plataforma CoDeSys é muito abrangente, uma vez que é usada para PLC de gama baixa assim como PLC de gamas mais altas.

1.1. CONTEXTUALIZAÇÃO

Esta tese surgiu do desejo de realizar um trabalho no âmbito do controlo difuso, com aplicação prática na área da automação industrial.

Face à enorme presença da automação na indústria e face à pouca utilização que o controlo difuso tem, no que respeita ao controlo industrial, surgiu a necessidade de desenvolver uma forma de efetuar este tipo de controlo numa plataforma cada vez mais utilizada.

Outro fator que impulsionou o desenvolvimento da biblioteca de controlo difuso, foi tornar o controlo difuso disponível para ser utilizado em qualquer autómato que seja programado com base na norma IEC61131-3. O controlo difuso, à data, só está disponível em alguns fabricantes, nas suas gamas altas de PLC e sempre com um elevado custo monetário associado.

Na Figura 1 pode-se observar a particularidade da abrangência do CoDeSys no que à automação industrial diz respeito. O CoDeSys permite a programação de PLC com base na norma IEC 61131-3, a programação de sistemas de interface homem-máquina, programação de PLC de segurança, gestão de redes de campo e programação de sistemas de movimentos sincronizados e controladores de controlo numérico (CNC). Atualmente o CoDeSys é usado por mais de 200 fabricantes

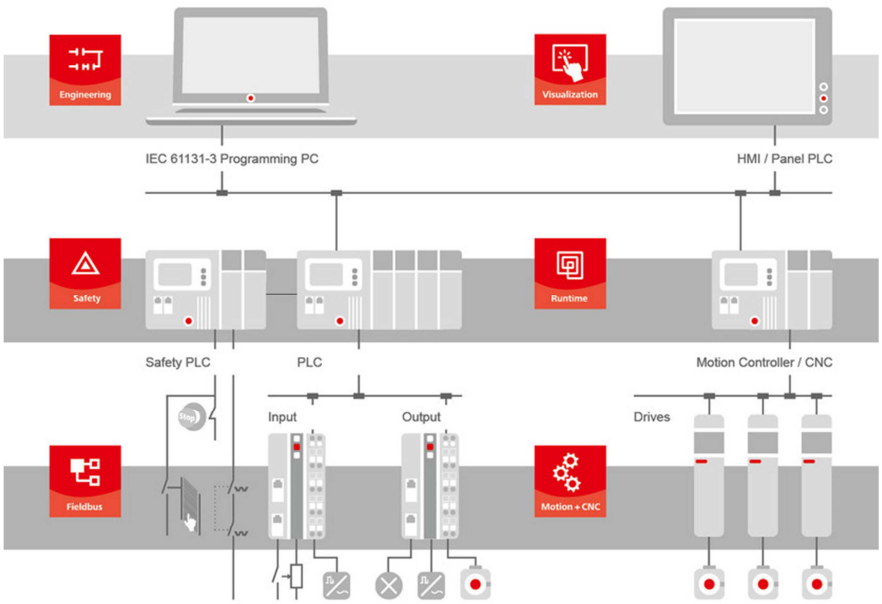


Figura 1 – Arquitetura CoDeSys [12]

1.2. OBJETIVOS

O objetivo principal desta tese é o desenvolvimento de uma biblioteca de controlo difuso com validação prática. Depois de valida, publicar a biblioteca junto das comunidades de utilizadores de CoDeSys. Face a estes objetivos houve a necessidade de dividir a tarefa em subtarefas, tais como:

- Elaboração relatório teórico
    - Estudo da história e estado atual do controlo difuso
    - Estudo da plataforma CoDeSys
    - Estudo da norma IEC 61131-7 Programação Controlo difuso
- Desenvolvimento da biblioteca de controlo difuso
- Testes em simulação *offline*
- Testes *online* com sistema real
- Elaboração do manual da biblioteca de controlo difuso

1.3. CALENDARIZAÇÃO

Tendo por base os objetivos propostos e a necessidade de os dividir em subtarefas, foi necessária a calendarização das mesmas, apresentada na Figura 2.

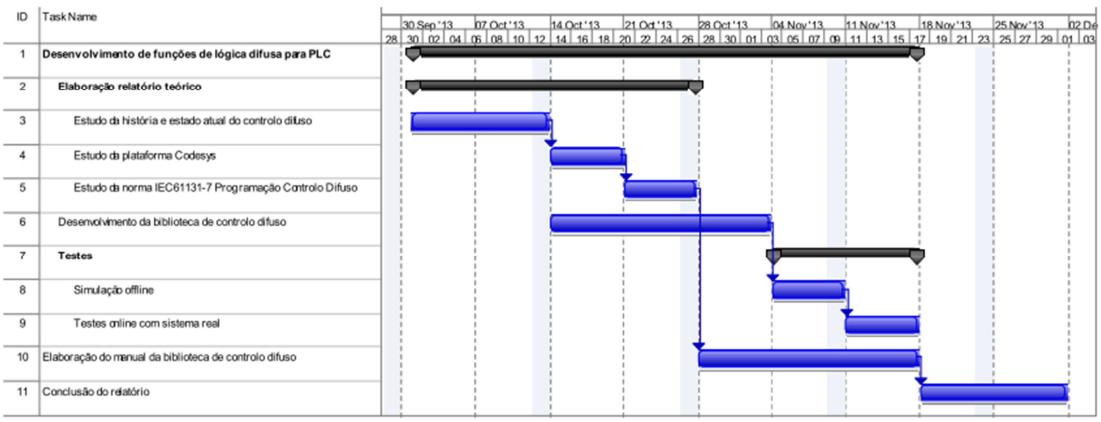


Figura 2 – Calendarização do projeto

1.4. ORGANIZAÇÃO DO RELATÓRIO

No Capítulo 1 é efetuada uma introdução estrutural e dos conteúdos deste trabalho. No capítulo seguinte, 2, são apresentados os desenvolvimentos históricos assim como os fundamentos do controlo difuso. No capítulo 3 serão descritas as principais soluções existentes ao nível do controlo difuso nos PLC de alguns dos principais fabricantes da atualidade. No capítulo 4 será analisada a norma IEC 61131-7 referente à programação de sistemas difusos. No capítulo 5 será descrita a plataforma CoDeSys, abordada a biblioteca criada com o respetivo descritivo de todas as funções implementadas (manual de ajuda). No capítulo 6 serão demonstrados os testes e respetivos resultados obtidos. No último capítulo, o 7º, são reunidas as principais conclusões e perspetivados futuros desenvolvimentos.

2. FUNDAMENTOS DO CONTROLO DIFUSO

Este capítulo é destinado à análise do estado da arte nos diversos fabricantes de autómatos existentes no mercado. Começando por uma abordagem sucinta do que é o controlo difuso.

A lógica difusa fornece aos PLC a capacidade de tomar decisões "racionais" sobre um processo. Neste capítulo, serão abordados os princípios da lógica difusa, incluindo conceitos fundamentais e origens históricas. Será demonstrado como a lógica difusa pode ser usada em aplicações práticas para fornecer em tempo real, o controlo lógico difuso de um processo.

2.1. HISTÓRIA DO CONTROLO DIFUSO

A lógica difusa existe desde os tempos antigos, quando Aristóteles desenvolveu a lei do terceiro excluído. Nesta lei, Aristóteles apontou que termo intermédio é perdido na arte do raciocínio lógico, as afirmações são verdadeiras ou falsas, nunca no meio. Quando os PLC foram desenvolvidos, a sua lógica discreta baseou-se nas técnicas de raciocínio antigos. Assim, as entradas e saídas poderiam pertencer a apenas um conjunto (ou seja, ON ou OFF), todos os outros valores foram excluídos. A lógica difusa quebra a lei do terceiro

excluído nos PLC, permitindo que elementos possam pertencer a mais do que apenas um conjunto.

As origens da lógica difusa datam para o início do século XX quando Bertrand Russell descobriu um antigo paradoxo grego que afirma:

- Um cretense assevera que todos os cretenses mentem. Então, ele está a mentir? Se ele está, então ele está a dizer a verdade e não mente. Se ele não está, então ele diz a verdade e, portanto, ele mente.

Em ambos os casos, em que todos os cretenses mentem ou que todos os cretenses não mentem, existe uma contradição, porque ambas as afirmações são verdadeiras e falsas. Russell descobriu que este mesmo paradoxo é aplicado à teoria dos conjuntos usados na lógica discreta. As declarações devem ser totalmente verdadeiras ou totalmente falsas, levando a áreas de contradição.

A lógica difusa superou este problema na lógica clássica, permitindo que as declarações devem ser interpretadas como verdadeiras e como falsas. Portanto, a aplicação da lógica difusa ao paradoxo grego produz uma declaração de que é ao mesmo tempo verdadeiro e falso: cretenses dizem a verdade 50% do tempo e mentira 50% do tempo. Esta interpretação é muito semelhante à ideia de um copo de água está meio cheio ou meio vazio. Na lógica difusa o copo está ambos, 50% cheio e 50% vazio. Mesmo quando a quantidade de água diminui, o copo ainda mantém percentagens de ambas as condições.

Por volta da década de 1920, independente de Bertrand Russell, um lógico polaco chamado Jan Lukasiewicz começou a trabalhar na lógica de valores múltiplos, que criou valores binários fracionários entre lógico 1 e o valor lógico 0. Num artigo de 1937 em Philosophy of Science, Max Black, um filósofo quântico, aplicando esta lógica de valores múltiplos a listas (ou conjuntos) desenhou o primeiro conjunto de curvas difusos, chamando-os conjuntos vagos. Vinte e oito anos mais tarde, Dr. Lofti Zadeh, o presidente do Departamento de Engenharia Eletrotécnica da Universidade da Califórnia, em Berkeley, publicou um documento marco intitulado "Fuzzy Sets", que deu o nome ao campo da "fuzzy logic" (lógica difusa). Neste artigo, o Dr. Zadeh aplicou a lógica de Lukasiewicz a todos os objetos, num conjunto que resultou numa álgebra completa para conjuntos difusos. Face a este trabalho pioneiro, o Dr. Zadeh é considerado como o pai da lógica difusa moderna [3].

Por volta do ano 1975, Ebrahim Mamdani e S. Assilian do Queen Mary College, da Universidade de Londres (Inglaterra) publicaram um artigo intitulado "Uma Experiência em síntese linguística com um controlador Lógica Difusa", onde a viabilidade de controlo lógico difuso foi comprovado através da aplicação de controlo difuso a um motor a vapor. Desde então o termo lógica difusa passou a significar o raciocínio matemático ou computacional que utiliza conjuntos difusos [1] [2].

2.2. INTRODUÇÃO AO CONTROLO DIFUSO (*FUZZY LOGIC*)

A lógica difusa é um ramo da inteligência artificial que lida com algoritmos de raciocínio usado para emular o pensamento humano e tomar decisões em máquinas. Estes algoritmos são utilizados em aplicações onde os dados do processo não podem ser representados em forma binária. Por exemplo, as declarações "o ar parece fresco" e "ele é jovem" não são declarações discretas. As declarações não fornecem dados concretos sobre a temperatura do ar ou a idade da pessoa (ou seja, a temperatura do ar é de 18ºC ou o menino tem 12 anos de idade). A lógica difusa interpreta declarações vagas como estas para que elas façam sentido lógico. No caso do ar fresco, um PLC com capacidades de lógica difusa iria interpretar tanto o nível de frieza e a sua relação com o calor para verificar se "fresco" significa algo entre quente e frio. Na lógica binária direta, quente seria um valor discreto (por exemplo, valor lógico 1) e frio seria o outro (por exemplo, valor lógico 0), não deixando nenhum valor para representar uma temperatura fresca (ver Figura 3).

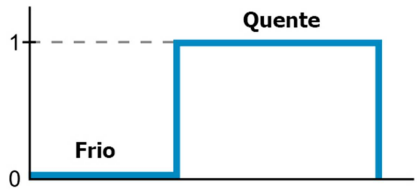


Figura 3 - Representação binária lógica de um valor de temperatura discreta

Em contraste com a lógica binária, a lógica difusa pode ser pensada como lógica de cinzento, o que gera uma forma de expressar entre os valores de 0 e 1. Lógica difusa associa nível, com um intervalo de dados, dando-lhe um valor de 1 no seu máximo e 0 no seu mínimo. Por exemplo, a Figura 4(a) ilustra uma representação de uma gama de temperatura de ar fresco, onde 21ºC indica perfeitamente ar fresco (isto é, um valor de 1). Qualquer temperatura acima de 26ºC é considerada quente, ou qualquer temperatura inferior a 15ºC é considerada fria. Assim, a temperaturas acima de 26ºC e abaixo de 15ºC

tem um valor de 0 para o fator fresco, o que significa que não são frescas. A Figura 4 (b) mostra uma outra representação da faixa de temperatura fresca, onde a linha tracejada (diagonal) mostra que as temperaturas quentes e frias são não frescas. Aos 18ºC, o algoritmo de lógica difusa considera que a temperatura deve ser de 50% fresca e 50% frio, o que indica um nível de frescura. Abaixo de 15ºC, o algoritmo de lógica difusa considera a temperatura como frio.

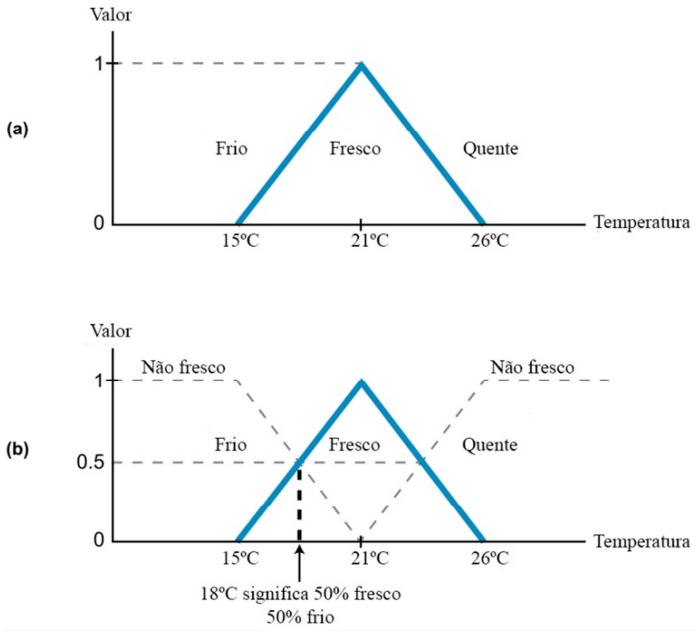


Figura 4 - (a) Variação da temperatura fresca do ar com (b) variação da temperatura não fresca

Na vida real, esse algoritmo da temperatura lógica difusa pode ser associado com a decisão a tomar sobre o tipo de roupas a vestir em diferentes épocas do ano. O tipo de vestuário é baseado na temperatura (de entrada) e o seu grau de representação. Como mostrado na Figura 5, a 21ºC, pode ser necessário apenas uma camisa de mangas curtas e calças. No entanto, se a temperatura cair para 18ºC, você pode optar por usar uma camisa de manga comprida, em vez de uma de mangas curtas. Além disso, se a entrada for 25% fresco e 75% frio (17ºC), então você pode decidir adicionar outra camada, uma camisola, com base na temperatura e seu valor de frescura. Como será explicado mais tarde, a saída de um sistema difuso pode ser baseado em várias entradas, não apenas numa, como a temperatura. Nesta situação, a saída de decisão é feita utilizando a base de conhecimento representado no gráfico lógica difusa.

Lógica difusa exige conhecimento, a fim de raciocinar. Esse conhecimento, que é fornecido por uma pessoa que conhece o processo ou máquina (o especialista), é armazenado no sistema difuso.

```
IF temperatura 21ºC (Valor 1-100% fresco),
THEN usar camisa de mangas curtas e calças

IF temperatura 18ºC (0,5 frio, 0,5 fresco),
THEN usar camisa de mangas compridas e calças compridas

IF a temperatura é de 17ºC (0,25 fresco, 0,75 frio),
THEN usar camisa de mangas compridas com uma camisola e
calças compridas
```

Por exemplo, se a temperatura subir num sistema de temperatura variável, o especialista pode dizer que a válvula de vapor precisa ser rodada "um pouco" no sentido horário. Um sistema difuso pode interpretar essa expressão como uma rotação de 10 graus no sentido horário que fecha a abertura atual da válvula em 5%. Como o nome indica, tal como a descrição "um pouco" é uma descrição difusa, o que significa que não tem um valor definido.

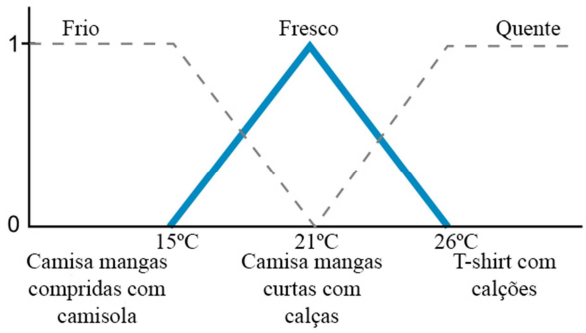


Figura 5 – Gráfico de lógica difusa, ilustrativo da escolha de roupa baseada na temperatura

2.3. SISTEMAS DE CONTROLO DIFUSO

A Figura 6 ilustra um sistema lógico de controlo difuso. A entrada para o sistema difuso é o resultado do processo, que é introduzido no sistema através de interfaces de entrada no sistema.

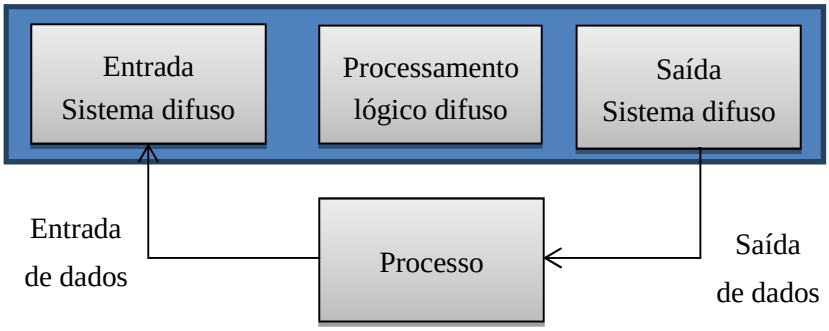


Figura 6- Sistema lógico de controlo difuso

Por exemplo, numa aplicação de controlo da temperatura, a entrada de dados podia ser efetuada através de um módulo analógico de entrada. Esta informação de entrada, então, passa pelo processo de lógica difusa, onde o processador iria analisar os dados para obter uma saída. O processamento difuso envolve a execução de regras IF - THEN, as quais são baseadas nas condições de entrada.

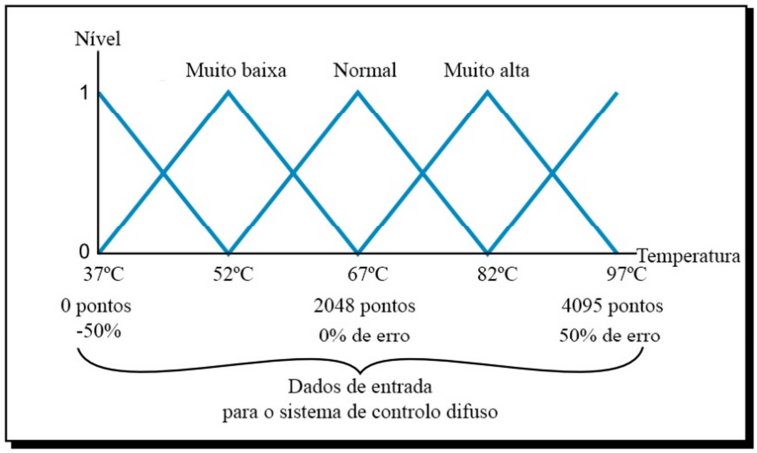


Figura 7 – Dados de entrada para um sistema de lógica difusa representado como pontos de resolução e percentagens

O nível de uma entrada especifica o quão bem ele se encaixa num determinado conjunto gráfico (por exemplo, muito baixa, normal, muito alta). Note-se que os dados de entrada, como mostrado na Figura 7, também podem ser representados como um valor de pontos que varia 0-4095 (amostragem com 12 bits de resolução) ou como uma percentagem de desvios de erro. Se o sistema utiliza uma lógica difusa de entrada analógica, que tem um intervalo de pontos 0-4095, os gráficos que representam a entrada vão cobrir o intervalo 0-4095 pontos. Além disso, a informação de entrada analógica (0-4095 pontos) pode representar um intervalo de erro, de -50% a 50%, de um processo.

A saída de um controlador difuso é também definida pelos níveis, com a classificação a determinar o valor de saída apropriado para o elemento de controlo. A saída do sistema de lógica difusa na Figura 8, por exemplo, controla uma válvula de vapor, que abre ou fecha de acordo com o seu nível no gráfico de saída.

A Figura 9 representa um sistema lógico difuso com representação gráfica simultânea dos níveis de entrada e saída, em que o eixo horizontal representa a condição de entrada (temperatura) e o eixo vertical representa a saída (velocidade do motor de ar condicionado).

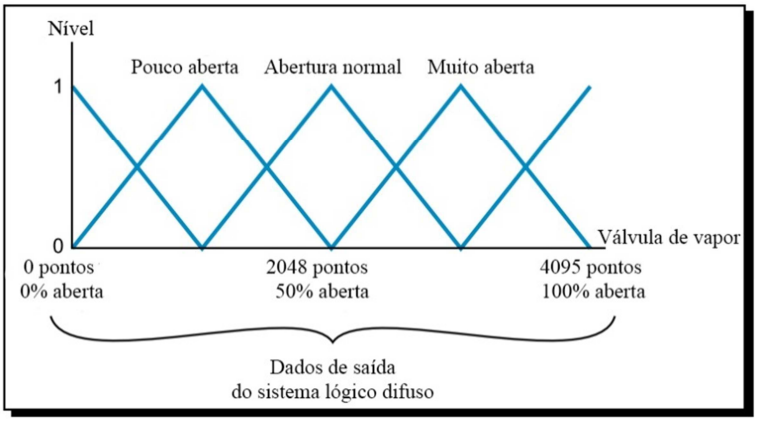


Figura 8 – Dados de saída do sistema lógico difuso representada sobre a forma de percentagens e pontos de resolução

Neste gráfico, uma única entrada pode acionar mais de uma condição de saída. Por exemplo, se a temperatura de entrada é de 59ºC, a temperatura faz parte de duas curvas de entrada, é 50% muito fresco e 50% normal. Consequentemente, a entrada irá acionar duas saídas, a condição de entrada muito fresco irá acionar uma saída menos velocidade, enquanto a entrada normal irá acionar uma condição de saída de velocidade normal.

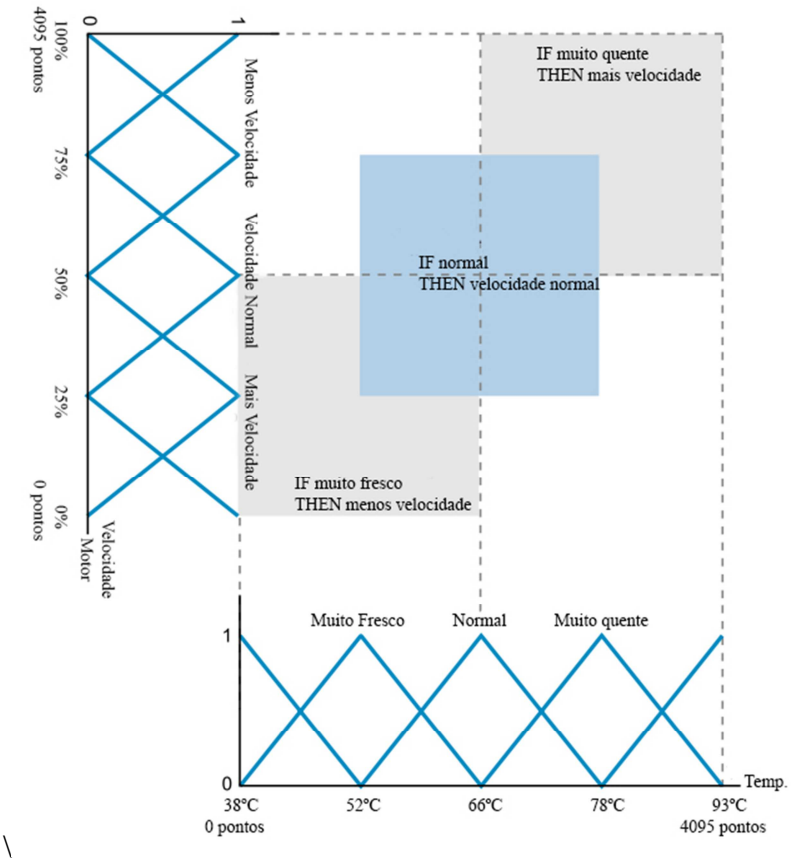


Figura 9 – Sistema lógico difuso com representação gráfica simultânea dos níveis de entrada e saída

Como o controlador de lógica difusa apenas tem uma saída, ele efetua um processo chamado de desfusificação (explicado mais tarde) para determinar o valor efetivo de saída final.

A implantação e operação de um sistema de controlo de lógica difusa é semelhante à implementação do controlo PID utilizando interfaces inteligentes, onde o módulo lê a entrada, processa as informações e fornece uma saída.

2.4. COMPONENTES DE UM SISTEMA DE CONTROLO DIFUSO

Nesta seção, são explicados os principais componentes de um controlador de lógica difusa e também como implementar um programa de controlo difuso simples. As três principais ações realizadas por um controlador de lógica difusa são:

- Fusificação
- Processamento difuso

- Desfusificação

Como se pode ver na Figura 10, quando o controlador difuso recebe os dados de entrada, estes são traduzidos sobre uma forma difusa. Este processo é chamado fusificação. O controlador, em seguida, executa o processamento difuso, que envolve a avaliação das informações de entrada de acordo com as regras IF...THEN criadas pelo utilizador durante a programação e projeto do sistema de controlo difuso. Uma vez que o controlador difuso termina a fase de transformação de regras e chega a uma conclusão/resultado, inicia-se o processo de desfusificação.

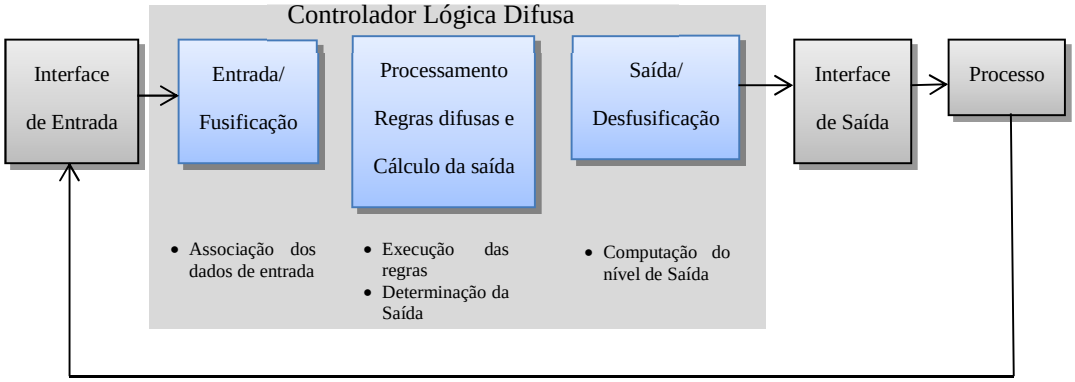


Figura 10 – Funcionamento de um controlador de lógica difusa

Nesta etapa final, o controlador difuso converte as conclusões/resultados de saída para saída de dados "reais" (por exemplo, valores analógicos) e envia esses dados para o processo através de um módulo de interface de saída. Se o controlador de lógica difusa está localizado no PLC e não tem um interface direto de Entradas e Saídas com o processo, então a saída de desfusificação será mapeada para a localização de memória do PLC e por sua vez mapeada para um módulo de interface de saída do processo.

2.5. COMPONENTES DO SISTEMA DE FUSIFICAÇÃO

O processo de fusificação é a interpretação dos dados de entrada do controlador difuso. A fusificação consiste em dois componentes principais:

• Funções de pertença

• Etiquetas

2.5.1. FUNÇÕES DE PERTENÇA

Durante a fusificação, um controlador de lógica difusa recebe dados de entrada, também conhecido como a variável difusa, e analisa-os de acordo com gráficos definidos pelo utilizador, chamados de funções de pertença (ver Figura 11). As funções de pertença agrupam os dados de entrada, tais como temperaturas que são muito frias, velocidade do motor que são aceitáveis, etc. O controlador atribui aos dados de entrada um nível entre 0 a 1 conforme eles se encaixam em cada função de pertença (por exemplo, 0,45 demasiado frio, 0,7 velocidade aceitável).

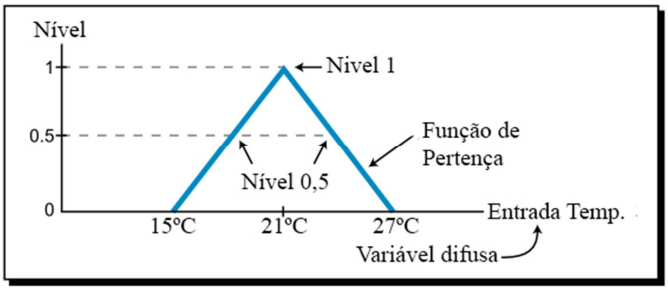


Figura 11 – Gráfico de funções de pertença

As funções de pertença podem ter diversas formas, dependendo do conjunto de dados, mas os mais comuns são o S, Z, Λ, e Π, mostradas na Figura 12. Note-se que estas funções de pertença são feitas de segmentos de reta definida por pontos. A função de cada membro só pode ter até três segmentos de reta com um máximo de quatro pontos.

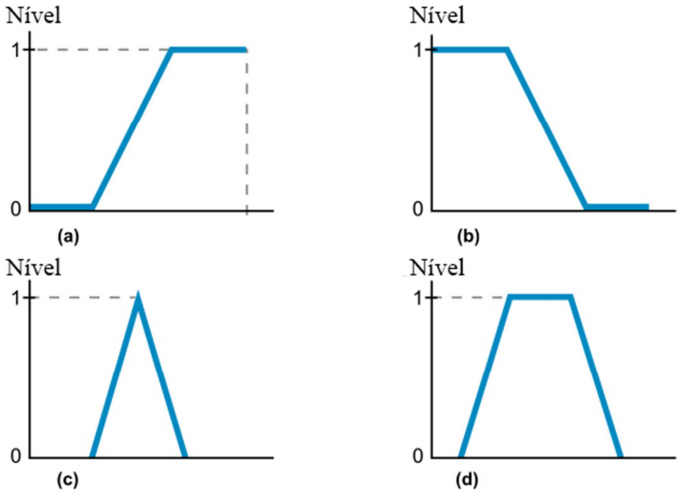


Figura 12 – Formas das funções de pertença (a) S, (b) Z, (c) Λ, e (d) Π

O nível em cada ponto deve ter um valor entre 0 e 1. Como mostrado na Figura 12, a forma de uma função de pertença não tem de ser simétrica, no entanto, deve obedecer às

especificações previamente discutidas. A Figura 13 ilustra algumas formas de funções de pertença incorretas.

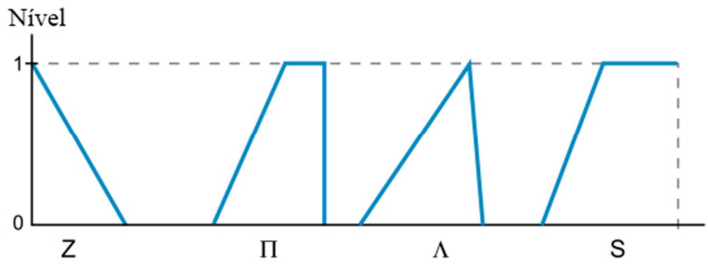


Figura 13 – Funções de pertença assimétricas

2.5.2. ETIQUETAS

Cada entrada do controlador difuso pode ter várias funções de pertença, sendo sete o máximo, que definem as suas condições. Cada função de pertença é definida por um nome, uma etiqueta. Por exemplo, uma variável de entrada, tal como a temperatura pode ter cinco funções de pertença etiquetados como frio, fresco, normal, tépido e quente. Quando definidas as sete funções de pertença, genericamente, têm as seguintes etiquetas, que se iniciam desde o ponto mínimo do intervalo de dados (negativo grande) até ao seu ponto máximo (positivo grande):

• NL (negativo grande) (*negative large*)

• NM (negativo médio) (*negative médium*)

• NS (negativo pequeno) (*negative small*)

• ZR (0)

• PS (positivo pequeno) (*positive small*)

• PM (positivo médio) (*positive médium*)

• PL (positivo grande) (*positive large*)

A Figura 14 ilustra um exemplo de uma variável de entrada com sete funções de pertença sob a forma Λ usando todas as possíveis etiquetas. Um grupo de funções de pertença formam um conjunto difuso.

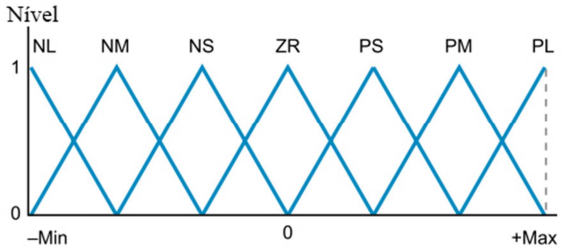


Figura 14 – Entrada de um sistema lógico difuso com sete etiquetas de funções de pertença

A Figura 15 mostra um conjunto difuso com cinco funções de pertença. Embora a maioria dos conjuntos difusos tenham um número ímpar de etiquetas, um conjunto pode também ter um número par de etiquetas. Por exemplo, um conjunto difuso pode ter quatro ou seis etiquetas em qualquer forma, dependendo de como as entradas são definidas em relação à função de pertença.

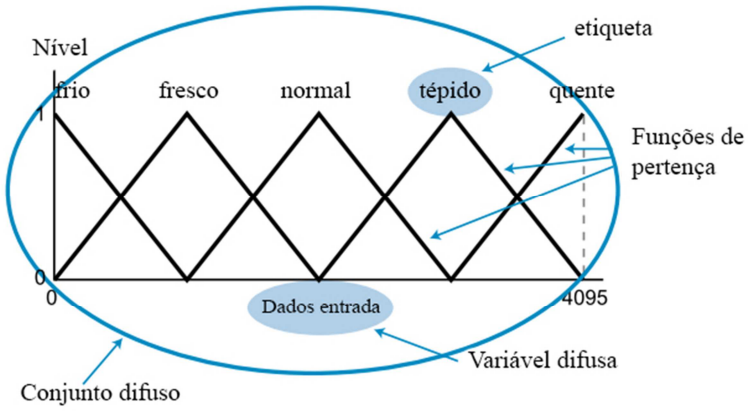


Figura 15 – Conjunto difuso com 5 funções de pertença

2.6. COMPONENTES DO SISTEMA DE PROCESSAMENTO DIFUSO

Durante o processamento difuso, o controlador analisa os dados de entrada, tal como definido pelas funções de pertença, para se chegar a uma saída de controlo. Durante esta fase, o processador executa duas ações:

• Avaliação das regras

• Cálculo do resultado de saída

2.6.1. AVALIAÇÃO DAS REGRAS

A lógica difusa é baseada no conceito de que os problemas mais complicados são formados por um conjunto de problemas simples, e pode, portanto, ser facilmente resolvidos. A lógica difusa utiliza um raciocínio ou inferência, o processo é composto por regras IF...THEN, cada uma fornece uma resposta ou saída.

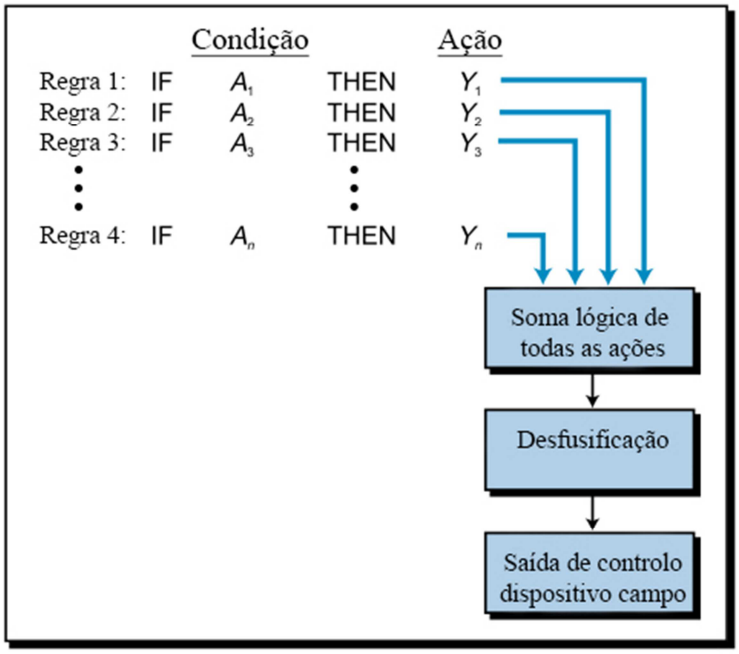


Figura 16 – Múltiplas regras num sistema difuso

Basicamente, a regra é ativada ou desencadeada, se uma condição de entrada satisfaz a parte IF da declaração da regra. Isso resulta numa saída de controlo com base na parte THEN da declaração da regra. Num sistema de lógica difusa, podem existir muitas regras, correspondendo a uma ou mais condições IF (ver Figura 16).

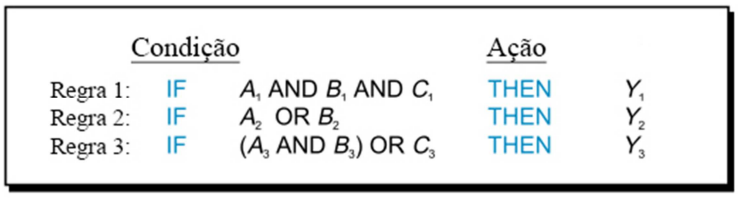


Figura 17 – Regras com múltiplas entradas relacionadas através de AND ou OR lógico

Uma regra também pode ter várias condições de entrada, que são ligadas através de um conetores lógicos OR ou AND para acionar o resultado/saída da regra (ver Figura 17). Por vezes, num mesmo instante, mais do que uma regra é acionada num processo de controlo

difuso. Neste caso, o controlador avalia todas as regras para se chegar a um resultado único e prossegue para o processo de desfusificação. Por exemplo, se duas entradas lógicas são multiplicadas (AND) ou somadas (OR) em várias regras, então produzirão vários resultados, dos quais apenas um será logicamente adicionado para determinar o resultado final. A Figura 18 (a) mostra um exemplo de duas entradas difusas, X1 e X2, e uma saída difusa, Y1 (Figura 18 (c)). As expressões mostradas na Figura 18 (b) representam quatro das nove possíveis regras, que abrangem as duas entradas. As quatro regras mostram, no entanto, os quatro pontos possíveis para as duas leituras de entrada, X1 e X2.

Tendo em conta os valores de entrada na Figura 18 (a), as entradas irão desencadear a regra 1 porque X1 = ZR e X2 = NL. Isto gerará duas saídas para Y1 = NL, um num nível de 0,6 (em função do valor de entrada de X1) e o outro com um nível de 0,75 (em função do valor de X2). Numa situação de lógica difusa, quando uma regra de duas entradas, com uma relação AND produz dois valores de resultados, o controlador irá escolher o resultado com um nível menor, no caso exemplificado será 0.6NL. Se a regra utiliza a lógica OR, o resultado escolhido será aquele com o maior nível. Se a regra 1 na Figura 18 tinha usado uma função OR em vez de uma função AND, em seguida, o controlador teria selecionado o resultado Y1 = 0.75NL, o maior dos dois resultados.

Diferentes controladores de lógica difusa têm diferentes capacidades de avaliação de regras.

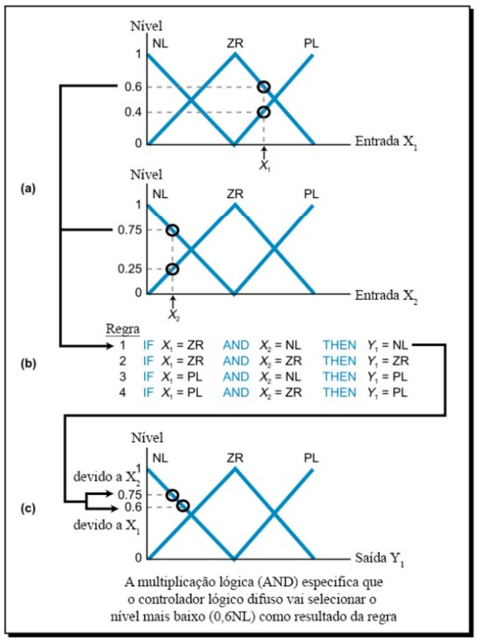


Figura 18 – Exemplo de processamento difuso com (a) duas entradas difusas (b) quatro regras que podem ser acionadas, e (c) o resultado de saída

O controlador de lógica difusa da Omron Electronics mostrado na Figura 19, por exemplo, é capaz de lidar com oito entradas e quatro saídas, onde cada entrada pode ser representado por um máximo de sete funções de pertença para um total de 56 funções de pertença (8 × 7). O controlador também permite um máximo de 128 regras programadas. Cada regra pode ter até oito condições de entrada (que pode ser logicamente AND ou OR) e dois resultados.

Regras de lógica difusa com duas entradas são muitas vezes representadas na forma matricial para representar condições AND. Por exemplo, a Figura 20 ilustra uma matriz de 3 x 3 (9 regras) que utiliza duas entradas, X1 e X2, e uma saída Y1. Uma vantagem desta representação matricial é que torna mais fácil de representar todas as regras de um sistema.

Figura 19 – Módulo de lógica difusa para PLC da OMRON

Um sistema de cinco etiquetas traduz-se numa matriz de 5 × 5, com 25 regras, enquanto um sistema de sete etiquetas produz uma matriz de 7 × 7 com 49 regras. Uma combinação par de funções de pertença (por exemplo, um sistema com 6 etiquetas para uma entrada e 4 etiquetas para outra) terá uma matriz de 24 regras.

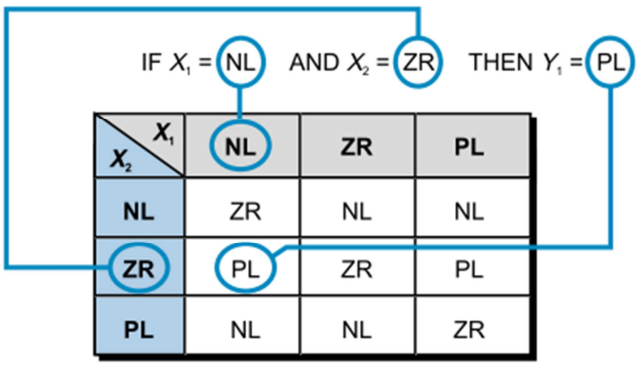


Figura 20 – Matriz de regras de lógica difusa

Quando são usadas mais de três entradas, a matriz torna-se mais difícil para representar, uma vez que se torna uma matriz tridimensional semelhante a um cubo (três entradas). Neste tipo de sistema complexo, as regras devem ser divididas em várias matrizes bidimensionais.

2.6.2. CÁLCULO DO RESULTADO DE SAÍDA

Uma vez ativada a regra, significa que os dados de entrada pertencem a uma função de pertença que satisfaz a regra IF, a regra vai gerar um resultado de saída. Esta saída difusa é composta por uma ou mais funções de pertença (com etiquetas), que têm níveis a elas associadas. O nível da função de pertença da saída é afetado pelo nível dos dados de entrada na sua função de pertença de entrada. Na Figura 21(a), o FI entrada difusa de 60% pertence a duas funções de pertença, ZR e PS, correspondentes aos níveis de 0,6 e 0,4, respetivamente. Estes dois níveis terão um impacto sobre o valor da saída (ver Figura 21 (b)) pela intersecção das funções de pertença de saída com os mesmos níveis (0,6 e 0,4). No entanto, a função de pertença de saída que é selecionado para o valor final de saída, depende da programação das regras IF...THEN pelo utilizador.

Por exemplo, na Figura 22, a entrada ativa as regras 3 e 4 porque a entrada FI pertence às funções de pertença ZR e PS. Estas regras indicam as duas saídas difusas ZR e PL, que devem ser aplicadas ao processo.

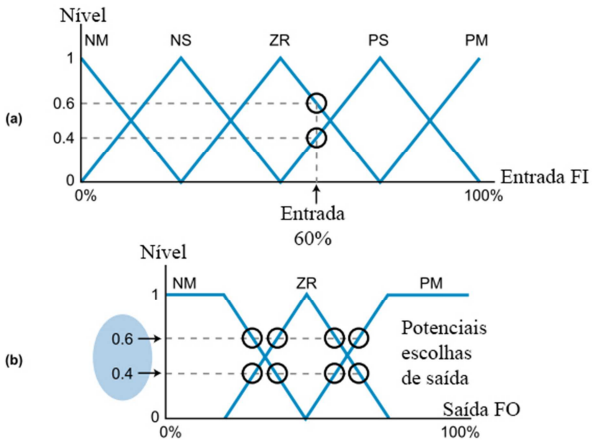


Figura 21 – (a) Níveis de entradas difusas (b) e os níveis dos resultados de saída

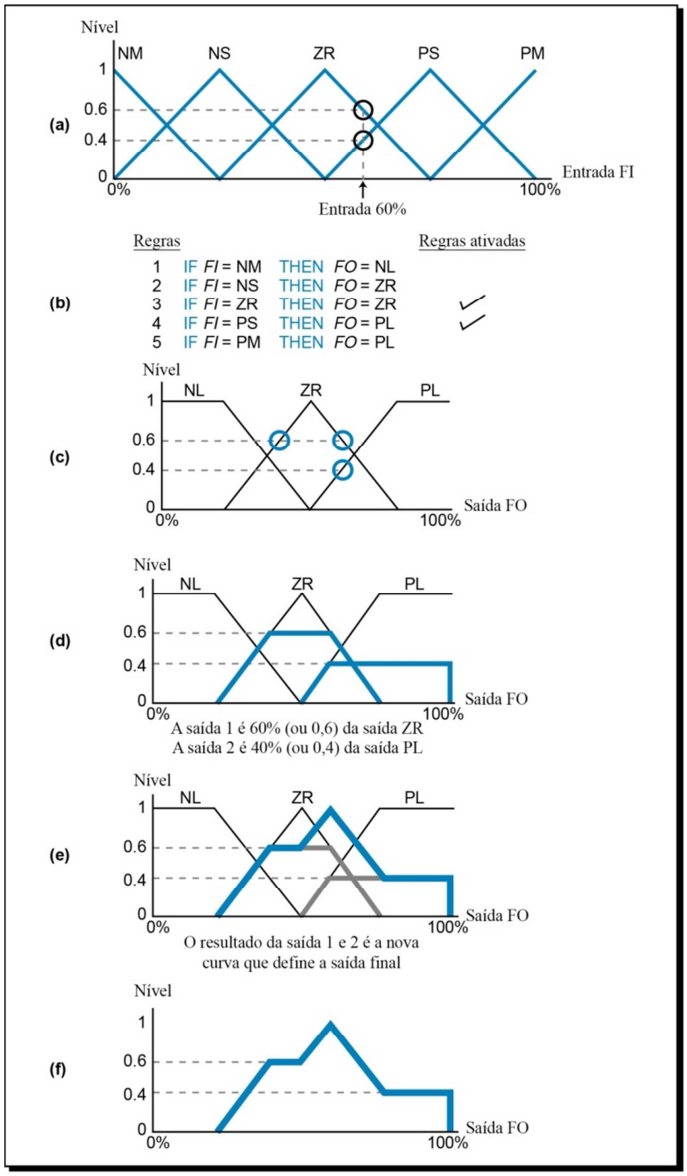


Figura 22 – Processo de lógica difusa (a) entradas, (b) regras, (c) saídas, (d) curvas de saída, (e) curvas de saída combinadas, e (f) o sinal de saída para o dipositivo de campo

Note-se que o nível de 0.6 é aplicado à saída ZR e 0,4 é aplicado à saída PL, porque o utilizador programou as regras dessa forma. A Figura 22 (C) mostra estas duas saídas. Para chegar a um valor final de saída, o controlador de lógica difusa, logicamente, soma ambos os resultados difusos para produzir uma curva de saída conjunta, ilustrada na Figura 22 (e). Então, o controlador gera um sinal de saída (durante desfusificação) que controla o dispositivo de campo do processo (por exemplo, válvulas, motor, etc.) de acordo com os dados de entrada (ver Figura 22 (f)).

Um controlador de lógica difusa pode implementar as funções de pertença de saída como funções descontínuas que se assemelham, a picos em vez de formas geométricas.

A Figura 23 ilustra um exemplo de função de pertença com sete saídas representadas como picos e descritas por etiquetas. Cada etiqueta tem uma relação com a interface de saída. Por exemplo, cada etiqueta mostrada na Figura 23 corresponde a um valor entre 0 e 4095 pontos. Como outro exemplo, as três funções de adesão de saída apresentados na Figura 22 podem ser representadas como picos descontínuos (ver Figura 24), onde os níveis de saída são 0,6 ZR e 0,4 de PM.

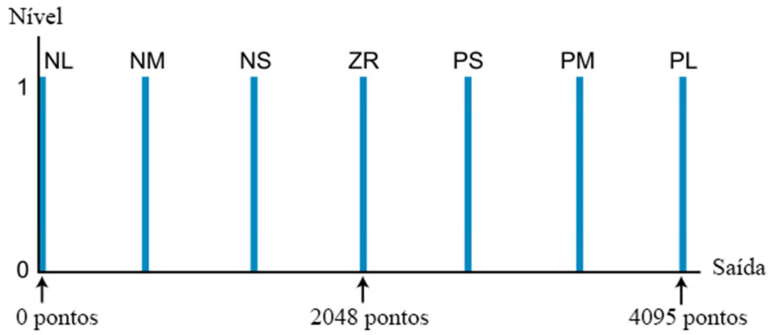


Figura 23 – Função de pertença discreta

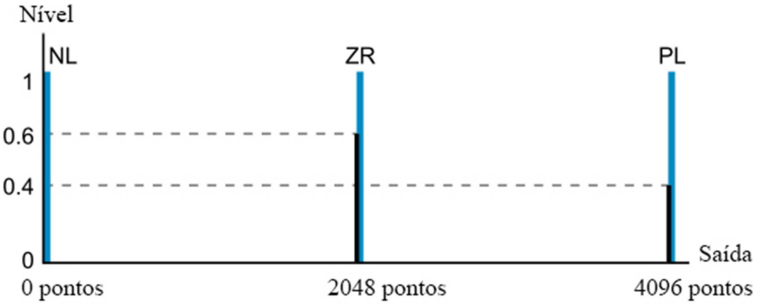


Figura 24 – Três funções de pertença de saída da Figura 22 representadas como picos

2.7. COMPONENTES DO SISTEMA DE PROCESSAMENTO DIFUSO

O valor final de saída do controlador difuso depende do método de desfusificação usado para calcular os valores dos resultados correspondentes a cada etiqueta. O processo de desfusificação analisa todos os resultados das regras, após terem sido adicionados e, logicamente, em seguida, calcula um valor que será o produto final do controlador difuso. O PLC envia esse valor para o módulo de saída. Assim, durante a desfusificação, o controlador converte a saída difusa para um valor de dados reais (por exemplo, 1.720 pontos).

Existem muitos métodos de desfusificação, mas todos são baseados em algoritmos matemáticos. Os dois métodos desfusificação mais comuns são:

• Método do valor máximo

• Método do centro de gravidade

2.8. MÉTODO DO VALOR MÁXIMO

O método do valor máximo baseia-se no maior valor de saída da função de pertença com o maior nível. Este método é usado principalmente com funções de pertença de saída discreta. Referindo-se à Figura 24 (mostrado novamente na Figura 25), o método de desfusificação do valor máximo irá especificar que o valor de saída corresponde ao nível de 2048, porque esse ponto tem o valor com maior nível. Se dois ou mais resultados a partir de duas ou mais regras têm o mesmo nível, então o controlador irá selecionar o resultado que será o valor final com base em critérios fornecidos pelo utilizador durante o projeto ou definição de programação do sistema difuso. Tais critérios são determinados pela escolha do valor mais à esquerda ou mais à direita das etiquetas com níveis iguais e seu correspondente ponto. O critério mais à esquerda seleciona a menor saída (pontos), enquanto o critério mais à direita seleciona a maior saída (maior número de pontos).

A Figura 26 (a) ilustra o resultado de saída de três regras. Se o método desfusificação do máximo valor for usado, ZR será o valor final de saída, o que significa que a saída do controlador pode ser 2340 pontos. Se as regras desencadeiam dois valores máximos com nível igual, como é o caso na Figura 26 (b), então o controlador usaria os critérios programados para selecionar o valor de saída apropriado.

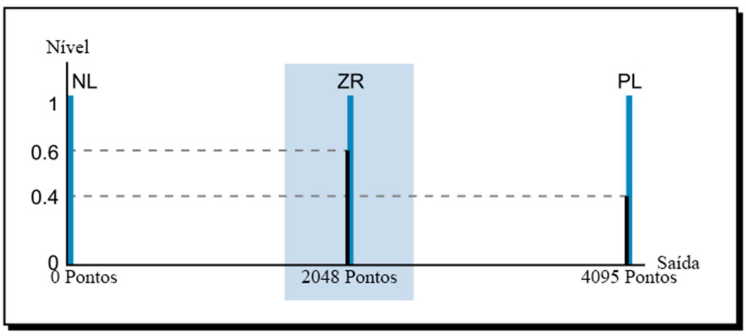


Figura 25 – O Método do valor máximo seleciona o maior nível de saída

Se o critério especificado for o valor máximo mais à esquerda, então o controlador escolheu a etiqueta NM, o que proporcionaria uma saída de 1170 pontos. Note-se que, durante o processo de desfusificação, o controlador difuso envia para o dipositivo de saída

o valor real (por exemplo, pontos), não o valor do nível. Assim, na Figura 26 (a), a saída será de aproximadamente 2340 pontos. Na Figura 26 (b), o máximo valor de saída mais à esquerda será de aproximadamente 1170 pontos e o valor máximo de saída mais à direita, se escolhido, será 3510 pontos.

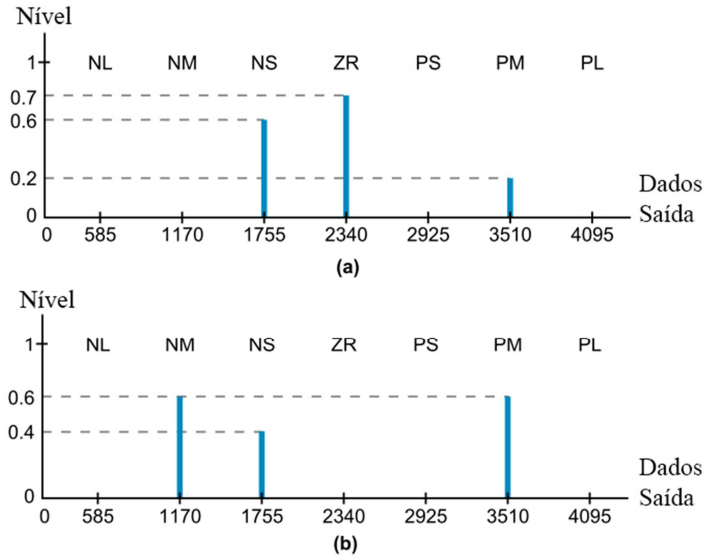


Figura 26 – (a) Valor máximo de saída única (b) Múltiplos valores máximos de saída

2.8.1. MÉTODO DO CENTRO DE GRAVIDADE

O método do centro de gravidade, também conhecido como "cálculo do centroide" obtém-se calculando matematicamente o centro de massa das funções de pertença de saída ativadas. A Figura 27 ilustra o cálculo do centroide para o exemplo anteriormente ilustrado na Figura 22. Em termos matemáticos, um centroide é o ponto numa figura geométrica cuja coordenada é igual à média de todos os outros pontos que compreendem a figura.

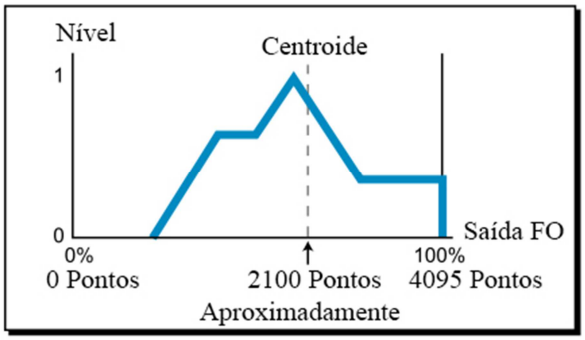


Figura 27 – Cálculo do centroide da saída da Figura 22

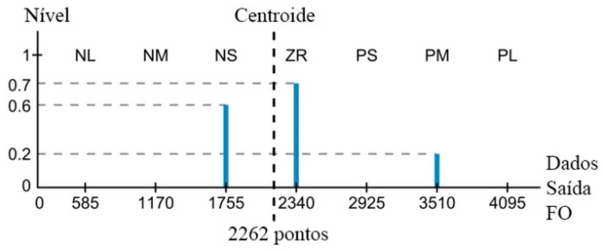
Este ponto é o centro de gravidade da figura. Em termos simples, o centro de gravidade para uma saída difusa é o valor de saída de dados (como se mostra no eixo dos X), que divide a área sob a curva de função de pertença difusa em duas partes iguais. O centro de gravidade é o método mais vulgarmente usado de desfusificação porque proporciona resultados exatos com base nos valores ponderados de várias funções de pertença de saída. O valor de saída que é enviada para o módulo de interface de saída é o valor dos dados de saída na intersecção do eixo horizontal e do centroide.

O método do centro de gravidade aplica-se a funções de pertença de saída não contínuas, ou discretas, bem como contínuas. Em funções não contínuas, o valor de saída final que vai ser obtido para uma função de pertença de saída de sete etiquetas (etiquetas de A até G) é expresso pela fórmula:

$$\text{Dados Saída} = \frac{\sum_{n=A}^{n=G}[(FO_n)(Fnível_n)]}{\sum_{n=A}^{n=G} Fnível_n} \qquad (1)$$

Em que:

- Dados Saída: O número de pontos a ser colocado na saída
- *FO*: o valor da saída difusa, expressa em pontos, para as etiquetas n=A até G
- *Fnível*: o valor do nível para as etiquetas n=A até G



$$\text{Valor final de saída} = \frac{\sum_{n=PL}^{n=NL}[(FO_n)(Fnível_n)]}{\sum_{n=PL}^{n=NL} Fnível_n}$$

$$= \frac{(FO_{NS})(Fnível_{nNS}) + (FO_{ZR})(Fnível_{ZR}) + (FO_{PM})(Fnível_{PM})}{(Fnível_{nNS}) + (Fnível_{ZR}) + (Fnível_{PM})}$$

$$= \frac{(1755)(0{,}6) + (2340)(0{,}7) + (3510)(0{,}2)}{(0{,}6) + (0{,}7) + (0{,}2)}$$

$$= \frac{3393}{1{,}5}$$

$$= 2262\ pontos$$

Figura 28 – Cálculo do centroide numa função de pertença descontínua

Com base no exemplo da função de pertença descontínua, na Figura 28, esta equação indica que o valor final da saída é igual à soma do nível de cada resultado de cada regra pelo seu valor real de dados de saída (isto é, os pontos), dividido pela soma do nível de cada resultado de cada regra.

Neste caso, o controlador de lógica difusa irá decidir o envio de uma saída de 2262 pontos para a interface de saída depois de completar o cálculo do centro de gravidade. A saída do controlador difuso é menor do que teria sido utilizando o método do valor máximo (a etiqueta de saída máxima é ZR, que é 2340 pontos), indicando que o valor ponderado da etiqueta 0.6NS puxa o valor para a esquerda (menos pontos). No entanto, o valor de saída é ligeiramente puxado para a direita face à contribuição da ação direita da etiqueta 0,2PM.

Os controladores difusos que utilizam funções de pertença contínuas e o método do centro de gravidade na desfusificação, também pode ser usada a equação de soma anterior, para aproximar o valor do centroide (ver Figura 29). No entanto, neste caso, o controlador usa os valores digitalizados aproximados para cada função de adesão para calcular cada um dos pontos no somatório [3] [4] [5].

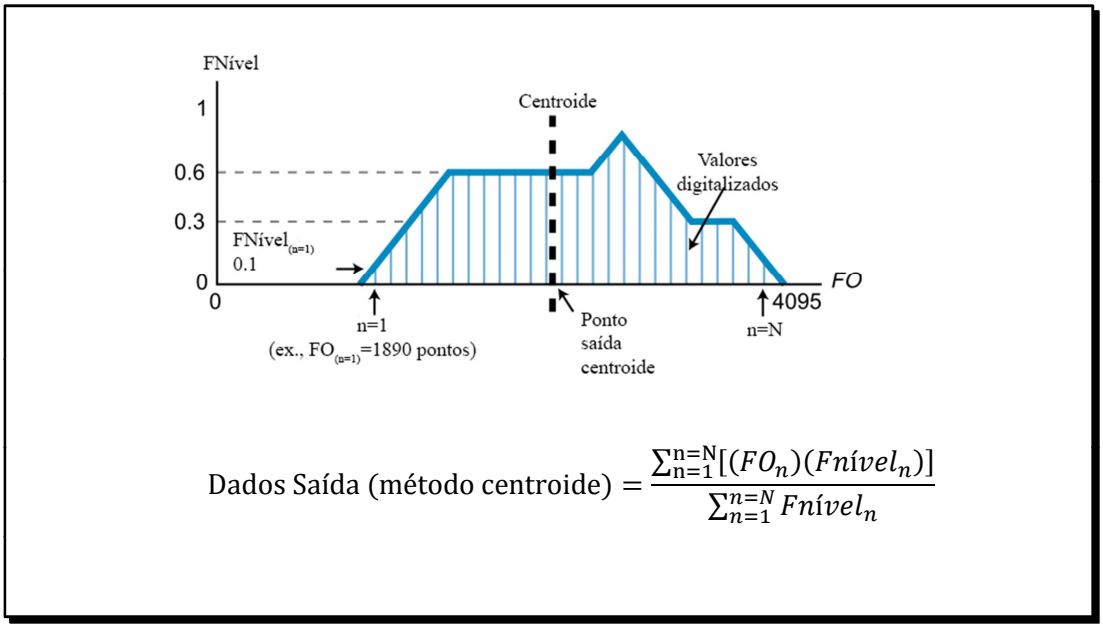


Figura 29 – Aproximação do valor do centroide

3. CONTROLO DIFUSO EM PLC

Este capítulo aborda a presença do controlo difuso nos PLC, quer ao nível do *hardware*, havendo marcas que comercializam interfaces de lógica difusa, quer ao nível do *software* para programação da lógica difusa dentro dos autómatos normais.

3.1. *HARDWARE*

Os interfaces de lógica difusa, que são oferecidos por poucos fabricantes de PLC, fornecem uma maneira de implementar algoritmos de lógica difusa em PLC. Os algoritmos de lógica difusa analisam os dados de entrada para fornecer o controlo de um processo. Tal como mostrado na Figura 30, os módulos de lógica difusa não funcionam como interfaces de entrada e de saída, por si só. Em vez disso, eles funcionam com outras interfaces de entrada e de saída, fornecendo uma ligação inteligente entre os dois.

Os módulos de lógica difusa são parte integrante das capacidades avançadas de controladores programáveis de hoje. Eles ajudam a preencher a lacuna entre as funções de tomada de decisão discretas e analógicas de um PLC. Em essência, os módulos de lógica difusa permitem que os PLC interpretem os dados de uma forma tipo analógico em vez de apenas como ON ou OFF.

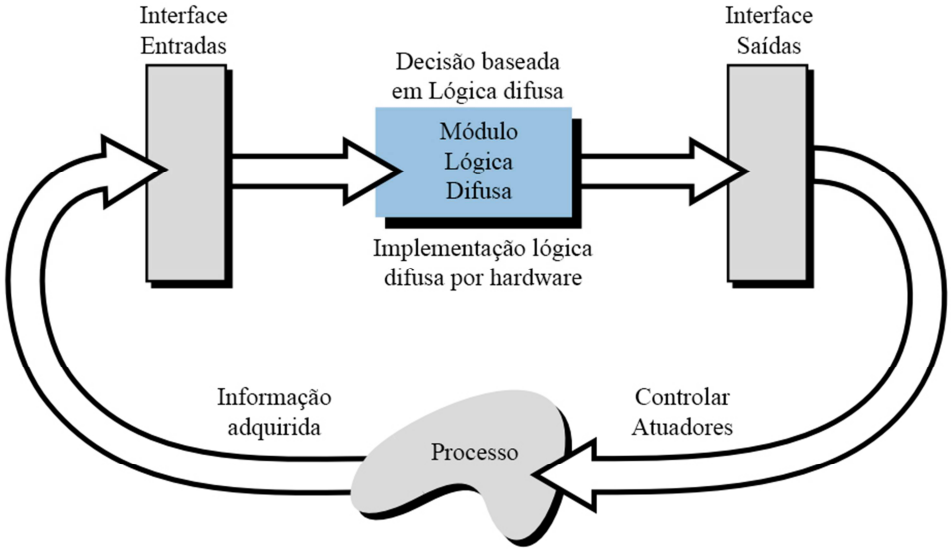


Figura 30 – Interface de aplicação de lógica difusa

Por exemplo, um PLC típico ligado a um dispositivo sensor de temperatura só pode sentir se uma temperatura é aceitável ou inaceitável (ver Figura 31(a)). Isto é, as temperaturas entre 15°C e 27°C são aceitáveis (valor lógico 1), todas as outras temperaturas são inaceitáveis (valor lógico 0). Um PLC com capacidades de lógica difusa, no entanto, pode discernir entre os intervalos de temperaturas aceitáveis e não aceitáveis, e julgar a temperatura para ser mais aceitável ou menos aceitável (ver Figura 31 (b)). Assim, um módulo de lógica difusa pode determinar que 17°C é uma temperatura aceitável, mas que não é tão aceitável como 21°C.

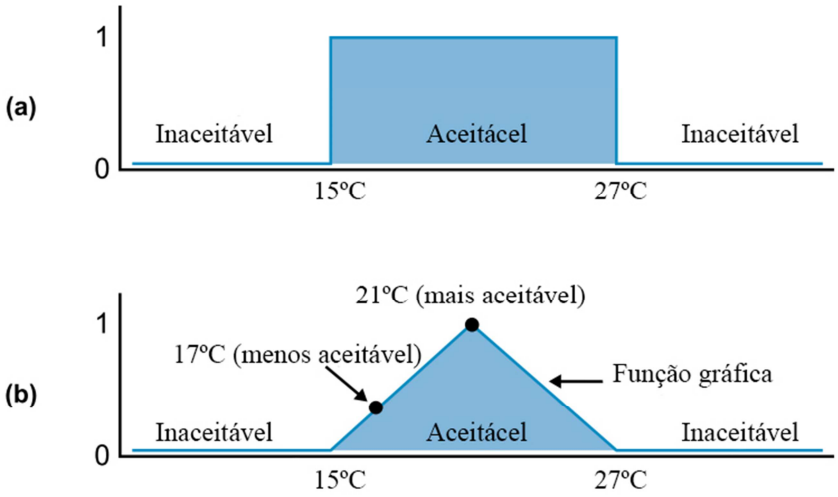


Figura 31 – Sensação de temperatura num (a) PLC normal e (b) num PLC com capacidade processamento de lógica difusa

A capacidade de "raciocínio" de módulos difusos permite-lhes um controlo ajustado de processos analógicos, assim como processos não-lineares e variantes no tempo, como controlo de tensão e de posição. Estes tipos de sistemas de difícil controlo geralmente têm grandes desvios de entrada ou resoluções de entrada insuficientes, que muitas vezes requerem a intuição humana e julgamento. Os módulos de lógica difusa podem fornecer este tipo de julgamento humano.

3.1.1. CONTROLADORES OMRON

Com o objetivo de criar harmonia entre pessoas e máquinas, OMRON perseguiu o desenvolvimento da lógica difusa, que foi originalmente introduzido pelo matemático Lotfi A. Zadeh, como sendo a principal tecnologia da próxima geração. No desenvolvimento de lógica difusa, OMRON entrou numa área inexplorada, que incorpora a ambiguidade e ideias humanas subjetivas para o controlo de uma máquina. Realizações da OMRON nessa área incluem o mais rápido controlador difuso do mundo (1987), um controlador difuso multitarefa ultra-alta velocidade à época e um chip de lógica difusa para computador, primeiro do mundo (ambos em 1988).

Na atualidade a OMRON apenas tem alguns dos seus processadores com lógica difusa nativa. No passado existia um controlador dedicado que apenas efetuava o controlo difuso, tratava-se do "C200H-FZ001 Fuzzy Logic Unit" face às capacidades mais limitadas dos autómatos existentes à época, este controlador dedicado conseguia melhores desempenhos.

Com a evolução dos autómatos normais estes passaram a ter capacidades para controlar o processo, para isso basta fazer a escolha correta do processador (ver Anexo A) e usar um bloco de função já existente na libraria do *software* de programação do processo, o CX-Process tool.

3.2. *SOFTWARE*

No campo dos *softwares* as opções são mais variadas, uma vez que muitas marcas de automação têm bibliotecas de controlo para alguns dos seus produtos, normalmente para os equipamentos que permitem controlo de processo, ou seja, deixam de fora os autómatos das gamas mais baixas.

3.2.1. BLOCK 016 OMRON

A OMRON foi uma das pioneiras no que diz respeito ao controlo difuso. Nos seus equipamentos listados na subsecção 3.1.1 é possível a utilização da biblioteca “Block 016 – Fuzzy Logic” (Figura 32).

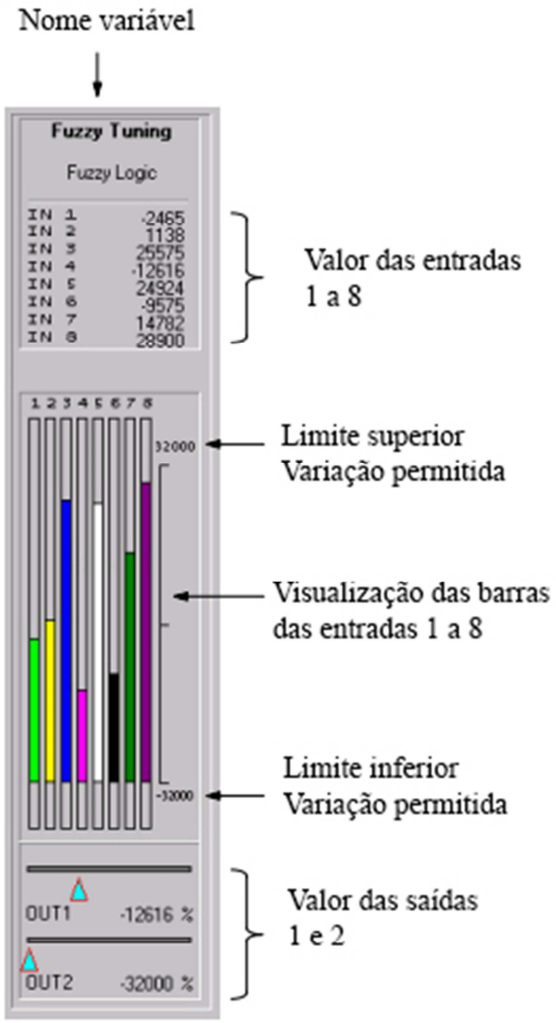


Figura 32 – Interface de visualização do bloco de função de controlo difuso da OMRON

Este bloco de função gera até duas saídas analógicas baseadas nas regras de lógica difusa de até 8 entradas analógicas, X1 a X8. Usando lógica difusa permite a aplicação do controlador de processo para controlar as aplicações que utilizam o conhecimento de profissionais experientes expressos em expressões ambíguas, como "um pouco" ou "bastante". Quando nenhuma das regras é satisfeita o sistema adota os valores de S1 ou S2. As funções de pertença de um bloco lógica difusa podem ser exibidos como gráficos na versão 2.50 ou posterior do CX-Process Tool.

As especificações do bloco de lógica difusa são as seguintes:

- I/O: 8 entradas e 2 saídas
- Formato Regra: 8 condições e duas conclusões
- Número de regras: 64 máx.
- Cinco etiquetas (NL, NS, ZR, PS e PL)
- Método de inferência: Máx.-Min. ANDs lógicas
- Cálculos de saída determinísticos: centro de gravidade
- Saída quando as regras não são cumpridas: Valor constante ou anterior (selecionável)

As especificações das funções de pertença são as seguintes:

Condições:

- Resolução: 64000 máx.
- Funções contínuas: 4 pontos de inflexão máx. (S, Z, Λ e Π) (subseção 2.5.1)
- Tamanho: 0 ou 10,000

Conclusões:

- Resolução: 64, 000 max.
- Tamanho: Sempre 10 000

Os blocos de funções de lógica difusa devem ser aplicados nas seguintes situações:

- Sistemas de controlo com muitas interferências externas.
- Substituir uma Unidade Lógica Difusa C200H-FZ001 da OMRON.

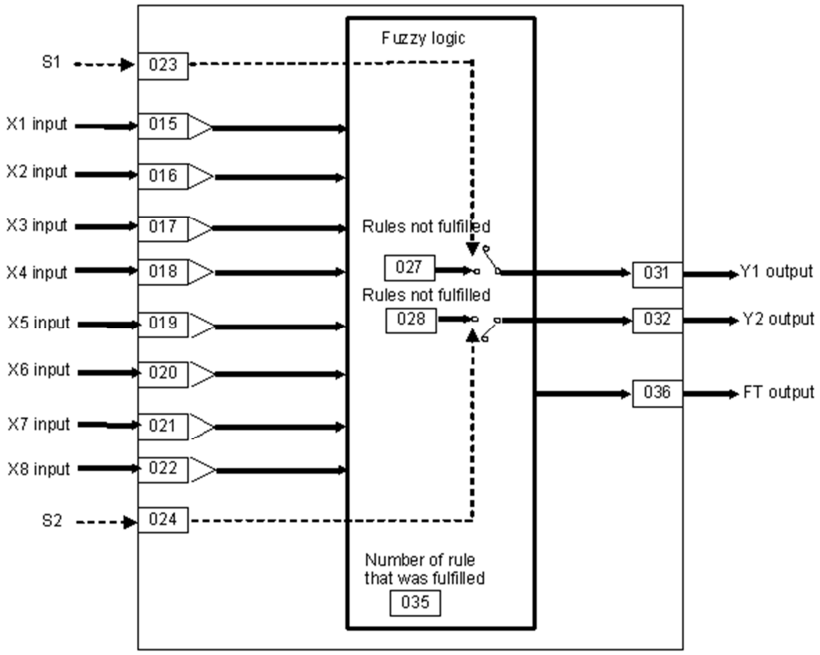


Figura 33 – Diagrama de blocos do controlador

Nas Figura 34 pode-se observar como é efetuada a parametrização do controlador de lógica difusa.

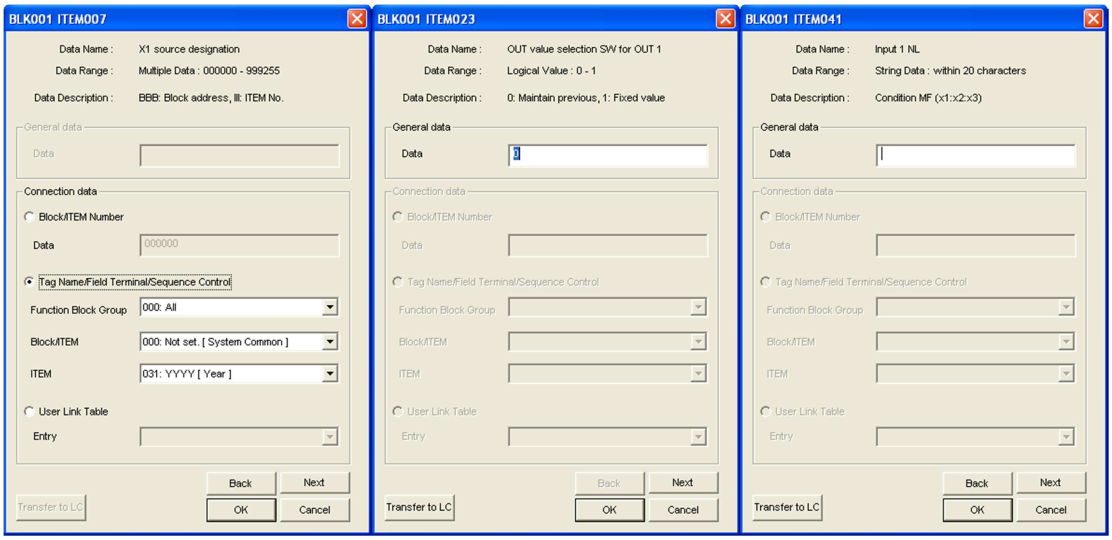


Figura 34 - Configuração das funções de pertença das entradas e saídas do sistema

A Figura 35 mostra um resumo de toda a parametrização efetuada [6].

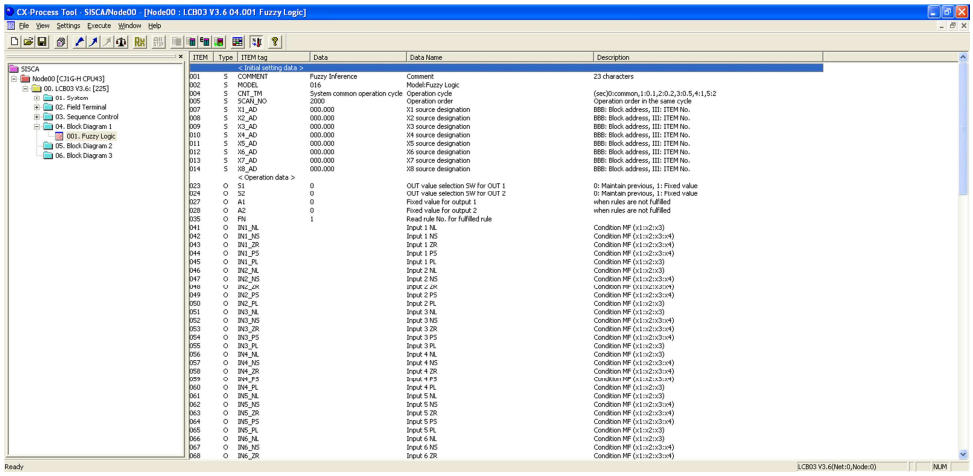


Figura 35 – Visualização de todos os parâmetros associados ao controlador

3.2.2. FUZZYCONTROL++ SIEMENS

A Siemens não tem *hardware* dedicado ao controlo lógico difuso, apenas dispõe de um *software* que uma vez instalado permite a interligação com o Step7 (*software* de programação de autómatos). O *software* chama-se FuzzyControl++ e permite a parametrização e programação de sistemas de lógica difusa para os autómatos das gamas S7-300 e S7-400.

Nas Figura 36 pode ver-se a configuração gráfica das funções de pertença de um sistema com duas entradas. A temperatura tem uma variação de 0 a 30 com duas funções de pertença (frio e quente) e de pressão que varia entre 50 a 100 com duas funções de pertença (baixa e alta).

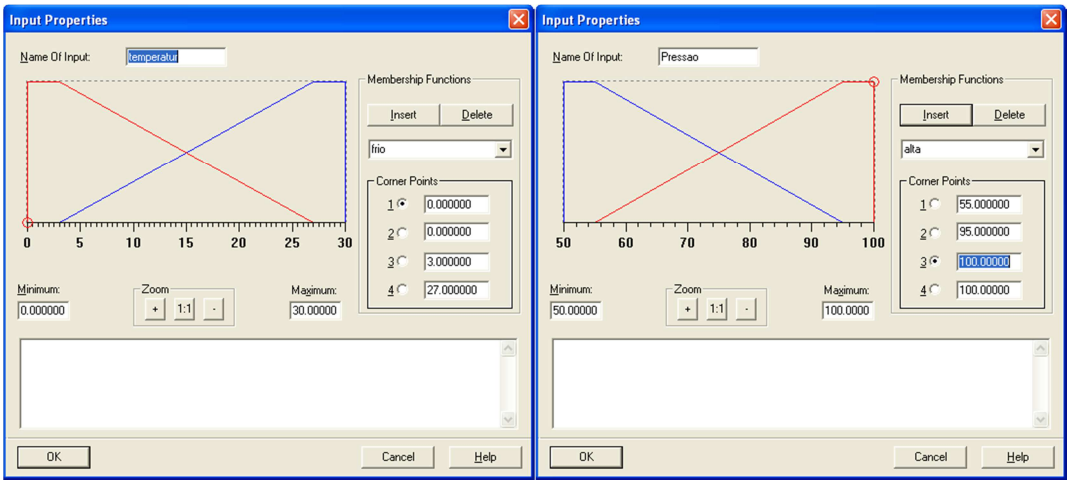


Figura 36 – Configuração das duas funções de pertença das duas entradas do sistema

Por sua vez na Figura 37 é ilustrada a configuração das funções de pertença (entrada, fechada e saída) da válvula de saída que varia entre -101 a 101.

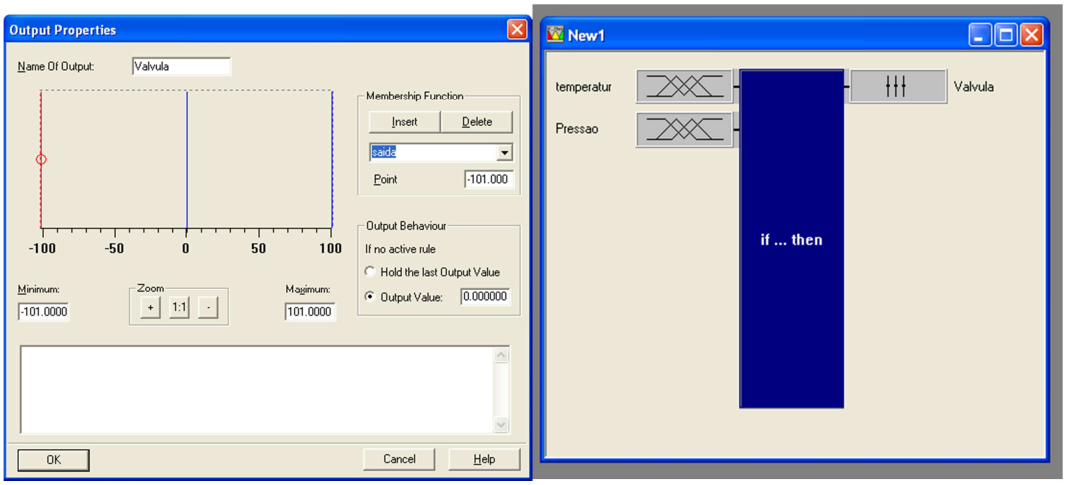


Figura 37 – Configuração das três funções de pertença da saída do sistema

Na Figura 38 é exemplificado como se definem as regras de controlo do sistema que tem 4 regras, uma vez que só tem 2 funções de pertença por cada entrada (2x2=4).

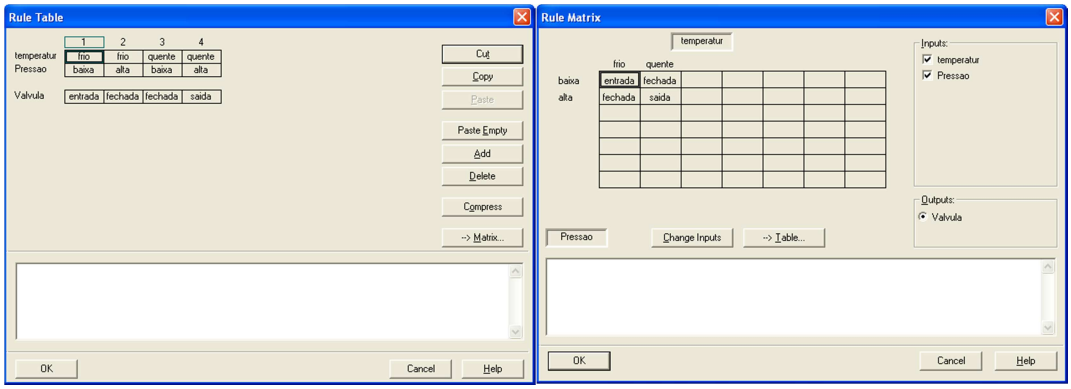


Figura 38 – Definição das regras do sistema

A Figura 39 ilustra a representação gráfica tridimensional de todo o sistema de lógica difusa que permite de uma forma intuitiva ter a noção do comportamento da saída (Válvula) em função das duas entradas (Temperatura e Pressão) [7].

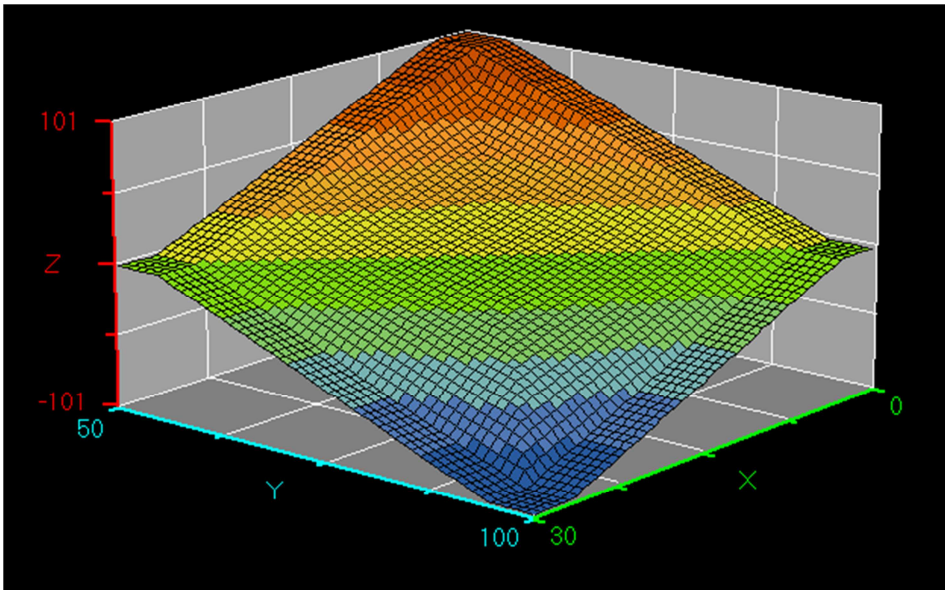


Figura 39 – Representação gráfica das duas entradas (X: Temperatura, Y:Pressão) e da saída (Z: Válvula)

3.2.3. FUZZYDESIGNER ALLEN BRADLEY (ROCKWELL)

A Allen Bradley tem disponível um pacote de *software* dedicado ao controlo difuso, não tendo disponível nenhum *hardware* dedicado. O pacote de *software* FuzzyDesigner é destinado ao projeto de sistemas difusos implementados sobre a forma de um sistema difuso hierárquico como pode ser visto na Figura 40.

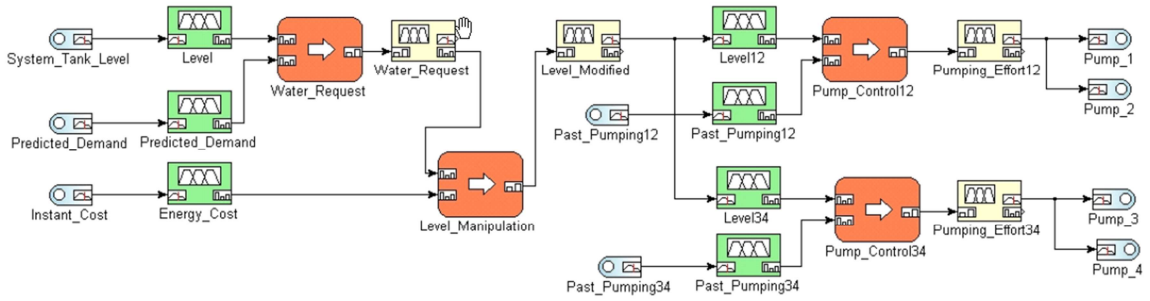


Figura 40 – Sistema difuso hierárquico implementado no *software FuzzyDesigner*

O FuzzyDesigner inclui uma biblioteca de componentes que podem ser usados para projetar um sistema difuso, que inclui o mapeamento de entradas e saídas não-lineares. Pode ser usada uma estrutura hierárquica para decompor um sistema difuso complexo em partes menores e mais simples. Isso reduz a complexidade interna de um modelo difuso e resulta em menos regras difusas e fornece uma visão mais fácil para a operação do sistema.

O FuzzyDesigner é projetado para trabalhar com a família de controladores Logix5000 da Rockwell Automation.

A utilização prevista para FuzzyDesigner é ilustrada abaixo na Figura 41. O pacote de *software* FuzzyDesigner é utilizado na conceção de instruções difusas como *add-on* para aplicações Logix.

O ciclo de desenvolvimento de soluções de lógica difusa para as aplicações Logix consiste nos seguintes passos, ilustrados na Figura 42:

1. Projetar o sistema lógico difuso no FuzzyDesigner.
2. Gerar a instrução difusa do *Add-on*.
3. Integrar (importar e instanciar) a instrução do *Add-on* no projeto RSLogix 5000
4. Afinar e monitorizar a lógica difusa em tempo real no Logix, usando o FuzzyDesigner

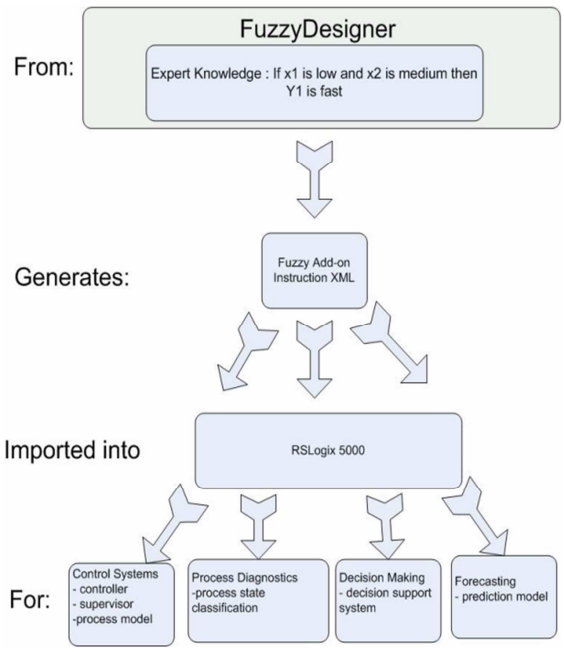


Figura 41 – Utilização prevista para o FuzzyDesigner [8]

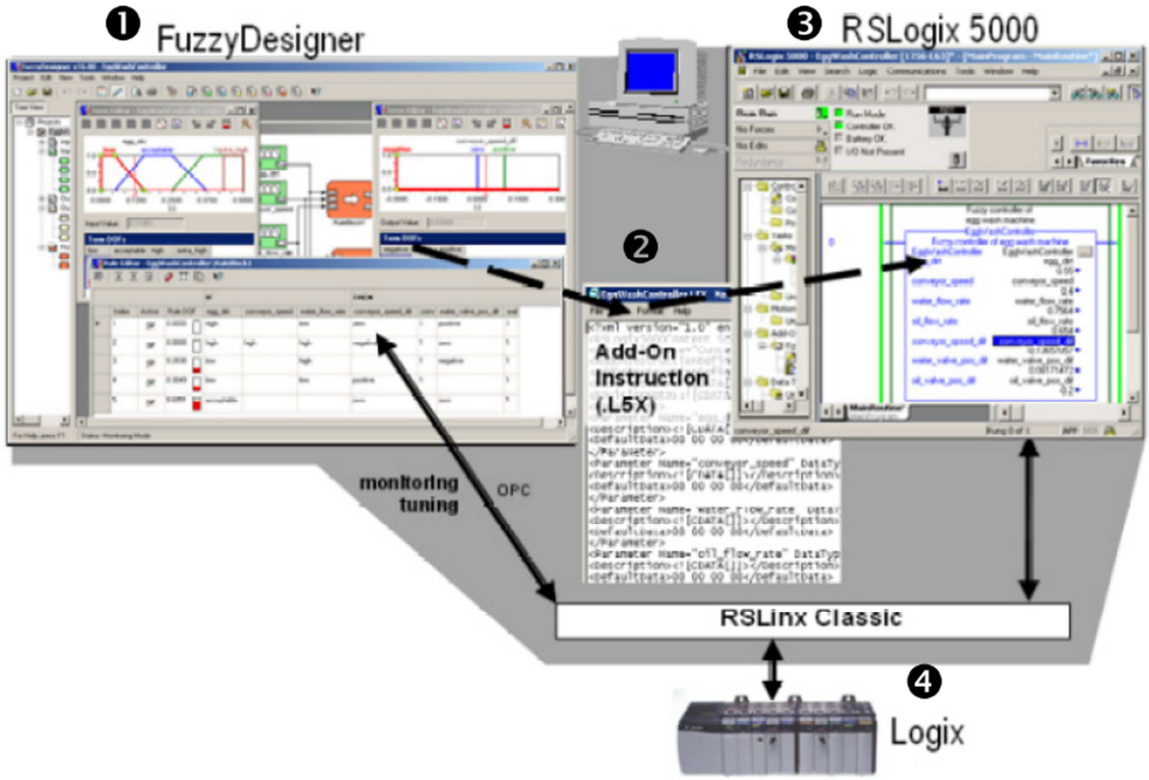


Figura 42 – Utilização do FuzzyDesigner com o RSLogix 5000 [8]

Na Figura 43 apresenta-se a forma como são parametrizadas as funções de pertença de entrada e de saída.

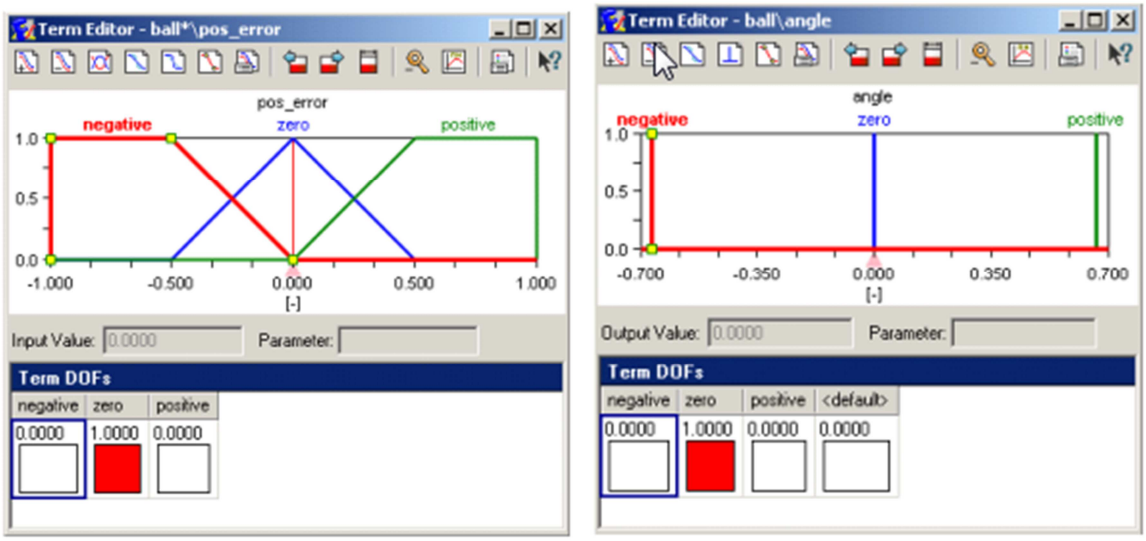


Figura 43 – Editor das funções de pertença da entrada de saída [8]

Na Figura 44 está ilustrada a forma como podem ser editadas as regras difusas no sistema.

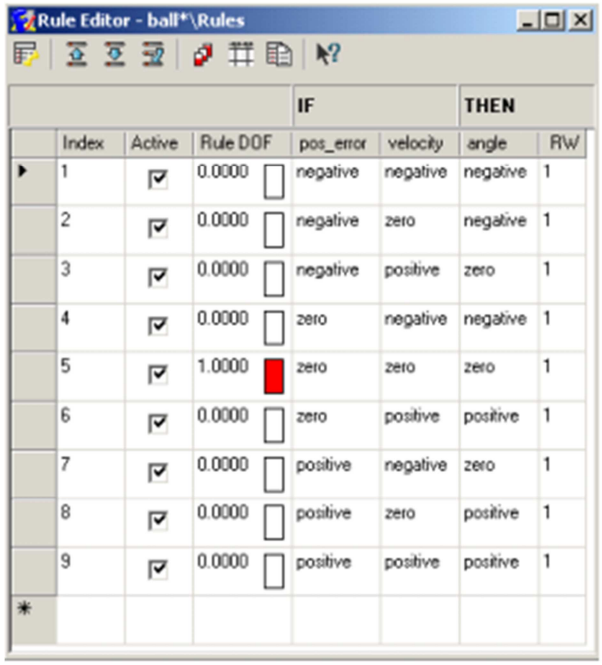


Figura 44 – Edição das regras difusas [8]

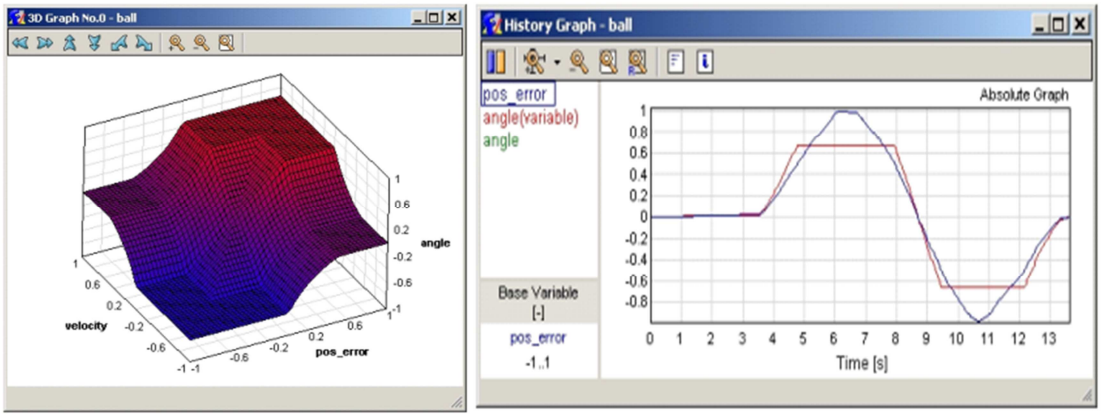


Figura 45 – Gráfico 2D e 3D do sistema [8]

Na Figura 45 pode-se ver a representação gráfica do sistema, para uma mais fácil interpretação dos resultados [8].

3.2.4. FUZZY CONTROL LIBRARY SCHNEIDER ELECTRIC

A Schneider Electric também só tem *software* de controlo difuso, a biblioteca Fuzzy Control Library V1.2. O controlo difuso está disponível através da plataforma Unity Pro, usada para a programação dos autómatos M340, Premium e Quantum.

Tabela 1 – Funções da biblioteca de controlo difuso

Operação	Função	Família	Descrição
Fusificação	FUZ_STERM	Fuzzify	Fusificação de um termo
	FUZ_ATERM	Fuzzify	Fusificação até 9 termos de uma vez
	FUZ_ATERM_STI, FUZ_ATERM_STR	Fuzzyfy_Struct	Fusificação até 9 termos de uma vez Guarda os resultados numa estrutura
Processamento difuso	FUZ_MAX	Operators_OR	Operador OR: Máximo
	FUZ_MIN	Operators_AND	Operador AND: Mínimo
	FUZ_SUM	Operators_OR	Operador OR: Soma
	FUZ_PROD	Operators_AND	Operador AND: Produto
Desfusificação	DEFUZ	Defuzzify	Desfusificação com picos descontínuos
	DEFUZ_STI, DEFUZ_STR	Defuzzify_Struct	Desfusificação com picos descontínuos Leitura dados da estrutura de entrada

Com base nos comandos descritos na Tabela 1 pode ser efetuada a parametrização dos sistemas de controlo difuso. Inicialmente é necessária a fusificação através da função FUZ_ATERM, que permite a fusificação até 9 termos de uma só vez. Na Figura 46 está ilustrada a utilização do bloco FUZ_ATERM.

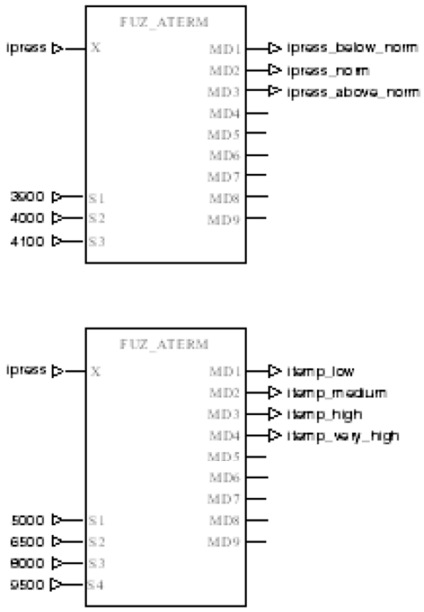


Figura 46 – Fusificação de duas variáveis, pressão e temperatura através da função FUZ_ATERM [9]

Tabela 2 – Descrição dos parâmetros de entrada e saída do bloco FUZ_ATERM [9]

Description of input parameters:

Parameters	Data type	Meaning
X	INT, REAL	linguistic variable
S1	INT, REAL	Point S1
S2	INT, REAL	Point S2
:	:	:
S9	INT, REAL	Point S9

Description of output parameters:

Parameters	Data type	Meaning
MD1	INT, REAL	Output MD1 Membership Degree
MD2	INT, REAL	Output MD2 Membership Degree
:	:	:
MD9	INT, REAL	Output MD9 Membership Degree

Depois da fusificação de todas as variáveis de entrada será necessária a definição das funções de pertença da saída com base no gráfico da Figura 47.

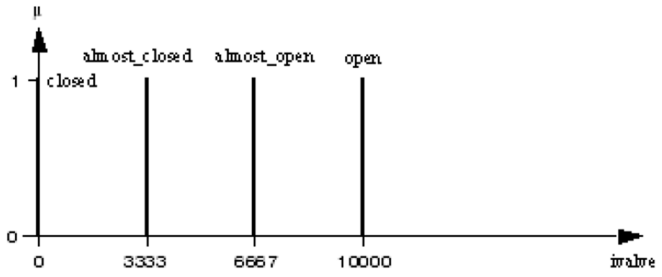


Figura 47 – Definição da saída ivalve com quatro funções de pertença [9]

Na Figura 48 pode ser observada a desfusificação com base nas regras de inferência definidas pelo utilizador.

A descrição dos parâmetros das entradas e saídas do bloco DEFUZ podem ser consultados na Tabela 3. No Unity as regras de inferências têm como critério de seleção os valores máximos e mínimos de cada comparação de entradas, como pode ser visto nas Tabela 4 e na Tabela 5 [9].

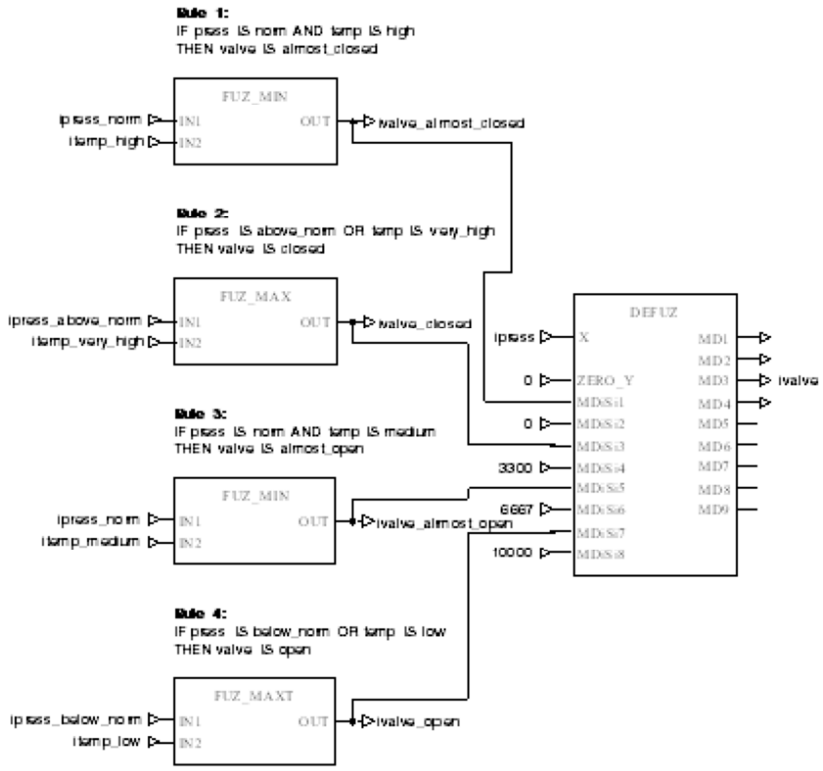


Figura 48 – Desfusificação do sistema [9]

Tabela 3 – Descrição dos parâmetros de entrada e saída do bloco DEFUZ [9]

Description of input parameters:

Parameter	Data type	Meaning
ZERO_Y	BOOL	0: Output of the last Y value
MDiSi1	INT, REAL	Singleton: 1; MD1 Membership Degree
MDiSi2	INT, REAL	Singleton: 1; S1 Support point
MDiSi3	INT, REAL	Singleton: 2; MD2 Membership Degree
MDiSi4	INT, REAL	Singleton: 2; S2 Support point
:	:	:
MDiSi17	INT, REAL	Singleton: 9; MD9 Membership Degree
MDiSi18	INT, REAL	Singleton: 9; S9 Support point

Description of output parameters:

Parameter	Data type	Meaning
Y	INT, REAL	Output

Tabela 4 – Descrição dos parâmetros de entrada e saída do bloco FUZ_MAX [9]

Parameter	Data type	Meaning
IN1	INT, REAL	1st Input value
IN2	INT, REAL	2nd Input value
:	:	:
INn	INT, REAL	n. Input value
Output	INT, REAL	Maximum value

Tabela 5 – Descrição dos parâmetros de entrada e saída do bloco FUZ_MIN [9]

Description of input parameters:

Parameter	Data type	Meaning
IN1	INT, REAL	1st Input value
IN2	INT, REAL	2nd Input value
:	:	:
INn	INT, REAL	n. Input value

Description of output parameters:

Parameter	Data type	Meaning
Output	INT, REAL	Maximum value

3.3. ANÁLISE COMPARATIVA

Ao longo deste capítulo foi possível observar várias soluções nos principais fabricantes de autómatos a nível mundial. Quase todos têm interfaces gráficos que permitem ajudar a parametrização e definição do sistema difuso, exceto a Schneider que apenas tem uma biblioteca com blocos para usar na programação. De um ponto de vista da análise teórica dos manuais, todos são capazes de gerir as funções de pertença de entrada e de saída.

Tabela 6 – Comparação entre os diversos fabricantes

Função/Marca	Omron	Siemens	Rockwell	Schneider
Número de entradas difusas	8	8	Sem limite	Sem limite
Número Saídas difusas	2	4	Sem limite	Sem limite
Número de regras difusas	64	200/2000	Sem limite	Sem limite
Número de etiquetas	5	-	Sem limite	9
Método de inferência	Max-Min OR lógicos	MAX-Min/MAX_PROD	Mamdami/Aritmética Fuzzy	FUZ_MIN/FUZ_MAX FUZ_PROD/FUZ_SUM
Cálculo da saída	Centro de gravidade	Centro de gravidade	Centro de gravidade	-
Saída quando regras não satisfeitas	Constante ou valor anterior	-	-	-
Gráfico funções pertença	Sim	Sim	Sim	Não
Gráfico 2D saída	Sim	Sim	Sim	Não
Gráfico 3D saída	Sim	Sim	Sim	Não

As regras do sistema podem ser criadas de uma forma mais simples em alguns e mais complexas noutros. Na Tabela 6 é apresentada uma análise comparativa entre as várias soluções dos fabricantes.

Uma das limitações de todos os fabricantes é apenas estar disponíveis para autómatos de gamas mais elevadas, ficando de fora as gamas mais baixas, que nos dias de hoje já têm capacidades de processamento elevadas.

Outra das lacunas dos fabricantes é que só disponibilizam os *softwares*/Bibliotecas de uma forma paga, não estando incluída nas versões base dos *softwares* de programação.

4. NORMA IEC 61131-7 PROGRAMAÇÃO DE SISTEMAS LÓGICOS DIFUSOS

Na norma IEC 61131-7 (Anexo C) é definida uma linguagem para a programação de aplicações de controlo difuso utilizando controladores programáveis.

O intuito da norma IEC 61131-7 é oferecer aos fabricantes e aos utilizadores um entendimento comum bem definido dos meios básicos para integrar aplicações de controlo difuso nas linguagens de controladores programáveis de acordo com a norma IEC 61131-3 (Anexo B), bem como a possibilidade de trocar programas de controlo difuso entre os diferentes sistemas de programação. Face a importância da norma, considerou-se relevante a transcrição para português, apresentadas neste capítulo.

4.1. INTEGRAÇÃO NO CONTROLADOR PROGRAMÁVEL

As aplicações de controlo difuso programadas na Linguagem de controlo difuso (FCL), conforme secção 4.2, devem ser encapsuladas em Blocos de Função (FB), ou programas,

como definido na norma IEC 61131-3. O conceito de blocos de função e tipos de FB definidos na norma IEC 61131-3 são aplicados à norma IEC 61131-7.

Os tipos de Blocos de Função definidos na linguagem de controlo difuso devem especificar os parâmetros de entrada e saída, assim como as declarações e regras de controlo difuso.

As instâncias correspondentes dos Blocos de Função devem conter dados específicos das aplicações de controlo difuso.

Os blocos de função definidos na linguagem de controlo difuso FCL podem ser usados em programas e blocos de função escritos em qualquer uma das linguagens da IEC 61131-3, por exemplo, diagrama *Ladder*, Lista de instruções, etc. O tipo de dados dos parâmetros de entrada e saída do bloco de função ou programa escrito em FCL deve ser igual aos correspondentes no ambiente onde são invocados, como ilustrado na Figura 49.

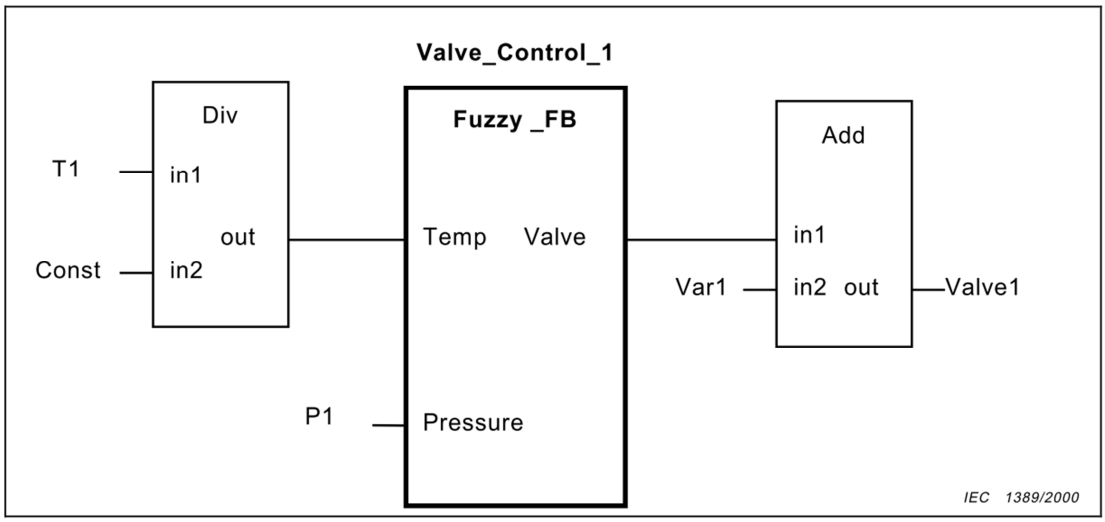


Figura 49 – Exemplo de um bloco de função de controlo difuso em representação de diagrama de bloco de função (FBD)(Anexo C)

Neste exemplo, Valve_Control_1 é um Bloco de Função definido pelo utilizador do tipo Bloco de função Fuzzy_FB. O bloco de função do tipo Fuzzy_FB pode ser programado na linguagem de programação FCL, de acordo com a secção 4.2. O bloco de função Fuzzy_FB é usado no exemplo num programa ou bloco de função representado na linguagem gráfica FBD da IEC61131-3.

4.2. LINGUAGEM DE CONTROLO DIFUSO FCL

Nesta secção é abordada a forma como se deve efetuar a programação utilizando a notação segundo a norma IEC61131-7.

4.2.1. TROCA DE PROGRAMAS DE CONTROLO DIFUSO

A definição da Linguagem de Controlo Difuso FCL baseia-se nas definições das linguagens de programação da norma IEC 61131-3. A interação do algoritmo de controlo difuso com o seu ambiente de programação faz com que seja "escondido" do programa. O algoritmo de controlo difuso é, portanto, representado externamente como um bloco de função de acordo com a norma IEC 61131-3. Os elementos necessários para descrever as partes linguísticas internas do Bloco de funções de controlo difuso como funções de pertença, regras, operadores e métodos devem ser definidos de acordo com esta secção. Os elementos de linguagem de controlo difuso FCL criam um padrão, para uma representação comum, para o intercâmbio de dados entre as ferramentas de configuração de controlo difuso de diferentes fabricantes mostrados na Figura 50. Usando esta representação comum, cada fabricante de controladores programáveis pode manter o seu *hardware*, editores de *software* e compiladores.

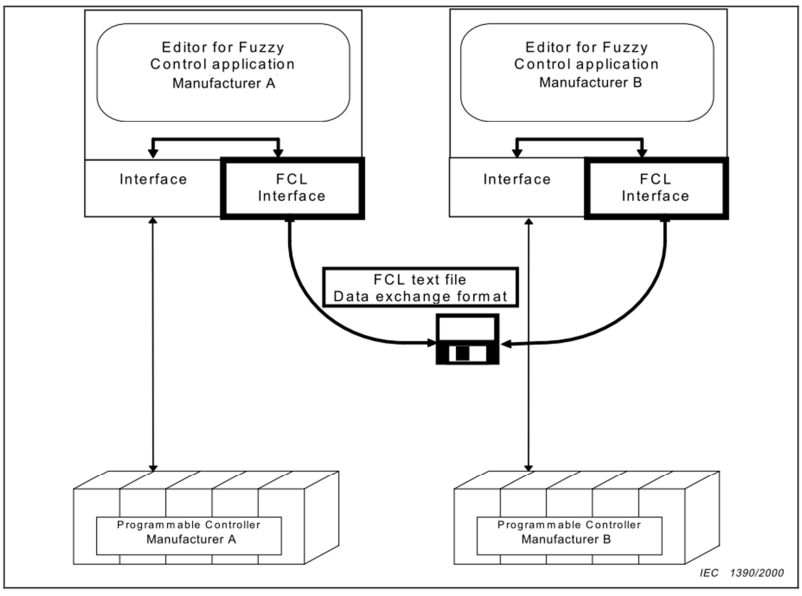


Figura 50 – Troca de programas em linguagem de controlo difuso FCL)(Anexo C)

O fabricante só tem que implementar a interface de dados no seu editor, com um compilador capaz de processar as palavras chave definidas na norma. O cliente seria capaz de trocar projetos de controlo difuso entre diferentes fabricantes.

4.2.2. INTERFACE DO BLOCO DE FUNÇÃO

De acordo com a subsecção 4.2.1, a visão externa de um bloco de função difuso necessita que sejam usados os seguintes elementos padrão da IEC61131-3:

```
FUNCTION_BLOCK function_block_name  Bloco de função
VAR_INPUT              Declaração dos parâmetros de entrada
variable_name: data_type;
....
END_VAR
VAR_OUTPUT             Declaração dos parâmetros de saída
variable_name: data_type;
....
END_VAR
....
VAR                    Variáveis locais
variable_name: data_type;
END_VAR
END_FUNCTION_BLOCK
```

Com estes elementos da linguagem, é possível descrever a interface de um bloco de função. A interface do bloco de função é definida com parâmetros que podem ser passados para dentro e para fora do bloco de função. O tipo de dados destes parâmetros deve ser definido de acordo com a norma IEC 61131-3.

A Figura 51 mostra o exemplo da declaração de um bloco de função em Linguagem estruturada (ST) e num diagrama de blocos de função.

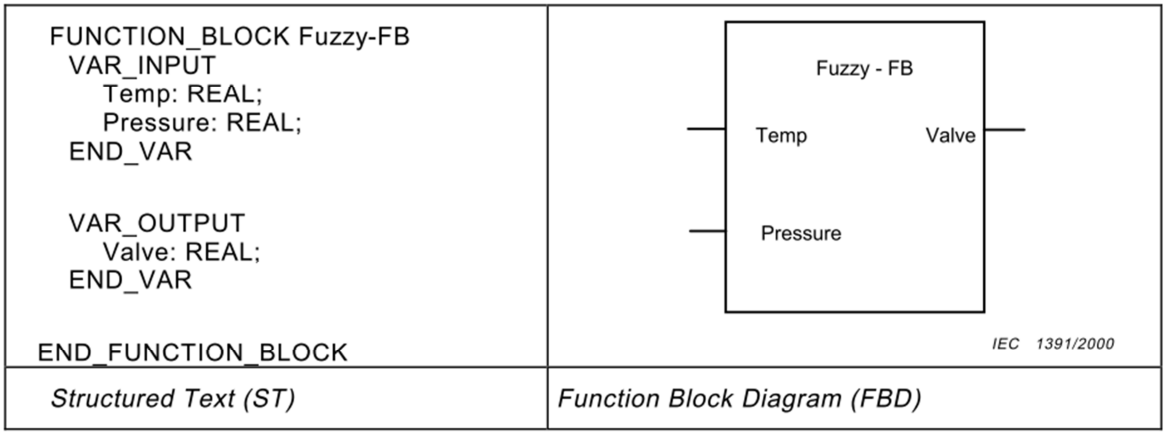


Figura 51 – Exemplo do interface declarado nas linguagens ST e FBD)(Anexo C).

4.2.3. FUSIFICAÇÃO

Os valores das variáveis de entrada têm de ser convertidos em níveis de pertença para as funções de pertença definidos na variável. Esta conversão é descrita entre as expressões FUZZIFY e END_FUZZIFY.

```
FUZZIFY variable_name
TERM term_name:= membership_function;
....
END_FUZZIFY
```

Depois da palavra-chave FUZZIFY, é definido o nome da variável, que é usado para a fusificação. A variável foi definida anteriormente na secção VAR_INPUT. Esta variável linguística deve ser descrita por um ou mais termos linguísticos. Os termos linguísticos introduzidas pela palavra-chave TERM descritos por funções de pertença, a fim de fusificar a variável. A função de pertença é uma função linear definida por trechos e definida por uma tabela de pontos.

```
membership_function ::= (point i), (point  j), ...
```

Cada ponto representa um par de valores da variável e os níveis de pertença aparecem separados por uma vírgula. Os pares estão entre parênteses e separados por vírgulas.

```
point i ::= value of input i | variable_name of input i ,
value i of membership degree
```

Com esta definição, todos os elementos simples, por exemplo a rampa e um triângulo, podem ser definidos. A ordem dos pontos deve ser introduzida por ordem de valor crescente da variável. A função de pertença é linear entre pontos sucessivos. O nível de pertença, para cada termo, por conseguinte, é calculado a partir do valor de entrada por interpolação linear entre os dois pontos adjacentes da função de pertença. O número mínimo de pontos é de dois.

Exemplo da função de pertença com três pontos para o termo linguístico "warm":

```
TERM warm:= (17.5, 0.0) (20.0, 1.0) (22.5, 0.0);
```

Se o valor de uma variável linguística é menor do que o primeiro ponto de referência na tabela de consulta, todos os valores abaixo do primeiro ponto na tabela de consulta devem ter o mesmo grau de pertença, tal como definido no primeiro ponto. Se o valor de uma

variável linguística é maior do que o último ponto de base na tabela de consulta, todos os valores maiores do que o último ponto na tabela de consulta têm o mesmo grau de adesão, tal como definido no último ponto (ver Figura 52).

```
Definition of ramps:

FUZZIFY temp;
   TERM cold:=        (3, 1) (27, 0);
   TERM warm:=        (3, 0) (27, 1);
END_FUZZIFY
```

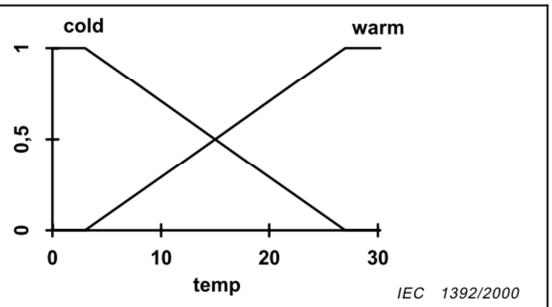


Figura 52 – Definição de funções de pertença em rampa)(Anexo C)

A fim de se adaptar a aplicação de controlo difuso *online*, os pontos da base nas funções de pertença podem ser modificados. Isto pode ser feito usando as variáveis que são a entrada para o bloco de função. Essas variáveis devem ser declaradas na seção VAR_INPUT do bloco de função. Um exemplo para a utilização de variáveis para a definição dos pontos para as funções de pertença é dado na Figura 53.

```
VAR_INPUT
        temp: REAL;                         (* this input shall be fuzzified *)
        pressure: REAL;                     (* this input shall be fuzzified *)
        bp_warm1, bp_warm2: REAL;           (* these inputs are for on-line adaptation *)
END_VAR
FUZZIFY temp
        TERM warm:= (bp_warm1, 0.0), (21.0, 1.0), (bp_warm2, 0.0);
..
END_FUZZIFY
                                                                    IEC 1393/2000
```

Figura 53 – Exemplo de uso de variáveis em funções de pertença)(Anexo C)

4.2.4. DESFUSIFICAÇÃO

Uma variável linguística para uma variável de saída tem de ser convertida num valor. Esta conversão é descrita entre as expressões DEFUZZIFY e END_DEFUZZIFY. Depois da palavra-chave DEFUZZIFY, é designada a variável que será usada para a desfusificação. Esta variável foi definida anteriormente na secção VAR_OUTPUT.

A definição dos termos linguísticos foi definida na subsecção 4.2.3.

```
DEFUZZIFY variable_name
       RANGE(min..max);
       TERM term_name:= membership_function;
       defuzzification_method;
       default_value;
END_DEFUZZIFY
```

Os *Singletons*, ou picos, são funções de pertença especiais utilizadas para as saídas, com o objetivo de simplificar a desfusificação. Eles são caracterizados por um único valor para a expressão linguística. Na Figura 54, são apresentados exemplos de termos *singletons*.

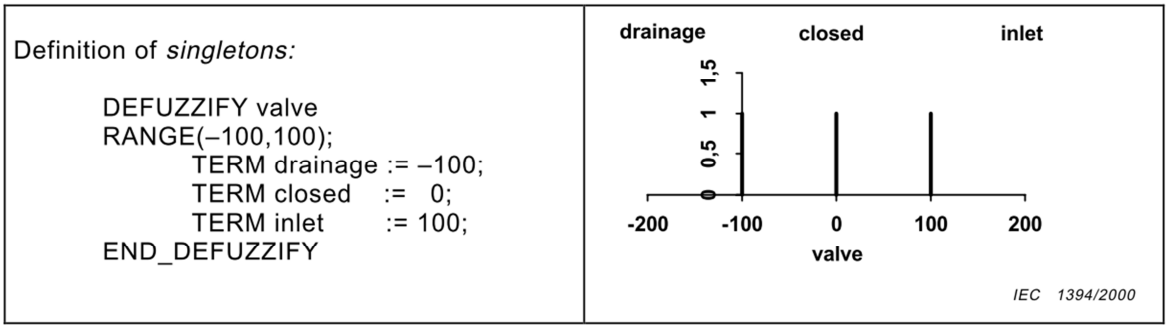


Figura 54– Exemplo de termos *singletons*)(Anexo C)

O método de desfusificação será definido pelo método dos elementos de linguagem.

```
METHOD: defuzzification_method;
```

São permitidos os métodos de desfusificação (e) apresentados na Tabela 7.

Tabela 7 – Métodos de desfusificação

Palavra-chave	**Explicação**
CoG	Centro de gravidade (NOTA 1)
CoGS	Centro de gravidade para Singletons
CoA	Centro da área (NOTA 2 e 3)
LM	Máximo mais à esquerda (NOTA 4)
RM	Máximo mais à direita (NOTA 4)
NOTA 1 – O centro de gravidade é equivalente ao centroide da área NOTA 2 – O centro da área é equivalente à bissetriz da área NOTA 3 – CoA não se aplica se forem usados singletons NOTA 4 – Os métodos de desfusificação LM e RM são assimétricos em relação à origem	

Tabela 8 – Fórmulas dos métodos de desfusificação

COG	$U = \frac{\int_{Min}^{Max} u\ \mu(u)\, du}{\int_{Min}^{Max} \mu(u)\, du}$
COGS	$U = \frac{\sum_{i=1}^{p} [u_i \mu_i]}{\sum_{i=1}^{p} [\mu_i]}$
COA	$U = u', \int_{Min}^{u'} \mu(u)du = \int_{u'}^{Max} \mu(u)du$
RM	$U = \sup(u'), \mu(u') = \sup_{u \in [Min, Max]} \mu(u)$
LM	$U = \inf(u'), \mu(u') = \inf_{u \in [Min, Max]} \mu(u)$

Onde

U	é o resultado da desfusificação;
u	é a variável de saída;
p	é o número de singletons;
μ	é a função de pertença de conjuntos difusos acumulados;
i	é o índice ;
Min	é o mínimo valor para a desfusificação definido num intervalo No caso dos singletons, Min= -infinito;
Max	é o máximo valor para a desfusificação definido num intervalo No caso dos singletons, Max= +infinito;
sup	é o maior valor;
inf	é o menor valor;

Se o nível de pertença é 0 para todos os termos linguísticos de uma variável de saída, o que significa: nenhuma regra para esta variável está ativa. Nesse caso, o método de desfusificação não é capaz de gerar uma saída válida. Portanto, é permitido definir um valor padrão para a saída. Esse valor padrão é o valor para a variável de saída apenas no caso em que nenhuma regra foi ativada.

```
DEFAULT:= value | NC;
```

Após a palavra-chave DEFAULT, o valor padrão deve ser especificado. Caso contrário, a palavra-chave NC (sem alteração (*no change*)) devem ser especificados para indicar que a

saída deve permanecer inalterada, se nenhuma regra foi ativada. O intervalo é uma especificação de um valor mínimo e um valor máximo separada por dois pontos.

```
RANGE:= (minimum value .. maximum value);
```

O RANGE (intervalo) é usado para limitar cada função de pertença para o intervalo de cada variável de saída. Se são usados *singletons* para funções de pertença de saída, o RANGE não tem efeito. Se não houver um intervalo definido, o intervalo padrão deve ser o intervalo do tipo de dados da variável especificada na IEC 61131-3.

4.2.5. BLOCO DE REGRAS

A inferência do algoritmo difuso deve ser definida num ou mais blocos de regras. Para um manuseamento adequado e para atender a possibilidade de dividir a base de regras em diferentes módulos, é permitido o uso de vários blocos de regras. Cada bloco regra deve ter um nome exclusivo.

As regras devem ser definidas entre as palavras-chave RULEBLOCK e END_RULEBLOCK.

```
RULEBLOCK
       ruleblock_name
       operator_definition;
[      activation_method;]
       accumulation_method;
       rules;
END_RULEBLOCK
```

Os operadores difusos são usados dentro do bloco de regras.

Para satisfazer as leis de Morgan, o algoritmo para os operadores AND e OR deve ser usado aos pares, por exemplo, MAX deve ser usado para o OR e MIN é usado para o AND, como se pode ver na Tabela 9.

```
operator_definition ::= operator:  algorithm
```

Tabela 9 – Algoritmos usados aos pares)(Anexo C)

operator **OR**		operator **AND**	
keyword for **Algorithm**	**Algorithm**	keyword for **Algorithm**	**Algorithm**
MAX	Max ($\mu_1(x)$, $\mu_2(x)$)	MIN	Min($\mu_1(x)$, $\mu_2(x)$)
ASUM	$\mu_1(x)+\mu_2(x) - \mu_1(x)\ \mu_2(x)$	PROD	$\mu_1(x)\ \mu_2(x)$
BSUM	Min(1, $\mu_1(x) + \mu_2(x)$)	BDIF	Max (0, $\mu_1(x) + \mu_2(x)$ -1)

Um exemplo de bloco de regras:

```
RULEBLOCK first
       AND: MIN;
       ..
END_RULEBLOCK
RULEBLOCK second
       AND: PROD;
       ..
END_RULEBLOCK
```

O seguinte elemento linguístico define o método de activação:

```
ACT: activation_method;
```

Os métodos de ativação que podem ser usados estão apresentados na Tabela 10.

Tabela 10 – Métodos de ativação)(Anexo C)

Name	**Keyword**	**Algorithm**
Product	PROD	$\mu_1(x)\ \mu_2(x)$
Minimum	MIN	Min($\mu_1(x)$, $\mu_2(x)$)

NOTA: O método de ativação não é relevante para os *singletons*.

O método de acumulação é definido pelo elemento linguístico seguinte:

```
ACCU: accumulation_method;
```

Os métodos de acumulação que podem ser usados estão apresentados na Tabela 11.

Tabela 11 – Métodos de acumulação)(Anexo C)

Name	Keyword	Formula
Maximum	MAX	$\mathrm{MAX}\ (\mu_1(x), \mu_2(x))$
Bounded sum	BSUM	$\mathrm{MIN}\ (1, \mu_1(x) + \mu_2(x))$
Normalized sum	NSUM	$\dfrac{\mu_1(x) + \mu_2(x)}{\mathrm{MAX}\ (1, \mathrm{MAX}_{x' \in X}\ (\mu_1(x') + \mu_2(x')))}$

As entradas de um bloco de regras são variáveis linguísticas com um conjunto de termos linguísticos. Cada termo tem um nível de pertença que lhe é atribuída. As regras são definidas dentro do bloco regra. Cada um começa com a palavra-chave RULE seguida por um nome para a regra e deve ser terminado por um ponto e vírgula. Cada regra tem um número único dentro do bloco de regras.

```
RULE numbers: IF condition THEN conclusion [ WITH weighting
factor];
```

A regra em si deve começar com a palavra-chave IF seguida pela condição. Após a condição, segue-se a conclusão, começando com a palavra-chave THEN. É permitido combinar várias subcondições e variáveis de entrada numa regra. O objetivo das variáveis é permitir que os níveis de pertença difusos sejam importados para o bloco de função difuso. Todos eles devem ser definidos entre as palavras-chave IF e THEN, e combinados pelos operadores com as palavras-chave AND, OR ou NOT.

A prioridade dos operadores (ver Tabela 12) é tratada de acordo com a álgebra booleana dada na Tabela 9.

Tabela 12 – Prioridade dos operadores)(Anexo C)

Priority	operator
1	( ) parenthesis
2	NOT
3	AND
4	OR

Exemplo simplificado para uma regra:

```
RULE 1: IF subcondition1 AND variable1 OR variable2 THEN
conclusion;
```

A operação OR pode ser implementada através da definição de duas regras:

```
RULE 3: IF subcondition 1 OR subcondition 2 THEN conclusion;
Substituida por:
RULE 3a: IF condition 1 THEN conclusion;
RULE 3b: IF condition 2 THEN conclusion;
```

A subcondição começa com o nome da variável linguística seguida pela palavra-chave IS com um NOT opcional e um termo linguístico da variável linguística usada pela condição.

```
Subcondition := linguistic_variable IS [NOT] linguistic_term
```

Os termos linguísticos que são usados na condição devem corresponder à variável linguística na mesma condição. O termo utilizado tem de ser previamente definido com a palavra-chave TERM.

Exemplo de subcondições:

```
temp IS hot
temp IS NOT hot
```

Também é possível utilizar a palavra-chave NOT antes da Subcondição. Neste caso, podem ser utilizados parênteses.

```
IF NOT temp IS hot THEN ...   or IF NOT (temp IS hot) THEN
...
```

A conclusão pode ser dividida em várias conclusões parciais e variáveis de saída. A conclusão prévia começa com o nome de uma variável linguística seguido pela palavra-chave IS e um termo linguístico da variável linguística dada.

```
Subconclusion := linguistic_ variable IS linguistic_ term
```

Exemplo com várias conclusões parciais:

```
IF temp IS cold AND pressure IS low
THEN var1,
       valve1 IS inlet,
       valve2 IS closed;
```

Opcionalmente, é permitido dar a cada uma das conclusões parciais um fator de ponderação que é um número real com um valor entre 0 e 1, ou uma variável, para poder

ser alterada posteriormente. Isso deve ser feito por uma palavra-chave WITH seguido pelo fator de ponderação.

O fator de ponderação deve reduzir o nível de pertença (função de pertença) da conclusão parcial através da multiplicação do resultado da conclusão parcial com o fator de ponderação.

Com o objetivo de manipular os parâmetros da aplicação de controlo difuso externamente, o fator de ponderação pode ser uma variável. Neste caso, a variável tem que ser declarada na seção VAR_INPUT. Isso permite mudar o fator de ponderação durante a execução, a fim de adaptar o programa de controlo difuso de acordo com as necessidades do processo.

Se não existe a declaração WITH atribuída à conclusão parcial, deverá ser assumido um fator de ponderação padrão de 1.

```
IF condition THEN subconclusion [ WITH weighting_factor]
subconclusion;
```

Um exemplo de um fator de ponderação constante:

```
IF temp IS cold AND pressure IS low THEN
       valve1 IS inlet WITH 0.5,
       valve2 IS closed;
```

Um exemplo de um fator de ponderação variável:

```
VAR_INPUT
       w_myrule1: REAL:= 0.8;
END_VAR
RULEBLOCK temp_rule
       RULE 1:  IF temp  IS cold AND pressure IS low
            THEN valve IS inlet WITH w_myrule1;
..
END_RULEBLOCK
```

4.2.6. PARÂMETROS OPCIONAIS

Para a implementação em diferentes sistemas de destino, pode ser necessário fornecer informações adicionais para o sistema, a fim de permitir a melhor possível conversão de aplicações de controlo difuso.

Essa informação adicional pode ser necessária num elemento de linguagem definido entre as palavras-chave OPTIONS e END_OPTIONS.

```
OPTIONS
       application_specific_parameters
END_OPTIONS
```

4.2.7. EXEMPLO FCL

Na Figura 55 é exemplificado um bloco de função difuso.

```
FUNCTION_BLOCK Fuzzy_FB
VAR_INPUT
        temp:     REAL;
        pressure: REAL;
END_VAR
VAR_OUTPUT
        valve:    REAL;
END_VAR
FUZZIFY temp
        TERM cold       := (3, 1)  (27, 0);
        TERM hot        := (3, 0)  (27, 1);
END_FUZZIFY
FUZZIFY pressure
        TERM low        := (55, 1) (95, 0);
        TERM high       := (55, 0) (95, 1);
END_FUZZIFY
DEFUZZIFY valve
        TERM drainage   := –100;
        TERM closed     := 0;
        TERM inlet      := 100;
        METHOD: CoGS;
        DEFAULT         := 0;
END_DEFUZZIFY
RULEBLOCK No1
        AND: MIN;
        ACCU: MAX;
        RULE 1:  IF temp IS cold AND pressure IS low   THEN valve IS inlet;
        RULE 2:  IF temp IS cold AND pressure IS high  THEN valve IS closed WITH 0.8;
        RULE 3:  IF temp IS hot AND pressure IS low    THEN valve IS closed;
        RULE 4:  IF temp IS hot AND pressure IS high   THEN valve IS drainage;
END_RULEBLOCK
END_FUNCTION_BLOCK
                                                              IEC 1395/2000
```

Figura 55 – Exemplo de um bloco de função difuso (Anexo C)

4.2.8. REGRAS DE PRODUÇÃO E PALAVRAS-CHAVE DA LINGUAGEM DE CONTROLO DIFUSO (FCL)

O Anexo A da norma IEC 61131-3 (ver Anexo B) define o método de especificação para linguagens textuais para controladores programáveis. Esse método de especificação é utilizado na norma IEC 61131-7 (ver Anexo C).

O Anexo B da IEC 61131-3 define a especificação formal de elementos de linguagem para as linguagens de programação textuais do IEC 61131-3. Para a FCL é usado o seguinte subconjunto de elementos de linguagem do anexo B da IEC 61131-3:

B.1.1 - Letras, dígitos e identificadores

B.1.2 Constantes

B.1.3 Os tipos de dados

B.1.4 Variáveis

Além dos elementos de linguagem listados acima, da norma IEC 61131-3, podem ser utilizados os elementos de linguagem descritos no Anexo D IEC61131-7 Regras de produção.

O Anexo E IEC61131-7 Palavras-chave reservadas para FCL resume todas as palavras-chave reservadas para a linguagem de programação FCL [10] [11].

5. BIBLIOTECA DE CONTROLO LÓGICO DIFUSO

Nos autómatos, para controlar um processo é necessário desenvolver um programa para cada sistema a controlar. A tarefa pode ser facilitada/automatizada se se conseguir criar blocos de função para serem utilizados mais tarde, dispensando assim um novo desenvolvimento. Por sua vez, estes blocos podem ser organizados numa biblioteca para ser mais fácil o controlo da versão, estando também centralizados num só sítio. Desta forma conseguimos agrupar os blocos por tipo de função. Neste caso particular vamos encontrar na biblioteca todos os blocos necessários para efetuar o controlo difuso num PLC.

Na seção 5.1 encontra-se uma descrição da plataforma Codesys e da sua versatilidade. Esta plataforma está embebida dentro do *software* Somachine da Schneider Electric. Pode-se encontrar no Anexo F uma descrição do interface usado para a criação da biblioteca.

A linguagem de programação usada foi o ST (Linguagem estruturada) respeitando a norma IEC61131-3. Desta forma permite a portabilidade dos blocos para outros sistemas que não usem Codesys, mas respeitem a mesma norma.

Na Tabela 13 podemos analisar o resumo de todas as funções disponíveis na biblioteca

Os blocos criados garantem o correto funcionamento do controlador difuso apesar de estar limitado nos métodos usados na fusificação, inferência e desfusificação.

Tabela 13 – Funções disponíveis na biblioteca de controlo lógico difuso

Operação	Bloco de função	Descrição
Fusificação	Fuzzify_9_terms	Fusificação até 9 termos de variáveis da entrada
Inferência	Rule_and_max	Operador AND com saída de valor máximo
	Rule_and_min	Operador AND com saída de valor mínimo
	Rule_or_max	Operador OR com saída de valor máximo
Desfusificação	Defuzz_ST	Desfusificação até 9 termos por *singletons*

5.1. PLATAFORMA CODESYS

O CoDeSys é o principal *software* de automação, com base na norma IEC 61131-3, que é independente dos fabricantes de *hardware*. Com uma forte intuição de mercado e profundo conhecimento tecnológico, o CoDeSys transformou-se numa ferramenta de confiança, inovadora para todas as necessidades de *software* de automação, com uma presença no mercado em todo o mundo há quase 20 anos. Todos os produtos CoDeSys são criados pela 3S-Smart Software Solutions de acordo com os processos de qualidade certificados. Equipas especializadas são responsáveis pela constante manutenção e desenvolvimento. A 3S-Smart Software Solutions contribui de uma forma significativa para a normalização internacional da norma IEC 61131-3, através da cooperação em comitês importantes.

Na Figura 1 – Arquitetura CoDeSys, apresentada na página 2, podem ser observadas várias componentes integrantes do *software*.

O CoDeSys é desenvolvido e comercializado pela empresa alemã de *software* 3S-Smart Software Solutions localizada na cidade bávara de Kempten. A versão 1.0 foi lançada em 1994. O termo CoDeSys é uma sigla e significa COntroller DEvelopment SYStem. As licenças CoDeSys são gratuitas e podem ser instalados legalmente sem proteção contra

cópia em novas estações de trabalho. A ferramenta de *software* abrange diferentes aspectos da tecnologia de automação industrial, apenas com um interface.

O *software* Codesys divide-se em seis ferramentas muito úteis na área de automação como engenharia, *runtime*, visualização, redes de campo, controlo de eixos e CNC e segurança.

No Anexo G pode ser encontrado uma listagem de todos os fabricantes que usam Codesys. A listagem é extensa, o que garante o interesse em ser desenvolvida a biblioteca de controlo difuso [12].

5.1.1. ENGENHARIA

Entende-se por engenharia a parte que respeita ao desenvolvimento e programação das aplicações.

O sistema de desenvolvimento IEC 61131-3 CoDeSys, para programação de dispositivos de automação inteligentes, é o coração da plataforma de engenharia. O *software* oferece uma variedade de funções de engenharia amigas do utilizador para fazer o seu processo de desenvolvimento mais rápido e mais eficiente.

Podem ser projetadas aplicações IEC 61131-3 por técnicos e engenheiros de software que vão desde as linguagens LD a UML numa plataforma extensível. A plataforma moderna de desenvolvimento de acordo com a IEC 61131-3 contém editores padrão compatíveis, bem como opções completas de depuração. Existem compiladores integrados para todas as plataformas de CPU importantes (incluindo ARM / Cortex, X86 / PentiumX, Power Architecture, TriCore), otimizados para aplicações industriais. Existem ferramentas adicionais para conveniência de programação de linguagem de alto nível. O *software* é modularmente expansível através do desenvolvimento *plugins* por parte dos utilizadores.

O código de fonte não pode ser alterado e, portanto, é protegido contra acessos não autorizados.

As cinco linguagens de programação, definidas na IEC 61131-3, para a programação de aplicativos estão disponíveis no ambiente de desenvolvimento CoDeSys:

- IL (lista de instruções) é uma linguagem de programação idêntica à linguagem assembly

- ST (texto estruturado) é semelhante à programação em PASCAL ou C
- LD (diagrama *Ladder*) permite ao programador combinar virtualmente contatos e bobines de relés
- FBD (diagrama de blocos de funções), permite ao utilizador programar rapidamente, expressões booleanas e expressões analógicas, interligando os blocos entre si
- SFC (controlo sequencial de funções) é conveniente para os processos de programação sequenciais e/ou fluxogramas

O Codesys dispõe ainda de um editor gráfico adicional disponível não definido na norma IEC:

- CFC (Continuous Function Chart) é uma espécie de editor FBD à mão livre. Ao contrário do editor FBD onde as conexões entre entradas e saídas, os operadores são definidos automaticamente, neste caso têm que ser desenhadas pelo utilizador. Todas as caixas podem ser colocadas livremente, o que torna possível programar ciclos de *feedback*, sem variáveis temporárias.

5.1.2. *RUNTIME*

Com o objetivo de fazer um dispositivo programável compatível com o Sistema de Desenvolvimento CoDeSys IEC 61131-3, este tem que ser equipado com o *software* certo: o SoftPLC Runtime System CoDeSys Control. Este sistema de *runtime* transforma qualquer dispositivo embebido ou PC num controlador IEC 61131-3 completo. Mais ainda, o Sistema Runtime CoDeSys Control oferece importantes funções adicionais para que o dispositivo controlador seja capaz de se comunicar com outros componentes de automação.

Como utilizador CoDeSys que quer programar um controlador compatível, não é necessária a preocupação com o sistema de *runtime*. Todos os dispositivos listados no diretório de dispositivos CoDeSys são equipados com o Runtime System CoDeSys Control e estão prontos a ser utilizados. O que significa que o dispositivo não requer qualquer configuração adicional, mas pode ser programado com CoDeSys imediatamente.

Devido às propriedades modulares e escaláveis do Runtime System CoDeSys Control um fabricante do dispositivo pode adaptar o Runtime System para atender às mais diferentes plataformas e exigências.

5.1.3. VISUALIZAÇÃO

A componente de visualização do Codesys permite o desenvolvimento de páginas (ecrãs) diretamente a partir do sistema de desenvolvimento com base na norma IEC61131-3. Este sistema tem um editor gráfico integrado que permite a criação de ecrãs de visualização simples ou complexos conforme a necessidade da aplicação. A integração do editor gráfico permite o desenvolvimento das visualizações paralelo com o programa do PLC. O acesso às variáveis usadas no programa do PLC é direto, não havendo a necessidade de exportar as variáveis do PLC, importar para o HMI nem a constante atualização das mesmas. A aplicação e visualização são verificadas em simultâneo. As visualizações criadas podem ter propósitos diversos, como por exemplo, usadas no comissionamento ou num terminal na máquina.

5.1.4. REDES DE CAMPO

No campo das redes de campo o Codesys oferece configuradores integrados para diversas redes de campo, como por exemplo, PROFIBUS, PROFINET, EtherCAT, CANopen, J1939, DeviceNet, EtherNet/IP, sercos, AS-i, Modbus, IO-Link.

A possibilidade de configuração das redes de campo no mesmo *software* de programação, permite reduzir o risco de existência de erros.

5.1.5. CONTROLO DE EIXOS E CNC

Desde o simples movimento de um eixo até à interpolação de caminhos CNC multidimensionais, o Codesys SoftMotion permite o desenvolvimento sem ter que deixar o interface do sistema. Em contraste com os sistemas clássicos de controlo de eixos, o Codesys fornece os requisitos de funcionalidade sobre a forma de *toolkit* totalmente integrada do sistema de desenvolvimento do PLC. A programação do controlo dos movimentos é independente da rede de campo ou do equipamento usado. A configuração das redes de dados e do controlo de movimento é feito de uma forma amiga do utilizador.

5.1.6. SEGURANÇA

A empresa 3S Smart Software Solutions é a verdadeira pioneira quando se trata de integrar toda uma gama de diferentes funcionalidades em uma única ferramenta. As soluções integradas SIL 2 e SIL3 de segurança na norma IEC 61131-3 CoDeSys, por exemplo, oferecem a funcionalidade completa necessária para soluções de automação seguras.

Extenso *know-how* em tecnologia de compiladores para arquiteturas de CPU de 32 bits permite a garantia de uma execução fiável do *software* de segurança. A solução mais fácil de usar e mais confiável para a combinação de redes de campo seguras e não seguras.

O Sistema Runtime CoDeSys pode ser facilmente exportado para diferentes plataformas de *hardware* e *software*, sendo que as soluções de segurança já disponíveis podem ser exportadas para novos ambientes.

5.2. BLOCO DE FUNÇÃO FUZZIFY_9_TERMS

O bloco de função Fuzzify_9_terms implementa a fusificação até 9 termos de variáveis da entrada X indicando o grau de pertença individual (saídas MD1…MD9). A variável de entrada X deve ser do tipo real. As funções de pertença são definidas através de pontos de suporte (entradas expansíveis X1 a X9) (Figura 56).

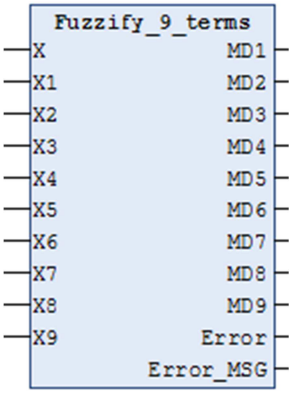


Figura 56 – Representação FBD do bloco Fuzzify_9_terms

O bloco de função implementa uma simplificação da definição de funções de pertença, essa simplificação implica as seguintes condições:

- A primeira e última função de pertença têm que ser representadas como rampas;

- As funções intermédias entre a primeira e a última são representadas sob a forma de triângulo;
- A soma entre duas funções consecutivas é sempre 1;
- A soma de todos os termos é sempre 1.

O número de pontos de suporte (X1…X9) utilizados tem que ser no mínimo dois e no máximo nove. Cada ponto de suporte gera uma função de pertença. Quando apenas estiverem definidas algumas funções de pertença as restantes tem um grau de valor 0 (por exemplo, para quatro funções de pertença são calculados quatro níveis de pertença MD1…MD4, os restantes MD5…MD9 são 0).

Tabela 14 – Descrição dos parâmetros do bloco Fuzzify_9_terms

Entrada		
Parâmetro	**Tipo de dados**	**Significado**
X	REAL	Variável linguística
X1	REAL	Ponto de suporte X1
X2	REAL	Ponto de suporte X2
X3	REAL	Ponto de suporte X3
X4	REAL	Ponto de suporte X4
X5	REAL	Ponto de suporte X5
X6	REAL	Ponto de suporte X6
X7	REAL	Ponto de suporte X7
X8	REAL	Ponto de suporte X8
X9	REAL	Ponto de suporte X9
Saída		
Parâmetro	**Tipo de dados**	**Significado**
MD1	REAL	Nível de saída da função de pertença MD1
MD 2	REAL	Nível de saída da função de pertença MD2
MD 3	REAL	Nível de saída da função de pertença MD3
MD 4	REAL	Nível de saída da função de pertença MD4
MD 5	REAL	Nível de saída da função de pertença MD5
MD 6	REAL	Nível de saída da função de pertença MD6
MD 7	REAL	Nível de saída da função de pertença MD7
MD 8	REAL	Nível de saída da função de pertença MD8
MD 9	REAL	Nível de saída da função de pertença MD9
Error	INT	Código de erro
Error_MSG	STRING	Descrição do erro

Através do bloco Fuzzify_9_terms todos os termos da variável linguística podem ser fusificados de uma só vez. As funções de pertença são determinadas através dos pontos de suporte (S1, S2, S3,…). Nas entradas do bloco podem ser encontradas, a variável linguística e os pontos de suporte às funções de pertença (Tabela 14).

As saídas são constituídas, pelo grau calculado das funções de pertença, o código de erro e a sua descrição (Tabela 14).

Na Figura 57 é possível identificar os pontos de suporte e o respetivo grau calculado das funções de pertença. Nove pontos de suporte originam 9 funções de pertença.

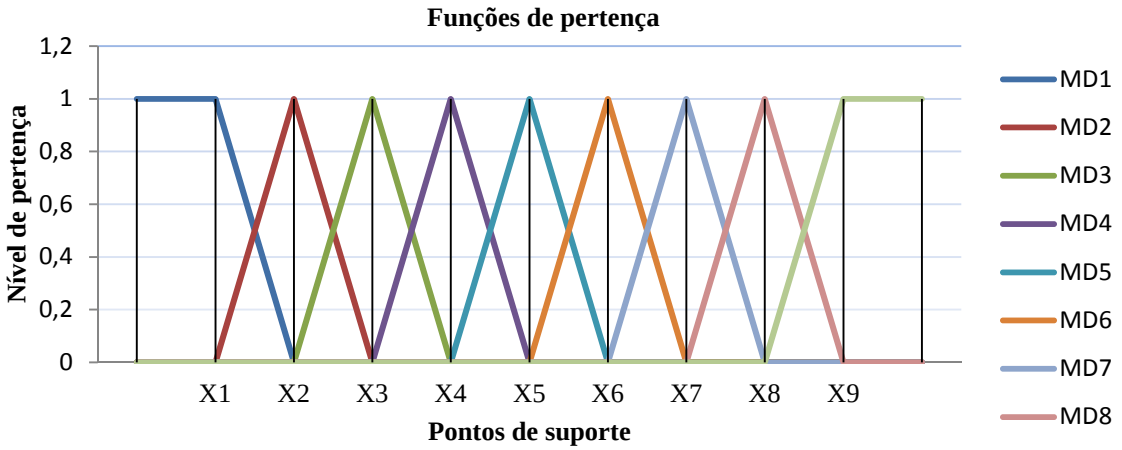


Figura 57 – Representação das funções de pertença geradas pelo bloco Fuzzify_9_terms

O cálculo das funções de pertença é efetuado com base em várias premissas. O cálculo do declive é efetuado com base nas seguintes condições:

```
IF X1 <> X2 AND X > X1 AND X < X2 THEN
    m12:=1/(X1-X2);
ELSE
    m12:=0;
END_IF
```

Sempre que o valor X se encontra os entre dois pontos de suporte, é calculado o declive da reta que liga os pontos, caso contrário o declive é nulo.

As funções de pertença de início e de fim têm características diferentes das restantes, uma vez que são rampas e não triângulos.

No caso da primeira função de pertença MD1 o cálculo é efetuado da seguinte forma:

```
//função de pertença 1
IF X <= X1 THEN
      MD1 :=1;
ELSIF X>X1 AND X<X2 AND m12<>0 THEN
      MD1:= m12*(X-X1)+1;
ELSE
      MD1 :=0;
END_IF
```

Quando o valor de X é inferior ao ponto de suporte X1 o valor calculado de MD1 é sempre 1, se o valor de X se encontra entre os pontos X1 e X2 o valor calculado respeita a equação descendente da reta que os liga. Se não for satisfeita nenhuma das condições anteriores o valor de MD1 é 0, como ilustrado na Figura 58.

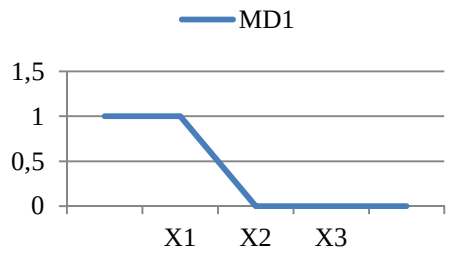


Figura 58 – Função de pertença MD1

O valor da função de pertença MD9 é calculado com base nas condições seguintes:

```
//função de pertença 9
IF X <= X8 OR X9=0 THEN
    MD9 :=0;
ELSIF X>X8 AND X<X9 AND m89<>0 THEN
    MD9:= -m89*(X-X8)+0;
ELSE
    MD9 :=1;
END_IF
```

Quando o valor de X é inferior ao ponto de suporte X9 ou o valor de X9 é igual a 0 então o valor de MD9 é igual a 0, como ilustrado na

Figura **60**. Se o valor de X se encontra entre os pontos X8 e X9 o valor calculado respeita a equação ascendente da reta que os liga. Se não for satisfeita nenhuma das condições anteriores o valor de MD9 é 1. Estas condições são demonstradas na Figura 60.

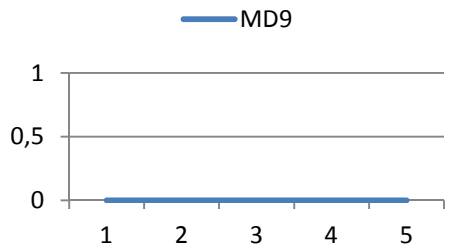


Figura 59 – Função pertença MD9 com valor nulo

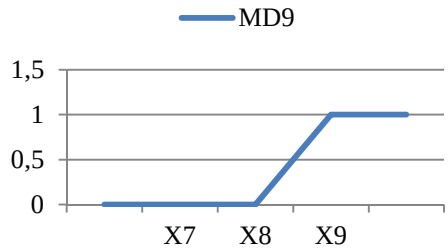


Figura 60 – Função de pertença MD9 com valor saída normal

Nas restantes funções que podem ser caracterizadas por triângulos foi desenvolvida um bloco de função para o cálculo das mesmas, ver Figura 61. No caso de uma destas funções ser a última que caracteriza o sistema, nesse caso o comportamento é igual ao da função de pertença MD9.

O cálculo das funções de pertença através do bloco de função MD_triang é explicado na subsecção 5.2.1. A chamada do bloco de função em ST dever ser efetuado com a entrada e saída dos seguintes parâmetros:

```
//função de pertença 2
MD_2(
    Sinput:=X,
    S1:=X1,
    S2:=X2,
    S3:=X3,
    MD=>MD2
);
```

A definição da função MD_2 implica definir um valor de entrada para os 3 pontos do triângulo no caso desta função exemplo X1, X2, X3 (S1, S2, S3), um valor de entrada do valor atual X (Sinput). A saída MD2 (MD) também é definida e por sua vez usada no bloco de função Fuzzify_9_terms.

5.2.1. BLOCO DE FUNÇÃO MD_TRIANG

O bloco de função MD_Triang (Figura 61) implementa o cálculo das funções de pertença que não são as funções dos extremos do bloco de função Fuzzify_9_terms.

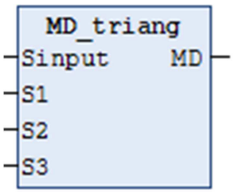


Figura 61 – Representação FBD do bloco MD_triang

O bloco aceita três pontos de suporte que podem ser definidos da mesma forma que no bloco principal. Na Tabela 15 encontram-se descritas as variáveis possíveis de entrada e saída do bloco.

Tabela 15 – Descrição dos parâmetros do bloco MD_triang

Entrada		
Parâmetro	**Tipo de dados**	**Significado**
Sinput	REAL	Variável linguística
S1	REAL	Ponto de suporte S1
S2	REAL	Ponto de suporte S2
S3	REAL	Ponto de suporte S3
Saída		
Parâmetro	**Tipo de dados**	**Significado**
MD	REAL	Nível de saída da função de pertença MD

O cálculo da função de pertença está sujeito a um conjunto de condições. O cálculo do declive é efetuado com base nas seguintes condições:

```
//cálculo declives
IF S1 <> S2 AND Sinput > S1 AND Sinput < S2 THEN
    s12:=1/(S1-S2);
ELSE
    s12:=0;
END_IF

IF S2 <> S3 AND Sinput > S2 AND Sinput < S3 THEN
    s23:=1/(S2-S3);
ELSE
    S23:=0;
END_IF
```

No caso das funções intermédias existem dois declives, um declive ascendente (s12) e um declive descendente (s23) correspondentes à forma geométrica do triângulo (Figura 63). No caso particular de dois pontos de suporte consecutivos serem iguais, os declives calculados serão iguais a 0 (Figura 62).

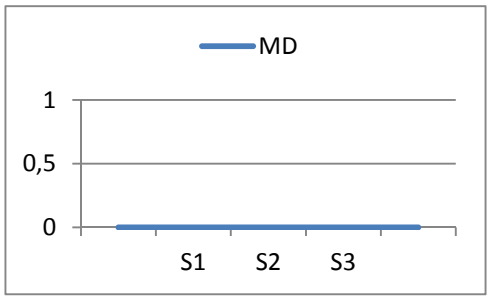


Figura 62– Função pertença MD com valor nulo

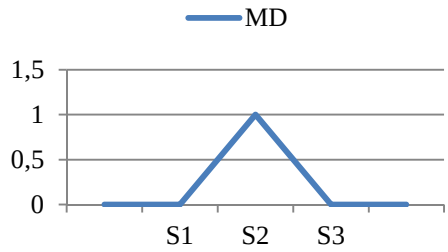


Figura 63– Função pertença MD com saída em triângulo

O cálculo da função de pertença obedece a um conjunto de condições e cálculos demonstrados no seguinte código:

```
//função de pertença x
IF Sinput <= S1 OR (Sinput>=S3 AND S3<>0) or S2=0 THEN
    MD :=0;
ELSIF Sinput>S1 AND Sinput<S2 AND S12<>0 THEN
    MD:= -S12*(Sinput-S1)+0;
ELSIF Sinput>S2 AND Sinput<S3 AND s23<>0 THEN
    MD:= s23*(Sinput-S2)+1;
ELSIF S3=0 AND Sinput>=S2 THEN
    MD:=1;
END_IF
```

Se o valor da entrada Sinput for inferior a S1, ou superior a S3 com S3 diferente de zero, ou S2 igual a 0, o valor da saída MD é igual a 0. No caso de Sinput se encontrar entre S1 e S2 ou entre S2 e S3 o valor de MD é calculado com base na equação da respetiva reta que liga os pontos S1 e S2 ou S2 e S3. No caso de S3=0 e Sinput maior que S2 o valor de MD é sempre 1 (Figura 64), representa uma rampa uma vez que é considerada a última função do conjunto.

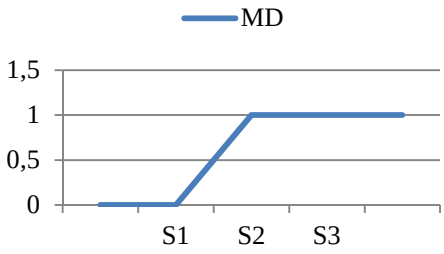


Figura 64– Função pertença MD com saída em rampa

5.3. BLOCO DE FUNÇÃO RULE_AND_MAX

O bloco de função Rule_and_max (Figura 65) é uma regra de comparação lógica AND (E) que coloca na saída o valor máximo de uma das entradas.

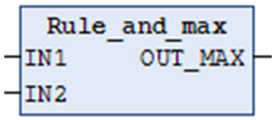


Figura 65 – Representação FBD do bloco Rule_and_max

Na Tabela 16 podemos analisar os parâmetros de entrada e saída do bloco Rule_and_max.

Tabela 16 – Descrição dos parâmetros do bloco Rule_and_max

Entrada		
Parâmetro	**Tipo de dados**	**Significado**
IN1	REAL	Entrada 1 de comparação
IN2	REAL	Entrada 2 de comparação
Saída		
Parâmetro	**Tipo de dados**	**Significado**
OUT_MAX	REAL	Saída da regra

O cálculo da saída é efetuado a partir das seguintes regras:

```
IF IN1>0 AND IN2>0 THEN
    IF IN1 > IN2 THEN
        OUT_MAX := IN1;
    ELSE
        OUT_MAX:=IN2;
    END_IF
ELSE
    OUT_MAX:=0;
END_IF
```

Quando as duas entradas tem um valor superior a 0 ativam a regra. É efetuada uma verificação se a primeira entrada IN1 é superior à segunda IN2, se sim coloca o valor de IN1 na saída OUT_MAX, caso contrário coloca o valor de IN2. Quando a regra não está ativa o valor de OUT_MAX é 0.

5.4. BLOCO DE FUNÇÃO RULE_AND_MIN

O bloco de função Rule_and_min (Figura 66) é uma regra de comparação lógica AND (E) que coloca na saída o valor mínimo de uma das entradas.

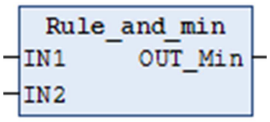


Figura 66 – Representação FBD do bloco Rule_and_min

Na Tabela 17 podemos analisar os parâmetros de entrada e saída do bloco Rule_and_min

Tabela 17 – Descrição dos parâmetros do bloco Rule_and_min

Entrada		
Parâmetro	**Tipo de dados**	**Significado**
IN1	REAL	Entrada 1 de comparação
IN2	REAL	Entrada 2 de comparação
Saída		
Parâmetro	**Tipo de dados**	**Significado**
OUT_Min	REAL	Saída da regra

O cálculo da saída é efetuado a partir das seguintes regras:

```
IF IN1>0 AND IN2>0 THEN
    IF IN1 < IN2 THEN
        OUT_Min := IN1;
    ELSE
        OUT_Min:=IN2;
    END_IF
ELSE
    OUT_Min:=0;
END_IF
```

Quando as duas entradas tem um valor superior a 0 ativam a regra. É efetuada uma verificação se a primeira entrada IN1 é inferior à segunda IN2, se sim coloca o valor de IN1 na saída OUT_Min, caso contrário coloca o valor de IN2. Quando a regra não está ativa o valor de OUT_Min é 0.

5.5. BLOCO DE FUNÇÃO RULE_OR_MAX

O bloco de função Rule_or_max (Figura 67) é uma regra de comparação lógica OR (OU) que coloca na saída o valor máximo de uma das entradas.

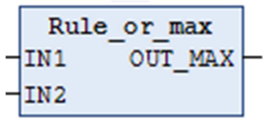


Figura 67 – Representação FBD do bloco Rule_or_max

Na Tabela 18 podemos analisar os parâmetros de entrada e saída do bloco Rule_or_max.

Tabela 18 – Descrição dos parâmetros do bloco Rule_or_max

Entrada		
Parâmetro	**Tipo de dados**	**Significado**
IN1	REAL	Entrada 1 de comparação
IN2	REAL	Entrada 2 de comparação
Saída		
Parâmetro	**Tipo de dados**	**Significado**
OUT_MAX	REAL	Saída da regra

O cálculo da saída é efetuado a partir das seguintes regras:

```
IF IN1>0 OR IN2>0 THEN
    IF IN1 > IN2 THEN
        OUT_MAX := IN1;
    ELSE
        OUT_MAX:=IN2;
    END_IF
ELSE
    OUT_MAX:=0;
END_IF
```

Quando uma das entradas tem um valor superior a 0 ativam a regra. É efetuada uma verificação se a primeira entrada IN1 é superior à segunda IN2, se sim coloca o valor de IN1 na saída OUT_MAX, caso contrário coloca o valor de IN2. Quando a regra não está ativa o valor de OUT_MAX é 0.

5.6. BLOCO DE FUNÇÃO DEFUZZ_ST

O bloco de função Defuzz_ST (Figura 68) efetua a desfusificação dos termos linguísticos através de uma representação por *singletons*.

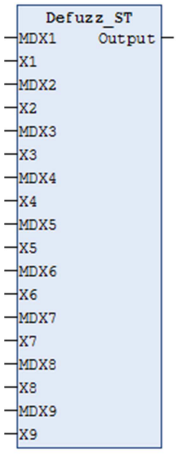


Figura 68 – Representação FBD do bloco Defuzz_ST

A posição dos *singletons* é definida através dos pontos de suporte (X1… X9). Cada termo é ponderado com a respetiva função de pertença (MDX1…MDX9). Na Tabela 19 encontramos os parâmetros de entrada e saída do bloco

Tabela 19 – Descrição dos parâmetros do bloco Defuzz_ST

Entrada		
Parâmetro	**Tipo de dados**	**Significado**
X1	REAL	Ponto de suporte X1
MDX1	REAL	Função de pertença para ponderação de X1
X2	REAL	Ponto de suporte X2
MDX2	REAL	Função de pertença para ponderação de X2
X3	REAL	Ponto de suporte X3
MDX30	REAL	Função de pertença para ponderação de X3
X4	REAL	Ponto de suporte X4
MDX4	REAL	Função de pertença para ponderação de X4
X5	REAL	Ponto de suporte X5
MDX5	REAL	Função de pertença para ponderação de X5
X6	REAL	Ponto de suporte X6
MDX6	REAL	Função de pertença para ponderação de X6
X7	REAL	Ponto de suporte X7
MDX7	REAL	Função de pertença para ponderação de X7
X8	REAL	Ponto de suporte X8
MDX8	REAL	Função de pertença para ponderação de X8
X9	REAL	Ponto de suporte X9
MDX9	REAL	Função de pertença para ponderação de X9
Saída		
Parâmetro	**Tipo de dados**	**Significado**
Output	REAL	Saída de controlo

O número de pontos de suporte e respetivas funções de pertença para ponderação podem ser definidos desde 2 a 9. Quantos mais forem definidos melhor será o controlo do sistema.

A fórmula usada para o cálculo da saída é a seguinte:

$$Output = \frac{\sum_{i=1}^{n} MDXi \times Xi}{\sum_{i=1}^{n} MDXi} \text{ com n=número de singletons e } 2 \leq n \leq 9 \qquad (2)$$

A aplicação da formula (2) à linguagem de programação está descrita de seguida:

```
IF MDX1<>0 OR MDX2<>0 OR MDX3<>0 OR MDX4<>0 OR MDX5<>0 OR
MDX6<>0 OR MDX7<>0 OR MDX8<>0 OR MDX9<>0 THEN

Output:=(MDX1*X1+MDX2*X2+MDX3*X3+MDX4*X4+MDX5*X5+MDX6*X6+MDX
7*X7+MDX8*X8+MDX9*X9)/(MDX1+MDX2+MDX3+MDX4+MDX5+MDX6+MDX7+MD
X8+MDX9);
ELSE
    Output:=0;
END_IF
```

Para garantir que o processador não entra em erro por existir uma divisão por zero é necessário garantir que todos os termos das funções de pertença não são 0. Se todos forem 0 então a saída de controlo é colocada a 0.

Na Figura 69 podemos observar um possível resultado calculado através do bloco de desfusificação. A função resultante do bloco é uma média dos máximos das funções de pertença ativas. Partindo do exemplo da Figura 69, podíamos calcular o resultado da saída da seguinte forma:

$$Output = \frac{1 \times X1 + 0{,}4 \times X2 + 0{,}6 \times X3 + 0{,}7 \times X4 + 0{,}9 \times X5 + 0{,}3 \times X6 + 0{,}3 \times X7 + 0{,}2 \times X8 + 0{,}1 \times X9}{1 + 0{,}4 + 0{,}6 + 0{,}7 + 0{,}9 + 0{,}3 + 0{,}5 + 0{,}2 + 0{,}1}$$

O resultado final varia conforme a definição dos pontos de suporte X1…X9. Alterando o valor dos pontos X consegue-se ajustar o comportamento do sistema.

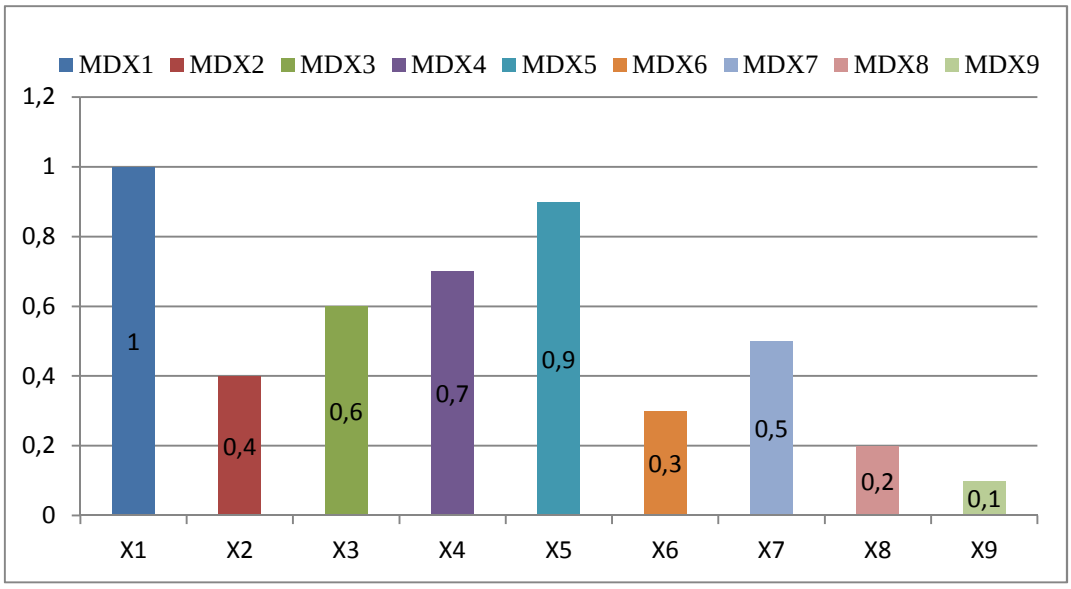


Figura 69 – Representação gráfica de um possível resultado de desfusificação

6. VALIDAÇÃO RESULTADOS

Neste capítulo podemos comprovar as dificuldades sentidas e os resultados obtidos ao longo do estudo, desenvolvimento e testes da biblioteca de controlo difuso.

Este capítulo divide-se em três secções distintas, *hardware*, *software* e resultados obtidos.

6.1. *HARDWARE*

O *Hardware* utilizado (Figura 70), é composto pelos seguintes equipamentos:

- Autómato programável Schneider Electric M258
- Variador de velocidade Schneider Electric Altivar 32 1,1 kW
- *Encoder* incremental Schneider Electric XCC 1024 Impulsos
- Motor Universal Motors 0,37 kW
- PC para programação e comissionamento

O *hardware* utilizado na realização destes ensaios foi gentilmente fornecido a título de empréstimo pela empresa INESE – www.inese.pt.

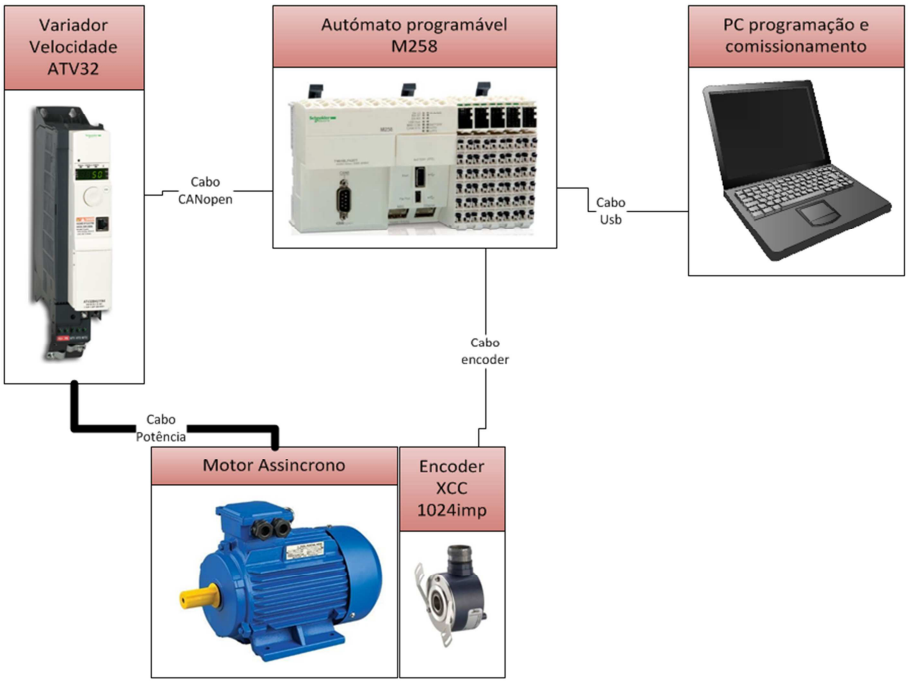


Figura 70 - Diagrama do *hardware* do sistema

Na Figura 71 encontra-se o sistema ligado por forma a validar a biblioteca de controlo difuso para PLC.

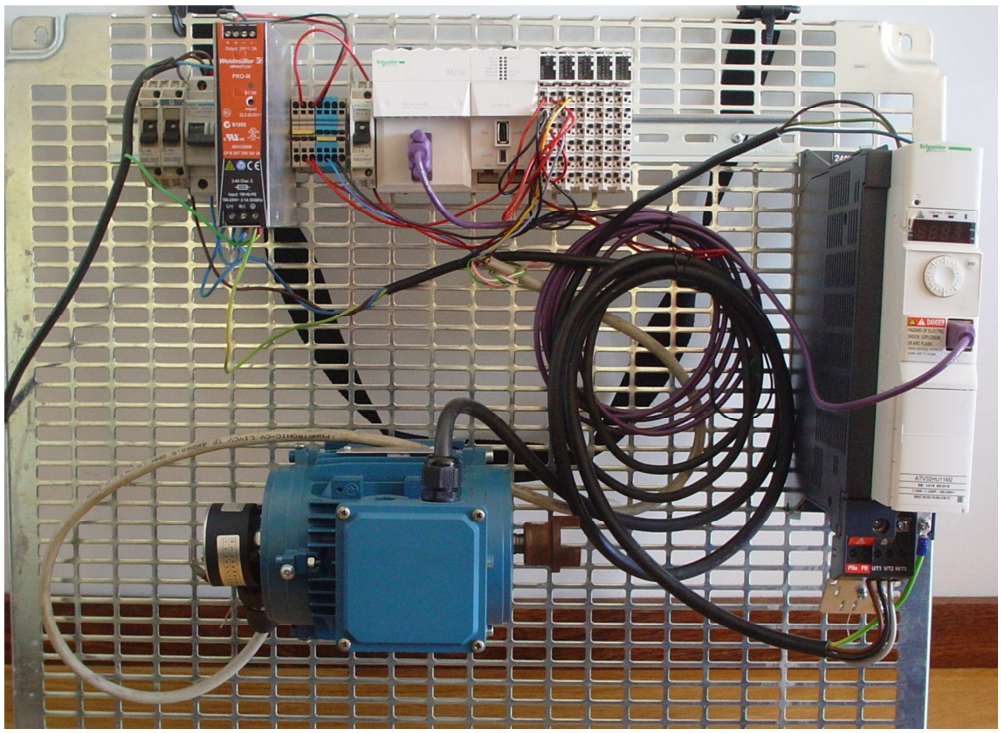


Figura 71 – Implementação do sistema para validação de resultados

6.1.1. AUTÓMATO PROGRAMÁVEL SCHNEIDER ELECTRIC M258

A gama de autómatos programáveis M258 está incluída dentro da solução Machine Struture da Schneider, que está vocacionada para controlo de máquinas industriais, sendo dotado de especificações mais exigentes em relação ao autómato que controle, por exemplo, uma instalação de uma Estação de Tratamento de Águas.

O autómato usado nesta aplicação tem a referência TM258LF42DT (Figura 72) do qual se destacam as seguintes características técnicas [13]:

- 26 Entradas digitais (10 entradas rápidas e 16 normais)
- 16 Saídas digitais (4 saídas rápidas e 12 normais)
- Relógio tempo real
- 8 Entradas parametrizáveis para contagem até 200 kHz
- Modbus *master*/*slave* RTU/ASCII ou ASCII (RS232/RS485), 300...115200 bps
- Ethernet Modbus TCP/IP *slave* (10BASE-T/100BASE-TX)
- CANopen *master* 1000 kbit/s
- Funções avançadas de registo de dados, servidor *web* e servidor FTP
- CPU de elevadas prestações (22 ns/instrução)

Figura 72 – Autómato programável Schneider Electric M258

No autómato pode ser encontrada a ligação da alimentação 24Vdc, conforme

Figura **73.**

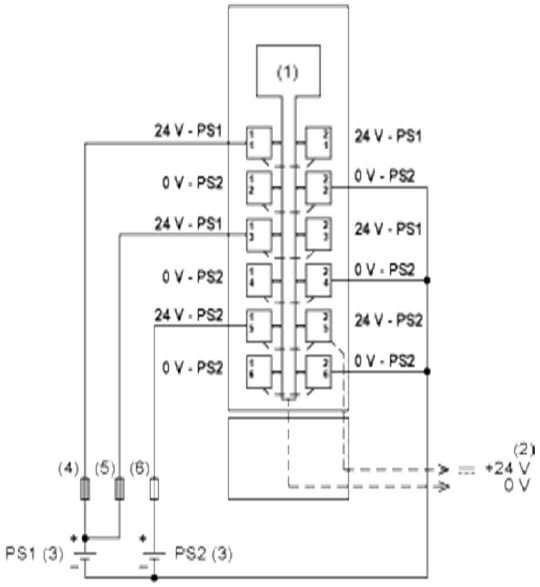


Figura 73 - Esquema de alimentação autómato

A ligação entre o autómato e o *encoder* foi realizada de acordo com o esquema de ligações representado na Figura 74, onde os canais que foram usados foram o A, B e o Z.

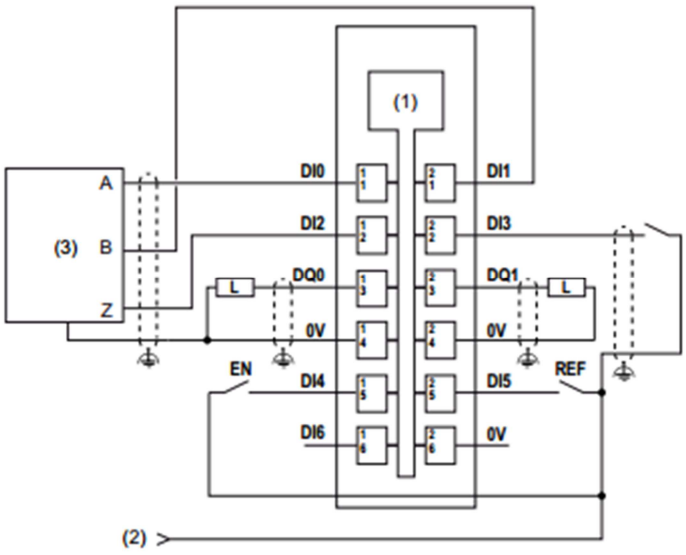
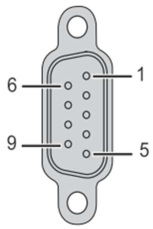


Figura 74 - Esquema de ligação do *encoder* ao autómato

Na Tabela 20 encontra-se descrita a lista de pinos de ligação necessários para a comunicação CANopen com o variador de velocidade.

Tabela 20 – Lista de pinos ligação CANopen do autómato

	Nº Pino	Sinal	Descrição
	1	Não ligado	Reservado
	2	CAN_L	Sinal de comunicação CAN Low
	3	CAN_GND	Sinal CAN 0 Vdc
	4	Não ligado	Reservado
	5	CAN_SHLD	Ligação malha de proteção EMC do cabo
	6	GND	Sinal 0 Vdc
	7	CAN_H	Sinal de comunicação CAN High
	8	Não ligado	Reservado
	9	Não ligado	Reservado

6.1.2. VARIADOR DE VELOCIDADE SCHNEIDER ELECTRIC ALTIVAR 32 1,1 KW

A gama de variadores Altivar 32, tem um formato extra fino que lhe permite ser montado lado a lado, conseguindo assim uma otimização de espaço dentro do quadro elétrico. Projetado para controlar motores assíncronos/síncronos com funcionamento em malha aberta.

O variador usado nesta aplicação tem a referência ATV32H11M2 (Figura 75), do qual se destacam as seguintes características técnicas [14]:

- Alimentação monofásica 200…240 Vac
- 1,1 kW de potência nominal máxima, 6,9 A de corrente nominal máxima
- Vários modos de controlo do motor (tensão/frequência, Vetorial, poupança de energia)
- Modbus *Slave*
- CANopen *Slave*
- Bluetooth para parametrização através de *software* SoMove através do PC
- Funções de segurança integradas (*Safe torque off, safety limited speed, Safe stop 1*)

Figura 75 – Variador de velocidade Schneider Electric Altivar 32 1,1 kW

O variador transmite a tensão e a frequência ao motor através do cabo de potência (ver Figura 76). As ordens e referências para o funcionamento do variador são recebidas por CANopen (protocolo de comunicação industrial) através de um cabo específico com a referência VW3M3805R030, com ligação SUB-D9 fêmea e RJ45.

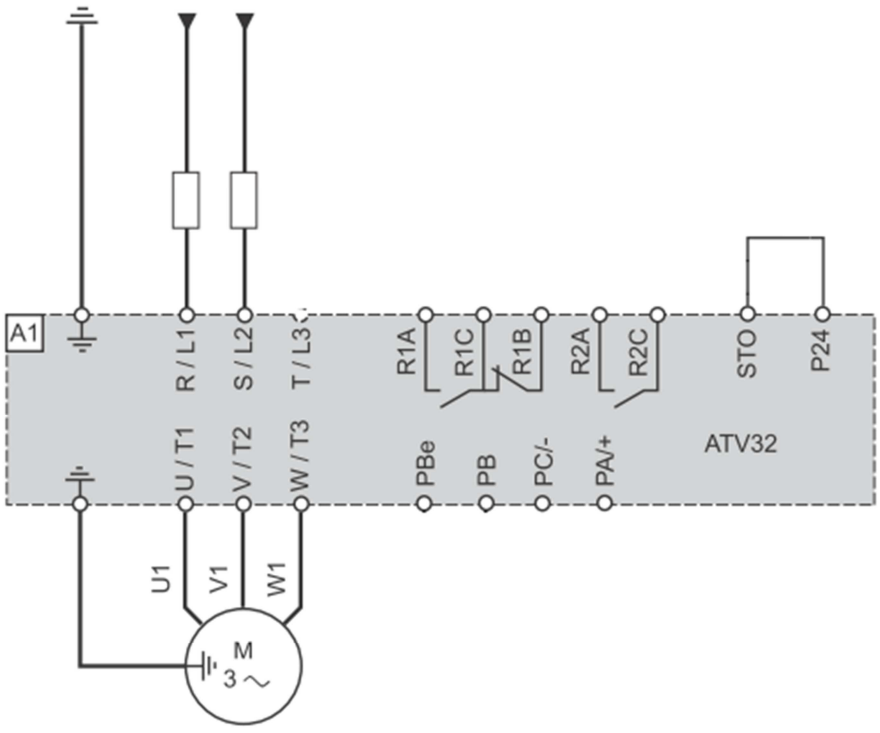
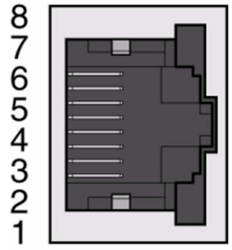


Figura 76 – Ligações potência variador velocidade

Na Tabela 21 encontra-se descrita a lista de pinos de ligação necessários para a comunicação CANopen com o autómato.

Tabela 21 – Lista de pinos ligação CANopen do variador de velocidae

	Nº Pino	Sinal	Descrição
	1	CAN_H	Sinal de comunicação CAN High
	2	CAN_L	Sinal de comunicação CAN Low
	3	CAN_GND	Sinal CAN 0 Vdc
	4	D1 – RS485 (Modbus)	Sinal de comunicação Modbus D1
	5	D0 – RS485 (Modbus)	Sinal de comunicação Modbus D0
	6	Não ligado	Reservado
	7	VP	Reservado conversor RS232/RS485
	8	Comum	Comum comunicação Modbus

6.1.3. *ENCODER* INCREMENTAL SCHNEIDER ELECTRIC XCC 1024IMP

Os *encoder*s são largamente usados na indústria, quer para controlo de velocidade de motores, fechando a malha para um controlo mais preciso, quer para aplicações de posicionamento.

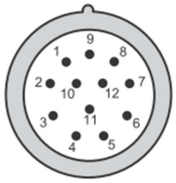
O *encoder* usado nesta aplicação tem a referência XCC1514TS11Y (Figura 77), do qual se destacam as seguintes características técnicas [15]:

- *Encoder* incrementar com resolução 1024 impulsos
- 58mm de diâmetro com veio oco de 14 mm
- Push-pull 5…30 Vdc
- Frequência máxima saída 300 kHz
- Canal A /A, B /B, 0 /0

Figura 77 – *Encoder* incremental Schneider Electric XCC 1024Imp

A ligação do *encoder* ao autómato é feita através de um cabo multilinhas que permite a alimentação 24Vdc e os três canais usados (A, B e 0), como descrito na Tabela 22.

Tabela 22 – Lista de pinos ligação *Encoder* através da ficha M23



Nº Pino	Sinal
1	/A
2	+V
3	0
4	/0
5	B
6	/B
7	R
8	A
9	R
10	0V
11	0V
12	+V

6.1.4. MOTOR UNIVERSAL MOTORS 0,37 KW

Os motores assíncronos nos dias de hoje são dos componentes mais usados na indústria, a sua eficiência e rendimento permitem ganhos de produtividade e redução de custos de manutenção e produção.

O motor usado nesta aplicação tem a referência BF31 71M-4 (Figura 78) com as seguintes características [16]:

- Ligação triângulo 220…240 Vac
- Frequência: 50 Hz
- Velocidade do motor: 1370 rpm
- Potência: 0,37 kW
- Corrente nominal: 1,92 A
- Cosφ 0,74

Figura 78 – Motor Universal Motors 0,37kW

O motor está ligado ao variador através de um cabo de potência que permite o movimento do mesmo.

6.2. *SOFTWARE*

Esta secção relativa ao *software* pode ser dividida em duas subsecções distintas, a primeira relativa à parametrização dos equipamentos/aplicação e uma segunda onde se demostra a implementação do algoritmo.

O programa completo do PLC encontra-se no Anexo J.

6.2.1. PARAMETRIZAÇÃO EQUIPAMENTOS/APLICAÇÃO

Nesta subsecção é abordada a parametrização efetuada no variador de velocidade e no autómato.

6.2.1.1. PARAMETRIZAÇÃO VARIADOR VELOCIDADE

No Anexo I podem ser observados os valores usados para a parametrização do variador de velocidade. Neste sistema pretende-se que o variador de velocidade seja apenas um atuador, que responde a uma referência de velocidade, tratando-a e processando-a o mais rapidamente possível. A precisão do controlo vai ser afetada pelo atraso no processamento da referência. Foi importante diminuir as rampas de aceleração e desaceleração, do variador, para o valor mínimo de 0,1 s para atingir a velocidade pretendida o mais rapidamente possível.

Foi efetuado o *autotuning* do motor, desta forma o variador tem um melhor conhecimento das características do motor a controlar. O procedimento de *autotuning* é habilitado no

menu de configurações, nesse momento o variador inicia o cálculo, de forma estática, dos parâmetros do motor.

Os parâmetros do variador que não constam no Anexo I, ficaram com os valores por defeito.

6.2.1.2. PARAMETRIZAÇÃO AUTÓMATO

Antes da implementação da biblioteca de controlo difuso é necessária uma preparação inicial. Esta preparação passa pela configuração da comunicação, do *encoder*, calibração dos valores de entrada e saída do sistema, para ir ao encontro dos admitidos na biblioteca de controlo difuso. Esses preparativos podem ser consultados no Anexo H.

6.2.2. IMPLEMENTAÇÃO DO ALGORITMO

Esta subsecção descreve a forma como foram interligados os vários blocos de função da biblioteca de controlo lógico difuso. Como já foi referido anteriormente na secção 2.4 um controlador difuso é composto por 3 componentes, a entrada/fusificação, o processamento regras difusas e a saída/desfusificação (ver **Figura 79**)

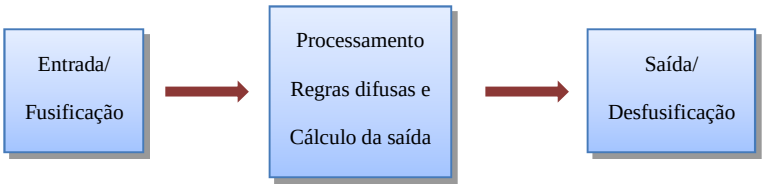


Figura 79 – Componentes de um controlador lógico difuso

6.2.2.1. FUSIFICAÇÃO

Na sub-subseção de fusificação foram definidas para as variáveis de entrada, os pontos de suporte dos blocos de fusificação e as funções de pertença.

Para a variável de entrada relativa ao erro foram definidos 5 pontos de suporte e obrigatoriamente 5 funções de pertença (Figura 80) (Tabela 23).

Tabela 23 – Pontos de suporte e funções de pertença do erro

Ponto de suporte	Função de pertença
X1	Negativo alto
X2	Negativo baixo
X3	Nulo
X4	Positivo baixo
X5	Positivo alto

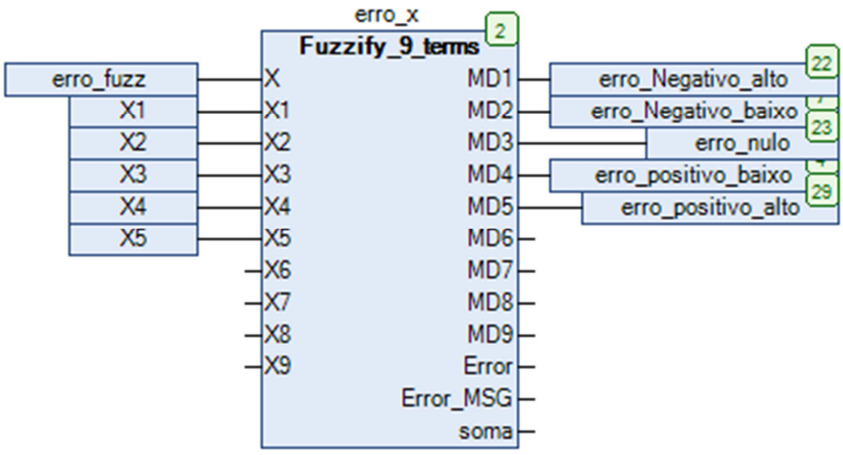


Figura 80 – Fusificação do erro

Para a variável de entrada relativa à variação do erro foram definidos 3 pontos de suporte e obrigatoriamente 3 funções de pertença (Figura 81) (Tabela 24).

Tabela 24 – Pontos de suporte e funções de pertença da variação do erro

Ponto de suporte	Função de pertença
Y1	Negativa
Y2	Nula
Y3	Positiva

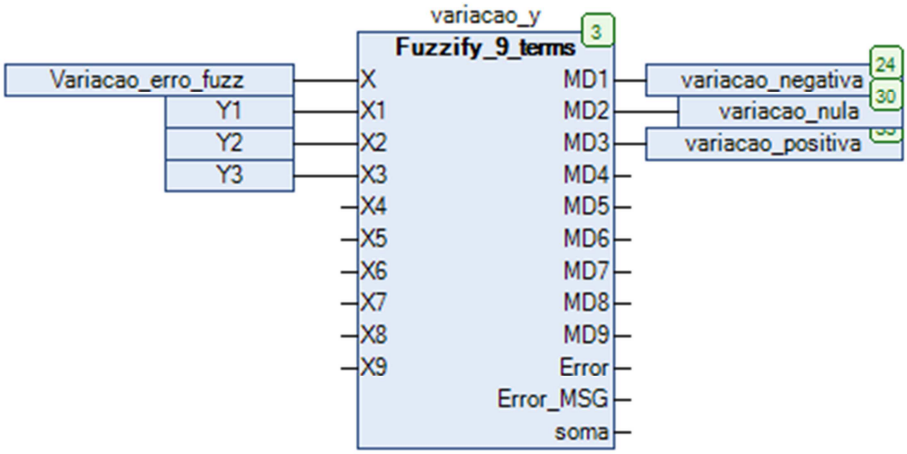


Figura 81 – Fusificação da variação do erro

6.2.2.2. PROCESSAMENTO DAS REGRAS DIFUSAS

As regras difusas permitem ao utilizador desenvolver um conjunto de condições que visam cobrir o universo do conhecimento empírico que ele tem sobre a aplicação. Por uma questão de organização de ideias e para que não falhe nenhuma regra, deve ser criada uma Tabela 25 com o número de células igual à multiplicação do número de funções de

pertença das suas variáveis. Nesta aplicação há cinco regras para o erro e três regras para a variação do erro, logo o número máximo de regras é quinze.

Tabela 25 – Conjunto de regras difusas

Entradas sistema	Variação negativa	Variação nula	Variação positiva
Erro Negativo Alto	VNA	VNA	VNA
Erro Negativo Baixo	VNB	VNB	VNB
Erro Nulo	VNB	VN	VPB
Erro Positivo Baixo	VPB	VPB	VPB
Erro Positivo Alto	VPA	VPA	VPA

Legenda saídas	
VNA	Velocidade negativa alta
VNB	Velocidade negativa baixa
VN	Velocidade nula
VPB	Velocidade positiva baixa
VPA	Velocidade positiva alta

Com análise da Tabela 25 pode-se observar o número de vezes que uma saída é ativada. Como há saídas que são ativas mais que uma vez, estas podem ser agrupadas.

Na Figura 82 podemos observar as regras implementadas com os blocos Rule_and_max. Posteriormente as regras são agrupadas através dos blocos Rule_or_max para gerarem uma única saída velocidade negativa alta (VNA). Esta vai servir de entrada no bloco de desfusificação.

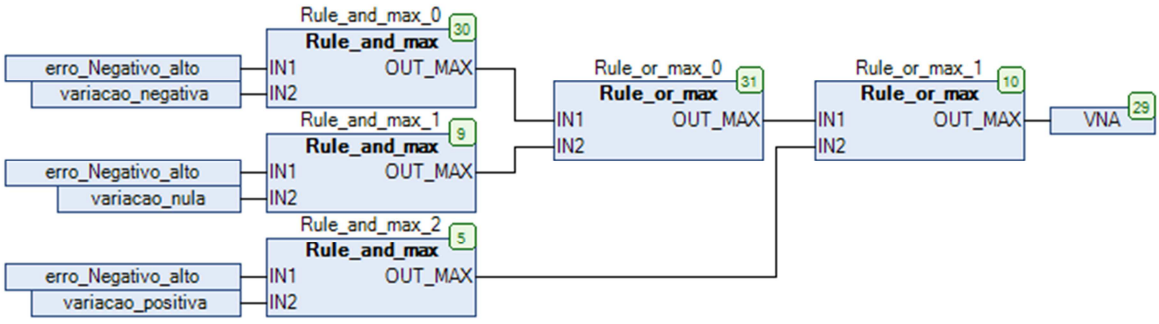


Figura 82 – Saída regra velocidade negativa alta (VNA)

Na Figura 83 podemos observar as regras implementadas com os blocos Rule_and_max. Posteriormente as regras são agrupadas através dos blocos Rule_or_max para gerarem uma única saída velocidade negativa baixa (VNB). Esta vai servir de entrada no bloco de desfusificação.

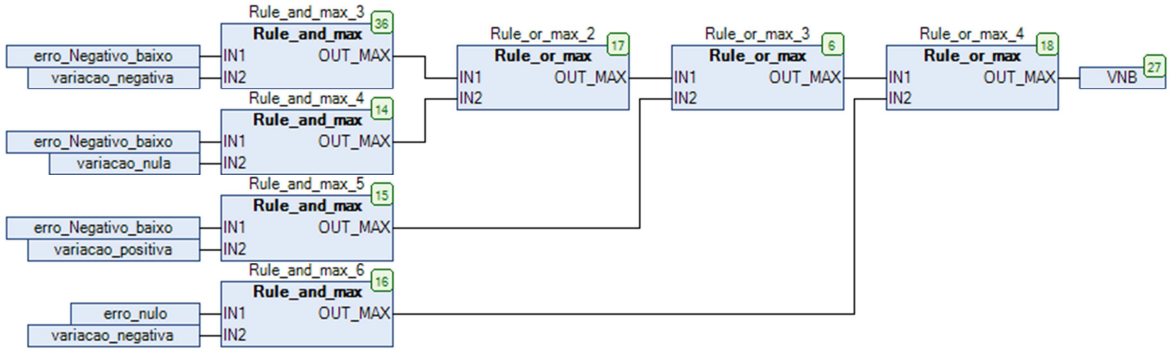


Figura 83 – Saída regra velocidade negativa baixa (VNB)

Na Figura 84 podem observar-se as regras implementadas com os blocos Rule_and_max para gerar uma única saída velocidade nula (VN). Esta vai servir de entrada no bloco de desfusificação.

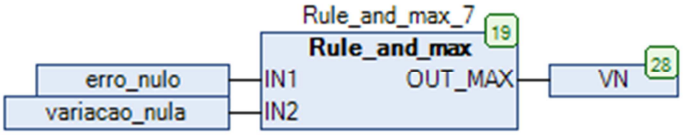


Figura 84 – Saída regra velocidade nula (VN)

Na Figura 85 apresentam-se as regras implementadas com os blocos Rule_and_max. Posteriormente as regras são agrupadas através dos blocos Rule_or_max para gerarem uma única saída velocidade positiva baixa (VPB). Esta vai servir de entrada no bloco de desfusificação.

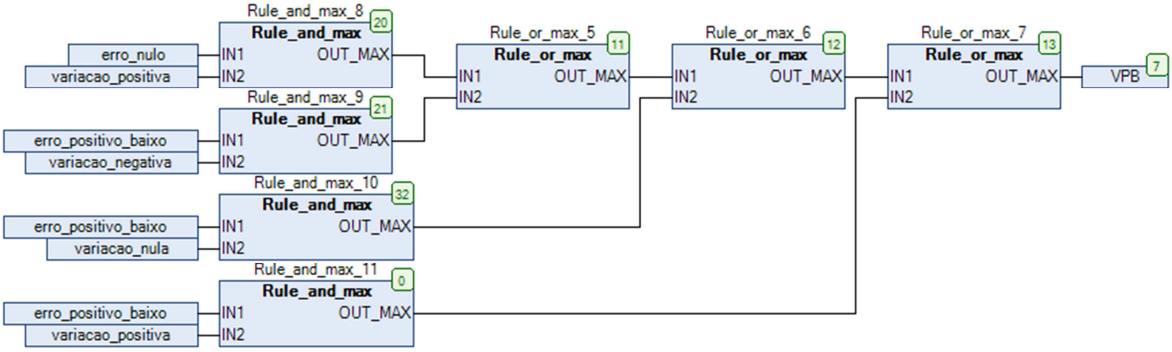


Figura 85 – Saída regra velocidade Velocidade Positiva Baixa (VPB)

Na Figura 86 apresentam-se as regras implementadas com os blocos Rule_and_max. Posteriormente as regras são agrupadas através dos blocos Rule_or_max para gerarem uma única saída velocidade positiva alta (VPA). Esta vai servir de entrada no bloco de desfusificação.

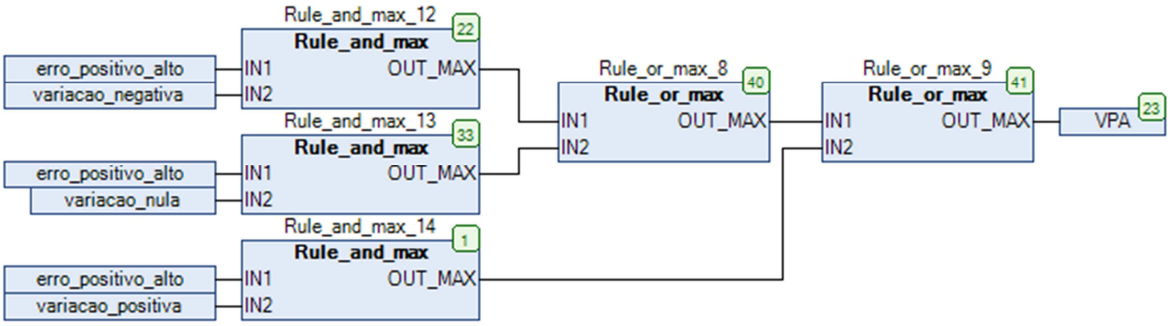


Figura 86 – Saída regra velocidade Velocidade Positiva Alta (VPA)

6.2.2.3. DESFUSIFICAÇÃO

A desfusificação é o processo que recolhe todas as saídas das regras aplicando a respetiva ponderação aos pontos de suporte (*singletons*), colocando o valor calculado na saída do controlador. Para o cálculo do valor das saídas, como se pode observar na Figura 87, é usado o bloco Defuzz_ST.

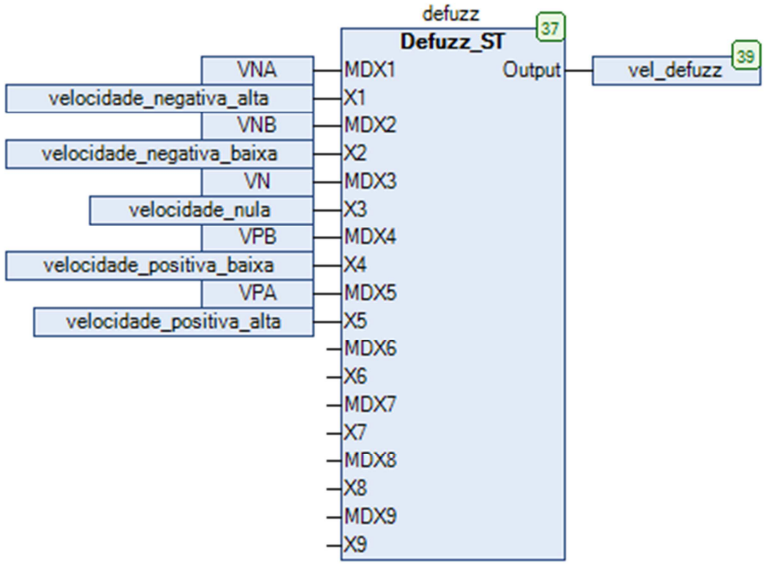


Figura 87 – Bloco de desfusificação implementado

Na Figura 88 podemos ter uma visão geral do controlador com as suas 3 componentes.

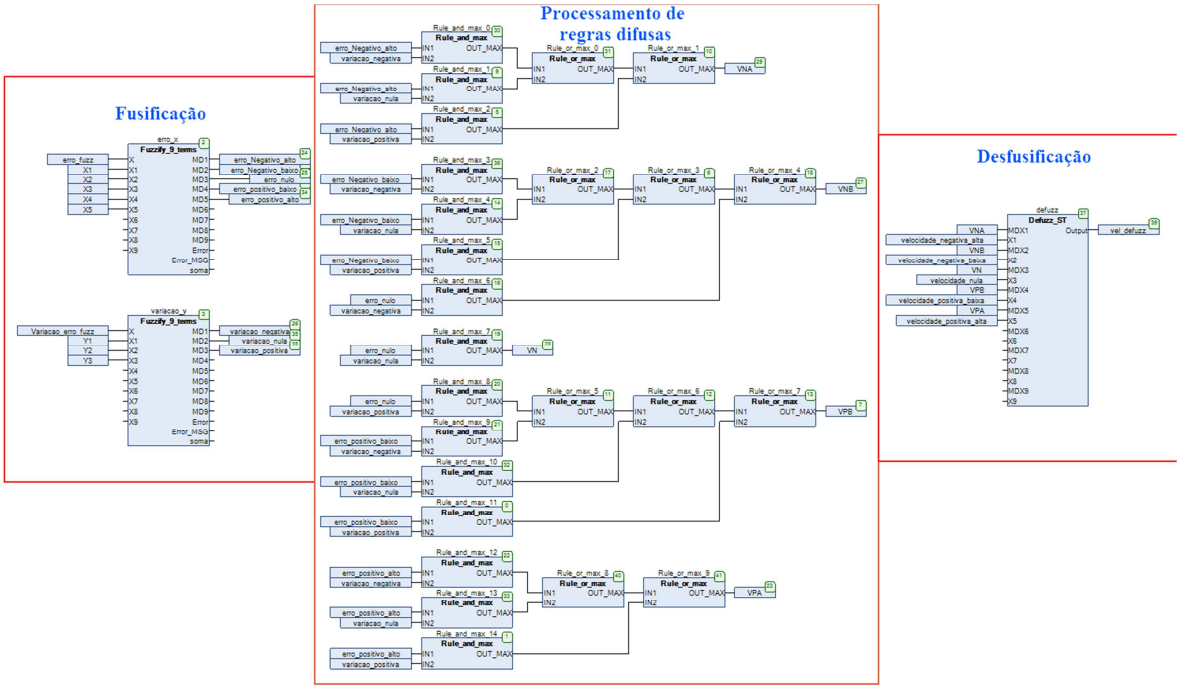


Figura 88 – Vista geral do controlador difuso

6.3. RESULTADOS OBTIDOS

Nesta secção vamos analisar os resultados obtidos durante a fase de validação da biblioteca recorrendo a gráficos e tabelas que relacionam a posição pretendida, com a posição atual e com a velocidade do motor (variação do erro).

Neste caso de estudo será efetuado o posicionamento entre duas posições, simulando uma situação real. Neste tipo de aplicações é essencial que a resposta do sistema não tenha *overshoot* e que o tempo de estabelecimento seja o mais rápido possível.

Para um melhor registo dos dados de modo a efetuar-se a comparação, foi necessário criar uma tabela para registo de dados. Foram definidos uns valores iniciais (ver Tabela 26) para começar a testar a biblioteca e a sua respetiva implementação.

Tabela 26 - Registo de valores dos testes

Pontos de suporte Erro					Pontos de suporte Variação do erro			Pontos de suporte Velocidade pretendida				
X1	X2	X3	X4	X5	Y1	Y2	Y3	VNA	VNB	VN	VPB	VPA
100	300	500	700	900	100	500	900	0	250	500	750	1000

Erro Calibração		Variação Erro Calibração		Velocidade saída Calibração		Posição Pretendida	
Mínimo	Máximo	Mínimo	Máximo	Mínimo	Máximo	Inferior	Superior
-500	500	-1500	1500	-200	200	-1000	1000

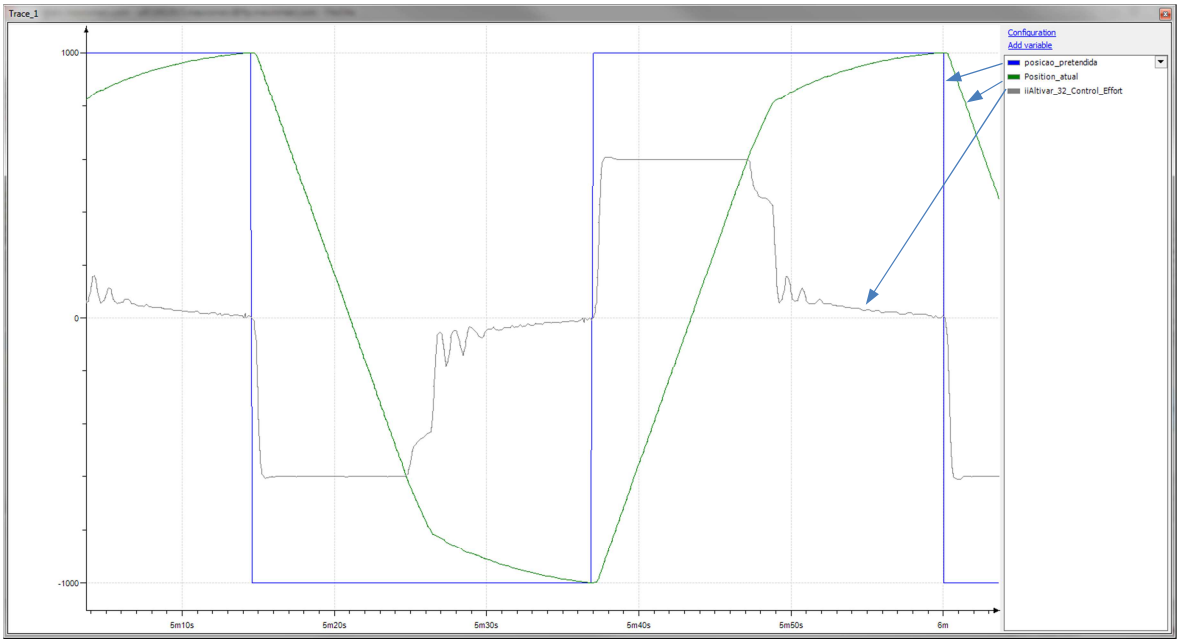


Figura 89 – Resposta inicial do sistema

Nos gráficos ilustrativos do comportamento do sistema deve ser tido em conta as cores das linhas do gráfico, a azul está representada a posição pretendida, a verde a posição atual e a cinzento a velocidade do motor.

Observando a resposta do sistema (Figura 89) aos parâmetros da Tabela 26, foi percetível que o sistema antecipava muito cedo a chegada à posição pretendida.

Uma das soluções para fazer com que o sistema comece a reagir mais tarde é alterar os pontos de suporte das funções de pertença do erro, ou seja, é necessário aproximar os pontos de suporte laterais, do ponto de suporte central X3. Assim sendo foram alterados os pontos X1=200, X2=400, X4=600, X5=800.

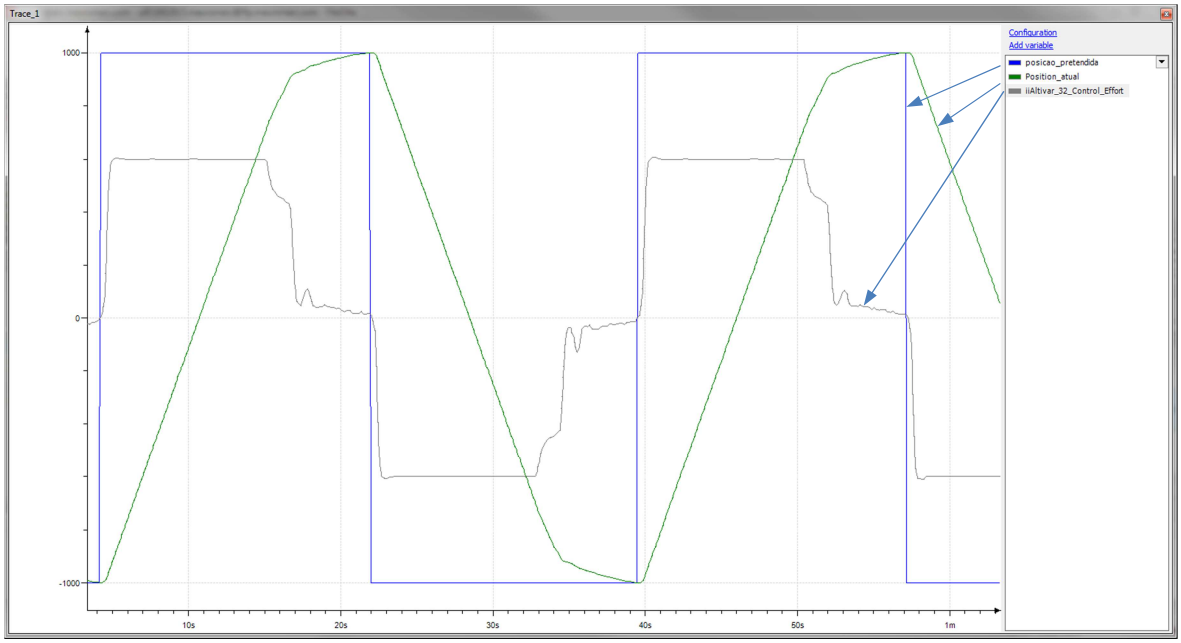


Figura 90 – Resposta aproximando os pontos de suporte

Após análise da Figura 90, conclui-se que a resposta do sistema melhorou, mas no entanto ainda pode ser melhor. É necessário ajustar os pontos de suporte laterais novamente. Foram alterados os pontos X1=400, X2=450, X4=550, X5=600. Na Figura 91 pode-se avaliar que o sistema responde ainda melhor. Com esta resposta melhorada vai ser aumentada a velocidade para ver se o sistema continua com o mesmo tipo de resposta. A velocidade de calibração de saída mínima passou para -300 e a máxima para 300.

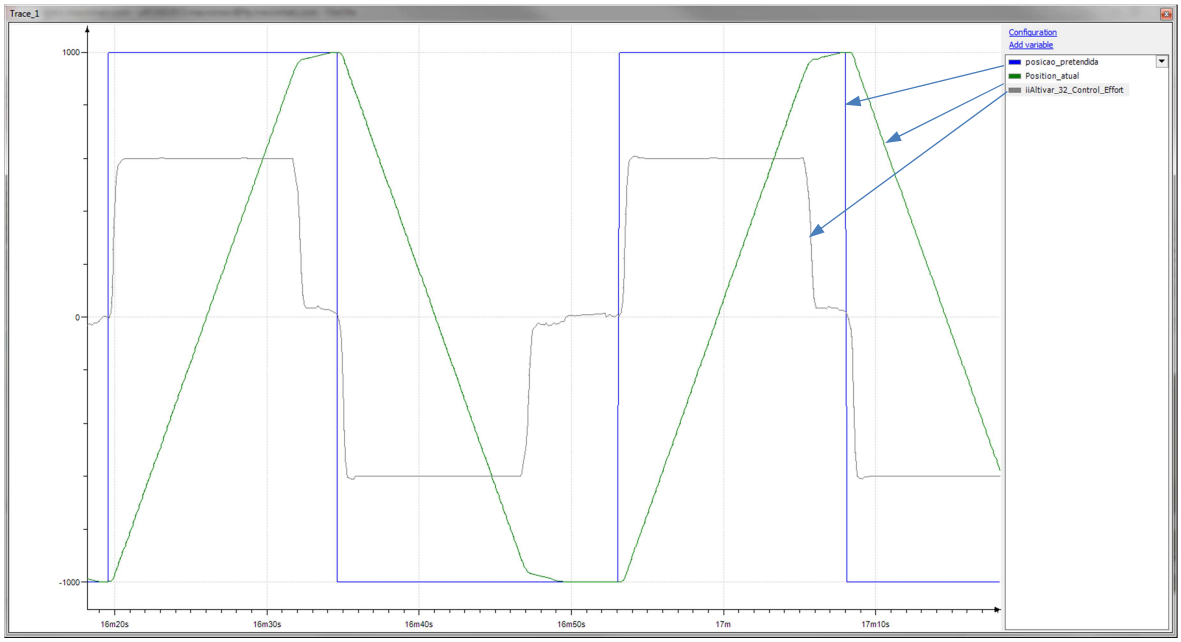


Figura 91 – Resposta sistema com pontos de suporte ainda mais próximos

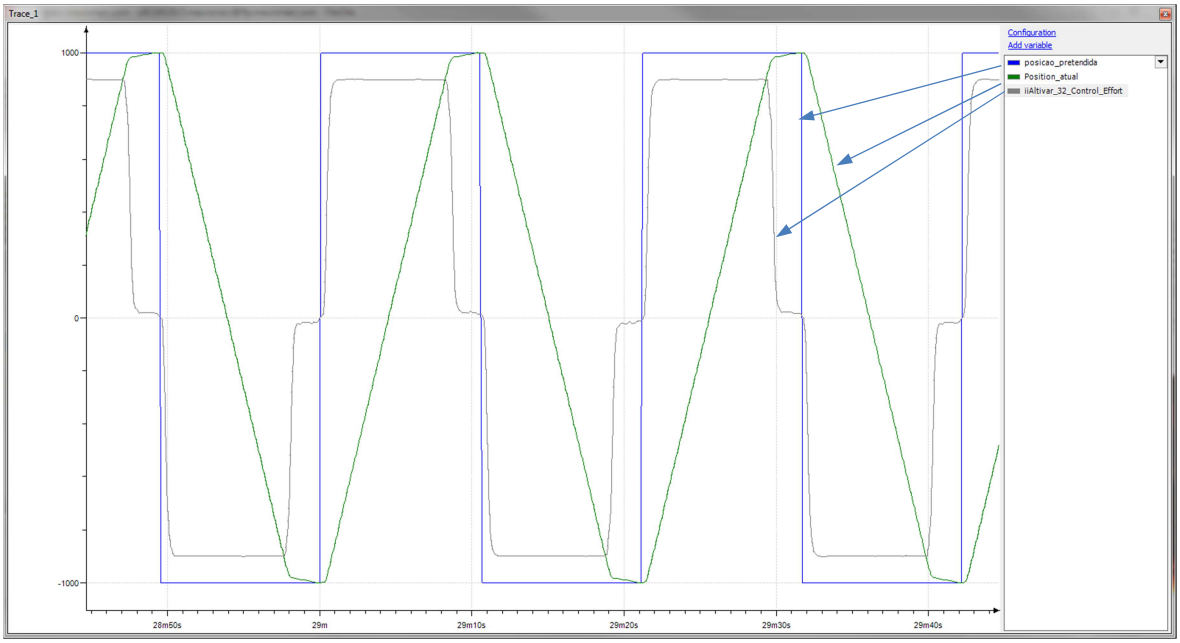


Figura 92 - Resposta do sistema para uma velocidade máxima de 30 Hz

Conforme é visível na Figura 92, o sistema continua a ter um comportamento desejado. Aumentou-se a velocidade máxima para 40 Hz.

Analisando o resultado da Figura 93 percebeu-se que o sistema por vezes tinha dificuldade em estabilizar. Alterou-se então o valor das funções de pertença da saída de velocidade, para tentar diminuir a instabilidade junto do valor central da velocidade. Foram alterados os valores VNB=300 e VPB=700.

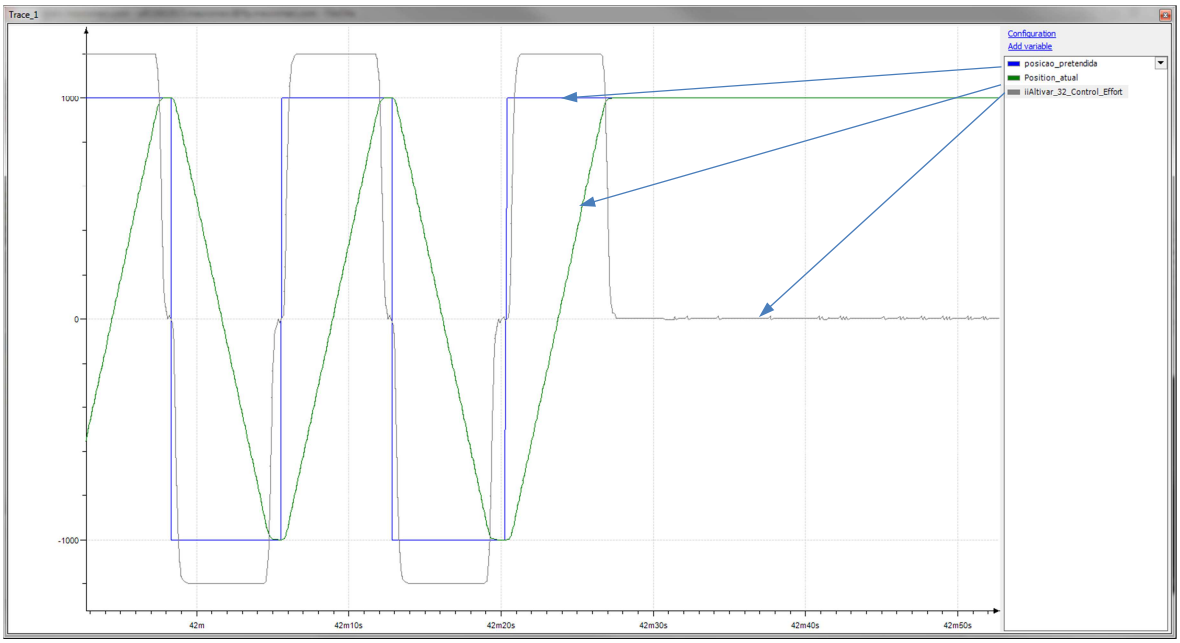


Figura 93 - Resposta do sistema para uma velocidade máxima de 40 Hz

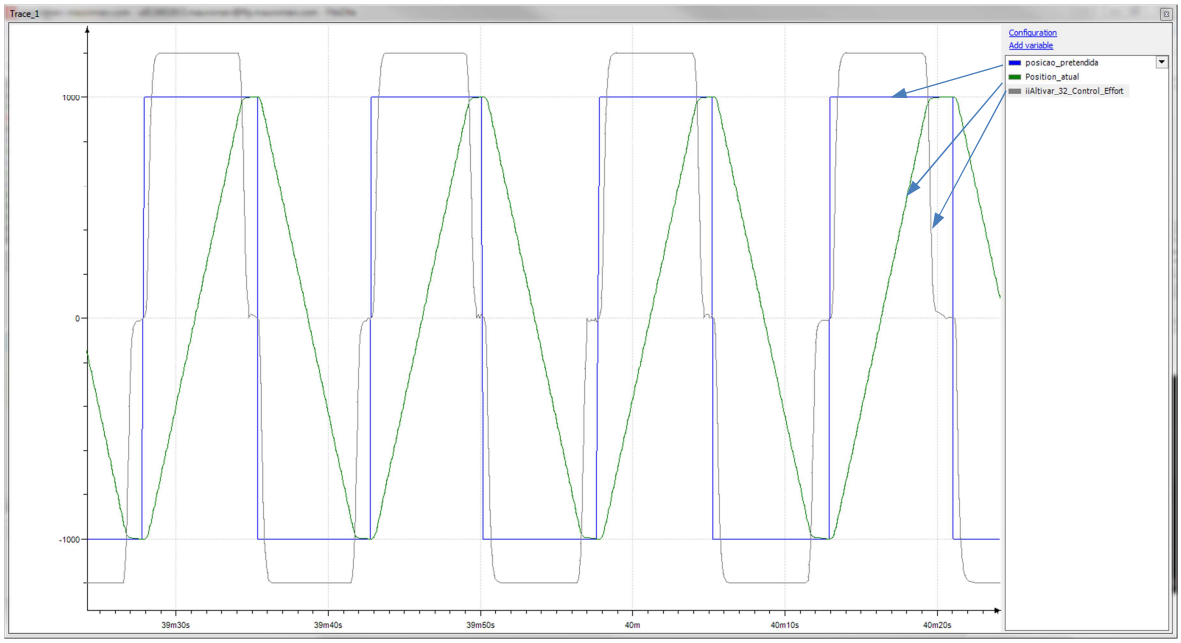


Figura 94 - Resposta do sistema para uma velocidade máxima de 40 Hz com VNB e VPB alterados

Analisando a resposta da Figura 94 percebe-se que a resposta do sistema voltou a ter repetibilidade.

Aumentou-se a velocidade máxima para 50 Hz.

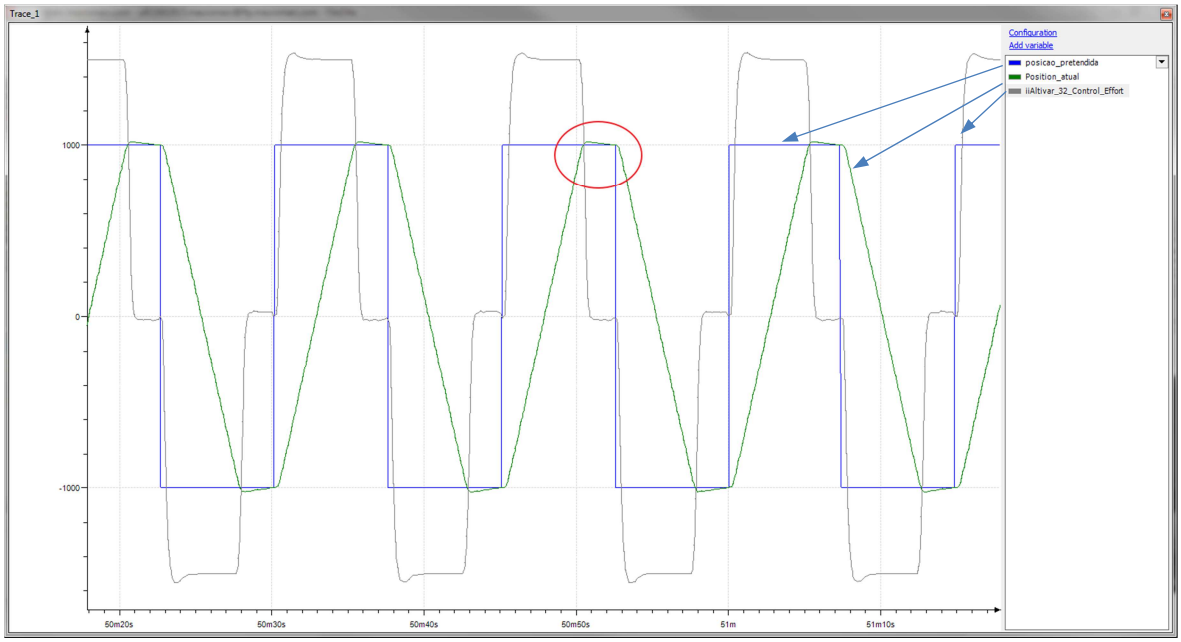


Figura 95 - Resposta do sistema para uma velocidade máxima de 50 Hz com *overshoot*

Com a velocidade nominal máxima do motor observa-se na Figura 95 o aparecimento de *overshoot*. O sistema tem uma resposta tardia, logo ultrapassa o valor pretendido.

Alterando os valores dos pontos de suporte do erro, X1=375, X2=425, X4=575, X5=625 e VNA=325 e VNB=675

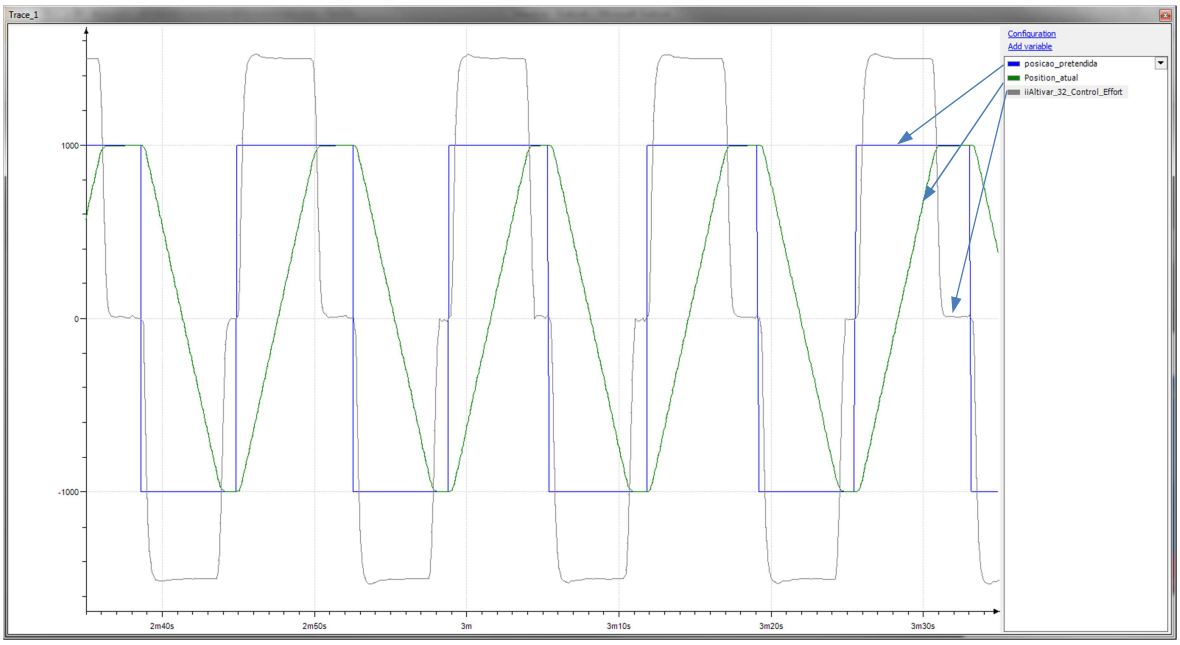


Figura 96 - Resposta do sistema para uma velocidade máxima de 50 Hz sem *overshoot*

Na Figura 96 pode-se observar uma boa resposta do sistema à velocidade nominal máxima do motor.

7. CONCLUSÕES

Ao longo desta tese podem observar-se varias soluções nos principais fabricantes de autómatos a nível mundial. Uma das limitações de todos os fabricantes é estarem disponíveis apenas para autómatos de gamas mais elevadas, ficando de fora as gamas mais baixas, que nos dias de hoje já têm capacidades de processamento elevadas.

Outra das lacunas dos fabricantes é que só disponibilizam os *softwares*/Bibliotecas de forma paga, não estando incluída nas versões base dos *softwares* de programação.

Estes pressupostos serviram de base à realização desta tese.

Um dos objetivos foi usar a norma IEC61131-7 de programação controladores lógicos para efetuar a implementação de um controlador lógico difuso num PLC tendo como base a plataforma Codesys. Mas essa implementação não foi possível, pois o compilador não está preparado para aceitar as instruções da norma IEC 61131-7.

Foi então desenvolvida uma biblioteca com blocos de função capazes de efetuarem todos os componentes do controlador lógico difuso (fusificação, inferência e desfusificação).

Face aos resultados obtidos pode-se concluir que os objetivos foram plenamente cumpridos. Salienta-se que o facto do sistema de testes não ter uma carga acoplada ao veio do motor, dificulta a estabilização do sistema, pois não amortece pequenas oscilações.

Fica em aberto, como trabalho futuro, a implementação de mais métodos de fusificação, inferência e desfusificação. Desta forma o controlador ficaria mais completo.

A portabilidade desta biblioteca para outros sistemas não baseados em Codesys, mas que também respeitem a norma IE61131-3, é um objetivo. Por exemplo para a plataforma TIA Portal da Siemens, Studio 5000 da Rockwell entre outros.

Referências Documentais

[1] L. A. a. B. F. A. BRYAN, Programmable Controllers - Theory and Implementation., Atalanta Georgia: Industrial Text, 1997.

[2] H. A. S. MAMDANI E, "An experiment in linguistic synthesis with a fuzzy logic controller," *International Journal of Man-Machine Studies*, vol. 7, pp. 1-13, 1975.

[3] A. ZADEH L, Fuzzy Sets, Berkeley, California: Intl J. Information Control, 1965.

[4] E. COX, The Fuzzy Systems Handbook, Second Edition: A Practitioner's Guide to Building, Using, and Maintaining Fuzzy Systems, Chappaqua, New York: AP Professional, 1998.

[5] T. ROSS, Fuzzy Logic with engineering applications, New Mexico: WILEY, 1995.

[6] OMRON, "CX-Process Tool Ver. 5.2 OPERATION MANUAL," 02 2013. [Online]. Available: http://www.fa.omron.com.cn/data_pdf/mnu/w372-e1-15_cx-processtool.pdf?id=16. [Accessed 02 2014].

[7] SIEMENS AG., FUZZYCONTROL++ User's Manual, 2008.

[8] Rockwell Allen-Bradley, “RSLogix 5000 Fuzzy Designer,” 03 2007. [Online]. Available: http://literature.rockwellautomation.com/idc/groups/literature/documents/um/logix-um004_-en-p.pdf. [Acedido em 03 2014].

[9] Schneider Electric, “Fuzzy Control Library V1.2,” 02 2009. [Online]. Available: http://www.global-download.schneider-electric.com/85257849002EB8CB/all/AA7F73E7197E08D8852578720077CADA/$File/33004219_k01_000_02.pdf. [Acedido em 03 2014].

[10] International Electrotechnical Commission, “Programmable controllers - Part 7: Fuzzy control programming,” 10 08 2000. [Online]. Available: http://webstore.iec.ch/webstore/webstore.nsf/artnum/026319!opendocument. [Acedido em 12 2013].

[11] International Electrotechnical Commission, “Programmable controllers - Part 3: Programming languages,” 20 02 2013. [Online]. Available: http://webstore.iec.ch/webstore/webstore.nsf/Artnum_PK/47556. [Acedido em 01 2014].

[12] 3S-Smart Software Solutions GmbH, “CODESYS Device Directory,” [Online]. Available: http://www.codesys.com/the-codesys-device-directory.html. [Acedido em 10 2014].

[13] Schneider Electric, “Modicon M258 Logic Controller Hardware Guide,” 04 2014. [Online]. Available: http://download.schneider-electric.com/files?p_File_Id=455436659&p_File_Name=EIO0000000432.06.pdf. [Acedido em 10 2014].

[14] Schneider Electric, "Altivar 32 Installation manual," 03 2010. [Online]. Available: http://download.schneider-electric.com/files?p_File_Id=27524460&p_File_Name=ATV32_installation_manual_EN_S1A28686_01.pdf. [Accessed 10 2014].

[15] Schneider Electric, “XCC1514TS11Y Product data sheet,” [Online]. Available: http://www.ops-ecat.schneider-electric.com/cut.CatalogueRetrieverServlet/CatalogueRetrieverServlet?fct=get_element&env=publish&scp_id=Z018&lc=us&el_typ=product&cat_id=BU_AUT_1440_L3_Z018&maj_v=1&min_v=7&nod_id=0000000003&prd_id=XCC1514TS11Y&frm=pdf&pdf_fr. [Acedido em 10 2014].

[16] Universal Motors, “BF31 Informações técnicas,” [Online]. Available: http://www.universalmotors.pt/pg45-produto-4-ii-iv-polos-3000-1500-rpm-pt?potcv=&potkw=0.30%20/%200.22&tipo=BF31%2071%20M1%202%20/%204. [Acedido em 10 2014].

Anexo A. Processadores OMRON com capacidades controlo difuso

Tabela A. 1 – Processadores com capacidades de controlo do processo e bibliotecas de controlo difuso

PLC Series	CS Series		CJ Series
Size (Height)	**130 mm**		**90 mm**
Unit Name	**Loop Control Board**	**Process-control CPU Unit**	**Loop-control CPU Unit**
Model	CS1W-LCB01/05	CS1D-CPU[][]P	CJ1G-CPU[][]P
Appearance			
Unit Classification	Inner Board	Process-control CPU Unit (CS1D-CPU[][]H and Loop Control Board set)	CJ1 CPU Unit Built-in Loop Control Board
Compatible CPU Unit	CS1[]-CPU[][]H or CS1[]-CPU[][]S	Built-in Loop Control Function in CS1D CPU Unit	Built-in Loop Control Function in CS1G CPU Unit
Duplex Mode	Not supported	Supported	Not supported
Number of Function Blocks	**Control Block + Operation Block:** **LCB01:** 50 blocks max. **LCB05:** 500 blocks max. • Fuzzy Logic <Block Model 016> • Arithmetic Operation <Block Model 126> • Time Sequence Data Statistics <Block Model 153>: 100 Blocks max.	**Control Block + Operation Block:** 500 blocks max. However, external automatic controller block *(ES100X Controller Terminal Block <Block Model 045>)*cannot be used. • Fuzzy Logic (Block Model 016) • Arithmetic Operation <Block Model 126> • Time Sequence Data Statistics <Block Model 153>: 100 Blocks max.	**Control Block + Operation Block:** **CJ1W-CPU42P:** 50 blocks max. **CJ1W-CPU43/44/45P:** 300 blocks max However, external automatic controller block *(ES100X Controller Terminal Block <Block Model 045>)*cannot be used. • Fuzzy Logic (Block Model 016) • Arithmetic Operation <Block Model 126> • Time Sequence Data Statistics <Block Model 153>: 100 Blocks max.

Anexo B. Norma IEC 61131-3 (Formato Digital)

INTERNATIONAL STANDARD

IEC 61131-3

Second edition
2003-01

Programmable controllers –

Part 3:
Programming languages

Automates programmables –

Partie 3:
Langages de programmation

Reference number
IEC 61131-3:2003(E)

Anexo C. Norma IEC 61131-7 (Formato Digital)

NORME INTERNATIONALE

INTERNATIONAL STANDARD

CEI
IEC

61131-7

Première édition
First edition
2000-08

Automates programmables –

Partie 7:
Programmation en logique floue

Programmable controllers –

Part 7:
Fuzzy control programming

Numéro de référence
Reference number
CEI/IEC 61131-7:2000

Anexo D. IEC61131-7 Regras de produção

Além dos elementos de linguagem listados acima da norma IEC 61131-3, podem ser utilizados os seguintes elementos de linguagem:

function_block_declaration ::= ‘FUNCTION_BLOCK’ function_block_name
{fb_io_var_declarations}
{other_var_declarations}
function_block_body
‘END_FUNCTION_BLOCK’

fb_io_var_declarations ::= input_declarations | output_declarations

other_var_declarations ::= var_declarations

function_block_body ::= {fuzzify_block}
{defuzzify_block}
{rule_block}
{option_block}

fuzzify_block ::= ‘FUZZIFY’ variable_name
{linguistic_term}
‘END_FUZZIFY

defuzzify_block ::= ‘DEFUZZIFY’ f_variable_name
[range]
{linguistic_term}
defuzzification_method
default_value
‘END_DEFUZZIFY’

rule_block ::= ‘RULEBLOCK’ rule_block_name
operator_definition
[activation_method]
accumulation_method
{rule}
‘END_RULEBLOCK’

option_block ::= ‘OPTION’
any manufacturer specific parameter
‘END_OPTION’

linguistic_term ::= ‘TERM’ term_name ‘:=’ membership_function ‘;’

membership_function ::= singleton | points

singleton ::= numeric_literal | variable_name

points ::=	{‘(‘numeric_literal | variable_name ‘,’
	numeric_literal ‘)’}
defuzzification_method ::=	‘METHOD’ ‘:’ ‘CoG’ | ‘CoGS’ | ‘CoA’ | ‘LM’ | ‘RM’ ‘;’
default_value ::=	‘DEFAULT’ ‘:=’ numeric_literal | ‘NC’ ‘;’
range ::=	‘RANGE(‘numeric_literal ‘..’ numeric_literal’)’ ‘;’
operator_definition ::=	[(‘OR’ ‘:’ ‘MAX’ | ‘ASUM’ | ‘BSUM’)]
	[(‘AND’ ‘:’ ‘MIN’ | ‘PROD’ | ‘BDIF’)] ‘;’
activation_method ::=	‘ACT’ ‘:’ ‘PROD’ | ‘MIN’ ‘;’
accumulation_method ::=	‘ACCU’ ‘:’ ‘MAX’ | ‘BSUM’ | ‘NSUM’ ‘;’
rule ::=	‘RULE’ integer_literal ‘:’
	‘IF’ condition ‘THEN’ conclusion [WITH weighting_factor] ‘;’
condition ::=	x{(‘AND’ x)|(‘OR’ x)}
x ::=	[‘NOT’] (subcondition | (‘ (‘ condition ’)’))
subcondition ::=	variable_name | (variable_name ‘IS’ [`NOT’] term_name)
conclusion ::=	{ (variable_name | (variable_name ‘IS’ term_name)) ‘,’}
	(variable_name | variable_name ‘IS’ term_name)
weighting_factor ::=	variable | numeric_literal
function_block_name ::=	identifier
ruleblock_name ::=	identifier
term_name ::=	identifier
f_variable_name ::=	identifier
variable_name ::=	identifier
numeric_literal ::=	integer_literal | real_literal
input_declarations ::=	ver IEC 61131-3, anexo B
output_declarations ::=	ver IEC 61131-3, anexo B
var_declarations ::=	ver IEC 61131-3, anexo B
identifier ::=	ver IEC 61131-3, anexo B

Anexo E. IEC61131-7 Palavras-chave reservadas para FCL

Tabela E. 1 – Palavras-chave reservadas para FCL

Palavra-chave	Significado
( )	Parênteses nas condições, termos e intervalos
ACCU	Método de acumulação
ACT	Método de ativação
AND	Operador E
ASUM	Operador OU, soma algébrica
BDIF	Operador E, diferença delimitada
BSUM	Método de acumulação, operador E, soma delimitada
CoA	Método de desfusificação por centro de área
CoG	Método de desfusificação por centro de gravidade
CoGS	Método de desfusificação por centro de gravidade por *singletons*
DEFAULT	Valor da saída para o caso de não ser ativada nenhuma regra
DEFUZZIFY	Desfusificação de uma variável de saída
END_DEFUZZIFY	Fim das definições de desfusificação
END_FUNCTION_BLOCK	Fim das definições do bloco de função
END_FUZZIFY	Fim das definições de fusificação
END_OPTIONS	Fim das definições de parâmetros opcionais
END_RULEBLOCK	Fim das definições do bloco de regras
END_VAR	Fim da definição de variáveis de entrada/saída
FUNCTION_BLOCK	Início das definições do bloco de função
FUZZIFY	Fusificação das variáveis de entrada
IF	Início das regras que são seguidas das condições
IS	Segue a variável nas condições e conclusões
LM	Método de desfusificação do máximo mais à esquerda
MAX	Método de acumulação Máximo, operador OU
METHOD	Método de desfusificação
MIN	Mínimo como operador E, método de ativação

NC	Valor da saída sem alterações para o caso de não ser ativada nenhuma regra
NOT	Operador Negação
NSUM	Método de acumulação através de soma normalisada
OPTIONS	Definição de parâmetros opcionais
OR	Operador OU
PROD	Produto com operador E, método de ativação
RANGE	Limites das funções de pertença
RM	Método de desfusificação do máximo mais à esquerda
RULE	Início da definição de uma regra difusa
RULEBLOCK	Início da definição de um bloco de regras
TERM	Definição do termo linguístico (função de pertença) para uma variável linguística
THEN	Separa a condição da conclusão
VAR	Definição de uma variável local
VAR_INPUT	Definição de uma variável de entrada
VAR_OUTPUT	Definição de uma variável de saída
WITH	Definição do fator de ponderação

Anexo F. SOMACHINE (SCHNEIDER ELECTRIC)

Nesta tese, o *hardware* a utilizar será da marca Schneider Electric. A programação do *hardware* será efetuada com recurso ao *software* Somachine, que integra a interface gráfica convencional do Codesys para configuração e programação dos dispositivos.

Para projetos de programação, a interface gráfica do usuário convencional CoDeSys é integrado na interface gráfica do utilizador SoMachine. Ele fornece as funções de configuração, programação e monitorização do controlador (ver Figura F. 1).

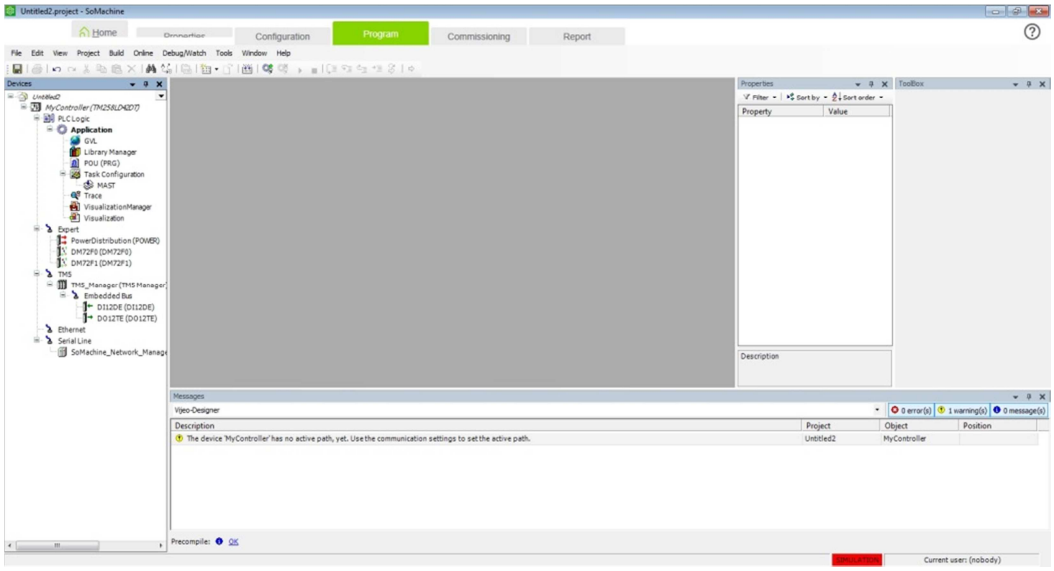


Figura F. 1 – Interface de programação Somachine (Codesys)

MENU INICIAL

Este menu é exibido sempre que se inicia o Somachine, como ilustrado na Figura F. 2. À esquerda é possível escolher abrir um projeto existente, criar um projeto novo, monitorizar um projeto existente ou consultar o centro de aprendizagem. Ao centro são exibidos os projetos recentemente abertos. Do lado direito são exibidas as propriedades dos projetos, quando selecionados ao centro.

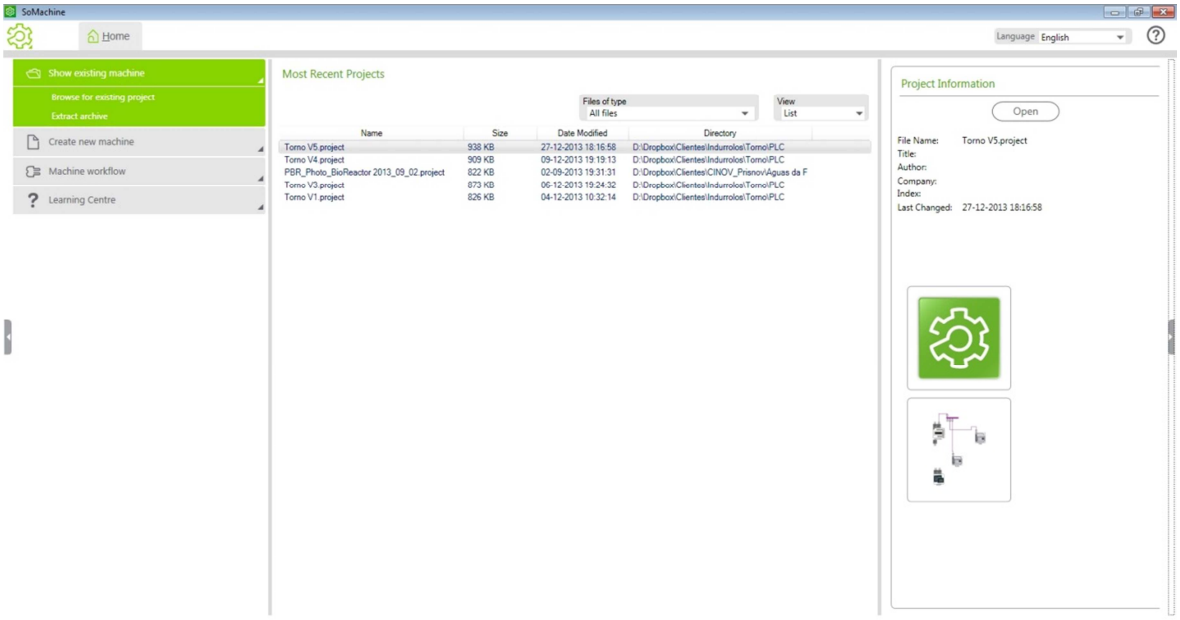


Figura F. 2 – Somachine – Menu arranque

Se a opção selecionada do lado esquerdo for abrir um projeto existente, abre uma nova janela com um navegador de pastas do lado esquerdo, permitindo a seleção do local onde existe o projeto, como se pode ver na Figura F. 3.

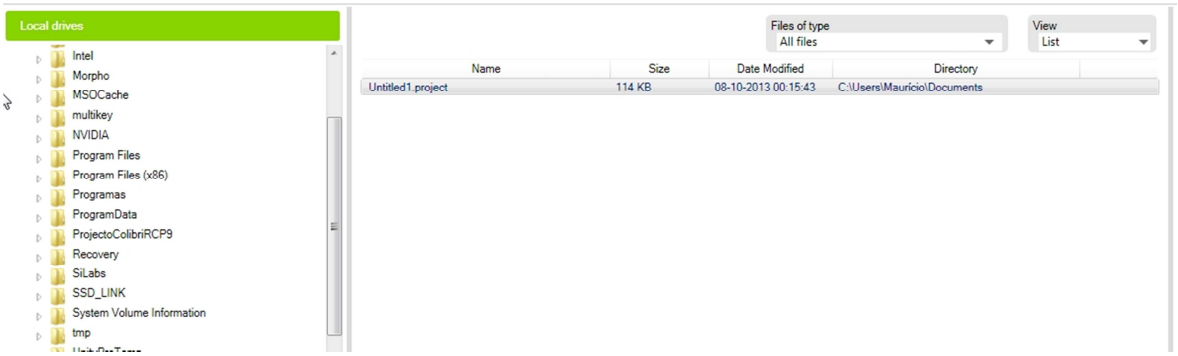


Figura F. 3 – Somachine – Menu arranque – Abrir projeto

Na opção de criar um novo projeto, é possível escolher se pretendemos criar um projeto vazio, ou se pretendemos iniciar o projeto com predefinições das aplicações da Schneider Electric (SE) (ver Figura F. 4)

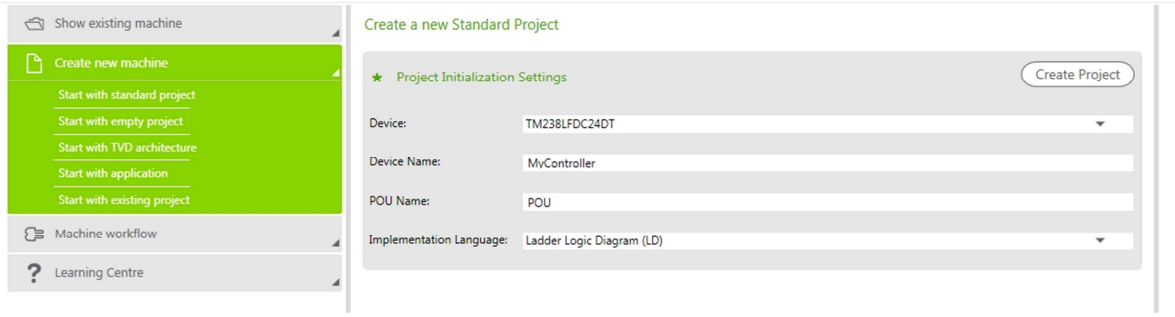


Figura F. 4 – Somachine – Menu arranque – Criar novo projeto

Quando se pretende efetuar o comissionamento de uma máquina ou efetuar uma atualização de *software* tem que se selecionar a opção Machine workflow como é visível na Figura F. 5.

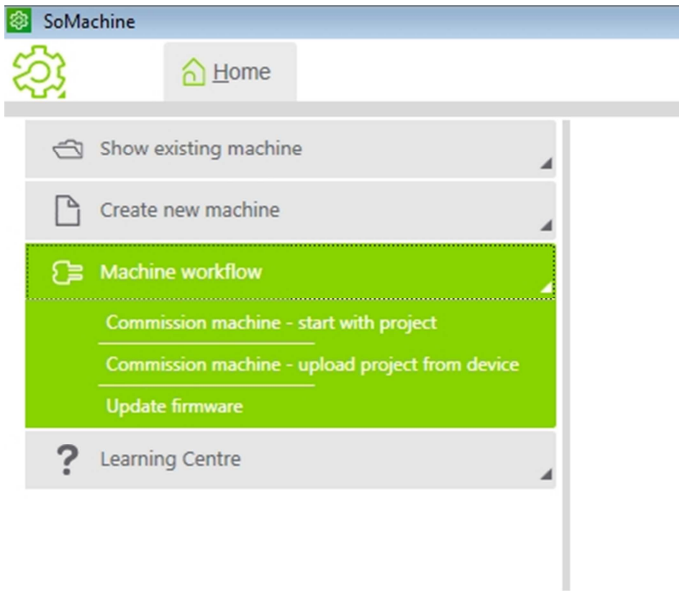


Figura F. 5 – Somachine – Menu arranque – Machine workflow

A última das opções disponíveis no menu de arranque é referente ao centro de aprendizagem (*Learning Centre*), onde podem se selecionadas as opções listadas na Figura F. 6.

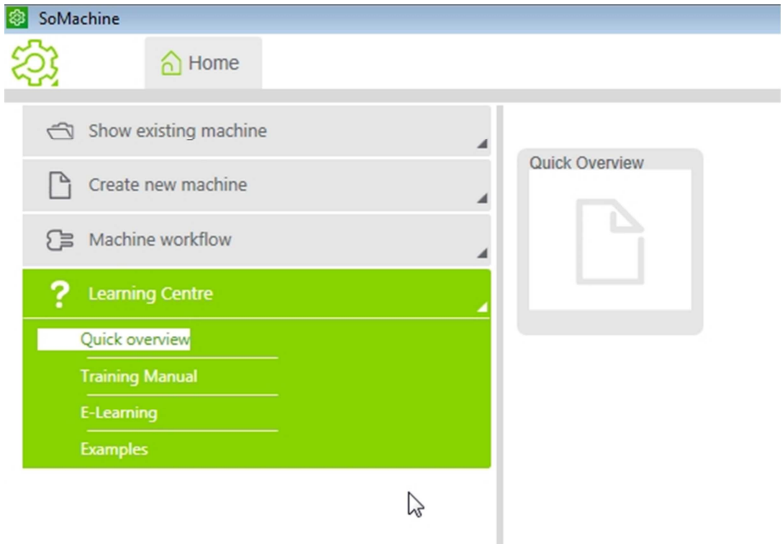


Figura F. 6 – Somachine – Menu arranque – Centro de aprendizagem

AMBIENTE PRINCIPAL

No ambiente principal do Somachine é possível encontrar na parte superior uns separadores que dividem as várias seções do *software* (ver Figura F. 7).

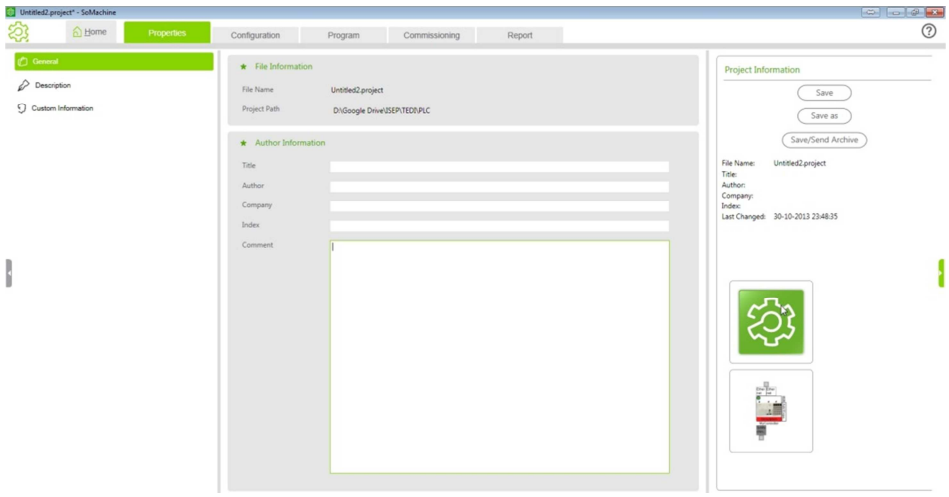


Figura F. 7 – Somachine – Ambiente principal

Os separadores permitem escolher as seguintes secções:

- *Properties* – Informações adicionais ao projeto podem ser definidas (Titulo, autor, empresa, etc…)
- *Configuration* – Definição do *hardware* e redes de comunicação utilizados
- *Program* – Ambiente gráfico Codesys que permite a programação e desenvolvimento
- *Commissioning* – Comissionamento, *backup* e restauro do sistema
- *Report* – Gerar toda a documentação associada ao projeto

Anexo G. Lista de fabricantes que usam Codesys

3xS Smart Systems Solutions S.r.l.
A&R TECH Automatisierungs- und Regelungstechnik GmbH
Aartec AG
ABB Automation Products
ABB Oy Distribution Automation
ABB Schweiz AG
ACD Antriebstechnik GmbH
ADLINK TECHNLOGY Inc.
AEG ursatronics GmbH
AIXIA S.A.
Altus Sistemas de Informática S.A.
AMALINSYA SDN. BHD.
AMK Arnold Müller GmbH
andron GmbH
Ansaldo Sistemi Industriali S.p.A.
ASEM S.p.A.
AUTOMATA GmbH & Co.KG
AVAT Automation GmbH
Bachmann Electronic GmbH
BATEC (S-SYS bvba)
BECK IPC GmbH
BECKHOFF Automation GmbH
Beijing Digital Control Automation Ltd,
Co. (DCA)
BERGHOF Automationstechnik GmbH
BESI Marine Systems GmbH & Co. KG
BIATEC AG
BINAR AB
Biviator AG
Bizerba GmbH & Co. KG
Bosch Rexroth AG
Bosch Rexroth Electric Drives and Controls
B.V.
Bosch Rexroth Mobile Hydraulics
Bosch Rexroth Pneumatics GmbH
Bosshard Elektronik
Brunner-Elektronik AG
Camtec - CAM Technology Corporation
CLAAS Industrietechnik GmbH
CMC Compressor and Machine Controls
N.V.
CNC Systeme - Steuerungstechnik
Contec Steuerungstechnik & Automation
GmbH
cpa Computer Process Automation GmbH
CrossControl AB
CTS GmbH
Cybelec Infranor Group Company
Danaher Motion S.r.l.
Danieli Automation
Datawatt
DEIF A/S
Drescher Industrieelektronik GmbH
E. DOLD & SÖHNE KG
EAE Ewert Ahrensburg Electronic GmbH
Eaton Automation AG
Eaton Hydraulics Inc.
Eaton Industries GmbH
ECKELMANN AG
EES - Elektra Elektronik GmbH & Co Stör-
controller KG
EIA Electronics n.v.
Electronics Corporation of India Limited
ECIL
Elektronik-Systeme Lauer GmbH & Co. KG
EleSy Company
ELGO ELECTRIC GmbH
elrest Automationssysteme GmbH
EMKO Elektronik A. S.
EMTrust GmbH
EPEC Oy
epis Automation GmbH & Co. KG
Erhardt + Leimer GmbH
Escarré, Automatización y Servicios, s.l.
esd electronic system design gmbH
esitron-electronic GmbH
Eule Industrial Robotics GmbH & Co. KG
Exertus Oy
EXOR Bediensysteme GmbH
Fastwel Inc.
Faude Automatisierungstechnik GmbH
Feller Engineering GmbH
Ferrocontrol Steuerungssysteme GmbH
& Co.
Festo AG & Co. KG
Forbes Marshall pvt Ltd.
frenzel + berg electronic GmbH & Co. KG
Ganz Transelektro Közlekedési Rt.
Gebrüder Trox GmbH
Googol Technology (HK/SZ) Limited
Grossenbacher Systeme AG
Hangzhou ReboTech Co., Ltd.
Hans Turck GmbH & Co. KG
Helmut Mauell GmbH
Hesmor GmbH
Hilscher Swiss GmbH
Hirschmann Automation and Control GmbH
Hitachi Industrial Equipment Systems Co.
Ltd.
HollySys Co. Ltd
Hottinger Baldwin Messtechnik GmbH
HYDAC System GmbH
IDS GmbH
IEP Ingenieurbüro für Echtzeitprogrammie-
rung GmbH
ifm electronic gmbh
Industrial Computer Source (Deutschland) GmbH
Ingeteam Technology, S.A.

Inter Control Hermann Köhler Elektrik GmbH & Co.KG
ISG-Industrielle Steuerungstechnik GmbH
Janz Tec AG
JUMO GmbH & Co. KG
Karl E. Brinkmann GmbH
KEBA AG
KEP France SA
Kinco Electric (Shenzhen) Ltd.
kk-electronic a/s
Kontron AG
Kuhnke Automation GmbH & Co. KG
Lenord, Bauer & Co. GmbH
Lenze AG
LTi DRiVES GmbH
M-System Co. Ltd
MANZ Automation AG
MCC Singrock Electric Technology Co., Ltd.
Meier GmbH & Co.KG
Mestronic Steuerungstechnik GmbH
MicroNet Automation GmbH
Mikrap AG
Mita-Teknik A/S
Mitsubishi Electric EUROPE B.V.
MKT-Systemtechnik GmbH & Co.KG
MOBA Mobile Automation AG
MOBIL ELEKTRONIK GmbH
MOEHWALD GmbH
Montelec Montajes Electrónicos S.L.
Moog GmbH
MSC-Tuttlingen GmbH
MT ElectroniX GmbH
Müller-Elektronik GmbH & Co. KG
NETWORK Corporation
NEURON
NUM AG
OPIT Solutions GmbH
Ormec Systems Corp.
Owen Co.
Parker Hannifin GmbH Electromechanical Division Hauser
Pearson Engineering Ltd.
Pleiger Elektronik GmbH & Co. KG
PMA Prozeß- und Maschinen-Automation GmbH
PRAXIS Automation Technology B.V.
PRIMA ELECTRONICs S.p.A.
PSG Plastic Service GmbH
PsiControl Mechatronics
QUIN Systems Ltd.
RAFI GmbH & Co. KG
Raskat PKO SAO
Real Time Automation
repas AEG Automation GmbH
Resotec Realtime Software Technik GmbH
Ritter Elektronik GmbH
RLDA (NIL AP)
SAACKE GmbH & Co. KG
SABO Elektronik
SAE IT-systems GbmH & Co.KG
Schneider Electric Industries S.A.S.
Schraml GmbH
Schub’s Antriebstechnik GmbH
Schweitzer Engineering Laboratories, Inc.
Sensor-Technik Wiedemann GmbH
SETEX Schermuly textile computers GmbH
SEW EURODRIVE GmbH & Co. KG
Shanghai Pal-Fin Automatic Control T
Technology Co. Ltd.
Shanghai Sheng Mao Control Euipment
SIEB & MEYER Elektronik GmbH
SITEK S.p.A. Produzioni Elettroniche
SKB PSIS, Ltd
SMART Electronic Development GmbH
SND Schmidt Nachrichten- und Datentech-
nik GmbH
sontheim Industrie Elektronik GmbH
STANGE Elektronik GmbH
Störk-Tronic Störk GmbH & Co. KG
Sütron electronic GmbH
TCE TeleControlExpert GmbH
Technion Oy / Ltd.
Telestar S.r.l.
Tornatech Inc.
TQ-Components GmbH
TQ-Systems GmbH
TRsystems GmbH
TTControl S.r.l.
Veltru AG
Völkel Mikroelektronik GmbH
WABCO GmbH & Co. OHG
Wachendorff Elektronik GmbH & Co.KG
WAGO Kontakttechnik GmbH & Co. KG
Weber Schraubautomaten GmbH
Wieland Electric GmbH
Wipotec Wiege- und Positioniersysteme GmbH
Xuzhou Hirschmann Electronics Co., Ltd.
ZheJiang SUPCON Electronics Co. Ltd.
Ziehl-Abegg AG

Anexo H. Parametrização Autómato

Para parametrizar a comunicação com o variador de velocidade é necessário primeiro definir a que velocidade vai ser realizada a comunicação, ver Figura 97, de seguida é necessário adicionar o variador ao barramento de comunicação e definir qual é número de escravo que ele vai ter.

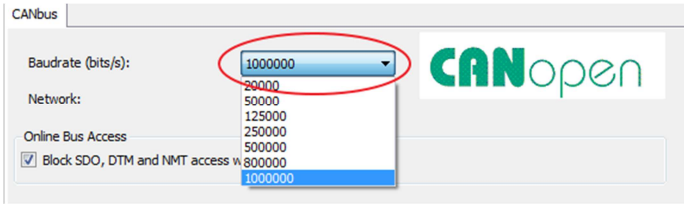


Figura 97 – Escolha velocidade comunicação CANopen

Os valores que são parametrizados na Figura 97 e Figura 98 têm que ser iguais aos que foram colocados no variador de velocidade (Tabela I. 1 – Parâmetros de configuração variador de velocidade)

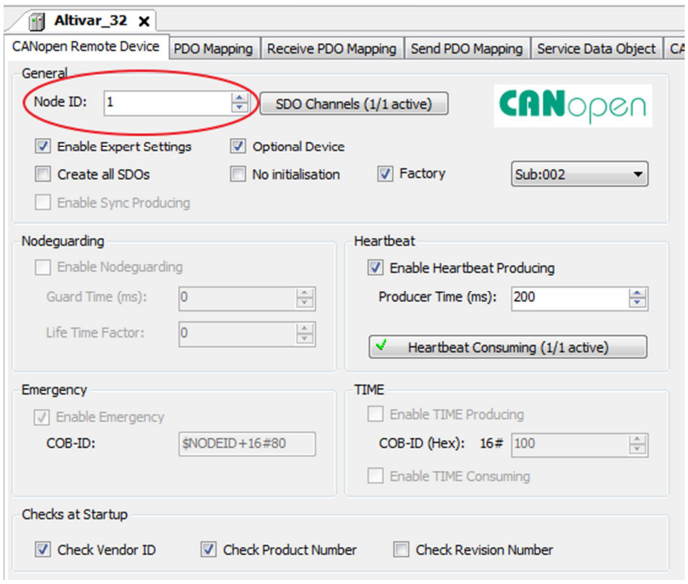


Figura 98 - Escolha número nó do escravo (variador)

Após a escolha dos parâmetros de comunicação é necessário mapear as variáveis que se pretende ler e escrever bidireccionalmente entre o autómato e variador. Essa definição é efetuada no autómato sob a forma de *Process data object* (PDO) (Figura 99).

Tabela 27 – Mapeamento variáveis comunicação

PDO Envio (Autómato –> Variador)			
Variável	Endereço simbólico	Endereço	Descrição
Control word		%QW2	Variável de controlo do variador
LFR(8502)	qiAltivar_32_LFR_8502	%QW3	Variável de referência de velocidade
PDO Receção (Autómato <– Variador)			
Variável	Endereço simbólico	Endereço	Descrição
OL1R(5212)	iuiAltivar_32_OL1R_5212	%IW5	Variável de estado das saídas digitais físicas
Statusword	iuiAltivar_32_Statusword	%IW6	Variável de estado do variador
ETI(3206)	iuiAltivar_32_ETI_3206	%IW7	Varável estendida de estado do variador
Control effort	iiAltivar_32_Control_Effort	%IW8	Variável velocidade saída
IL1R(5202)	iuiAltivar_32_IL1R_5202	%IW9	Variável de estado das saídas digitais físicas

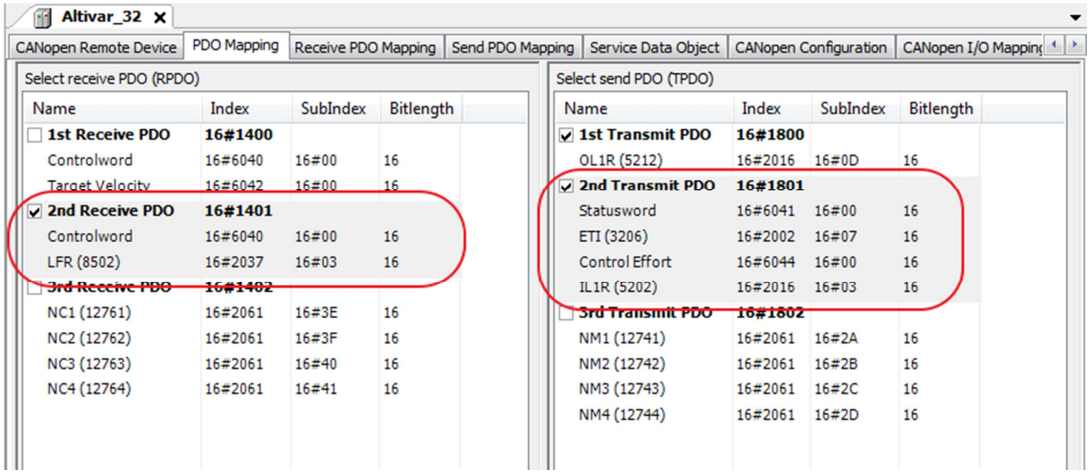


Figura 99 – Mapeamento CANopen das variáveis a comunicar

As variáveis Controlword e Statusword são representadas como dois bytes. O controlo do variador pode ser efetuado escrevendo um valor nos dois bytes ou então pode ser efetuado atuando os bits que os compõe individualmente (Tabela 28). O mesmo se aplica ao estado do variador, ou se lê o valor dos dois bytes ou dos bits individuais que os compõe (Tabela 29).

Tabela 28 – Mapeamento dos bits da Controlword

Controlword	
Bit	Descrição
0	Ligar
1	Habilitar tensão AC de alimentação
2	Paragem rápida (0=Paragem emergência)
3	Habilitar operação(1=Comando Funcionamento)
4	Reservado
5	Reservado
6	Reservado
7	Limpar erros
8	Interromper movimento
9	Reservado
10	Reservado

Controlword	
Bit	Descrição
11	Parâmetro livre programável pelo fabricante
12	Parâmetro livre programável pelo fabricante
13	Parâmetro livre programável pelo fabricante
14	Parâmetro livre programável pelo fabricante
15	Parâmetro livre programável pelo fabricante

Tabela 29 – Mapeamento dos bits da Statusword

Statusword	
Bit	Descrição
0	Pronto para ligar
1	Ligado
2	Operação habilitada
3	Erro
4	Tensão habilitada
5	Paragem rápida
6	Falha tensão de alimentação AC
7	Aviso/alarme
8	Reservado
9	Remoto ativado (comando ou referência via comunicação)
10	Posição/velocidade alcançada
11	Limite ativo – referência fora dos limites
12	Reservado
13	Reservado
14	Parâmetro livre programável pelo fabricante
15	Parâmetro livre programável pelo fabricante

No que respeita ao encoder a sua parametrização é relativamente fácil. Respeitando o esquema de ligações da Figura 74, a configuração é simples, basta definir as unidades de escalonamento, ou seja, por cada volta do veio do motor obtêm-se 1024 impulsos no encoder, mas por vezes há necessidade de ajustar essa relação para o veio final que controla uma determinada aplicação. No caso em estudo os 1024 impulsos correspondem a 16 unidades do utilizador. Para ativar a contagem de impulsos e obter o valor absoluto de impulsos, basta chamar um bloco de função (ver Figura 100) específico que já vem incorporado nas bibliotecas padrão do Somachine.

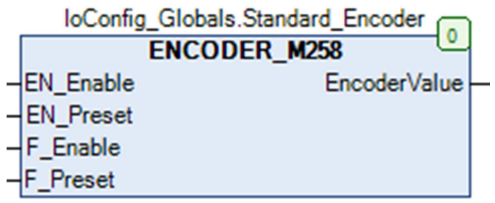


Figura 100 - Bloco de função ENCODER_M258

No bloco de função ENCODER_M258 temos vários parâmetros de entrada e saída (ver Tabela 30), através desses parâmetros é possível efetuar habilitar a contagem (EN_Enable

e F_Enable) ou colocar um novo valor de início de contagem (por exemplo 0, para reiniciar a contagem) (EN_Preset e F_Preset).

Tabela 30 – Descrição dos parâmetros do bloco de função ENCODER_M258

Entrada		
Parâmetro	**Tipo de dados**	**Significado**
EN_Enable	BOOL	Habilita a contagem
EN_Preset	BOOL	Habilita a predefinição de valor
F_Enable	BOOL	Inicia contagem
F_Preset	BOOL	Coloca o valor predefinido
Saída		
Parâmetro	**Tipo de dados**	**Significado**
EncoderValue	DINT	Valor atual de contagem do *encoder*

No capítulo das calibrações houve a necessidade de calibrar as variáveis de entrada erro_fuzz e variacao_erro_fuzz. Também foi necessário calibrar a variável de saída vel_pretendida. Para efetuar a calibração foi usado o bloco de função SE_TBX.FB_Scaling com os parâmetros de entrada e saída da Tabela 31.

Tabela 31 – Descrição dos parâmetros do bloco de função SE_TBX.FB_Scaling

Entrada		
Parâmetro	**Tipo de dados**	**Significado**
i_xEn	BOOL	Habilita a o bloco de função
i_rIput	REAL	Valor de entrada a ser calibrado
i_rMinIput	REAL	Mínimo do valor de entrada
i_rMaxIput	REAL	Máximo do valor de entrada
i_rMinOput	REAL	Mínimo do valor de saída
i_rMaxOput	REAL	Máximo do valor de entrada
Saída		
Parâmetro	**Tipo de dados**	**Significado**
q_rOput	REAL	Valor da saída calibrada

No gráfico da Figura 101 pode-se observar a relação da entrada a calibrar com a saída calibrada.

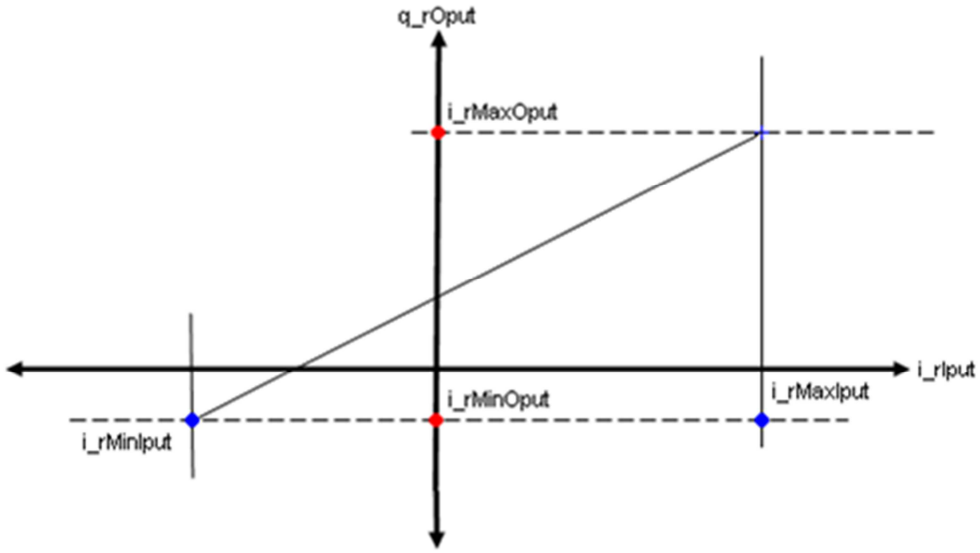


Figura 101 – Comportamento do bloco de função SE_TBX.FB_Scaling

A necessidade de calibração surge devido ao facto de se ter definido que as variáveis entrada e de saída do sistema com uma variação de 0 a 1000.

A variável do erro_fuzz é uma das variáveis de entrada que tem que ser calibrada. O valor do erro é dado pela seguinte expressão: $erro = posicao_{pretendida} - position\_atual$, que é a variável de entrada a ser calibrada (Figura 102). O valor mínimo e máximo de saída do bloco de calibração estão definidos, sento o mínimo 0 e o máximo 1000. O mínimo e o máximo de entrada estão colocados como variáveis (erro_min e erro_max) para ser possível ajustar os valores conforme o comportamento do sistema.

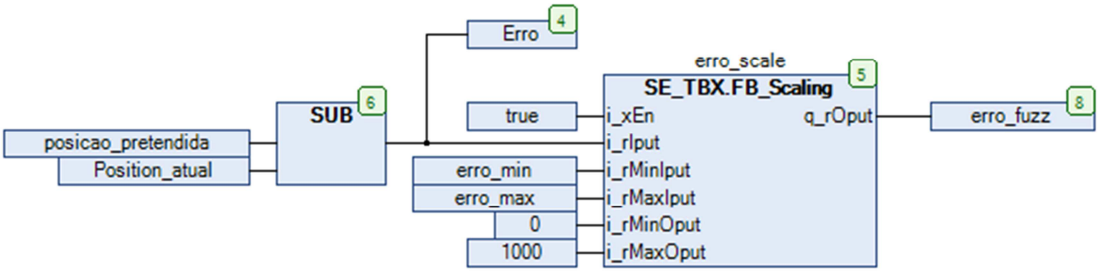


Figura 102 – Calibração do erro

A variável do variacao_erro_fuzz é uma das variáveis de entrada que também tem que ser calibrada. O valor da variação do erro é pela leitura do valor da velocidade do motor (velocidade lida por comunicação). A velocidade do motor tem um sinal contrário ao desejado para funcionar como variação do erro, face a essa questão, o valor da velocidade tem que ser multiplicado por -1, após essa multiplicação essa é a variável de entrada a ser calibrada (Figura 103). O valor mínimo e máximo de saída do bloco de calibração estão

definidos, sento o mínimo 0 e o máximo 1000. O mínimo e o máximo de entrada estão colocados como variáveis (variacao_min e variacao_max) para ser possível ajustar os valores conforme o comportamento do sistema.

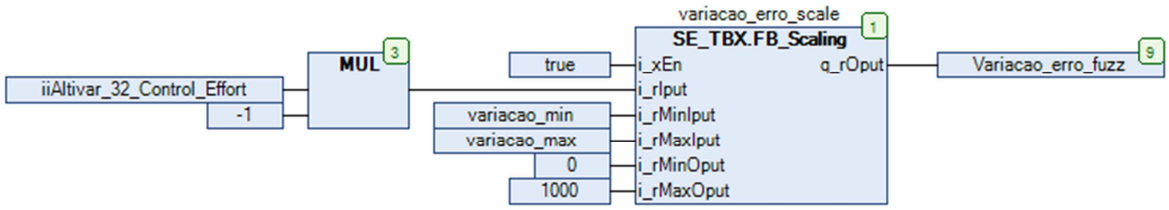


Figura 103 – Calibração da variação do erro

A variável vel_pretendida resulta da calibração de um valor de entrada com um valor mínimo de 0 e máximo 1000 (ver Figura 104). Ao contrário das variáveis de entrada, a variável de saída tem os limites de saída em variáveis (vel_min, vel_max) para desta forma ser possível, no decorrer da aplicação, ajustar as velocidades mínima e máxima do sistema.

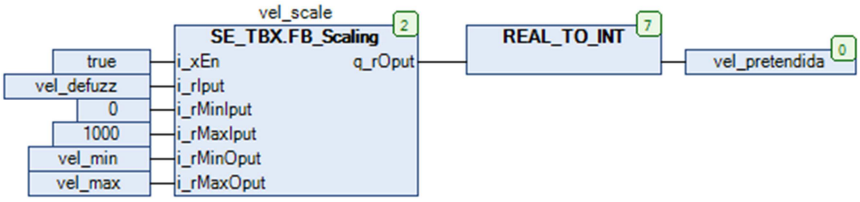


Figura 104 – Calibração da velocidade pretendida

Anexo I. Parâmetros de configuração variador velocidade

Tabela I. 1 – Parâmetros de configuração variador de velocidade

Parâmetro	**Valor**
Potência do motor	0,37 kW
Tensão alimentação motor	230 V
Corrente nominal motor	1,9 A
Velocidade nominal motor	1370 rpm
Proteção térmica motor	1,9 A
Velocidade máxima	50 Hz
Velocidade mínima	0 Hz
Rampa de aceleração	0,1 s
Rampa de desaceleração	0,1 s
Constante K *loop* corrente	20
Integral velocidade	200 ms
Ganho proporcional velocidade	7%
Tipo controlo motor	Controlo vetorial sem *feedback*
Entrada de referência	CANopen
Entrada de controlo	CANopen
Modo de paragem normal	Paragem por rampa
Endereço CANopen	1
Baudrate CANopen	1 Mbps

Anexo J. Programa PLC (Formato digital)

Project Documentation

File:	fuzzy 12.project
Date:	11/9/2014
Profile:	V1.52.15.0