Aplicação de lógica Fuzzy e Arduino em controle de processos

Luis Antonio Prado & Yvo Marcelo Chiaradia Masselli

Abstract - ***The growing use of hardware platforms for low cost has stimulated the development of applications in diverse areas. This paper presents an example of practical application using the Arduino platform and fuzzy logic to control a conveyor belt. The differences are related to the final cost, ease of deployment, flexibility and extensive documentation available, both software and hardware.***

Index Terms — Arduino, Fuzzy Logic, Artificial Intelligence.

Resumo – A crescente utilização de plataformas de hardware de baixo custo tem estimulado o desenvolvimento de aplicações nas mais diversas áreas. Este trabalho apresenta um exemplo de aplicação prática utilizando a plataforma Arduino e a lógica Fuzzy no controle de uma esteira de transporte. Os diferenciais estão relacionados ao custo final, facilidade de implementação, flexibilidade e vasta documentação disponível, tanto de software quanto de hardware.

Palavras chave — Arduino, Lógica Fuzzy, Lógica Nebulosa, Conjuntos Difusos, Inteligência Artificial.

I. INTRODUÇÃO

A área de inteligência artificial (IA) é uma das áreas que tem crescido significativamente em função do número de aplicações práticas em que está presente. Atualmente podemos nos deparar com a utilização de técnicas de IA em *games*, mecanismos de busca na Internet, controle de estabilidade em veículos e diversas outras áreas. De forma simples e objetiva, pode-se entender a área de inteligência artificial como aquela que reúne um conjunto de técnicas que buscam esboçar características dos seres vivos, como tomar decisões, jogar xadrez e etc.

A inteligência artificial possui vários ramos que são aplicados em diversos setores para percepção e aprendizado, e também em áreas especificas como demonstrações de matemáticos, jogos, na medicina no diagnostico de doenças, processamento de imagens e na resolução de muitos outros problemas que são considerados tecnicamente difíceis de serem resolvidos por meio de sistemas comuns [1]. Neste trabalho é explorada uma técnica de IA conhecida por lógica difusa, ou *Fuzzy Logic*.

Trabalho de Conclusão de Curso apresentado ao Instituto Nacional de Telecomunicações, como parte dos requisitos para a obtenção do Certificado de Pós-Graduação em Engenharia de Sistemas Eletro-Eletrônicos, Automação e Controle Industrial. Orientador: Prof. Yvo Marcelo Chiaradia Masselli. Trabalho aprovado em 03/2014

II. TÉCNICAS DE IA APLICADAS À AUTOMAÇÃO

Uma das principais características da automação é a capacidade de diminuir a interferência e a dependência humana em um processo ou uma maquina, e assim tornar o sistema mais eficiente, seguro e otimizado. A inteligência artificial associa-se à área da automação com o propósito de tornar mais "inteligente" o controle de processos e fazer com que as decisões resultantes destes sistemas sejam cada vez mais próximas daquelas tomadas por um especialista humano.

III. LÓGICA FUZZY

A lógica *Fuzzy* é utilizada para tratar problemas onde a imprecisão e a incerteza são variáveis complexas que dificultam a implementação nos moldes convencionais [2]. O modo convencional apresenta entradas e saídas de forma binária, por exemplo, 0 ou 1, verdadeiro ou falso, e também com valores em uma faixa especifica e finita. A lógica *Fuzzy* trabalha com valores que geralmente não são bem definidos numericamente.

Os valores utilizados na lógica *Fuzzy* podem ser mostrados por meio de expressões lingüísticas conforme cada problema que se quer trabalhar, por exemplo, no controle de um ar condicionado a temperatura pode estar quente para uma pessoa e ao mesmo tempo frio para outra pessoa, assim, o resultado gerado por um sistema *Fuzzy* pode ser 65% de quente e 35% frio [3]. Pode haver também resultados classificados em verdade, muito verdade, falso, muito falso.

Para lidar com informações imprecisas e vagas presentes em alguns problemas, Zadeh, em 1965 desenvolveu a teoria dos conjuntos nebulosos e em 1978, a teoria de possibilidades. Esta pode ser comparada com a teoria de probabilidade, porém menos restritiva [4]. Ambas, quando associadas, possibilitam o tratamento de imprecisões, inconsistências e incertezas comuns nas variáveis envolvidas em problemas do mundo real.

A. *Variáveis linguísticas:*

As variáveis lingüísticas são utilizadas para fazer uma caracterização aproximada de um fenômeno complexo ou com pouca precisão na sua definição. Estas variáveis simplificam o tratamento de problemas e sistemas complexos que demandam muito processamento para serem tratados de maneira convencional, utilizando métodos matemáticos [5]. Um exemplo de variável lingüística é a temperatura, que pode assumir valores como muito baixa, baixa, média, alta e muito alta apresentado na figura [1]. Graficamente estas variáveis são representadas por funções de pertinência. Estas tem por objetivo identificar a qual termo linguístico um valor numérico está associado e com qual intensidade. O valor de intensidade é chamado grau de pertinência.

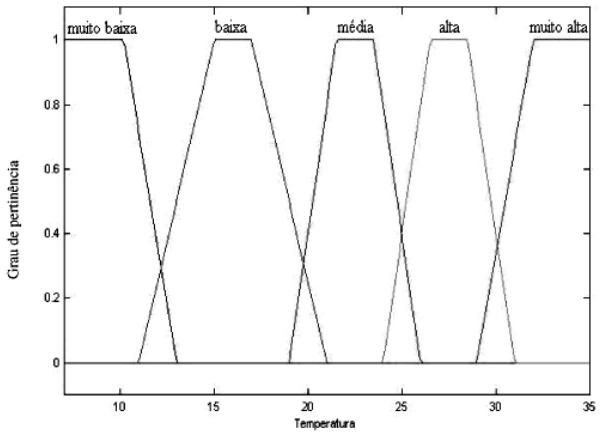

Figura 1 – Exemplo de Variável Lingüística para temperatura

B. *Controlador Fuzzy:*

Os controladores convencionais utilizam a modelagem matemática para fazer o controle de seus processos, enquanto que os controladores *Fuzzy* utilizam-se do conhecimento de um especialista na área em questão, representado na forma de regras lógicas, chamadas de regras de produção. Os controladores *Fuzzy* realizam, basicamente, três etapas: fuzzyficação, inferência e defuzzyficação [4][5][6]. Os blocos que compõe um controlador *Fuzzy* são apresentados na figura 2.

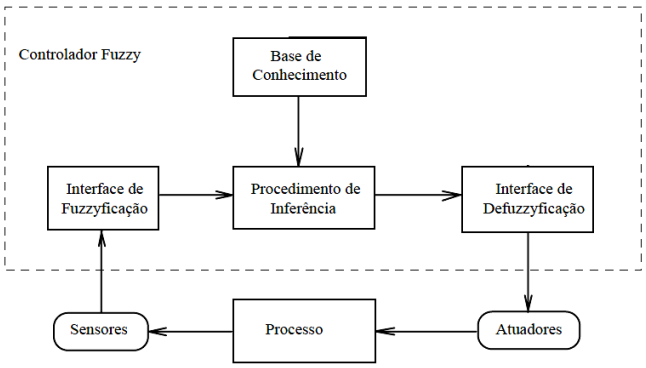

Figura 2 – Controlador Fuzzy

- Interface de fuzzyficação:

Esta interface é responsável associar um termo linguístico a cada valor numérico de entrada.

- Inferência:

Este bloco realiza o processo inferência entre as variáveis linguísticas de entrada e as variáveis linguísticas de saída. Isto significa identificar as regras que são ativadas na base de conhecimento.

- Base de Conhecimento:

A base de conhecimento é formada por uma base de dados e uma base de regras. A base de dados fornece dados numéricos e a base de regras caracteriza o funcionamento do sistema. Esta geralmente é desenvolvida por um especialista no processo.

- Interface de defuzzyficação:

Esta interface transforma as variáveis lingüísticas de saída em valores numéricos para serem enviados aos atuadores.

Os conjuntos nebulosos permitem que seus elementos possuam um grau de pertinência para que possam representar um universo multivalente e não bivalente (falso ou verdadeiro) como os conjuntos convencionais.

IV. ARDUINO

O Arduino foi criado em 2005 e é uma plataforma open-source que pode ser facilmente adaptada de acordo com as necessidades de cada projeto [7]. Sua principal vantagem é a facilidade de utilização e de customização, pois toda a documentação, incluindo esquema elétrico e arquivos CAD estão disponíveis para download. Até mesmo o ambiente de desenvolvimento possui código fonte aberto para que qualquer pessoa possa customizá-lo. O ambiente de desenvolvimento esta disponível para as plataformas Windows e Linux.

A. *Arquitetura:*

O Arduino é comercializado em diversas versões que se diferem, basicamente, pela quantidade de entradas e saídas digitais e analógicas.

O Arduino utilizado no presente trabalho é o Duemilanove [8] apresentado na figura 3.

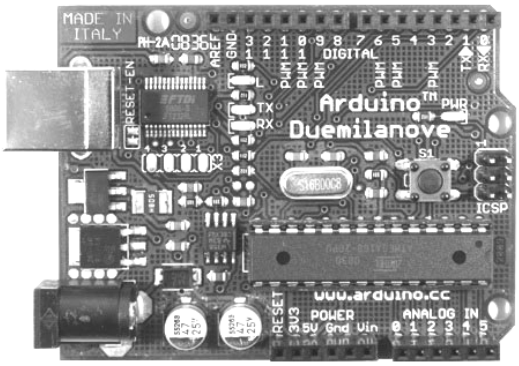

Figura 3 – Arduino Duemilanove

Esta versão é baseada no microcontrolador ATmega328, possui uma memória *flash* de 32 KB para armazenar o código fonte e disponibiliza 14 pinos de entrada e saída digitais sendo que podem ser utilizados 6 pinos para saída PWM e mais 6 pinos de entradas analógicas.

A integração com outros sistemas como computadores, outros Arduinos e microcontroladores é facilitada por meio da comunicação serial. Por se tratar de uma plataforma aberta, a comunidade Arduino é muito ativa e participativa desenvolvendo e disponibilizando diversas bibliotecas gratuitas para facilitar o desenvolvimento de software. Entre as diversas bibliotecas, a SoftwareSerial, que possibilita a comunicação serial.

As principais características deste Arduino são apresentadas na

Tabela 1 – Principais Características do Arduino Duemilanove:

Tabela 1 – Principais Características do Arduino Duemilanove

Microcontrolador	ATmega168
Voltagem de operação	5V
Voltagem de entrada recomendada	7-12V
Limite de voltagem de entrada	6-20V
Pinos digitais de entrada/saída	14 (6 podem ser utilizados como PWM)
Entradas Analógicas	6
Corrente DC por I/O	40 mA
Memória flash	32 KB as quais 2 KB são utilizados para o *bootloader*
SRAM	2 KB
EEPROM	1 KB
Clock	16 MHz

B. Software:

A linguagem de programação utilizada no Arduino é baseada em um *framework open-source* de programação para microcontroladores chamado Wiring [9]. Além do ambiente oficial disponibilizado no próprio site do Arduino, existem vários outros *Integrated Development Environment (*IDE) como o Atmel Studio e Eclipse e ainda vários projetos de IDE como o Ardublock [10], S4A-Scratch for Arduino [11] e Minibloq [12]. Toda a documentação e *Application Programming Interface (*API) da linguagem oficial esta disponível no site oficial do Arduino. A comunidade Arduino também disponibiliza diversos códigos de exemplo e projetos feitos por outros usuários do Arduino. Também é possível programar o Arduino utilizando blocos em linguagem C/C++.

V. Implementação Lógica Fuzzy e Arduino

O exemplo selecionado para mostrar a utilização da lógica *Fuzzy* em um problema real da automação é o controle de esteiras.

Em usinas de açúcar e álcool a matéria prima é a cana-de-açúcar que chega em caminhões e é despejada em esteiras que que as levam aos trituradores, como ilustrado pela figura 4 [14][15].

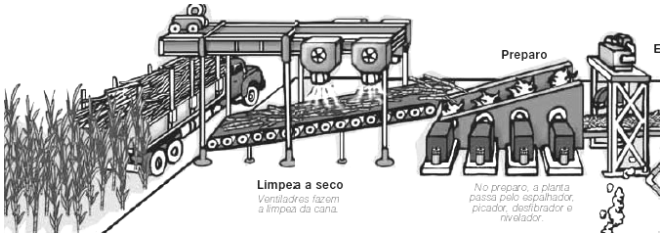

Figura 4 – Chegada da cana-de-açúcar

A. Variáveis Linguísticas utilizadas:

Serão utilizadas três variáveis cuja suas funções de pertinência, assim como seus valores de entrada, foram escolhidos arbitrariamente uma vez que o objetivo primário consiste na avaliação dos resultados obtidos pelo controlador *Fuzzy* executado pelo Arduino.

- Peso na Esteira:

Esta variável indica a quantidade de cana-de-açúcar na esteira, seus valores possíveis são leve, normal e pesado. O grau de pertinência para cada estado pode ser visualizado na figura 5. Esta é a variável de entrada do processo.

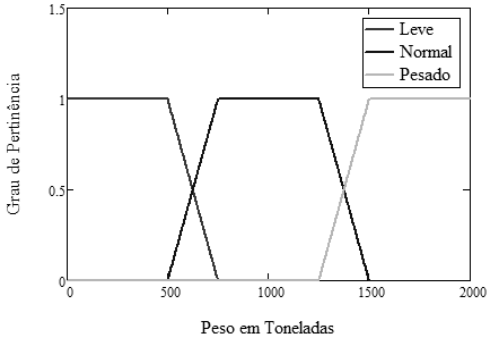

Figura 5 – Variável Peso na Esteira

- Utilização do Processo de Preparo:

Indica a porcentagem de utilização dos equipamentos no processo de preparo da cana-de-açúcar. Durante este processo ela passará por espalhadores, picadores, desfibradores e niveladores. Esta variável é utilizada na entrada do processo. Seus possíveis valores são: alta, normal, baixa e parado, como pode ser observado na figura 6.

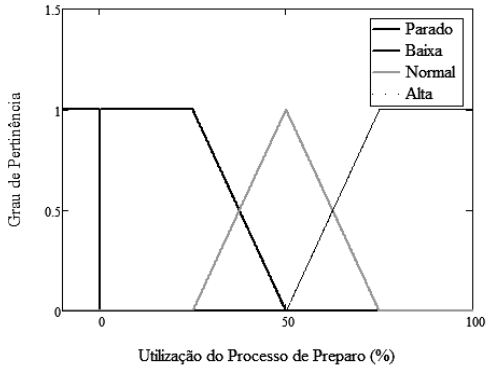

Figura 6 – Variável Utilização do Processo de Preparo

- Potência do Motor da Esteira:

Esta é a variável de saída que será utilizada para o controle da esteira. Indica a potência que deverá estar o motor da esteira para que a velocidade de alimentação do processo de preparo seja a ideal. Os possíveis valores são alta, normal, baixa e parado como pode ser visualizado na figura 8.

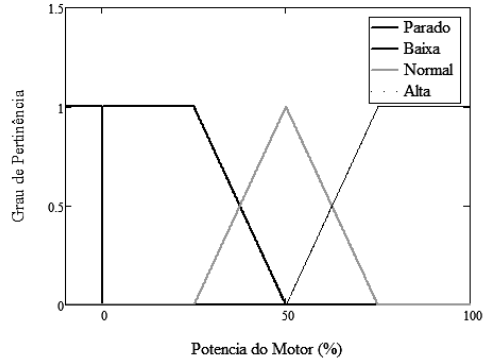

Figura 4 – Variável Potência do Motor

B. Base de regras utilizada:

As regras utilizadas para a implementação do sistema estão descritas na tabela abaixo.

Tabela 2 – Base de Regras

Peso na Esteira	Utilização do Processo de Preparo	Potencia do Motor
Leve	Baixa	Alta
Normal	Baixa	Alta
Pesado	Baixa	Normal
Leve	Normal	Normal
Normal	Normal	Normal
Pesado	Normal	Baixa
Leve	Alta	Normal
Normal	Alta	Baixa
Pesado	Alta	Parado
----------	Parado	Parado

Estas foram criadas para ilustrar o funcionamento da lógica Fuzzy sendo que para o sistema real, novas regras podem ser criadas.

C. Biblioteca utilizada:

Para trabalhar com a lógica *Fuzzy* no Arduino, existe a biblioteca *Embedded Fuzzy Logic Library* (eFLL) que foi desenvolvida utilizando-se apenas a biblioteca "stdlib.h", de forma que a eFLL seja utilizada em qualquer sistema que suporte linguagem C. A eFLL foi criada pelo *Robotic Research Group* na Universidade Estadual do Piauí [16].

Seu motor de inferência utiliza os métodos de Max-Min e Mínimo de Mamdani. Para o processo de DeFuzzyficação o método utilizado é o Centro de Área.

Os principais componentes da biblioteca eFLL são apresentados à seguir.

- Objeto Fuzzy:

Este é o objeto que engloba todo o sistema *Fuzzy*, com ele manipulamos os conjuntos *Fuzzy*, variáveis lingüísticas de entrada e saída e a base de regras.

- Objeto FuzzyInput:

Este objeto representa uma variável lingüística de entrada do sistema. As variáveis de entrada são modeladas por meio dos FuzzySet.

- Objeto FuzzySet:

É utilizado para a modelagem da função de pertinência de uma variável (Baixo, Alto, Normal), é possível modelar funções triangulares, trapezoidais e *singleton*. O construtor deste objeto é:

FuzzySet (float a, float b, float c, float d)

Com estes 4 valores de entrada formam-se quatro pontos, (a,0), (b,1), (c,1) e (d,0). Por meio destes pontos defini-se a função de pertinência como pode ser observado em Figura 5 – Exemplos de funções de pertinência com FuzzySet.

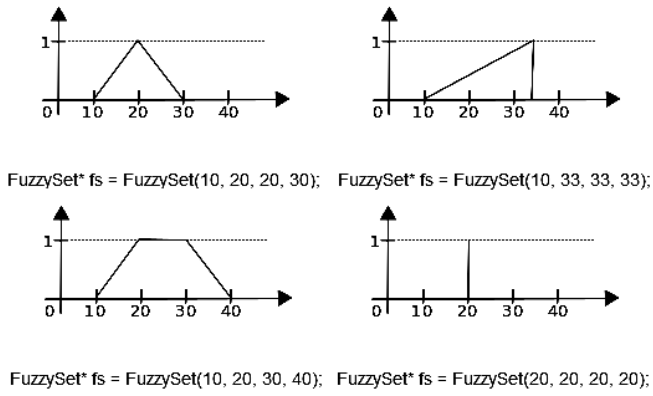

Figura 5 – Exemplos de funções de pertinência com FuzzySet

A função de pertinência é obtida ao ligar os pontos (a,0), (b,1), (c,1) e (d,0) nesta mesma ordem.

- Objeto FuzzyOutput:

  Assim como o FuzzyInput, o objeto FuzzyOutput agrupa vários FuzzySet, a diferença é que os FuzzySet do FuzzyOutput modelaram as variáveis de saída.

- Objeto FuzzyRule:

  Este objeto representa uma regra que será adicionada a base de conhecimento do objeto Fuzzy. Ele é composto por um FuzzyRuleAntecedent e um FuzzyRuleConsequent. A base de conhecimento deve conter um ou mais objetos FuzzyRule.

- Objeto FuzzyRuleAntecedent:

  Guarda a condição de ativação de uma regra.

- Objeto FuzzyRuleConsequent:

  Guarda qual será o resultado se a regra for ativada.

D.Codificação:

Com a utilização da biblioteca eFLL, a codificação fica mais simples de ser entendida, tornando a manutenção e também a inclusão de novas funcionalidades, entradas e saídas uma tarefa mais fácil de ser implementada.

O primeiro passo da implementação é criar o objeto que armazenará todos os componentes do sistema *Fuzzy*:

```
Fuzzy* Fuzzy = new Fuzzy();
```

A função "void setup()" será executada apenas uma vez na inicialização do sistema e esta contém todo o código de setup do sistema Fuzzy e a configuração da porta de saída.

Na função setup a variável de entrada "Peso na Esteira" é criada utilizando o objeto FuzzyInput e seu id (1) deve ser passado como parâmetro do construtor FuzzyInput:

```
FuzzyInput* pesoEsteira = new FuzzyInput(1);
```

A modelagem da variável é feita utilizando os objetos FuzzySet. O próximo código faz a modelagem para os estados Leve, Normal e Pesado para a variável de entrada "Peso na Esteira":

```
FuzzySet* leve = new FuzzySet(0, 0, 500, 750);
FuzzySet* normal = new FuzzySet(500, 750, 1250, 1500);
FuzzySet* pesado = new FuzzySet(1250, 1500, 2000, 2000);
```

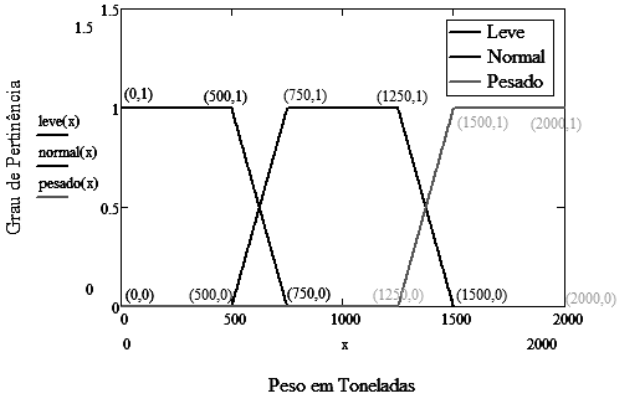

Figura 6 – Modelagem FuzzySet

Após a modelagem, associamos os estados possíveis para a variável FuzzyInput pesoEsteira:

```
pesoEsteira->addFuzzySet(leve);
pesoEsteira->addFuzzySet(normal);
pesoEsteira->addFuzzySet(pesado);
```

E finalmente deve-se incluir a FuzzyInput no objeto Fuzzy:

```
Fuzzy->addFuzzyInput(pesoEsteira);
```

Estes passos devem ser feitos para todas as variáveis de entrada e saída. A única diferença entre entrada e saída é o tipo do objeto que para entrada é o FuzzyInput e para a saída é o FuzzyOutput:

```
FuzzyOutput* potMotor = new FuzzyOutput(1);
```

Após cadastrar todas as entradas e saídas do sistema, é a hora de incluir a base de regras. Na biblioteca eFLL, a regra é formada por uma condição antecedente e por um estado de conseqüência. O primeiro passo é fazer a condição antecedente, o código a seguir é a implementação da primeira regra que é:

Se o peso na esteira **é** Leve **E** a utilização do processo de preparo **é** Baixa **ENTÃO** potencia do motor **será** Alta.

Primeiro passo é criar a condição de ativação da regra:

```
FuzzyRuleAntecedent* ifLeveEBaixa = new
FuzzyRuleAntecedent();
```

Para fazer a operação **E** (AND) é utilizado a função joinWithAND passando por parâmetro os FuzzySet correspondentes de cada entrada:

```
ifLeveEBaixa->joinWithAND(leve, baixa);
```

O estado de conseqüência a ativação da regra é modelado pelo sistema Fuzzy utilizando o FuzzyRuleConsequent:

```
FuzzyRuleConsequent* entaoPotAlta = new
FuzzyRuleConsequent();
```

Para associar o valor de saída, utiliza-se a função addOutput que adiciona o FuzzySet referente ao estado de saída ao FuzzyRuleConsequent:

```
entaoPotAlta->addOutput(PotAlta);
```

Ao final, é criada a própria regra em si utilizando o objeto FuzzyRule com parâmetros de entrada do construtor o id da regra, o FuzzyRuleAntecedent e a FuzzyRuleConsequent, por fim adicionar a FuzzyRule criada, no objeto Fuzzy:

```
FuzzyRule* FuzzyRule01 = new FuzzyRule(1,
ifLeveEBaixa, entaoPotAlta);
Fuzzy->addFuzzyRule(FuzzyRule01);
```

Estes passos devem se repetir até criar todas as regras que compõe a base de conhecimento do sistema Fuzzy.

A entrada do sistema *Fuzzy* é feita na função “loop”, que é a responsável em fazer a leitura das portas de entrada e escrever a saída do sistema Fuzzy na porta de saída em intervalos de tempos em tempos.

A leitura da entrada “Peso na Esteira” é feita pela porta analógica 0 e a leitura da “Utilização do Processo de Preparo” é feita pela porta analógica 2, o próximo código faz a leitura das portas analógicas:

```
int pesoEsteira = analogRead(A0);
int utPreparo = analogRead(A2);
```

Os valores lidos, estarão na faixa 1~1024 e por este motivo devem ser convertidos para a faixa de valores correspondente de cada variável de entrada. O mesmo processo deve ser feito para a variável de saída que também deve possuir valores entre 0 ~ 1024.

O valores convertidos são passados para o objeto Fuzzy utilizando a função setInput passando como parâmetros o id da entrada e o valor numérico da entrada. O id 1 corresponde a variável criada anteriormente pesoEsteira e o id 2 corresponde a variável quantPreparo:

```
Fuzzy->setInput(1, pesoEsteira);
Fuzzy->setInput(2, quantPreparo);
```

Para iniciar o processo de Fuzzyficação, é utilizada a função fuzzify():

```
Fuzzy->fuzzify();
```

Esta função faz a inferência conforme o valor de entrada de cada FuzzyInput.

Para fazer a deFuzzyficação, utiliza-se a função defuzzify passando como parâmetro o ID do objeto FuzzyOutput correspondente a variável de saída “Potencia Motor”:

```
int output = Fuzzy->defuzzify(1);
```

Finalmente, convertemos o valor da variável output, que será entre 0 e 100, para o valor utilizado na saída PWM do Arduino, que é entre 0 e 255, para assim controlar a velocidade do motor da esteira. Um passo importante é a configuração da porta de saída que será utilizada como PWM na função setup:

```
pinMode(3, OUTPUT);
```

A próxima linha de código escreve o valor convertido na porta 3:

```
analogWrite(3, valorConvertido);
```

E. Montagem do Circuito:

Para a implementação do protótipo foram utilizados dois potenciômetros de 1k ohms para simular as variáveis de entrada “Peso na Esteira” e “Utilização do processo de Preparo”. Quando a resistência medida no potenciômetro for igual a 1k ohm, indicará entrada com valor Mínimo (0V) e quanto menor a resistência medida, maior será o valor de entrada (próximo a 5V).

Para simular a variável de saída “Potencia no Motor da Esteira”, será utilizado um LED. Quanto maior o valor da variável de saída “Potência do Motor”, mais o LED vai brilhar. O acionamento do LED será feito utilizando o PWM (Pulse Width Modulation) no Arduino.

A modulação por largura de pulso será utilizada para o controle da tensão fornecida para o motor que no protótipo será simulado por um LED. A placa Duemilanove possui 6 saídas que podem ser utilizadas para o PWM.

O conceito de PWM consiste em regular a tensão média na carga por meio de ondas quadradas, com pulsos de largura variável. A figura 11 mostra exemplos para tensão na carga de 0%, 25%, 50%,75% e 100% da tensão total (5v).

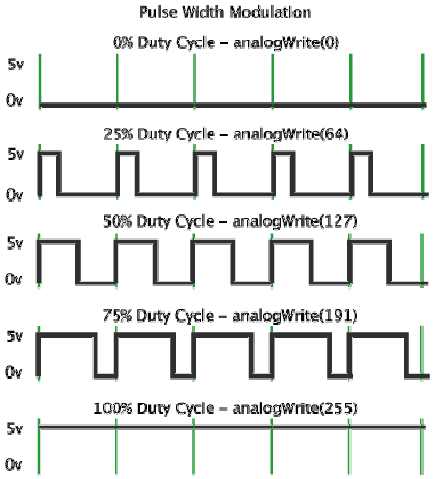

Figura 11 – Modulação por Largura de Pulso

VI. CONCLUSÕES

Existem varias técnicas de Inteligência Artificial que podem ser utilizadas na automação de modo a diminuir cada vez mais a interferência humana e também aumentar a eficiência do sistema.

Este trabalho teve como foco principal a lógica *Fuzzy* e o Arduino. A lógica *Fuzzy* se mostrou ser uma técnica muito simples de ser implementada e apresentou resultados satisfatórios durante os testes. Por aceitar entradas com conceitos lingüísticos (Baixo, Alto,..) que são facilmente compreendidos pelo ser humano (o operador), a manutenção do sistemas e supervisão do mesmo fica mais fácil.

O Arduino por sua vez é considerado também uma ótima escolha para a automação de sistemas que não dependem de grande poder de processamento. O suporte para o desenvolvimento do software e hardware para o Arduino é muito bom, pois a comunidade Arduino é ativa e colaborativa onde é possível encontrar diversos exemplos de implementações e bibliotecas para facilitar do processo de codificação. Uma vantagem em utilizar o Arduino é o custo do desenvolvimento que é baixo. Com a utilização dos Shields pode-se incrementar as funções da placa Arduino utilizada e assim possibilitar também novas funcionalidades como o monitoramento remoto, facilitando assim as tarefas de gerência e sistemas supervisores.

A implementação exemplo, mostrou como é fácil a integração entre a Lógica *Fuzzy* e o Arduino e os testes e simulações realizados foram satisfatórios por ter tempo de resposta semelhante a outros sistemas implementados utilizando hardware mais poderosos. Os valores de saída obtidos foram comparados com a simulação deste sistema Fuzzy no software FuzzyTECH e os resultados estão na tabela 3.

Tabela 3 – Resultados Obtidos

Peso na Esteira	Utilização/ Preparo	FuzzyTECH*	Arduino*
448	74	50	50
600	30	78	75
900	45	57	57
1100	50	50	50
1600	80	0	0
100	15	87	79
1000	20	87	79
1500	25	50	50
550	35	72	71
700	50	50	50
1400	50	27	28
500	70	50	50
1500	90	0	0

** Valor de Saída Fuzzy*

Ao analisar as informações da tabela, percebe-se que as saídas dos dois sistemas possuem baixa variação, apenas dois testes tiveram uma variação de 8%, esta baixa variação é uma característica que valida a implementação.

A biblioteca utilizada possui a vantagem de tornar mais fácil a implementação, pois o desenvolvedor precisa apenas modelar as entradas e saídas do sistema *Fuzzy*, não se preocupando com o processo de inferência e defuzzyficação. Por outro lado essa vantagem pode se tornar uma desvantagem se o desenvolvedor precisar utilizar outro método de inferência ou defuzzyficação diferente do padrão utilizado pela biblioteca.

REFERÊNCIAS

[1] RUSSELL, Stuart, NORVIG, Peter. Inteligência Artificial. 2. Ed. Rio de Janeiro: Campos, 2004.

[2] L. A. Zadeh. Fuzzy sets. Fuzzy Sets, Information and Control, 8:338 – 353., 1965.

[3] V Escola de Redes Neurais, Promoção: Conselho Nacional de Redes Neurais pp. c073-c090, 19 de julho, 1999 - ITA, São José dos Campos – SP

[4] L. A. Zadeh. Fuzzy sets as a basis for a theory of possibility. Fuzzy Sets and Systems, 1:3–28, 1978.

[5] Gomide, Fernando A. C., Gudwin, Ricardo R., Tanscheit, Ricardo, Conceitos Fundamentais da Teoria de Conjuntos Fuzzy, Lógica Fuzzy e Aplicações.

[6] FILHO, Fernando de M. L., GoSMANN, Hugo Leonardo, BAUCHSPIESS, Adolfo, Controle Fuzzy para Sistema de Nível de Líquidos. XIV – Congresso Brasileiro de Automática, 2 a 5 Setembro 2002, Natal, RN.

[7] Arduino - HomePage. Disponível em: http://arduino.cc/ Acessado em 16 de Agosto 2013.

[8] Arduino - ArduinoBoardDuemilanove. Disponível em: http://arduino.cc/en/Main/ArduinoBoardDuemilanove Acessado em 17 de Agosto 2013.

[9] Wiring. Disponível em: http://www.wiring.org.co/ Acessado em 17 de Agosto 2013.

[10] Scratch for Arduino – S4A. Disponível em: http://s4a.cat/ Acessado em 17 de Agosto 2013.

[11] Ardublock – A GRAPHICAL PROGRAMMING LANGUAGE FOR ARDUINO. Disponível em: http://blog.ardublock.com Acessado em 17 de Agosto 2013.

[12] Minibloq. Disponível em: http://blog.minibloq.org/ Acessado em 18 de Agosto 2013.

[13] MCROBERTS, Michael. Arduino Básico. 1; Ed. São Paulo: Novatec, 2011.

[14] A Usina por Dentro. Disponível em: http://www.weg.net/acucar-e-alcool/usina_por_dentro.html Acessado em 15 de Novembro 2013.

[15] Usina Virtual. Disponível em: http://www.unica.com.br/usina-virtual.php Acessado em 15 de Novembro 2013.

[16] eFLL – Embedded Fuzzy Logic Library. Disponível em: http://zerokol.com/post/51e9324ee8 4c55a1f5000007/1/en Acessado em 15 de Novembro 2013.