UNIVERSIDADE DO ESTADO DO AMAZONAS

ESCOLA SUPERIOR DE TECNOLOGIA

LUCICLEYTON MEDEIROS DO NASCIMENTO

CONTROLE DE TEMPERATURA DO TIPO PID APLICADO A ZONA DE AQUECIMENTO DE UM FORNO ELÉTRICO RESISTIVO

Manaus

2018

LUCICLEYTON MEDEIROS DO NASCIMENTO

CONTROLE DE TEMPERATURA DO TIPO PID APLICADO A ZONA DE AQUECIMENTO DE UM FORNO ELÉTRICO RESISTIVO

Monografia desenvolvida durante a disciplina Projeto Final II e apresentada à banca avaliadora do Curso de Tecnologia em Automação Industrial da Escola Superior de Tecnologia da Universidade do Estado do Amazonas, como pré-requisito para a obtenção do título de Tecnólogo em Automação Industrial.

Orientador: Israel Mazaira Morales, Dr.

Manaus

2018

Universidade do Estado do Amazonas – UEA
Escola Superior de Tecnologia - EST

Reitor:
Cleinaldo de Almeida Costa
Vice-Reitor:
Mario Augusto Bessa de Figueiredo
Diretor da Escola Superior de Tecnologia:
Roberto Higino Perereira da Silva
Coordenador do Curso de Tecnologia em Automação Industrial:
Daniel Guzman Del Rio

Banca Avaliadora composta por: *Data da defesa: 22/06/2018.*
Prof. Israel Mazaira Morales (Orientador)
Prof. Daniel Guzman Del Rio
Prof. Luiz Benigno Corrales Barrios

CIP – Catalogação na Publicação

Medeiros do Nascimento, Lucicleyton

Controle de temperatura do tipo pid aplicado a zona de aquecimento de um forno elétrico resistivo / Lucicleyton Medeiros do Nascimento; orientado por Israel Mazaira Morales. – Manaus: 2018.
86p.: il.

Trabalho de Conclusão de Curso (Graduação em Tecnologia em Automação Industrial). Universidade do Estado do Amazonas, 2018.

1. Controle PID. 2. Controle *ON-OFF*. 3. Controle de Potência PWM I. Morales, Israel Mazaira.

LUCICLEYTON MEDEIROS DO NASCIMENTO

CONTROLE DE TEMPERATURA DO TIPO PID APLICADO A ZONA DE AQUECIMENTO DE UM FORNO ELÉTRICO RESISTIVO

Monografia desenvolvida durante a disciplina Projeto Final II e apresentada à banca avaliadora do Curso de Tecnologia em Automação Industrial da Escola Superior de Tecnologia da Universidade do Estado do Amazonas, como pré-requisito para a obtenção do título de Tecnólogo em Automação Industrial.

Nota Obtida:_____ (_____)

Aprovada em _____/_____/_____.

Área de concentração: Controle Industrial

BANCA EXAMINADORA

_____
Orientador: Israel Mazaira Morales, Dr.

_____
Avaliador: Daniel Guzman Del Rio, Dr.

_____
Avaliador: Luiz Benigno Corrales Barrios, Dr.

Manaus, 2018

DEDICATÓRIA

Dedico ao meu pai Luiz Ferreira do Nascimento e minha mãe Iranilza da Silva Medeiros pela oportunidade que me deram de concluir o ensino médio e poder ter chegado onde estou.

AGRADECIMENTOS

Ao Deus vivo. Aos professores André Prints e Paulo Cavalcante pelas valiosas dicas. À minha querida esposa Hellen Nascimento e aos meus filhos Luiz Fernando e Maria Fernanda. Ao Sr. Carlos Delcaro pela ajuda na idéia do trabalho. À minha grande irmã Geise Gabrielle pela ajuda na montagem do protótipo e aos meus colegas de classe Claudio Calvacante e Felipe Luniere pelo apoio durante todo o curso.

RESUMO

O presente trabalho apresenta o desenvolvimento de um controle do tipo PID aplicado a zona de aquecimento de um forno elétrico resistivo, está dividido em quatro capítulos: referencial teórico, método proposto, implementação do projeto e resultados obtidos. O primeiro capítulo apresenta a fundamentação teórica dos assuntos mais relevantes para o desenvolvimento deste trabalho, são eles: sistemas de controle industrial, onde são descritas formas de controle automático; microcontroladores e seu funcionamento em um sistema de controle; controle de potência *PWM* (modulação da potência consumida por uma carga); comunicação serial RS232 e Linguagem de programação. O segundo capítulo é a metodologia descrevendo as etapas e materiais necessários para o desenvolvimento do controle tipo PID. O terceiro capítulo é a implementação que faz execução dos passos propostos na metodologia, detalhando as etapas descritas na metodologia. O quarto capítulo é a análise e interpretação dos resultados, constam a descrição dos comportamentos observados dos resultados obtidos por meio da compração entre o controle do tipo *ON-OFF* e PID. Ao final é apresentada a conclusão com base nos resultados obtidos.

Palavras-chaves: Controle PID; Controle de Potência PWM; Controle *ON-OFF*.

ABSTRACT

The present work presents the development of a PID control applied to the heating zone of a resistive electric furnace. It is divided in four chapters: theoretical reference, proposed method, project implementation and results obtained. The first chapter presents the theoretical basis of the most relevant subjects for the development of this work, they are: industrial control systems, where are described forms of automatic control; microcontrollers and their operation in a control system; PWM power control (modulation of the power consumed by a load); RS232 serial communication and programming language. The second chapter is the methodology describing the steps and materials required for the development of PID control. The third chapter is the implementation that performs the proposed steps in the methodology, detailing the steps described in the methodology. The fourth chapter is the analysis and interpretation of the results, the description of observed behaviors of the obtained results through the purchase between the control of type ON-OFF and PID. At the end the conclusion is presented based on the results obtained.

Keywords: PID control; PWM Power Control; ON-OFF control.

LISTA DE FIGURAS

LISTA DE EQUAÇÕES

LISTA DE GRÁFICOS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

PID	proporcional, integral e derivativo
CC	Corrente Contínua
GND	*Ground* ou Terra
USB	*Universal Serial Bus* (Barramento Serial Universal)
CPU	*Central Processor Unit* (Unidade de Processamento Central)
ULA	Unidade Lógica Aritmética
PIC	Controlador Integrado de Periféricos
BPS	*Bits* por segundo
TTL	Lógica Transistor-Transistor
PWM	Modulação de Largura de Pulso
FET	*Field Effect Transitor* (Transistor de Efeito de Campo)
IGBT	*Insulated Gate Bipolar Transistor* (Transistor Bipolar de Porta isolada)
SCR	*Silicon Controlled Rectifier* (Retificador Controlado de Silício)
ASCII	Código Padrão Americano Para o Intercâmbio de Informação
RAD	Desenvolvimento Rápido de Aplicações

SUMÁRIO

INTRODUÇÃO

O uso de algumas técnicas rudimentares de controle na Grécia e na Alexandria são descritas em documentos históricos. Nos séculos XVII e XVIII, muitos dispositivos de controle foram desenvolvidos, visando resolver alguns problemas práticos. Mas foi a revolução industrial no século XVIII, com o desenvolvimento de processos industriais que impulsionou o desenvolvimento das técnicas de controle. (FUNDAMENTOS DE CONTROLE CLÁSSICO, 2012?).

Com a revolução industrial surgiram as máquinas a vapor, o tear mecânico e as máquinas de fiar. Outro fator importante foi a introdução do processo de produção em série, as mercadorias passaram a ser produzidas de maneira uniforme e padronizadas.

Com o avanço dos processos industriais, várias áreas desse segmento foram se desenvolvendo, é o caso do forno industrial, muito usado em fundição, processos de pintura industrial, extrusoras de carga, enfim uma série de utilidades (FONTOURA, 2010).

"O forno elétrico resistivo é um dos equipamentos térmicos mais conhecidos e sua utilização na indústria e em centros de pesquisa não é recente. Atualmente a temperatura dos fornos varia entre 400° e 1.100°C[1], podendo atingir temperaturas mais altas". (GUERRA, 2006, p.10).

Os fornos elétricos a resistência, são largamente utilizados na indústria, sendo constituídos basicamente de uma ou mais câmaras (zonas) de aquecimento, um conjunto de resistências elétricas e uma carcaça metálica.

A maioria dos processos onde são utilizados fornos a resistência requerem uma grande precisão na temperatura da câmera de aquecimento. Em alguns processos (fabricação de cerâmica, por exemplo) a temperatura não deve variar bruscamente. (GUERRA, 2006).

Geralmente passam por essa câmera de aquecimento, produtos e matéria prima que precisam ser submetidos a uma temperatura especifica, ou seja, materiais que necessitam estar

[1] O símbolo ºC (grau *Celsius*) designa uma unidade de temperatura.

sujeitos a uma temperatura estabilizada, em algumas indústrias, o controle de temperatura dessas zonas de aquecimento é do tipo *ON-OFF*.

A forma mais simples de controle de temperatura é o sistema *ON-OFF* ou liga-desliga, a grande desvantagem desse sistema é que a grandeza controlada (temperatura) não estabiliza em nenhum ponto e sim oscila entre o ponto desejado, indo do limite inferior ao superior.

Quando é necessário um controle de temperatura com precisão, uma opção é utilizar o controle PID, que é na verdade a combinação de três elementos: Proporcional, integral e derivativo. (CONTROLE DE AUTOMAÇÃO INDUSTRIAL, 2012?).

Neste trabalho é apresentado o desenvolvimento de um sistema de controle de temperatura do tipo PID aplicado a zona de aquecimento de um forno elétrico resistivo, com o objetivo de tornar nulo ou próximo de nulo a inércia térmica (oscilação de temperatura) tradicional do sistema *ON-OFF*. Um protótipo para simulação da aplicação realizada foi implementado e foram realizadas comparações entre os dois sistemas (*ON-OFF* e PID para verificar a diferença em seus comportamentos.

Este trabalho está dividido em quatro capítulos e estruturado da seguinte forma:

No capítulo 1 – Referêncial Teórico: são apresentados os fundamentos teóricos mais relevantes.

No capítulo 2 - Método Proposto: neste capítulo esta relatado a estrutura do sistema executado e detalhamento das tecnologias utilizadas para a elaboração do projeto.

No capítulo 3 – Implementação do Projeto: descreve detalhadamente os procedimentos realizados durante a execução do projeto.

No capítulo 4 – Resultados Obtidos: estão relacionados os resultados obtidos com a implementação do controle de temperatura PID.

1 REFERENCIAL TEÓRICO

Neste capítulo serão apresentados os fundamentos teóricos necessários para uma melhor compreensão do trabalho. Os tópicos estão divididos da seguinte forma:

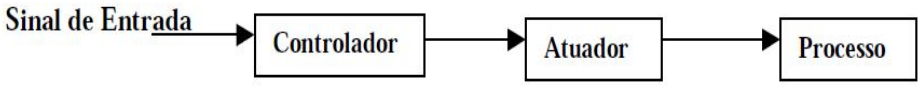
a) Sistemas de controle industrial, sensores e transdutores: aborda as diversas formas de desenvolvimento de controle automático. Interfaceamento entre o sistema físico e o sistema de controle eletrônico.
b) Microcontroladores: Sua função em um sistema de controle.
c) Controle de potência: Modulação da potência consumida por uma carga.
d) Comunicação Serial RS232: Principio de seu funcionamento.
e) Linguagem de programação: Sua empregação no controle automático.

1.1 SISTEMAS DE CONTROLE INDUSTRIAL

Existem várias formas de se implementar sistemas de controle automático, entretanto, a mais utilizada são os sistemas eletroeletrônicos devido principalmente a versatilidade e dinamismo necessários à um controle de processo. Além disso, sistemas elétricos são mais fáceis de implementar que sistemas dinâmicos. Dado que um sistema de controle é predominante elétrico e os processos envolvem transformações mecânicas, químicas e físicas deve-se converter o sinal de um controlador eletrônico no sinal adequado ao processo, tanto do ponto de vista da natureza, quanto do ponto de vista de magnitude. Este elemento é o atuador, ele quem atua direto sobre o processo, sempre em resposta a saída do controlador. (CONTROLE DE AUTOMAÇÃO INDUSTRIAL, 2012?).

Para que o controlador gere o sinal de controle para o atuador gerar o sinal de controle do atuador ele precisa de uma referência, ou seja, um sinal na sua entrada que diga ao controlador o que ele deve fazer com o processo. Este é o sinal de referência, ou sinal de entrada. (CONTROLE DE AUTOMAÇÃO INDUSTRIAL, 2012?). A Figura 1 ilustra o relacionamento entre o controlador, atuador e processo:

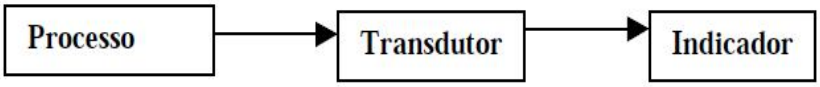
Figura 1- Relação entre controlador, atuador e processo.



Fonte: (CONTROLE DE AUTOMAÇÃO INDUSTRIAL, 2012?, p. 4)

Em um sistema de controle é preciso saber como anda o processo e obter informações a respeito de parâmetros do mesmo. Ou seja, é necessário um dispositivo capaz de converter uma grandeza física do processo em uma grandeza elétrica para que seja possível medir o andamento do processo. Este elemento é o transdutor e ele se relaciona com o processo conforme a Figura 2.

Figura 2- Relação do transdutor com o processo.



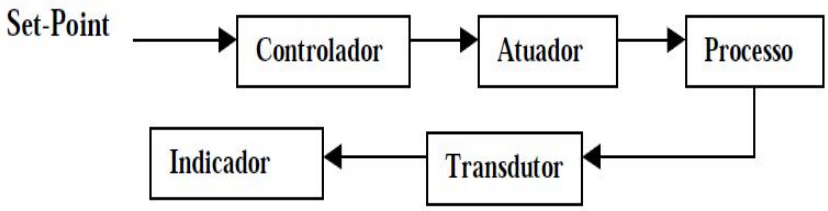
Fonte: (CONTROLE DE AUTOMAÇÃO INDUSTRIAL, 2012?, p. 4)

1.1.1 Sistemas de controle em malha aberta e malha fechada

Com relação a forma de implementação os sistemas de controle podem ser classificados de duas formas: em malha aberta e em malha fechada.

a) **Malha aberta:** Quando o controlador gera o sinal para o atuador, com base no sinal piloto, sem obter nenhuma informação sobre o andamento do processo. Ou seja, é um sistema sem realimentação, sendo que o sinal de entrada é o próprio *set-point*, conforme mostra a Figura 3.

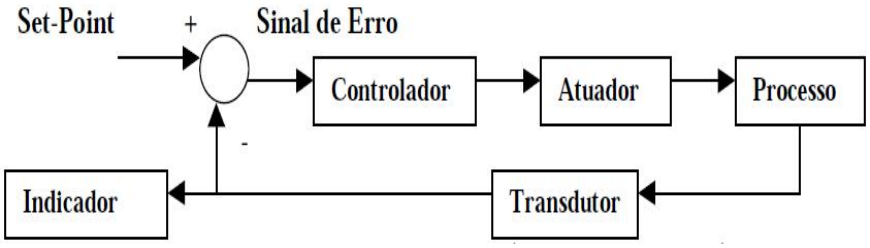
Figura 3- Malha aberta (sem realimentação)



Fonte: (CONTROLE DE AUTOMAÇÃO INDUSTRIAL, 2012?, p. 4)

b) **Malha fechada:** Quando o controlador gera o sinal para o atuador, com base no sinal piloto, porém agora ele recebe informação sobre o andamento do processo, através de um transdutor. O sinal de entrada, no caso, corresponde a diferença entre o *set-point* e o sinal do transdutor, por isso, também é chamado de sinal de erro, conforme mostra a Figura 4.

Figura 4- Malha fechada (com realimentação)



Fonte: (CONTROLE DE AUTOMAÇÃO INDUSTRIAL, 2012?, p. 5)

No caso da malha aberta o transdutor e o indicador são itens opcionais, não sendo importantes para o controle. Já no caso da malha fechada o indicador é um item opcional.

A malha fechada apresenta algumas vantagens em relação à malha aberta, principalmente no que tange a menor sensibilidade a interferências e ruídos. Isto porque uma vez que o sistema é realimentado, qualquer desvio gera um erro que tende a ser compensado. Além disso, o sistema fica mais independente dos parâmetros da planta, já que ele passa a atuar sobre o sinal do erro. Entretanto, existem também desvantagens como o custo mais elevado e a possibilidade do sistema atingir a instabilidade quando o ganho do controlador for muito alto. (CONTROLE DE AUTOMAÇÃO INDUSTRIAL, 2012?).

1.1.2 Transistório e Indicadores de performance

Quando o *set-point* é ajustado a saída leva um tempo para atingir seu valor final. Este tempo é chamado de transitório e é muito importante seu conhecimento para fins de determinação de comportamento do sistema e avaliação do desempenho do controlador. Para fins de avaliação da performance de um sistema de controle, existem alguns indicadores básicos, muito utilizados para a especificação de um sistema de controle. São os principais:

a) **Regulação**: É uma avaliação do sistema com relação a sua capacidade de reduzir o erro entre o valor real da grandeza física controlada e o valor esperado ao final do

transitório. O erro no caso é chamado de erro em regime permanente. Se o erro for grande, a regulação do sistema é ruim, se o erro for pequeno a regulação será boa.

b) **Estabilidade**: É a capacidade que um sistema tem de dada certa entrada limitada, fornecer uma resposta limitada. Ou seja, se o processo converge para algum ponto, para uma dada entrada é um sistema estável, se não é um sistema instável.
c) **Tempo de acomodação**: É o intervalo de tempo em que ajustada uma entrada, o sistema demora a convergir. Ou seja, é o intervalo de tempo em que dura a fase de transitório.
d) **Tempo de subida**: É o tempo necessário para que a saída vá de 0 a 100%, ou de 10 a 90% do seu valor final.
e) **Sobrelevação**: Conhecido como "*overshoot*" é o valor máximo atingido pela grandeza física da planta em relação ao valor esperado. É medida em percentagem da entrada ajustada. Ocorre na fase do transitório.
f) **Sensibilidade**: Avaliação da mudança do comportamento do sistema frente a pequenas variações de parâmetros do sistema.
g) **Rejeição de distúrbios**: capacidade de um sistema de rejeitar distúrbios ou ruídos oriundos de perturbações no sistema.

1.1.3 Tipos de Controladores Industriais

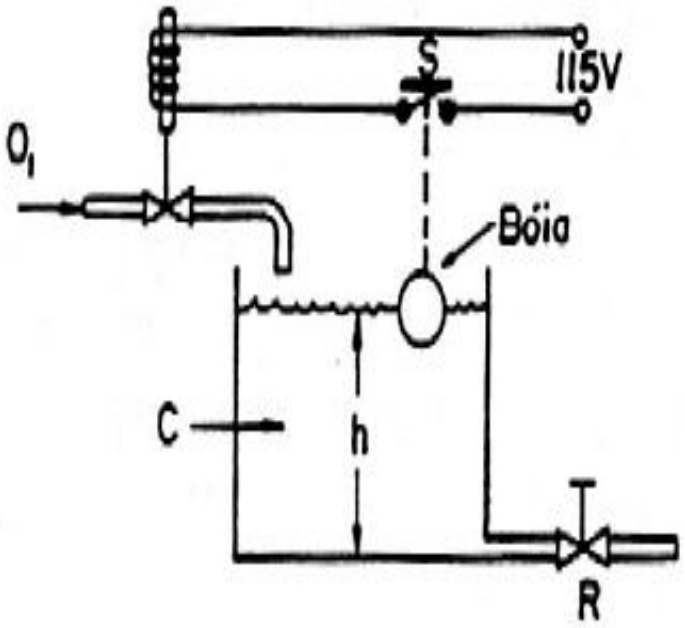
Há principalmente cinco tipos básicos de controladores usados largamente na indústria. São eles:

1.1.3.1 Controle *ON-OFF* / LIGA-DESLIGA ou de histerese

É a forma de controlar mais simples que existe e consiste em um circuito comparador que compara o sinal de entrada com dois sinais de referência, chamados de limite inferior e limite superior. Quando o sinal de entrada fica menor que o limite inferior, a saída do controlador é ativada e o atuador é acionado com sua potência máxima. Quando o sinal de entrada fica maior que o limite superior a saída é desligada e o atuador é desligado. A diferença entre o limite superior e o limite inferior é chamada de histerese. Normalmente, a histerese é ajustável de forma tal que o *set-point* fique entre o limite inferior e o superior. Desta forma o sistema de controle fica oscilando de um valor máximo a um mínimo e não atinge nenhum valor específico. (CONTROLE DE AUTOMAÇÃO INDUSTRIAL, 2012?).

Para exemplificar um controle *ON-OFF* é mostrado na Figura 5 um sistema de controle de nível. Neste sistema, para se efetuar o controle de nível utiliza-se um flutuado para abrir e fechar o contato (S), energizando ou não o circuito de alimentação da bobina de uma válvula do tipo solenoide. Este solenoide estando energizado permite a passagem da vazão máxima e estando desenergizado bloqueia totalmente o fluxo do líquido para o tanque. Assim este sistema efetua o controle estando sempre em uma das posições extremas, ou seja, totalmente aberto ou totalmente fechado. (OLIVEIRA, 1999)

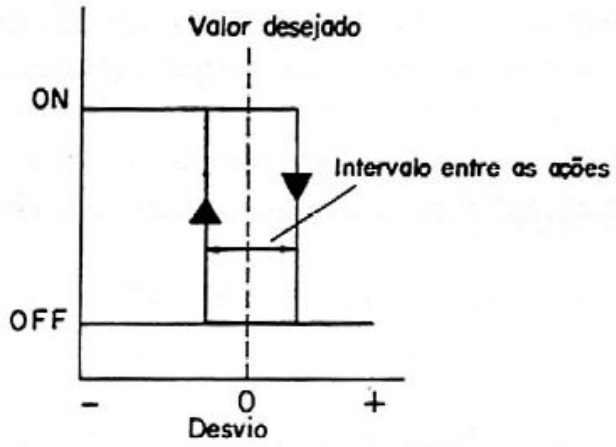
Figura 5- Sistema ON-OFF de controle de nível de liquido



Fonte: (OLIVEIRA, 1999, p. 24)

Neste tipo de controle sempre vai existir um intervalo entre o comando “liga” e o comando “desliga”, conforme mostra a Figura 6. Este intervalo diferencial faz com que a saída do controlador mantenha seu valor presente até que o sinal de erro tenha se movido ligeiramente além do valor zero. (OLIVEIRA, 1999)

Figura 6- Intervalo entre as ações de liga-desliga

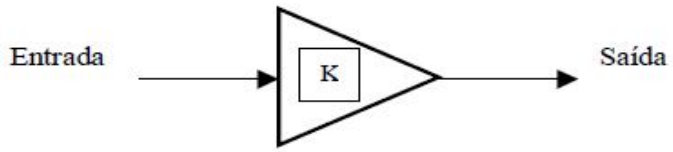


Fonte: (OLIVEIRA, 1999, p. 24)

1.1.3.2 Controle Proporcional ou P

O controle proporcional já é mais sofisticado que o controlador *ON-OFF*, dado que a resposta controle é proporcional ao sinal na sua entrada. Se o sinal na sua entrada é pequeno, a resposta será um valor pequeno também. Se a entrada for grande a saída será grande também. Em suma, um controlador proporcional é na verdade um amplificado, conforme representa a Figura 7.

Figura 7- Representação de um controlador proporcional

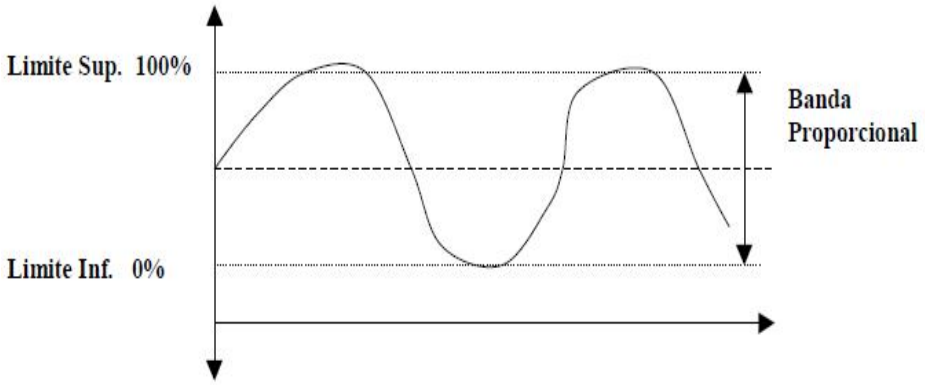


Fonte: (CONTROLE DE AUTOMAÇÃO INDUSTRIAL, 2012?, p. 7)

No caso a saída é um sinal K vezes maior que a entrada. Entretanto o sinal de saída não pode crescer indefinidamente, porque há limite tanto inferior quanto superior. Quando estes limites são atingidos é dito que o sistema saturou. Portanto, há uma região onde o sinal responde proporcionalmente ao sinal de entrada, e outra onde o sistema satura. Na Figura 8 abaixo, percebe-se que acima do limite superior, o atuador está com 100% de sua capacidade e abaixo do limite inferior o atuador está com 0% de sua capacidade, ou seja, totalmente

desligado. Na região entre o limite inferior e superior o atuador está com uma saída proporcional à entrada, esta região é chamada de banda proporcional do sistema. (CONTROLE DE AUTOMAÇÃO INDUSTRIAL, 2012?).

Figura 8- Representação banda proporcional



Fonte: (CONTROLE DE AUTOMAÇÃO INDUSTRIAL, 2012?, p. 7)

A banda proporcional de um sistema é dada de forma percentual e está relacionada com o ganho K do controlador. Ele é determinado conforme mostra a Equação 1.

Equação 1- Cálculo da banda proporcional

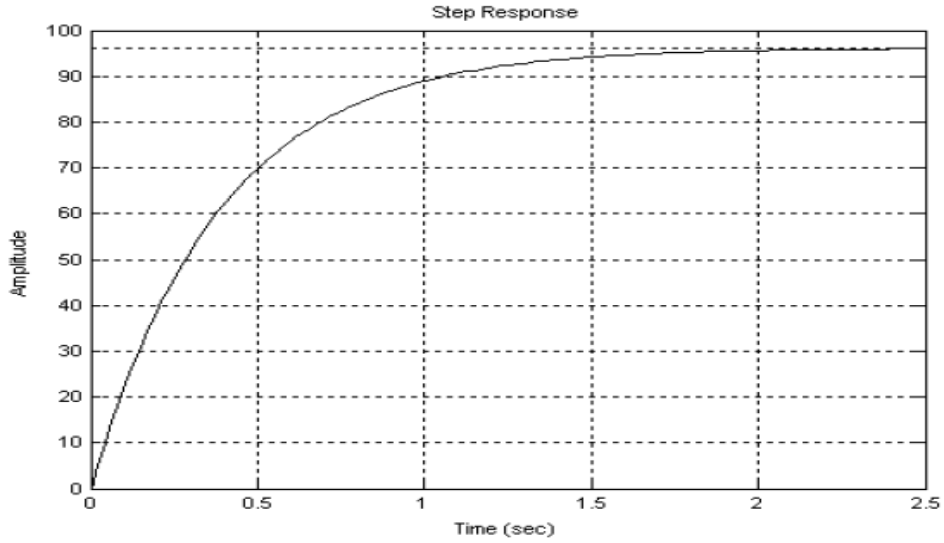
$$\text{Banda proporcional} = 100\% / K \text{ ; onde } K \text{ é o ganho do controlador.}$$

Muitos controladores possuem o ajuste da banda larga proporcional disponível. A técnica mais recomendada é deixar a banda proporcional no máximo possível e verificar a resposta do sistema.

O grande problema do controlador proporcional é que ele permite erros em regime, pois em sistema realimentados, a entrada do controlador P é o sinal de erro. Como o sinal de erro vai ficando pequeno a medida que se aproxima do valor do *set-point*, a saída do controlador que é proporcional a entrada vai ficando pequena também. Assim o sistema para mesmo sem ter atingido plenamente o *set-point*, permanecendo um erro sempre constante.

Na simulação mostrada na Figura 9, tem-se uma estufa com um *set-point* de 100ºC e um controlador P com um ganho K=10. Percebe-se que o sistema estabiliza em 96ºC, permanecendo um erro de 4ºC. Se o ganho for aumentado poderá reduzir o erro, mas sempre haverá um erro, por menor que seja.

Figura 9- Controlador P aplicado a uma estufa



Fonte: (CONTROLE DE AUTOMAÇÃO INDUSTRIAL, 2012?, p. 8)

1.1.3.3 Controlador Proporcional + Integral PI

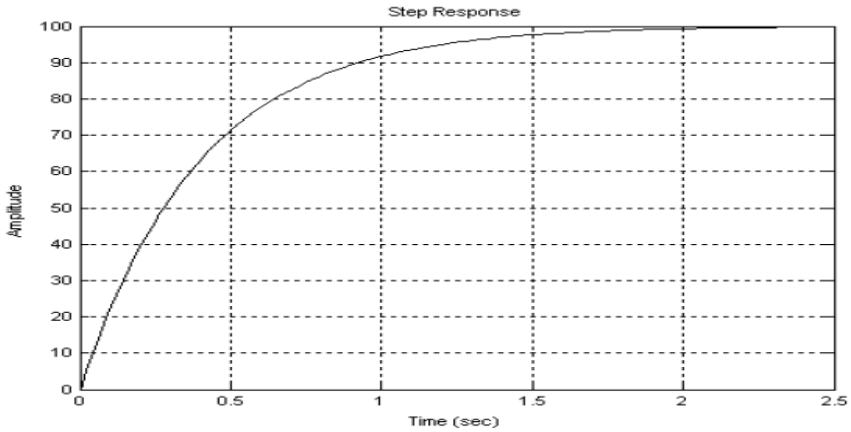
O controlador PI é uma combinação da ação proporcional com uma ação de integração. O integrador, dentre suas propriedades, permite com que o erro em regime do caso anterior seja zerado. Isto ocorre porque embora o erro possa ser pequeno, o integrador vai somando ao longo do tempo e a sua saída vai aumentando até que seja capaz de acionar o atuador. Assim sendo, quando o erro é grande o proporcional fornece uma saída grande e predomina sobre o integrador. Mas à medida que o sistema vai chegando perto do objetivo, o erro vai diminuindo e assim a resposta do proporcional vai ficando cada vez mais fraca. A partir desse ponto o domínio passa a ser do integrador.

Dessa forma o bloco integrador é usado quando se precisa de uma convergência precisa do valor, com erro muito pequeno. Da mesma forma que existe um ganho K para o proporcional, existe também um ganho KI para o integrador. O ajuste do ganho KI não deve ser indiscriminado, pois ele pode levar o sistema a se tornar muito lento as transações ou até mesmo levar o sistema a instabilidade. O melhor ajuste é uma combinação do ganho K e do ganho KI. É possível mediante simulações e outras técnicas encontrar o melhor ajuste possível.

Na simulação mostrada na Figura 10, retorna-se ao caso da estufa que fora ajustada para uma temperatura de 100ºC. Com o controlador somente proporcional, havia um erro de

4ºC, ou seja, a temperatura ficava a 96ºC. Com o integrador há o zeramento do erro e a temperatura atinge exatamente os 100ºC.

Figura 10- Controlador PI aplicado a uma estufa



Fonte: (CONTROLE DE AUTOMAÇÃO INDUSTRIAL, 2012?, p. 9)

1.1.3.4 Controlador Proporcional + Derivativo PD

Da mesma forma que o controle PI é uma combinação do controle proporcional e o controle integral, controle PD é uma combinação do controle proporcional e o controle derivativo. O derivativo é um bloco cuja saída é proporcional a variação do erro. Ou seja, se o erro estiver variando muito rápido ele atua fortemente visando minimizar ou eliminar esta variação.

Portanto é um bloco adequado para sistemas que precisam de um ataque rápido as variações de erro. Entretanto, se houver um erro de grande valor, mas variando lentamente, o sinal na saída do derivativo será baixo. Por isso, o derivativo nunca é usado sozinho, pois ele só atua nos momentos em que o erro varia rapidamente.

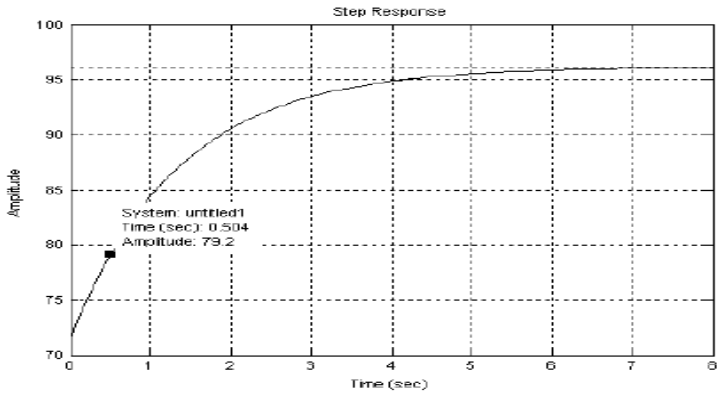
Além disso, o derivativo é sensível a ruídos que podem enganá-lo, fazendo acreditar que há uma transição brusca. Por isso o ganho do derivativo nunca é muito alto. Na verdade, evita-se o uso de derivativos. Quando o sistema não pode responder bem as variações bruscas de sinal então se apela para o derivativo.

O bloco derivativo não tem nenhuma influencia sobre o erro em regime. De modo geral, ele deixa o sistema mais rápido e reduz a máxima sobrelevação. O derivativo também tem um ganho chamado KD.

Na simulação mostrada na Figura 11, ainda usando o exemplo da estufa ajustada para 100ºC, retirando o integrador e adicionando o derivativo, percebe-se que o derivativo não atua sobre o erro em regime, o erro de 4ºC voltou e, portanto o sistema agora converge a 96ºC como antes.

Em compensação, na região de 0 a 0,5 segundo, quando o erro varia muito rápido, o sistema atua rapidamente, pois em 0,5 segundo a temperatura já é de 79ºC e no caso do controlador P ou PI era de apenas 70ºC. Mas a partir de 0,5 segundo em diante, o erro já passa a variar lentamente e a resposta do derivativo já não é mais adequada. Nesta parte, onde a variação do erro é lenta, o integrador responde melhor, além é claro de o integrador atuar sobre o erro em regime. (CONTROLE DE AUTOMAÇÃO INDUSTRIAL, 2012?).

Figura 11- Controlador PD aplicado a uma estufa



Fonte: (CONTROLE DE AUTOMAÇÃO INDUSTRIAL, 2012?, p. 10)

1.1.3.5 Controlador Proporcional + Integral + Derivativo PID

O controle proporcional associado ao integral e ao derivativo é o mais sofisticado tipo de controle utilizado em sistemas de malha fechada. (OLIVEIRA, 1999)

A proporcional elimina as oscilações, a integral elimina o desvio de *off-set*[2], enquanto a derivativa fornece ao sistema uma ação antecipada evitando previamente que o desvio se torne maior quando o processo se caracteriza por ter uma correção lenta comparada com a velocidade de desvio (por exemplo, alguns controles de temperatura). (OLIVEIRA, 1999)

[2] Desvio residual (histerese)

O controlador PID, parece ser a opção ideal para se trabalhar, entretanto, esta é a opção mais cara e a mais difícil de ajustar, pois agora temos três ganhos para ajustar (K, KI e KD). (CONTROLE DE AUTOMAÇÃO INDUSTRIAL, 2012?).

Duas formas principais podem representar esse tipo de controlador, a primeira das quais sendo expressa por:

Equação 2- Representação do Controlador PID

$$u(t) = Ae(t) + B\int_0^t e(t)dt + C\frac{de(t)}{dt} = A\left[e(t) + \frac{B}{A}\int_0^t e(t)dt + \frac{C}{A}\frac{de(t)}{dt}\right]$$

Por razão de homogeneidade, os coeficientes (B/A) e (C/A) são necessariamente do tipo (1/T1) e T2, onde T1 e T2 são constantes de tempo. Aplicando-se a transformada de Laplace obtém-se uma primeira forma para o controle PID, conforme mostra a **Erro! Fonte de referência não encontrada.**.

Equação 3- Primeira forma para o controle PID

$$C(s) = A\left[1 + \frac{1}{sT_1} + sT_2\right]$$

$$C(s) = \frac{A}{sT_1}\left[T_1T_2s^2 + T_1s + 1\right]$$

Em geral uma segunda forma é preferida, na qual se representa o controlador PID como resultado da colocação em série de um controlador PI seguido de um controlador PD. Fisicamente, no caso de um controlador analógico, isto corresponde a uma placa eletrônica (PI) cuja saída alimenta uma segunda placa (PD). Escreve-se então:

Equação 4- Segunda forma para o controle PID

$$C(s) = K\left(1 + \frac{1}{sT_I}\right)(1 + sT_D)$$

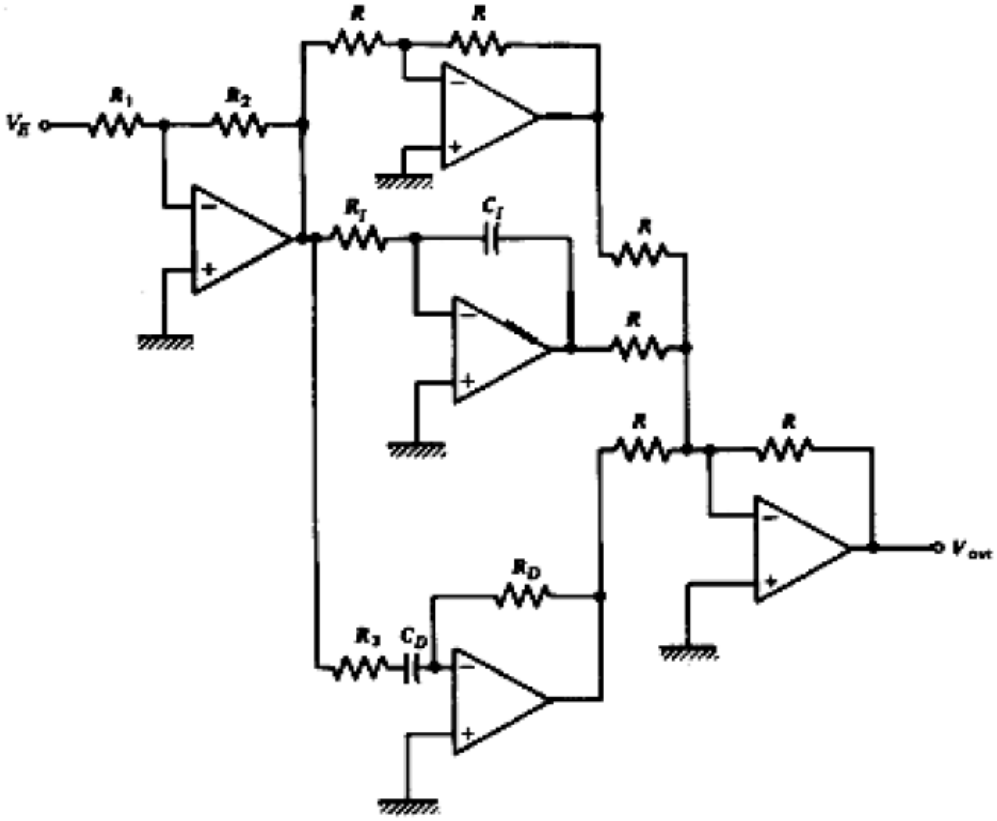
A comparação das equações anteriores fornece:

Equação 5- Comparação entre as equações

$$\frac{K}{T_I} = \frac{A}{T_1},\ T_I + T_D = T_1 \qquad \text{e} \qquad T_I\, T_D = T_1\, T_2$$

O controlador PID e as variantes (P, PI e PD) são implementáveis por meio de amplificadores operacionais. No caso são necessários três blocos: um para o proporcional, um para o derivativo e um para o integrador. Na Figura 12 é possível verificar o esquema completo do controlador PID.

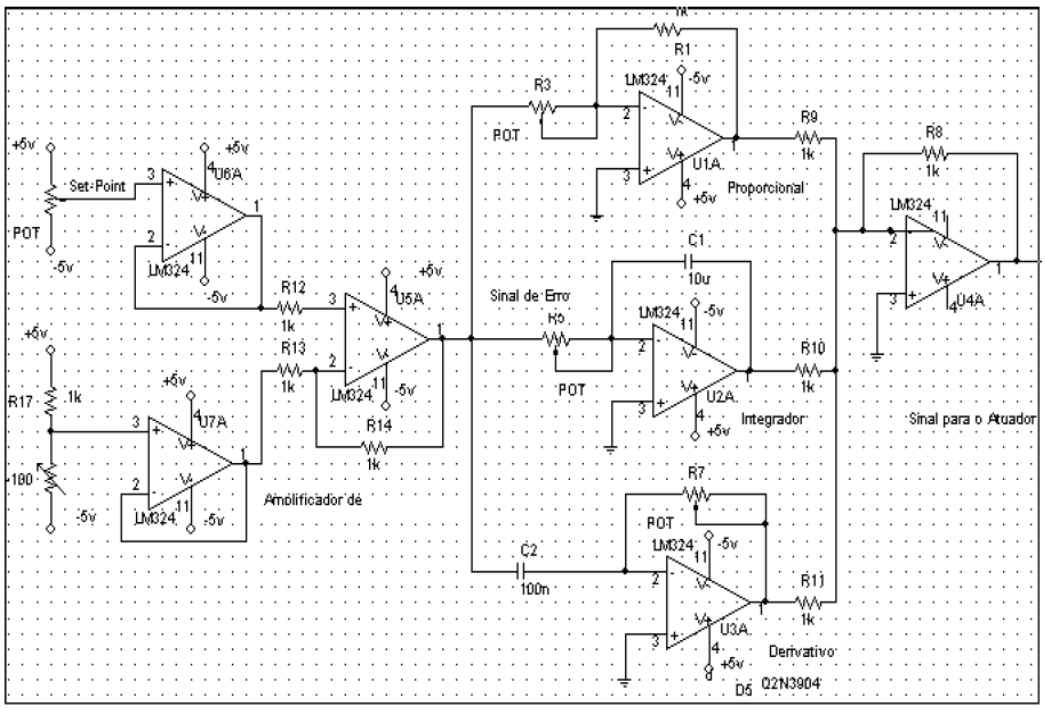
Figura 12- Bloco PID completo



Fonte: (CONTROLE DE AUTOMAÇÃO INDUSTRIAL, 2012?, p. 15)

A Figura 13 mostra o esquemático completo de um controlador de temperatura. É importante lembra que existem técnicas de controle mais sofisticados que o PID são chamado de controles modernos baseados em espaços de estados. Há também o controle adaptativo, controle robusto, redes neurais artificiais e outras técnicas, porém são usadas apenas em casos especiais por enquanto.

Figura 13- Esquemático completo de um controlador de temperatura



Fonte: (CONTROLE DE AUTOMAÇÃO INDUSTRIAL, 2012?, p. 18)

1.1.4 Dispositivos de entrada (sensores e transdutores)

São dispositivos utilizados para realizar o interfaceamento entre o sistema físico e o sistema de controle eletrônico, levando informações do campo para o controlador.

Podem ser classificados da seguinte forma:

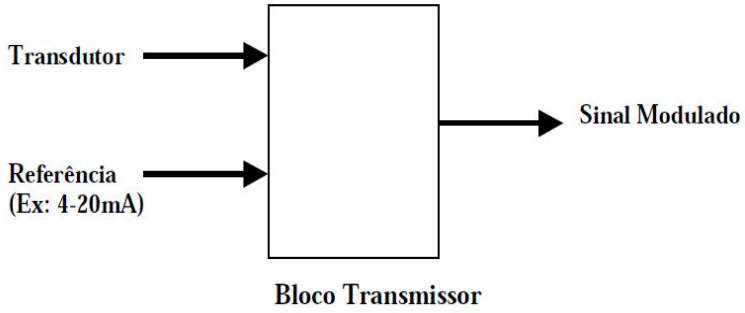
a) **Sensores**: Dispositivos projetados para detectarem algum evento no campo e emitirem um sinal em resposta a esse evento. Um exemplo é o sensor de proximidade, quando algum objeto atinge seu campo de visualização ele ativa um sinal em resposta a presença desse objeto.

b) **Transdutores**: Dispositivos que convertem uma grandeza física em outra. No caso de controle PID o interessante são os transdutores elétricos que convertem grandeza física (temperatura, pressão, etc.) em sinal elétrico (normalmente tensão). Podem ser de dois tipos: direto e indireto.

Os do tipo direto convertem a grandeza física em sinal elétrico diretamente. É o caso dos termopares que convertem temperatura em tensão. Os do tipo indireto modificam algum parâmetro interno, como resistência por exemplo, de forma proporcional a grandeza física. É o caso das termoresistências que aumentam sua resistência com o aumento da temperatura.

Entretanto, sensores, mas principalmente transdutores têm alcance limitado poucas dezenas de metros. Isto porque o comprimento do fio que liga o sensor ou transdutor, que possui alguma resistência e indutância, pode interferir no valor da medida. Alem disso pode captar ruídos e afetar a precisão da informação. Neste caso faz-se necessário um equipamento especifico para enviar informações a distancias maiores, que é o transmissor. Transmissor é um equipamento que recebe o sinal de um transdutor ou sensor e "modula" um sinal de referencia (4-20mA, 0-5V, etc.) de forma proporcional ao sinal do transdutor ou sensor. A Figura 14 mostra uma ilustração dessa modulação.

Figura 14- Bloco Transmissor



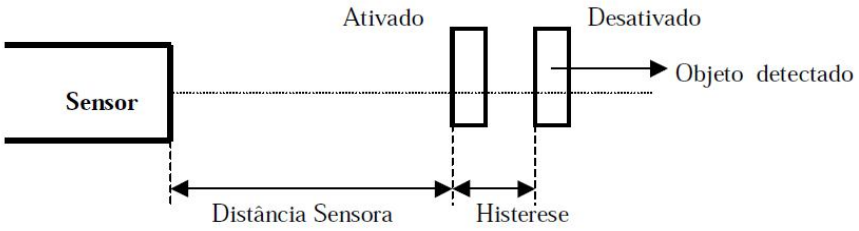
Fonte: (CONTROLE DE AUTOMAÇÃO INDUSTRIAL, 2012?, p. 22)

1.1.4.1 Parâmetros Fundamentais de Sensores

Os sensores são caracterizados por diversos parâmetros, mas alguns são usados com mais freqüência, pois precisam ser levados em consideração. Abaixo é possível analisar alguns destes.

a) **Distância Sensora**: É a distância perpendicular da face sensora até o ponto onde o sensor atua. Tipicamente é simbolizado pelo símbolo Sn.
b) **Histerese**: É a diferença entre a distância onde o sensor é ativado quando objeto se aproxima dele e a distância na qual o sensor é desativado quando o objeto se afasta dele. Normalmente é dada de forma percentual. A Figura 15 ilustra bem essas propriedades.

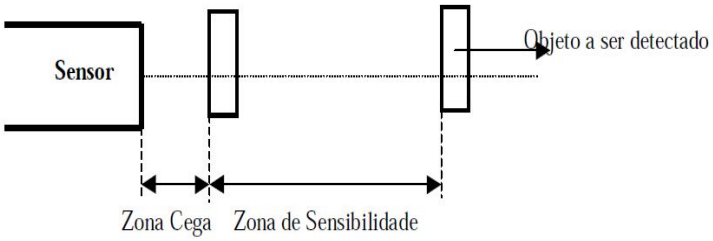
Figura 15- Histerese do sensor



Fonte: (CONTROLE DE AUTOMAÇÃO INDUSTRIAL, 2012?, p. 22)

c) **Zona Cega:** Região dentro da distância sensora, que o sensor por quesões tecnológicas ou de montagem, não consegue detectar o objeto. Não se trata de uma falha do sensor, mas sim de característica do mesmo que deve ser levada em conta.
d) **Zona de Sensibilidade:** Região da zona detectável, onde o dispositivo é efetivamente sensibilizado, conforme mostra a Figura 16.

Figura 16- Representação da zona de sensibilidade



Fonte: (CONTROLE DE AUTOMAÇÃO INDUSTRIAL, 2012?, p. 23)

e) **Repetibilidade:** Pequena variação na distância sensora quando se procede duas ou mais tentativas de detecção. Normalmente é indicada de forma percentual. Não deve ser confundida co ma histerese.
f) **Frequência de operação:** Número máximo de comutações por segundo que um sensor consegue realizar. É medida em Hertz.
g) **Corrente de Consumo:** Valor da corrente necessária ao funcionamento do sensor.
h) **Corrente de Carga:** É a máxima corrente possível na saída de um sensor.
i) **Corrente de Pico:** É o valor máximo de corrente consumido pelo sensor no momento da ativação.

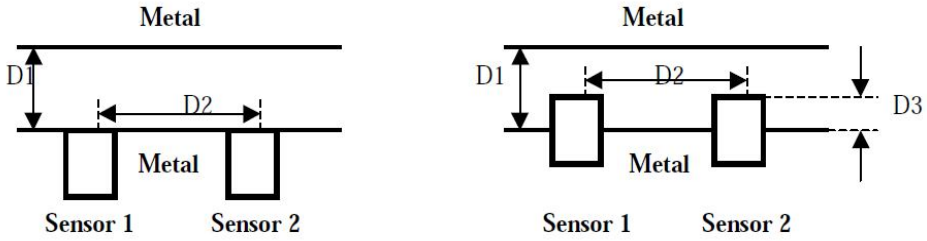
j) **Tensão de *Ripple*:** Máxima oscilação da tensão CC[3] de alimentação permitida.

k) **Tensão de Estabilização:** Tempo que se deve aguardar logo após a energização do sensor, para que as leituras sejam confiáveis.

l) **Proteção Intrínseca ou IP:** Grau de proteção do sensor a penetração de sólidos e líquidos. É indicado por dois dígitos (Ex. IP 66). O 1º refere-se aos sólidos e o 2º à líquidos. Deve-se consultar a tabela de graus de proteção para verificar o significado de cada código.

m) **Versão de Montagem:** Refere-se a forma como o sensor deve ser montado e as distâncias que devem ser respeitadas para assegurar o bom funcionamento do sensor. A Figura 17 ilustra isto.

Figura 17- Versão de montagem do sensor



Fonte: (CONTROLE DE AUTOMAÇÃO INDUSTRIAL, 2012?, p. 23)

1.1.4.1]2 Parâmetros Fundamentais de Transdutores

Alguns dos parâmetros utilizados pelos sensores se aplicam aos transdutores também como tensão de *ripple*, versão de montagem, etc. Mas existem características que são peculiares aos transdutores. Dentre eles estão:

a) **Linearidade**: Parâmetro importante, dado que uma conversão de grandezas, somente pode ser feita se houver uma relação linear entre a grandeza física e elétrica. Quando isto não ocorre, pode-se lançar mão de técnicas para fins de obtenção da linearidade.

b) **Região de Atuação**: Faixa de valores da grandeza que se deseja converter onde o dispositivo efetivamente deve trabalhar. Normalmente estão relacionados com a região onde vale a linearidade do transdutor, mas podem haver outros limitantes como integridade física do material, detalhes construtivos, entre outros.

[3] Corrente Contínua

c) **Fator de Proporcionalidade**: admitindo-se a linearidade do transdutor, a grandeza elétrica está relacionada com a grandeza física por um certo fator, chamado de fator de proporcionalidade. Exemplo: Um transdutor com 1mV/ºC de fator de proporcionalidade.

d) **Precisão e Exatidão**: Parâmetros relacionados ao erro de conversão de uma da grandeza. Influenciado por vários fatores, tais como condições ambientais, posicionamento, presença de ruído elétrico e outros.

Os transdutores podem ter saída analógica (termopares) ou digital (*encoder*). De qualquer forma, ruídos podem afetar a precisão de um transdutor, assim cuidados especiais devem ser tomados com estes dispositivos. Normalmente, os fabricantes sugerem medidas já consagradas para eliminação ou redução destes problemas. (CONTROLE DE AUTOMAÇÃO INDUSTRIAL, 2012?).

1.1.4.3 Sensor de Temperatura LM35

O LM35 é um sensor de precisão, fabricado pela *National Semiconductor*. A tensão de saída será linear e relativa à temperatura em que se encontra no momento em que for alimentado por uma tensão de 4-20Vdc e GND[4]. O valor da tensão de saída será de 10mV para cada grau *Celsius* de temperatura, sendo assim, apresenta uma boa vantagem com relação aos demais sensores de temperatura calibradis em *Kelvin*. Esse, não necessita de subtração de variáveis para obter uma escala de temperatura em graus *Celsius*. Outrossim, não necessita de calibração externa ou *trimming* para fornecer com exatidão, valores de temperatura com variações de ¼°C ou até mesmo ¾°C dentro da faixa de temperatura de -55 a 150°C. Apresenta na saída baixa impedância, tensão linear e calibração inerente precisa, fazendo com que a *interface* para a leitura seja especificamente simples, desta forma o sistema apresenta um menor custo.

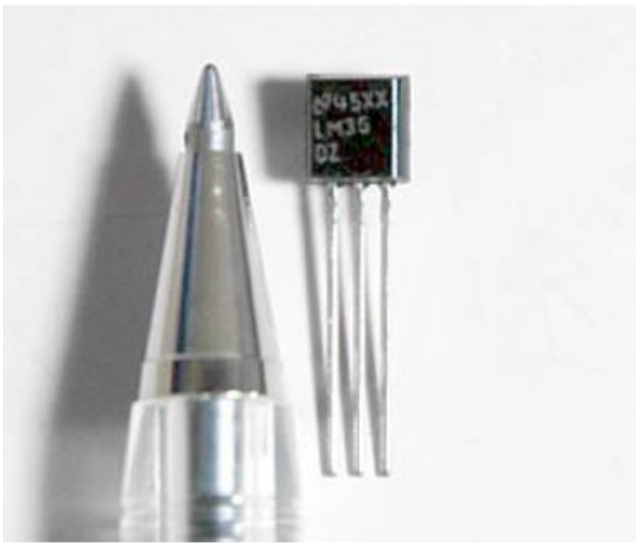
A alimentação do sensor poderá ser simples ou simples ou simétrica, dependendo do que se deseja como sinal de saída. Porém, independente disso, a saída continuará sendo de 10mV/°C. Ele drena apenas 60µA para a alimentação, sendo assim, seu auto-aquecimento é de aproximadamente 0,1°C ao ar livre. (BLUM, 2011?).

O sensor LM35 é apresentado com vários tipos de encapsulamentos, sendo o mais comum o T0-92, que mais se parece com um transistor de três pés, dois deles para

[4] GND: *Ground* ou Terra.

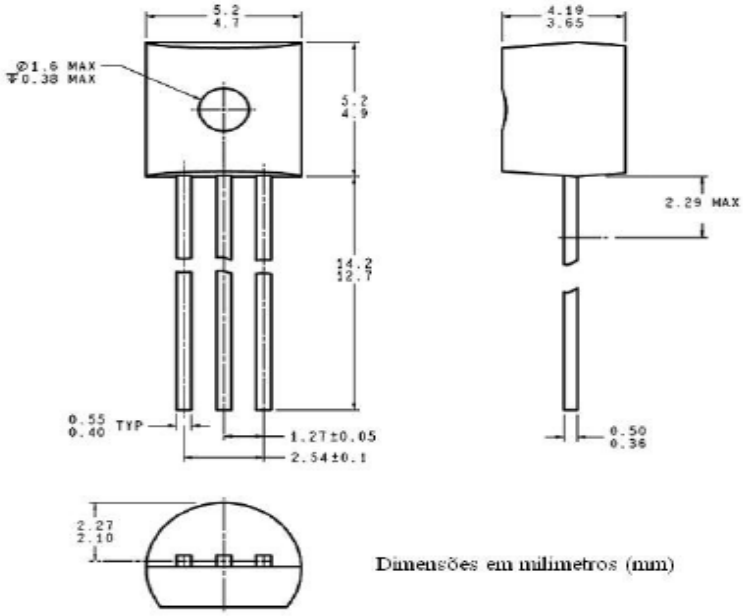
alimentação e o terceiro apresenta os valores de tensão proporcional à temperatura medida pelo sensor. A escala do sensor e suas medidas são apresentadas respectivamente na Figura 18 e Figura 19. (AQUISIÇÃO DE DADOS, 2009?)

Figura 18- Escala do sensor LM35



Fonte: (AQUISIÇÃO DE DADOS, 2009?, p. 2)

Figura 19- Dimensões do sensor LM35

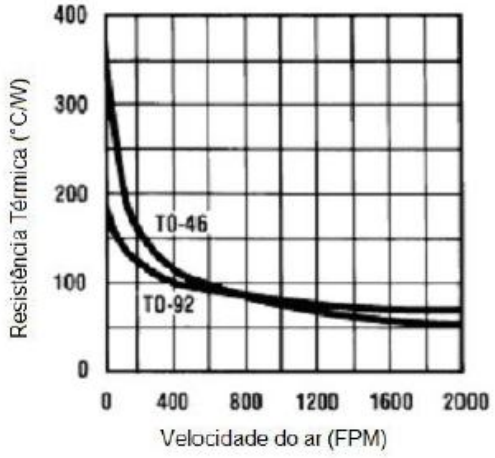


Fonte: (AQUISIÇÃO DE DADOS, 2009?, p. 2)

O sensor LM35 pode ser facilmente utilizado, da mesma forma que qualquer outro sensor de temperatura. Colando-o sobre a superfície que se desja medir a temperatura a sua temperatura estará em torno de 0,01°C abaixo da temperatura da superfície que se encontra fixado. Para isto, pressupõe que a temperatura da superfície seja a mesma que a temperatura do ar que se encontra ao redor deste ambiente. Se a tempertura do ar fosse muito mais elevada ou mais baixa do que a temperatura da superfície, a temperatura real do LM35 estaria em uma temperatura intermediária entre a temperatura da superfície e a temperatura do ar. Essa regra se aplica especialmente para o encapsulamento plástico do tipo T0-92, o qual as ligações de cobre são o trajeto térmico principal para carregar o calor através do dispositivo, fazendo com que a temperatura fique mais próxima da temperatura do ar do que da superficie em que se encontra fixado. Para amenizar este problema, deve-se ter certeza de que a fiaçãoconectada ao LM35 esteja presa juntamente à superfície de interesse, para que ambas as partes estejam prarticamente sempre na mesma temperatura. A maneira mais fácil de fazer isto é fixar os fios e o próprio LM35 com um leve revestimento de cola epóxi à superfície de interesse, assim, o LM35 e seus condutores não estarão em contato com o ar, e assim, a temperatura do ar não afetará na medição do componente.

No Gráfico 1, é se observa a relação entre a velocidade do ar e a resistência térmica. A resistência térmica junção-ar é a medida da habilidade que um dispositivo tem em dissipar calor da superfície de interesse ao meio externo por todos os trajetos possíveis. Isto é relevante para encapsulamentos que não utilizam dissipadores acoplados externamente. (BLUM, 2011?).

Gráfico 1- Relação entre velocidade do ar e resistência térmica



Fonte: (BLUM, 2011?, p. 3)

1.2 MICROCONTROLADORES

Atualmente um grande número de microcontroladores, integrados em diversos equipamentos, exercem um papel importante no dia a dia das pessoas. Despertar ao som de um *CD Player* programável, tomar café da manhã preparado por um microondas digital, e ir ao trabalho de carro, cuja injeção de combustível é microcontrolada, são apenas alguns exemplos.

Muitos produtos disponíveis hoje em dia, simplismente não existiriam, ou não teriam as mesmas funcionalidades sem um microcontrolador. É o caso, por exemplo, de vários instrumentos biomédicos, instrumentos de navegação por satélites, detetores de radar, equipamentos de áuido e vídeo, eletrodomésticos, entre outros.

Entretanto, o alcance dos microcontroladores vai além de oferecer algumas facilidades. Uma aplicação crucial, onde os microcontroladores são utilizados, é na redução de consumo de recursos naturais. Existem sistemas de aquecimento modernos que captam a luz solar e, de acordo com a demanda dos usuários, controlam a temperatura de forma a minimizar perdas. Outro exemplo, de maior impacto, é o uso de microcontroladores na redução do consumo de energia em motores elétricos, que são responsáveis pelo consumo de, aproximadamente 50% de toda eletricidade produzida no planeta. Portanto, o alcance dessa tecnologia tem influência muito mais importante do que se possa imaginar.

O universo de aplicação dos microcontroladores está em grande expansão, sendo que a maior parcela dessas aplicações é em sistemas embarcados. A expressão “sistemas embarcados” (do inglês *embedded system*) se refere ao fato do microcontrolador se inserido nas aplicações (produtos) e usado de forma exclusiva por elas. Como a complexidade desses sistemas cresce vertiginosamente, o *software* tem sido fundamental para oferecer as respostas às necessidades desse mercado. Tanto é, que o *software* para microcontroladores representa uma fatia considerável do mercado de *software* mundial. Segundo Edward Yourdon a proliferação dos sistemas embarcados, juntamente com o advento da Microsoft, são os responsáveis pela retomada do crescimento da indústria de *software* nos Estados Unidos da América.(DENARDIN, 2000?)

1.2.1 Definição de Microcontrolador

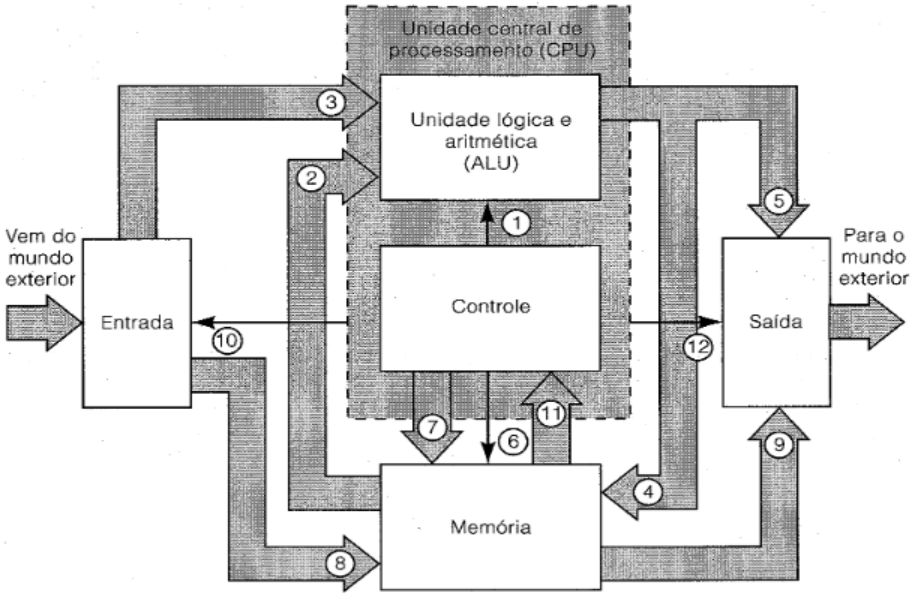
Um microcontrolador é um sistema computacional completo, no qual estão incluídos uma CPU (*Central Processor unit*), memória de dados e programa, um sistema de clock[5], portas de I/O[6] (*Input/Output*), além de outros possíveis periféricos, tais como, módulos de temporização e conversores A/D entre outros, integrados em um mesmo componente. As partes integrantes de qualquer computador, e que também estão presentes, em menor escala, nos microcontroladores são:

1.2.1.1 Unidade Central de Processamento (CPU)

A unidade central de processamento é composta por uma unidade lógica aritmética (ULA) , por uma unidade de controle e por unidades de memórias especiais conhecidas por registradores. Para que a CPU possa realizar tarefas é necessário que se agregue outros componentes, como unidades de memória, unidades de entrada e unidades de saída. A

Figura 20 apresenta um diagrama de blocos com uma possível interface entre a CPU e outros dispositivos.

Figura 20- Interface entre CPU e outros dispositivos



Fonte: (DENARDIN, 2000?, p. 3)

[5] *Clock:* Significa relógio.
[6] I/O: Entradas/Saídas físicas.

A unidade de memória permite armazenar grupos de dígitos binários que podem representar instruções que o processador irá executar ou dados que serão manipulados pelo processador. A unidade de entrada consiste em todos os dispositivos utilizados para obter informações e dados externos ao processador. A unidade de saída consiste em dispositivos capazes de transferir dados e informações do processador para o exterior.

A ULA é a área de uma CPU na qual as operações lógicas e aritméticas são realizadas sobre os dados. O tipo de operação realizada é determinada pelos sinais da unidade de controle. Os dados a serem operados pela ULA podem ser oriundos de uma memória ou de uma unidade de entrada. Os resultados das operações realizadas na ULA podem ser transferidos tanto para uma memória de dados como para uma unidade de saída.

A função da unidade de controle é comandar as operações da ULA e de todas as outras unidades conectadas a CPU, fornecendo sinais de controle e temporização. De certa maneira, a unidade de controle é como um maestro que é responsável por manter cada um dos membros da orquesta em sincronismo. Essa unidade contém circuitos lógicos e de temporização que geram os sinais apropriados necessários para eecutar cada instrução de um programa.

A unidade de controle busca uma instrução na memória enviando um endereço e um comando de leitura para a unidade de memória. A palavra da instrução armazenada na posição de memória é transferida para um registrador conhecido por registrador de instruções (RI) da unidade de controle. Essa palavra de instrução, que está de alguma forma de código binário, é então decodificada pelos circuitos lógicos na unidade de controle para determinar a instrução que está sendo invocada. A unidade de controle usa essa informação para os sinais apropriados para as unidades restantes a fim de executar a operação específica.

Essa sequência de busca de um código de instrução e de execução da operação indicada é repetida indefinidamente pela unidade de controle. Essa sequência repetitiva de busca/execução continua até que a CPU seja desligada ou até que o *RESET*[7] seja ativado. O *RESET* sempre faz a CPU buscar sua primeira instrução no programa.

Uma CPU, também conhecida por processador, repete indefinidamente as mesmas operações básicas de busca e execução. Naturalmente, os diversos ciclos de execução serão diferentes para cada tipo de instrução à medida que a unidade de controle envia sinais diferentes para as outras unidades de execução de uma instrução em particular.

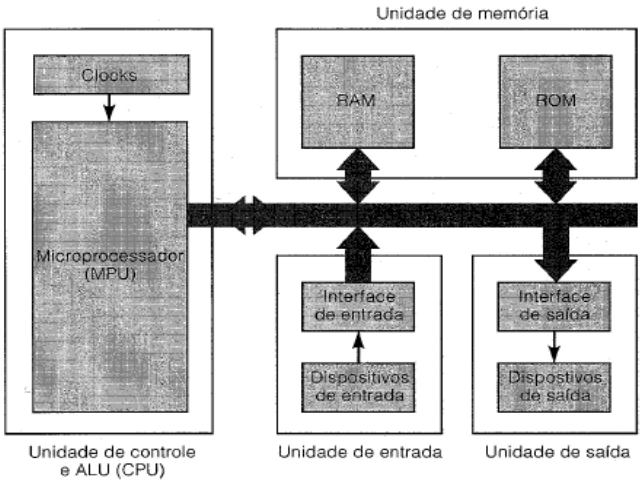
[7] *Reset*: Restabelece a configuração ou condição inicial.

Um registrador é um tipo de memória de pequena capacidade, porém muito rápida, contida na CPU, utilizado no armazenamento temporário de dados durante o processamento. Os registradores estão no topo da hierarquia de memória, sendo desta forma o meio mais rápido e de maior custo para armazenar um dado.

Cada registrador de um processador possui uma função especial. Um dos mais importantes é o contador de programas (*program counter* – PC), que armazena os endereços dos códigos das instruções à medida que são buscadas da memória. Outros registradores são utilizados para realizar funções como: armazenamento de códigos de instrução (RI), manutenção dos dados operados pela ULA (acumulador), armazenamento de endereços de dados a serem lidos na memória (ponteiro de dados), além de outras funções de armazenamento e contagem. Todos os processadores possuem um registrador em especial muito utilizado chamado de acumulador ou registrador A. Ele armazena um operando para quaisquer instruções, lógicas ou matemática. O resultado da operação é armazenado no acumulador após a instrução ser executada.

Para que exista comunicação entre as unidades que formam um processador é necessário definir uma forma de conexão entre estas unidades. Em um processador tradicional com arquitetura *Von Neuman* este meio é o barramento de dados. A largura do barramento de dados em *bits* é o que determina o número de *bits* para um dado processador. A Figura 21 apresenta a interface dos principais dispositivos que compõem um sistema microprocessado através de um barramento de dados.

Figura 21- Principais dispositivos que compõem um sistema microprocessado



Fonte: (DENARDIN, 2000?, p. 5)

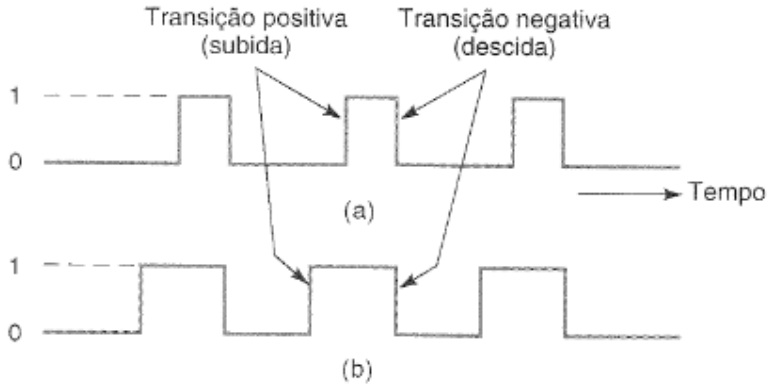
A CPU é o centro de todo sistema computacional, e não é diferente quando se trata de microcontroladores. O trabalho da CPU é executar rigorosamente as instruções de um programa, na sequência programada, para uma aplicação específica. Um programa computacional (*software*) instrui a CPU a ler informações de entradas, ler e escrever informações na memória de dados, e escrever informações nas saídas.

1.2.1.2 Sistema de C*lock*

O sistema computacional utiliza um *clock* para fornecer a CPU uma maneira de se mover de instrução em instrução, em uma sequência pré-determinada. (DENARDIN, 2000?)

O sinal de *clock* determina os momentos nos quais qualquer uma das saídas pode mudar de estado. Este sinal de *clock* é geralmente um trem de pulsos retangulares, ou uma onda quadrada como pode ser visto na Figura 22. O sinal de *clock* é distribuído para todas as partes do sistema, e a maioria das saídas (senão todas) do sistema pode mudar de estado somente quando o *clock* faz uma transição. As transições (também chamadas de bordas) estão indicadas na Figura 22. Quando o *clock* faz uma transição de 0 para 1, esta é chamada de transição positiva (subida). Quando o *clock* faz uma transição de 1 para 0, esta é chamada de transição negativa (descida). (RONALD, 2007)

Figura 22- Sinais de *clock*



Fonte: (RONALD, 2007, p. 407)

Uma fonte de *clock* de alta frequência (normalmente derivada de um cristal ressonador conectado a CPU) é utilizada para controlar o sequenciamento das instruções da CPU. Normalmentes as CPUs dividem a frequência básica do cristal por dois ou mais para

chegar ao *clock* do barramento interno. Cada ciclo de leitura ou escrita a memória é executado em um ciclo de *clock* do barramento interno, também denominado ciclo de barramento. (DENARDIN, 2000?).

1.2.1.3 Memória

Pode-se pensar na memória como sendo uma lista de endereços postais, onde o conteúdo de cada endereço é um valor fixo de 8 bits (para CPU de 8 bits). Se um sistema computacional tem n linhas (bits) de endereços, ele pode endereçar 2^n (lê-se dois elevado a n) posições de memória. Entre os diversos tipos de memória encontram-se:

a) **RAM** (*Random Access Memory*): Memória de acesso aleatório. Pode ser lida ou escrita pela execução de instruções da CPU e, normalmente é utilizada para manipulação de dados pela CPU. O conteúdo é perdidona ausência de energia (memória volátil).
b) **ROM** (Read Only Memory): Memória apenas de leitura. Pode ser lida, mas não é alterável. O conteúdo deve ser determinado antes que o circuito integrado seja fabricado. O conteúdo é mantido na ausência de energia. (memória não volátil).
c) **EPROM** (*Erasable and programmable Read-Only Memory*): Memória ROM programável e apagável. O conteúdo dessa memória pode ser apagado com luz ultravioleta, e posteriormente, reprogramado com novos valores. As operações de apagamento e programação podem ser realizadas um número limitado de vezes depois que o circuito integrado for fabricado. Da mesma forma que a ROM, o conteúdo é mantido na ausência de energia (memória não volátil).
d) **OTP** (*One Time Programmable*): Memória programável uma única vez. Semelhante a EPROM quanto à programação, mas que não pode ser apagada.
e) **EEPROM (***Electrically Erasable and Programmable Read-Only Memory*): Memória ROM programável e apagável eletricamente. Pode ter seu conteúdo alterado através da utilização de sinais elétricos convenientes. Tipicamente, um endereço de uma EEPROM pode ser apagada e reprogramada até 100.000 vezes.

f) **FLASH:** Memória funcionalmente semelhante a EEPROM, porém com ciclos de escrita bem mais rápidos.
g) **I/O** (*Input/Output*): Registradores de controle, estado e sinais de I/O são um tipo especial de memória porque a informação pode ser sentida (lida) e/ou alterada (escrita) por dispositivos diferentes da CPU.

1.2.1.4 Sinais de Entrada

Dispositivos de entrada fornecem informação para a CPU processar, vindas do mundo externo. A maioria das entradas que os microcontroladores processam são denominadas sinais de entradas digitais, e utilizam níveis de tensão compatíveis com a fonte de alimentação do sistema. O sinal de 0V (GND ou VSS) indica o nível lógico 0 e o sinal de fonte positiva, que tipicamente é +5VCC (VDD) indica o nível lógico 1 (atualmente os microcontroladores começaram a reduzir a tensão de VDD para valores na faixa de 3V).

Naturalmente que no mundo real existem sinais puramente analógicos (com uma infinidade de valores) ou sinais que utilizam outros níveis de tensão. Alguns dispositivos de entrada traduzem as tensões do sinal para níveis compatíveis com VDD e VSS. Outros dispositivos de entrada convertem os sinais analógicos em sinais digitais (valores binários formados por 0s e 1s) que a CPU pode entender e manipular. Alguns microcontroladores incluem circuitos conversores analógicos/digitais (ADC) encapsulados no mesmo componente.

Sinais de Saída

Dispositivos de saída são usados para informar ou agir no mundo exterior através do processamento de informações realizados pela CPU. Circuitos eletrônicos (algumas vezes construídos no próprio microcontrolador) podem converter sinais digitais em níveis de tensão analógicos. Se necessário, outros circuitos podem alterar os níveis de tensão VDD e VSS nativos da CPU em outros níveis.

Códigos de Operação (*opcodes*)

Os programas usam códigos para fornecer instruções para a CPU. Estes códigos são chamados de códigos de operação ou *opcodes*. Cada *opcode* instrui a CPU a executar uma sequência específica para realizar sua operação. Microcontroladores de diferentes fabricantes

usam diferentes conjuntos de *opcodes* porque são implementados internamente por *hardware* na lógica da CPU. O conjunto de instruções de uma CPU especifica todas as operações que podem ser realizadas. *Opcodes* são uma representação das instruções que são entendidas pela máquina, isto é, uma codificação em representação binária a ser utilizada pela CPU.

1.2.1.5 Mnemônicos das instruções e assembler

Um *opcode* como 0x4C é entendido pela CPU, mas não é significativo para os humanos. Para resolver esse problema, um sistema de instruções mnemônicas equivalentes foi criado (linguagem assembly). O *opcode* 0x4C corresponde ao mnemônico INCA, lê-se "incrementa o acumulador", que é muito mais inteligível. Para realizar a tradução de mnemônicos em códigos de máquina (*opcodes* e outras informações) utilizados pela CPU é necessário um programa computacional chamado assembler (compilador para linguagem assembly). Um programador utiliza um conjunto de intruções na forma de mnemônicos para desenvolver uma determinada aplicação, e posteriormente, usa um assembler para traduzir essas instruções para *opcodes* que a CPU pode entender. (DENARDIN, 2000?).

1.2.2 O Microcontrolador PIC

PIC é o nome que a *Microchip* adotou para a sua família de microcontroladores, sendo que a sigla significa Controlador Integrado de Periféricos.

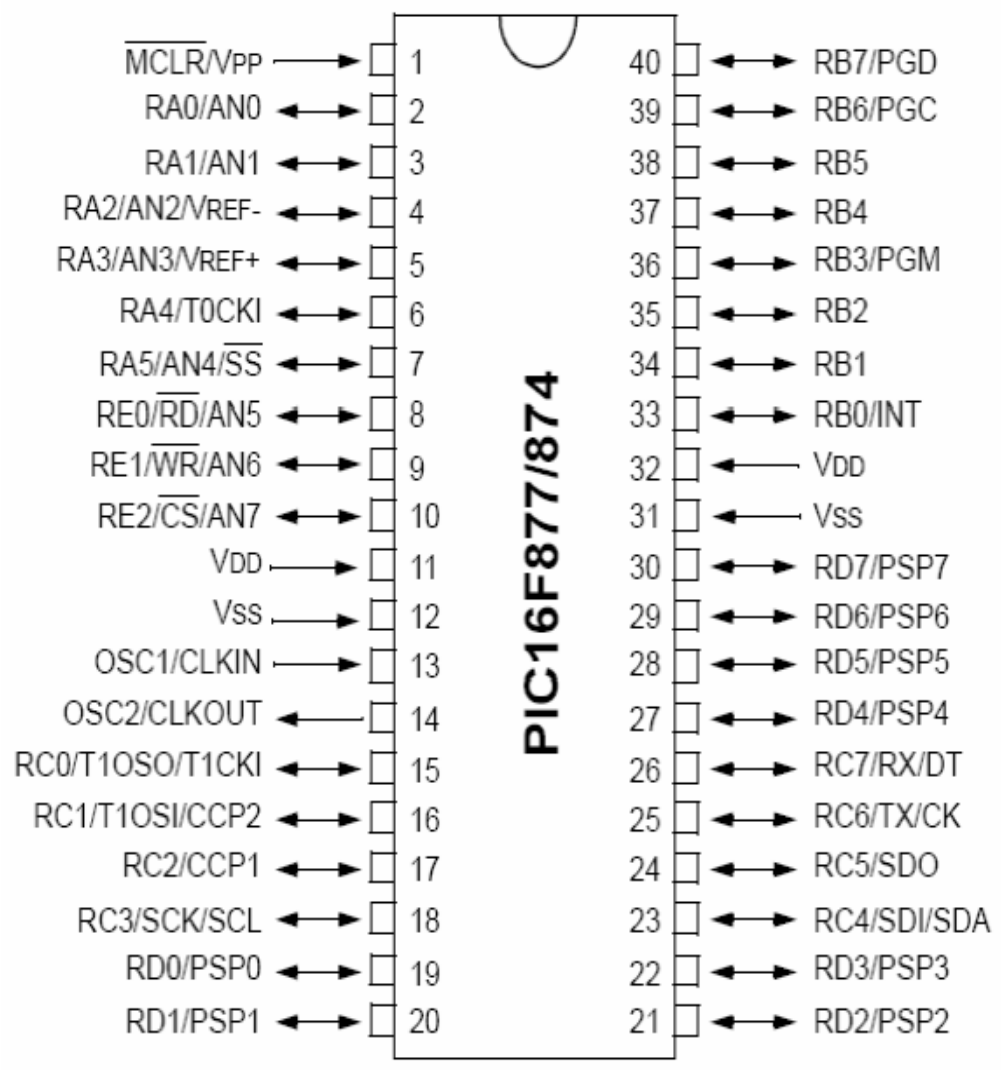
O PIC é um circuito integrado produzido pela *Microchip Technology Inc*., que pertence da categoria dos microcontroladores, ou seja, um componente integrado que em um único dispositivo contém todos os circuitos necessários para realizar um completosistema digital programável. (ANTÔNIO, 2006)

Internamente dispõe de todos os dispositivos típicos de um sistema microprocessado, ou seja, possui uma CPU, uma memória PROM, uma memória RAM, uma série de linhas de I/O para controlar dispositivos externos ou receber pulsos (sensores, chaves, etc) e uma série de dispositivos auxiliares ao funcionamento como gerador de *clock*, *bus*, contador, etc.

A presença de todos estes dispositivos em um espaço extremamente pequeno, da ao projetista ampla gama de trabalho e enorme vantagem em usar um sistema microprocessado, onde em pouco tempo e com poucos componentes externos é possível realizar o que seria oneroso fazer com circuitos tradicionais.

O PIC está disponível em uma ampla gama de modelos para melhor adaptar-se as exigências de projetos específicos, diferenciando-se pelo número de linha de I/O e pelo conteúdo do dispositivo. Inicia-se com modelo pequeno identificado pela sigla PIC12CXX dotado de 8 pinos, até chegar a modelos maiores com sigla PIC17CXX dotados de 40 pinos. Uma descrição detalhada da tipologia do PIC é disponível no site da *microchip*, onde é possível encontrar grandes e variadas quantidades de informações técnicas software de apoio, exemplos de aplicações e atualizações disponíveis. A Figura 23 mostra o modelo PIC16F877.

Figura 23- Microcontrolador PIC16F877



Fonte: (ANTÔNIO, 2006, p.4)

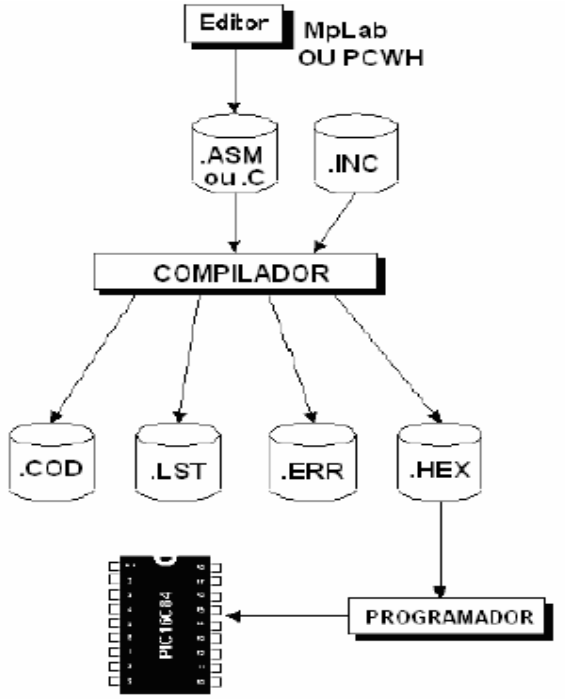
1.2.2.1 Programação do PIC

Como o PIC é um dispositivo programável, o programa tem como objetivo deixar instruções para que o PIC possa fazer atividades definidas pelo programador. Um programa é constituído por um conjunto de instruções em sequência, onde cada uma identificará precisamente a função básica que o PIC irá executar.

Um programa em linguagem *assembler* ou em C pode ser escrito em qualquer PC utilizando-se qualque processador de textos que possa gerar arquivos ASCII (*word, Notpad*, etc). Um arquivo de texto que contenha um programa em *assembler* é denominado de *source* ou código *assembler*.

Uma vez preparado o código *assembler* ou C, será preciso um programa para traduzir as instruções mnemônicas e todas as outras formas convencionais do código em uma série de números (o *opcode*) reconhecível diretamente pelo PIC. Este programa se chama compilador *assembler* ou assemblador. Na huhufhufhuf está esquematizado o fluxograma de operações e arquivos que deverá ser realizado para passar um código *assembler* a um PIC a ser programado. (ANTÔNIO, 2006)

Figura 24- Fluxograma de compilação de um programa e gravação de um PIC

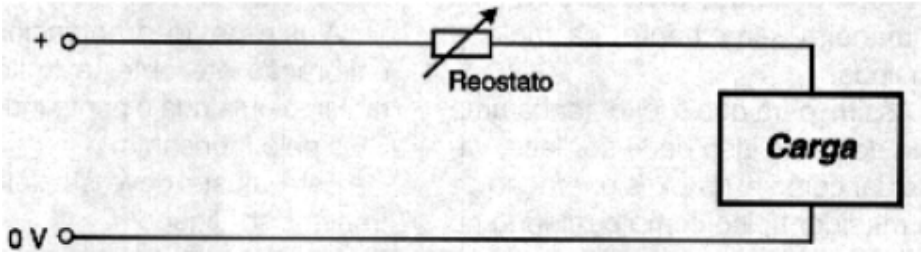


Fonte: (ANTÔNIO, 2006, p.5)

1.3 CONTROLE DE POTÊNCIA

Os controles de potência, inversores de frequência, conversores para servomotor, fontes chaveadas e muitos outros circuitos utilizam a tecnologia do PWM (*pulse widht modulation*) ou modulação de largura de pulso como base de seu funcionamento. A maneira tradicional, ou mais simples de se controlar uma carga de potência é através de um reostato em série, conforme mostra a Figura 25. (GHIRARDELLO, 2007?)

Figura 25- Controle Tradicional (linear) de potência



Fonte: (GHIRARDELLO, 2007?, p. 1)

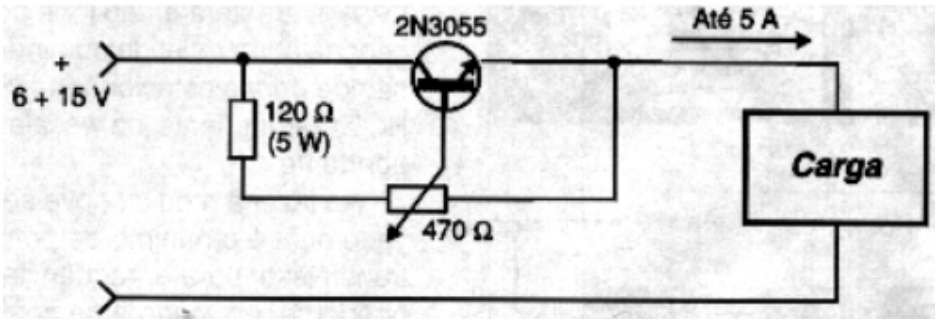
Variando-se a resistência apresentada pelo reostato é possível modificar a corrente na carga e, portanto, a potência aplicada a ela. Este tipo de controle ainda é encontrado nas lâmpadas de painéis de alguns carros mais antigos.

A grande desvantagem desse tipo de controle, denominado “linear”, é que a queda de tensão no reostato multiplicada pela corrente que ele controla representa uma grande quantidade de calor gerada.

O controle passa a dissipar (e pedir) mais potência que a aplicada na própria carga em determinadas posições do ajuste. Além de esta perda ser inadmissível, ela faz com que o componente usado no controle seja capaz de dissipar elevadas potências, ou seja, torna-se caro e grande (normalmente reostatos ou potenciômetros de fio, mesmo para potências relativamente baixas).

O uso de transistores e ou circuitos integrados em um controle mais elaborado, que pode variar linearmente a potência aplicada pelo controle direto da corrente, pode ser feito conforme ilustra a Figura 26.

Figura 26- Reostato Eletrônico Usando Transistor de Potência



Fonte: (GHIRARDELLO, 2007?, p. 1)

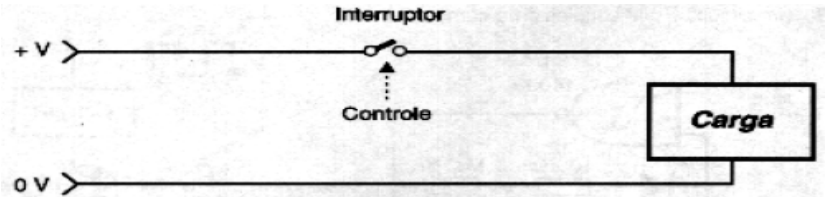
Embora o potenciômetro usado no controle dissipe pequena potência, pois a corrente nele é menor, este tipo de controle ainda tem um problema: a potência dissipada pelo dispositivo que controla a corrente principal é elevada. Esta potência depende da corrente e da queda de tensão no dispositivo e, da mesma forma, em certas posições do ajuste, pode ser maior que a própria potência aplicada ao dispositivo.

Na eletrônica moderna, o rendimento com pequenas perdas e a ausência de grandes dissipadores que ocupem espaço é fundamental, principalmente quando circuitos de alta potência estão sendo controlados. Desta forma, este tipo de controle de potência linear não é conveniente, sendo requisitadas outras configurações de maior rendimento como as que fazem uso da tecnologia PWM. (GHIRARDELLO, 2007?)

1.3.1 Tecnologia PWM

PWM é a abreviação de *pulse widht modulation* ou modulação de largura de pulso. Para que se entenda como funciona esta tecnologia no controle de potência, é observado na Figura 27 um circuito imaginário formado por um interruptor de ação muito rápida e uma carga que deve ser controlada. (GHIRARDELLO, 2007?)

Figura 27- Controle de carga por interruptor

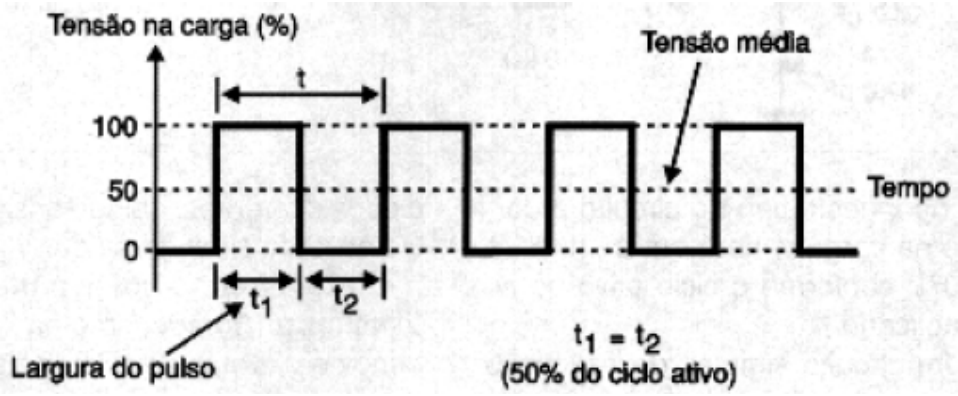


Fonte: (GHIRARDELLO, 2007?, p. 2)

Quando o interruptor está aberto não há corrente na carga e a potência aplicada é nula. No instante em que o interruptor é fechado, a carga recebe a tensão total da fonte e a potência aplicada é máxima.

Para obter uma tensão intermediária, ou seja, 50% aplicada a carga, uma idéia é fazer com que a chave seja aberta e fechada rapidamente de modo a ficar 50% do tempo aberta e 50% fechada. Isso significa que, em média, tem-se metade do tempo com corrente e metade do tempo sem corrente, conforme mostra a Figura 28.

Figura 28- Variação da tensão

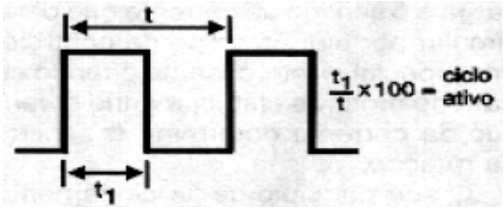


Fonte: (GHIRARDELLO, 2007?, p. 2)

A potência média é, portanto, a própria tensão média aplicada à carga, neste caso 50% da tensão de entrada. O interruptor fechado pode definir uma largura de pulso pelo tempo em que ele fica nesta condição, e um intervalo entre pulsos pelo tempo em que ele fica aberto. Os dois tempos juntos definem o período e, portanto, uma frequência de controle.

A relação entre o tempo em que se tem o pulso e a duração de um ciclo completo de operação do interruptor define o ciclo ativo, conforme é mostrado na Figura 29.

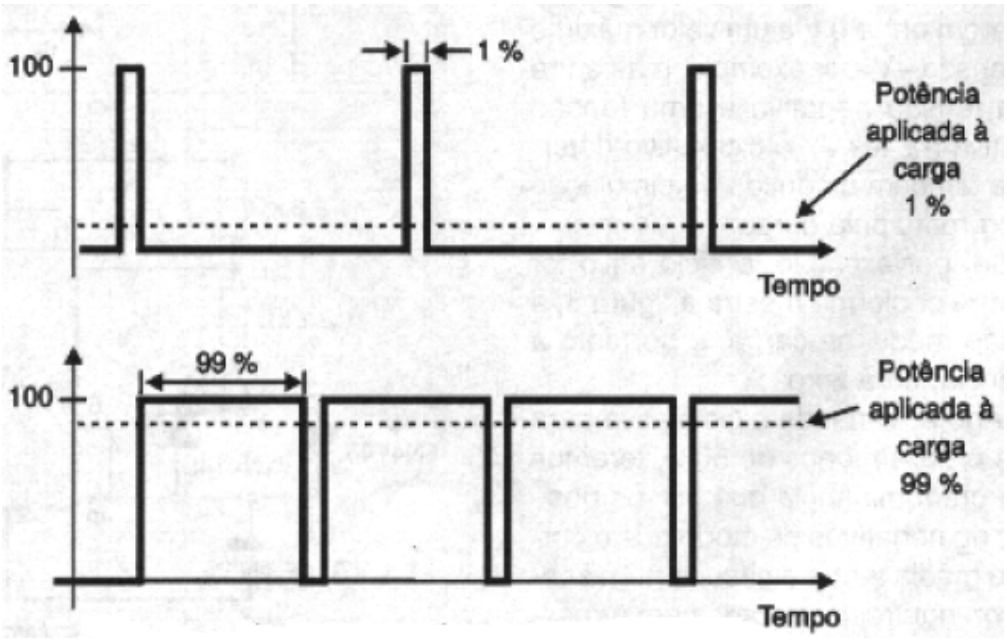
Figura 29- Definindo o ciclo ativo



Fonte: (GHIRARDELLO, 2007?, p. 2)

Variando-se a largura do pulso e também o intervalo de modo a obter ciclos ativos diferentes, é possível controlar a potência média aplicada a uma carga. Assim quando a largura do pulso varia de zero até o máximo, a potência também varia na mesma proporação, conforme está indicado na Figura 30.

Figura 30- Controlando a potência pelo ciclo ativo



Fonte: (GHIRARDELLO, 2007?, p. 3)

Este principio é usado justamente no controle PWM: Modula-se (varia-se) a largura do pulso de modo a controlar o ciclo ativo do sinal aplicado a uma carga e, com isso, a potência aplicada a ela. (GHIRARDELLO, 2007?).

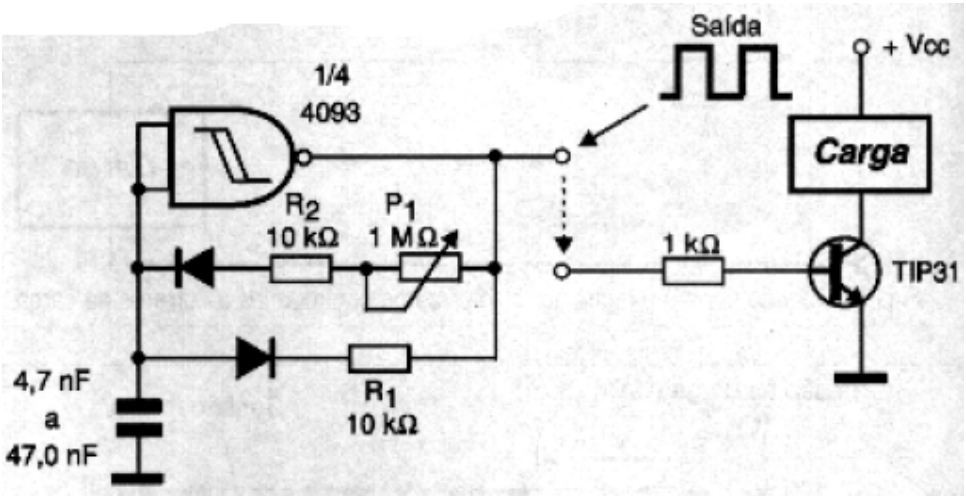
1.3.1.1 Pwm na Prática

Na pratica, o interruptor é substituido por algum dispositivo de estado sólido que possa abrir e fechar o circuito rapidamente como, por exemplo, um transistor bipolar, um FET[8] de potência, um IGBT[9] ou até mesmo um SCR[10].

[8] FET: Transistor de Efeito de Campo
[9] IGBT: Transistor Bipolar de Porta Isolada
[10] SCR: Retificador Controlado de Silício

A este dispositivo é então ligado um oscilador que possa ter seu ciclo ativo controlado numa grande faixa de valores. Na prática, é difícil chegar a duração zero do pulso e à 100%, já que isso implicaria na parada do oscilador, porém é possivel chegar bem próximo disso. Na Figura 31 tem-se o exemlo de um circuito que pode ser usado num controle PWM simples para um motor DC de pequena potência, com corrente de até 2 *amperes*.

Figura 31- Um circuito PWMsimples para carga de até 2A



Fonte: (GHIRARDELLO, 2007?, p. 3)

O oscilador, montado com um circuito integrado 4093 tem sua saída no nível alto determinada pelo ajuste do potenciômetro, enquanto que sua saída no nível baixo é determinada pelo resistor R1 (fixo). Assim, fazendo R1 suficientemente pequeno em relação ao valor do potenciômetro, o circuito poderá gerar sinais numa ampla faixa de ciclos ativos. Estes sinais são então aplicados ao transistor de potência que comanda a carga.

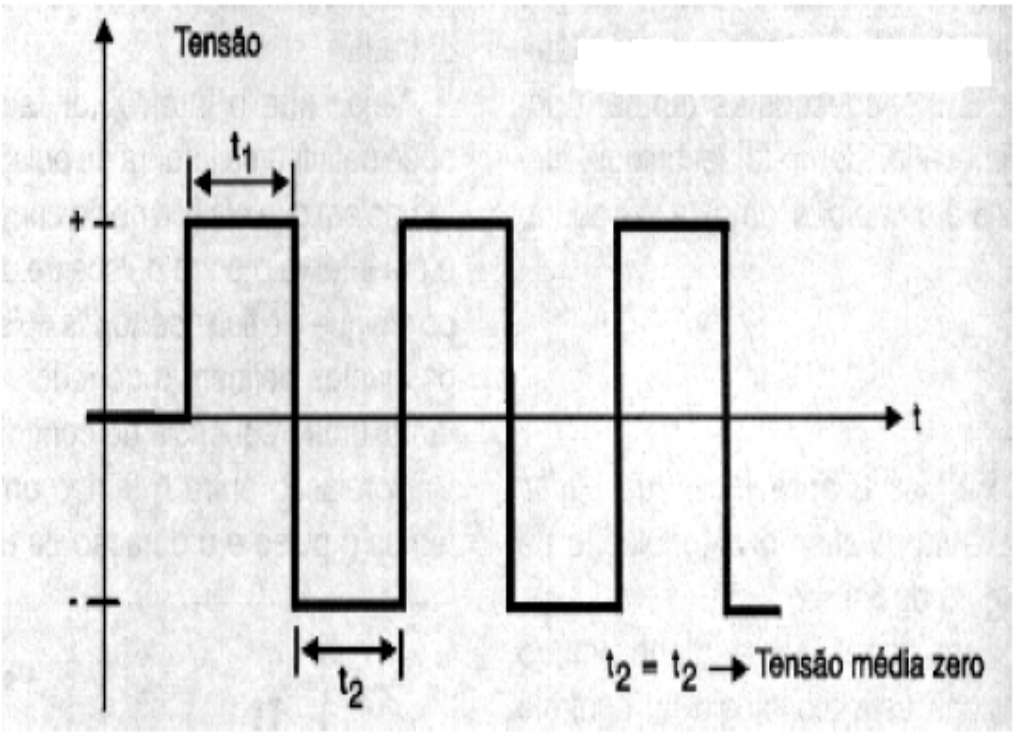
1.3.1.2 Tipos de Pwm

O exemplo mostrado na Figura 31 é o que se denomina de "*simple magnitude PWM"*, onde o sinal aplicado à carga determina simplesmente a potência que ela deve receber, pela largura do pulso. No entanto, existe um segundo tipo de controle PWM denominado "*Locked anti-phase PWM"*, que pode incluir na modulação do sinal informações sobre a potência aplicada à carga e o sentido da corrente que deve circular por ela. Este tipo

de controle, em especial, é interessante quando se trata de motores elétricos onde o sentido da corrente determina o sentido da rotação ou do torque.

Se os pulsos aplicados à carga não variarem entre 0V e um valor máximo de tensão +V, por exemplo, mas em uma tensão negativa e uma tensão positiva (-V a +V), o ciclo ativo determina também o sentido de circulação da corrente pela carga. Se, por exemplo, o ciclo ativo for de 50% conforme mostra a Figura 32, a tensão média na carga, e portanto a potência será zero.

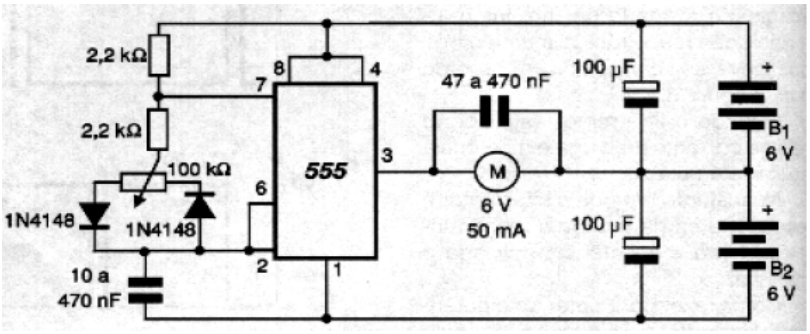
Figura 32- O *"Locked Anti-phase PWM"*



Fonte: (GHIRARDELLO, 2007?, p. 4)

Agora, variando o ciclo ativo para mais e para menos de 50%, tem-se uma predominância dos pulsos positivos ou negativos de modo que a corrente média tende a circular num sentido ou noutro. Desta forma, neste tipo de circuito, a corrente na carga variará entre -100% e +100%, conforme o ciclo ativo do sinal aplicado. Um circuito simples de aplicação para este tipo de controle é fornecido na Figura 33.

Figura 33- Circuito Prático de PWM anti-fase



Fonte: (GHIRARDELLO, 2007?, p. 4)

Uma fonte simétrica de +6v/-6v controla um perquno motor de 50mA a partir de um integrado LM555. Uma etapa de potência com transistores pode ser acrescentada a este circuito, para o uso com motores de maior corrente.

O potênciometro ajusta tanto a largura como os intervalos entre os pulsos de modo que a carga e descarga do capacitor sejam derivadas por diodos diferentes, agindo assim no ciclo ativo do sinal de saída.

Um ponto importante que deve ser observado nesse tipo de circuito é que na posição de 50% de ajuste do potênciometro (potência média nula na carga), na verdade tem-se uma corrente circulando o tempo todo por ela, o que vai causar dissipação de calor.

Assim sendo, para cargas elevadas, este tipo de controle não é dos mais indicados e não funcionaria, por exemplo, se a carga controlada fosse justamente um elemento de aquecimento ou uma lâmpada.

Mesmo no caso de motores DC é preciso ter muito cuidado na escolha da frequência de operação do circuito, para que na condição de parado (0% de potência) ele não se mantenha vibrando na frequência do oscilador. Eventualmente componentes adicionais podem ser previstos em paralelo com o motor, como por o exemplo, um capacitor, para evitar esse problema.

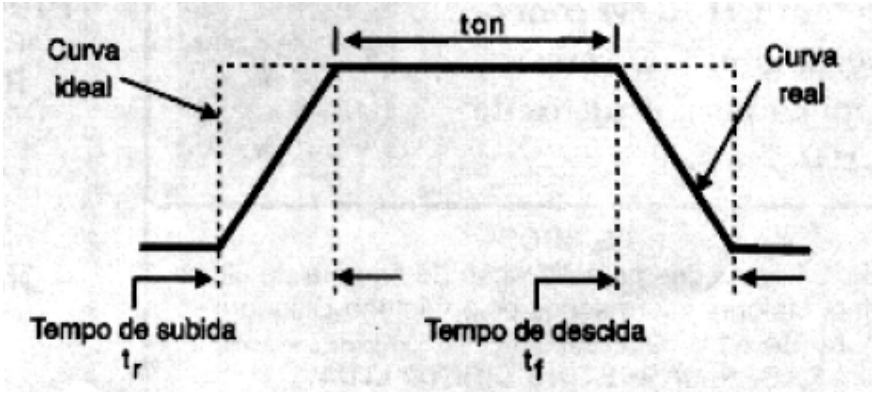
1.3.1.3 Vantagens do PWM

Na operação de um controle por PWM existem diversas vantegens a serem consideradas e alguns pontos para os quais o projetista deve ficar atento para não “jogar fora” estas vantagens.

Na condição de aberto, nenhuma corrente circula pelo dispositivo de controle e, portanto, sua dissipação é nula. Na condição de fechado, teoricamente, se ele apresenta uma resistência nula, a queda de tensão é nula, e ele não dissipa também nenhuma potência. Isso significa que, na teoria, os controles PWM não dissipam potência alguma e, portanto, consistem em soluções ideais para este tipo de aplicação.

Na prática, entretanto, isso não ocorre. Em primeiro lugar os dispositivos usados no controle não são capazes de abrir e fechar o circuito num tempo infinitamente pequeno. Eles precisam de um tempo para mudar de estado e, neste intervalo de tempo, sua resistência sobe de um valor muito pequeno até infinito e vice e versa, numa curva de comutação semelhante a mostrada na Figura 34.

Figura 34- Nos intervalos tr e tf, o dispositivo gera calor em boa qualidade



Fonte: (GHIRARDELLO, 2007?, p. 5)

Neste intervalo de tempo a queda de tensão e a corrente através do dispositivo não são nulas, e dependendo da frequência de controle e da resposta do dispositivo usado, uma boa quantidade de calor poderá ser gerada neste processo de comutação.

Entretanto, mesmo com este problema, a potência gerada num controle PWM ainda é muito menor do que num circuito de controle linear equivalente. Transistores de comutação rápidos, FETs de potência, e outros componentes de chaveamento podem ser suficientemente rápidos para permitir que projetos de controles de potências elevadas sejam implementados sem a necessidade de grandes dissipadores de calor ou que tenham problemas de perdas de energia por geração de calor que possam ser preocupantes.

O segundo problema que poderá surgir vem justamente do fato de que os transistores de efeito de campo ou bipolares usados em comutação não se comportam como resistências nulas, quando saturados. Os transistores bipolares podem apresentar uma queda de tensão de

até alguns volts quando saturados, o mesmo ocorrendo com os FETs. (GHIRARDELLO, 2007?).

1.4 COMUNICAÇÃO SERIAL RS232

A comunicação serial do tipo RS232 ainda hoje é muito utilizada para permitir a comunicação entre dispositivos. Dispositivos como mouses e modens são apenas alguns da variedade de equipamentos que podem ser conectados á porta serial. Apesar desta porta atualmente perder grande parte do seu mercado para a USB, é importante entender como ela funciona. A porta RS232 está disponível nos PCs no conector do tipo DB9 ou DB25, sendo ambos do tipo macho. O Conector DB9 geralmente é o mais usado no dia a dia. A Figura 35 mostra estes tipos de conectores e na Tabela 1 é possível visualizar a pinagem interna do conector DB9. (COMUNICAÇÃO SERIAL RS232, 2013?).

Figura 35- Conector DB9 (esquerda) e Conector DB25 (direita)

Fonte: (COMUNICAÇÃO SERIAL RS232, 2013?, p. 1)

Tabela 1- Pinagem do conector DB9

Pino	Descrição
1	DCD
2	RX
3	TX
4	DTR
5	GND
6	DSR
7	RTS
8	CTS
9	NC

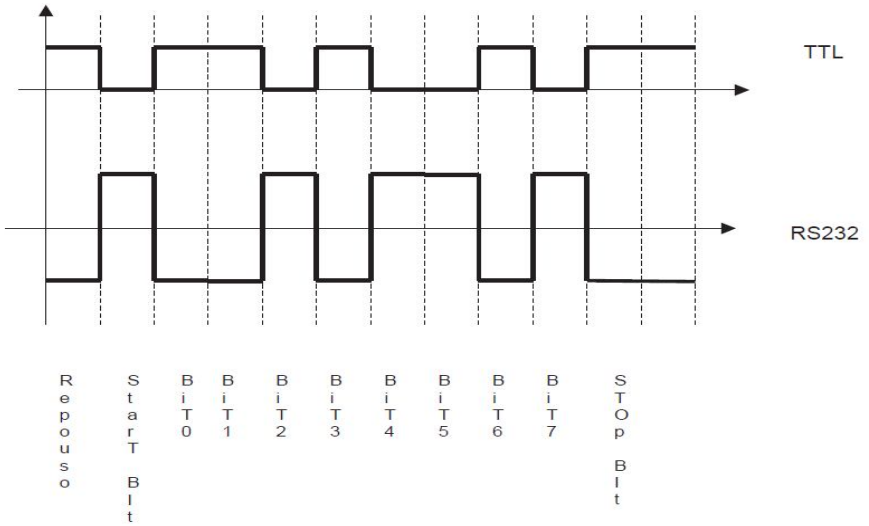
Fonte: (COMUNICAÇÃO SERIAL RS232, 2013?, p. 1)

Destes pinos, somente o 2, 3 e 5 são utilizados para prover comunicação entre dispositivos, ficando o restante para o controle do tráfego de dados. O padrão de comunicação utilizado pelo RS232 é diferente do utilizado pelo TTL[11], onde o nível 1 está associado a 5V e o nível 0 ao 0V. No padrão RS232, o nível 1 está associado a uma tensão de -3V a -18V enquanto o 0 está associado a uma tensão de 3V a 18V. Qualquer tensão dentro desta faixa será entendido como 1 ou 0.

Quando se trabalha com o RS232, é preciso primeiramente saber alguns parâmetros, como por exemplo a sua taxa de comunicação que é chamada de *baud rate*. O *baud rate* informa quantos *bits* no período de 1 segundo serão transferidos na linha. *Baud rates* comuns são o 2400, 4800 e 9600 bps[12].

Quando não há comunicação na linha RS232, ela normalmente fica no seu estado de repouso, que é o nível lógico 1 (de-3V a -18V no RS232). Quando inicia a comunicação o primeiro *bit* transferido é o chamado *bit* de *start*, que mantém a linha de comunicação no intervalo de 1 período em nível baixo. Em seguida vem os 8 *bits* de dados do *byte* a ser transmitido e finalmente o *bit* de *stop*, que volta a deixar a linha no seu estado de repouso. Na Figura 36 está plotado um gráfico que permite observar a comunicação tanto na linha RS232 quanto na linha TTL transmitindo neste caso o *byte* 01001011.

Figura 36- Plotagem do gráfico de comunicação



Fonte: (COMUNICAÇÃO SERIAL RS232, 2013?, p. 2)

[11] TTL: Lógica transistor-transistor
[12] Bps: *bits* por segundo

A linha permanece em estado alto inicialmente pelo fato de estar na condição de repouso. Logo em seguida, a comunicação é inicializada com um *bit* de *start*, que fica em nível lógico baixo por pelo menos 1 período e logo em seguida vêm os oito *bits* de dados referentes ao *byte* sendo transmitido. Após a transmissão dos oito *bits*, a comunicação é encerrada com um *bit* de *stop*, que deixa novamente a linha de dados em nível lógico alto, voltando desta forma ao seu estado de repouso. (COMUNICAÇÃO SERIAL RS232, 2013?).

1.5 LINGUAGEM DE PROGRAMAÇÃO

Um algoritimo é uma sequência lógica de ações a serem realizadas para se executar uma determinada tarefa. Um programa é a formalização de um algoritimo em uma determinada linguagem de programação, segundo suas regras de sintaxe e semântica, de forma a permitir que o computador possa entender a sequência de ações. (APOSTILA DE TÉCNICAS DE PROGRAMAÇÃO E LINGUAGEM PASCAL, 2008?).

Uma linguagem de programação é um conjunto de símbolos (comandos, identificadores, caracteres ASCII[13], etc.) e regras de sintaxe que permitem a construção de sentenças que descrevem de forma precisa ações compreensíveis e executáveis para o computador.

Uma linguagem de programação é uma notação formal para descrição de algoritimos que serão executados por um computador. Como todas as notações formais, uma linguagem de programação tem dois componetes: Sintaxe e Semântica. Sintaxe consiste em um conjuto de regras, que especificam a composição de programas a partir de letras, digitos e outros símbolos. Por exemplo, regras de sintaxe podem especificar que cada parênteses aberto em uma expressão aritmética deve corresponder a um parênteses fechado, e que dois comandos quaisquer devem ser separados por um ponto e vírgula. As regras de semântica especificam o "significado" de qualque programa, sintatitamente válido, escrito na linguagem.

Há diversas linguagens de programação, cada uma com suas características específicas e com níveis de complexidade e objetivos diferentes, como pode ser visto na Tabela 2.

[13] ASCII: Código Padrão Americano para o Intercâmbio de Informação

Tabela 2- Tipos de linguagens de programação e suas características

LINGUAGEM	CARACTERÍSTICAS
Linguagem de Máquina	Única compreendida pelo computador. Específica de cada computador.
Linguagens de Baixo Nível	Utiliza mnemônicos para representar instruções elementares Ex.: Assembly
Linguagens de Alto Nível	Utiliza instruções próximas da linguagem humana de forma a facilitar o raciocínio. Ex.: Uso Científico : Fortran Propósito Geral : Pascal, C, Basic Uso Comercial : Cobol, Clipper Uso específico : Lisp, Prolog

Fonte: (APOSTILA DE TÉCNICAS DE PROGRAMAÇÃO E LINGUAGEM PASCAL, 2008?, p. 5)

1.5.1 Processo de Criação e Execução de um Programa

Embora seja teoricamente possível a construção de computadores especiais, capazes de executar programas escritos em uma linguagem de programação qualquer, os computadores, existentes hoje em dia são capazes de executar somente programas em linguagem de baixo nível, a linguagem de máquina.

Lingugens de máquina são projetadas levando-se em conta os seguintes aspectos:

a) Rapidez de execução de programas;

b) Custo de sua implementação;

c) Flexibilidade com que permite a construção de programas de nível mais alto.

Por outro lado, linguagens de programação de alto nível são projetadas em função da facilidade de construção de programas, gerando uma confiabilidade maior.

Entendo a projeção das linguagens de máquinas e de nível alto surge um problema: como a linguagem de nível mais alto pode ser implementada em um computador, cuja linguagem é bastante diferente e de nível mais baixo?

Para solucionar o problema é preciso realizar a tradução de programas escritos em linguagens de alto nível para a linguagem de baixo nível do computador. Para isso existem três tipos de programas tradutores:

- **Montador:** Efetua a tradução de linguagem de montagem (*Assembly*) para a linguagem de máquina da seguinte forma:

a) Obtém a próxima instrução do *Assembly*;

b) Traduz para as instruções correspondentes em linguagem de máquina;

c) Executa as instruções em linguagem de máquina;

d) Repete o passa “a” até o fim do programa.

- **Interpretador:** Efetua a tradução de todo o código fonte em linguagem de alto nível para as instruções correspondentes em linguagem de máquina, gerando o código objeto do programa. Em seguida é necessário o uso de outro programa (*Link-Editor*) que é responsavel pela junção de diversos códigos objeto em um único programa executável.

Existe a probabilidade dos seguintes erros no programa:

a) Erros de compilação: causados geralmente por erros de digitação e de uso da sintaxe da linguagem;
b) Erros de *Link*-Edição: erro no uso de bibliotecas de sub-programas necessárias ao programa principal;
c) Erros de execução: erros relacionados a lógica do programa (algoritmo).

1.5.1.1 Critérios de Qualidade de um Programa

Abaixo segue alguns critérios para escrever um programa com qualidade:

a) **Integridade:** refere-se à precisão das informações manipuladas pelo programa, ou seja, os resultados gerados pelo processamento do programa devem estar corretos, caso contrário o programa simplesmente não tem sentido;
b) **Clareza:** refere-se a facilidade de leitura do programa. Se um programa for escrito com clareza, deverá ser possível a outro programador seguir a lógica do programa sem muito esforço, assim como o próprio autor do programa entendê-lo após ter estado um longo período afastado dele;
c) **Simplicidade:** a clareza e precisão de um programa são normalmente melhoradas tornando seu entendimento o mais simples possível, consistente com os objetivos do programa. Muitas vezes torna-se necessário sacrificar alguma eficiência de processamento, de forma a manter a estrutura do programa mais simples;
d) **Eficiência:** refere-se a velocidade de processamento e a correta utilização da memória. Um programa deve ter performance suficiente para atender às necessidades do problema e do usuário, bem como deve utilizar os recursos de memória de forma moderada, dentro das limitações do problema;
e) **Modularidade:** consiste no particionamento do programa em módulos menores bem identificáveis e com funções específicas, de forma que o conjunto desses módulos e a interação entre eles permite a resolução do problema de maneira mais simples e clara;

f) **Generalidade:** é interessante que um programaseja tão genérico quanto possível de forma a permitir a reutilização de seus componentes em outros projetos.

1.5.2 A linguagem C

A Linguagem C, criada em 1970 por Dennis Ritchi é uma evolução da Linguagem B que, por sua vez, foi uma adaptação feita, a partir da Linguagem BCPL, por Ken Thompson. Ela é estreitamente associada ao sistema operacional UNIX, já que as versões atuais do próprio sistema foram desenvolvidas utilizando-se esta linguagem. (MARTINS, 2005?).

Devido ao crescente uso e interesse da comunidade de computação pela linguagem, em 1983, o *American National Standards Institute* (ANSI), estabeleceu um comitê para prover uma definição moderna e abrangente da linguagem C. O resultado desta comissão foi uma padronização denominada ANSI-C em 1988. A maior contribuição deste trabalho foi a incorporação de uma biblioteca padrão, presentes em todas as variações/versões da linguagem, que fornece funções de acesso ao sistema operacional, entrada e saída formatada, alocação de memória, manipulação de *strings*[14] etc.

1.5.2.1 Conceitos Básicos

A filosofia básica da linguagem C é que os programadores devem estar cientes do que estão fazendo, ou seja, supõe-se que eles saibam o que estão mandando o computador fazer, e explicitem completamente as suas instruções. Assim, ela alia a elegância e a flexibilidade das linguagens de alto nível (ex: suporte ao conceito de tipo de dados) com o poderio das linguagens de baixo nível (ex: manipulação de *bits*, *bytes* e endereços).

O C é uma linguagem de programação de finalidade geral, utilizada no desenvolvimento de diversos tipos de aplicação, como processadores de texto, sistemas operacionais, sistemas de comunicação, programas para solução de problemas de engenharia, física, química e outras ciências, etc.

O código-fonte de um programa C pode ser escrito em qualquer editor de texto que seja capaz de gerar arquivos em código ASCII (sem formatação). Como o ambiente de programação utilizado (Turbo C) é para o sistema operacional DOS, estes arquivos devem ter um nome de no máximo 8 caracteres e a extensão “c” (exemplo: NONAME.C).

[14] *String*: Cadeia de Caracteres

Após a implementação, o programa-fonte (um ou mais arquivos-fonte) é submetido aos processos de compilação e linkedição para gerar o programa executável (com extensão “exe”). Durante o processo de compilação, cada arquivo-fonte é compilado separadamente, produzindo um arquivo de código-objeto com a extensão “obj”. Estes arquivos-objeto contêm instruções em linguagem de máquina (códigos binários) entendidas somente pelos microprocessadores. Na linkedição, todos os arquivos-objetos pertencentes ao projeto, bem como as bibliotecas declaradas nos códigos-fonte são processadas em conjunto, visando a produção do arquivo executável correspondente.

Normalmente, tanto o arquivo-objeto quanto o arquivo executável possuem o mesmo nome do arquivo-fonte. Entretanto, quando desejado, o usuário poderá definir diferentes nomes para cada tipo de arquivo. (MARTINS, 2005?).

1.5.2.2 Características Gerais

A linguagem C possui as seguintes características:

a) Alta portabilidade inerente da padronização ANSI, ou seja, é possível tomar um código-fonte escrito para uma máquina, compilá-lo e rodá-lo em outra com pouca ou nenhuma alteração;
b) Gera programas formados basicamente por funções, o que facilita a modularização e a passagem de parâmetros entre os módulos;
c) Inicia a execução a partir da função *main*(), necessária em todos os programas;
d) Uso de chaves ({ }) para agrupar comandos pertencentes a uma estrutura lógica (ex: *if-else*, *do-while*, *for*, etc.) ou a uma função;
e) Uso do ponto e vírgula (;) ao final de cada comando;
f) É “*case sensitive*”, ou seja, o compilador difere maiúsculas de minúsculas. Assim, se declarar uma variável de nome *idade*, esta será diferente de *Idade*, *IDADE*, etc. Além disso, todos os comandos da linguagem devem ser escritos em minúsculo.

1.5.3 Delphi

O *Delphi* é uma ferramenta RAD (*Rapid Application Development* – Desenvolvimento Rápido de Aplicações) criada pela *Borland*. É uma ferramenta de propósito geral, permitindo o desenvolvimento de aplicações tanto científicas como comerciais com a

mesma facilidade e alto desempenho. Integra-se facilmente com a API (*Application Program Interface*) do *Windows*, permitindo a criação de programas que explorem ao máximo os seus recursos, assim como os programas escritos em linguagem C/C++. (DESENVOLVENDO APLICAÇÕES ORIENTADAS A OBJETOS COM O BORLAND DELPHI, 2003?).

Possui um compilador extremamente rápido, que gera executáveis nativos (em código de máquina, não interpretado), obtendo assim melhor performance e total proteção do código fonte.

O *Delphi* é extensível, sua IDE (Ambiente de Desenvolvimento Integrado) pode ser ampliada e personalizada com a adição de componentes e ferramentas criadas utilizando-se o *Object Pascal*, a linguagem de programação do *Delphi*. Neste ambiente constroem-se as janelas das aplicações de maneira visual, ou seja, arrastando e soltando componentes que irão compor a interface com o usuário.

O *Object Pascal* é uma poderosa linguagem Orientada a Objeto, que além de possuir as características tradicionais das mesmas como classes e objetos, também possui interfaces (semelhantes às encontradas em COM e Java), tratamento de exceção, programação *multithreaded* e algumas características não encontradas nem mesmo em C++, como RTTI (*Runtime Type Information*). Assim como o C++, o *Object Pascal* é uma linguagem híbrida, pois além da orientação a objeto possui também uma parte da antiga linguagem estruturada (Pascal).

Devido ao projeto inicial da arquitetura interna do *Delphi* e da orientação a objeto, suas características básicas mantêm-se as mesmas desde o seu lançamento em 1995 (ainda para o *Windows* 3.1, pois o *Windows* 95 ainda não havia sido lançado), o que demonstra um profundo respeito com o desenvolvedor. Isto permite que uma aplicação seja portada de uma versão anterior para uma nova, simplesmente recompilando-se o código fonte. (DESENVOLVENDO APLICAÇÕES ORIENTADAS A OBJETOS COM O BORLAND DELPHI, 2003?).

Programadores viram nesse ambiente revolucionário a possibilidade de administrar com mais facilidade e segurança o que antes era penoso e cansativo. Com *Object Pascal* o desenvolvimento de um programa utiliza o conceito de classes e objetos, tornado-se uma poderosa ferramenta de programação. Várias versões foram lançadas ao longo de décadas, as versões mais famosas foram: (LINGUAGEM DE PROGRAMAÇÃO ORIENTADA A OBJETO, 2012).

a) **Delphi 4**: Com Interface Gráfica melhorada, barra de ferramentas e janelas de encaixe, IDE com recursos diferenciados;

b) **Delphi 5**: Foram integrados componentes para ambiente multicamada e *internet*;
c) **Delphi 7**: Com a suíte Intraweb, surgia a possibilidade de desenvolver aplicações *web* para rodar num servidor *Apache* com muito mais facilidade.

1.5.3.1 Princípios da Programação para *Windows*

É importante ter algumas noções do que está envolvido na programação *Windows* e no *Delphi* em particular. Algumas coisas tornam a tarefa de programação no *Windows* (e ambientes baseados em eventos e interface gráfica) bem diferente de outros ambientes e das técnicas de programação estruturada normalmente ensinadas nos cursos de lógica de programação:

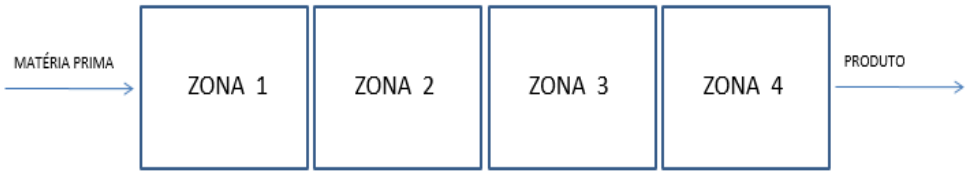
a) **Independencia do hardware**: No *Windows*, o acesso aos dispositivos de *hardware* é feito com intermédio de *drivers* fornecidos pelo fabricante do *hardware*, o que evita que o programador tenha que se preocupar com detalhes específicos do *hardware*. Como acontecia com a programação em DOS;
b) **Configuração padrão**: O *Windows* armazena centralmente as configurações de formato de números, moeda, datas e horas, além da configuração de cores, livrando o programador de se preocupar com esses detalhes específicos;
c) **Multitarefa**: Antes, no DOS, um programa geralmente tomava o controle da máquina só para si, e outros programas não rodavam até que o mesmo fosse fechado. Já no *Windows* vários programas são executados de maneira simultânea e não há como evitar isso;
d) **Controle da tela**: No DOS geralmente um programa ocupa todo o espaço da tela, e o usuário via e interagia apenas com aquele programa. Já no *Windows*, todas informações mostradas e todas entradas recebidas do usuário são feitas por meio de uma janela, uma área separada da tela que pode ser sobreposta por outras janelas do mesmo ou de outros programas;
e) **Padrões de interface**: No *Windows*, todos os elementos de interface aparecem para o usuário e interagem da mesma forma. Além disso, existem padrões definidos pela *Microsoft* que são recomendados para conseguir a consistência entre aplicativos;

2 MÉTODO PROPOSTO

Fornos elétricos resistivos são largamente utilizados nas indústrias, geralmente são compostos por uma ou mais zonas de aquecimento, um conjunto de resistências elétricas e uma carcaça metálica. Vários utilitários como pratos, talheres, cerâmicas e até mesmo uma silples caneta utilizam fornos em seu processo de fabricação.

No processo de fabricação de produtos que utilizam forno elétrico resistivo, a matéria-prima frequentemente é depositada de forma bruta na primeira zona de aquecimento do forno, nesta primeira zona a matéria-prima é derretida formando um material “liguento” que será moldado a medida em que for transferido de uma zona para outra até formar o produto final, conforme ilustra a Figura 37.

Figura 37- Processo Produtivo Utilizando Forno Elétrico Resistivo



Fonte: (O AUTOR

Para ter sucesso na fabricação de produtos utilizando forno elétrico resistivo, é fundamental que a temperatura das zonas de aquecimento não sofram oscilações durante o processo.

Visando tornar nulo ou próximo de nulo as oscilações de temperatura em uma zona de aquecimento do forno elétrico resistivo, foi proposto neste trabalho o desenvolvimento de um controle PID, corrigindo a inércia térmica tradicionais em alguns sistemas de controle (os do tipo *ON-OFF* por exemplo), possibilitando ao operador do processo visualizar gráficamente a temperatura real da zona de aquecimento e atuando de forma preventiva, parando o processo caso se perceba alguma irregularidade no sistema, como por exemplo uma temperatura superior ao *set-point*.

2.1 ETAPAS PARA O DESENVOLVIMENTO DO PROJETO

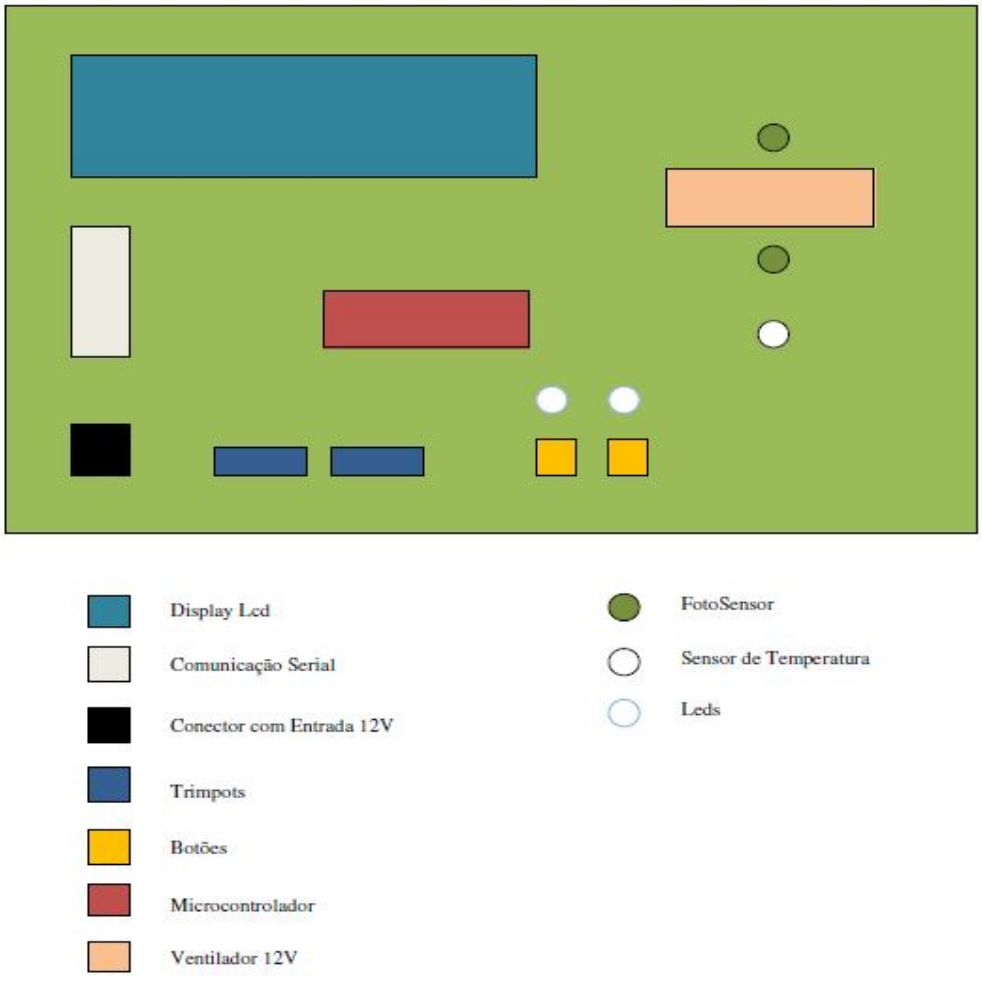
Para uma melhor estruturação, o desenvolvimento do controle PID foi divido em 4 etapas, dessa forma seguiu-se um cronagroma, a fim de que ao final da última etapa o trabalho pudesse está finalizado. As etapas estão descritas a seguir:

2.1.1 Etapa 1 – Pesquisas Bibliográficas

Na primeira etapa foram realizadas pesquisas bibliográficas em assuntos relacionados ao controle do tipo PID, com o objetivo de reunir as informações e dados que serviram de base para o desenvolvimento deste trabalho. Um estudo aprofundado sobre tipos de controle industrial foi realizado, destacam-se nessa pesquisa a disciplina de microcontroladores, onde foi possível específicar uma placa contendo o kit representado na

Figura 38.

Figura 38- Detalhe do Kit PID usado no projeto



Fonte: (O AUTOR)

A partir da pesquisa bibliográfica foi possível elaborar a melhor metodologia a ser aplicada no trabalho. O microcontrolador utililizado no kit é o PIC16F876A. Uma vez que o microcontrolador do projeto foi especificado determinou o software MIKROC para a programação do PIC16F876A e o compilador WINPIC800 para gravar o programa elaborado no software MICROC no PIC16F876A. Definindo estes itens seguiu-se para a segunda etapa.

2.1.2 Etapa 2 – Elaboração do algoritimo para o controle PID

Nessa etapa foi elaborado o programa no software MICROC, o programa foi elaborado utilizando linguagem C. O algoritimo descreve as etapas que precisam ser efetuadas para que um programa execute as tarefas que lhe são designadas. No apêndice A está disponível o algoritimo utilizado neste trabalho. O algoritimo está comentado para melhor compreensão do programa.

2.1.3 Etapa 3 – Elaboração da tela de controle e monitoramento

Nessa etapada foi construída uma tela, uma espécie de supervisório, para determinar o tipo de controle a ser utilizado, no caso desse projeto o do tipo ON-OFF e PID, o valor de *set point* do sistema, os limites de histerese, os ajuste do controle PID. Também nessa tela é possível verificar graficamente o comportamento do sistema. Esta tela foi confeccionada utilizando o ambiente DELPHI 7. Logo após seguiu-se para a quarta etapa.

2.1.4 Etapa 4 – Testes de funcionalidade do Sistema

Nessa última etapa, já com o programa desenvolvido e transferido para o microcontrolador PIC16F876A e as tela de controle e monitoramento do processo elaboradas no DELPHI 7, foi realizados testes de funcionalidade tanto no controle ON-OFF como também no PID, desta forma foi possível comparar os dois tipos de controle que é o objetivo desse projeto.

3 IMPLEMENTAÇÃO DO PROJETO

Após realizadas as pesquisas bibliográficas determinou-se *hardwares* e *softwares* necessários para o desenvolvimento do controle de temperatura PID, que foram específicados de modo a reduzir os custos com a implementação e visando atender as necessidades do projeto. Este tópico detalha exatamente isso.

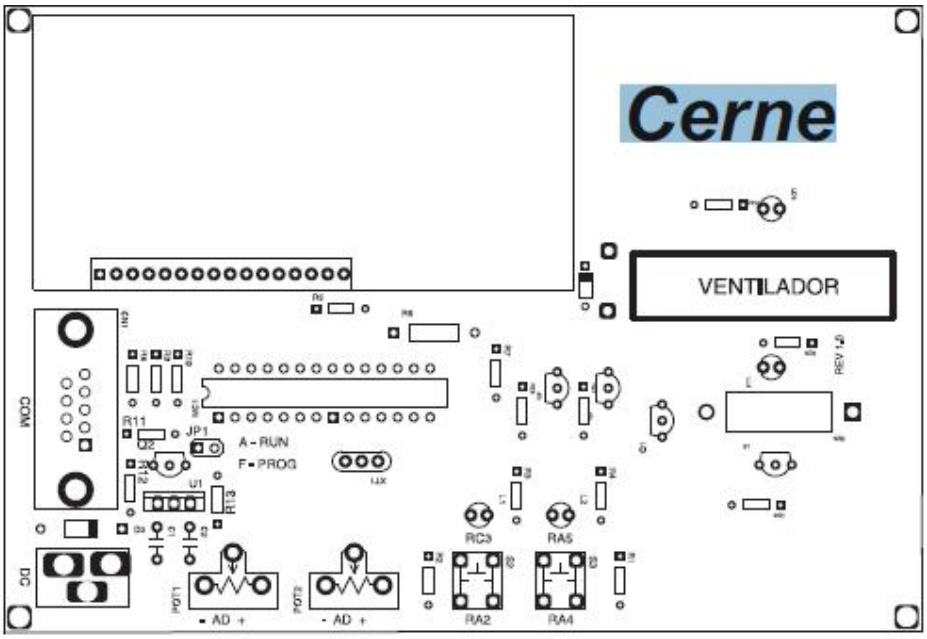
3.1 HARDWARES UTILIZADOS

Para o desenvolvimento do controle PID foram utilizadas várias ferramentas, dispositivos, equipamentos, são eles:

3.1.1 Placa com o KIT PID

Detalhada na Figura 39. A placa de fornecimento da Cerne Tecnologia, contém um display onde é mostrado o valor de temperatura real do ambiente. Possui também um ventilador que é utilizado como atuador do sistema. A placa dispõe de um microcontrolador PIC PIC16F876A onde é feito todo o controle do processo, led's para indicar alarme de sobretemperatura, porta serial para comunicação RS232, pino de entrada para alimentação da placa 12vcc. Abaixo é respresentado o *layout* da placa com o KIT PID.

Figura 39- Layout do placa utilizada no projeto



Fonte: (O AUTOR)

A placa também dispões de um cabo serial RS232, utilizado para comunicação entre placa PID e o computador, sendo possível transferir (gravar) o código fonte dentro do microcontrolador e interfacear com o o supervisório contruído, para receber os comandos e enviar as respostas (*feedback*) do sistema.

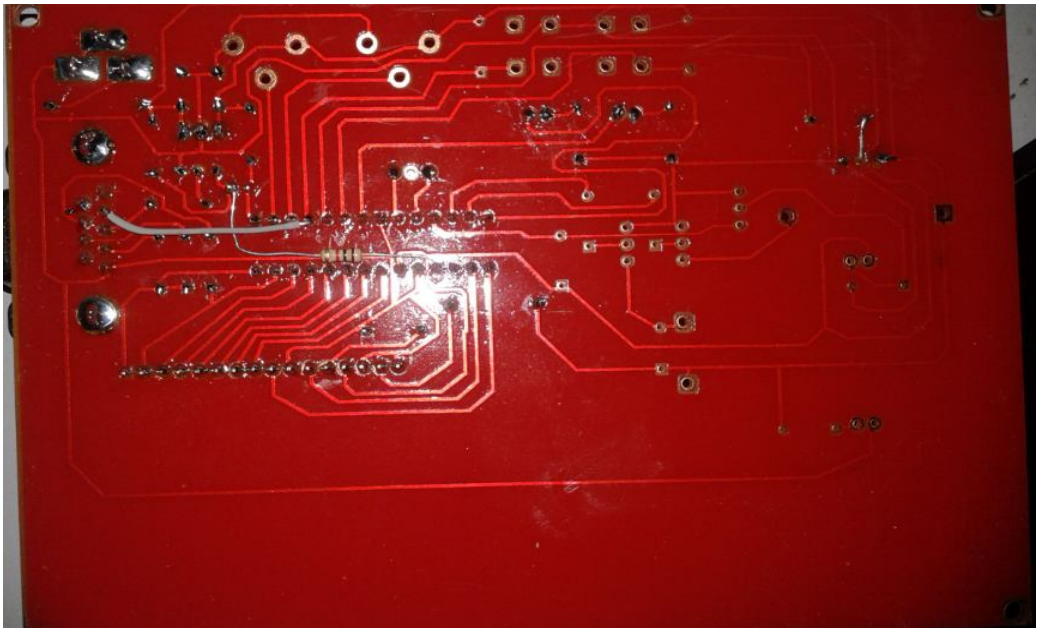
Para simular a zona de aquecimento do forno elétrico resistivo utilizou-se a própria placa com o KIT PID, pois a mesma também possui o sensor LM35, possibilitando assim realizar simulações e testes na própria placa.

O uso da placa reduziu o custo com mão de obra, pois todos os dispositivos necessários para o desenvolvimento deste trabalho encontram-se reunidos na mesma placa.

3.1.2 Ferro de Solda Tramontina 25w Modelo 43752/502

O ferro de solda foi utilizado para fixar e extrair dispositivos da placa. Para o funcionamento correto do comunicação serial RS232, foi necessário realizar modificações no circuito, para tanto uilizou o ferro de solda para fixar componentes na placa PID conforme mostra a Figura 40. O ferro de solda também foi utilizado para os testes de funcionamento, servindo de apoio ao ventilador existente na placa, dessa forma o ventilador ao ligar joga ar quente, simulando dessa forma uma resistência.

Figura 40- Alterações feitas na Placa com o KIT PID



Fonte: (O AUTOR)

3.2 SOFTWARES UTILIZADOS

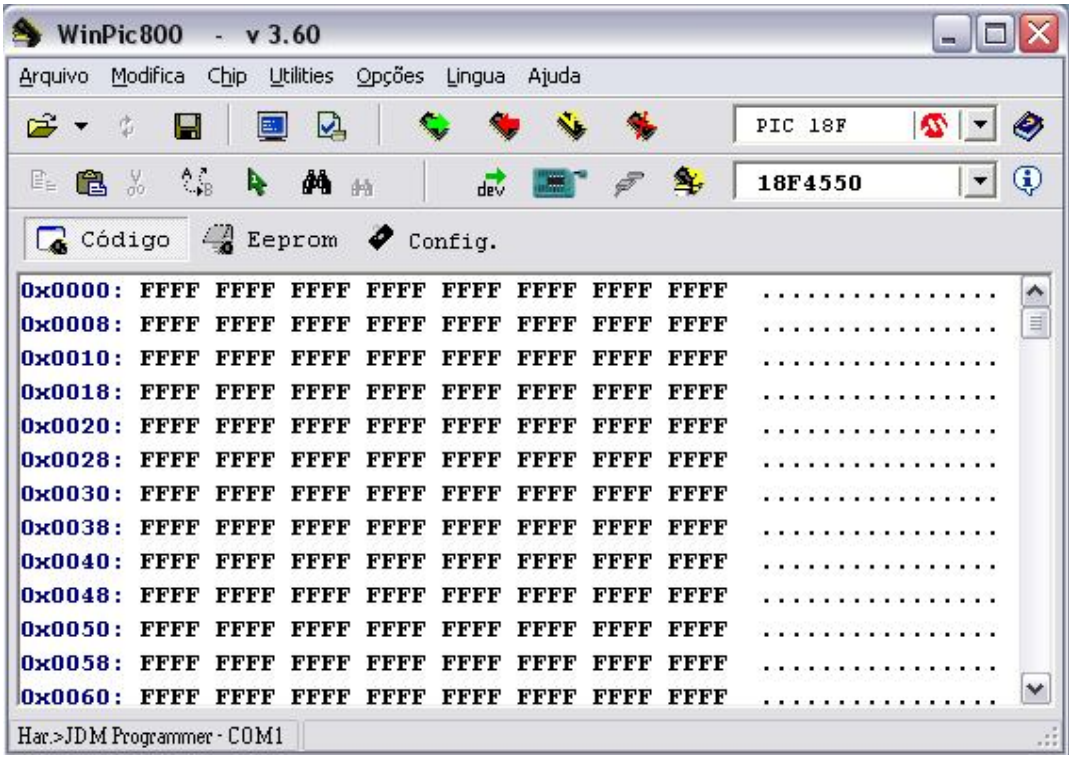
3.2.1 Software MikroC

Nesta plataforma foi desenvolvida toda lógica do controle. Utilizando linguagem C, o *script* do programa pode ser verificada abaixo no Apêndice A. O programa elaborado está comentado para melhor compreensão da lógica de controle.

3.2.2 Software WINPIC800

Neste software foi possível gravar o *script* do programa elaborado no Softawre MikroC. Funciona como uma espécie de gravador, nesse projeto foi utilizado a V 3.60. A Figura 41 apresenta a tela inicial da plataforma WinPic800.

Figura 41- Tela inicial do WINPIC800

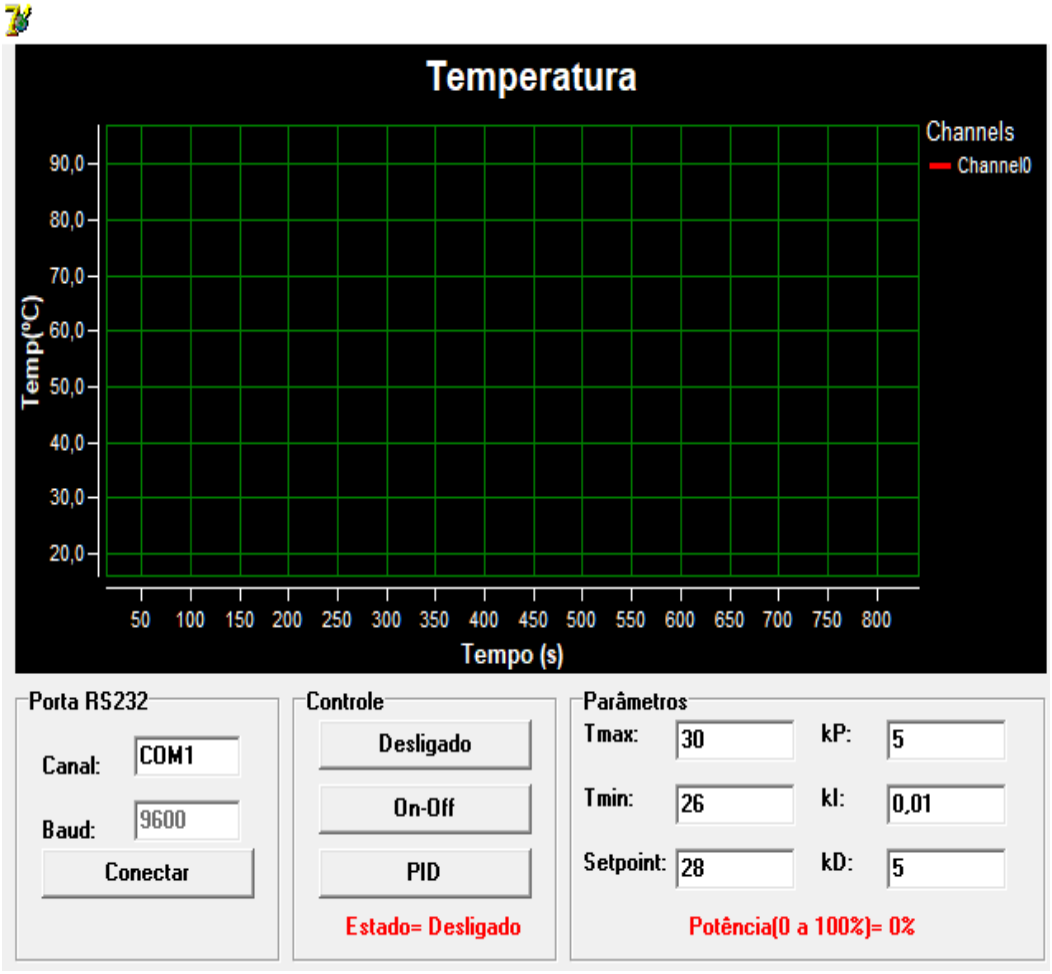


Fonte: (Software Winpic800)

3.2.3 Delphi 7

Nessa ferramenta foi desenvolvido a tela de controle e monitoramento do sistema, nesta tela define-se o tipo de controle utilizado, o valor de *set-point* do sistema, níveis de histerese, configuração da porta serial RS232, parâmetros para configuração do controle PID e é possível observar nessa tela o comportamento do sistema através do gráfico que esboça em tempo real as informações trabalhadas no sistema, neste gráfico é possível avaliar o tempo de entre o estado de partida do sistema até chegar no valor de temperatura definido pelo operador do processo. A tela pode ser visuliazida na Figura 42.

Figura 42- Tela de Controle e Monitoração do projeto



Fonte: (O AUTOR)

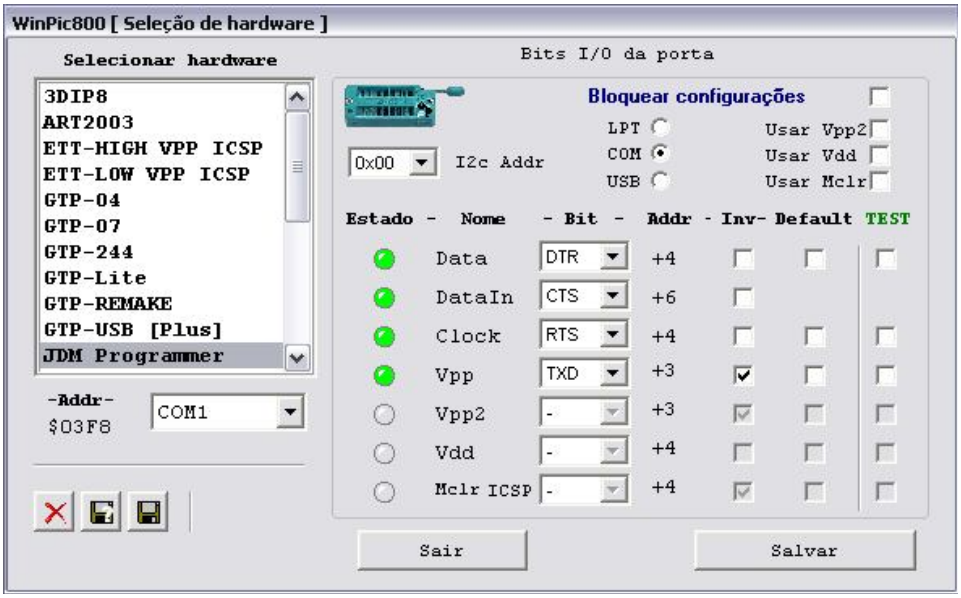
3.3 PROCESSO DE PREPARAÇÃO PARA FUNCIONAMENTO DO SISTEMA

Com a placa PID já modificada conforme mostra a Figura 40 do item 3.1.2 deve-se conectar o cabo serial que acompanha o kit na placa PID da cerne ao PC. Logo após é necessário ligar a fonte de alimentação na tomada e o conector que sai dela na placa PID.

Feito as conexões deve-se abrir o programa WinPic800 que deverá estar instalado no PC, após esse passo a tela de inicialização do programa será exibida que pode ser visualizada na Figura 41 do item 3.2.2.

Com o programa aberto é hora de configurar o gravador utilizado pelo WinPic800. Para isso é precisso ir no menu opções e selecionar a aba *hardware*. Na janela que for aberta, o gravador escolhido deve ser o *JDM Programmer*. Após este passo, a configuração deste gravador deve estar de acordo como mostra a Figura 43. Depois é só pressionar o botão salvar e sair.

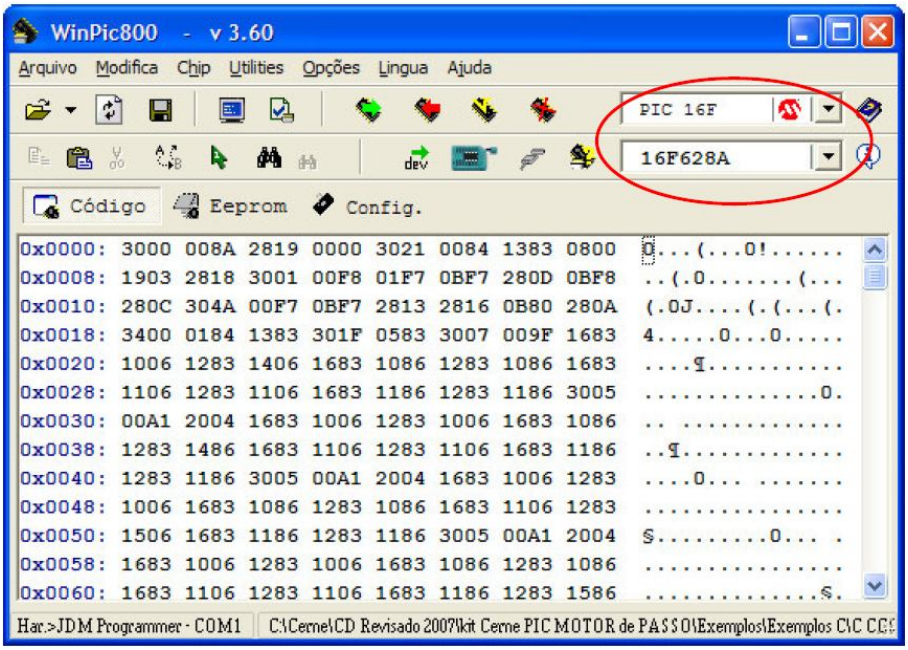
Figura 43- Configuração do JDM programmer



Fonte: (O AUTOR)

O próximo passo é selecionar o microcontrolador a ser gravado, conforme ilustra a Figura 44. O microcontrolador escolhido deve ser o PIC16F876A

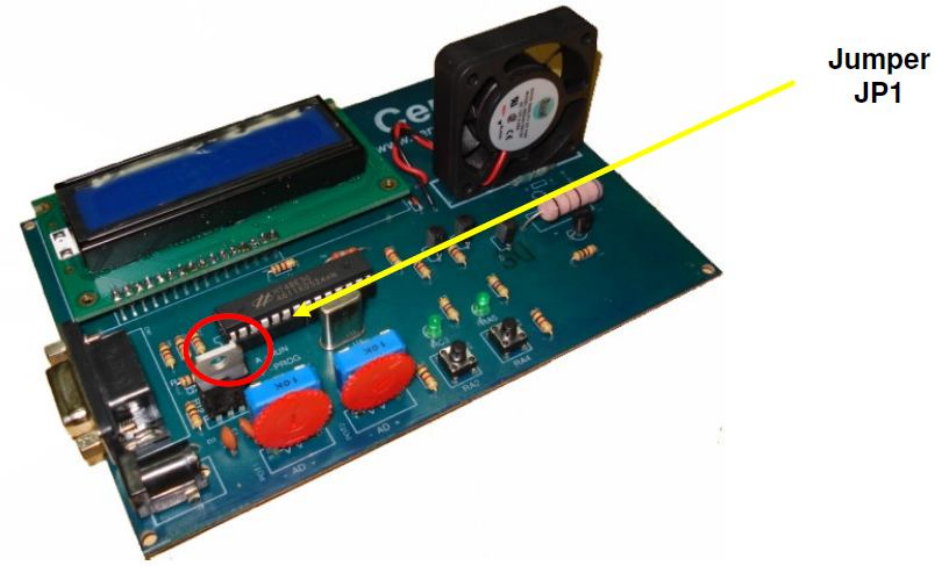
Figura 44- Selecionando o microcontrolador



Fonte: (O AUTOR)

Para gravar o programa no microcontrolador é necessário deixar o *jumper* JP1 fechado, assim que a gravação finalizar, este *jumper* deve ser aberto novamente. O *Jumper* pode ser visualizado na Figura 45.

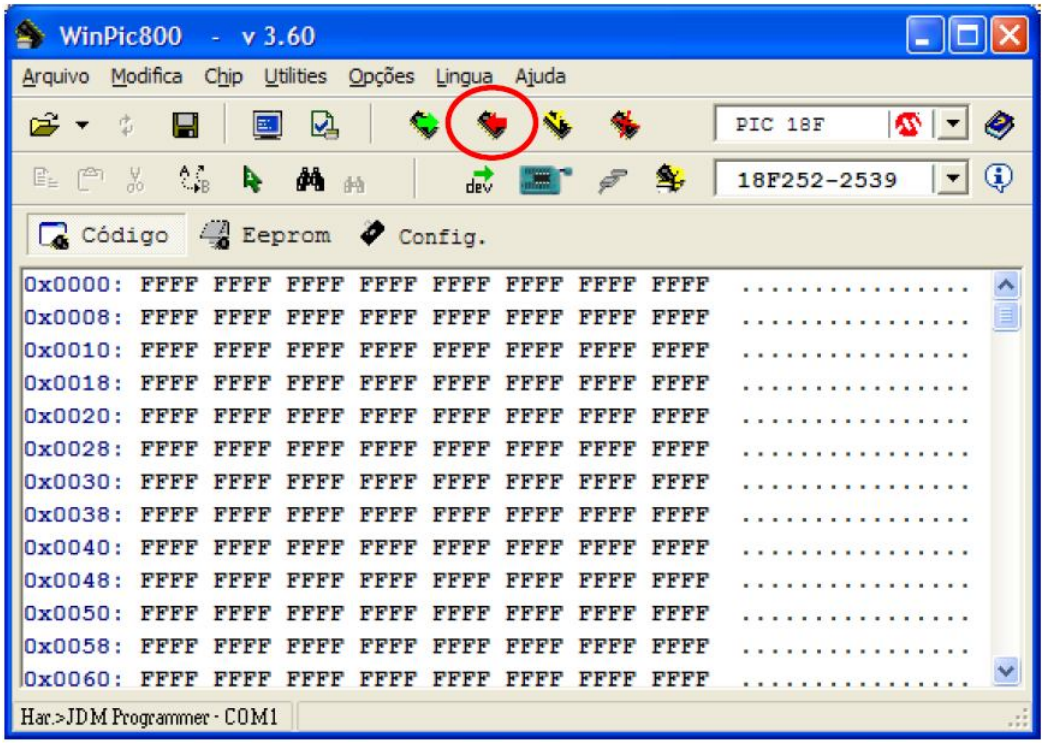
Figura 45- Jumper JP1



Fonte: (O AUTOR)

Feito isso é necessário abrir o arquivo hex, este arquivo foi gerado pelo compilador MikroC, ele nada mais é do que a lógica do programa de controle que foi elaborada. Pra isso é preciso ir no menu Arquivo do WinPic800 e escolher a opção Abrir e selecionar o arquivo em hex que foi salvo, nesse projeto o arquivo foi salvo com o nome 'controle_pid.hex'. Com o arquivo carregado pode iniciar a gravação, para isso deve-se pressionar o botão marcado na Figura 46 abaixo:

Figura 46- Botão para íniciar gravação



Fonte: (O AUTOR)

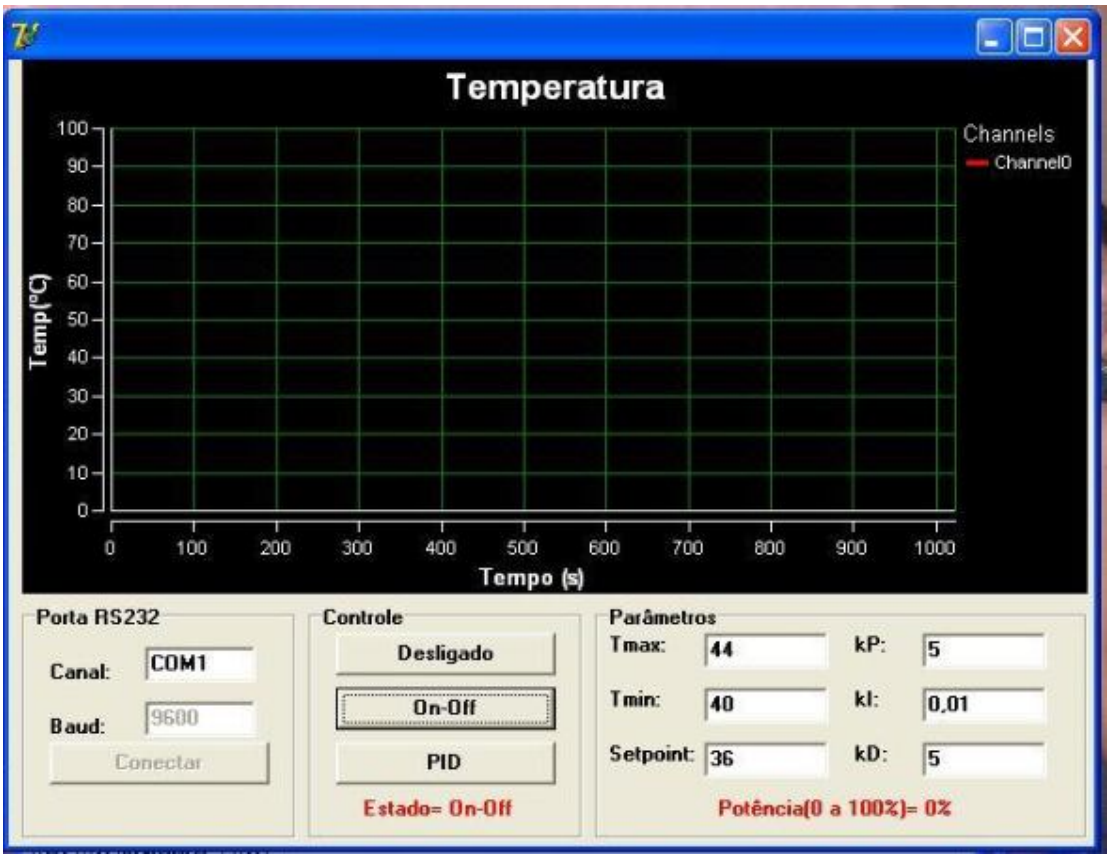
Após o término da gravação, deverá aparecer uma caixa indicando o sucesso na gravação. Caso esta mensagem não apareça, é preciso revisar as conexões e tentar gravar novamente.

Agora é só abrir o jumper JP1 e realizar os testes do programa.

4 RESULTADOS OBTIDOS

Para realização dos testes foi utilizado o ferro de solda descrito no item 3.1.2, o ferro de solda encostava próximo ao ventilador, simulando o aumento da temperatura. O primeiro teste realizado foi referente ao controle on-off, para isso abriu-se a tela desenvolvida no Delphi 7 e configurou o processo de acordo como mostra a Figura 47.

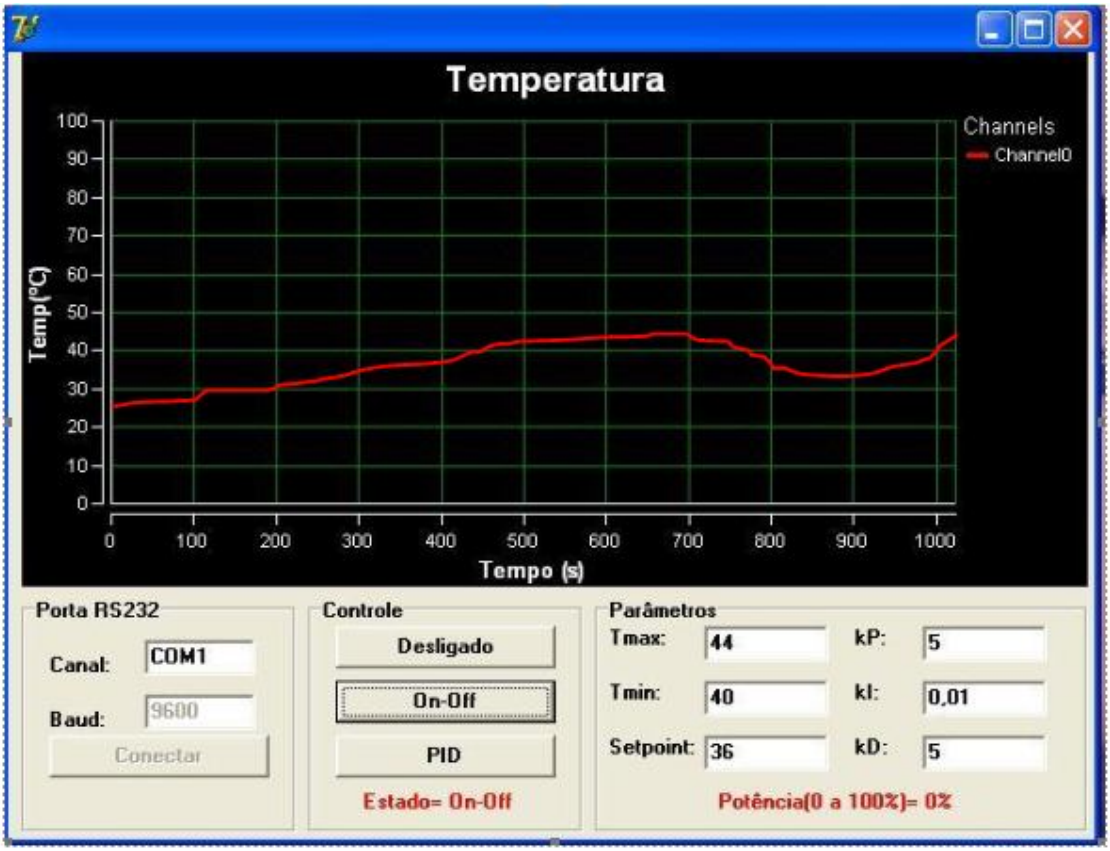
Figura 47- Processo com o controle ON-OFF



Fonte: (O AUTOR)

O canal COM1 foi definido automaticamente quando conectou a porta serial ao PC, no setpoint foi atribuído o valor de 40°C. O sistema respondeu da seguinte forma: Inicialmente o sistema partiu com a temperatura real de 26°C, para atingir o setpoint 40°C levou em torno de 450 segundos, após atingir o setpoint a inércia térmica foi de 4°C, após isso continuou-se o ciclo de acordo como mostra a Figura 48.

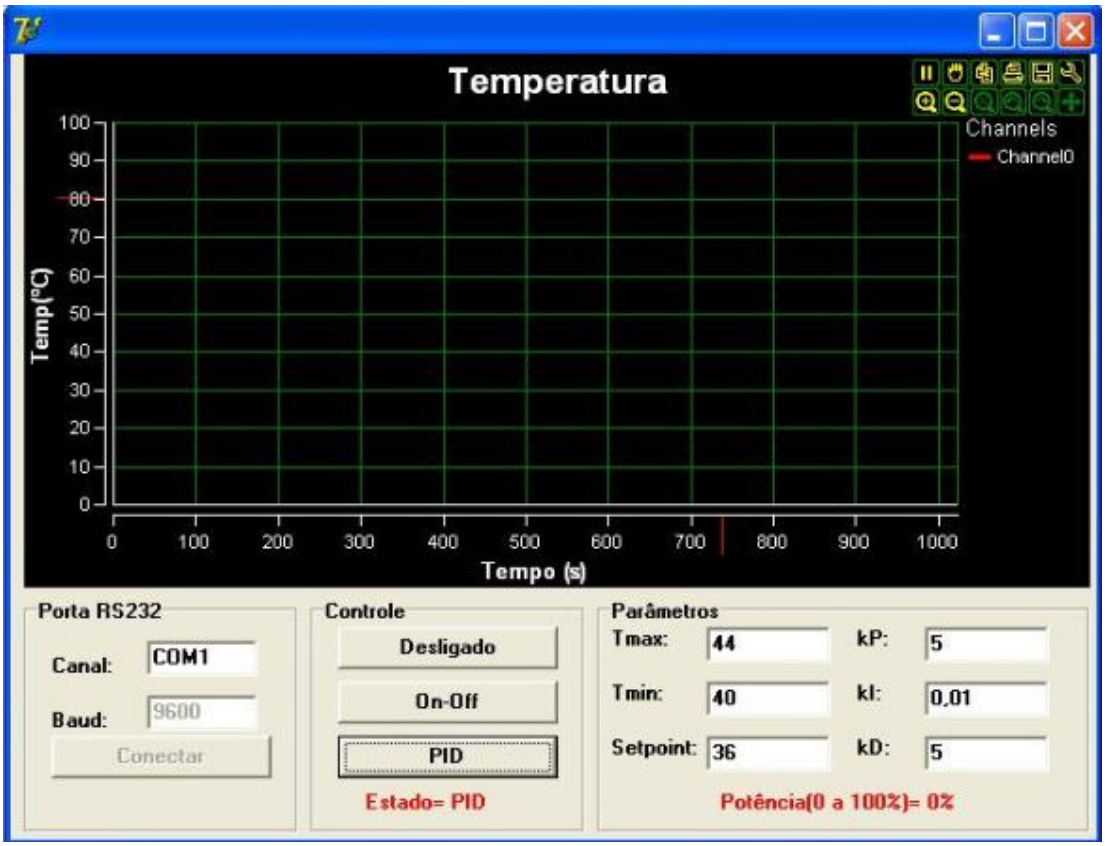
Figura 48- Resposta do Controlador ON-OFF



Fonte: (O AUTOR)

O segundo teste realizado foi referente ao controle PID, manteve as mesmas configurações do controle ON-OFF, apenas mudou o tipo de controle para PID conforme mostra a figura Figura 49. Para o ajuste das constantes KP, KI e KD foram realizados testes até o gráfico apresentar valores condizentes ao esperado, dessa forma atribuiu-se em KP e KD o valor de 5 e em KI o valor de 0,1.

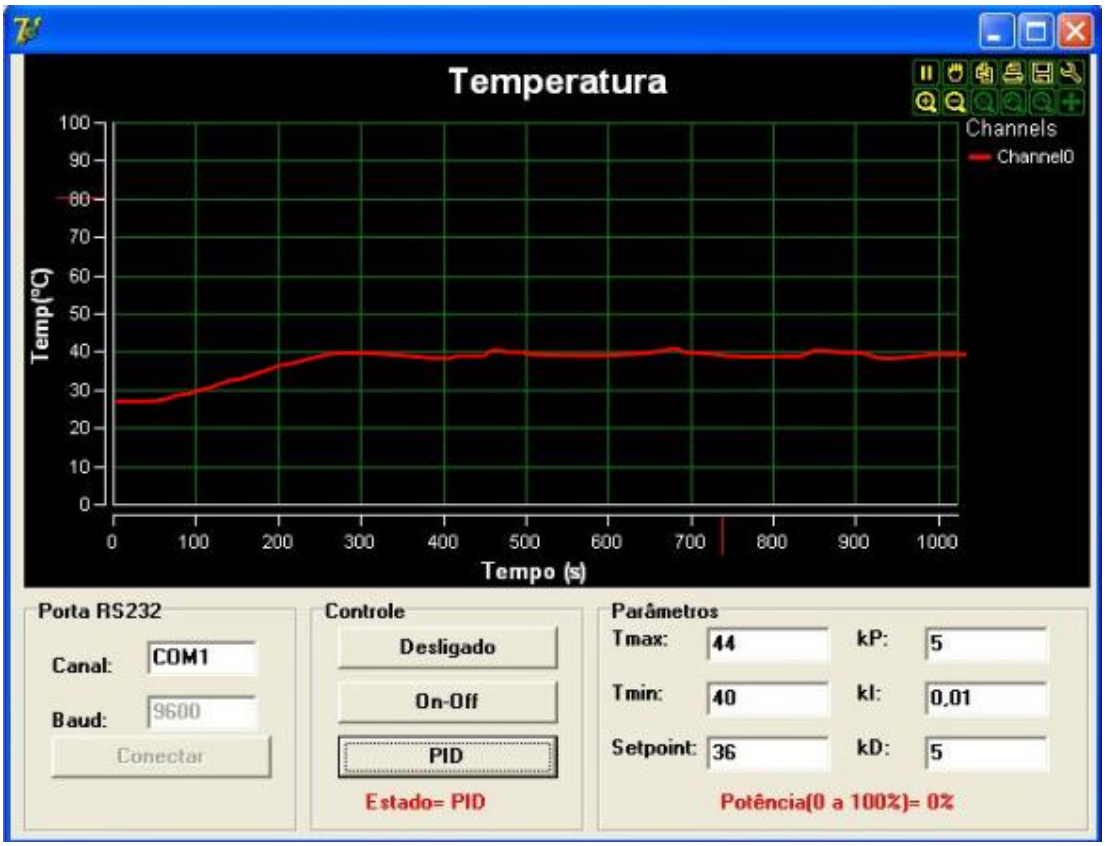
Figura 49- Figura 47- Processo com o controle PID



Fonte: (O AUTOR)

O sistema respondeu da seguinte forma: Inicialmente o sistema partiu com a temperatura real de 28°C, para atingir o setpoint 40°C levou em torno de 270 segundos, após atingir o setpoint a inércia térmica foi de menos de 1°C, após isso continuou-se o ciclo de acordo como mostra a Figura 50.

Figura 50- Resposta do Controlador PID



Fonte: (O AUTOR)

Comparando a resposta dos dois tipos de controle percebe-se que o controle PID atingiu o *setpoint* em um intervalo de tempo mais curto. E que a inércia térmica permaneceu abaixo de 1°C durante os primeiros 1000 segundos de funcionamento. Dessa forma é possível afirmar que o controle do tipo PID é vantajo em relação ao controle do tipo ON-OFF no sentido de eliminar gradativamente a variação do sistema e mantendo a temperatura sempre próximo do valor do *setpoint*.

CONCLUSÃO

Com os resultados obtidos no desenvolvimento deste projeto, chegou-se a conclusão de que aplicando o sistema de controle de temperatura do PID é possível tornar nulo ou próximo de nulo a inércia términca tradicionais do sistema de controle do tipo *ON-OFF*.

Atualmente os fabricantes de controladores disponibilizam o controlador de temperatura já com o controle PID embutido no sistema, facilitando a instalação, ajustes dos parâmetros e etc.

Para trabalhos futuros sugere-se aplicar o controle PID em mais de uma zona de aquecimento, visto que este trabalho se propôs em apenas simunar uma zona de aquecimento de um forno elétrico resistivo.

REFERÊNCIAS BIBLIOGRÁFICAS

ANTÔNIO, Marco. **Apostila de: Programação de Microcontroladores PIC usando Linguagem C.** Vitória: CEFETES, 2006. 97p. Disponível em: <http://www.pictronics.com.br/downloads/apostilas/Apostila-Pic-C.pdf>. Acesso em 29 fev. 2016.

APOSTILA DE TÉCNICAS DE PROGRAMAÇÃO E LINGUAGEM PASCAL. [S.l.: s.n.], [2008?]. 63p. Disponívem em: <http://www.cos.ufrj.br/~sergio/ApostilaPascal.pdf>. Acesso em 24 mar. 2016.

AQUISIÇÃO DE DADOS**.** Rio de Janeiro: [s.n.], [2009?]. 13p. Disponível em: <http://www.maxwell.vrac.puc-rio.br/11512/11512_6.pdf>. Acesso em 22 fev. 2016.

BARCELOS, Érica. **Linguagem de Programação Orientada a Objeto**. [S.l.: s.n.], 2012. Disponivel em: <http://www.simonsen.br/its/pdf/apostilas/base-tecnica/2/linguagem-de-programacao-1-capitulo-2-ano-informatica.pdf>. Acesso em 01 mar. 2016.

BLUM, Maria Auxiliadora. **Sensor de temperatura com o uso do amplificador operacional.** Ceará: [s.n.], [2011?]. 8p. Disponível em: <http://files.comunidades.net/mutcom/sensor_de_temperatura_com_OPAmp.pdf>. Acesso em 22 fev. 2016.

COMUNICAÇÃO SERIAL RS232**.** Rio de Janeiro: [s.n.], [2013?]. 4p. Disponível em: <http://www.cerne-tec.com.br/Artigo_07_ComunicacaoSerial.pdf>. Acesso em 10 out. 2015.

CONTROLE DE AUTOMAÇÃO INDUSTRIAL. [S.l.: s.n.], [2012?]. 130 p. Disponível em: <http://www.trajanocamargo.com.br/wpcontent/uploads/2012/05/Controle_e_Automacao_industrial_II.pdf>. Acesso em: 27 mar. 2014.

DENARDIN, Gustavo Weber. **Microcontroladores.** Rio de Janeiro: [s.n.], [2000?]. 34p. Disponível em: <http://www.joinville.udesc.br/portal/professores/eduardo_henrique/materiais/apostila_micro_do_Gustavo_Weber.pdf>. Acesso em 22 fev. 2016.

DESENVOLVENDO APLICAÇÕES ORIENTADAS A OBJETOS COM O BORLAND DELPHI. [S.l.: s.n.], [2003?]. 75p. Disponível em: <http://www2.fateb.br/ftp/apostilas/Delphi/OO-Delphi.pdf>. Acesso em: 01 mar. 2016.

FONTOURA, Martiniano Antônio. **Revolução Industrial**. Santa Catarina: [s.n.], 2010. 59 p. Disponivel em: < http://livrozilla.com/doc/1444627/2-%E2%80%93-revolu%C3%A7%C3%A3o-industrial---universidade-federal-de-santa-... >. Acesso em: 03 out. 2017.

FUNDAMENTOS DE CONTROLE CLÁSSICO. Santa Catarina: [s.n.], [2012?]. 210 p. Disponível em: <http://www.labspot.ufsc.br/~aguinald/ensino/eel7063/controle.pdf>. Acesso em: 15 mar. 2014.

GHIRARDELLO, Ariovaldo. **Apostila Sobre Modulação PWM**. [S.l.: s.n.], [2007?]. 6p. Disponível em: <http://www.eletronica.org/arq_apostilas/apostila_pwm.pdf>. Acesso em 29 fev. 2016.

GUERRA, Leonardo Ney de Araujo. **Uso de compensador PID no controle da taxa de variação de temperatura em um forno elétrico a resistência**. Rio de Janeiro: [s.n.], 2006. 52 p. Disponível em: <http://monografias.poli.ufrj.br/monografias/monopoli10000326.pdf>. Acesso em: 27 mar. 2014.

MARTINS, Luiz Gustavo Almeida. **Apostila de Linguagem C (Conceitos Básicos).** Uberlandia, UFU: [s.n.], [2005?]. 83p. Disponivel em: <http://www.facom.ufu.br/~gustavo/ED1/Apostila_Linguagem_C.pdf>. Acesso em 29 mar. 2016.

OLIVEIRA, Adalberto Luiz de Lima. **Fundamentos de Controle de Processo.** Espírito Santo: Senai, 1999. 72 p. Disponível em: < http://www.dequi.eel.usp.br/~felix/Controle.pdf >. Acesso em: 03 out. 2017.

RONALD, J. Tocci. **Sistemas Digitais: Príncipios e Aplicações.** Rio de Janeiro: Editora *Prentice Hall* Brasil, 2007 10° ed. 588p.

APÊNDICE A - SCRIPT DA LÓGICA DE CONTROLE EM C

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *  Trabalho de conclusão de curso                                    *
 *  controle de temperatura pid                                       *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

#include <16F883.h>
#device adc=10
#use delay(clock=4000000)
#fuses xt,nolvp,nowdt
#define TBIT 100
#use rs232(baud=9600, xmit=pin_c6, rcv=pin_c7)

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                    Definição de I/Os                                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#define  RS    PIN_C5    // Pino de seleção de modo do display
#define  EN    PIN_C4    // Pino de habilitação do display
#define  D7    PIN_B7    // Pino de dados 7
#define  D6    PIN_B6    // Pino de dados 6
#define  D5    PIN_B5    // Pino de dados 5
#define  D4    PIN_B4    // Pino de dados 4
#define  TX    PIN_A2    // Pino de transmissão serial
#define  LED   PIN_C3    // LED do controlador ON-OFF
#define  SINALEIRA PIN_A5// Sinaleira

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                    Registradores                                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#byte    txsta  =  0x98
#byte    rcsta  =  0x18
#define   oerr     1
#define   ferr     2
#define   cren     4
#define   txen     5

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Prototipagem de Funções                            *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
void inicializa_lcd(void);
void lcd_inst4(unsigned char dado);
void lcd_inst(unsigned char dado);
void lcd_dado(char dado);
void envia232(int dado);
void checa_estouro_serial(void);
```

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Bloco Principal de Programa                    *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

 void main(void)
 {
    setup_oscillator( OSC_4MHZ );

    //setup_adc_ports(ra0_ra1_ra3_analog);  //Configura os pinos analógicos
    setup_adc_ports(sAN0);               //Configura os pinos analógicos
    setup_adc(adc_clock_internal);       //Configura AD para clock interno
    set_adc_channel(3);                 //Seleciona o canal 3
    setup_ccp1(CCP_PWM_H_H);             //Configura PWM
    setup_timer_2 (T2_DIV_BY_1,0xFF,1);   //Configura base de tempo
    set_pwm1_duty (0);                  //Inicializa desligado
    enable_interrupts(int_rda);          //Liga a interrupção serial
    enable_interrupts(global);           //Liga a interrupção global
    inicializa_lcd();                  //Chama função de inicialização lcd

    while(1)
    {
      float temp;
      temp=read_adc();
      temp=temp*0.5;                     //Converte para temperatura
      lcd_inst(0x82);                    //Posiciona na segunda linha do display
      printf(lcd_dado,"Temp: %02.1f C   ",temp);//Mostra mensagem
      printf(envia232,"%02f\r\n",temp);//Envia mensagem
      checa_estouro_serial();       //Checa estouro de serial
      delay_ms(500);                     //Aguarda 50 ms
    }

}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *        Rotina de Verificação de Estouro de Buffer Serial          *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */

void checa_estouro_serial(void)
 {
    if (bit_test(rcsta,oerr))
    {
         char le;

         bit_clear(rcsta,cren);
         le=getc();
         le=getc();
         le=getc();
         bit_set(rcsta,cren);
    }
 }
```

```
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                Tratamento de Interrupção Serial                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
#int_rda
void Isr_Serial(void)
{
  char dado;

  if (bit_test(rcsta,ferr))
  {
    dado=getc();
    return;
  }

  dado=getc();              //Recebe dado da usart
  if(dado=='$')
  {
    dado=getc();
    if(dado=='A')
      output_high(LED);       //Liga led
    if(dado=='a')
      output_low(LED);        //Desliga led

    if(dado=='B')
    {
      output_high(SINALEIRA); //Liga sinaleira
      output_low(LED);        //Desliga led
    }
    if(dado=='b')
      output_low(SINALEIRA);  //Desliga sinaleira
  }
  else if(dado=='*')
  {
    dado=getc();
    set_pwm1_duty (dado);
  }


}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Função de envio de caracteres                  *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
void envia232(int dado)
{
  disable_interrupts(global);
  output_high(TX);                //Start bit
  delay_us(TBIT);
  output_bit(TX,!bit_test(dado,0)); //Bit 0
```

```
  delay_us(TBIT);
  output_bit(TX,!bit_test(dado,1)); //Bit 1
  delay_us(TBIT);
  output_bit(TX,!bit_test(dado,2)); //Bit 2
  delay_us(TBIT);
  output_bit(TX,!bit_test(dado,3)); //Bit 3
  delay_us(TBIT);
  output_bit(TX,!bit_test(dado,4)); //Bit 4
  delay_us(TBIT);
  output_bit(TX,!bit_test(dado,5)); //Bit 5
  delay_us(TBIT);
  output_bit(TX,!bit_test(dado,6)); //Bit 6
  delay_us(TBIT);
  output_bit(TX,!bit_test(dado,7)); //Bit 7
  delay_us(TBIT);
  output_low(TX);               //Stop bit
  enable_interrupts(global);
  delay_us(500);
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                  Rotina de Inicialização de LCD                *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
void inicializa_lcd(void)
{

  lcd_inst4(0b00000011);
  delay_ms(15);
  lcd_inst4(0b00000011);
  delay_ms(15);
  lcd_inst4(0b00000011);
  delay_ms(15);
  lcd_inst4(0b00000010);
  delay_ms(15);

  lcd_inst(0b00101000);
  delay_ms(15);
  lcd_inst(0b00001100);
  delay_ms(15);
  lcd_inst(0b00000110);
  delay_ms(15);
  lcd_inst(0b00000001);
  delay_ms(100);
}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *               Rotina de Envio de Comando para o LCD            *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
void lcd_inst4(unsigned char dado)
{
```

```
  output_low(RS);                       //Limpa o pino RS, modo comando
  delay_us(100);

  if (dado & 0b00001000) output_high(D7);       //Se o bit estiver em 1 liga a saída
  else              output_low(D7);       //Caso contrário desliga a linha

  if (dado & 0b00000100) output_high(D6);       //Se o bit estiver em 1 liga a saída
  else              output_low(D6);       //Caso contrário desliga a linha

  if (dado & 0b00000010) output_high(D5);       //Se o bit estiver em 1 liga a saída
  else              output_low(D5);       //Caso contrário desliga a linha

  if (dado & 0b00000001) output_high(D4);       //Se o bit estiver em 1 liga a saída
  else              output_low(D4);       //Caso contrário desliga a linha

  delay_us(100);

  output_high(EN);                      //Gera pulso de enable
  delay_us(150);                        //Aguarda 150 us
  output_low(EN);

  delay_ms(5);                          //Faz a inicialização do display

}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *              Rotina de Envio de Comando para o LCD            *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
void lcd_inst(unsigned char dado)
{
  output_low(RS);                       //Limpa o pino RS, modo comando
  delay_us(100);

  if (dado & 0b10000000) output_high(D7);       //Se o bit estiver em 1 liga a saída
  else              output_low(D7);       //Caso contrário desliga a linha

  if (dado & 0b01000000) output_high(D6);       //Se o bit estiver em 1 liga a saída
  else              output_low(D6);       //Caso contrário desliga a linha

  if (dado & 0b00100000) output_high(D5);       //Se o bit estiver em 1 liga a saída
  else              output_low(D5);       //Caso contrário desliga a linha

  if (dado & 0b00010000) output_high(D4);       //Se o bit estiver em 1 liga a saída
  else              output_low(D4);       //Caso contrário desliga a linha

  delay_us(100);

  output_high(EN);                      //Gera pulso de enable
  delay_us(1);                          //Aguarda 150 us
```

```
  output_low(EN);

  if (dado & 0b00001000) output_high(D7);       //Se o bit estiver em 1 liga a saída
  else                   output_low(D7);        //Caso contrário desliga a linha

  if (dado & 0b00000100) output_high(D6);       //Se o bit estiver em 1 liga a saída
  else                   output_low(D6);        //Caso contrário desliga a linha

  if (dado & 0b00000010) output_high(D5);       //Se o bit estiver em 1 liga a saída
  else                   output_low(D5);        //Caso contrário desliga a linha

  if (dado & 0b00000001) output_high(D4);       //Se o bit estiver em 1 liga a saída
  else                   output_low(D4);        //Caso contrário desliga a linha


  delay_us(10);

  output_high(EN);                      //Gera pulso de enable
  delay_us(150);                        //Aguarda 150 us
  output_low(EN);

  delay_ms(5);                          //Faz a inicialização do display

}

/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *              Rotina de Envio de Dados para o LCD              *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
void lcd_Dado(char dado)
{

  output_high(RS);                      //Limpa o pino RS, modo comando
  delay_us(100);

  if (dado & 0b10000000) output_high(D7);       //Se o bit estiver em 1 liga a saída
  else                   output_low(D7);        //Caso contrário desliga a linha

  if (dado & 0b01000000) output_high(D6);       //Se o bit estiver em 1 liga a saída
  else                   output_low(D6);        //Caso contrário desliga a linha

  if (dado & 0b00100000) output_high(D5);       //Se o bit estiver em 1 liga a saída
  else                   output_low(D5);        //Caso contrário desliga a linha

  if (dado & 0b00010000) output_high(D4);       //Se o bit estiver em 1 liga a saída
  else                   output_low(D4);        //Caso contrário desliga a linha

  delay_us(10);

  output_high(EN);                      //Gera pulso de enable
  delay_us(1);                          //Aguarda 150 us
```

```
  output_low(EN);

  if (dado & 0b00001000) output_high(D7);       //Se o bit estiver em 1 liga a saída
  else                   output_low(D7);        //Caso contrário desliga a linha

  if (dado & 0b00000100) output_high(D6);       //Se o bit estiver em 1 liga a saída
  else                   output_low(D6);        //Caso contrário desliga a linha

  if (dado & 0b00000010) output_high(D5);       //Se o bit estiver em 1 liga a saída
  else                   output_low(D5);        //Caso contrário desliga a linha

  if (dado & 0b00000001) output_high(D4);       //Se o bit estiver em 1 liga a saída
  else                   output_low(D4);        //Caso contrário desliga a linha


  delay_us(10);

  output_high(EN);                      //Gera pulso de enable
  delay_us(150);                       //Aguarda 150 us
  output_low(EN);Figura 41
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
 *                    Fim do programa                                        *
 * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
```