FUNDAÇÃO DE ENSINO "EURÍPEDES SOARES DA ROCHA "CENTRO UNIVERSITÁRIO EURÍPEDES DE MARÍLIA – UNIVEM CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

APLICAÇÃO DO CONTROLE PROPORCIONAL INTEGRAL E DERIVATIVO EM UM PROTÓTIPO DE CONTROLE DE MÃO MECÂNICA UTILIZANDO MICROCONTROLADORES

ANDERSON ZAPATERRA BATILANI

MARÍLIA

2012

FUNDAÇÃO DE ENSINO "EURÍPEDES SOARES DA ROCHA "CENTRO UNIVERSITÁRIO EURÍPEDES DE MARÍLIA – UNIVEM CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

APLICAÇÃO DO CONTROLE PROPORCIONAL INTEGRAL E DERIVATIVO EM UM PROTÓTIPO DE CONTROLE DE MÃO MECÂNICA UTILIZANDO MICROCONTROLADORES

Trabalho de Curso apresentado ao Curso de Bacharelado em Ciência da Computação da Fundação de Ensino "Eurípedes Soares da Rocha", mantenedora do Centro Universitário Eurípedes de Marília - UNIVEM, como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientador
Prof. César Giacomini Penteado

MARÍLIA
2012

CENTRO UNIVERSITÁRIO EURÍPIDES DE MARÍLIA
BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

TRABALHO DE CONCLUSÃO DE CURSO – AVALIAÇÃO FINAL

Anderson Zapaterra Batilani

Aplicação do Controle Proporcional Integral e Derivativo em um Protótipo Utilizando Microcomtroladores

Banca examinadora da monografia apresentada ao Curso de Bacharelado em Ciência da Computação do UNIVEM/F.E.E.S.R., para obtenção do Título de Bacharel em Ciência da Computação.

Nota: 8,5 (oito e meio)

Orientador: César Giacomini Penteado

1°. Examinador: Ildeberto de Gênova Bugatti

2°. Examinador: Fábio Dacêncio Pereira

Marília, 04 de dezembro de 2012.

RESUMO

No meio industrial, muitas máquinas necessitam ser controladas automaticamente de modo que seu desempenho atenda os requisitos de produção de forma que consiga produzir produtos com qualidade, velocidade e com precisão. Para que seja possível obter um nível de automação alto é preciso implementar métodos de controle que necessitam de algoritmo e equações matemáticas complexas.

O projeto tem como foco principal o desenvolvimento de algoritmos de controle e sua implementação sobre um motor DC simples. A forma de controle que será implementado é o controlador proporcional, integral e derivativo ou controle PID que tem a função de estabilizar a ação de um dispositivo eliminando erros de estado.

Para a implementação do algoritmo PID, será utilizado à linguagem Java para simular o estado do controlador PID e o programa PROTEUS 7 que será utilizado para criar uma esquema eletrônico no objetivo de simular o algoritmo PID sobre um motor DC.

Com as simulações realizadas o algoritmo PID será inserido em um microcontrolador PIC 18F542 onde suas portas de entrada e saída que são gerenciado pelo kit PICGenios com o propósito de controlar o motor DC e colher dados do sensor de movimento que no caso será um potenciômetro.

No final deste projeto é possível entender os conceitos de controle PID e sua função como meio de controle com alta precisão.

Palavra-Chave: controlador PID, Algoritmo PID, Simulação do algoritmo PID.

LISTA DE ILUSTRAÇÂO DE FIGURAS

LISTA DE ILUSTRAÇÃO DE TABELAS

LISTA DE ILUSTRAÇÃO DE EQUAÇÕES

SUMÁRIO

1 - INTRODUÇÃO

A utilização de algoritmos de controle vem se aprimorando através dos tempos. Sua função é controlar dispositivos que necessitam de certa precisão para cumprir seu objetivo. Para que tal precisão seja alcançada geralmente é preciso utilizar cálculos matemáticos complexos que conseguem descrever e preveja situações adversas.

Os algoritmos de controle estão presentes em vários dispositivos eletrônicos como em ar condicionados, geladeiras e fornos modernos, regulando a sua temperatura, controlando a agulha de leitura de um HD, e nas indústrias de construção de veículos controlando robôs no objetivo de posicionar vários parafusos com precisão milimétrica.

Para se construir dispositivos eletrônicos com o fim de obter grande precisão não basta ter um bom algoritmo de controle, mas também é preciso ter sensores que obtenham o estado do dispositivo que será controlado. Os sensores têm um papel fundamental no controle de máquinas. Sua função é obter o valor de uma variável como posição, temperatura ou pressão e transformá-lo em um valor digital.

Com os grandes avanços em pesquisas voltados para os diferentes tipos de algoritmos de controle uma delas vem se destacando possuindo um grande grau de controle sobre dispositivos eletrônicos, o controlador PID. Sua estrutura é composta pelas formulas matemáticas sendo a proporcional a integral e a derivativo que juntas buscam o máximo de velocidade e despenho de um dispositivo de forma a eliminar todos os erros de estado como erro de posição, temperatura pressão e entre outros.

Com o auxilio da computação gráfica, algoritmos que faz a analise e tratamento da imagem de câmeras, estão sendo desenvolvidos para controlar veículos, câmera de vigilância, e podem futuramente controlar máquinas informando seu estado de posição e localização.

Um dos projetos que estão em desenvolvimento utilizando controle por visão computacional é o projeto CaRINA(Carro Robótico Inteligente para Navegação Autônoma) desenvolvido na USP de São Carlos. O projeto tem o objetivo de controlar um veiculo utilizando uma câmera que captura dados do ambiente fazendo sua interpretação. Deste modo é possível analisar a imagem como verificar se há pessoas na rua, se o carro a sua frente esta em movimento ou parado e até colher dados de um semáforo.

O projeto apresentado neste trabalho tem como foco principal mostrar os meios de controles modernos de um dispositivo utilizando algoritmo PID. Desta forma serão

mostrados os meios para implementar algoritmo de controle PID com a linguagem PIC BASIC. Para tanto serão utilizados um motor DC e um kit de desenvolvimento de microcontroladores para testar o algoritmo.

1.1 - OBJETIVO

O objetivo principal deste projeto é mostrar as formas de implementação de sistemas de controles modernos utilizando um microcontrolador e um protótipo de mão mecânica. Desta forma o projeto tem como foco a construção do algoritmo de controle PID, e suas possíveis simulações utilizando software Proteus 7 e uma implementação simulada utilizando programação Java. É mostrado também as formas de implementação física do algoritmo utilizando o kit PICGenios usando o microcontrolador PIC18F452 e um motor DC simples.

1.2 - ORGANIZAÇÃO DO TRABALHO

O trabalho está organizado em sete sessões visando demonstrar todos os passos para se construir um dispositivo com um nível de automação.

O segundo capítulo aborda os conceitos de Sistemas de Controle Modernos, descrevendo as suas formas de atuação sobre dispositivos eletrônicos. E também será apresentado o conceito de Controladores Proporcionais, Integral e Derivativos com o objetivo de explicar suas funções e suas respectivas fórmulas.

No terceiro capítulo, introdução linguagem BASIC para PIC, é abordado de maneira simples a estrutura da linguagem PIC BASIC, bem como seus aspectos mais importantes ajudando assim na compreensão do algoritmo PID.

O quarto capítulo sobre o algoritmo PID, é abordado uma parte da sua estrutura, que é a tradução das fórmulas matemáticas proporcional, integral e derivativa para a linguagem PIC BASIC. Neste capítulo são comentadas todas as linhas do código explicando de forma detalhada o seu funcionamento.

No quinto capítulo sobre as formas de implementação são abordados os meios para simular o algoritmo PID. Neste capítulo é apresentado o esquema de um circuito

eletrônico contendo motor, microcontrolador, sensor de movimento e o oscilador para a geração do clock. Todos os componentes do circuito serão documentados.

O sexto capítulo apresenta os testes físicos, onde são descrita a forma de implementação física mostrando os métodos de inserir o algoritmo PID no kit PICGenios PIC18F com a utilização do programa WinPIC800 fornecido pela MICROCHIP, e a construção do esquema físico.

Para concluir, o sétimo capítulo é mostrado as conclusões e enfatizando os pontos fortes e fracos deste projeto.

2 - SISTEMAS DE CONTROLE MODERNOS

Com o advento dos controladores modernos, foi possível ter grandes avanços no meio industrial favorecendo no crescimento do número de produtos fabricados com maior qualidade, superando a produção utilizando a mão de obra humana. Muitos processos que eram repetitivos e realizados por mão de obra humana, como por exemplo, tampar a pasta de dente, que não necessitavam de funcionários especializados, ou como manusear produtos de alto teor químico.

Com os novos sistemas de controladores modernos é possível controlar vários processos onde sua presença pode ser notada nas empresas de alimento controlando todo o processo de manufatura incluindo caldeiras de pressão e temperatura, em mísseis de longo alcance controlando sua posição referente às coordenadas que foram passadas e até mesmo controlando a temperatura do ar condicionado.

Muitas empresas vêm desfrutando dos benefícios desta tecnologia aumentando a produtividade e qualidade de seus produtos. Por serem máquinas automáticas seu desempenho e sua velocidade não podem ser comparado a de um ser humano ocorrendo à desvalorização da mão de obra não especializada.

Para que se possa construir e manipular sistemas de controle é preciso ter uma boa base de conceitos de eletrônica, matemática, programação e o entendimento teórico de sistema de controle modernos. Isso se deve ao fato que cada dispositivo de controle necessita de vários componentes para seu funcionamento sendo eles componentes físicos como o próprio dispositivo eletrônico, formula matemáticas que descrevem o estado do

dispositivo e algoritmos que fazem o relacionamento entre o dispositivo eletrônico e matemática.

Este capítulo mostra as formas de sistemas de controle e sua possível utilização dando uma maior visão sobre o assunto. Aborda também o método de controle Proporcional Integral e Derivativo muito utilizado em sistemas de controle de alta precisão.

2.1 - SISTEMAS DE CONTROLE EM MALHA-FECHADA

O sistema de controle em malha-fechada tem o objetivo de controlar o estado de dispositivos. Este sistema de controle é também chamado de sistema realimentado, pois o controlador esta sempre analisando a saída do dispositivo que esta sendo controlado. Desta forma para cada leitura da saída do dispositivo o controlador dará uma resposta de entrada para o dispositivo controlado, evitando ou prevendo erros atuais ou futuros.

Para ilustrar o conceito de controle em malha fechada, considere um motorista em uma estrada onde a velocidade permitida é de 80 km/h. Os dados de saída do carro (Dispositivo) seria o velocímetro, enquanto o controlador do velocímetro seria o motorista deste carro. O carro ao exceder a velocidade em uma decida informa no velocímetro a velocidade de 97 km/h. Ao analisar os dados do velocímetro o motorista na função de controlador tem a ação de pisar o pé no freio até que a velocidade atinja seus 80km/h, podemos dizer que a saída do controlador seria o acionamento do freio para ajustando a velocidade.

Geralmente, para que seja possível a implementação desse processo, é preciso de algoritmos de controle que consigam interpretar o estado do dispositivo que esta sendo controlado. Estes algoritmos na grande maioria são constituídos de funções contendo bases matemáticas. Como foi ilustrado acima podemos relacionar o cérebro do motorista como sendo o algoritmo em execução a cada momento que este verifica o valor do velocímetro.

2.2 - SISTEMA DE CONTROLE EM MALHA-ABERTA

Pode-se dizer que no controle de malha-aberta a saída do dispositivo controlado não tem influência sobre o controlador, ou seja, se houver algum distúrbio no meio o controlador não faz o regulamento de sua saída. A cada entrada do controlador ha uma resposta fixa para uma determinada ação, ou seja, a precisão do sistema sempre dependerá de uma calibração.

Um exemplo prático seria a máquina de lavar roupa com o sistema em malha-aberta, suas ações são baseadas em um determinado tempo, onde cada tempo é bem definido podendo ser dividido em molhar, enxaguar e lavar. A máquina não mede os sinais de entrada, pois ao colocar uma só peça de roupa a máquina ira inserir a mesma quantidade de água que daria para lavar 30 peças de roupa.

2.3 - DIFERENÇAS ENTRE SISTEMA DE CONTROLE EM MALHA-FECHADA E EM MALHA-ABERTA.

Os dois tipos de controles têm suas particularidades onde é possível destacar pontos fortes e fracos. Uma das vantagens de se utilizar o sistema em malha-aberta seria a fácil implementação e baixo custo, pois não envolvem algoritmos complexos de controle, tendo como efeito um menor tempo de desenvolvimento e peças como sensores adicionais podem ser descartados para estes sistemas. Sua utilização abrange aparelhos que não necessitam corrigir distúrbios no meio controlado e sim seguir uma rotina preestabelecida pelo controlador.

Para implementar um sistema de controle em malha-fechada é necessário a utilização de algoritmos complexos que utilizam formulas matemáticas que tratam o estado do dispositivo controlado e sensores que captam o estado do dispositivo. Desta forma o custo para desenvolver este sistema pose ser mais alto, pelo fato da sua complexidade e tempo dedicado. Porém, para aparelhos que necessitam buscar alto desempenho e precisão, onde seu meio de atuação ocorre distúrbios no estado do controle, é necessário que o sistema seja em malha-fechada.

Utilizar um dos meios de controle define o tipo de ambiente que este aparelho atua. O desempenho de atuação do dispositivo é definido pelo tipo de algoritmo que o controlador tem.

2.4 - SISTEMAS DE CONTROLE NUMÉRICO

Os sistemas de controle numérico são utilizados para interpretar um estado físico como pressão, velocidade, temperatura e movimento e transformá-lo em um dado numérico. A conversão do estado físico para uma representação numérica é possível por meio de conversores digital - analógicos.

Os conversores digitais - analógicos tem um papel fundamental na interpretação de estados físicos. Seu funcionamento segue a seguinte linha de raciocínio: para cada grandeza física de um determinado estado pode ser relacionado um nível elétrico e, a cada nível elétrico, pode ser interpretado com um valor binário.

As informações binárias são convertidas em variáveis de estado, que são analisadas pelo controlador que gera uma ação em resposta a esta variável, podendo ser um nível de sinal elétrico.

2.5 - CONTROLADOR PID (PROPORCIONAL, INTEGRAL E DERIVATIVO)

O controlador Proporcional, Integral e Derivativo com a sigla PID é utilizado para controlar dispositivos onde é requisitado níveis de precisão adequados. Seu objetivo é explorar o máximo desempenho possível do dispositivo na busca de obter o menor erro admissível.

Na década de 50 as máquinas industriais eram controladas por pessoas que regulavam seu estado de forma manual. Um operador de caldeira tinha a função de regular a temperatura da caldeira de forma que a mesma não explodisse. A figura 1 ilustra um operador de caldeira regulando a saída de gás de uma caldeira.

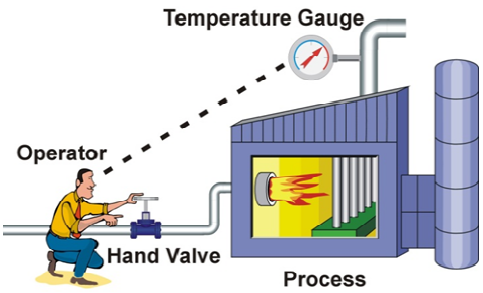

Figura 1 - Operador de Caldeira

O operador da caldeira para controlar a sua temperatura sempre esta analisando o termômetro da caldeira. Caso a temperatura da caldeira aumente o operador tem a ação de fazer diminuir a vazão de gás, se a temperatura da caldeira abaixar o operador tem a ação de abrir a vazão de gás. Este processo é feito repetidamente pelo operador da caldeira, de forma que ao longo do processo o operador poderia se confundir, ocorrendo erros no processo de produção.

Com os surgimento dos microcontroladores e avanços na tecnologia, foi criado o primeiro método de controle o ON-OFF (Liga Desliga) que é utilizado até hoje. Podemos citar o ar-condicionado que lê a temperatura ambiente, caso a temperatura não esteja satisfatória o ar-condicionado é ligado; caso contrário, ocorre o desligamento do mesmo. Este modo de controle se mostra muito ineficaz para ambientes que necessitam de um estado constante, pois o método ON-OFF só atua no sistema quando seu estado mudar.

Para sanar os problemas ON-OFF foram desenvolvidos métodos para aumentar a precisão dos dispositivos utilizando os controladores proporcional, integral e derivativo. Na figura 2 é ilustrado o controle PID que controla a temperatura de uma caldeira industrial.

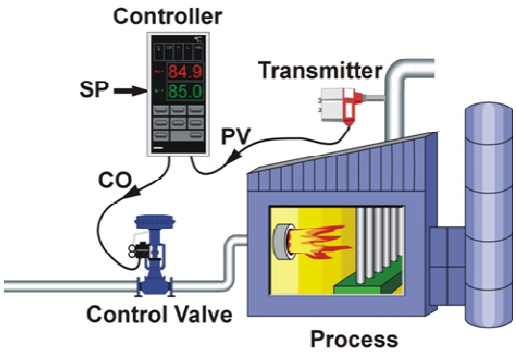

Figura 2 - Controle de caldeira utilizando controlador PID

Com o grande avanço da tecnologia os operadores de maquinas são substituídos por controladores eletrônicos que tem a mesma função de um operador humano que é controlar o estado da máquina. Na figura 2 é ilustrado um controlador PID, sua função é controlar a temperatura da caldeira utilizando os dados do termômetro, desta forma o controle abre ou fecha a válvula de gás à medida que a temperatura se desequilibra.

A implementação do controle PID, ocorre inserindo sobre um microcontrolador o algoritmo contendo a formula PID. É utilizado quando o estado de um processo não pode

variar, como por exemplo, em fornos elétricos, caldeiras de pressão, esteiras e muitos outros equipamentos.

Para demonstrar a forma como o PID atua em um controle em malha-fechada é possível ver a sua forma de atuação com o diagrama da figura 3.

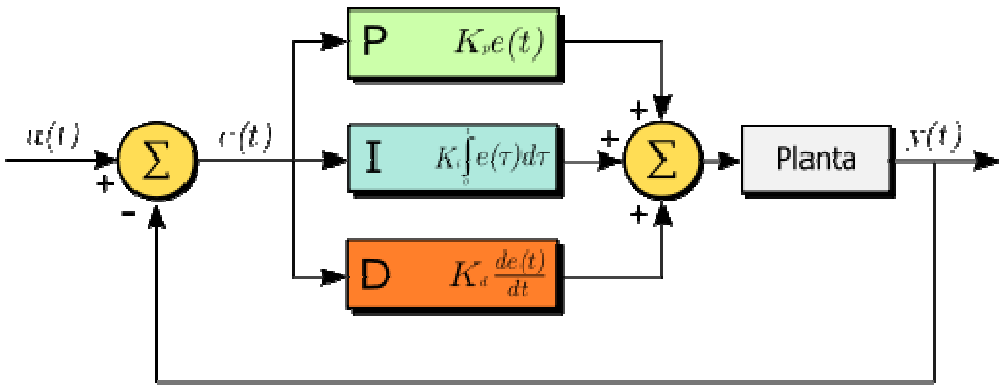

Figura 3 - Diagrama em bloco do controle PID. [1]

A sua saída é a soma de três resultados diferentes, sendo ela a ação proporcional, integral e derivativa. A formula abaixo representa o diagrama PID.

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)d\tau + K_d \frac{de}{dt}$$

Equação 1 - Formula PID (Brom, 2012)

Quanto maior for o valor da constante K_p mais rápido é a chegado no ponto desejado porem irá ocasionar uma maior oscilação no sistema.

O valor da constante K_i tem que ser maior que 0.01 e menor que 1. Caso o valor seja maior que 1 ocorrera instabilidade no sistema.

O valor de constante K_d define a estabilidade do sistema, quanto maior seu valor mais estável é o sistema, porem valores muito elevados podem desestabilizar o sistema.

A ação proporcional busca a aceleração do processo de forma controlada enquanto ação integral tenta corrigir as oscilações geradas pela a ação proporcional. Já ação derivativa é utilizada para acelerar e estabilizar o processo.

2.5.1 - CONTROLE PROPORCIONAL

Proporcional no dicionário refere-se:

> "*Que tem proporção; que está em proporção; simétrico; regular.Matemática Relativo a uma proporção.*" [2]

O controle proporcional é o método mais simples para controlar um sistema, busca uma aceleração de forma controlada. Seu objetivo é chegar em um determinado estado (SetPoint) com a máxima velocidade tentando corrigir os erros de desvios.

A correção do desvio é dito proporcional ao erro, pois é onde eu tenho a diferença entre o valor ideal e o valor atual da variável controlada.

Na ação proporcional tem-se:

$$u(t) = K_p\, e(t)$$

Equação 2 - Formula Proporcional (Brom, 2012)

Para o entendimento desta ação proporcional ficaria da seguinte forma:

$$Proporcional = K_p * (SetPoint - EstadoAtual)$$

Equação 3 - Formula Proporcional

Quando Kp for maior que zero maior será a velocidade de chegada ao SetPoint porem menor será a sua precisão. O valor de Kp é referente à constante proporcional, seu valor tem a seguinte função:

- Quanto menor o valor da constante proporcional Kp maior será a precisão de chegada ao setPoint, porem, maior será o tempo de chegada no ponto do setPoint, deste modo, o sistema fica com uma alta precisão com o inconveniente da operação do sistema ficar lenta.

- Quanto maior o valor da constante proporcional Kp menor será a precisão de chegada ao setPoint, porem, menor será o tempo de chegada no ponto de setPoint, deste modo, a chegada no setPoint é mais rápida porem o sistema fica com uma menor precisão de chegada ao setPoint podendo ultrapassar o ponto desejado ocorrendo oscilações.

Já o valor do SetPoint e EstadoAtual refere-se respectivamente ao valor do estado a onde se quer chegar e o estado do dispositivo no momento.

2.5.2 - CONTROLE PROPORCIONAL E INTEGRAL

O controlador Proporcional Integral (PI) tem a função de completar a função proporcional tentando fazer com que a saída fique perto do estado desejado removendo as oscilações indesejadas causadas pela ação proporcional.

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau$$

Equação 4 - Formula Proporcional e Integral (Brom, 2012)

O controle PI pode ser descrita da seguinte maneira considerando a equação 3:

$$PI = Proporcional + (IntegralAnterior + (SetPoint - EstadoAtual)) * (kI/100)$$

Equação 5 - Formula Proporcional e Integral

O valor da constante K_i define a quantidade de erro absoluto que vai ser eliminado, valores maiores que 100% tentam eliminar um erro que não existe ocasionando instabilidade no sistema.

2.5.3 - CONTROLE PROPORCIONAL INTEGRAL E DERIVATIVO

O controlador PID proporcional integral e derivativo reúne as vantagens das ações Proporciona, Integral e Derivativo. O controle PID pode ser representado como se segue:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau) d\tau + K_d \frac{de}{dt}$$

Equação 6 - Formula Proporcional Integral e Derivativa (Brom, 2012)

Esta equação pode ser simplificada da seguinte maneira considerando as equações 3 e 5:

$$PID = (Proporcional + Integral + (espacoPercorridoNoInstanteTempo * (Kd/100)))$$

Equação 7 - Formula Proporcional Integral e Derivativa

A ação derivativa tem a função de estabilizar todo o sistema, desta maneira o processo fique ágil. O valor da constate K_d ou constante derivativa influencia na estabilidade do sistema, quanto maior seu valor maior é a estabilidade do sistema, valores muito altos para esta constante pode fazer com que o sistema fique instável.

3 – INTRODUÇÃO A LINGUAGEM PIC BASIC

A linguagem PIC BASIC foi desenvolvida para auxiliar a programação de microcontroladores da MICROCHIP. Seu objetivo é oferecer funções que auxilia a programação de microcontroladores.

Muitos projetistas de circuitos adotam o PIC BASIC por ser uma linguagem simbólica de fácil entendimento. É uma linguagem de fácil manuseio quando comparada a linguagem assembly. Isso se deve ao fato de que cada função do PIC BASIC encapsula um bloco de código em ASSEMBLY. Desta forma o programador não precisa ter muito conhecimento sobre assembly, porém para manipular com precisão um microcontrolador o básico da linguagem assembly precisa ser assimilada.

Neste projeto foi utilizada a linguagem PIC BASIC pelo fato de aparentar facilidade de programação. Outras linguagens poderiam ser adotadas como linguagem C a qual apresenta flexibilidade programação.

Esta seção é dedicada à programação PIC BASIC com o objetivo de documentar suas funções básicas para o entendimento do projeto, tais como a utilização de variáveis, rotinas e loops.

3.1 - VARIÁVEISDE I/O (ENTRADA E SAÍDA)

As variáveis de entrada e saída do PIC BASIC são utilizadas para enviar ou colher dados do meio externo. Para cada porta do microcontrolador a uma variável estática correspondente para o PIC BASIC. Na figura 4 é possível ver á estrutura de um microcontrolador PIC 16F877 com suas portas que vai ser utilizado nas simulações utilizando o programa Proteus 7.

As portas de entrada e saída do microcontrolador têm a função de enviar e receber dados externos de modo que estes dados sejam tratados. Para se pegar e enviar dados para as portas do microcontrolador é preciso declarar essas portas de forma correta na programação PIC BASIC.

Algumas das portas do microcontrolador serão utilizadas para receber dados do potenciômetro e enviar dados de controle para o motor DC.

Na figura 4 referente ao retângulo vermelho pode-se visualizar 8(oito) portas, onde cada pino tem o seu nome e o seu número, RB0, RB1, RB2, RB3, RB4, RB5, RB6, RB7.

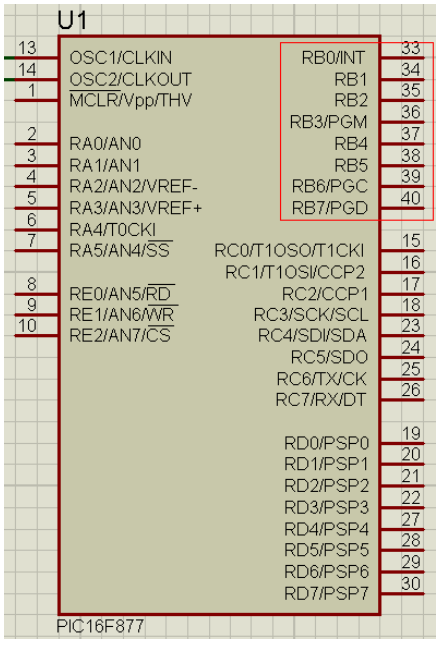

Figura 4 - Microcontrolador PIC 16f877

Para referenciar estas portas no PIC BASIC é necessário saber o nome da porta e o seu número.

Exemplo:

portb.0 refere-se a porta RB0.
portb.1 refere-se a porta RB1.
portb.2 refere-se a porta RB2.

Para porta do tipo RD0 para se referenciar esta porta em PIC BASIC é preciso declarar **portd.0.**

3.2 - DECLARAÇÃO DE VARIÁVEIS

As variáveis para microcontroladores são de três tipos, bit, bytes e Word onde cada uma delas comporta um valor máximo de bits.

O tipo bit só pode conter os valores 0 e 1
O tipo byte é composto por 8 bits que só pode conter valores entre 0 a 255
O tipo Word é composto por 16 bits que pode conter valores entre 0 a 65535

Abaixo podemos ver exemplo de declaração de uma variável em PIC BASIC:

```
NomeVarBit VAR Bit    ;declarando variável do tipo Bit

NomeVaByter VAR Byte    ;declarando variável do tipo Byte

NomeVarWord VAR Word    ;declarando variável do tipo Word
```

Primeiramente se coloca o nome da variável, em seguida declara se esse nome será uma variável e logo depois é informado o tipo desta variável podendo ser bit, byte ou Word.

Os valores que estas variáveis podem receber são valores em hexadecimal, decimal, octal ou binário.

3.3 - CONSTANTES

Na linguagem PIC BASIC é possível declarar variáveis constantes, para isso é preciso utilizar a diretiva CON. O objetivo de termos uma variável constante é o fato de não podemos alterar seu valor. Sua sintaxe é bem simples:

NomeConstante CON valor

3.4 - DEFINE

A diretiva define é utilizada para manipulação de variáveis relacionadas ao microcontrolador. Sua função é criar variáveis estáticas com valores definidos.Um exemplo prático do uso desta diretiva define é a definição do valor da freqüência do microcontrolador utilizando a variável Osc.

Define Osc 20 **;**Define o oscilador para o cristal de 20Mhz.

3.5 - OPERADORES ARITMÉTICOS

Os operadores aritméticos são essenciais para a programação em todas as linguagens. No PIC BAISC segue a mesma simbologia para as operações aritméticas como adição (+), subtração(-), multiplicação(*), divisão(/) e entre outros.

Na tabela 1, uma lista completa de todos os operadores predefinidos na linguagem PIC BASIC.

Tabela 1 - Símbolos de operadores aritmético

Operador	Descrição	Operador	Descrição
+	Soma	ABS	Valor Absoluto
-	Subtração	SIN	Seno do ângulo
*	Multiplicação	COS	Cosseno do ângulo
/	Divisão	MIN	Mínimo de um número
//	Sobra	MAX	Máximo de um número
<<	Desloca bit à esquerda	RER	Inversor de um bit
>>	Desloca bit à direita	DIG	Valor de um dígito para um número decimal
=	Igual		

3.6 - OPERADORES BINÁRIOS

Os operadores binários são utilizados para o tratamento de bits através de seus operadores AND, NOR, OR e entre outros.

A tabela 2 mostra todos os operadores lógicos do PIC BAISC.

Tabela 2 - Operadores binários

Operador	Descrição
&	AND Lógico
|	OR Lógico
^	XOR Lógico
~	NOT Lógico
&/	NAND Lógico
|/	NOR Lógico
^/	NXOR Lógico

3.7 - ESTRUTURA DE CONTROLE

As estruturas de controle são utilizadas em operações lógicas, e são essenciais para avaliar condições e estado de uma variável. O objetivo de se ter uma estrutura de controle é avaliar um estado e mediante a este estado ocorrer uma determinada ação.

Exemplo:

```
IF Condição THEN
Código...
ELSE
Código...
ENDIF
```

3.8 - WHILE

A função da estrutura de loop while é repetir um determinado trecho de código até encontrar uma condição falsa.

Segue a sintaxe do WHILE

```
WHILE Condição
Código....
WEND
```

3.9 - FOR

A função da estrutura de repetição for é baseada em uma condição de uma variável e um contador. Quando esta variável excede a condição estabelecida a repetição é finalizada.

Sintaxe:

```
FOR x=1 TO 5
Código...
NEXT
```

No código acima ocorrera 5 loops, quando x for igual a 6, a condição não será satisfeita e o loop finalizará.

3.10 - SUB-ROTINAS

As sub-rotinas são responsáveis por executar trechos de códigos chamados pela palavra reservada GOTO. As sub-rotinas são diferentes de variáveis, pois não guardam nenhuma informação. São muito utilizadas pois com ela é possível cria funções. Um exemplo simples de rotina pode ser visto abaixo:

```
InicioRotina:              ;Sub-Rotina
Código;
GOTO InicioRotina          ;GOTO inicia a sub-rotina InicioRotina
```

4 - ALGORITMO PID

Para que seja possível implementar o algoritmo de controle PID é preciso entender o uso de alguns conceitos básicos sobre as funções Integral, diferencial e proporcional.

A partir dos conceitos apresentados na seção 2 ‘Sistemas de controle modernos’, é possível implementar um algoritmo PID. Segue o exemplo deste algoritmo:

```
1   erro = setPoint - PosicaoAtual;
2
3   Proporcional = (kP*erro);
4   Integral = (Integral + (erro/divErro)) * kI;
5   Derivativa = ((erro-erro_ anterior)/divErro) * kD;
6
7 erro_anterior = erro;
8
9  PID = Proporcional + Integral + Derivativa;
```

Para que seja possível entender o código acima é preciso entender o que cada variável representa e como são obtidos alguns dados.

Linha 1:

A variável erro obtém a diferença entre a posição de chegada ou setPoint e a posição atual. Lembrando que a posição atual é obtida na saída do componente ADC (Conversor Digital Analógico).

Linha 3:

Nesta Linha a variável Proporcional recebe o valor de kP multiplicado pelo erro. O valor de kP é a representação da constante proporcional e seu valor é utilizado em todo o processo. Quanto maior o valor de kP menor é a precisão de chegada no setPoint, porém, a precisão da chegada no setPoint será afetada, ocorrendo a ultrapassagem no ponto de setPoint acarretando em oscilações. Caso o valor de kP tenda a zero menor é a velocidade de chegada no setPoint, tendo desta maneira uma maior precisão de chegada ao setPoint.

Linha 4:

Nesta linha a variável Integral receber a divisão do erro absoluto por uma variável constante divErro somando a sua integral anterior e multiplicando a sua constante integral.

O erro absoluto é dividido por uma constante divErro com objetivo de deixar o erro menor. Desta forma a chance de se ultrapassar o setPoint é minimizada, pois, quanto maior for meu erro absoluto, mais rápido será a chegada na posição de setPoint ocasionando uma possível ultrapassagem. O valor de divErro é dado pelo tempo de resposta do sistema.

A constante kI é um valor percentual indo 1% á 100%, este valor não pode ser excedido pois os valores acima de 100% faz com que o sistema fique instável. O seu valor decide a quantidade de erro absoluto que é eliminado pelo sistema.

Linha 5:

Nesta linha a variável Derivada ira receber o valor da diferença entre erro absoluto e o erro anterior multiplicado pela constante kD ou Constante Proporcional. Com a diferença do erro absoluto menos o erro anterior é possível prever o valor do erro futuro. Se é possível prever o erro futuro é possível se fazer uma antecipação e corrigir este erro em uma porcentagem kD ou constante derivativa. A Função da Derivada é estabilizar o sistema de forma a deixar o sistema ágil.

Linha 7:

A variável erro_anterior recebe o resultado do erro absoluto;

Linha 9:

Sabendo que o controlador PID é soma de três grandezas sendo elas Proporciona, Integral e Derivativa, a variável PID recebe a soma destes três valores. Seu valor representa o estado atual de um dispositivo físico. Quanto maior o valor da variável PID maior será a distância do setPoint e quanto mais próximo a variável PID estiver de

zero mais próximo estará do setPoint. Para se conseguir o valor de cada constante sendo elas a kP, kI e kD, é preciso fazer teste no sistema modificando o valor de cada variável até conseguir o estado ideal para o sistema.

Para testar este algoritmo foi utilizada a linguagem Java onde foi possível provar a funcionalidade do controlador PID. Na figura 3 mostra a interface onde foram colocados uma barra de controle deslizante, as variáveis de entrada como a constantes proporcional, Integral e Derivativa, e mais a baixo onde foi colocadas as saídas.

Figura 5 - Programa de teste do algoritmo PID

A primeira barra deslizante representa a posição atual. A segunda barra deslizante representa o setPoint ou ponto de chegada. Nos campos de entrada se encontra os valores de kP (Constante Proporcional), kI (Constante Integral) e kD (Constante Derivativa). Abaixo dos valores de entrada, são mostradas as saídas de cada variável sendo ela Proporcional, Integral e Derivativa.

5 – SIMULAÇÃO NO SOFTWARE PROTEUS

Para que seja possível a implementação do algoritmo PID foi preciso a utilização do software PROTEUS 7. O objetivo deste software é simular esquemas eletrônicos de uma forma simples e prática.

Esta capítulo mostrar a construção do esquema eletrônico utilizando o software PROTEUS 7 para simular o algoritmo PID. Neste processo são mostrados os componentes que foram utilizados e sua função no esquema eletrônico.

5.1 - ESQUEMA DE CIRCUITO - MÃO MECÂNICA

Para representar o esquema de circuito da mão mecânica, foi utilizado o Software Proteus 7. Neste software é possível montar e testar circuitos. Logo abaixo temos a representação esquema da mão mecânica.

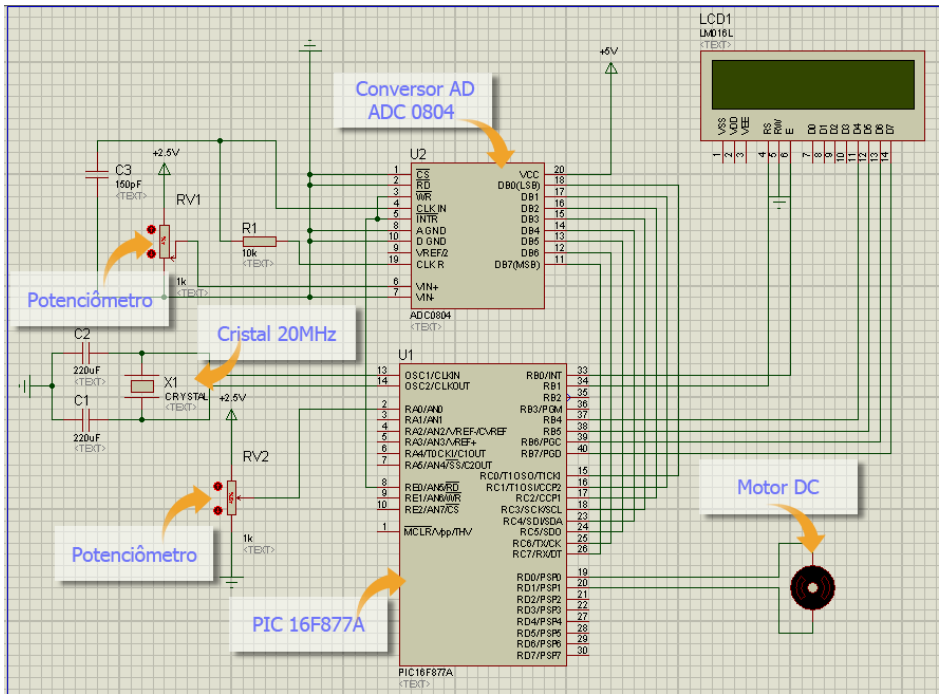

Figura 6 - Esquema para o teste do algoritmo PID

Como mostra a figura 6, foi utilizado para montar este esquema, um oscilador, um conversor digital ADC0804, um motor DC simples, um PIC 16F877A, potenciômetros, resistores, e capacitores. Nas simulações o microcontrolador PIC 16F877A foi utilizado pelo fato de ser simples de manusear, na implementação física o microcontrolador 18F452 tem a mesma forma de programação e função que o microcontrolador PIC 16F877A.

Para entender o que foi feito no esquema da figura 6, será mostrado às funções de cada componente.

5.2 - SENSOR DE MOVIMENTO

Os sensores são dispositivos físicos que capturam estímulos do ambiente externo como pressão, movimento, temperatura, entre outros. Ao captar um estimo, sua função é transformar essa informação em um dado numérico. Os sensores são utilizados para monitorar os diversos tipos de processo e com seu auxilio é possível que uma máquina consiga se orientar sem a ajuda humana. Sua aplicação abrange diversas áreas como, industrial e robótica.

Para se implementar o controlador PID, é preciso que haja um sensor monitorando o estado atual do motor. Assim é possível analisar o estado atual do dispositivo, ou seja, sua posição. Deste modo é possível corrigir o estado de uma máquina quando ocorrer uma alteração em seu estado.

Um meio simples para construir um sensor de posição seria a utilização de um conversor analógico digital 0804, capacitor e um potenciômetro 1k.

O conversor Analógico-Digital ADC0804 é um Circuito Integrado da National Semicondutores, sua função é converter uma amostra analógica entre 0 e 5v, em um valor binário de 8 bits. A figura 7 mostra a representação do ADC 0804:

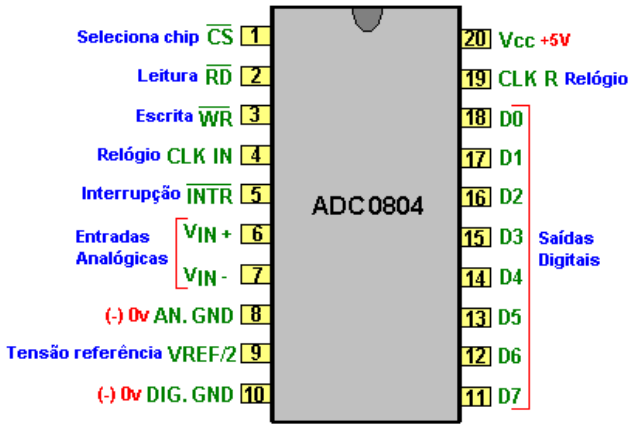

Figura 7 - Representação do componente ADC0804 (Messias, 2006)

Na tabela 3 na primeira coluna é referente a cada pino do ADC 0804 com sua respectiva função:

Tabela 3 - Descrição dos pinos do ADC0804 (Teixeira, 2009)

VIN(+) e VIN(-)	**Entradas analógicas diferenciais**
DB0 a DB7	**Saídas digitais**
/CS (entrada)	**Seleção do Chip quando em nível "0"**
/RD (entrada)	**Habilita leitura disponibilizando os dados na saída quando em nível "0"**
/WR (entrada)	**Comanda início de conversão quando em nível "0"**
CLOCKIN (entrada)	**Terminal de malha RC**
/INTR (saída)	**Sinal indicando fim de conversão quando comutado para nível "0"**
AGND	**Terra analógico**
DGND	**Terra digital**
VREF/2	**Tensão de referência no valor da metade de V+, para correção de fundo de escala**
CLOCK R (entrada)	**Segundo terminal para geração de clock interno**
V+ ou VREF	**Tensão de alimentação**

5.3 – ATIVANDO O ADC DO PIC

O ADC conversor analógico-digital tem função de gerar uma representação digital a partir de uma grandeza analógica. Alguns microcontroladores PIC possuem um ADC interno que pode ser ativado via programação. Para configurá-lo é preciso modificar as variáveis reservadas ADCON1 do PIC BASIC que é a representação do registrador adecon1 do PIC.

A variável ADCON1 é do tipo byte tendo oito posições que podem ser configuradas. A configuração do ADC deste projeto segue da seguinte forma:

ADCON1 = %00000100

Considerando o valor da variável ADCON1 o bit da posição 7 foi configurado para 0 para que o resultado dos valores de saída seja justificado a direita.

O valor do bit da posição 6 da variável ADCON1 é responsável pela escolha da categoria de conversão de clocks, ou seja, define a velocidade de atualização da conversão do ADC, neste caso o bit 6 foi configurado com valor 0.

Os valores do bit 5 ao 4 não são utilizados e os valores dos bits entre 3 ao 0 da variável ADCON1 são referentes as entradas do microcontrolador PIC 18F452 dos pinos da porta AN0, AN1 até AN7. O valor da posição dos bits entre 3 ao 0 define o tipo de entrada da porta, sendo uma entrada analógica ou uma entrada digital. Na configuração do bit 3 ao 0 foi dado o seguinte valor '0100' definindo as portas AN0, AN1 e AN3 como entrada analógica e as portas restantes como entrada digital.

Para a configuração de cada bit da variável ADCON1 foi utilizada as informações fornecidas pela Data Sheet do PIC 18FXX2. A tabela 4 mostra uma parte da documentação da configuração do registrado ADCON1.

Tabela 4 - Manual de configuração do AD do PIC 18FXX2 (microchip, 2001)

ADFM	ADCS2	—	—	PCFG3	PCFG2	PCFG1	PCFG0
bit 7							bit 0

bit 7 **ADFM:** A/D Result Format Select bit
1 = Right justified. Six (6) Most Significant bits of ADRESH are read as '0'.
0 = Left justified. Six (6) Least Significant bits of ADRESL are read as '0'.

bit 3-0 **PCFG3:PCFG0:** A/D Port Configuration Control bits

PCFG <3:0>	AN7	AN6	AN5	AN4	AN3	AN2	AN1	AN0	VREF+	VREF-	C / R
0000	A	A	A	A	A	A	A	A	VDD	VSS	8 / 0
0001	A	A	A	A	VREF+	A	A	A	AN3	VSS	7 / 1
0010	D	D	D	A	A	A	A	A	VDD	VSS	5 / 0
0011	D	D	D	A	VREF+	A	A	A	AN3	VSS	4 / 1
0100	D	D	D	D	A	D	A	A	VDD	VSS	3 / 0
0101	D	D	D	D	VREF+	D	A	A	AN3	VSS	2 / 1
011x	D	D	D	D	D	D	D	D	—	—	0 / 0
1000	A	A	A	A	VREF+	VREF-	A	A	AN3	AN2	6 / 2
1001	D	D	A	A	A	A	A	A	VDD	VSS	6 / 0
1010	D	D	A	A	VREF+	A	A	A	AN3	VSS	5 / 1
1011	D	D	A	A	VREF+	VREF-	A	A	AN3	AN2	4 / 2
1100	D	D	D	A	VREF+	VREF-	A	A	AN3	AN2	3 / 2
1101	D	D	D	D	VREF+	VREF-	A	A	AN3	AN2	2 / 2
1110	D	D	D	D	D	D	D	A	VDD	VSS	1 / 0
1111	D	D	D	D	VREF+	VREF-	D	A	AN3	AN2	1 / 2

A = Analog input D = Digital I/O
C/R = # of analog input channels / # of A/D voltage references

bit 6 **ADCS2:** A/D Conversion Clock Select bit (ADCON1 bits in **bold**)

ADCON1 <ADCS2>	ADCON0 <ADCS1:ADCS0>	Clock Conversion
0	00	Fosc/2
0	01	Fosc/8
0	10	Fosc/32
0	11	FRC (clock derived from the internal A/D RC oscillator)
1	00	Fosc/4
1	01	Fosc/16
1	10	Fosc/64
1	11	FRC (clock derived from the internal A/D RC oscillator)

bit 5-4 **Unimplemented:** Read as '0'

Para pegar os valores do ADC do microcontrolador é necessário chamar a função ADCIN que tem a seguinte sintaxe:

ADCIN 0, ValorADC

O primeiro parâmetro da função ADCIN se refere à porta AN0 do microcontrolador, caso este parâmetro tenha o valor 1 obtém os valores da porta AN1. O segundo parâmetro ValorADC se refere ao valor de conversão do ADC, neste projeto o tipo da variável ValorADC é dada como byte .

5.4 - MONTAGEM DO SENSOR DE MOVIMENTO

Para montar o sensor de movimento no software de simulação PROTEUS foram utilizados um potenciômetro de 1k , um capacitor e um conversor analógico digital ADC0804. O esquema pode ser visualizado na figura 8.

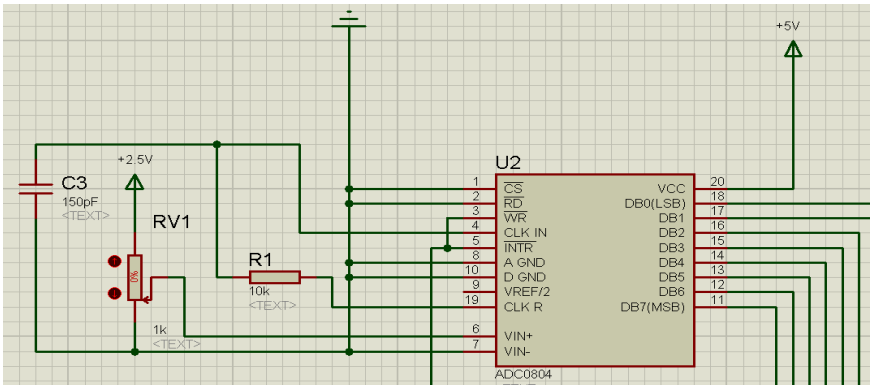

Figura 8 - Esquema do ADC0804

O processo para converter a saída do potenciômetro é obtido à posição atual do motor em representações binárias, é preciso compreender as operações envolvendo os pinos WR', RD', CLK IN, INTR e entre outros pinos do ADC0804.

O pino WR' junto com o pino INTR', quando ativados, faz o processo de conversão do sinal. Na execução do algoritmo PID estes pinos devem ser atualizados constantemente.

O pino RD' ao ser ativado coloca os dados lidos no barramento das portas DB1,DB2... DB7.

Os pinos VIN- e VIN+ coletam as faixas de tensão do potenciômetro. Estes pinos não deve ser excedida uma voltagem maior que 5 Volts**.**

O pino CLK IN e CLK R são responsáveis pelo clock interno do ADC, sua função é captar o tempo de clock e converter este tempo em um valor binário.

"[...] *Os pinos A GND e D GND tem a função de eliminar ruídos, sendo terra analógico (A GND) está ligado ao chão da fonte do sinal analógico e digital terrestre (D GND) está ligado à terra da fonte Vcc."* (S.Shet, 2009)

O pino 1 CS' é utilizado para ativar o ACD, em nível de tensão baixo o ADC é ativado; em nível de tensão alta o ADC é desligado.

O ADC 0804 pode converter níveis de tensão de 5Volts em até 255 unidades sendo de 0 a 127 números positivos e -128 a -1 números negativos. Para que o ADC converta somente números positivos, foi adotada uma tensão na entrada do potenciômetro de 2,5V. Desta forma o ADC ira converter a tensão em números entre 0 a 127.

Outra forma de se montar o sensor de movimento, seria a utilização do conversor analógico-digital interno do PIC 16F877A. Sua montagem é bem simples, para que funcione é preciso configurar os registradores do PIC 16F877A, isso é feito por meio da sua programação, na seção 5.3 foi mostrado à forma de configurar o conversor analógico-digital.

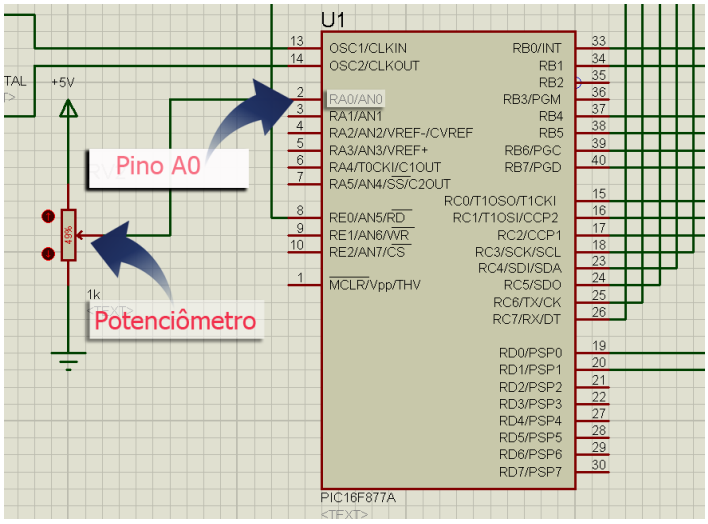

Figura 9 - Esquema ADC interno PIC 16F877A

Podemos notar na figura 9 que o pino central do potenciômetro com relação ao pino central chamado de cursor, esta ligado ao pino A0 do PIC 16F877A. A configuração do pino A0 é essencial para que possa ler corretamente o estado do potenciômetro. Sua voltagem é de 5 voltes e sua base esta aterrada.

5.5 - PIC 16F877A

O PIC 16F877 é um microcontrolador da família de 8 bits e núcleo de 14 bits fabricado pela Microchip Technology. Sua função no projeto é de vital importância, sendo o executor de todo o processo.

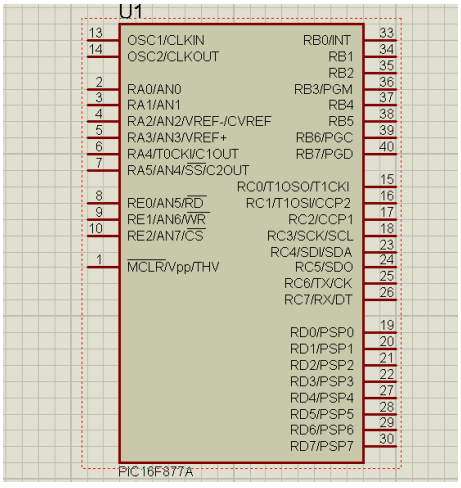

Figura 10 - PIC 16F877A

As características mais importantes do PIC 16F877A são:

- Sua freqüência de operação de clock vai até 20 MHz resultando em uma velocidade de processamento de 5 milhões de instruções por segundo. Isso se deve ao fato de que para cada instrução dentro do PIC são necessário 4 operações de clock.

- Sua faixa de alimentação pode variar entre 2 Volts a 5,5 Volts. No caso do projeto foi adotado uma voltagem de 5 Volts.

- Para armazenar os programas o PIC 16F877A contém uma memória Flash com 8k linhas sendo cada linha com 14 bits de tamanho. Também é contem uma memória EEPROM de 256 bytes e uma memória RAM com 368 bytes.

- É composto por conversor analógico-digital de 10 bits de resolução e 8 canais de entrada.

Para mais detalhes do funcionamento do PIC 16F877A como sua estrutura, configuração e funções, basta acessar o documento no site da MICROCHIP buscando o Data Sheet PIC 16F87X. (microchip, 2001)

5.6 - OSCILADOR

O PIC 16F877A consegue operar em uma freqüência máxima de 5MIPS (milhões de instruções por segundo) , onde cada instrução consome 4 clock. Para que podemos utilizar a sua freqüência máxima, foi utilizado na simulação do PROTEUS um oscilador de 20 MHz.

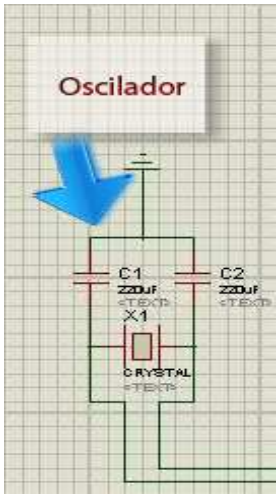

Figura 11 - Oscilador

5.7 - MOTOR DC SIMPLES

Para testar o algoritmo PID foi preciso a utilização de um motor DC ou motor de corrente continua. O motor DC serve para testar o funcionamento do algoritmo PID na prática de forma a se chegar perto do setPoint, o motor tem que reduzir a sua velocidade; caso a posição atual esteja longe do setPoint, o motor é acelerado.

5.8 – PWM PULSE-WIDTH MODULATION

Antes de analisar o algoritmo PID é preciso entender o funcionamento PWM e sua importância neste projeto.

O PWM pode ser definido com Modulação por Largura de Pulso (MLP) mais conhecida do inglês PWM (*Pulse-Width Modulation*). Sua função é gerar uma transferência de potência para o fio. Essa potência pode ser determinada em níveis de tensões baixas e altas geradas em um determinado tempo.

Na linguagem PIC BASIC existe a função PWM que recebe três parâmetros. A sintaxe da função PWM é a seguinte:

PWM PORTA, LARGURADEPULSO, CICLO

O primeiro parâmetro se refere à porta do microcontrolador que o PWM utiliza para a geração de largura de pulso. O segundo parâmetro se refere à quantidade de largura de pulso que o PWM gera, quanto maior for seu valor maior é a quantidade de pulso gerado pelo PWM. O terceiro parâmetro se refere à quantidade de ciclos que o PWM gera ao atuar sobre a porta do microcontrolador. O primeiro e segundo parâmetros são do tipo byte já o terceiro parâmetro é do tipo Word.

O método PWM tem a função de controlar a potência do motor DC através de picos de tensões geradas em um ciclo de tempo. No gráfico da figura 12 é possível analisar o funcionamento do PWM. A barra horizontal representa a quantidade de ciclos e a barra vertical esta representando a voltagem gerada pelo PWM.

Figura 12 - Gráfico PWM com escrita analógica em 0

No gráfico da figura 12 representa o PWM quando sua Largura de Pulso esta em 0%, ou seja, não gera nenhum pico de energia na saída do pino. Neste caso o motor DC encontra-se em repouso. Para simular este estado no PIC BASIC são passados os seguintes valores de parâmetros para o PWM:

```
Porta          = porta.0   //Escolhe a porta porta.0 PIC 16F877A
LarguraPulso = 0           //Valor que vai de 0 á 255
Ciclo          = 1         //Corresponde apenas um único ciclo do PWM

PWM(Porta,LarguraPulso, Ciclo)
```

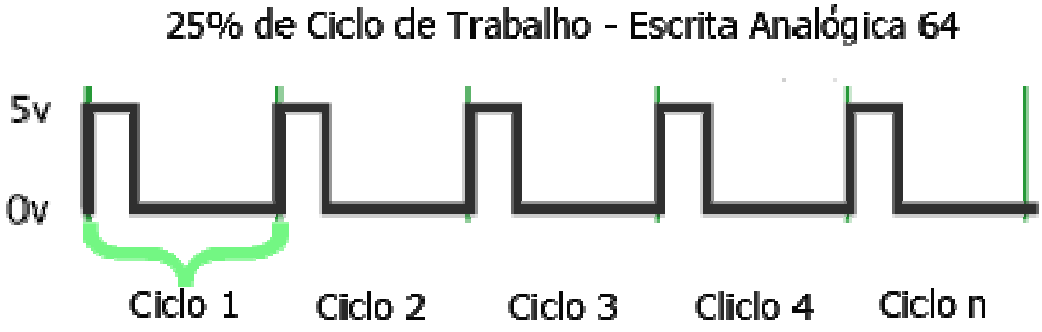

Figura 13 - Gráfico PWM com escrita analógica em 64

No gráfico da figura 13 o valor do ciclo de trabalho do PWM é de 25%. Neste estado é possível notar uma leve movimentação do motor DC. Para cada ciclo é possível notar uma potência de 25% de atuação, e o restante 75% seria a queda de potência. Para

simular este estado no PIC BASIC são passados os seguintes valores de parâmetros para o PWM:

```
Porta         = porta.0   //Escolhe a porta porta.0 PIC 16F877A
LarguraPulso = 64         //Valor que vai de 0 á 255
Ciclo         = 1         //Corresponde apenas um único ciclo do PWM

PWM(Porta, LarguraPulso, Ciclo)
```

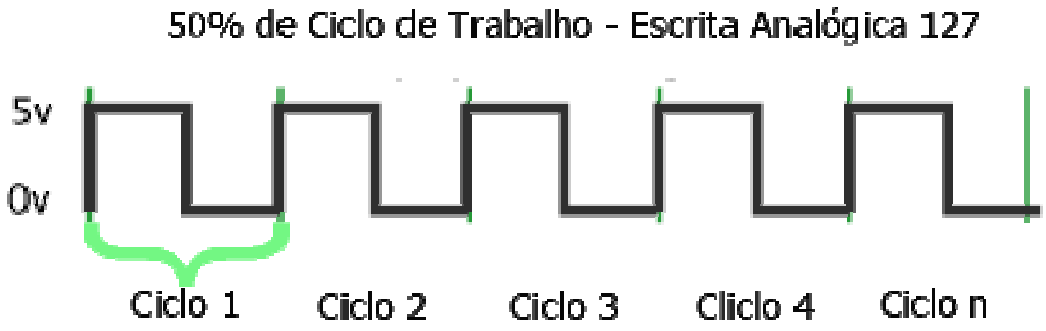

Figura 14 - Gráfico PWM com escrita analógica em 127

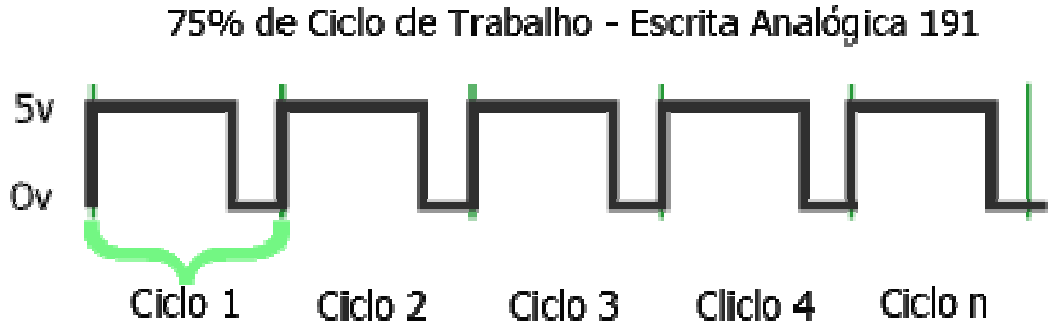

Figura 15- Gráfico PWM com escrita analógica em 191

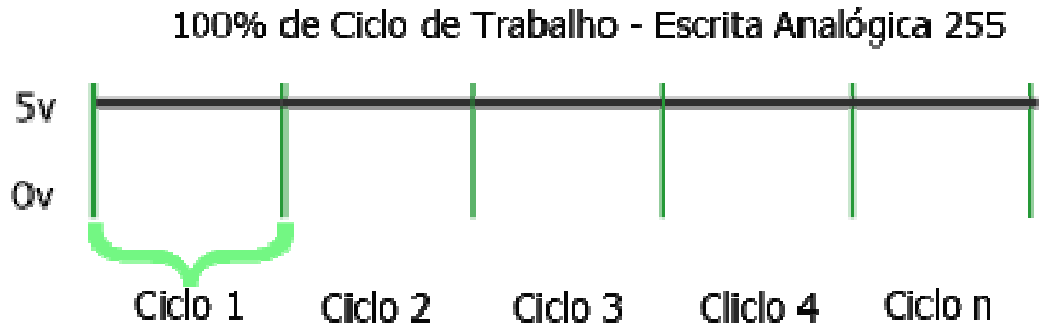

Figura 16 - Gráfico PWM com escrita analógica em 255

É possível ver na figura 16 que o potencial PWM está em seu valor máximo, neste estado é possível notar a velocidade máxima do motor DC. Para simular este estado, basta passar o método PWM fica da seguinte forma:

PWM(porta.0, 255, 1)

5.9 - IMPLEMENTAÇÃO DO ALGORITMO PID UTILIZANDO PIC BASIC

O algoritmo PID tem como base implementar as formulas proporcional, integral e derivativa como método de controle. Para a implementação e compilação do algoritmo PID foi utilizado o compilador MicroCode Studio disponível (mecanique).

Abaixo será apresentada toda a estrutura do algoritmo PID tentando buscar o máximo de clareza possível.

```
1  '************************Variáveis PID*****************************'
2  setPoint        VAR BYTE
3  PosicaoAtual    VAR BYTE
4  kP              VAR WORD
5  kI              VAR WORD
6  kD              VAR WORD
7  dt              VAR WORD
8
9  Proporcional    VAR WORD
10 Integral        VAR WORD
11 Derivativa      VAR WORD
12 PID             VAR WORD
13 erro            VAR WORD
14 erro_ant        VAR WORD
```

Figura 17 - Declaração das variáveis PID

Nas linhas 2 até 14 foram declaradas as variáveis que é utilizadas pela formula PID. A variável setPoint corresponde a variável de posição do destino ou objetivo desejado

do estado do dispositivo, essa variável foi declara como byte comportando valores correspondentes aos valores do ADC.

Na linha numero 3 variável de PosicaoAtual, como seu nome já diz, tem a função de guardar o estado atual do dispositivo. No caso, o motor foi declarado como byte com objetivo de pegar os valores da saída do ADC que é de 8 bits.

Nas linhas 4, 5, 6 foram declaradas respectivamente as constantes kP(Constante Proporciona), kI(Constante Proporcional) e kD(Constante Derivativa) .

As variáveis das linhas de 9 a 14 são utilizadas para receber os valores referentes á formula, e serão explicadas mais á diante.

```
15  '***************Variáveis para Movimentação do motor**************'
16  MOV_LEFT_BASE1  VAR PORTD.0
17  MOV_RIGHT_BASE1 VAR PORTD.1
18
19  MOV_LEFT_ONBRO  VAR PORTD.2
20  MOV_RIGHT_ONBRO VAR PORTD.3
21
22  MOV_LEFT_COT    VAR PORTD.4
23  MOV_RIGHT_COT   VAR PORTD.5
24
25  MOV_LEFT_PUNHO  VAR PORTD.6
26  MOV_RIGHT_PUNHO VAR PORTD.7
27
28  MOV_LEFT_MAO    VAR PORTB.1
29  MOV_RIGHT_MAO   VAR PORTB.2
```

Figura 18 - Variáveis representativa do motor

No trecho de código da figura 16 foi definido que para cada porta D e B de saída do PIC 16F877A tem uma variável correspondente ao motor e seu tipo de rotação. Como se pode ver na linha 15 existe a variável MOV_LEFT_BASE1 que representa a porta D0 do PIC 16F877A. Toda vez que esta porta estiver em alta o motor da base irá se movimentar para esquerda como se pode ver MOV_LEFT ou movimento á esquerda.

```
30  '***********************Variáveis de Controle*******************'
31  ControlePWM         VAR BIT
32  calcValInicioPWM    VAR BIT
33  valPWM              VAR BYTE
34  unPWM               VAR WORD
```

Figura 19 - Variável de controle

No trecho de código da figura 19 foram declaradas quatro variáveis de controle. Elas são usadas no começo da execução do algoritmo. Sua função será conhecida nos trechos de código da figura 21 e 22.

```
35  '************************Definição das Variáveis*********************'
36  setPoint = 127
37  dt = 1000
38  kP = 100
39  kI = 50
40  kD = 50
41  erro_ant = 0
42
43  ControlePWM = 0
44  calcValInicioPWM = 0
```

Figura 20 - Definição das variáveis

No trecho de código da linha 36 a 41 da figura 20 foram definidas as constantes. Essas variáveis, precisão ser definidas antes de colocar o algoritmo em execução. Lembrando-se que a variável kI é a constante Integral não pode ser maior que 100. Caso a constante kI seja maior que 100, pode ocorrer passagem do pondo desejado ou setPoint.

Os valores inseridos em kP, kI e kD são valores que definiram a precisão e velocidade de chegada ao setPoint. Quanto maior os valores de kP, kI e kD menor sua precisão e maior será a velocidade de chegada ao setPoint. Quanto menor for os valores de kP, kI e kD maior será a precisão e menor será a velocidade de chegada ao setPoint.

```
45  '***************************Programação*****************************'
46
47  Inicio:
48
49   IF ControlePWM  = 0 THEN
50        PosicaoAtual = 0
51        ControlePWM  = 1
52   ELSE
53          HIGH porte.0
54          PAUSE 5
55          LOW  porte.0
56          PosicaoAtual = portc
57   ENDIF
58
```

Figura 21 - Algoritmo PID passo 1

No código da figura 21 foi declarado uma estrutura de controle IF que faz a comparação da variável ControlePWM. Essa estrutura foi criada para que futuramente seja possível calcular o valor da posição do dispositivo quando este estiver no seu estado inicial, ou seja, PosicaoAtual seja igual a zero.

Na primeira chamada da rotina Inicio (linha 47) o controle IF (linha 49) considerara verdadeira a comparação, pois a variável ControlePWM é igual a zero. Na linha 50, a variável de PosicaoAtual é atualizada, recebendo o valor de posição zero. Na linha 51 a variável ControlePWM recebe o valor 1 para que na segunda e demais chamada da rotina Inicio, na estrutura IF ao ser comparado a variável ControlePWM retorne falso e caia dentro da estrutura ELSE.

Na estrutura ELSE (linha 51) referente às linhas 53 á 54 é utilizado para atualizar o estado o ADC 0804. Na linha 53 a porta E0 do PIC 16F877A que está conectada aos pinos WR' e INTR' é ativado. Desta forma o ADC 0804 lê o sinal de entrada do potenciômetro que logo depois é disponível no barramento da porta D do ADC. Na linha 54 é feito uma pausa de 5 milissegundos com objetivo de forçar a porta D do PIC ficar ativo. Na linha 55 a porta E0 do PIC fica desativada com objetivo de parar a leitura do ADC.

Para pegar o estado da posição do potenciômetro é preciso pegar a saída das portas D do ADC. As portas D do ADC estão conectadas diretamente as portas C do PIC 16F877A como podemos ver na figura 6. Na linha 55 a variável PosicaoAtual pega o valor da variável PORTC (representa todos os pinos C do PIC 16F877A) obtendo o assim o estado atual do potenciômetro.

```
59  '************************Formula PID*********************************'
60
61  erro = setPoint - PosicaoAtual
62
63     Proporcional = (kP*erro)
64     Integral = (Integral + (erro/dt)) * (kI/100)
65     Derivativa = (erro_ant-erro) * (kD/100)
66
67  erro_ant = erro
68
69  PID = Proporcional + Integral + Derivativa
70
```

Figura 22 - Algoritmo PID passo 2

Na figura 22 é inserido a formula PID, já comentado na seção 4.

```
71  '*************************  PWM    ******************************'
72
73      IF calcValInicioPWM = 0 THEN
74          unPWM = (PID/255)
75          calcValInicioPWM = 1
76      ELSE
77          valPWM = (PID/unPWM)
78          LOW MOV_LEFT_BASE1;
79          PWM MOV_RIGHT_BASE1,valPWM, 1
80      ENDIF
81
82  GOTO Inicio
83
84  END
```

Figura 23 - Algoritmo PID passo 3

No bloco de código da figura 23 foi criada uma estrutura de controle IF para calcular o correspondente valor PID para cada posição para cada unidade de potência do PWM. Na primeira chamada da rotina Inicio a variável calcValInicioPWM (linha 44) está com valor de atribuição em zero, ao ser comparado pela estrutura de controle IF da linha número 73 terá como valor verdadeiro. Na linha 74 a variável unPWM guarda o valor correspondente a uma unidade de posição para o valor de PID. Exemplo:

Se o valor PID da posição inicial for igual a 12700 qual é o valor para cada unidade da minha posição do meu potenciômetro?

Basta pegar o valor PID e dividir pela potência do meu PWM que é 255 desta forma teríamos:

$$unPWM = 25500/255$$

$$unPWM = 100$$

Pode-se concluir que para cada unidade de força do PWM obtêm-se um correspondente PID = 100.

Logo depois na linha número 75 a variável calcValInicioPWM é atribuída um valor 1. Desta forma ao se chamar à rotina Inicio pela segunda e demais vezes a estrutura IF da linha 73 retorna falso caindo dentro da estrutura ELSE.

Dentro da estrutura ELSE (linha 76) correspondente a linha 77 a variável valPWM recebe o valor de PWM com a divisão do valor PID pela unPWM (Unidade PWM).

Supondo que o valor do resultado PID seja igual a 800, e o valor de unPWM seja igual a 100 é gerado 8 unidades de largura pulso pelo PWM, sendo que 800 dividido por 100 é igual a 8.

Na linha 78 a variável MOV_LEFT_BASE1 correspondente a porta D0 do PIC 16F877A está em nível zero.

Na linha 79 o PWM é aplicado em cima da variável MOV_RING_BASE1 correspondente a porta D1 do PIC 16F877A. Desta forma o valor que a variável valPWM, guarda o valor da potência que é gerado no motor DC.

6 - TESTES FÍSICOS

No processo de testes físicos é utilizado um kit do modelo PICGenios que fornece um microcontrolador PIC 18F452, um dispositivo LCD para visualização dos resultados de entrada e saída do microcontrolador, e um conjunto de leds que representam algumas portas de saída do microcontrolador.

O microcontrolador PIC 18F452 foi utilizado no projeto de teste físico por dois motivos, sendo o primeiro pelo fato de o kit PicGenios fornecer o PIC 18F452 como seu componente padrão, e segundo por apresentar as mesmas funcionalidades do microcontrolador 16F877A onde a única diferença é o fato de ter maiores funcionalidades.

Para a inserção do algoritmo dentro do microcontrolador é utilizado o programa WinPIC800 fornecido pelo fabricante do kit PICGenios. Com este programa é possível carregar o código fonte e inseri-lo dentro do microcontrolador, bem como apagar ou copiar.

Nesta sessão será possível visualizar a forma de como foi montado o esquema do circuito eletrônico, mostrando as conexões entre as portas do microcontrolador com o motor dc e potenciômetro tendo uma idéia de sua implementação física.

6.1 - PROCESSO DE CONSTRUÇÃO DA MÃO MECÂNICA MODELO ROBOTIC ARM EDGE OWI-535.

Para desenvolver o projeto, foi realizada a compra do kit Mão Mecânica da marca ROBOTIC ARM EDGE OWI-535. O kit é constituído de 5 motores DC, base para

fixação da mão mecânica, um controle remoto e entre outras partes contendo todo o corpo da mão mecânica. O kit vem com todas as peças separadas sendo preciso à montagem de cada parte e seu valor pode chegar a 300 reais.

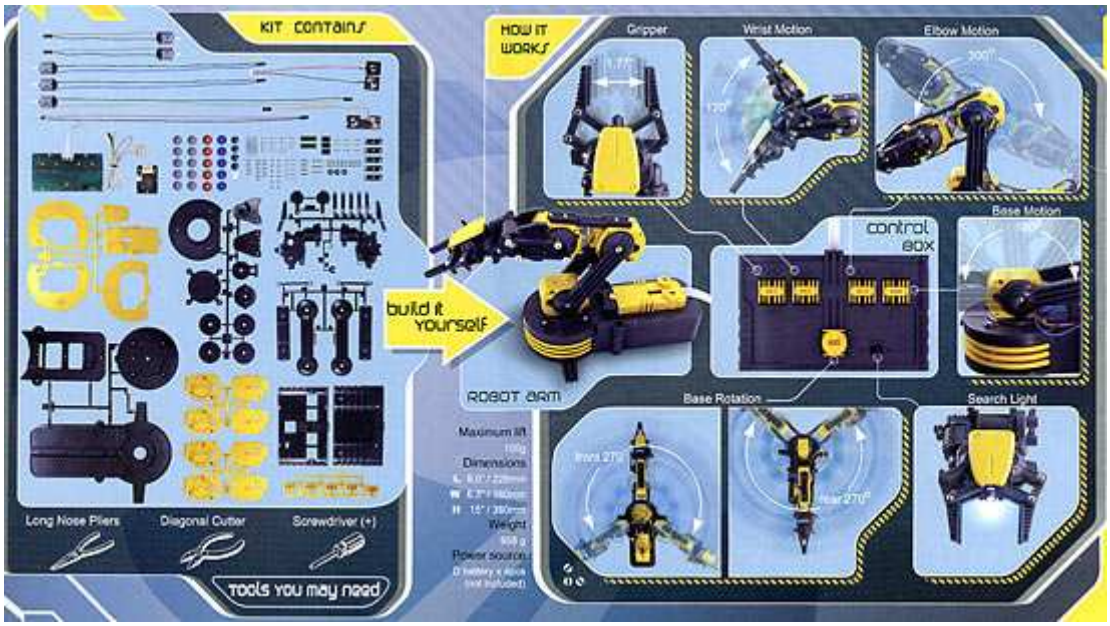

Figura 24 - Protótipo mão mecânica (microgenios, 2011)

Com relação a suas dimensões, a sua altura é de 15’’ polegadas, sendo seu raio de 12.6’’ polegadas. O motor da base tem uma movimentação de 180° de elevação e uma movimentação na horizontal de 270°. O motor cotovelo localiza-se acima da base tem uma movimentação de elevação de até 300°. Mais acima o motor que fica no pulso da mão mecânica tem a rotação na vertical de 120°. As pás que estão localizadas no topo da mão mecânica têm uma abertura de 1.77’’ polegadas.

Figura 25 - Demissão da mão mecânica[3]

O kit acompanha um controle que é ligado diretamente uma placa através de um cabo. A placa é responsável por gerenciar a energia das baterias para os motores e conectar os motores ao controle remoto.

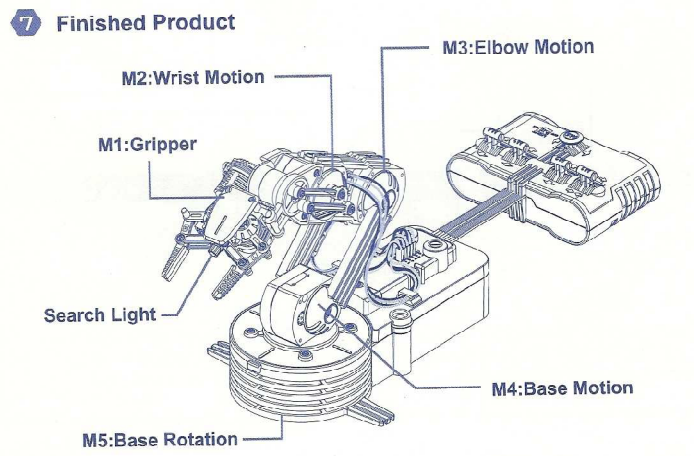

Figura 26 - Protótipo mão mecânica[3]

Seu peso pode chegar a 656 gramas e pode levantar objetos com peso Maximo de 100 gramas.

6.2 KIT PICGENIOS 18F V3.0

O kit PICGenios 18F V3.0 é um kit de desenvolvimento para microcontroladores contendo vários recursos. Este kit é muito utilizado por oferecer diferentes recursos como dispositivos LCD, Display de sete seguimentos, pinos de saída

das portas PORTA, PORTB, PORC e PORTD, entrada serial RS232 para auxilio na gravação do microcontrolador, entrada USB 2.0 e muitos outros recursos. Para que seja possível executar o algoritmo PID dentro do PIC 18F542 é preciso entender o funcionamento do kit e para isso será apresentado os componentes que serão utilizados no processo de gravação e teste.

Figura 27 - Kit PICGenios PIC 18F V3 (eletroncabrasil)

Para a gravação do PIC é utilizado uma porta serial RS232 com a função que auxilia na gravação do microcontrolador. Neste caso a entrada serial será ligada na saída seria do computador. Na figura 28 é possível ver a entrada seria do kit PICGenios:

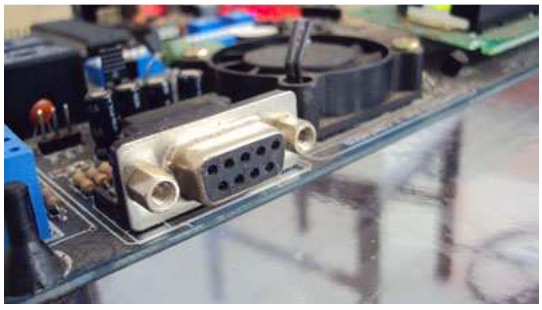

Figura 28 - Porta serial RS232

São disponibilizadas 5 portas de expansão para o microcontrolador correspondendo as portas PORTA, PORTB, PORTC e PORTD. No projeto, estas portas de

expansão serão utilizadas para a conexão e controle do dispositivo. Na figura 29 é mostrado as portas de expansão do microcontrolador:

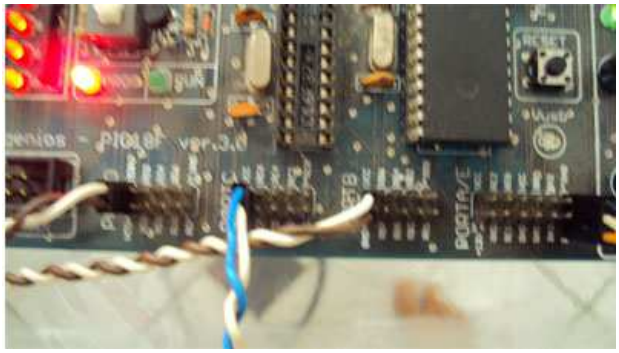

Figura 29 - Portas de expansão

O processo de gravação e execução do programa é utilizado uma chave com o nome Prog/Run representado por dois leds vermelho demonstra o processo de programação e o verde o processo de execução. Para a gravação do microcontrolador é preciso que a chave esteja ligada, desta forma o led vermelho fica aceso indicando que o kit estará pronto para receber o programa. Para executar o programa é preciso desativar a chave fazendo com que o led verde fique aceso indicando que o kit estará rodando o programa. A Figura 30 mostra a chave Prog/Run do kirPICGenios:

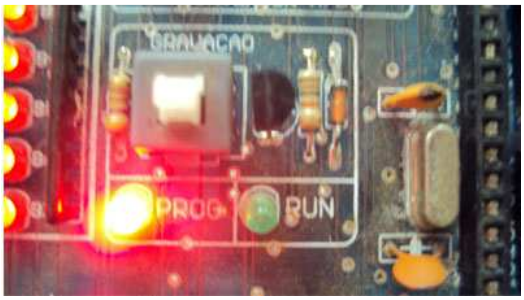

Figura 30 - Botão pressionado está no estado de programação

Figura 31 - Botão desapertado em modo de execução.

O kit também disponibiliza 16 leds que representam as portas PORTD e PORTB do microcontrolador. Esses leds são muito utilizados para ver os resultados das saídas das portas do microcontrolador. Os leds são representados na figura 32.

Figura 32 - Leds

6.3 PROGRAMA DE GRAVAÇÃO WINPIC800

O programa WinPIC800 V3.64H é utilizado para gravação de microcontroladoresPICs, este programa é gratuito e esta disponível em (winpic800) .No processo de gravação são utilizados uma das entradas do computador como porta serial, porta paralela ou USB. A tela inicial do WINPIC800 é mostrada na figura 33.

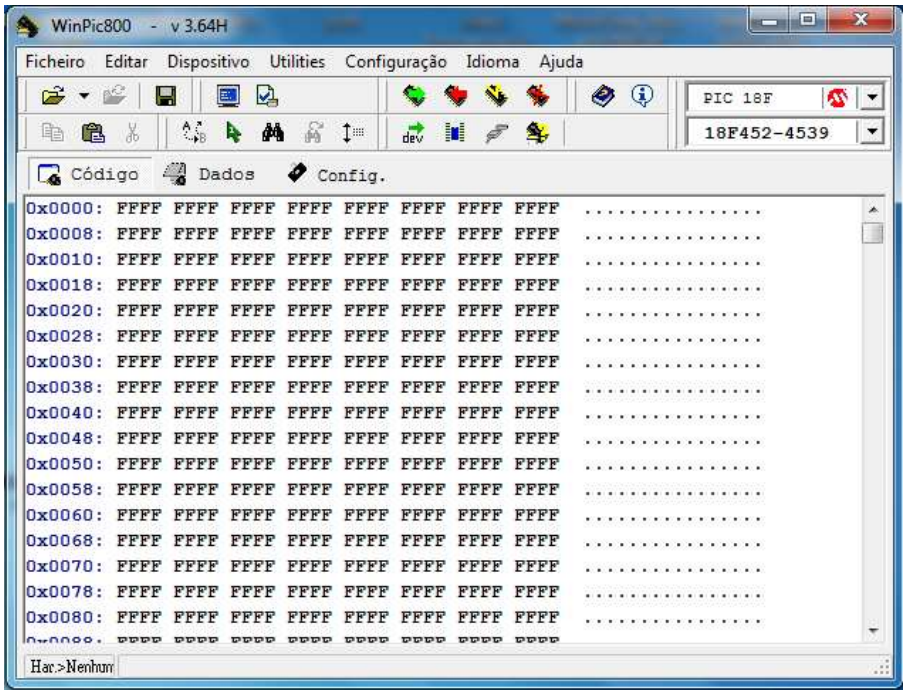

Figura 33 - Programa WinPIC800

O processo de gravação é mostrado no item 6.4, pois, para que seja possível estabelecer a conexão entre o WinPIC e o Kit PICGenios 18F, é necessário fazer algumas alterações na configuração de hardware do programa. Ao clicar no menu Configuração -> Hardware, vai aparecer uma tela conforme a figura 34.

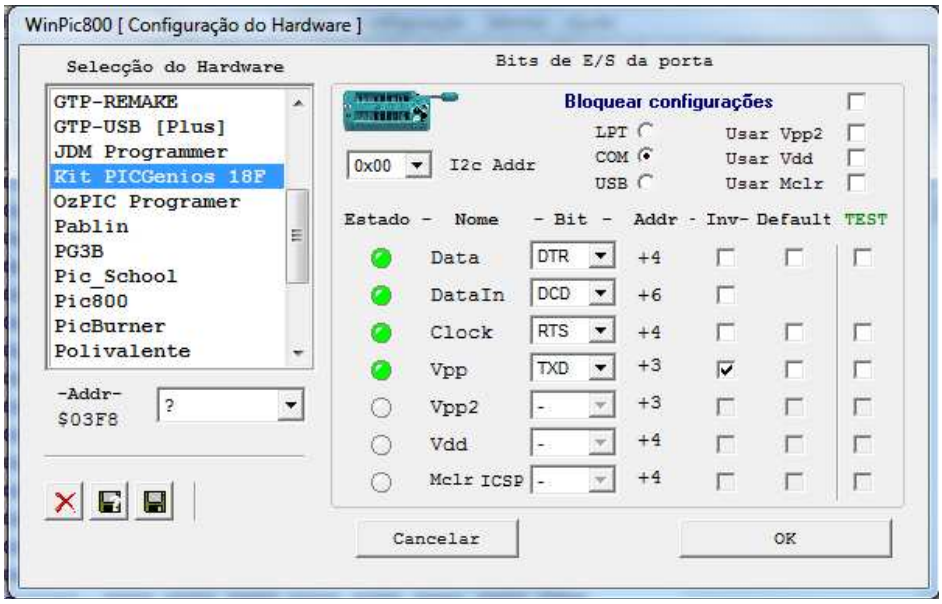

Figura 34 - Programa WinPIC800 tela de Configuração do Hardware

Estas configurações foram fornecidas pelo fabricante do Kit PICGenios em um tutorial. Esta configuração garante que o WinPIC faça a comunicação com Kit PICGenios

possibilitando a sua programação. O próximo passo é conhecer os botões na janela inicial do WinPIC.

Os primeiros passos para a configuração da janela inicial é a escolha do PIC que será utilizado. No caso do projeto é utilizado o PIC 18F452, desta forma é preciso escolher a família que o PIC pertence e logo depois o tipo do PIC que será trabalhado.

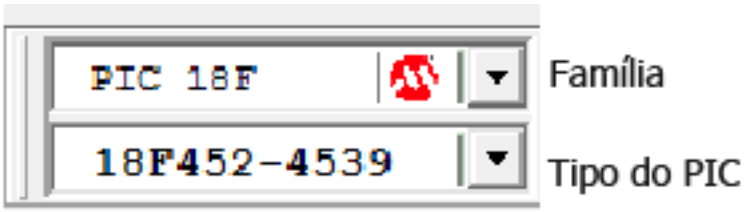

Figura 35 - Programa WinPIC800 escolha do PIC que será utilizado

Botão Abrir tem a função de abrir o código fonte com extensão .HEX

No processo de gravação é preciso carregar o arquivo que foi compilado pelo MicroCode Studio. Para isso basta clicar no botão abrir e localizar o código fonte do projeto , desta forma o código será carregado dentro do WinPIC.

Botão programar tudo usado para o envio do código fonte .HEX

Com o código carregado é preciso enviá-lo para o microcontrolador, isto é feito clicando no botão programar tudo. Com este processo vai aparecer uma tela com o status do envio do código fonte.

Botão Testar Hardware é usado para testar a conexão entre WinPIC e o Kit, ao ser clicado vai aparecer uma tela contendo o status de conexão com o Kit.

O Botão Apagar tudo tem a função de apagar todo o código dentro do PIC.

Botão Ler Tudo é utilizado para leitura do código que está inserida no PIC.

6.4 - PROCESSO DE GRAVAÇÃO DO PIC18F542 NO KIT PICGENIOS

Para que seja possível executar um programa no microcontrolador é preciso ter o código compilado pelo compilador, neste caso o MicroCode Studio. O arquivo compilado com a extensão .HEX tem que ser aberto pelo WinPIC800 que estará configurado para o kit PICGenios. O cabo serial RS232 será conectado ao computador junto ao kit PICGenios. No kit o botão PROG/RUN precisa estar pressionado para que ative o modo de gravação ocasionando o acendimento do led vermelho. Logo em seguida basta gravar o código que esta carregado no WinPIC800 clicando no botão programar tudo.

6.5 - TESTE DO ALGORITMO PID UTILIZANDO POTENCIÔMETRO E MOTOR DC.

Para a implementação do algoritmo PID foi utilizado um motor DC simples, um potenciômetro, um componente IRF640 e o kit PICGenios junto com o microcontrolador PIC 18F452. Na figura 36 é possível visualizar a construção física do projeto.

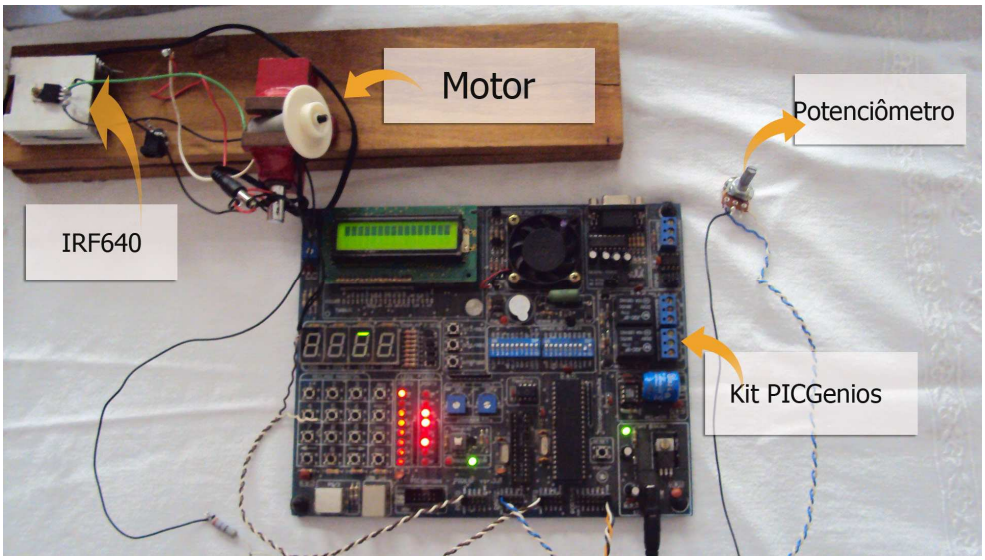

Figura 36 - Construção física do projeto de controle PID

O projeto referente à figura 36 tem o mesmo esquema da figura 6 da seção 5.1 com apenas duas modificações, como a alteração do microcontrolador PIC 16F877 para o microcontrolador PIC 18F452 e inserção do transistor IRF640.

Para que seja possível entender o que foi feito na figura 36 é preciso descreve as ligações feitas entre o potenciômetro e o motor junto ao kit e suas portas. Na figura 37 é mostrado o motor DC conectado ao transistor.

O motor DC tem a função de mostrar o comportamento do algoritmo PID conforme a saída dos dados gerados pelo potenciômetro. No projeto, poderia ter considerado outras formas de ver o comportamento do algoritmo PID como na regulação da temperatura de um dispositivo.

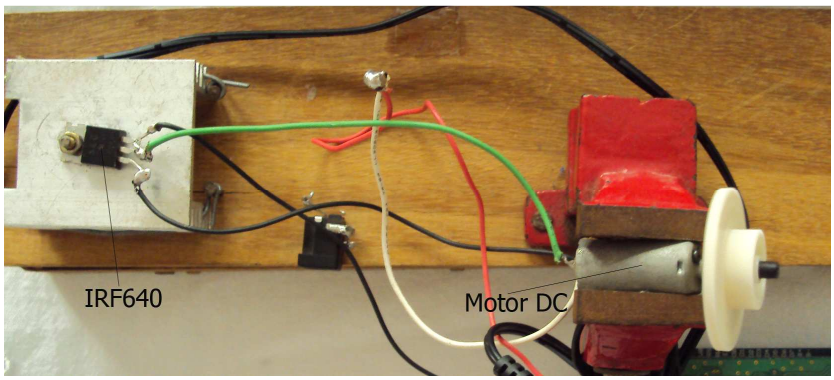

Figura 37 - Motor Conectado ao Transistor IRF640

O transistor IRF640 é constituído de três entradas, como pode ser visto na figura 38:

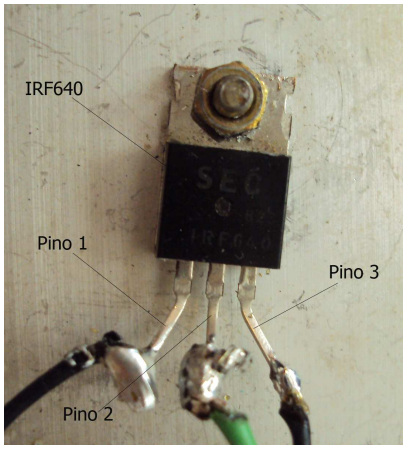

Figura 38 - Transistor IRF640

A função do IRF640 é permitir a passagem de energia do pino central para o pino 3 quando o pino 1 se encontrar e um nível de tensão alta, caso o nível de tensão do pino 1 esteja em baixa não ocorre à passagem de energia.

O pino 1 do IRF640 esta conectado a portd.0 da porta de extensão do kit PICGenios que é responsável por enviar os pulso do PWM. Desta forma quando o PWM enviar vários pulsos de tensão para esta portd.0 o IRF640 vai ativar e fechar a energia da segunda entrada do IRF640.

O pino central do IRF640 está ligado diretamente à entrada do motor DC, e a segunda entrada do motor DC esta ligada ao fio terra. Deste modo ao ser ativado o primeiro pino do IRF640, ocorrendo à passagem de uma corrente de energia de forma a ativar o motor DC. O potenciômetro pode ser visto na figura 39 sua função é servir como sensor de movimento.

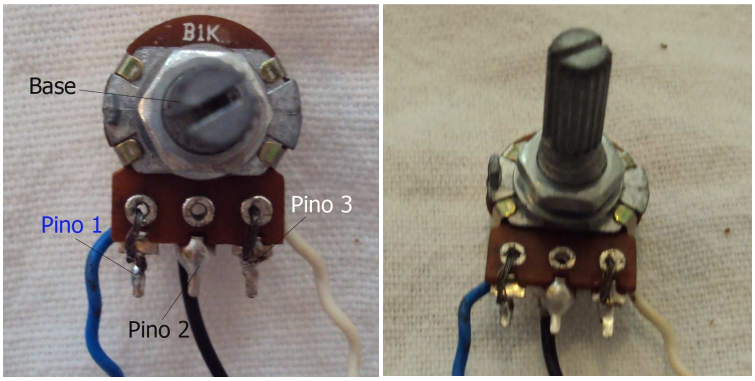

Figura 39 - Potenciômetro

A função do potenciômetro é oferecer uma resistência a passagem de energia de forma que ao girarmos a sua base uma quantidade de corrente que vai passar no pino central de forma que sua quantidade de energia pode ser nula ou total. O potenciômetro é constituído de três entradas onde as duas entradas laterais são conectadas ao fio terra e a outra sendo conectada a energia. Os pinos laterais do potenciômetro estão ligados nas portas de extensão do kit PICGenios que são VCC que é a porta que fornece uma energia de 6 volts e GND que seria o fio terra.

O pino central do potenciômetro esta ligado a porta.0 que faz a leitura do valor da tenção do potenciômetro e transforma em dados numéricos. A configuração da porta.0

foi mostrada na sessão 5.3, caso esta porta não seja configurada, não é possível pegar os valores do potenciômetro.

Com as orientações vistas na sessão 6.4 e tendo o processo de gravação realizado, é possível notar a atuação do algoritmo PID sobre o motor quando a base do potenciômetro é girada. Neste caso o setPoint está localizado na posição 127, considerando o potenciômetro em seu estado inicial onde a resistência é nula, ao girar a base do potenciômetro de forma que sua posição chegue perto do setPoint o motor começa acelerado e vai parando a medida que o potenciômetro chegue perto do setPoint.

6.6 - PROCESSO DE CONTROLE DA BASE DA MÃO MECÂNICA

Para que seja possível controlar a base da mão mecânica utilizando o controle PID, é preciso fazer a conexão entre a base da mão mecânica junto ao potenciômetro. Considerando o potenciômetro no seu estado inicial em zero e conectado corretamente a base da mão mecânica e sendo o estado de setPoint igual a 100 onde o estado do potenciômetro pode chegar de 0 á 127. Ao ser girado a base da mão mecânica, o potenciômetro acompanha o seu estado, modificando gradativamente sua posição, quanto mais perto do estado desejado ou setPoint, menor é a geração da largura dos pulsos do PWM. Deste modo, quando o valor do potenciômetro está em zero, o sistema começa a acelerar e à medida que o potenciômetro se aproxima do setPoint ocorre uma desaceleração gradual.

7 – CONCLUSÕES

Para que o projeto conseguisse ser concluído foi essencial a construção do algoritmo PID na linguagem Java de forma a ter uma visualização gráfica. Com este programa foi possível visualizar o comportamento do algoritmo PID, sendo possível modificar os valores de cada variável como as constantes proporcional, integral e derivativa visualizando o seu comportamento de forma clara. O motivo da construção do simulador Java foi para entender melhor o comportamento de cada ação relacionada a cada constante.

As simulações feitas pelo programa PROTEUS foram fundamentais para o entendimento inicial do comportamento PID sobre um motor DC. Com ele foi possível entender o processo de gravação do PIC e prever o comportamento do algoritmo PID sobre o motor DC.

No processo de testes físicos ocorreram grandes dificuldades como a inserção do algoritmo PID no microcontrolador PIC 18F542 pelo fato de fazer a configuração manual do programa WinPIC800. Todo o processo ficou claro com a utilização do tutorial fornecido pelo fabricante do Kit PICGenios, mostrando o processo de gravação do PIC.

Nas simulações e nos testes físicos o controlador PID mostrou-se eficiente. As ações proporcional, integral e derivativa apresentaram o comportamento esperado no controle do sistema.

Para o controle de dispositivos que tenham contato com objetos físicos o controle PID seria uma parte da solução, a outra parte seria a utilização de controle por visão computacional utilizando a biblioteca OpenCV.

REFERÊNCIAS BIBLIOGRÁFICAS

Brom, T. (03 de 01 de 2012). *Project 2: Elevator Simulator Part A*. Acesso em 02 de 06 de 2012, disponível em http://www.cse.msu.edu: http://www.cse.msu.edu/~cse251/project2a.html

eletroncabrasil. (s.d.). *Kit PICGenios 18F V3*. Fonte: http://www.eletronicabrasil.com.br: http://www.eletronicabrasil.com.br/produtos/foto1/grande/kitpicgenios_pic18fG.jpg

mecanique. (s.d.). *MicroCode Studio*. Acesso em 21 de 05 de 2012, disponível em http://www.mecanique.co.uk/: http://www.mecanique.co.uk/

Messias, A. R. (2006). *CARACTÉRISTICAS DE FUNCIONAMENTO DO CONVERSOR ANALÓGICO DIGITAL ADC0804 DE 8 BITS*. Acesso em 02 de 09 de 2012, disponível em http://www.rogercom.com: http://www.rogercom.com/pparalela/ConversorADC0804.htm

microchip. (2001). *PIC 16F87x Data Sheet 28/40-Pin 8-Bit CMOS FLASH.* Acesso em 05 de 09 de 2012, disponível em http://ww1.microchip.com: http://ww1.microchip.com/downloads/en/devicedoc/30292c.pdf

microgenios. (05 de 11 de 2011). *http://www.microgenios.com/*. Acesso em 03 de 08 de 2012, disponível em Robô OWI-535 - Robotic ARM Edge : http://www.microgenios.com/?11.57.0.0,463,robo-owi-535-robotic-arm-edge-|-permite-controle-via-arduino-e-pic-robotica-educacional-pronta-entrega.html

S.Shet, N. (10 de 2009). *Interfacing ADC to Microcontroller Chip 8051*. Acesso em 05 de 09 de 2012, disponível em http://www.oocities.org: http://www.oocities.org/techsoftronics/projects/adc.html

Teixeira, G. T. (15 de 09 de 2009). *ADC0804 (CONVERSOR A/D)* . Acesso em 03 de 09 de 2012, disponível em http://www.projetostecnologicos.com: http://www.projetostecnologicos.com/Componentes/CIsAnalogicos/ADC0804/ADC0804.html

winpic800. (s.d.). *WinPIC800*. Fonte: http://www.winpic800.com/: http://www.winpic800.com/

[1] http://es.wikipedia.org/wiki/Proporcional_integral_derivativo

[2] dicio.com.br

[3] Livro: ROBOTIC ARM-EDGE – wire control robotic arm kit