UNIVERSIDADE ESTADUAL PAULISTA

Campus de Ilha Solteira

PROGRAMA DE PÓS-GRADUAÇÃO EM ENGENHARIA ELÉTRICA

"Implementação de um controlador PID embarcado para o controle em malha fechada de um estimulador neuromuscular funcional"

THIAGO ALEXANDRE PRADO

Orientador: Prof. Dr. Aparecido Augusto de Carvalho

Dissertação apresentada à Faculdade de Engenharia - UNESP – Campus de Ilha Solteira, como requisito parcial para a obtenção do título de Mestre em Engenharia Elétrica.

Área de Conhecimento: Automação.

Ilha Solteira – SP

Julho/2009

UNIVERSIDADE ESTADUAL PAULISTA

Campus de Ilha Solteira

FICHA CATALOGRÁFICA

Elaborada pela Seção Técnica de Aquisição e Tratamento da Informação
Serviço Técnico de Biblioteca e Documentação da UNESP - Ilha Solteira.

P896i

Prado, Thiago Alexandre.
Implementação de um controlador PID embarcado para o controle em malha fechada de um estimulador neuromuscular funcional / Thiago Alexandre Prado. -- Ilha Solteira : [s.n.], 2009.
77 f. : il.

Dissertação (mestrado) - Universidade Estadual Paulista. Faculdade de Engenharia de Ilha Solteira. Área de conhecimento: Automação, 2009

Orientador: Aparecido Augusto de Carvalho
Bibliografia: p. 60-62

1. Estimulação elétrica neuromuscular. 2. Controladores PID.

unesp

UNIVERSIDADE ESTADUAL PAULISTA
CAMPUS DE ILHA SOLTEIRA
FACULDADE DE ENGENHARIA DE ILHA SOLTEIRA

CERTIFICADO DE APROVAÇÃO

TÍTULO: Implementação de um controlador PID embarcado para o controle em malha fechada de um estimulador neuromuscular funcional

AUTOR: THIAGO ALEXANDRE PRADO
ORIENTADOR: Prof. Dr. APARECIDO AUGUSTO DE CARVALHO

Aprovado como parte das exigências para obtenção do Título de MESTRE em ENGENHARIA ELÉTRICA , Área: AUTOMAÇÃO, pela Comissão Examinadora:

Prof. Dr. APARECIDO AUGUSTO DE CARVALHO
Departamento de Engenharia Elétrica / Faculdade de Engenharia de Ilha Solteira

Profa. Dra. ERICA REGINA M D MACHADO
Departamento de Matemática / Faculdade de Engenharia de Ilha Solteira

Prof. Dr. ERNANE JOSÉ XAVIER COSTA
Departamento de Ciências Básicas / Faculdade de Zootecnia e Engenharia de Alimentos de Pirassununga - USP

Data da realização: 03 de julho de 2009.

RESUMO

Por décadas, a aplicação de Estimulação Elétrica Neuromuscular (EENM) em pacientes paraplégicos e hemiplégicos tem melhorado significativamente sua qualidade de vida. Entretanto, comercialmente, essa técnica é aplicada sem o uso de uma lei de controle em malha fechada, o que limita sua eficiência. Assim, neste trabalho, projetou-se e implementou-se um controlador PID embarcado para o uso com estimuladores elétricos neuromusculares. O projeto desse controlador se baseou em um modelo matemático linear de segunda ordem, que representa o comportamento da força muscular devido a um estímulo elétrico. Posteriormente, desenvolveram-se algoritmos na linguagem de programação Python que possibilitam o projeto automático de controladores PID com diferentes especificações para diferentes pacientes. Dessa forma, o usuário informa os parâmetros do paciente e as especificações desejadas para obter a configuração do controlador PID adequada. Além disso, desenvolveu-se um programa em linguagem C para o microcontrolador PIC18F4520 que implementa este controlador utilizando o seu conversor A/D interno de 10 bits e um conversor D/A externo de 10 bits. Este controlador pode ser configurado via comunicação serial de forma simples e rápida, permitindo alterar os parâmetros do controlador PID e o tempo de amostragem. Por fim, os resultados obtidos a partir da simulação deste sistema no ambiente Proteus provou a viabilidade do controlador PID projetado.

Palavras-chave: paraplégico, hemiplégico, estimulação elétrica, EENM, controlador PID, Python, Proteus.

ABSTRACT

For decades the quality of life of hemiplegic and patients with spinal cord injuries has been improving through the research and use of Neuromuscular Electrical Stimulation (NMES) in rehabilitation engineering. However, most of the times it has been used without closed loop techniques, which is the approach used in this project, where an embedded PID controller has been designed and implemented to control the NMES. The plant to be controlled, i.e., the behaviour of the muscle response to an electrical stimulus, was represented using a second-order linear model. The language Python was then used to develop PID control algorithms allowing the use of different specifications so that the user can supply the patient parameters to properly configure the PID controller for different patients. Later, these algorithms were implement in a PIC18F4520 microcontroller using C language, exploiting its internal 10 bit A/D converter and an external 10 bit D/A. The final circuit can be configured serially via a PC, adjusting the controller parameters and the sampling rate. The whole system was simulated in the Proteus environment, proving its viability.

Keywords: paraplegic, hemiplegic, electrical situmulation, NMES, PID controller, Python, Proteus.

Índice de Figuras

Índice de Tabelas

Sumário

1. INTRODUÇÃO

A paralisia total ou parcial de um conjunto de músculos de uma pessoa compromete significativamente sua qualidade de vida. Por exemplo, conforme Johnstone (1979) citado por Silva (2007, p. 3), o paciente hemiplégico[1] tem dificuldades em manter uma postura correta, se equilibrar e realizar a marcha. O paciente paraplégico[2], segundo Rowley et al. (2000) citado por Cacho (2004, p. 17) com a perda da mobilidade sofre de várias complicações como a rápida atrofia muscular, osteoporose, aumento do risco de ocorrência de fraturas entre outras.

No entanto, os avanços na medicina têm mostrado que muitos dos pacientes que adquiriram paraplegia por causa de uma lesão medular têm condições físicas para realizar o tratamento para efetuar a marcha. Nesse contexto, alguns dos objetivos do uso de tratamentos com Estimulação Elétrica Neuromuscular (EENM), conforme Nene et al. (1996) e McDonald e Sadowsky (2002) citados por Vasconcelos Neto (2007, p. 35) têm sido restaurar os movimentos, prevenir complicações físicas decorrentes da lesão, além de promover o ortostatismo e a marcha.

Estudos sobre EENM têm avançado e trazido à tona bons resultados. Segundo Martin (1999) citado por Faria (2006, p. 2) há vários casos de pacientes que recuperaram a sensibilidade e o movimento parcial dos membros paralisados após sessões de EENM. Além disso, conforme Selkowitz (1985) citado por Peixoto (1995, p. 2) a EENM combate a atrofia muscular, aumenta a circulação de sangue no membro estimulado e previne problemas cardio vasculares.

Atualmente, o uso da EENM é comumente realizado em malha aberta, ou seja, com parâmetros pré-programados fixos. Para se obter o controle da marcha de um paciente faz-se necessário levar em consideração várias medidas como o ângulo formado entre a perna e o joelho, a força muscular, entre outros. Quando se faz isso, os parâmetros de estimulação não são mais fixos, mas variáveis. O controle da força muscular em malha

1 HEMIPLÉGICO: pessoa acometida de paralisia total ou parcial da metade lateral do corpo.
2 PARAPLÉGICO: pessoa acometida de paralisia das pernas e da parte inferior do tronco.

fechada devido à EENM possibilita a variação desses parâmetros de estimulação de forma apropriada.

Sendo assim, na subseção 1.1 descrever-se-á o trabalho realizado por Crago et al (1980), considerado pioneiro na área da EENM em malha fechada. Na subseção 1.2 serão descritos outros trabalhos realizados na área da EENM em malha fechada e na subseção 1.3 apresentar-se-á uma breve revisão do trabalho realizado por Law e Shields (2006) em relação a modelos matemáticos utilizados em EENM.

1.1 CONTROLE DA FORÇA MUSCULAR EM MALHA FECHADA

Em seus estudos, Crago et al. (1980, p. 306) citam como principais problemas do controle em malha fechada: a rápida fadiga do músculo e a mudança na eficácia do eletrodo devido ao movimento muscular. Felizmente, os esforços de vários pesquisadores reduziram a magnitude desses problemas. O desenvolvimento de eletrodos intramusculares que se movem junto com o músculo forneceu maior consistência na ativação do músculo. A resistência à fadiga das fibras musculares pôde ser aumentada por programas de exercícios eletricamente induzidos e empregando técnicas de modulação da força que minimizam a freqüência da estimulação. Dessa forma, com a melhora das técnicas de estimulação elétrica muscular pôde-se prosseguir com o estudo do controle em malha fechada.

1.1.1 METODOLOGIA

Na parte experimental, Crago et al (1980, p. 306-307) usaram o músculo soleus de um gato, visto que este tem propriedades mecânicas e metabólicas aproximadamente constantes, similares aos músculos estimulados em pacientes humanos. Além disso, este músculo tem alta resistência à fadiga, o que facilitou a realização dos experimentos. Toda a estimulação elétrica do músculo foi realizada utilizando eletrodos intramusculares formados por fios enrolados (bobinas).

O músculo foi estimulado com pulsos de corrente regulados monofásicos retangulares catódicos. Três parâmetros caracterizam a forma de onda destes pulsos: largura do pulso, amplitude do pulso e intervalo entre pulsos sucessivos. As técnicas de variação da

largura do pulso e do intervalo entre pulsos são denominadas controle por recrutamento e controle por somatória temporal, respectivamente. A amplitude dos pulsos foi fixada em cada experimento (CRAGO, et al. 1980, p. 307).

Para os experimentos em malha fechada, foram desenvolvidos circuitos com amplificadores operacionais, e os parâmetros do sinal de estimulação foram modulados por sinais analógicos aplicados a um estimulador controlado por tensão. Nesta análise em malha fechada, foram utilizados dois tipos de controladores: proporcional (P) e o proporcional-integral (PI) (CRAGO, et al. 1980, p. 307).

O desempenho dos sistemas de malha aberta e malha fechada foram avaliados através das respostas às entradas degrau e rampa. A estabilidade foi avaliada mantendo-se os comandos de entrada em vários níveis diferentes de força, por até 30 segundos. Os sistemas foram considerados estáveis se evidenciassem o amortecimento das oscilações. Adicionalmente, foi analisado a relação entre a saída (força) e a entrada (pulso) em regime permanente para avaliar a linearidade do sistema (CRAGO, et al. 1980, p. 307).

É relevante salientar que os resultados deste trabalho foram essencialmente experimentais, ou seja, os resultados não foram obtidos através de simulações computacionais.

1.1.2 RESULTADOS

Para a estimulação elétrica em malha aberta os resultados evidenciaram que no controle por recrutamento, a força (saída) é uma função não-linear da largura do pulso (entrada), com alta inclinação para larguras de pulso pequenas e uma baixa inclinação para larguras de pulso grandes. Já no controle por modulação do intervalo entre pulsos, obtém-se uma relação linear entre a força (saída) e o intervalo entre pulsos (entrada) (CRAGO, et al. 1980, p. 308).

No sistema em malha fechada com controlador proporcional o erro de regime diminui com o aumento do ganho de malha[3]. Além disso, a sensibilidade da força (saída) a uma mudança no ganho do músculo também diminui. Entretanto, o aumento do ganho de malha ocasiona oscilações indesejadas na saída. Estes resultados para o controlador

3 A expressão “ganho de malha” é um termo utilizado na área de controle de processos e se refere ao produto do ganho do músculo, do ganho do controlador e do ganho da realimentação negativa.

proporcional foram obtidos para a modulação de largura do pulso (CRAGO, et al. 1980, p. 308, 309).

Assim, o controlador proporcional foi considerado inadequado para o uso em aplicações de estimulação elétrica dos músculos, já que o alto ganho de malha necessário para diminuir os efeitos das variações dos parâmetros do músculo provocam oscilações indesejadas na modulação por recrutamento. Dessa forma, decidiu-se adicionar um integrador ao controlador que reduzirá o erro de regime a zero, mas deve contribuir para a instabilidade e um maior tempo de resposta do sistema em malha fechada (CRAGO, et al. 1980, p. 309).

Os sistemas com controlador PI foram estudados tanto para a modulação da força por recrutamento como por modulação do intervalo entre pulsos, utilizando ganho proporcional igual a um e várias taxas de ganho integral para proporcional. Os sistemas oscilaram para taxas maiores ou iguais a 20, mas não para taxas menores que 10. Assim, o controlador PI com ganho de malha proporcional unitário e taxa de ganho integral para proporcional igual a dez mostrou-se adequado para controlar a força muscular com a modulação por recrutamento ou com a modulação do intervalo entre pulsos (CRAGO, et al. 1980, p. 309).

O tempo de resposta do sistema em malha fechada foi tão rápido ou mais rápido que o do sistema em malha aberta. Por fim, um resultado muito importante é que a relação entre a força muscular (saída) e o estímulo elétrico (entrada) é linear para o sistema em malha fechada, o que não ocorre para o estímulo aplicado em malha aberta (CRAGO, et al. 1980, p. 309).

1.1.3 CONCLUSÃO

Os resultados mostram que é possível projetar sistemas de controle da força muscular em malha fechada tanto para modulação por recrutamento quanto para modulação por modulação do intervalo entre pulsos na estimulação intramuscular. Dessa forma, é viável o estudo de modelos matemáticos amplamente utilizados para estimulação neuromuscular funcional visando o controle em malha fechada (CRAGO, et al. 1980, p. 309).

1.2 OUTROS TRABALHOS NA ÁREA DE CONTROLE DA FORÇA MUSCULAR EM MALHA FECHADA

Após o trabalho de Crago et al. (1980), outros pesquisadores investiram seus tempo e energia em prol da reabilitação de pacientes por meio do uso da EENM em malha fechada. Nesse tópico relatar-se-ão alguns desses trabalhos.

Chizeck et al. (1983) projetaram um controlador digital do tipo um polo e um zero utilizando a modulação por largura de pulso. A fase de projeto consistiu em utilizar um modelo discreto de um músculo de gato e ferramentas clássicas de projetos de controladores, como alocação de pólos utilizando *root locus* (lugar das raízes). A análise de desempenho foi efetuada a partir das entradas degrau e rampa em músculos de gatos. Os primeiros resultados mostraram que o controlador se comportou de forma satisfatória e robusta por possuir baixa sensibilidade a erros de modelagem do músculo.

Whilhere et al. (1985) projetaram um controlador digital utilizando o método de síntese de Truxal, em que objetiva-se o cancelamento de parte da dinâmica da planta. O controlador implementado foi testado em músculos de gatos e avaliado quanto à estabilidade em malha fechada, linearidade e resposta à entrada degrau. Os resultados mostraram que o sistema é estável em malha fechada para várias entradas diferentes. No entanto, o resultado esperado para a entrada degrau foi diferente do obtido devido aos erros no cancelamento da parte dinâmica da planta por parte do controlador.

Bernotas et al. (1987) projetaram e analisaram um controlador adaptativo para o uso em EENM. Através de experimentos utilizando gatos, chegou à conclusão de que o controlador adaptativo pode ser muito eficiente no controle de um sistema com EENM. Entretanto, este controlador pode causar instabilidade devido a ruídos na obtenção de dados dos sensores quando o ganho de estimação de parâmetros é alto. Adicionalmente, realizou-se uma análise comparativa com um controlador de parâmetros fixos com um polo e um zero. O controlador de parâmetros fixos apresentou um bom desempenho, semelhante ao do controlador adaptativo. A diferença significativa entre os dois controladores é que o controlador de parâmetros fixos apresentou um pequeno erro de regime para a entrada rampa. Por fim, o autor sugere o uso de um controlador supervisório em conjunto com o controlador adaptativo para ajustar, ou até mesmo anular, o ganho de estimação em situações que

poderiam levar o sistema à instabilidade.

Lan et al. (1988) apresentaram um controlador digital para atuar em estimulação elétrica neuromuscular funcional utilizando modulação por largura de pulso (PW – *pulse width*) e modulação do período de estimulação (SP – *stimulus period*) simultaneamente. Este controlador PW+SP utiliza uma lei de controle baseada no controlador PI. Este sistema foi testado em músculos de gatos e apresentou melhor desempenho do que um controlador PI somente com modulação por largura de pulso.

Lan et al. (1991) realizaram testes com os três principais controladores propostos para o uso em EENM anteriormente. Dois controladores por modulação de largura de pulso, sendo um controlador PI e outro adaptativo, além de um terceiro controlador envolvendo modulação por largura de pulso e modulação do período de estimulação simultaneamente (PW+SP). Os testes foram realizados com músculo de gato. Os resultados informaram que o controlador com melhor desempenho foi o controlador com modulação PW+SP simultâneas. Dos controladores com modulação PW, ambos demonstraram desempenhos similares, exceto no que diz respeito ao sobre-sinal inicial. O controlador adaptativo apresentou sobre-sinal maior que os outros controladores devido a identificação dos parâmetros da planta.

Ferrarin et al. (1996) projetaram um controlador PID do tipo mestre-escravo para controlar a posição angular do joelho. A escolha dos parâmetros do controlador foi realizada utilizando-se o método de Ziegler e Nichols. Os resultados mostraram que para movimentos lentos o controlador PID (proporcional integral derivativo) teve bom desempenho, podendo-se aplicá-lo para auxiliar um paciente paraplégico a se levantar. Neste caso, a posição angular do joelho foi controlada tendo como referência o ângulo formado por outro goniômetro preso ao cotovelo.

Gwo-Ching Ghang et al. (1997) projetaram e analisaram um controlador neuro-PID[4] aplicado em EENM para o controle da posição angular do joelho. A análise dos resultados foi realizada com dois pacientes e a entrada do controlador não foi um degrau ou rampa. Os sinais de entrada utilizados foram do tipo senoidais de 0,5 Hz e 1,0 Hz. A análise do controlador neuro-PID mostrou que este é promissor na área de controle da posição angular do joelho.

4 Quando se diz neuro-PID refere-se a um controlador que possui um algoritmo de controle que combina a lei de controle PID com a teoria de redes neurais.

Ferrarin et al. (2001) analisaram o desempenho de diferentes estratégias para o controle da posição angular do joelho do paciente. Utilizaram o controle em malha aberta com o modelo inverso, o controle em malha fechada com o controlador PID e um controlador com a junção das duas estratégias. Foram utilizadas como entrada sinais do tipo senoidais. O controlador PID mostrou um bom desempenho em relação às outras estratégias, principalmente, no baixo sobre-sinal apresentado no início dos experimentos. Entretanto, os autores concluíram que um problema do controlador PID é o tempo de atraso em seguir a referência (entrada).

Jezernik et al. (2004) projetaram um controlador com a técnica denominada de modos deslizantes, aplicada a sistemas não-lineares. Esta técnica assegura erro nulo e estabilidade a todos os estados do sistema e robustez a distúrbios. O controlador foi testado em pacientes e demonstrou robustez, estabilidade e bom desempenho.

1.3 MODELOS MATEMÁTICOS UTILIZADOS EM ESTIMULAÇÃO NEUROMUSCULAR FUNCIONAL

O uso da estimulação elétrica de músculos tem como objetivo melhorar a qualidade de vida do paciente com membros paralisados, preservando a musculatura desses membros. Assim, faz-se necessário estudar padrões de estímulos que proporcionem o melhor resultado possível. O uso de modelos matemáticos fornece uma previsão do comportamento do músculo estimulado possibilitando escolher o melhor padrão de estímulos a ser utilizado.

Nesse contexto, Law e Shields (2006) discutiram qual o melhor modelo matemático a ser utilizado para prever o comportamento do músculo. Em seu estudo Law e Shields (2006, p. 1027) realizaram a comparação entre três modelos matemáticos amplamente conhecidos: um modelo linear simplificado, o modelo não-linear, proposto por Bobet e Stein, e o modelo de Hill-Huxley.

O modelo linear é composto por uma equação diferencial de segunda ordem linear. O modelo de Bobet e Stein prevê com maior exatidão o comportamento da força muscular, porém, é mais complexo e não-linear. Por fim, o modelo de Hill-Huxley é o mais complexo, sendo também não-linear (LAW, et al. 2006).

Comparando o resultado de simulações destes três modelos com experimentos reais envolvendo quatro pacientes, Law e Shields (2006, p. 1030) chegaram à conclusão de que o modelo de Hill-Huxley é o mais próximo do real, seguido pelo modelo de Bobet e Stein. Portanto, para uma boa previsão do comportamento da força muscular dever-se-ia utilizar o modelo de Hill-Huxley ou o modelo de Bobet e Stein. Entretanto, segundo Law e Shields (2006, p. 1034), para estímulos de baixa frequência, o modelo linear apresenta desempenho semelhante aos modelos não-lineares, ou seja, os erros causados pelos modelos testados são, praticamente, iguais para estímulos de baixa frequência.

Nesse sentido, o objetivo desse trabalho tem por ênfase o projeto de um controlador digital embarcado. Assim, escolheu-se o modelo linear simplificado para um estudo focado no projeto e implementação do controlador PID embarcado.

1.3.1 MODELO LINEAR

Neste modelo uma equação diferencial de segunda ordem é usada para prever a força muscular $f(t)$ decorrente do trem de pulsos de estimulação na entrada $r(t)$. A equação diferencial que representa esse modelo é apresentada na equação (1.1).

$$\frac{d^2 f(t)}{d t^2} + 2 \cdot \omega_n \cdot \xi \cdot \frac{d f(t)}{d t} + \omega_n^2 \cdot f(t) = \beta \cdot \omega_n^2 \cdot r(t) \tag{1.1}$$

Os parâmetros presentes na equação (1.1) não possuem um significado físico, porém, pode-se relacioná-los com coeficientes conhecidos da teoria de controle linear para sistemas de segunda ordem. Sendo assim, o coeficiente β é o ganho do estático do sistema, ξ o coeficiente de amortecimento e ω_n a freqüência natural.

Os parâmetros presentes na equação (1.1) foram obtidos de quatro pacientes. O procedimento foi realizado com o paciente sentado em sua própria cadeira de rodas e o pé esquerdo apoiado num dispositivo de medição de força, estando o joelho e o tornozelo posicionados em 90°. Adicionalmente, foram obtidas medidas eletromiográficas do músculo soleus utilizando-se um pré-amplificador com ganho igual a 35 em série com um amplificador diferencial com ganho igual a 5000. Os pulsos de estimulação elétrica utilizados possuíam amplitude constante de corrente e largura de 250 μs. Por fim, a força medida, os sinais eletromiográficos e a forma de onda da estimulação foram gravados simultaneamente

em uma fita magnética utilizando um código modulador, digitalizados a uma freqüência de 1 KHz e analisados posteriormente utilizando o ambiente Matlab 6.0 (LAW, et al. 2006, p. 1028).

Entretanto, mesmo com vários trabalhos na área de EENM em malha fechada, não se encontram disponíveis estimuladores controlados em malha fechada no mercado. Isto, provavelmente, se deve ao fato de os controladores projetados ainda não fornecerem respostas satisfatórias. Ou talvez pelo fato de não existir uma ferramenta de fácil manejo para pesquisadores da área de Engenharia de Reabilitação utilizarem estimuladores com um controlador embarcado.

1.4 OBJETIVOS

O objetivo desse trabalho é implementar um controlador embarcado para o controle de um neuroestimulador em malha fechada com uma interface amigável para o profissional da área de Engenharia de Reabilitação.

1.5 ORGANIZAÇÃO DO TEXTO

Esse texto foi organizado em cinco capítulos, incluída a Introdução apresentada no Capítulo 1. No Capítulo 2 descrevem-se as ferramentas e técnicas utilizadas para a implementação do controlador PID. No Capítulo 3 descreve-se a metodologia, enquanto que no Capítulo 4 são mostrados os resultados. Por fim, no Capítulo 5 são apresentadas as conclusões e as sugestões para trabalhos futuros.

2. FUNDAMENTOS TEÓRICOS

Visto que o objetivo deste trabalho é o desenvolvimento de um controlador PID embarcado para o uso em EENM, foi necessária uma revisão teórica de controladores PID e do efeito da implementação do mesmo em sistemas microcontrolados. Além disso, como o sistema não será testado em pacientes, abordar-se-á brevemente o modelo matemático utilizado para o projeto do controlador PID.

2.1 O CONTROLADOR PID

O controle de sistemas em malha fechada consiste, basicamente, em analisar o sinal de erro (diferença entre "valor da saída" e o "valor desejado") e aplicar uma correção no sinal de entrada da planta (ou sistema a ser controlado). O controlador PID como o significado de sua sigla sugere é composto de três partes. Na Figura 2.1 ilustra-se um sistema em malha fechada utilizando uma dessas partes, ou seja, utilizando um controlador proporcional.

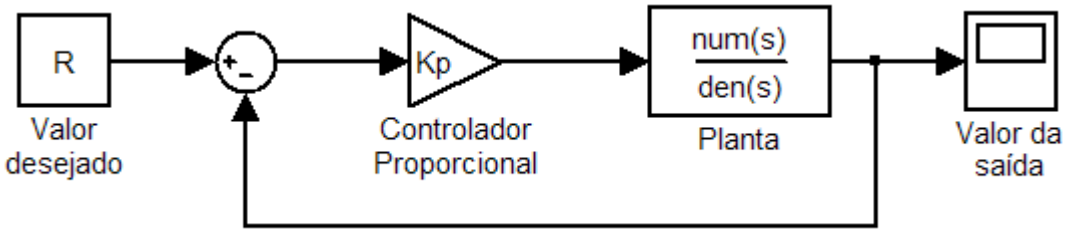

Figura 2.1 - Diagrama de blocos de um sistema de controle com controlador proporcional.

O controlador proporcional é alimentado com o sinal de erro $e(t)$, dependendo deste valor, atua de forma apropriada na planta em questão. A atuação do controlador na planta é dada pela equação (2.1), sendo *e(t)* o sinal de erro e *u(t)* a saída do controlador. Sendo assim, uma boa escolha da constante *Kp* é necessária para o funcionamento adequado do sistema.

$$u(t) = K_P * e(t) \tag{2.1}$$

A atuação do controlador integral é dada pela equação (2.2). Este controlador não é implementado individualmente como o controlador proporcional. Sistemas com este tipo de controlador costumam ser da forma PI ou PID. Na Figura 2.2 apresenta-se um exemplo de um sistema com um controlador PI.

$$u(t) = K_I * \int e(t) \tag{2.2}$$

A atuação do controlador PI é dada pela equação (2.3). A participação da parte integral é importante, visto que em muitos casos esta é responsável por anular o erro de regime do sistema em malha fechada.

$$u(t) = K_P * e(t) + K_I * \int e(t) \tag{2.3}$$

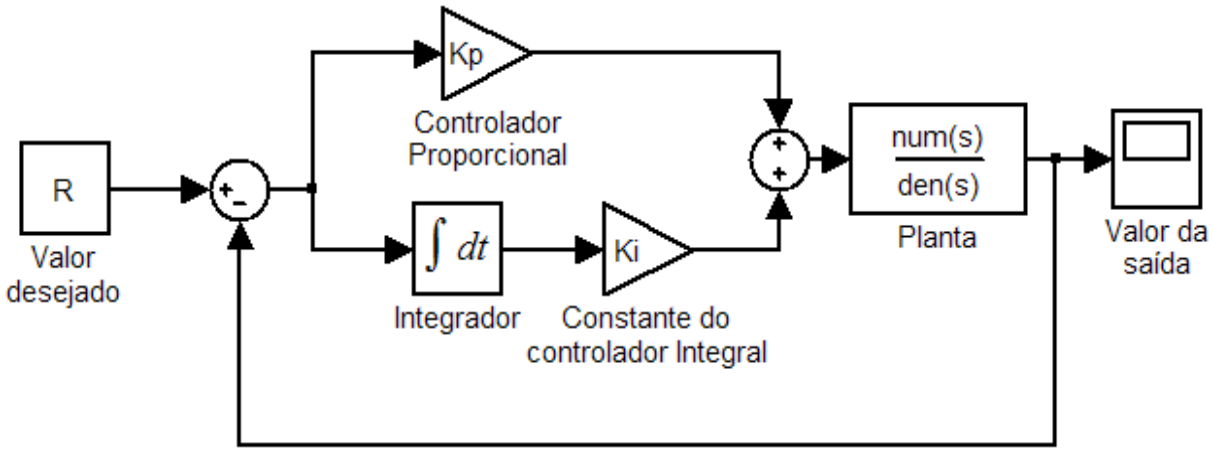

Figura 2.2 - Sistema de controle com controlador PI.

Por fim, a atuação do controlador derivativo é dada pela equação (2.4). Este controlador também não é implementado individualmente. Sistemas com este tipo de controlador costumam ser da forma PD ou PID. Na Figura 2.3 apresenta-se um exemplo de um sistema com um controlador PD.

$$u(t) = K_D * \frac{d\,e(t)}{d\,t} \tag{2.4}$$

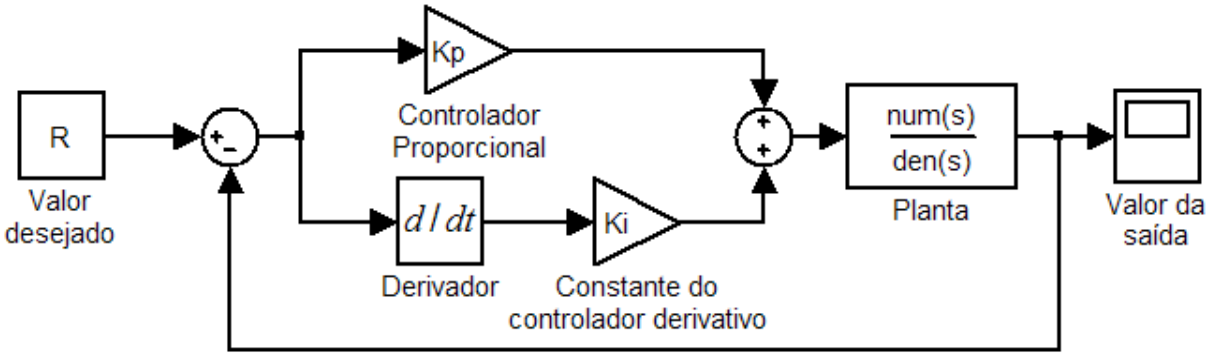

Figura 2.3 - Sistema em malha fechada com controlador PD.

O controlador PD é dado pela equação (2.5). A parte derivativa contribui para amortecer as oscilações do sistema visto que ele antecipa o sinal de erro calculando sua derivada. Já o controlador PID é implementado utilizando-se os três controladores em conjunto, conforme ilustrado na Figura 2.4. A atuação do controlador PID é, matematicamente, apresentada na equação (2.6).

$$u(t) = K_P * e(t) + K_D * \frac{de(t)}{dt} \tag{2.5}$$

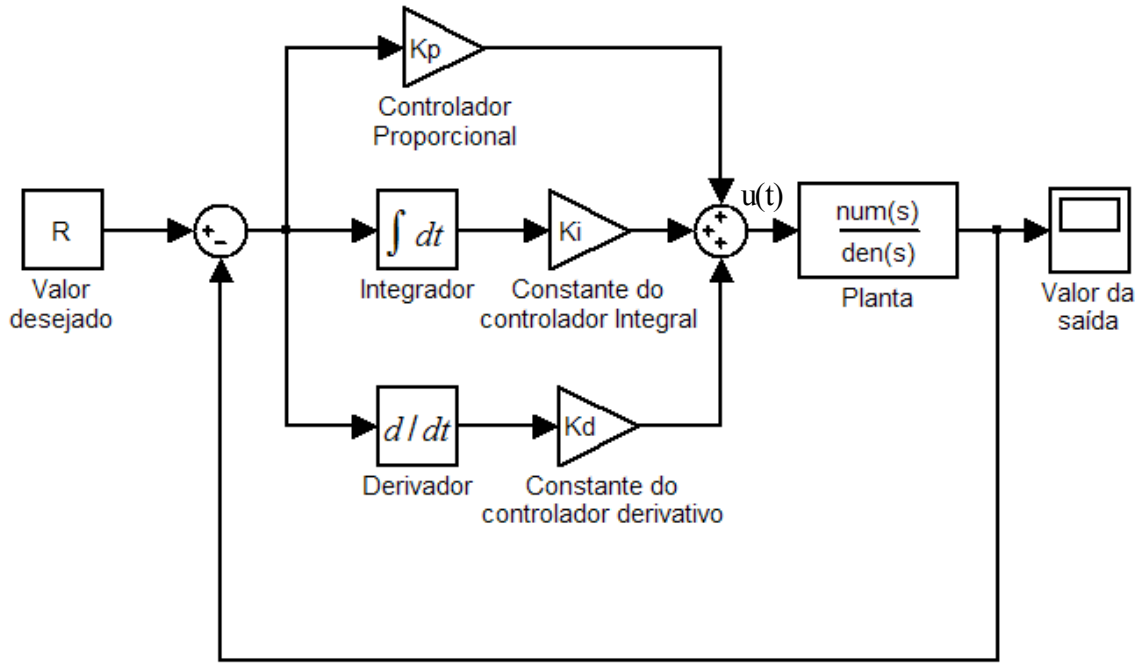

Figura 2.4 - Sistema em malha fechada com controlador PID.

$$u(t) = K_P * e(t) + K_I * \int e(t) + K_D * \frac{d\,e(t)}{d\,t} \quad (2.6)$$

O uso de controladores PID é muito difundido, visto que em muitos casos não há necessidade de se conhecer a planta do sistema. Um exemplo disso é a utilização dos métodos de Ziegler-Nichols para a regulação dos parâmetros deste controlador. Um destes métodos, chamado "curva de reação do processo" (válido para sistemas de primeira ordem), analisa a resposta do sistema sem nenhuma ação do controlador, e, em seguida, extrai-se desta curva parâmetros que resultarão nos ganhos *Kp*, *Ki* e *Kd*. Um segundo método proposto por Ziegler-Nichols, chamado "método do ciclo máximo" (válido para sistemas de segunda ou maior ordem), inicia-se com a análise da resposta do sistema apenas com o controlador proporcional, e, em seguida, extraindo-se parâmetros da curva de resposta obtêm-se os valores *Kp*, *Ki* e *Kd*. Entretanto, este método não permite a escolha de parâmetros, tais como: tempo de estabelecimento, amplitude de sobre sinal e tempo de subida (BOLTON, 1995).

Neste trabalho, o projeto do controlador PID tem por objetivo controlar uma planta modelada por uma equação diferencial de segunda ordem linear. O método utilizado é o de alocação de pólos, objetivando o cancelamento dos pólos da planta. Além desse método, há o método de projeto por pólos dominantes, projeto no domínio da frequência, além de variações desses métodos (ASTRÖM; HÄGGLUND, 1988).

2.2 EFEITOS DA IMPLEMENTAÇÃO DE ALGORITMOS DE CONTROLE EM SISTEMAS MICROCONTROLADOS

Praticamente todos os sistemas de controle implementados atualmente são baseados em sistemas controlados por computador. Basicamente, este tipo de sistema é composto de um processo real (planta analógica), um sistema digital processado e conversores A/D (analógico-digital) e D/A (digital-analógico). Apresenta-se na Figura 2.5 um diagrama de blocos deste tipo de sistema (ASTRÖM; WITTENMARK, 1997).

Alguns sistemas de controle digital[5] são implementados como uma aproximação dos sistemas de controle analógico, entretanto, este método de projeto não utiliza todo o potencial dos sistemas controlados por computador. Para aproveitar ao máximo

5 Deve-se entender, neste texto, que ao falar em sistema de controle digital, está se referindo a sistemas controlados por computador.

o potencial destes sistemas faz-se necessário entender as implicações da utilização de um processador para implementar a lei de controle e de conversores A/D e D/A para fazer a interface entre o mundo analógico e o digital (ASTRÖM; WITTENMARK, 1997).

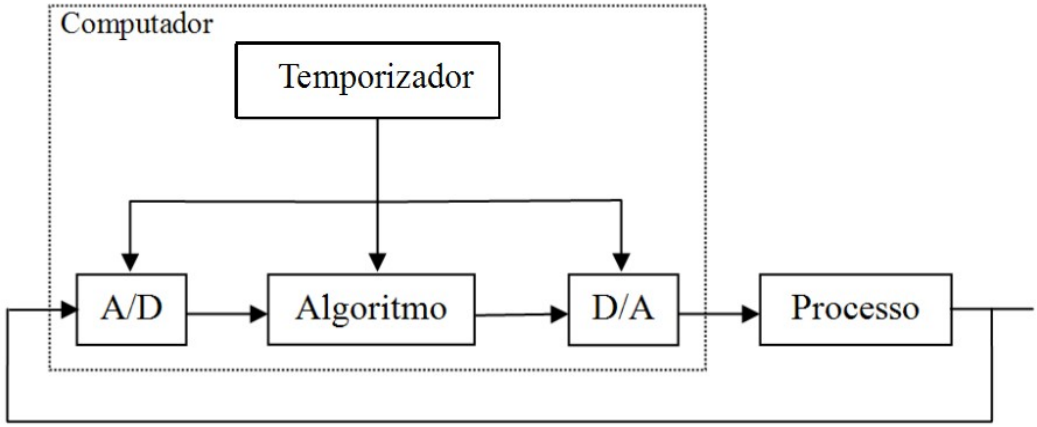

Figura 2.5 - Diagrama de blocos de um sistema controlado por computador.

Quando se pretende implementar um sistema controlado por computador é necessário responder a questões tais como: Qual deve ser a precisão dos conversores A/D e D/A? Que precisão é necessária nos cálculos computacionais? Para responder a essas questões é necessário entender os efeitos das limitações dos componentes do sistema e estimar suas conseqüências para um sistema de malha fechada.

2.2.1 Escolha do tamanho da palavra

Os algoritmos de controle digital são tipicamente implementados em microcomputadores com tamanhos de palavra de 8, 16 ou 32 bits. A principal limitação imposta pelo tamanho da palavra é exemplificada a seguir.

Um processador hipotético tem capacidade de trabalhar com números de ponto flutuante com precisão de três casas decimais. O cálculo do produto escalar entre os vetores *a* e *b* deve ser igual a um. No entanto, este processador calcula este produto escalar como sendo igual a zero. Isto ocorre por causa de sua representação numérica utilizando apenas três casas decimais. Considerando, então, a variável *c* como sendo o resultado desta operação, o processador segue as seguintes etapas.

$$a = (100\,;1\,;100) \qquad b = (100\,;1\,;-100)$$

ETAPA 1 → $c = a \cdot b = (100 * 100) + (1 * 1) + (100 * (-100))$

ETAPA 2 → $c = (1{,}000 * 10^4) + (1{,}000 * 10^0) + (-1{,}000 * 10^4)$

ETAPA 3 → $c = (1{,}000 * 10^4) + (-1{,}000 * 10^4)$

ETAPA 4 → $c = 0$

A soma do número $1{,}000*10^4$ com o número $1{,}000*10^0$ resultaria em $1{,}0001*10^4$. Entretanto, como visto na ETAPA 3, o resultado dado pelo processador despreza o valor na quarta casa decimal. Este tipo de erro numérico é chamado de erro por arredondamento. Isto pode deteriorar o desempenho de um sistema de controle causando, por exemplo, erro de regime.

Uma possível solução é trabalhar com uma representação numérica de maior precisão. Em alguns compiladores isso pode ser feito utilizando variáveis de dupla precisão no lugar de ponto flutuante.

Uma segunda possível solução é alterar a ordem dos cálculos. No exemplo citado poder-se-ia efetuar os cálculos da seguinte forma.

ETAPA 1 → $c = a \cdot b = (100 * 100) + (100 * (-100)) + (1 * 1)$

ETAPA 2 → $c = (1{,}000 * 10^4) + (-1{,}000 * 10^4) + (1{,}000 * 10^0)$

ETAPA 3 → $c = (0{,}000 * 10^0) + (1{,}000 * 10^0)$

ETAPA 4 → $c = 1$

Enfim, pode-se observar que é necessário mais do que apenas utilizar um tamanho de palavra grande para fornecer alta precisão. Precisa-se analisar também como são realizados os cálculos computacionais.

2.2.2 O Padrão IEEE-ANSI 754

Por muito tempo houve muitas representações numéricas para números de ponto flutuante. Porém, em 1985, foi publicado o padrão ANSI-IEEE 754 que padronizou a

representação dos números de ponto flutuante (ASTRÖM; WITTENMARK, 1997).

Neste padrão, o número em ponto flutuante é representado como um número da forma $\pm a \cdot 2^b$. Nesta representação $0 \leq a < 2$ é a mantissa e *b* é o expoente. Neste padrão, há três tipos de representações numéricas para um número de ponto flutuante. Estes três tipos diferem entre si no tamanho da palavra, ou quantidade de bits, utilizada. Apresenta-se na Tabela 2.1 os três tipos determinados pelo padrão ANSI-IEEE 754 (ASTRÖM; WITTENMARK, 1997).

Tabela 2.1 - Representação numérica de dados do tipo ponto flutuante conforme ANSI-IEEE 754

TIPO	SINAL	EXPOENTE	MANTISSA
Short real (32 bits)	1 bit	8 bits	23 bits
Long real (64 bits)	1 bit	11 bits	52 bits
Short temporary real (80 bits)	1 bit	15 bits	64 bits

O padrão ANSI-IEEE 754 ganhou larga aceitação e os fabricantes Intel e Motorola, por exemplo, têm sua representação numérica em ponto flutuante baseada nesse padrão. O fabricante dos microcontroladores PIC, Microchip, também se baseia nesse padrão, porém, com pequenas modificações. Mesmo assim, a Microchip fornece bibliotecas para a conversão do seu padrão de representação numérica para o padrão ANSI-IEEE 754 (TESTA, 1997). Apresenta-se na Tabela 2.2 a comparação entre estas duas representações.

Tabela 2.2 - Tabela comparativa entre os diferentes padrões de representação de números de ponto flutuante (TESTA, 1997)

	expoente	mantissa		
ANSI-IEEE 754 (32 bits)	**s**xxx xxxx	**y**.xxx xxxx	xxxx xxxx	xxxx xxxx
Microchip (32 bits)	xxxx xxxx	**s**.xxx xxxx	xxxx xxxx	xxxx xxxx
Microchip (24 bits)	xxxx xxxx	**s**.xxx xxxx	xxxx xxxx	

Além de considerar como o fabricante padroniza a forma de representação numérica, é necessário estudar o compilador utilizado. Neste trabalho, utilizou-se o compilador MikroC do fabricante MikroElektronika. Este compilador utiliza a linguagem C para a programação e disponibiliza ao programador os tipos de variáveis mostrados na Tabela 2.3. A representação numérica em ponto flutuante utilizada é a Microchip (32 bits) citada na Tabela 2.2. Além disso, não é disponível nesse compilador a representação de números de

dupla precisão, visto que, se o usuário utilizar o tipo “double" o compilador interpretará como “float” (MIKROELEKTRONIKA, 2009).

Tabela 2.3 - Tipos de variáveis disponibilizados pelo compilador MikroC (MIKROELEKTRONIKA, 2009)

Tipo	Tamanho da palavra	Faixa
(unsigned) char	8 bits	0 .. 255
signed char	8 bits	-128 .. +127
(signed) short (int)	8 bits	-128 .. +127
unsigned short (int)	8 bits	0 .. 255
(signed) int	16 bits	-32768 .. 32767
unsigned (int)	16 bits	0 .. 65535
(signed) long (int)	32 bits	-2147483648 .. 2147483647
unsigned long (int)	32 bits	0 .. 4294967295
Float	32 bits	±1.17549435082e-38 .. ±6.80564774407e38
Double	32 bits	±1.17549435082e-38 .. ±6.80564774407e38
Long Double	32 bits	±1.17549435082e-38 .. ±6.80564774407e38

Assim, será que a representação numérica em ponto flutuante fornecida pelo compilador MikroC é adequada? Depende do caso. Por exemplo, para resolver o problema citado no começo do tópico quanto ao cálculo do produto escalar entre os vetores *a* e *b*, esta representação é adequada. O cálculo se daria nas seguintes etapas.

ETAPA 1 → $c = a \cdot b = (100 * 100) + (1 * 1) + (100 * (-100))$

ETAPA 2 → $c = (1{,}0000000 * 10^4) + (1{,}0000000 * 10^0) + (-1{,}0000000 * 10^4)$

ETAPA 3 → $c = (1{,}0001000 * 10^4) + (-1{,}0000000 * 10^4)$

ETAPA 4 → $c = 1{,}0000000 * 10^0$

Foram utilizadas 7 casas decimais como uma aproximação da precisão fornecida pela representação numérica padronizada pela Microchip (32-bits), fazendo-se $2^{-23} = 0{,}00000011920928955078125 = 1{,}1920928955078125 * 10^{-7}$.

Finalmente, o projetista do sistema controlado por computador deve ter profundo conhecimento pelo controlador projetado e pela forma como fará os cálculos

computacionais a fim de aplicar a lei de controle de forma adequada ao processador utilizado.

2.2.3 Escolha da resolução dos conversores A/D e D/A

Atualmente, a resolução básica dos conversores D/A é em torno de 10 bits e dos conversores A/D é de 8 a 16 bits. Comparando-se isso à quantidade de bits geralmente utilizada para números representados em ponto flutuante, de 32 a 64 bits, pode-se concluir que a precisão de tais conversores terá grande influência no desempenho de sistemas controlados por computador. Assim, o projetista do sistema de controle precisa entender as implicações da resolução limitada dos conversores A/D e D/A (ASTRÖM; WITTENMARK, 1997).

2.2.3.1 Escolha da resolução mínima para o conversor A/D

O nível de quantização no conversor A/D é obtido dividindo-se a faixa de variação da tensão analógica pelo número de combinações possíveis fornecida pela quantidade de bits do conversor. Por exemplo, um conversor A/D com resolução de 8 bits operando em uma faixa de tensão analógica de -1 V a +1 V possui um nível de quantização igual a $\frac{\Delta V}{2^{bits}} = \frac{(+1)-(-1)}{2^8} = \frac{2}{256} = 7{,}8125\, mV$.

Este nível de quantização causa ondulações indesejadas na saída do processo. Por exemplo, note na Figura 2.6 que a saída de um processo qualquer apesar de estar visualmente adequada, apresenta pequenas ondulações como mostrado na Figura 2.7. Desta forma, a estimativa da resolução mínima necessária para o conversor A/D está ligada à precisão desejada na saída do processo a ser controlado. Ou seja, a resolução mínima do conversor A/D deve ser escolhida tal que o seu nível de quantização seja menor que a precisão desejada. Então, tem-se que a resolução mínima do conversor A/D deva ser calculada utilizando-se a equação (2.7).

$$bits_{AD} = -\log_2\left(\frac{\Delta V}{precisão}\right) \tag{2.7}$$

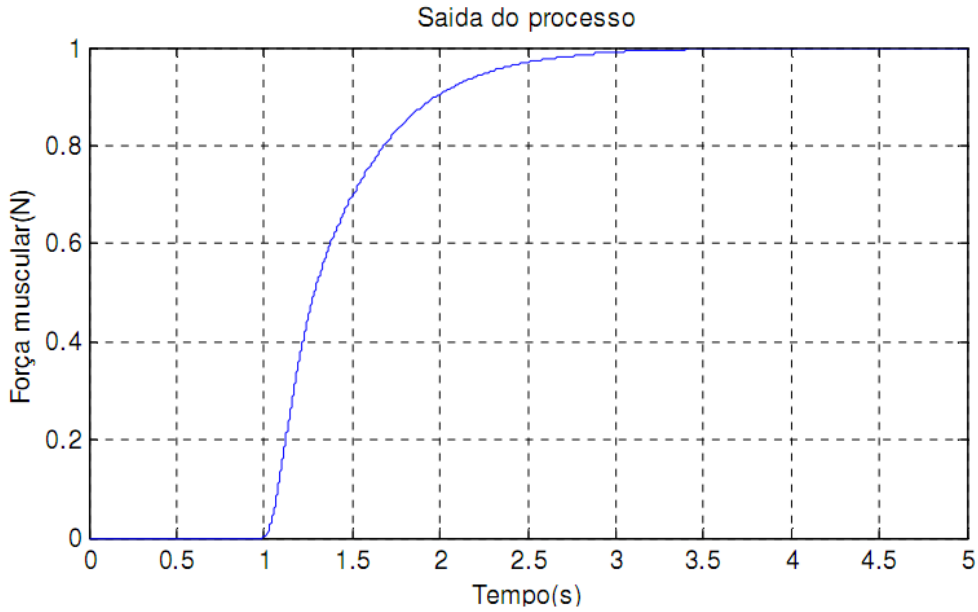

Figura 2.6 - Resposta de um sistema devido a uma entrada degrau.

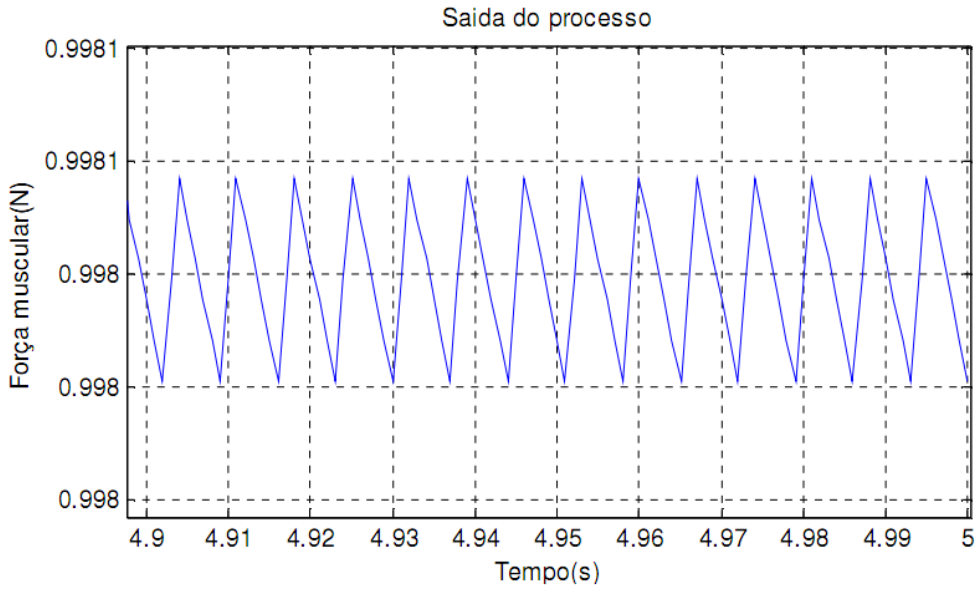

Figura 2.7 - Detalhe da resposta de um processo à entrada degrau.

Porém, é recomendado o uso de simulações específicas para estimar com maior precisão o efeito da quantização do conversor A/D (ASTRÖM; WITTENMARK, 1997). Ambientes de simulação como o Matlab/Simulink oferecem ferramentas práticas para avaliar estes efeitos.

2.2.3.2 Escolha da resolução mínima para o conversor D/A

O nível de quantização do conversor D/A também afeta a resposta do sistema similar ao considerado na Figura 2.7, porém, de uma forma diferente. A principal diferença é que o erro de quantização do conversor D/A é amplificado pelo ganho estático da planta do processo. Assim, a equação para o cálculo da resolução mínima do conversor fornecida para o conversor A/D não é suficiente para o conversor D/A.

O nível de quantização de um conversor D/A deve ser multiplicado pelo ganho estático da planta do processo para poder estimar o erro devido à quantização do mesmo. Assim, deve-se calcular a resolução mínima necessária para o conversor D/A utilizando a equação (2.8).

$$bits_{DA} = -\log_2\left(\frac{\Delta V * ganho}{precisão}\right) \tag{2.8}$$

Além disso, as oscilações na saída do processo devido à quantização do conversor D/A podem ser anuladas se a saída do conversor D/A for realimentada na lei de controle de forma adequada (ASTRÖM; WITTENMARK, 1997).

De forma análoga ao conversor A/D, recomenda-se, novamente, o uso de simulações específicas para determinar a resolução necessária para o conversor D/A (ASTRÖM; WITTENMARK, 1997).

2.2.4 Efeitos da implementação de controladores PID digitais

O controlador PID é, com certeza, um dos mais comuns algoritmos de controle utilizados atualmente. Muitos processos são controlados por um controlador PID ou por variações dele. Apresentar-se-á o algoritmo básico de controle PID, alguns efeitos indesejados implícitos na sua estrutura e a forma de resolver tais problemas (ASTRÖM; HÄGGLUND, 1988). Cabe salientar que não serão investigadas por completo todas as implicações da implementação de um controlador PID. Antes disso, pretende-se analisar alguns dos aspectos práticos necessários para o bom funcionamento do sistema.

O algoritmo básico do controlador PID é mostrado na equação (2.9). Os termos proporcional, integral e derivativo são apresentadas em seguida nas equações (2.10),

(2.11) e (2.12).

$$PID(k \cdot Ts) = P(k \cdot Ts) + I(k \cdot Ts) + D(k \cdot Ts) \tag{2.9}$$

$$P(k \cdot Ts) = K_P \cdot e(k \cdot Ts) \tag{2.10}$$

$$I(k \cdot Ts) = I((k-1) \cdot Ts) + \frac{K_P \cdot Ts}{T_I} \cdot e(k \cdot Ts) \tag{2.11}$$

$$D(k \cdot Ts) = \frac{K_P \cdot T_D}{Ts} \cdot [e(k \cdot Ts) - e((k-1) \cdot Ts)] \tag{2.12}$$

Os parâmetros utilizados nas equações (2.9) a (2.12) são:

- $P(k \cdot Ts)$ → termo proporcional
- $I(k \cdot Ts)$ → termo integral
- $D(k \cdot Ts)$ → termo derivativo
- Ts → tempo de amostragem
- K_P → ganho do termo proporcional
- $e(k \cdot Ts)$ → erro
- T_I → constante de tempo do termo integral
- T_D → constante de tempo do termo derivativo

No cálculo do termo integral, apresentada na equação (2.11), pode ocorrer um erro de *offset* causando no processo um erro de regime. Este erro é ocasionado quando o processador utiliza um tamanho de palavra pequeno. A segunda parcela da equação (2.11) contém o chamado termo de correção e este é o responsável por fornecer erro de regime nulo. Este termo de correção $(K_P.T_S./T_I)*e(k.T_S)$ que é, normalmente, bem menor que o termo $I((k-1)*T_S)$, pode ser desprezado nos cálculos efetuados pelo processador devido ao erro de arredondamento. Assim, o termo integral, responsável por anular o erro de regime, pode-se apresentar ineficiente caso seja utilizado baixa resolução (ou tamanho de palavra pequeno) no

cálculo do termo integral. Para se ter uma idéia de como este termo pode ser pequeno, considere o seguinte exemplo.

$$Ts = 0{,}02\,s;\quad T_I = 20\text{min} = 1200\text{s}\quad K_P = 0{,}1;$$

$$\frac{K_P \cdot Ts}{T_I} = 1{,}7 \cdot 10^{-6} = 2^{-19{,}2}$$

No exemplo acima, nota-se que é necessário, pelo menos, uma resolução de 20 bits para anular o erro de arredondamento. Portanto, a escolha do tamanho da palavra deve levar em consideração, principalmente, o termo de correção da parte integral do controlador PID (ASTRÖM; HÄGGLUND, 1988).

No cálculo do termo derivativo há que se considerar um aspecto prático simples, mas muito importante. Sabendo-se que o cálculo do termo derivativo é baseado na derivada do erro entre o sinal desejado $r(t)$ e a saída do processo $y(t)$, pode-se examinar melhor o que realmente acontece. Considere a equação (2.13) (ASTRÖM; HÄGGLUND, 1988).

$$\frac{d\,e(t)}{dt} = \left[\frac{d\,r(t)}{dt} - \frac{d\,y(t)}{dt}\right] \tag{2.13}$$

O sinal desejado $r(t)$ é normalmente constante com mudanças abruptas. Ou seja, este sinal não contribui para o termo derivativo. Porém, o termo dy(t)/dt irá mudar drasticamente quando o sinal desejado é alterado. Por esta razão, é comum, na prática, aplicar a ação derivativa somente à saída do processo. Então, o termo derivativo, diferentemente da equação (2.12), é implementado como na equação (2.14) (ASTRÖM; HÄGGLUND, 1988).

$$D(k \cdot Ts) = \frac{K_P \cdot T_D}{Ts} * \left[y((k-1) \cdot Ts) - y((k \cdot Ts))\right] \tag{2.14}$$

Dessa forma, para o bom funcionamento de um controlador PID digital deve-se considerar os efeitos da implementação em sistemas controlados por computador como os erros por arredondamento (principalmente no termo de correção da parte integral), resolução dos conversores A/D e D/A e a modificação no termo derivativo. Existem outras considerações práticas para a implementação de controladores PID em sistemas digitais, entretanto, avaliou-se neste trabalho apenas as que foram necessárias até o presente momento.

3. MATERIAIS E MÉTODOS

Descrever-se-á neste capítulo os materiais utilizados e os procedimentos adotados no decorrer do desenvolvimento deste trabalho. Serão abordados o projeto do controlador PID, a escolha dos materiais e a simulação do sistema em malha fechada para comprovar o seu funcionamento.

3.1 PROJETO DO CONTROLADOR PID

O objetivo ao se projetar o controlador PID para a estimulação elétrica neuromuscular funcional, é, definir o tempo de estabelecimento da força muscular. Inicialmente, projetou-se um controlador analógico PID e, em seguida, realizou-se o projeto do controlador digital.

3.1.1 PROJETO DO CONTROLADOR PID ANALÓGICO

Neste trabalho, o projeto do controlador PID analógico foi realizado através do método de alocação de pólos proposto por Aström e Hägglund (1988). O controlador PID proposto é como mostrado na Figura 2.4.

A função de transferência da planta é obtida a partir da equação diferencial apresentada na equação (1.1). Esta função de transferência contém dois pólos e nenhum zero, conforme o rearranjo mostrado na equação (3.1). Já a função de transferência do controlador PID é dada pela equação (3.2). Na equação (3.2), deve-se considerar que:

$$K_D = K_P * T_D$$

$$K_I = \frac{K_P}{T_I}$$

$$G_P(s)=\frac{K}{(1+s\cdot T_1)\cdot(1+s\cdot T_2)} \tag{3.1}$$

$$G_C(s)=K_P+\frac{K_I}{s}+K_D\cdot s=\frac{K_D\cdot s^2+K_P\cdot s+K_I}{s} \tag{3.2}$$

Assim sendo, a equação característica do sistema em malha fechada ilustrado na Figura 2.4 é obtido a partir do denominador da função de transferência de malha fechada. A função de transferência do sistema em malha fechada $G_{FTMF}(s)$ é apresentada na equação (3.3).

$$G_{FTMF}(s)=\frac{G_C(s)\cdot G_P(s)}{1+G_C(s)\cdot G_P(s)} \tag{3.3}$$

A equação característica deste sistema é dada pela equação $1+G_C(s)\cdot G_P(s)$ e é utilizada para modificar os parâmetros do controlador a fim de obter o desempenho desejado. Neste caso, a equação característica é obtida a partir das equações (3.1) e (3.2) e é apresentada na equação (3.4).

$$s^3+\left(\frac{1}{T_1}+\frac{1}{T_2}+\frac{K\cdot K_P\cdot T_D}{T_1\cdot T_2}\right)\cdot s^2+\left(\frac{1}{T_1\cdot T_2}+\frac{K\cdot K_P}{T_1\cdot T_2}\right)\cdot s+\frac{K\cdot K_P}{T_I\cdot T_1\cdot T_2} \tag{3.4}$$

No método de alocação de pólos, devem-se escolher os pólos desejados, formando uma equação característica. Em seguida, calculam-se os valores de K_P, T_I ou K_I e T_D ou K_D, para que se obtenha a função de transferência $G_{FTMF}(s)$ com os mesmos pólos impostos pela equação característica. A equação característica para este sistema é dada pela equação (3.5).

$$(s+\alpha\cdot\omega)\cdot(s^2+2\cdot\xi\cdot\omega\cdot s+\omega^2) \tag{3.5}$$

Igualando-se as equações (3.4) e (3.5), obtêm-se os valores de K_P, K_I e K_D a partir das equações (3.6), (3.7) e (3.8).

$$K_P=\frac{T_1\cdot T_2\,\omega^2\cdot(1+2\cdot\xi\cdot\alpha)-1}{K} \tag{3.6}$$

$$K_I = \frac{T_1 \cdot T_2 \omega^2 \cdot (1 + 2 \cdot \xi \cdot \alpha) - 1}{T_1 \cdot T_2 \cdot \alpha \cdot \omega^3} \tag{3.7}$$

$$K_D = \frac{T_1 \cdot T_2 \cdot \omega \cdot (\alpha + 2 \cdot \xi) - T_1 - T_2}{\omega^2 \cdot T_1 \cdot T_2 \cdot (1 + 2 \cdot \xi \cdot \alpha) - 1} \tag{3.8}$$

Então, os parâmetros de configuração do controlador PID são calculados a partir dos parâmetros da planta e das especificações desejadas. Além disso, deve-se atentar para que os parâmetros do controlador PID não resultem em valores negativos.

Nesse contexto, as variáveis da equação característica desejada podem ser dadas em função do desempenho desejado do sistema. Para uma entrada do tipo degrau, os parâmetros de desempenho desejados podem ser o máximo de sobre-sinal aceitável calculado pela equação (3.9), o tempo de estabelecimento para o critério de 2% na equação (3.10) e o tempo de subida conforme a equação (3.11) Ogata (2003). O parâmetro α da equação (3.5) deve ser escolhido a fim de enfraquecer a atuação do pólo $(s + \alpha \cdot \omega)$, ou seja, deve-se escolher um valor de α suficientemente grande.

$$sobre\ sinal = e^{\frac{-\xi \cdot \pi}{\sqrt{1-\xi^2}}} \cdot 100\% \tag{3.9}$$

$$tempo\ estabelecimento = \frac{4}{\xi \cdot \omega_n} \tag{3.10}$$

$$tempo\ de\ subida = \frac{pi - arcsen\left(\sqrt{(1-\xi^2)}\right)}{\omega_n * \sqrt{(1-\xi^2)}} \tag{3.11}$$

Por fim, nota-se que o projeto do controlador PID analógico pode ser realizado de forma automática com o método de alocação de pólos. Ou seja, um programa computacional pode calcular os parâmetros do controlador PID adequado para a planta e as especificações desejadas pelo usuário utilizando-se as equações (3.6), (3.7) e (3.8). A simulação deste sistema pode ser realizada utilizando, por exemplo, o ambiente Simulink, como ilustrado na Figura 3.1.

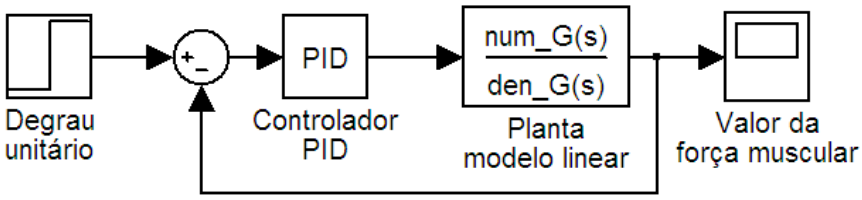

Figura 3.1 - Simulação de um sistema em malha fechada com controlador PID no Simulink

3.1.2 PROJETO DO CONTROLADOR PID DIGITAL

O projeto do controlador digital pode ser feito de duas maneiras básicas: utilizando a teoria de controle analógico ou a teoria de controle digital. A primeira maneira consiste em projetar o controlador analógico e, em seguida, utilizar estes valores de forma apropriada no controlador digital. A segunda maneira de se projetar um controlador digital é utilizando métodos discretos.

Entretanto, é necessário, primeiramente, entender como é a arquitetura de um sistema de controle digital. Apresenta-se na Figura 3.2 um diagrama de blocos que representa um sistema em malha fechada com controlador digital.

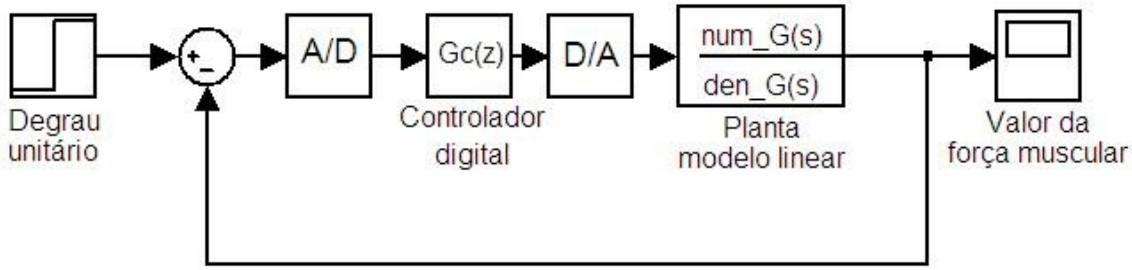

Figura 3.2 - Esquema básico de um sistema em malha fechada com controlador digital.

É intuitivo afirmar que os conversores A/D e D/A influenciarão a resposta do sistema. Então, na simulação deste sistema deve-se avaliar os seus efeitos. O primeiro efeito é o da discretização no tempo, podendo ser simulada com a utilização dos blocos amostradores (ou grampeadores) do ambiente Simulink. Porém, ainda não se pode chamar o sistema implementado de digital, mas, apenas, de discreto no tempo. Para ter-se a simulação do sistema digital deve-se inserir também um segundo efeito, o da quantização dos conversores. Este efeito pode ser simulado utilizando os blocos quantizadores do ambiente Simulink. Apresenta-se na Figura 3.3 o diagrama de blocos utilizado para simular o sistema

digital no ambiente Matlab/Simulink, incluindo os efeitos dos conversores e a mudança no termo derivativo, conforme apresentado na equação (2.14).

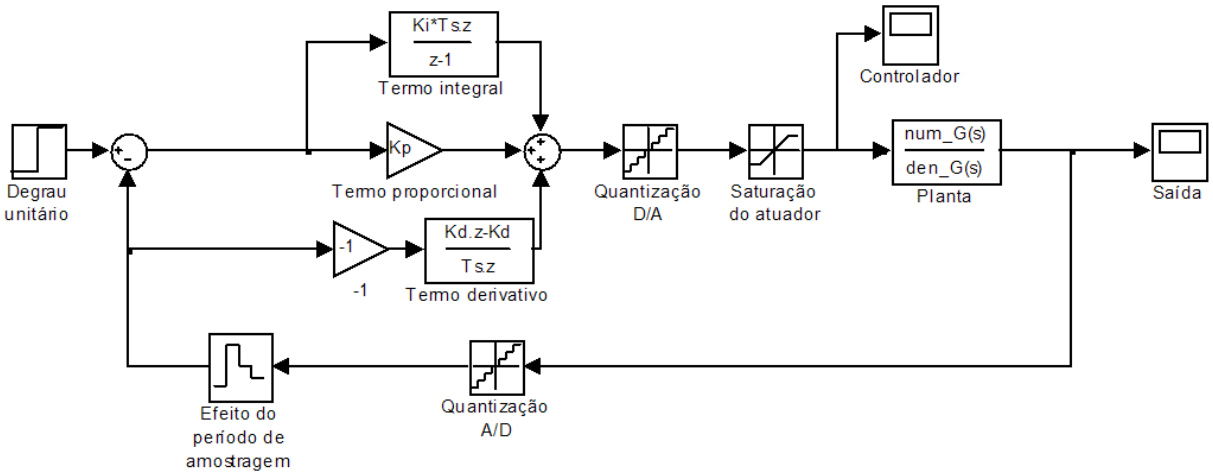

Figura 3.3 - Diagrama de blocos do sistema em malha fechada utilizado para a avaliação do controlador PID projetado.

Dessa forma, os resultados obtidos da simulação no ambiente Matlab/Simulink possibilitaram realizar os ajustes e a validação do controlador PID projetado. O efeito do tamanho da palavra, inerente em sistemas controlados por computador, não foi simulado neste ambiente, porém, pôde-se comprovar o bom funcionamento do controlador na implementação do mesmo, como será visto no capítulo 4.

3.1.2.1 CONTROLADOR ANALÓGICO DISCRETIZADO

O método de projeto de controladores discretos PID utilizando os parâmetros de um controlador PID analógico é bem prático, já que se utilizam os mesmos parâmetros em ambos os sistemas. Isto é possível graças à equivalência dos dois sistemas. É possível notar esta equivalência entre as equações (2.6) e (2.9). Fica evidente que o controlador PID analógico realiza as mesmas operações que o controlador PID discreto. Ademais, é de se esperar que, para tempos de amostragens bastante pequenos, a resposta do controlador PID discreto seja praticamente igual à do controlador PID analógico.

Assim, neste método utiliza-se o sistema com o controlador PID digital com os mesmos parâmetros K_P , K_I ou T_I e K_D ou T_D calculados para o controlador PID analógico.

3.1.2.2 CONTROLADOR DISCRETO PROJETADO PELO LUGAR DAS RAÍZES NO PLANO Z

O primeiro passo do projeto de um controlador discreto no plano Z é determinar a função de transferência na variável Z que representa a planta. Há vários métodos para o cálculo da representação discreta da planta no plano Z. O método utilizado neste trabalho considera a função de transferência no plano S de um grampeador de ordem zero ($G_{OZ}(s)$) em série com a função de transferência no plano S da planta ($G_P(s)$). Por fim, realiza-se a transformada Z dessa combinação de funções de transferências em série. Ou seja, a função de transferência da planta discretizada no plano Z é a transformada Z do produto $G_{OZ}(s)\cdot G_P(s)$. A função de transferência da planta discretizada no plano Z possui dois pólos e um zero.

É relevante salientar que a inserção da função de transferência do amostrador $G_{OZ}(s)$ faz com que esta discretização seja dependente do período de amostragem Ts . O ambiente Matlab possui a função *c2d*, com a opção "zoh", para realizar esta discretização.

O projeto de um controlador discreto do tipo PID pelo método de alocação de pólos no plano Z pode ser realizado de forma análoga ao do plano S. Apresenta-se, então, na equação (3.12) a função de transferência na variável Z de um controlador PID discreto.

$$PID(z) = K_P \cdot \left[1 + \frac{T_D}{Ts} \cdot (1 - z^{-1}) + \frac{Ts}{T_I} \cdot \frac{1}{1 - z^{-1}} \right] \tag{3.12}$$

O controlador PID no plano Z insere, no sistema em malha aberta, dois zeros, um pólo no círculo unitário (análogo à origem do plano S) e um pólo na origem do plano Z. Pode-se rearranjar a função de transferência do controlador PID discreto como a equação (3.13), para tornar mais evidente os dois zeros em malha aberta implementados pelo controlador PID.

$$PID(z) = K \cdot \frac{\left(1 - z_1 \cdot z^{-1}\right) \cdot \left(1 - z_2 \cdot z^{-1}\right)}{1 - z^{-1}} = \frac{a_0 \cdot z^2 + a_1 \cdot z + a0}{z \cdot (z - 1)} \tag{3.13}$$

Assim, pode-se escolher os zeros do controlador PID a fim de cancelar os pólos da função de transferência discretizada da planta e, em seguida, escolher o valor de

K apropriado. O sistema em malha aberta é equivalente a um sistema com dois pólos, um no círculo unitário e outro na origem, e, um zero (fornecido pela discretização da planta). Considerando que o pólo na origem do plano Z tem pouca influência no comportamento dos sistema, pode-se dizer que conforme aumenta-se o valor de K, o pólo do sistema em malha fechada se distancia do círculo unitário em direção a origem do plano Z.

Resumindo, o método de projeto do controlador PID discreto por alocação de pólos consiste em fazer os zeros do controlador PID iguais aos pólos da função de transferência discretizada da planta e escolher K de acordo com as especificações desejadas informadas pelo usuário. Depois de determinados os valores de a_0, a_1 e a_2 determinam-se os parâmetros do controlador PID utilizando-se as equações (3.14), (3.15) e (3.16).

$$K_D = a_2 \cdot Ts \tag{3.14}$$

$$K_P = -\left(a_1 + \frac{2 \cdot K_D}{Ts}\right) \tag{3.15}$$

$$K_I = \left(a_0 - K_P - \frac{K_D}{Ts}\right) \div Ts \tag{3.16}$$

3.2 MATERIAIS

Um dos objetivos deste trabalho é realizar um projeto com baixo custo que possibilite a implementação do mesmo utilizando a tecnologia de microcontroladores. Portanto, neste tópico abordar-se-á a escolha dos microcontroladores e dos conversores A/D e D/A necessários para a implementação do projeto. Adicionalmente, far-se-á uma breve descrição do ambiente de simulação utilizado para comprovar o bom funcionamento do controlador PID projetado com os componentes escolhidos.

3.2.1 ESCOLHA DO MICROCONTROLADOR

Atualmente, os microcontroladores possuem diversos periféricos e um processador embutidos em um único circuito integrado. Esta característica minimiza o

tamanho físico dos projetos e facilita a implementação de várias aplicações. Neste projeto, os elementos necessários são o processador, conversor A/D, conversor D/A e um temporizador. Assim, nas subseções a seguir descrever-se-ão os itens que auxiliaram na escolha do microcontrolador e dos periféricos.

3.2.1.1 ESCOLHA DO TAMANHO DA PALAVRA

Qual o tamanho de palavra necessário para que o sistema controlado por computador apresente bom desempenho? Pode-se começar estudando a precisão necessária para a implementação dos parâmetros do controlador PID. Os parâmetros são os ganhos K_P , K_I e K_D . O uso destes parâmetros é mostrado nas equações (2.10), (2.11) e (2.14), respectivamente.

No entanto, é necessário considerar-se, também, o termo de correção da parte integral apresentada na equação (2.11), que é geralmente muito pequeno devido à contribuição da constante Ts , ou tempo de amostragem.

A escolha adequada do tamanho da palavra é realizada analisando-se o menor destes parâmetros. Por exemplo, caso o menor destes parâmetros seja da ordem de $7{,}5\cdot10^{-5}$ seriam necessárias, pelo menos, 5 casas decimais de precisão e, consequentemente, são necessários 14 bits, já que $7{,}5\cdot10^{-5}\approx2^{-13{,}7}$. Considerando-se então o bit de sinal, são necessários, então, 15 bits de precisão.

Diante disso, como visto na subseção 2.2.2 , os compiladores são padronizados de acordo com o padrão IEEE-ANSI 754 e permitem a implementação de variáveis do tipo ponto flutuante com 32 bits, sendo 8 bits para o expoente, 1 bit para o sinal e 23 bits para a mantissa. A precisão fornecida pelo compilador refere-se diretamente à quantidade de bits reservada para a mantissa.

3.2.1.2 ESCOLHA DOS CONVERSORES A/D E D/A

A escolha da resolução dos conversores A/D e D/A basicamente se baseia na precisão desejada na saída do processo. Além disso, é necessário o conhecimento da faixa de variação da tensão analógica na saída do processo e na saída do controlador. Para isso,

realizou-se a simulação do sistema em malha fechada com a função *dstep* do Matlab.

Em seguida, deve-se aplicar as equações (2.7) e (2.8) a fim de obter a resolução necessária para os conversores A/D e D/A. O parâmetro "ganho estático da planta", necessário na equação (2.8), pode ser obtido utilizando o comando *dcgain* do Matlab.

3.2.2 AMBIENTE DE SIMULAÇÃO PROTEUS

O ambiente de simulação Proteus permite a simulação de circuitos eletrônicos analógicos e digitais de forma simultânea. Sendo assim, é possível avaliar, por exemplo, se o controlador PID projetado funciona de forma adequada. O que é relevante é que o usuário usa exatamente o microcontrolador desejado, com os conversores A/D e D/A de modelos específicos para avaliar se o seu projeto está correto. Então, as características dos dispositivos são inseridas como parâmetros da simulação, permitindo que o projetista desenvolva o seu primeiro protótipo com uma segurança maior.

Afim de possibilitar uma simulação completa, o ambiente Proteus fornece blocos de função de transferência com entrada e saída analógica para simulação de plantas, osciloscópio, multímetros e o terminal virtual que simula a comunicação entre um PC e o microcontrolador através da porta serial.

Assim, ferramentas, como o ambiente Proteus, podem ser extremamente úteis para agilizar o tempo gasto em pesquisa e desenvolvimento de protótipos.

4. IMPLEMENTAÇÃO E RESULTADOS

Descrever-se-ão neste capítulo o programa computacional desenvolvido para automatizar o projeto do controlador PID, a implementação do controlador PID e os resultados obtidos. Inicialmente, projetou-se um sistema embarcado apenas com o controlador PID para comprovar a viabilidade da proposta. Em seguida, agregou-se ao sistema funções que permitem a configuração automática do controlador PID embarcado a partir de um computador, não exigindo que o pesquisador modifique o código fonte, em linguagem C, do microcontrolador.

4.1 PROGRAMA EENM-PID

O programa EENM-PID foi desenvolvido inicialmente utilizando-se o ambiente de programação Matlab e, posteriormente, a linguagem Python. Nesse trabalho será abordado o uso do programa desenvolvido com a linguagem Python, entretanto, a utilização do programa no ambiente Matlab é fácil e intuitiva[6].

O objetivo do programa EENM-PID é calcular os parâmetros do controlador PID a fim de controlar uma planta baseada num modelo linear com dois pólos e nenhum zero. Adicionalmente, o programa realiza a simulação da resposta do sistema em malha fechada e mostra os resultados em um gráfico. São fornecidas também as configurações mínimas necessárias dos conversores A/D e D/A.

A tela principal é apresentada na Figura 4.1. Os três botões realizam todas as operações fornecidas pelo programa. Os botões "Parâmetros da planta" e "Especificações do controlador digital PID" possuem um utilitário simples para configurar os parâmetros da planta (paciente) e as configurações e especificações do controlador PID, respectivamente. O botão "Projeto e simulação" calcula os parâmetros do controlador PID, a configuração mínima dos conversores A/D e D/A e realiza a simulação do sistema em malha fechada em resposta a

6 O programa desenvolvido em Matlab "EENM-PID.m" necessita do arquivo "parametros.m". Estes arquivos estão disponíveis no apêndice.

uma entrada degrau unitário.

Os utilitários de configuração dos parâmetros da planta e da configuração das especificações do controlador digital PID são apresentados nas figuras 4.2 e 4.3, respectivamente. Os resultados são apresentados nas figuras 4.4 e 4.5.

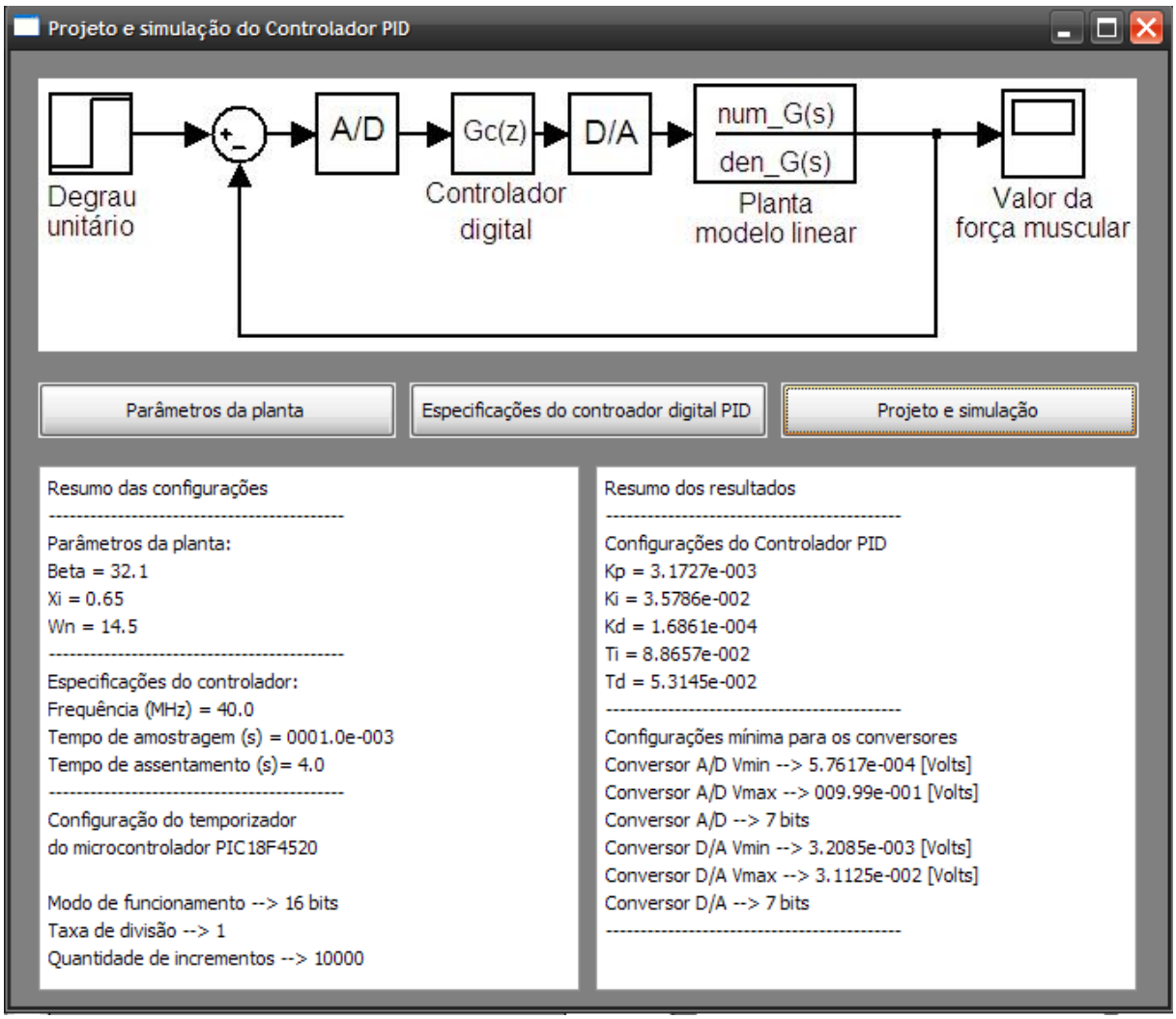

Figura 4.1 - Tela principal do programa EENM-PID.

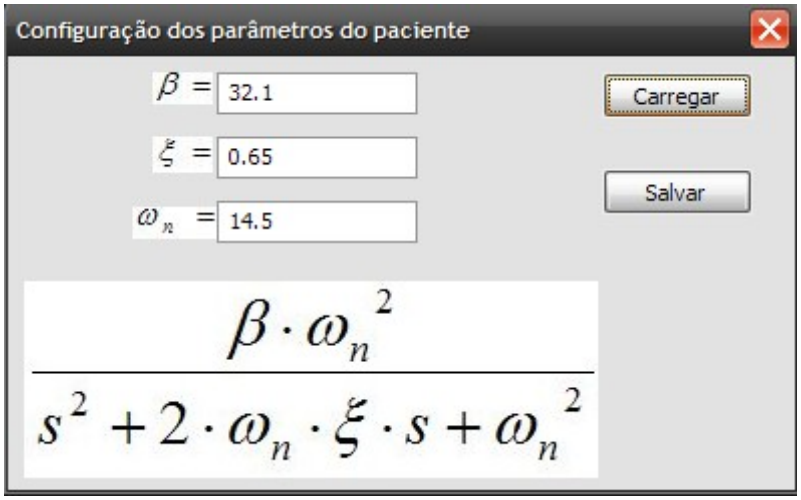

Figura 4.2 - Utilitário de configuração dos parâmetros da planta.

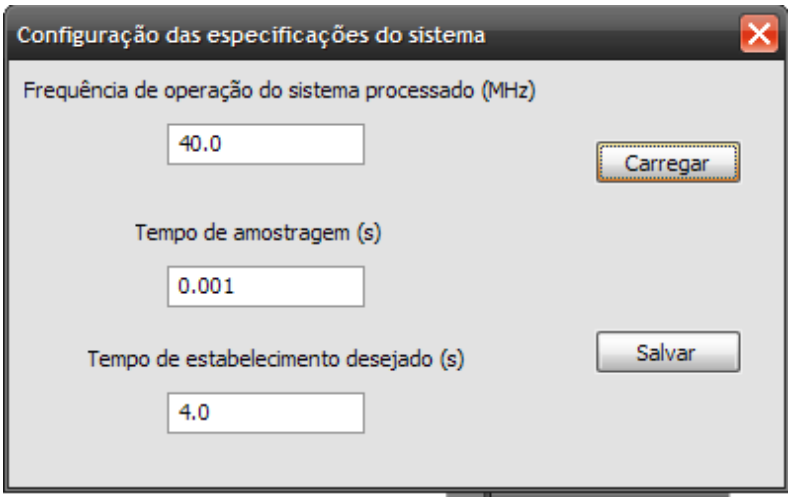

Figura 4.3 - Utilitário de configuração das especificações do controlador PID.

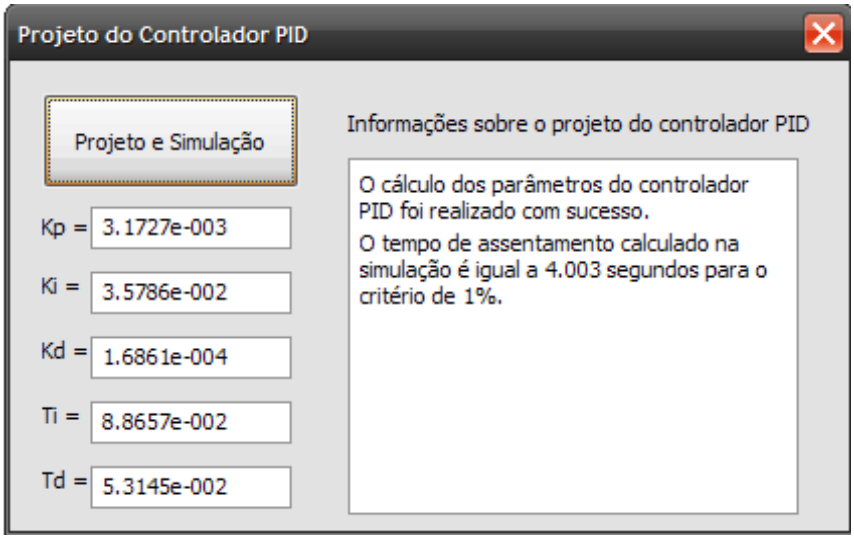

Figura 4.4 - Resultados do projeto do controlador PID.

Ao final da simulação, a tela principal do programa EENM-PID apresenta um resumo dos resultados com todas as informações necessárias para se implementar o controlador digital PID. O EENM-PID foi utilizado nos projetos dos controladores PID apresentados neste capítulo, como será visto nas subseções a seguir.

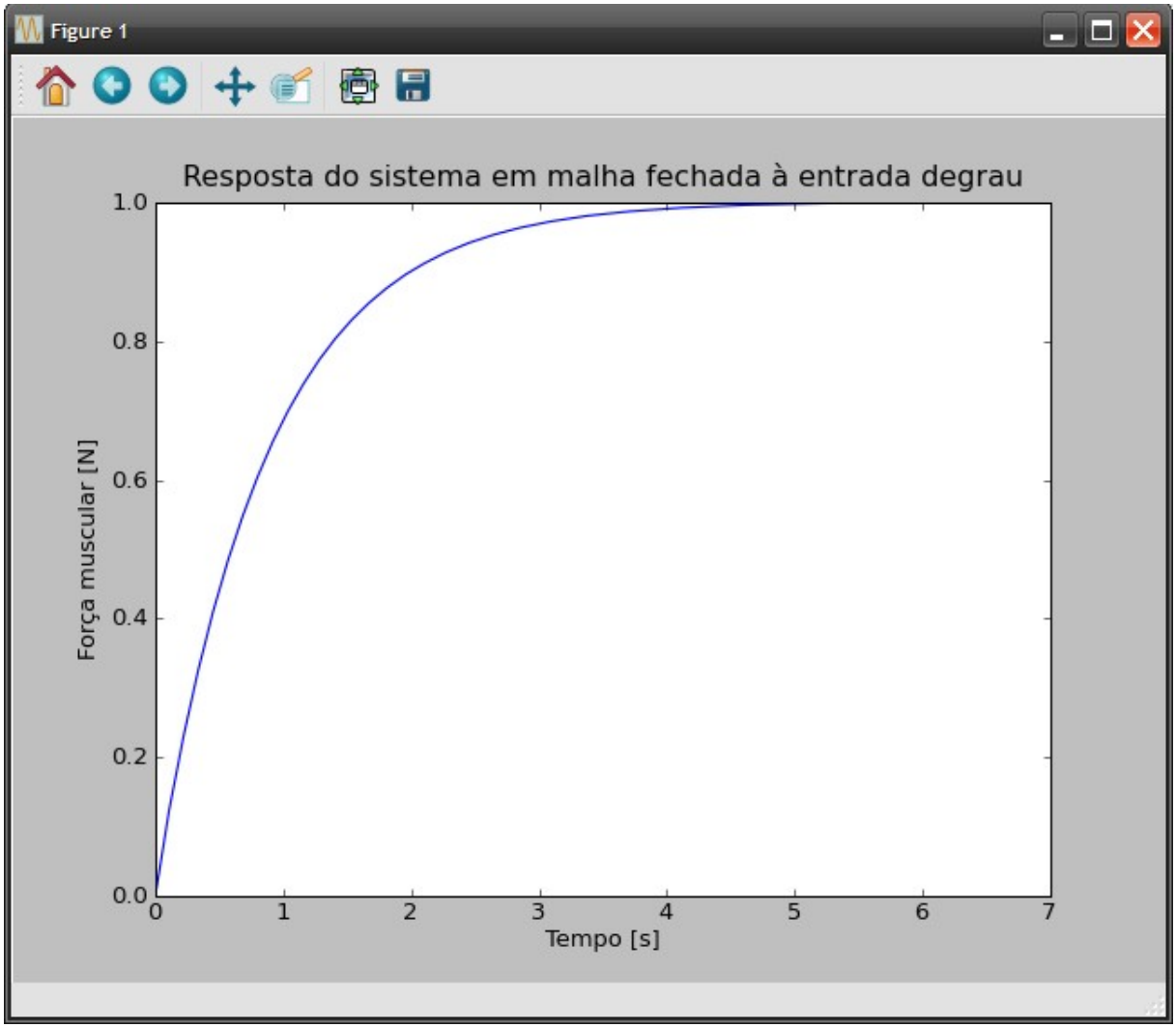

Figura 4.5 - Resultados da simulação do sistema em malha fechada.

4.2 PROJETO E IMPLEMENTAÇÃO DO CONTROLADOR PID

O projeto do controlador PID foi automatizado com o desenvolvimento do programa EENM-PID, utilizando os fundamentos teóricos apresentados no item 3.1.2 . Assim, apresentar-se-á neste tópico o projeto realizado com este programa, a simulação do

mesmo utilizando um diagrama de blocos implementado no ambiente Matlab/Simulink (Figura 3.3) e a sua implementação utilizando o microcontrolador PIC16F877A. Em seguida, será apresentado algumas modificações no sistema utilizando o microcontrolador PIC18F4520 com o objetivo de facilitar o uso do mesmo por pesquisadores da área de engenharia de reabilitação.

4.2.1 PROJETO DO CONTROLADOR PID

O primeiro controlador PID foi projetado utilizando a função de transferência mostrada na equação (4.1), a partir de dados coletados do trabalho realizado por Law e Shields (2006). As especificações desejadas são tempo de assentamento igual a 3 segundos, tempo de amostragem igual a 1 ms e frequência de operação do microcontrolador igual a 40 MHz.

$$G_P(s) = \frac{6324}{s^2 + 19{,}87 \cdot s + 207{,}4} \tag{4.1}$$

Dessa forma, configurou-se o programa EENM-PID com estes dados de entrada e os parâmetros do controlador PID calculados são apresentados a seguir e a simulação deste sistema em malha fechada é apresentada na Figura 4.6.

$K_P = 4{,}7605 \cdot 10^{-3}$

$K_I = 5{,}0198 \cdot 10^{-2}$

$K_D = 2{,}3969 \cdot 10^{-4}$

$T_I = 9{,}4835 \cdot 10^{-2}$

$T_D = 5{,}0349 \cdot 10^{-2}$

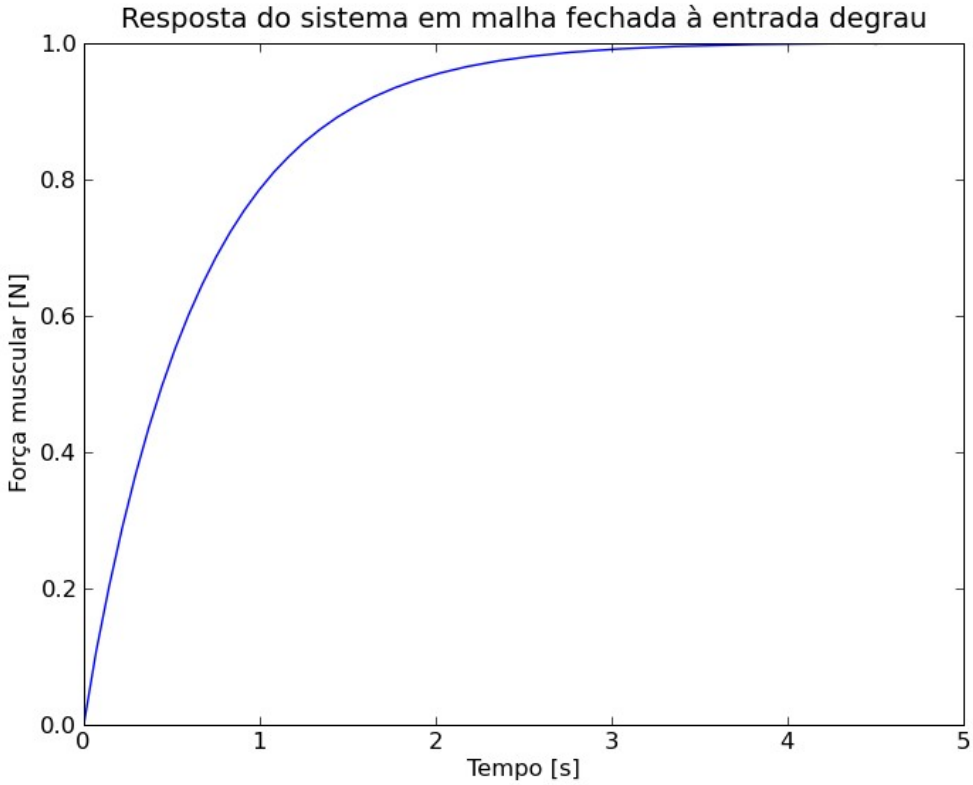

Figura 4.6 - Simulação inicial do sistema em malha fechada.

Analisando a simulação realizada, observa-se que o sistema não apresenta sobre-sinal e que seu tempo de assentamento é, aproximadamente, igual a 3 segundos. Sendo assim, pode-se projetar os outros itens do sistema, como os conversores e o microcontrolador.

4.2.1.1 ESCOLHA DO TAMANHO DA PALAVRA

Utilizando a metodologia apresentada no capítulo 3 analisou-se, inicialmente, o menor dos parâmetros do controlador PID. O parâmetro K_D é igual $2{,}3969 \cdot 10^{-4}$. Portanto, é necessário um tamanho de palavra igual a, pelo menos, 14 bits, sendo 13 bits para a mantissa e um bit para o bit de sinal.

Mas, deve-se analisar, também, o termo de correção da parte integral apresentada na equação (2.11). Neste caso, o termo de correção é apresentado a seguir.

$$Ts = 0{,}001\,;\ T_I = 9{,}4835 \cdot 10^{-2}\,;\ K_P = 4{,}7605 \cdot 10^{-3}\,;$$

$$\frac{K_P \cdot Ts}{T_I} = 5{,}0198 \cdot 10^{-5} \approx 2^{-14{,}28}$$

Então, é necessário um tamanho de palavra igual a, pelo menos, 16 bits, sendo 15 bits para a mantissa e um bit para o bit de sinal. Diante disso, os compiladores para os microcontroladores PIC, em linguagem C que seguem o padrão ANSI-IEEE 754 possibilitam a utilização de variáveis do tipo ponto flutuante com 32 bits, sendo 23 bits para a mantissa.

4.2.1.2 ESCOLHA DA RESOLUÇÃO DOS CONVERSORES

O programa EENM-PID apresenta como resultados a faixa de variação de tensão da saída da planta e da saída do controlador que influenciarão a faixa de operação do conversor A/D e do conversor D/A, respectivamente. No entanto, estes valores, em geral, não são praticáveis. Portanto, deve-se aproximá-los para valores que possam ser implementados com fontes de tensão reguláveis.

Além disso, para a validação do processo de escolha da resolução dos conversores A/D e D/A foi utilizado o modelo apresentado na Figura 3.3, que permite avaliar os efeitos dos conversores. Adicionalmente, desenvolveu-se um programa no ambiente Matlab denominado *configuracao_conversores.m,* para auxiliar nessa avaliação dos efeitos de quantização dos conversores.

Observando-se os resultados apresentados pelo programa EENM-PID, escolheu-se como faixa de variação do conversor A/D a faixa de tensão de -0,1V a +1,1V. Determinando que a precisão desejada seja de 1%, tem-se que a resolução do conversor A/D deve ser igual ou maior ao valor calculado pela equação (2.7).

$$bits_{AD} = \log_2\left(\frac{\Delta V}{precisão}\right) = \log_2\left(\frac{+1{,}1-(-0{,}1)}{0{,}01*1{,}0}\right) = \log_2\left(\frac{1{,}2}{0{,}01}\right) = 6{,}9069 \approx 7$$

Similarmente, observando-se os resultados apresentados pelo programa EENM-PID, escolheu-se como faixa de variação do conversor D/A a faixa de tensão de -0,1V a +0,1V. Determinando que a precisão desejada seja de 1%, tem-se que a resolução do conversor D/A deve ser igual ou maior ao valor calculado pela equação (2.8).

$$bits_{DA} = \log_2\left(\frac{(\Delta V)\cdot ganho}{precisão}\right) = \log_2\left(\frac{(+0{,}1-(-0{,}1))*30{,}5}{0{,}01*1{,}0}\right) = \log_2\left(\frac{6{,}1}{0{,}01}\right) = 9{,}2527 \approx 10$$

Por fim, realiza-se a simulação utilizando o modelo apresentado na Figura 3.3 considerando os efeitos dos conversores A/D e D/A. Como é possível observar-se na simulação apresentada na Figura 4.7, o sistema em malha fechada resultante apresenta o desempenho desejado.

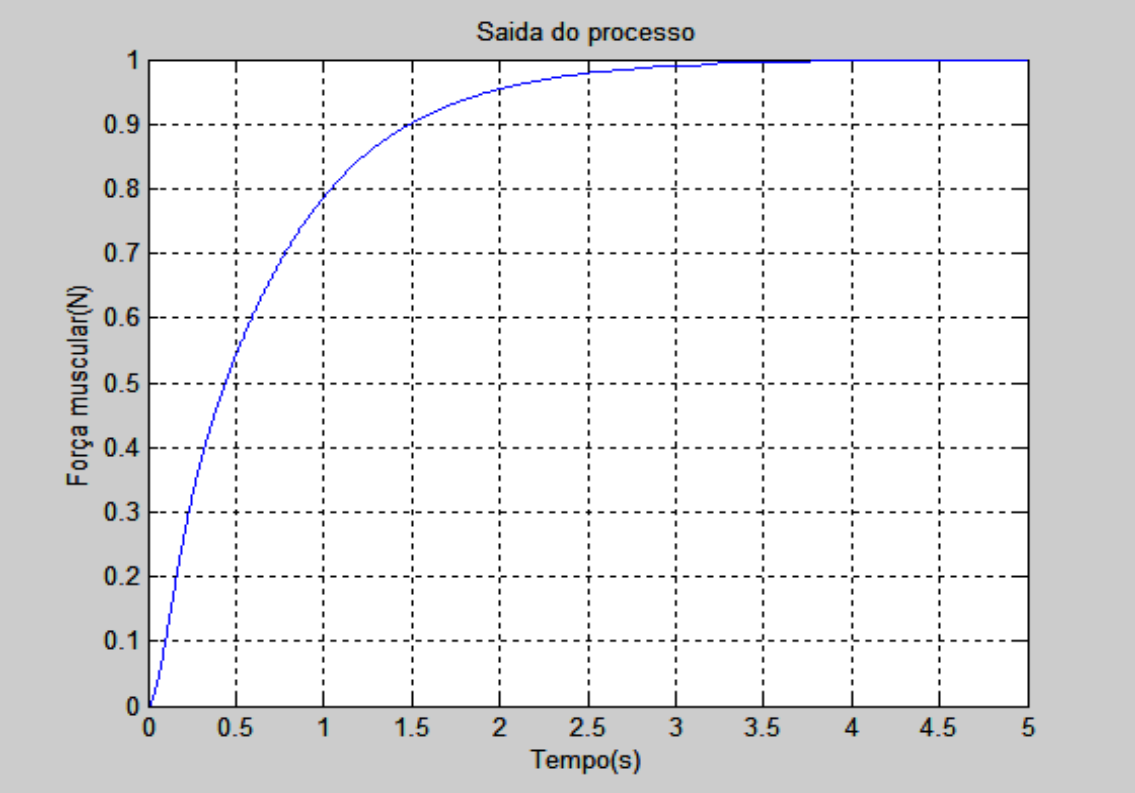

Figura 4.7 - Resposta do sistema em malha fechada considerando os efeitos dos conversores A/D e D/A.

4.2.2 IMPLEMENTAÇÃO DO CONTROLADOR PID

A implementação do controlador PID foi realizada em duas etapas. A primeira etapa consistiu em implementar um controlador simples utilizando o microcontrolador PIC16F877A que contém as especificações necessárias. Na segunda etapa, aperfeiçoou-se o sistema inserindo recursos de configuração automática do controlador a partir do PC, para que o pesquisador não tenha necessidade de modificar o código-fonte do microcontrolador. Para esta segunda etapa utilizou-se o microcontrolador PIC18F4520. Por

fim, realizaram-se alguns testes com o sistema implementado, variando-se significativamente o tempo de amostragem do controlador.

4.2.2.1 IMPLEMENTAÇÃO DO CONTROLADOR PID COM O MICROCONTROLADOR PIC16F877A

A implementação do sistema em malha fechada foi realizada no ambiente Proteus conforme ilustrado na Figura 4.8. É possível notar que a planta foi simulada com um bloco de função de transferência disponível no Proteus e que o único periférico é o conversor D/A. As fontes de tensão CC presentes nesta implementação são as referências para o bloco conversor D/A e para o conversor A/D interno do PIC16F877A. Por fim, o elemento no extremo direito é o osciloscópio que possibilita visualizar os resultados.

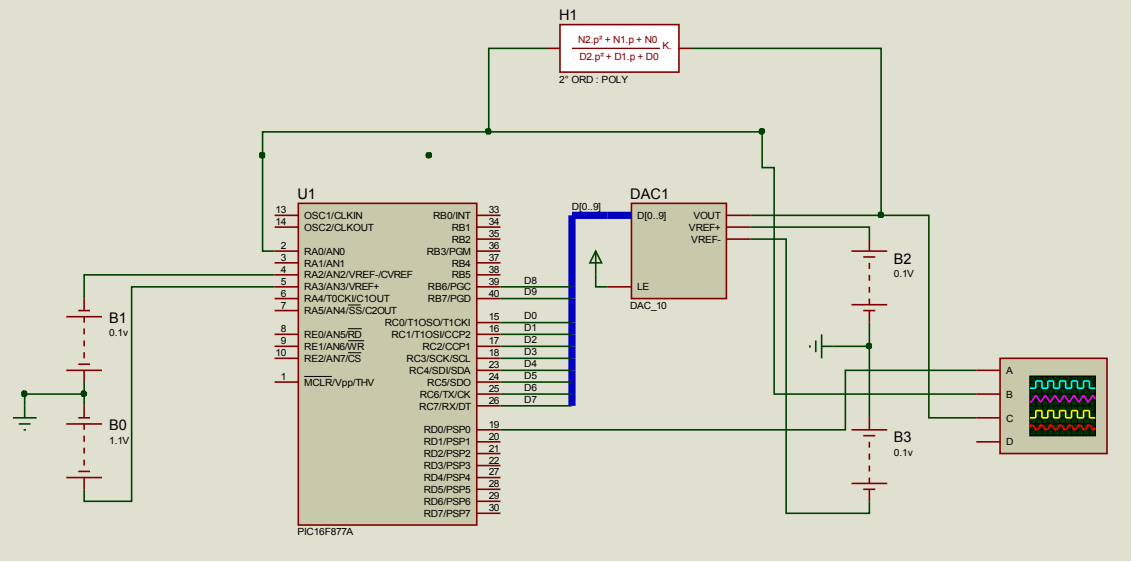

Figura 4.8 - Implementação do controlador PID no ambiente Proteus com o microcontrolador PIC16F877A.

O resultado observado no osciloscópio é apresentado na Figura 4.9. A curva superior (rosa) é a saída do conversor D/A (isto é, a atuação do controlador na planta) e a curva inferior (azul) é a saída da planta. Pode-se observar, a partir dos ajustes realizados no osciloscópio, que a saída da planta estabilizou em 1 Volt e que o tempo de assentamento é aproximadamente igual a 3 segundos.

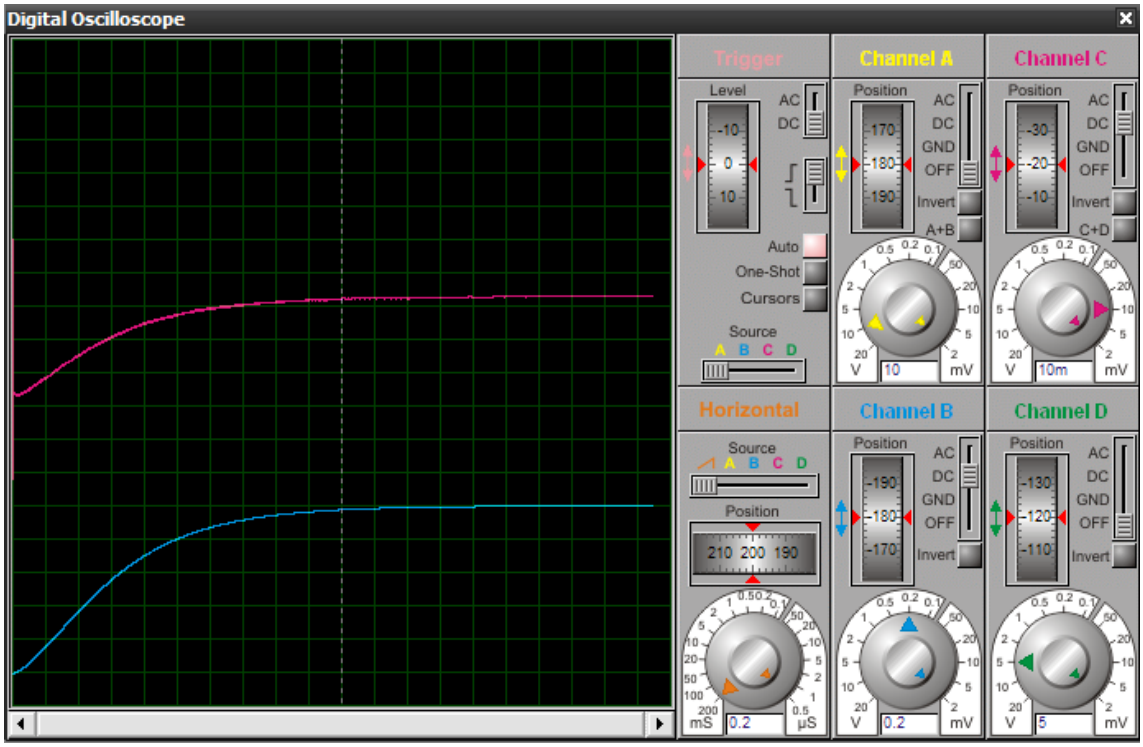

Figura 4.9 - Resultados obtidos com a implementação em malha fechada do controlador PID com o microcontrolador PIC16F877A.

4.2.2.2 IMPLEMENTAÇÃO DO CONTROLADOR PID COM O MICROCONTROLADOR PIC18F4520

A implementação do sistema em malha fechada com o microcontrolador PIC18F4520 também foi realizada no ambiente Proteus conforme ilustrado na Figura 4.10. Este sistema é similar ao sistema com o microcontrolador PIC16F877A, com exceção, da mudança do microcontrolador utilizado e da inserção de um terminal virtual, responsável por simular a comunicação do computador com o microcontrolador através da porta serial (RS-232). Adicionalmente, há uma chave SW1 adicional que seleciona o modo de operação do sistema entre operação normal ou configuração do controlador PID.

No modo de operação normal, a chave SW1 deve estar como mostrado na Figura 4.10, ou seja, transmitindo o nível lógico "1" ao pino RE0 do microcontrolador. Neste caso, o sistema funciona apenas como um simples controlador PID. Inicialmente ele lê os dados da EEPROM interna do microcontrolador e, em seguida, atua na planta de forma apropriada. A resposta da planta apresentado pelo controlador PID no modo de operação normal é apresentado na Figura 4.11. O terminal virtual também mostra ao usuário as

operações que foram realizadas na inicialização do sistema, conforme ilustrado na Figura 4.12.

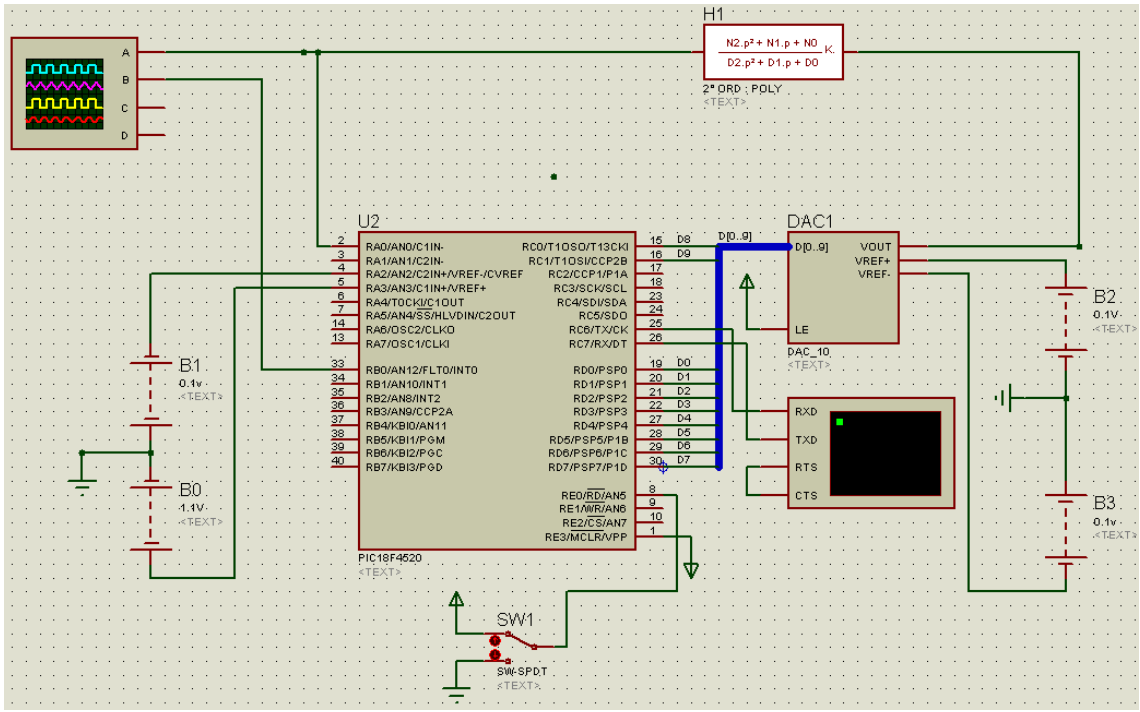

Figura 4.10 - Implementação do controlador PID no ambiente Proteus com o microcontrolador PIC18F4520.

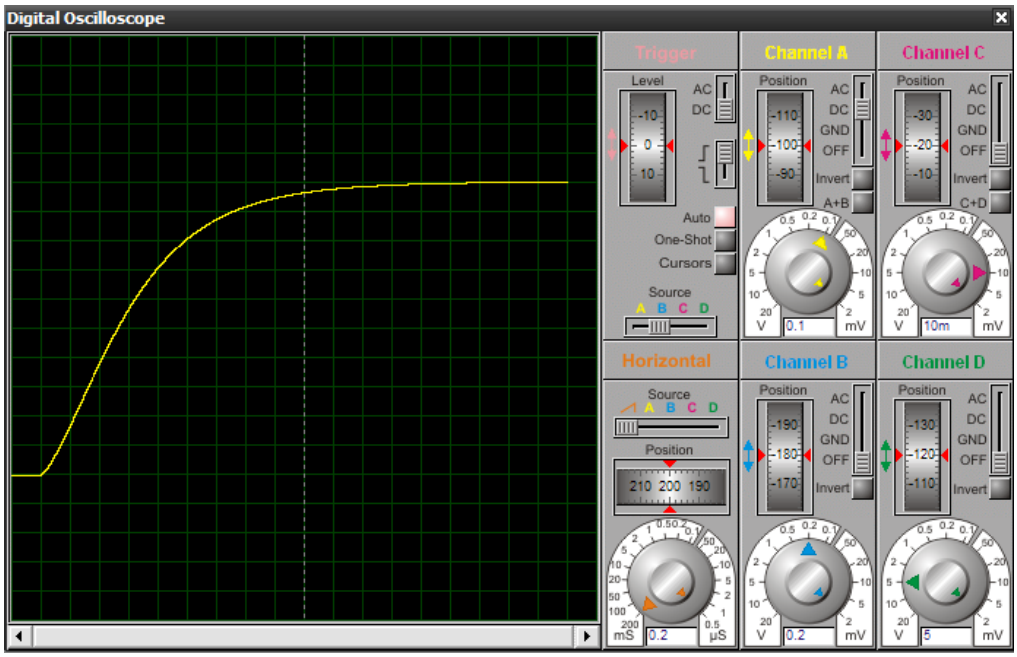

Figura 4.11 - Resposta obtida com a implementação em malha fechada do controlador PID com o microcontrolador PIC18F4520.

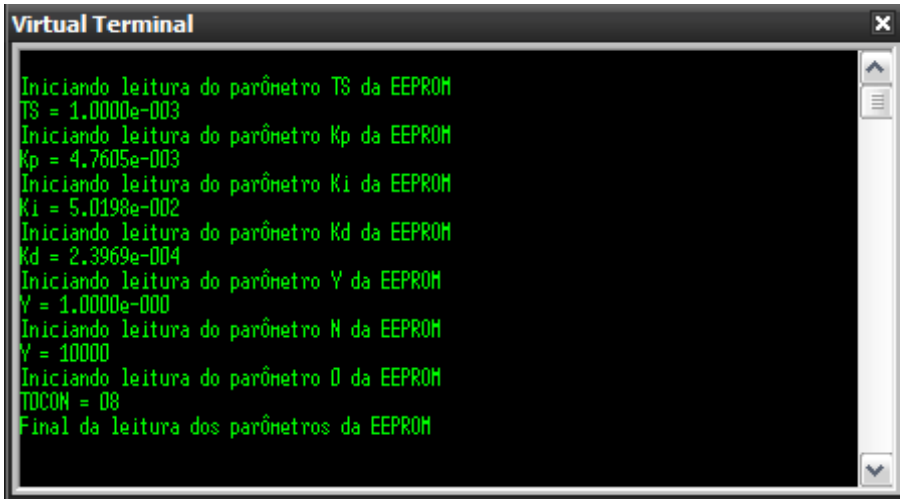

Figura 4.12 - Inicialização dos parâmetros do controlador PID no modo de operação normal.

Além disso, este sistema permite a configuração dos parâmetros do controlador PID. Quando a chave SW1 está no nível lógico "0", o microcontrolador não carrega os parâmetros pré-gravados na memória EEPROM do microcontrolador, mas espera que o usuário digite os parâmetros desejados do controlador PID. Este processo é ilustrado na Figura 4.13.

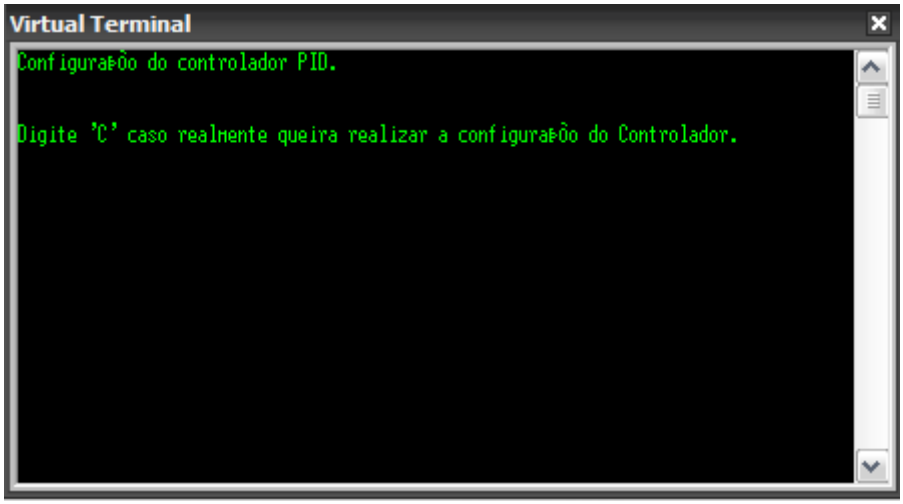

Figura 4.13 - Modo de configuração do controlador PID

Quando o usuário digita a letra "C", inicia-se a configuração do controlador PID. Ilustra-se na Figura 4.14 um exemplo desta configuração. É relevante salientar que o formato dos dados de entrada são compatíveis com o formato dos dados de saída do programa EENM-PID. Em seguida, o usuário deve configurar o temporizador do microcontrolador para que o tempo de amostragem esteja correto. Na Figura 4.15 ilustra-se a configuração do

temporizador do microcontrolador. Os dados de entrada nessa configuração são fornecidos como resultados pelo programa EENM-PID.

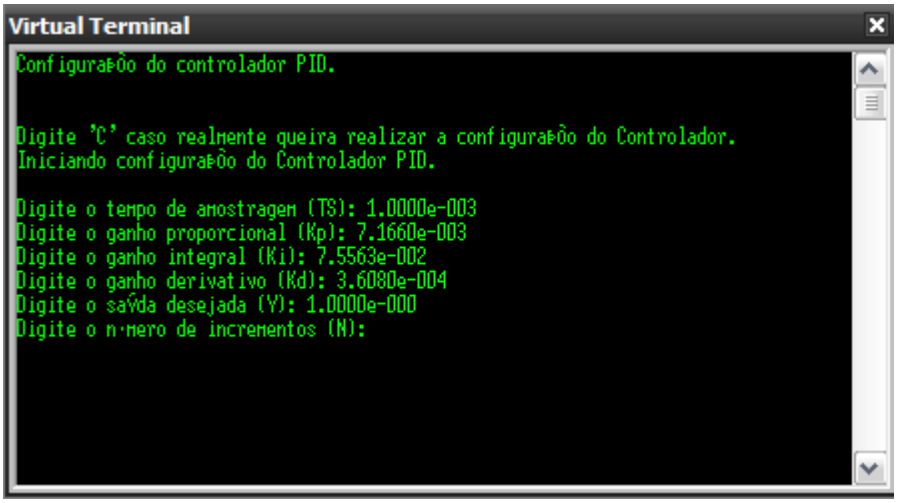

Figura 4.14 - Configuração dos parâmetros do controlador PID

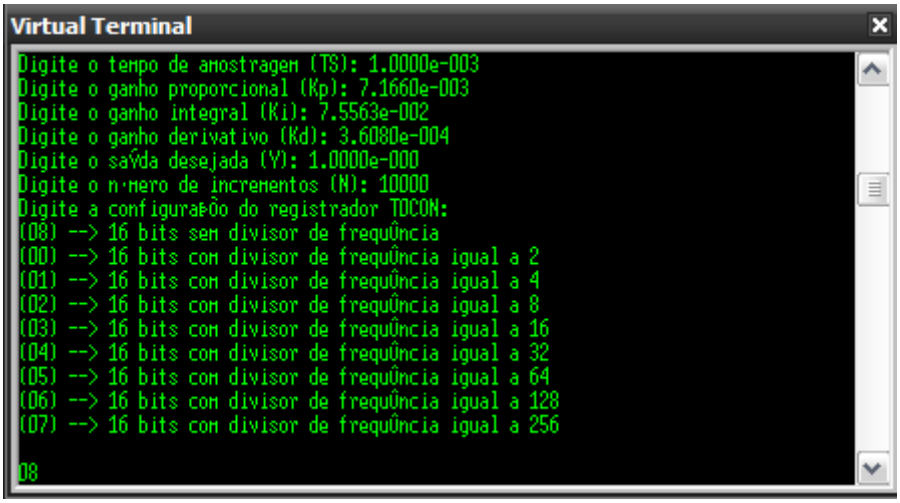

Figura 4.15 - Configuração do tempo de amostragem

Ao final da configuração, o sistema volta a operar no modo de operação normal, ou seja, como um controlador PID. Os dados de configuração que o usuário utiliza como entrada são gravados na memória EEPROM do microcontrolador possibilitando o uso posterior destes parâmetros mesmo com o desligamento do sistema.

4.2.2.3 TESTES COM O SISTEMA EM MALHA FECHADA UTILIZANDO O MICROCONTROLADOR PIC18F4520

Foram realizados quatro testes para comprovar o bom funcionamento do

sistema quanto à mudança nos parâmetros da planta. Os parâmetros utilizados foram obtidos para quatro pacientes reais (LAW; SHIELDS, 2006). O controlador opera com freqüência de 40 MHz e tempo de amostragem igual a 1 ms. O tempo de assentamento desejado escolhido é de 4 segundos. Mostram-se na Tabela 4.1 os parâmetros para cada paciente analisado. Na Tabela 4.2 apresentam-se os parâmetros de configuração do controlador PID para cada paciente e nas figuras 4.16 a 4.19 as respostas do sistema em malha fechada, conforme simulação no ambiente Proteus.

Tabela 4.1 - Parâmetros utilizados para teste conforme proposto no artigo de Law e Shields (2006).

Parâmetro	Paciente 1	Paciente 2	Paciente 3	Paciente 4
β [N.s]	30	32,8	27	32,1
ξ	0,68	0,8	0,63	0,65
ω_n [rad/s]	14,7	12,7	15,8	14,5

Tabela 4.2 - Parâmetros de configuração do controlador PID calculados para cada um dos pacientes do artigo de Law e Shields (2006).

Parâmetro	Paciente 1	Paciente 2	Paciente 3	Paciente 4
K_P	3,5043E-003	4,3772E-003	3,3503E-003	3,1727E-003
K_I	3,8291E-002	3,5021E-002	4,2545E-002	3,5786E-002
K_D	1,7544E-004	2,1494E-004	1,6874E-004	1,6861E-004

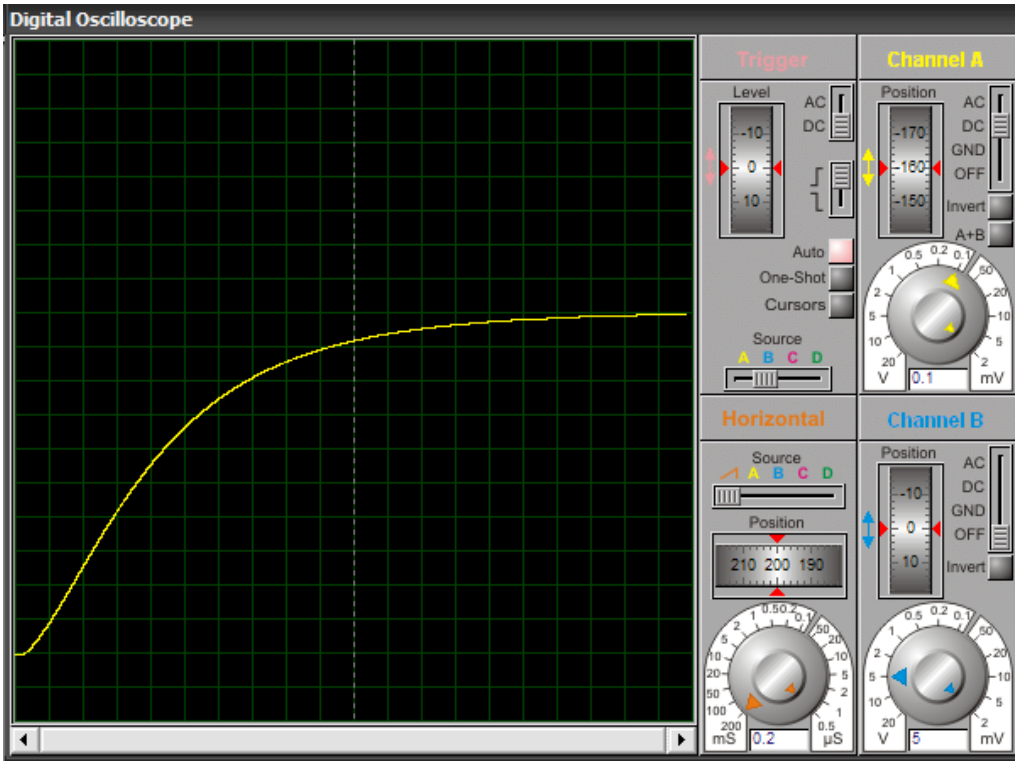

Figura 4.16 - Resposta do sistema em malha fechada para o paciente 1.

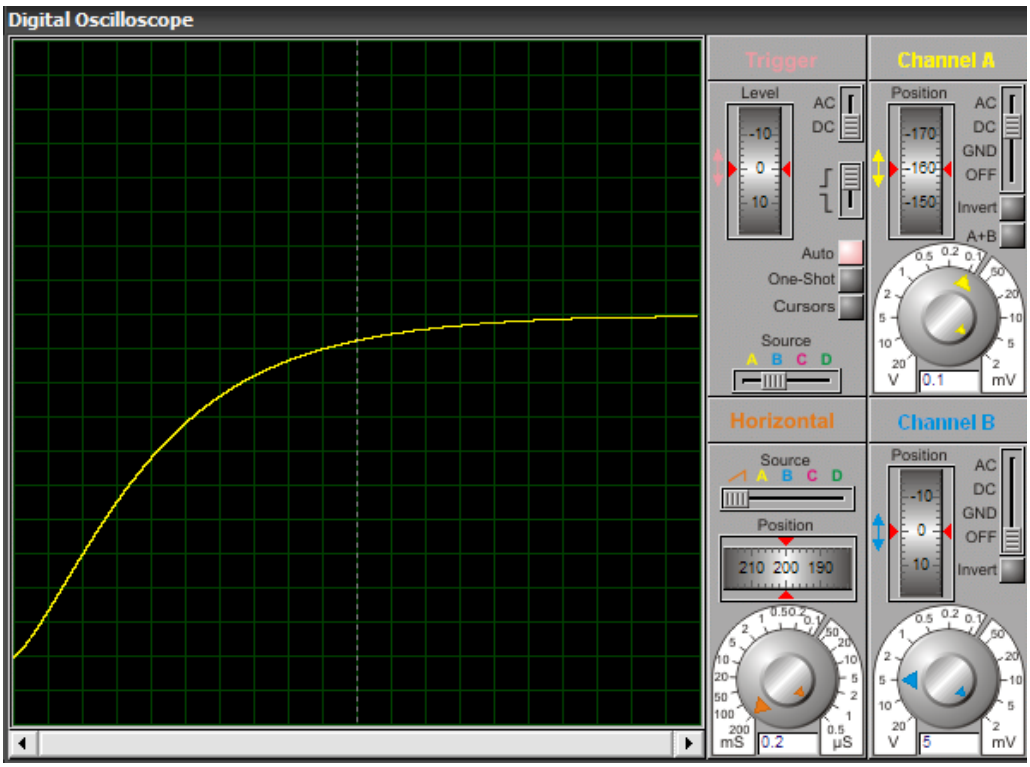

Figura 4.17 - Resposta do sistema em malha fechada para o paciente 2.

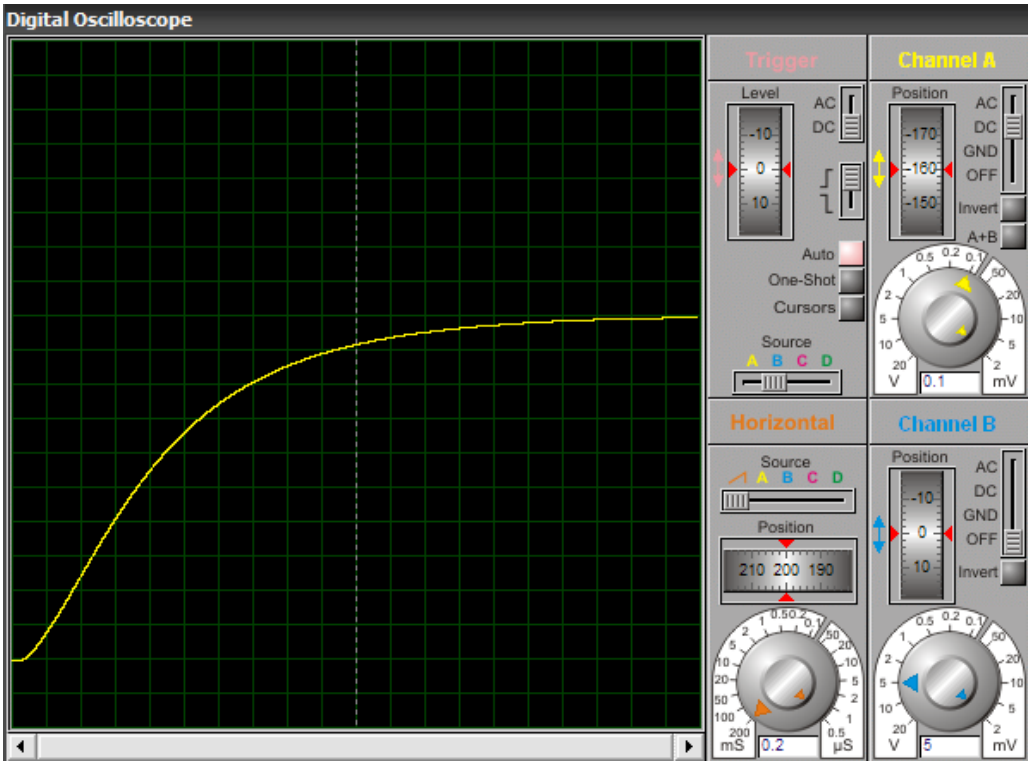

Figura 4.18 - Resposta do sistema em malha fechada para o paciente 3.

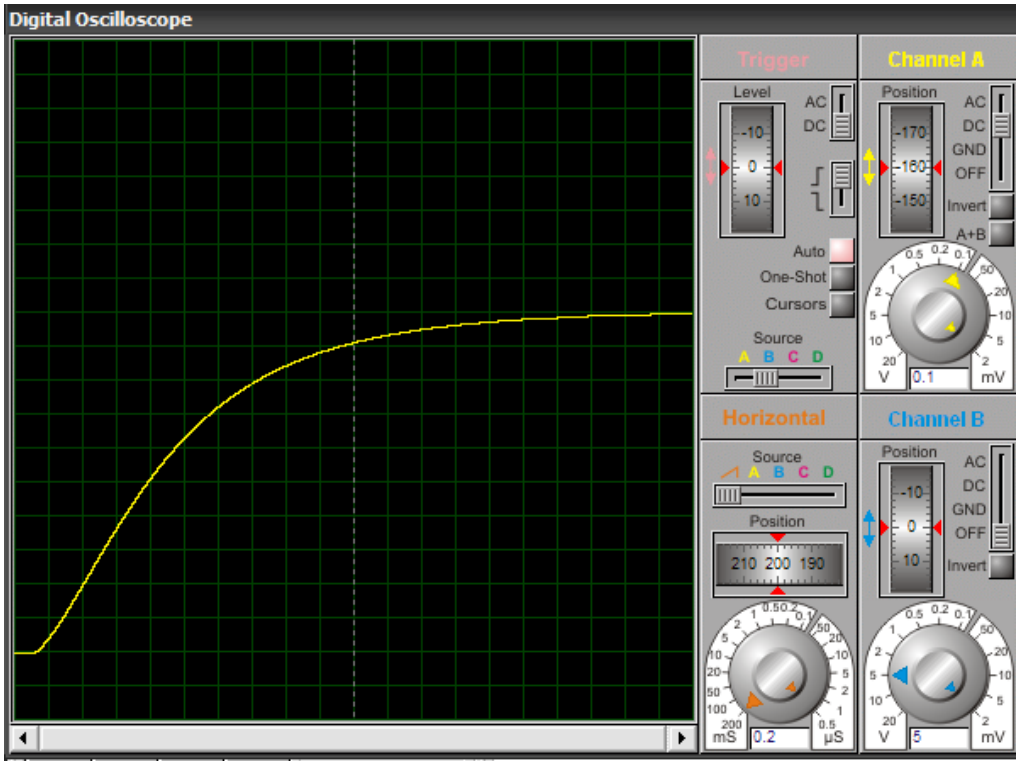

Figura 4.19 - Resposta do sistema em malha fechada para o paciente 4.

Os testes foram realizados para comprovar o funcionamento da ferramenta de projeto do controlador PID para diferentes pacientes. Assim, considerando que a base de

tempo do osciloscópio é de 0,2 segundos/divisão, constata-se que o sistema em malha fechada realmente possui tempo de assentamento praticamente igual a 4 segundos para todos os testes praticados.

5. CONCLUSÕES E TRABALHOS FUTUROS

Neste trabalho, projetou-se um controlador PID embarcado para controlar a força muscular através de estimulação elétrica neuromuscular. Para o projeto deste controlador, estudou-se um modelo matemático que prevê o comportamento da força muscular em função do estímulo elétrico aplicado. Os resultados apresentados na simulação no ambiente Matlab e na implementação no ambiente Proteus deste controlador se mostraram satisfatórios, já que o desempenho desejado foi alcançado.

É relevante salientar que, neste trabalho, foi chamado de implementação a simulação do sistema em malha fechada no ambiente Proteus. Apesar do ambiente Proteus ser um ambiente de simulação, a forma de se trabalhar com os componentes do sistema fornece uma visão prática da implementação do sistema físico. Além disso, os resultados fornecidos pelo ambiente Proteus são mais próximos da implementação real pelo fato de se utilizar um modelo específico de microcontrolador. A simulação no ambiente Matlab não prevê as características do microcontrolador, como o tempo gasto para a execução das instruções, para a conversão analógica-digital, entre outros. Assim, é interessante e proveitoso observar as diferenças entre os resultados da simulação no Matlab e os resultados obtidos a partir do ambiente Proteus, para determinação de melhorias a serem realizadas no sistema.

Além da implementação do sistema de controle em malha fechada para a utilização em EENM, é necessário fornecer uma ferramenta prática e eficiente ao profissional em engenharia de reabilitação que permita a ele manipular o sistema de acordo com suas necessidades. Nesse sentido, desenvolveram-se rotinas de comunicação serial no microcontrolador que possibilitam a configuração do controlador PID através de uma ferramenta computacional. Ou seja, através de um computador o usuário pode configurar o controlador PID como quiser, dentro dos limites estabelecidos pelo sistema. Além disso, implementou-se uma interface gráfica, com a linguagem Python, que permite ao usuário realizar tais configurações com facilidade.

Um grande desafio para os profissionais da área de reabilitação é ter que lidar com teoria de controle. Para resolver este problema, desenvolveram-se algoritmos no

ambiente Matlab e em linguagem Python que calculam os parâmetros do controlador PID de acordo com as especificações fornecidas pelo usuário como tempo de assentamento, por exemplo.

Por fim, neste trabalho desenvolveu-se uma ferramenta para a automatização do projeto de controladores PID aplicados em EENM e possibilitou o estudo da influência de controladores digitais microcontrolados aplicados em EENM (Prado, et al. 2008), (Gaino, et al. 2008).

Como trabalhos futuros, sugerem-se os itens alistados a seguir:

- Considerar os efeitos reais do sistema a ser controlado, o músculo. Isto envolve utilizar na prática um microcontrolador e conversores A/D e D/A. Para isso, é necessário envolver o paciente no sistema embarcado de estimulação elétrica neuromuscular. No entanto, antes dessa etapa é preciso aperfeiçoar o sistema. Assim, podem-se utilizar recursos da eletrônica analógica para simular o comportamento da força muscular em função de estímulos elétricos. O uso de amplificadores operacionais pode realizar esta simulação utilizando o modelo linear apresentado neste trabalho. A implementação real deste circuito eletrônico pode ser realizada com componentes discretos, ou ainda utilizando a tecnologia FPAA (*Field Programmable Analog Array*), que possibilita implementar funções analógicas como adição, subtração, multiplicação, integração, derivação, entre outras.
- Explorar os limites da técnica de projeto de controladores PID utilizada nesse trabalho. Ou seja, quais são os limites das especificações que o usuário pode utilizar (tempo de subida e máximo de sobre-sinal).
- Adaptar a saída do controlador PID para aplicar um sinal na forma de pulsos, através dos recursos de PWM do microcontrolador. Além disso, possibilitar o controle através da modulação por largura de pulsos e do intervalo entre pulsos.

REFERÊNCIAS

ASTRÖM, K. J., HÄGGLUND, T. **Automatic Tuning of PID Controllers.** Estados Unidos da América: [s.n.]. 1988. 141 p.

ASTRÖM, K. J., WITTENMARK, B. **Computer-controlled systems:** theory and design. 3.ed. New Jersey: Prentice Hall, 1997. 555 p.

BERNOTAS, L. A.; CRAGO, P. E.; CHIZECK, H. J. Adaptive control of electrically stimulated muscle. **IEEE transactions on biomedical engineering**, [S.1.], v. bme-34, n. 2, p. 140-147, Fev 1987.

BOLTON, W.; **Engenharia de controle**; tradução Valcere Vieira Rocha e Silva; revisão técnica Antonio Pertence Júnior; - São Paulo: Editora Makron Books, 1995. 497 p.

CACHO, E. W. A. **O efeito do treino de marcha com estimulação elétrica neuromuscular na atividade eletromiográfica de pacientes paraplégicos**. 2004. 92 f. Dissertação (Mestrado). Faculdade de Ciências Médicas, Universidade Estadual de Campinas, Campinas, 2005.

CHANG, G. et al. A neuro-control system for the knee joint position control with quadriceps stimulation. **IEEE transactions on rehabilitation engineering**, [S.1.], v. 5, n. 1, p. 2-11, Mar 1997.

CHIZECK, H. J.; KOFMAN, L.; CRAGO, P. E. et al. Pulse-train controllers for functional neuromuscular estimulation. **Decision and Control**, [S.1.], v. 22, n. 1, p. 1501-1503, Dez 1983.

CRAGO, P. E.; MORTIMER, J. T.; PECKHAM, P. H. Closed-Loop Control of Force During Electrical Stimulation of Muscle**. IEEE transactions on biomedical engineering**, [S.1.], v. bme-27, n. 6, p. 306-312, Jun 1980.

FARIA, U. C.; **Implementação de um sistema de geração de marcha para pacientes com lesões medulares**. 2006. 199 f. Tese (Doutorado) – Faculdade de Engenharia, Universidade Estadual Paulista, Ilha Solteira, 2006.

FERRARIN, M. et al. An experimental PID controller for knee movement restoration with closed loop FES system. In: ANNUAL INTERNATIONAL CONFERENCE OF THE IEEE ENGINEERING IN MEDICINE AND BIOLOGY SOCIETY, 18, 1996, Amsterdam. **FES III: Locomotion**. Amsterdam: [s.n.], 1996. p. 453-454.

FERRARIN, M. et al. Model-based control of FES-induced single joint movements. **IEEE Transactions on Neural Systems and Rehabilitation Engineering**, [S.1.], v. 9, n. 3, p. 245-247, Set 2001.

GAINO, R. et al. Integração de controle e instrumentação com a utilização do *software* Proteus para controle de movimentos na reabilitação de paraplégicos. In: CONGRESO TECNOLOGÍAS DE APOYO A LA DISCAPACIDAD, 5, 2008, Cartagena de Indias. **Comunicación Aumentativa y Alternativa**. Colombia: [s.n.], 2008. p. 134-137.

JEZERNIK, S.; WASSINK, R. G. V.; KELLER, T. Sliding mode closed-loop control of FES: controlling the shank movement. **IEEE transactions on biomedical engineering**, [S.1.], v. 51, n. 2, p. 263-272, Fev 2004.

LAN, N.; CRAGO, P. E.; CHIZECK, H. J. Closed-loop recruitment and frequency modulation of antagonistic muscles during co-activation in FNS. In: ANNUAL INTERNATIONAL CONFERENCE, 10., 1988, [s.n.]. **IEEE Engineering in medicine and biology**. [S.l.: s.n.], 1988. p. 1521-1522.

LAN, N.; CRAGO, P. E.; CHIZECK, H. J. Feedback control methods for task regulation by electrical stimulation of muscles. **IEEE transactions on biomedical engineering**, [S.l.], v.38, n. 12, p. 1213-1223, Dez 1991.

LAW, L. A. F.; SHIELDS, R. K. Predicting human chronically paralyzed muscle force: a comparison of three mathematical models. **Journal of Applied Physiology, Bethesda**, [S.l.], v. 100, n. 3, p. 1027-1036, 23 nov 2006.

MIKROELEKTRONIKA. **MikroC pro for PIC:** user manual. [S.l.: s.n.], 2009. Disponível em: <http://www.mikroe.com>. Acesso em: 28 maio 2009.

VASCONCELOS NETO, R. **Marcha com estimulação elétrica neuromuscular em paraplégicos: ângulos, pico de momento e teste de caminhada de seis minutos**. 2007. 133 f. Dissertação (Mestrado) Faculdade de Ciências Médicas, Universidade Estadual de Campinas, Campinas, 2007.

OGATA, K.; **Engenharia de controle moderno.** 4.ed. São Paulo: Editora Prentice Hall (Pearson), 2003. 800p.

PEIXOTO, B. O. **Redução da fadiga muscular sob estimulação elétrica neuromuscular**. 1995. 190 f. Dissertação (Mestrado). Curso de Pós-Graduação em Engenharia Elétrica, Universidade Estadual de Campinas, Campinas, 1996.

PRADO, T. A. et al. Desenvolvimento de uma ferramenta para o projeto de controladores PID aplicados em estimulação neuromuscular funcional. In: CONGRESSO BRASILEIRO DE ENGENHARIA BIOMÉDICA, 21., 2008, Bahia. **Engenharia biomédica:** a engenharia da saúde. Bahia: [s.n.], 2008. p. 891-894.

SILVA, T. I. **Implementação de um sistema para geração e avaliação de movimentos em pacientes hemiplégicos**. 2007. 171 f. Tese (Doutorado) Faculdade de Engenharia, Universidade Estadual Paulista, Ilha Solteira, 2007.

TESTA, F. J. **IEEE 754 compliant floating point routines**. [S.l.: s.n.], 1997. Disponível em: <http://www.microchip.com>. Acesso em: 28 maio 2009.

WHILHERE, G. F.; CRAGO, P. E.; CHIZECK, H. J. Design and evaluation of a digital closed-loop controller for the regulation of muscle force by recruitment modulation. **IEEE transactions on biomedical engineering**, [S.l.], v. bme-32, n. 9, p. 668-676, Set 1985.

APÊNDICE

APÊNDICE A - *EENM-PID.m*

```
clear all
format short e
% Discretizaçao da planta, considerando
% a açao do grampeador de ordem zero e
% tempo de amostragem Ts
Ts=input('Digite o valor desejado para o Tempo de Amostragem (Ts): ');
parametros
disp('Funcao de transferencia da planta discretizada')
Gpz=c2d(G,Ts,'zoh')
[num_Gpz,den_Gpz]=tfdata(Gpz);
num_Gpz=num_Gpz{1,1};
den_Gpz=den_Gpz{1,1};
% Zeros do controlador Gcz igual aos polos
% da planta Gpz
polos_planta=roots(den_Gpz);
z1=polos_planta(1);
z2=polos_planta(2);
% Parametros desejados
te=input('Digite o tempo de assentamento desejado: ');
psi=1
wn=4.6/(te*psi)
r=exp(-wn*psi*Ts);
% Calculo de K'
num_Gcz=conv([1 -z1],[1 -z2]);
den_Gcz=conv([1 -1],[1 0]);
Gcz=tf(num_Gcz,den_Gcz,Ts);
[K,polos]=rlocfind(series(Gpz,Gcz),[r]);
% Controlador discreto Gc(z)
num_Gcz=K*conv([1 -z1],[1 -z2]);
den_Gcz=conv([1 -1],[1 0]);
disp('Funçao de transferencia do controlador discreto PID')
Gcz=tf(num_Gcz,den_Gcz,Ts)
% Sistema de malha fechada
FTMFz=feedback(series(Gpz,Gcz),tf(1));
[nz,dz]=tfdata(FTMFz);
nz=nz{1,1};
dz=dz{1,1};
% Parametros do controlador PID
disp('Parametros do controlador PID')
Kd=num_Gcz(3)*Ts;
Kp=-(num_Gcz(2)+2*Kd/Ts)
```

```
Ki=(num_Gcz(1)-Kp-Kd/Ts)/Ts
Kd
Ti=Kp/Ki
Td=Kd/Kp
%Simulaçao do sistema em malha fechada
step(FTMFz)
```

APÊNDICE B - *parametros.m*

```
% Parametros para o modelo linear da resposta
% da força produzida por um musculo devido a
% estimulos eletricos na forma de trem de pulsos
% baseado no artigo da Laura para a media entre
% os pacientes envolvidos

beta=30.5;      % Ganho [N.s]
psi=0.69;       % Coeficiente de amortecimento
wn=14.4;        % frequencia natural de oscilaçao [rad/s]

% Funçao de transferencia para o modelo linear
% conforme proposto no artigo da Laura, sendo
%              beta*wn^2
% H(s)= -------------------------
%       s^2 + (2*wn*psi)*s + wn^2

num_G=[beta*wn^2];
den_G=[1 2*wn*psi wn^2];
disp('Funçao de transferencia da planta')
G=tf(num_G,den_G)
GanhoDC_G=dcgain(G);
```

APÊNDICE C - *configuracao_conversores.m*

```
x=input('Voce deseja adicionar os efeitos de quantizaçao dos conversores? (1)sim ou (2)nao ');

if x==1
        limite_inf_AD=input('Qual o limite inferior da variaçao da tensao analogica no conversor A/D? ');
        disp(' ')
        limite_sup_AD=input('Qual o limite superior da variaçao da tensao analogica no conversor A/D? ');
        if limite_sup_AD > limite_inf_AD
      faixa_AD=limite_sup_AD-limite_inf_AD;
        else
      faixa_AD=-limite_sup_AD+limite_inf_AD;
        end
        disp(' ')
        limite_inf_DA=input('Qual o limite inferior da variaçao da tensao analogica no conversor D/A? ');
        disp(' ')
        limite_sup_DA=input('Qual o limite superior da variaçao da tensao analogica no conversor D/A? ');
        if limite_sup_DA > limite_inf_DA
      faixa_DA=limite_sup_DA-limite_inf_DA;
        else
      faixa_DA=-limite_sup_DA+limite_inf_DA;
        end
        disp(' ')
   bits_AD=input('Qual a resoluçao do conversor A/D em bits? ');
   disp(' ')
   bits_DA=input('Qual a resoluçao do conversor D/A em bits? ');
   disp(' ')
   quantizacao_AD=(limite_sup_AD-limite_inf_AD)/(2^bits_AD);
   quantizacao_DA=(limite_sup_DA-limite_inf_DA)/(2^bits_DA);
else if x==2
      limite_inf_AD=-1e-6; limite_sup_AD=1e6;
      limite_inf_DA=-1e-6; limite_sup_DA=1e6;
      quantizacao_AD=1e-38; quantizacao_DA=1e-38;
   else
      disp('Voce nao digitou uma opçao valida! TENTE DE NOVO! ')
   end
end
```

APÊNDICE E - *configuracao_temporizador.m*

```
frequencia=input('Qual a frequencia de clock utilizada em MHz? ');
tempo_amostragem=input('Qual o tempo de amostragem desejado em ms ? ');
periodo_incremento=4/frequencia;
incrementos=tempo_amostragem*1000/periodo_incremento;
if incrementos < 256
   sprintf('\nPode-se utilizar o timer no modo 8 bits sem divisor de frequencia\n')
   modo_timer=8;
   taxa_divisao=1;
else if incrementos < 65536
      sprintf('\nPode-se utilizar o timer no modo 16 bits sem divisor de frequencia\n')
      modo_timer=16;
      taxa_divisao=1;
   else
      taxa_divisao=incrementos/65536;
      if taxa_divisao>256
         sprintf('\nO tempo de amostragem escolhido e muito grande\n')
         modo_timer=0;
      else
         modo_timer=16;
         taxa_divisao=2^(ceil(log2(taxa_divisao)));
          sprintf('\nDeve-se utilizar o timer no modo 16 bits com divisor de frequencia igual a
%i\n',taxa_divisao)
      end
   end
end
incrementos=floor(incrementos/taxa_divisao);
sprintf('O timer deve invocar a rotina de interrupçao a cada %i incrementos.\n',incrementos)
```

APÊNDICE F - *Controlador_PID_PIC18F4520.c*

```
/*
-----Projeto: Controlador PID versão 2.5
-----Autor:   THIAGO ALEXANDRE PRADO
-----Data: 30/dez/2008
-----Clock = 40 MHz - Tempo de amostragem de 1 ms
-----PIC18F4520 (32 Kb ROM)
-----Funcionando com passagem de parâmetros (TS,KP,KI,KD,Y,incrementos,T0CON)no
-----início do programa através do terminal virtual. Chave no pino RE0 deve
-----estar em "0".
-----Incluindo o recurso de valores padrões caso o usuário não configure o
-----controlador e não haja nenhuma configuração prévia na EEPROM.
*/

#define ts_constante          "1.0000e-003"
#define kp_constante          "6.3668e-003"
#define ki_constante          "7.4033e-002"
#define kd_constante          "3.2329e-004"
#define y_constante           "1.0000e-000"
#define incrementos_constante "10000"
#define T0CON_constante       "08"

signed float sensor[2], saida=0, erro[2];
signed float termo_derivativo=0, termo_integral[2], termo_proporcional=0;
signed int temp1 = 0;
unsigned int temp2 = 0;
long int incrementos;
signed float TS,Kp,Kd,Ki,saida_desejada;
char ts_string[12],kp_string[12],ki_string[12],kd_string[12],y_string[12];
char incrementos_string[6], T0CON_string[3];

void configuracao_inicial()
{
////////////////// Configuração do módulo serial ////////////////////////////
  Usart_Init(9600);             // Configura módulo serial para operar em 9600 bps
////////////////// Configuração do conversor A/D ////////////////////////////
  ADCON0 = 0x01;                // Habilita módulo A/D no pino AN0
  ADCON1 = 0x3E;                // Configura módulo para tensões de referência
                                // externa e apenas AN0 como entrada analógica
  ADCON2 = 0xFE;                // Configura TAD = 64*TOSC e TACQ = 20*TAD
////////////////// Configuração das portas de E/S ///////////////////////////
  TRISB = 0xFC;                 // Pino RB0 como saída digital: indica "1" quando
                                // estiver dentro da rotina do controlador PID
  TRISD = 0x00;                 // Porto C como saída digital = bits [7..0] do
                                // conversor D/A
  TRISC = 0xFC;                 // Pinos RC7, RC6 como saídas digitais = bits
```

```
                                // [9..8] do conversor D/A
  TRISE.F0 = 1;                 // Pino RE0 como entrada digital
////////////////// Inicialização de variáveis ////////////////////////////////
  PORTC=0x01;PORTD=0xFF;      // Inicializa conversor D/A com tensão de 0 Volts
  erro[0]=erro[1]=0;
  sensor[0]=sensor[1]=0;
  termo_integral[0]=termo_integral[1]=0;
}
void configuracao_temporizador()
{
////////////////// Configuração do Timer 0 ///////////////////////////////////
  int divisor;
  incrementos = atol(incrementos_string);
  divisor = atoi(T0CON_string);

  T0CON = 0x80 + divisor;                  // Timer 0 configurado para 16 bits

  TMR0H = (65536-incrementos)>>8;   // Ajuste para ocorrer a quantidade de
  TMR0L = (65536-incrementos);          // incrementos determinada pelo usuário
  INTCON = 0xA0;                            // Habilita interrupção do TIMER 0
}
void controlador_PID()
{
    PORTB.F0=1;                              // Sinalização de começo do processo
                                             // de leitura do A/D e do controle PID

    sensor[0]=sensor[1];                    // Gravação da última leitura do A/D

    temp2 = ADC_Read(0);                  // Leitura da saída da função de
    temp1 = (signed int) temp2;          // transferência e conversão para
    temp1 = temp1 - 0x55;                 // float
    sensor[1] = (signed float) ((temp1)*(1.2/1024.0));

//// Este espaço é reservado para a lei de controle ////////////////////////////
    erro[0]=erro[1];                    // Gravação do último sinal de erro
    erro[1]=saida_desejada-sensor[1];     // Cálculo do sinal de erro
//// Cálculo da parte derivativa ////////////////////////////////////////////////
    termo_derivativo = (Kd/TS)*(sensor[0]-sensor[1]);
//// Cálculo da parte integral //////////////////////////////////////////////////
    termo_integral[0] = termo_integral[1];
    termo_integral[1] = termo_integral[0] + Ki*erro[1]*TS;
//// Cálculo da parte proporcional //////////////////////////////////////////////
    termo_proporcional = Kp * erro[1];
//// Cálculo da saída ///////////////////////////////////////////////////////////
    saida = termo_proporcional + termo_integral[1] + termo_derivativo;

////////////////////////////////////////////////////////////////////////////////
    saida = (saida*1024.0/0.2);          // Conversão da saída de float para
```

```
    temp1 = (signed int) saida;        // um número inteiro e escalonado para
    temp1 = (temp1+0x1FF);             // enviar ao conversor D/A
    temp2 = (unsigned int) temp1;
    PORTC = temp2 >> 8;
    PORTD = temp2;
    PORTB.F0=0;                        // Sinalização de final do processo de
                                       // leitura do A/D e do controle PID
}
void interrupt()
{
  INTCON.F2 = 0;             // Zera o flag do TMR0
  TMR0H = (65536-incrementos)>>8;   // Ajuste para ocorrer 10000 incrementos
  TMR0L = (65536-incrementos);          //
  controlador_PID();                    // Executa a rotina de controle
}

void escreve_serial(char a[100], int tamanho)
{
  int i;
  tamanho=strlen(a);
  for(i=0;i<tamanho;i++)
  Usart_Write(a[i]);
}

void configuracao_PID()
{
  char a[100], temp;
  unsigned char end_eeprom=0;
  int tamanho,i;
////////////////// Recepção de dados pelo protocolo UART (RS-232) ////////////
  escreve_serial(a,sprinti(a,"Configuração do controlador PID.\r"));
    escreve_serial(a,sprinti(a,"\r\rDigite 'C' caso realmente queira realizar a configuração do
Controlador. "));
  while(!Usart_Data_Ready()) {}
  temp=Usart_Read();
  if (temp != 'C')
  {
    escreve_serial(a,sprinti(a,"\r\rConfiguração cancelada.\r\r"));
          escreve_serial(a,sprinti(a,"Tentando obter a configuração a partir dos dados da
EEPROM."));
    return;
  }
  else
  {
    escreve_serial(a,sprinti(a,"\rIniciando configuração do Controlador PID.\r"));
  }
  Eeprom_Write(end_eeprom,'T');
  end_eeprom++;
```

```
delay_ms(20);
escreve_serial(a,sprinti(a,"\rDigite o tempo de amostragem (TS): "));
for(i=0;i<11;i++)
{
 while(!Usart_Data_Ready()) {}
 ts_string[i]=Usart_Read();
 Eeprom_Write(end_eeprom,ts_string[i]);
 delay_ms(20);
 Usart_Write(Eeprom_Read(end_eeprom));
 delay_ms(20);
 end_eeprom++;
}
ts_string[11]=0;
escreve_serial(a,sprinti(a,"\rDigite o ganho proporcional (Kp): "));
Eeprom_Write(end_eeprom,'P');
end_eeprom++;
delay_ms(20);
for(i=0;i<11;i++)
{
 while(!Usart_Data_Ready()) {}
 kp_string[i]=Usart_Read();
 Eeprom_Write(end_eeprom,kp_string[i]);
 delay_ms(20);
 Usart_Write(Eeprom_Read(end_eeprom));
 delay_ms(20);
 end_eeprom++;
}
kp_string[11]=0;
escreve_serial(a,sprinti(a,"\rDigite o ganho integral (Ki): "));
Eeprom_Write(end_eeprom,'I');
end_eeprom++;
delay_ms(20);
for(i=0;i<11;i++)
{
 while(!Usart_Data_Ready()) {}
 ki_string[i]=Usart_Read();
 Eeprom_Write(end_eeprom,ki_string[i]);
 delay_ms(20);
 Usart_Write(Eeprom_Read(end_eeprom));
 delay_ms(20);
 end_eeprom++;
}
ki_string[11]=0;
escreve_serial(a,sprinti(a,"\rDigite o ganho derivativo (Kd): "));
Eeprom_Write(end_eeprom,'D');
end_eeprom++;
delay_ms(20);
for(i=0;i<11;i++)
```

```
{
 while(!Usart_Data_Ready()) {}
 kd_string[i]=Usart_Read();
 Eeprom_Write(end_eeprom,kd_string[i]);
 delay_ms(20);
 Usart_Write(Eeprom_Read(end_eeprom));
 delay_ms(20);
 end_eeprom++;
}
kd_string[11]=0;
Eeprom_Write(end_eeprom,'Y');
end_eeprom++;
delay_ms(20);
escreve_serial(a,sprinti(a,"\rDigite o saída desejada (Y): "));
for(i=0;i<11;i++)
{
 while(!Usart_Data_Ready()) {}
 y_string[i]=Usart_Read();
 Eeprom_Write(end_eeprom,y_string[i]);
 delay_ms(20);
 Usart_Write(Eeprom_Read(end_eeprom));
 delay_ms(20);
 end_eeprom++;
}
y_string[11]=0;
Eeprom_Write(end_eeprom,'N');
end_eeprom++;
delay_ms(20);
escreve_serial(a,sprinti(a,"\rDigite o número de incrementos (N): "));
for(i=0;i<5;i++)
{
 while(!Usart_Data_Ready()) {}
 incrementos_string[i]=Usart_Read();
 Eeprom_Write(end_eeprom,incrementos_string[i]);
 delay_ms(20);
 Usart_Write(Eeprom_Read(end_eeprom));
 delay_ms(20);
 end_eeprom++;
}
incrementos_string[5]=0;
Eeprom_Write(end_eeprom,'0');
end_eeprom++;
delay_ms(20);
escreve_serial(a,sprinti(a,"\rDigite a configuração do registrador T0CON:"));
escreve_serial(a,sprinti(a,"\r(08) --> 16 bits sem divisor de frequência"));
escreve_serial(a,sprinti(a,"\r(00) --> 16 bits com divisor de frequência igual a 2"));
escreve_serial(a,sprinti(a,"\r(01) --> 16 bits com divisor de frequência igual a 4"));
escreve_serial(a,sprinti(a,"\r(02) --> 16 bits com divisor de frequência igual a 8"));
```

```
 escreve_serial(a,sprinti(a,"\r(03) --> 16 bits com divisor de frequência igual a 16"));
 escreve_serial(a,sprinti(a,"\r(04) --> 16 bits com divisor de frequência igual a 32"));
 escreve_serial(a,sprinti(a,"\r(05) --> 16 bits com divisor de frequência igual a 64"));
 escreve_serial(a,sprinti(a,"\r(06) --> 16 bits com divisor de frequência igual a 128"));
 escreve_serial(a,sprinti(a,"\r(07) --> 16 bits com divisor de frequência igual a 256\r\r"));
 for(i=0;i<2;i++)
 {
  while(!Usart_Data_Ready()) {}
  T0CON_string[i]=Usart_Read();
  Eeprom_Write(end_eeprom,T0CON_string[i]);
  delay_ms(20);
  Usart_Write(Eeprom_Read(end_eeprom));
  delay_ms(20);
  end_eeprom++;
 }
 T0CON_string[2]=0;
 Eeprom_Write(end_eeprom,0);
 delay_ms(20);
}

void configura_parametros()
{
 unsigned char end_eeprom=0;
 char temp, a[100];
 int i;
 temp = Eeprom_Read(end_eeprom);
 delay_ms(20);
 if(temp == 'T')
 {
  escreve_serial(a,sprinti(a,"\rIniciando leitura do parâmetro TS da EEPROM\r"));
  for(i=0;i<11;i++)
  {
   end_eeprom++;
   ts_string[i]=Eeprom_Read(end_eeprom);
   delay_ms(20);
  }
  ts_string[11]=0;
  escreve_serial(a,sprinti(a,"TS = "));
  escreve_serial(ts_string,12);
 }
 else
 {
       escreve_serial(a,sprinti(a,"\rConfiguração TS do controlador não está gravada na
EEPROM\r"));
   sprinti(ts_string,ts_constante);
 }
 end_eeprom++;
 temp = Eeprom_Read(end_eeprom);
```

```
 delay_ms(20);
 if(temp == 'P')
 {
  escreve_serial(a,sprinti(a,"\rIniciando leitura do parâmetro Kp da EEPROM\r"));
  for(i=0;i<11;i++)
  {
   end_eeprom++;
   kp_string[i]=Eeprom_Read(end_eeprom);
   delay_ms(20);
  }
  kp_string[11]=0;
  escreve_serial(a,sprinti(a,"Kp = "));
  escreve_serial(kp_string,12);
 }
 else
 {
      escreve_serial(a,sprinti(a,"\rConfiguração Kp do controlador não está gravada na
EEPROM\r"));
  sprinti(kp_string,kp_constante);
 }
 end_eeprom++;
 temp = Eeprom_Read(end_eeprom);
 delay_ms(20);
 if(temp == 'I')
 {
  escreve_serial(a,sprinti(a,"\rIniciando leitura do parâmetro Ki da EEPROM\r"));
  for(i=0;i<11;i++)
  {
   end_eeprom++;
   ki_string[i]=Eeprom_Read(end_eeprom);
   delay_ms(20);
  }
  ki_string[11]=0;
  escreve_serial(a,sprinti(a,"Ki = "));
  escreve_serial(ki_string,12);
 }
 else
 {
      escreve_serial(a,sprinti(a,"\rConfiguração Ki do controlador não está gravada na
EEPROM\r"));
  sprinti(ki_string,ki_constante);
 }
 end_eeprom++;
 temp = Eeprom_Read(end_eeprom);
 delay_ms(20);
 if(temp == 'D')
 {
  escreve_serial(a,sprinti(a,"\rIniciando leitura do parâmetro Kd da EEPROM\r"));
```

```
  for(i=0;i<11;i++)
  {
   end_eeprom++;
   kd_string[i]=Eeprom_Read(end_eeprom);
   delay_ms(20);
  }
  kd_string[11]=0;
  escreve_serial(a,sprinti(a,"Kd = "));
  escreve_serial(kd_string,12);
 }
 else
 {
      escreve_serial(a,sprinti(a,"\rConfiguração Kd do controlador não está gravada na
EEPROM\r"));
  sprinti(kd_string,kd_constante);
 }
 end_eeprom++;
 temp = Eeprom_Read(end_eeprom);
 delay_ms(20);
 if(temp == 'Y')
 {
  escreve_serial(a,sprinti(a,"\rIniciando leitura do parâmetro Y da EEPROM\r"));
  for(i=0;i<11;i++)
  {
   end_eeprom++;
   y_string[i]=Eeprom_Read(end_eeprom);
   delay_ms(20);
  }
  y_string[11]=0;
  escreve_serial(a,sprinti(a,"Y = "));
  escreve_serial(y_string,12);
 }
 else
 {
      escreve_serial(a,sprinti(a,"\rConfiguração Y do controlador não está gravada na
EEPROM\r"));
  sprinti(y_string,y_constante);
 }
 end_eeprom++;
 temp = Eeprom_Read(end_eeprom);
 delay_ms(20);
 if(temp == 'N')
 {
  escreve_serial(a,sprinti(a,"\rIniciando leitura do parâmetro N da EEPROM\r"));
  for(i=0;i<5;i++)
  {
   end_eeprom++;
   incrementos_string[i]=Eeprom_Read(end_eeprom);
```

```
    delay_ms(20);
   }
   incrementos_string[5]=0;
   escreve_serial(a,sprinti(a,"Y = "));
   escreve_serial(incrementos_string,12);
  }
  else
  {
        escreve_serial(a,sprinti(a,"\rConfiguração N do controlador não está gravada na
EEPROM\r"));
   sprinti(incrementos_string,incrementos_constante);
  }
  end_eeprom++;
  temp = Eeprom_Read(end_eeprom);
  delay_ms(20);
  if(temp == '0')
  {
   escreve_serial(a,sprinti(a,"\rIniciando leitura do parâmetro 0 da EEPROM\r"));
   for(i=0;i<2;i++)
   {
    end_eeprom++;
    T0CON_string[i]=Eeprom_Read(end_eeprom);
    delay_ms(20);
   }
   T0CON_string[2]=0;
   escreve_serial(a,sprinti(a,"T0CON = "));
   escreve_serial(T0CON_string,12);
  }
  else
  {
        escreve_serial(a,sprinti(a,"\rConfiguração 0 do controlador não está gravada na
EEPROM\r"));
   sprinti(T0CON_string,T0CON_constante);
  }

  escreve_serial(a,sprinti(a,"\rFinal da leitura dos parâmetros da EEPROM\r"));
}
void main() {
  PORTB.F1=0;
  configuracao_inicial();
  if (PORTE.F0 == 0) configuracao_PID();
  configura_parametros();
  TS=atof(ts_string);
  Kp=atof(kp_string);
  Ki=atof(ki_string);
  Kd=atof(kd_string);
  saida_desejada=atof(y_string);
  delay_ms(3000);
```

```
  configuracao_temporizador();
  PORTB.F1=1;
  while(1);
}
```

Livros Grátis

(http://www.livrosgratis.com.br)

Milhares de Livros para Download:

Baixar livros de Literatura
Baixar livros de Literatura de Cordel
Baixar livros de Literatura Infantil
Baixar livros de Matemática
Baixar livros de Medicina
Baixar livros de Medicina Veterinária
Baixar livros de Meio Ambiente
Baixar livros de Meteorologia
Baixar Monografias e TCC
Baixar livros Multidisciplinar
Baixar livros de Música
Baixar livros de Psicologia
Baixar livros de Química
Baixar livros de Saúde Coletiva
Baixar livros de Serviço Social
Baixar livros de Sociologia
Baixar livros de Teologia
Baixar livros de Trabalho
Baixar livros de Turismo