PROPOSTA DE EXPERIMENTO REMOTO DIDÁTICO APLICADO AO CONTROLE DE ROTAÇÃO DE MOTOR DE CORRENTE CONTÍNUA

Luiz Henrique Cassettari[1]
João Mota Neto[2]
Eron Da Silva[3]
Pedro Henrique Biava Spillere[4]

Resumo: Desenvolver um experimento remoto para o controle em malha fechada da rotação de um motor de corrente contínua. É apresentada a fundamentação teórica sobre experimento remoto, visando controlar o motor remotamente usando a internet comomeio de comunicação. Apresentam-se também os recursos tecnológicos para desenvolver uma ferramenta de auxílio ao estudo, que permitirá o controle da rotação do motor pela internet. Desenvolveu-se um aplicativo na linguagem deprogramação Java para estabelecer a comunicação entre o usuário e o experimento,permitindo a ação de controle e sua visualização por meio do gráfico. O motor será controlado pela plataforma Arduino, a placa Arduino Uno possui um microcontrolador programável responsável pelo algoritmo de controle, e quando unida ao *Ethernet Shield* permite comunicar-se com a rede internet. A partir desse, desenvolveu-se o projeto e a estratégia de controle, bem como, o algoritmo de controle. O modelo de um controlador PID digital foi implementado à plataforma Arduino, assim como a lógica de controle pela internet. Validou-se o algoritmo PID implementado no Arduino Uno por meio da ferramenta computacional MATLAB, comparando os gráficos gerados pelo aplicativo e simulados, referente ao experimento na sua totalidade após interações via internet.

Palavras-chave: Experimento remoto, Java, Arduino, Controlador PID.

1 INTRODUÇÂO

A internet proporciona uma ferramenta ampla de comunicação, podendo ser utilizada por diversas áreas e finalidades. No âmbito da educação aplica-se este recurso para desenvolvimento de aplicações que proporcionem maior interação interdisciplinar e na implementação de cursos à distância.

Contudo faz-se necessário o uso de dispositivos eletrônicos físicos com a capacidade de serem acessados e controlados pela internet. Sendo estes nomeados de experimento remoto, os alunos e professores podem utilizar esse recurso para

[1] Graduando em Engenharia Elétrica, UniSATC. E-mail: ricaun@gmail.com
[2] Professor do Centro Universitário UniSATC. E-mail: joao.neto@satc.edu.br
[3] Professor do Centro Universitário UniSATC. E-mail: eron.silva@satc.edu.br
[4] Graduando em Engenharia Mecatrônica, UniSATC. E-mail: pedrospillerecri@gmail.com

interagir e receber informações reais de experimentos, validando teorias discutidas em sala de aula.

A proposta de experimentos remotos foca-se no auxílio de atividades práticas a serem realizadas pelos estudantes na compreensão de conteúdos curriculares ministrados nas aulas. Podendo desfrutar desse recurso em qualquer horário, pois se encontra ligado a rede mundial de computadores disponível 24 horas por dia e 7 dias por semana. Devido à disponibilidade de acesso, flexibiliza o horário do usuário ao interagir com a atividade prática virtual.

Neste contexto, o presente trabalho propõe um experimento remoto didático prático. Esse proporcionará a interação do acadêmico ao controle da rotação de um motor de corrente contínua. Desse modo, os alunos e os professores poderão utilizar essa ferramenta, disponível na internet, para estudos relacionados a disciplinas de sistema de controle.

O controle empregado ao motor, especificamente, será um controlador PID. Nele o usuário terá acesso aos parâmetros desse controlador, a fim de modificá-los e poder avaliar o desempenho do mesmo, utilizando a interface específica do experimento que permite o controle do motor e visualização da rotaçãopor meio de gráficos.

2 FUNDAMENTAÇÃO TEÓRICA

2.1 ARDUINO

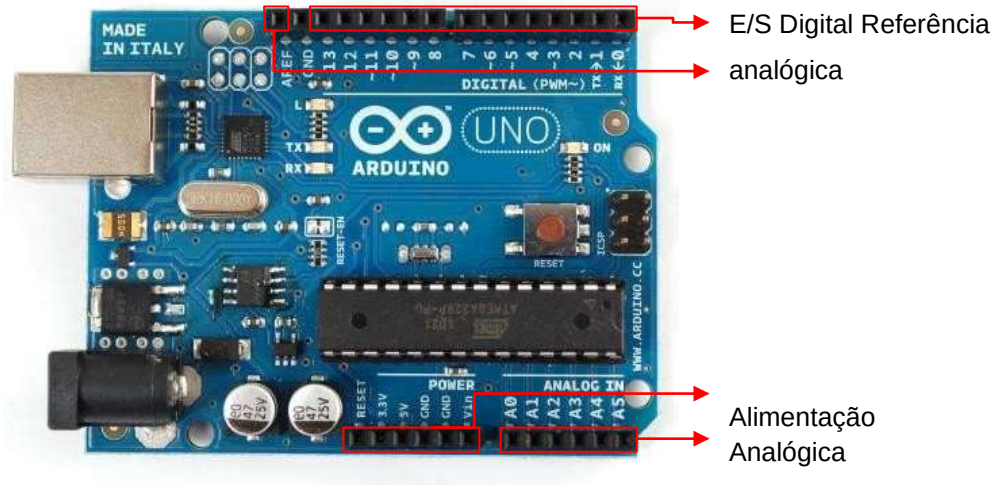
Este dispositivo integra o conceito de hardware e software livre e está aberto para uso e contribuição de toda sociedade. O conceito Arduino surgiu na Itália, em 2005, com o objetivo de criar um equipamento para desenvolver projetos/protótipos construídos de uma forma menos dispendiosa do que outros sistemas disponíveis no mercado.

2.1.2 Arduino Uno

Esta placa eletrônica é baseada no chip ATmega328 fabricado pela Atmel, possui entrada/saída (E/S) digital e analógica, além de interface serial atravésda porta USB, permite ao usuário programar e interagir em tempo real. Os conectores de entrada

e saída são expostos de maneira que possam ser interligados a outros módulos expansivos, conhecidos como *Shields*. A Figura 1 demonstra a placa Arduino Uno com destaque nos conectores de: E/S digital, referência analógica, alimentação e entrada analógica.

Figura 1 – Placa Arduino Uno.



Fonte: Arduino. Disponível em: <http://arduino.cc/en/Main/ArduinoBoardUno>. Acesso em: 30 ago. 2011.

O Arduino Uno que será utilizado no experimento é composto porquatorze pinos de E/S digital, seis entradas analógicas, oscilador de cristal 16 MHz, controlador USB, uma entrada de alimentação, conector ICSP e um botão de *reset*.

Para sua utilização, basta conectá-lo a um computador com um cabo USB ou ligá-lo com um adaptador AC para CC ou bateria.

O chip Atmega328 possui conversor analógico-digital de 10 bits,nomeados como entrada analógica no Arduino Uno. O conversor suporta mapear sinais de tensões entre 0 e 5 Volts em valores inteiros entre 0 e 1023, resulta na resolução nas leituras de: 5 Volts / 1024 unidades ou, 0,0049 Volts por unidade.

Outra característica importante do Arduino Uno é a facilidade de gravar novos programas no Atmega328, pois essa operação é executada na placa do Arduino Uno, sendo utilizado o conversor USB para serial e o *firmware* gravado no chip Atmega328. Desse modo o usuário não necessita retirá-lo do *socket* ou utilizaruma placa eletrônica específica para realizar a operação.

2.1.2 Ethernet Shield

Para realizar a conexão do experimento proposto neste trabalho utilizandoa

rede internet como comunicação entre o usuário e o experimento, faz-se necessário adicionar a placa do Arduino Uno responsável por controlar o motor elétrico e por meio da leitura do taco gerador à placa eletrônica *Ethernet Shield.*

O principal componente empregado na placa *Ethernet* é o chip ENC28J60da Microchip, que suporta diversos protocolos de comunicação como: TCP, UDP, entre outros. Esse chip foi projetado para permitir que qualquer dispositivo que possua comunicação *Serial Peripheral Interface* (SPI) possa estabelecer uma conexão *Ethernet*.

O Arduino Uno possui a comunicação SPI, que é localizada através das E/S digitais 10, 11, 12 e 13. Assim a placa *Ethernet Shield* possui o *layout* que permite o encaixe rápido, conectando todos os pinos do Arduino Uno aos respectivos pinos do *Shield*, desse modo não é necessário a utilização de cabos externos para interligar as placas.

Para comunicar-se com a internet é preciso que diversos pacotes sejam trocados de forma organizada para estabelecer a conexão.

Para que o *Ethernet Shield* possa trocar pacotes e se conectar com a rede é preciso incluir uma biblioteca específica da placa, que inclui funções e comandos de configurações que permitem transferência dos pacotes e conexão do *Ethernet*.

Os pacotes da rede recebidos pelo *Shield* são enviados ao Arduino Uno por SPI, esses pacotes são armazenados em um buffer interno do chip Atmega328, são analisados pela biblioteca do *Ethernet*. Esse buffer é criado ao incluir essa biblioteca, porém o tamanho do buffer é limitado pelo tamanho da SRAM sendo de 2000 Bytes.

2.1.3 Ambiente de Programação do Arduino

O monitor serial será utilizado para auxiliar o andamento do experimento, pois toda a programação e funcionalidades previstas no projeto serão desenvolvidasno ambiente de programação do Arduino, como o controlador PID e a lógica de controle.

2.1.4 Tipos de controladores

Nos processos industriais existem diversos tipos de controladores usados para atender uma infinidade de problemas industriais. Os controladores amplamente utilizados nas indústrias normalmente são sistemas de controle de malha fechada, pois

possuem a realimentação do sinal e permitem resolver diversos problemas encontrados na indústria.

Dentre os controladores de malha fechada podemos destacar o controlador PID. Conforme Campos e Teixeira (2006, p.7):

> O controlador do tipo Proporcional-Integral-Derivativo (PID) é, sem dúvida, o mais usado em sistemas de malha fechada na área da indústria. As vantagens deste controlador são:
>
> - Bom desempenho em muitos processos;
> - Estrutura versátil;
> - Poucos parâmetros a serem sintonizados ou ajustados;
> - Fácil associação entre parâmetros de sintonia e o desempenho.

O controlador PID utiliza o erro[1] de três métodos distintos para controlar o sinal de saída: o termo proporcional (P), o integral (I) e o derivativo (D).

As características e os efeitos dos parâmetros desses controladores são distintas e são descritas a seguir:

a) Proporcional: A saída do controlador é diretamente proporcional ao sinal do erro. O controlador proporcional pode ser comparado com um amplificador, com ganho Kp.

A ação de controle proporcional não possui a habilidade de eliminar o erro em regime permanente, porém o controlador proporcional pode ser usado com bons resultados em processos de baixa ordem, utilizando um valor de Kp adequado permite acelerar a reposta do sistema. No entanto, em sistemas de ordem superior autilização de ganhos elevados pode levar o sistema a se tornar instável.

b) Integral: A ação de controle integral faz com que a saída do controlador seja proporcional a integral do sinal do erro. Desse modo o controlador considera todos os erros anteriores dentro de um espaço de tempo definido no processo.

A inclusão da ação integral a um controle de malha fechada tem por objetivo eliminar o erro em regime permanente, porém a utilização de um ganho integral elevado pode criar instabilidade no processo. A ação integral não é utilizada separada da ação proporcional.

c) Derivativo: O controle derivativo resulta em uma saída proporcional a taxa de variação do erro. Como mostra a equação (3) a taxa de variação é multiplicada pelo Kd.

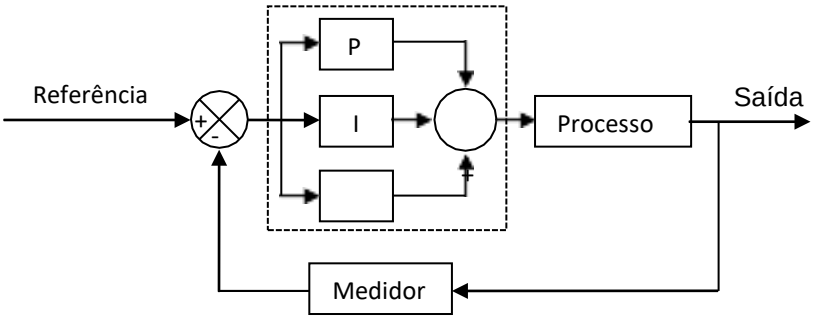
A utilização desse controlador permite corrigir a resposta transitória do

sistema. Quando usada em sistemas acionados por referências constantes, a ação derivativa não terá efeito sobre o regime permanente. Por outro lado, essa ação será predominante nos instantes em que a variação do erro é rápida. Isto acontece geralmente nas mudanças de referência ou nos momentos em que o processo é afetado por perturbações de carga. [15] A ação derivativa pode ser interpretada como uma maneira de gerar uma atuação que possa prever um determinado efeito na resposta do sistema e evitá-lo ou, ao menos, diminuí-lo.

Segundo Campos e Teixeira (2006) a união entre a ação proporcional, integral e derivativa resulta no controlador PID, que podem ser encontrados em diversos formatos.

Dentre os formatos existentes o escolhido para ser implementado ao experimento foi o PID em paralelo, por ser o amplamente encontrado dentre os controladores industriais. A Figura 2 demonstra o diagrama de blocos de um sistema de controle de malha fechada utilizando um controlador PID.

Figura 2 – Diagrama de blocos de um controlador PID.



Fonte: Do autor, 2011.

A figura anterior mostra o digrama de blocos do controlador PID, esse possui um somador que integra as ações dos controladores: proporcional, integral e derivativo. Resultando no controlador PID em paralelo, sendo a equação desse controlador é demonstrada logo abaixo pela equação 1.

$$PID = K_p\ e(t) + K_i \int_0^t e(t)\ dt + K_d \frac{d\ e(t)}{dt} \quad (1)$$

Utilizando a equação acima do controlador PID no domínio do tempo pode-se desenvolver uma equação de um controlador PID digital, para poder utilizá- la no Arduino. Para aproximar a equação 4 é utilizado uma taxa de amostragem com período T, assim a integral se aproxima ao somatório de todos os valores amostrados do erro multiplicado pelo período e a derivada pela diferença do erro atual e anterior dividido pelo período. A equação do PID digital é mostrada pela equação 2 [15].

$$PID = K_p\ e(t) + K_i\ T \sum_{i=0} e(i) + K_d \frac{e(t)-e(t-1)}{T}$$

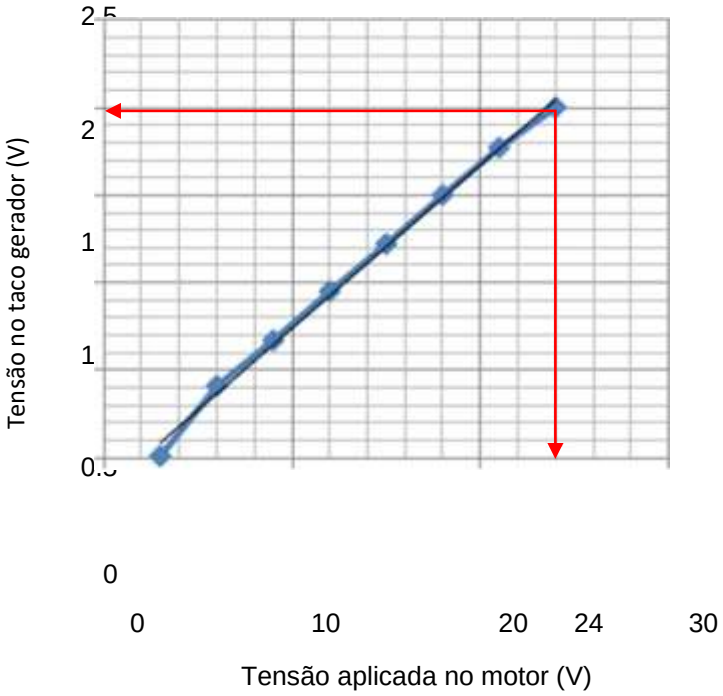
Essa equação será utilizada para desenvolver um algoritmo do controlador PID digital para a plataforma do *software* do Arduino. Será utilizado para controlar a rotação do motor, permitido modificar os parâmetros do controlador PID, o ganho proporcional, integral e derivativo.

3 PROCEDIMENTOS METODOLÓGICOS

3.1 LEVANTAMENTO TDOS PARÂMETROS ELÉTRICOS PARA CONTROLAR OMOTOR

No primeiro momento o foco do projeto foi voltado para identificação das características elétricas básicas do motor e do taco gerador, a fim de avaliar e dimensionar os circuitos eletrônicos auxiliares necessários para a interação do Arduino Uno aos periféricos. Desse modo, levantou-se as características do motor CC, que possui tensão nominal de 24 V e corrente nominal de 1 A. Também foi visualizado o comportamento do taco gerador em função da tensão aplicada no motor, mostrado pela Figura 3.

Figura 3 – Comportamento do taco gerador.



Fonte: Do autor, 2011.

Observa-se pela figura acima que a relação entre a tensão aplicada no motor e a gerada pelo taco gerador é linear, atingindo sua amplitude máxima gerada no taco gerador de 2 V com uma tensão nominal no motor de 24 V. A tensão fornecida pelo taco gerador será posteriormente conectada a uma entrada analógica do Arduino Uno, a fim de criar uma realimentação. Dessa forma fez-se necessário identificar as características da entrada analógica do Arduino para verificar o limite de tensão.

Conforme a fundamentação, a entrada analógica possui 10 bits de resolução e tem a capacidade de ler sinais elétricos de 0 a 5 Volts. Sabendo que a tensão máxima fornecida pelo taco gerador é de 2 V, possibilitou-se utilizar o recursoS do hardware do Arduino, a entrada de referência analógica que permite diminuir o range de tensão analógico. Nesse caso conecta-se uma tensão de 2 Volts a entrada de referência analógica e a entrada analógica passou a ter uma escala de 1.95 mV, pois pode ler sinais elétricos de 0 a 2 Volts. Assim não há necessidade de utilizar circuitos de amplificação para a tensão do taco gerador.

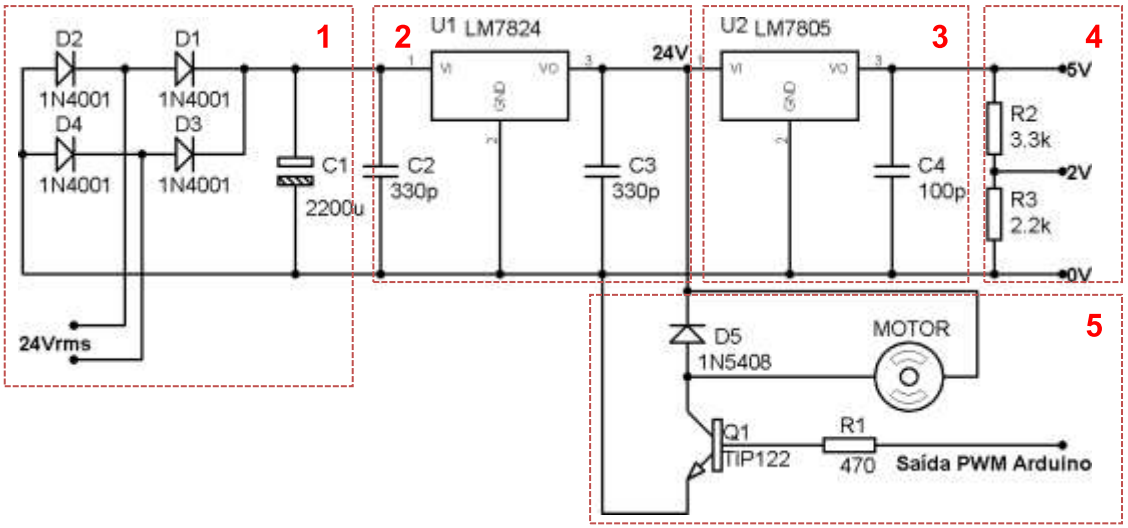
Definida a etapa de condicionamento do sinal do taco gerador observou-

se a corrente elétrica consumida pelo motor e os níveis de tensão necessários para alimentar a placa eletrônica do Arduino Uno. Abaixo é listada a descrição dos pré-requisitos necessários para posterior construção de uma placa eletrônica auxiliar.

- Tensão de alimentação do Arduino, sendo 5 Volts;
- Tensão de 2 Volts para referência analógica;
- Possuir um circuito para ligar e desligar o motor;
- Necessidade de utilizar um transistor para acionar o motor, devido a limitação de corrente elétrica na saída digital do Arduino, sendo no máximo 40 mA.

Após definido os níveis de tensões necessárias para suprir a demanda do experimento, elaborou-se o circuito com o auxilio do *software* Proteus. A Figura 4 a seguir mostra o circuito elaborado, este deve ser alimentado por uma tensão de 24 Volts alternado, possuir uma ponte retificadora de onda completa, um regulador de tensão de 24 Volts e de 5 Volts, um divisor de tensão de 2 Volts e um transistor de potência para ligar o motor.

Figura 4 – Circuito da placa de alimentação.



Fonte: Do autor, 2011.

A Figura 4 mostra o circuito da placa auxiliar que é dividida em cinco circuitos, nomeados de: ponte retificadora de onda completa, regulador LM7824, regulador LM7805, divisor de tensão e controle do motor. A Tabela 1 a seguir apresenta a finalidade de cada circuito.

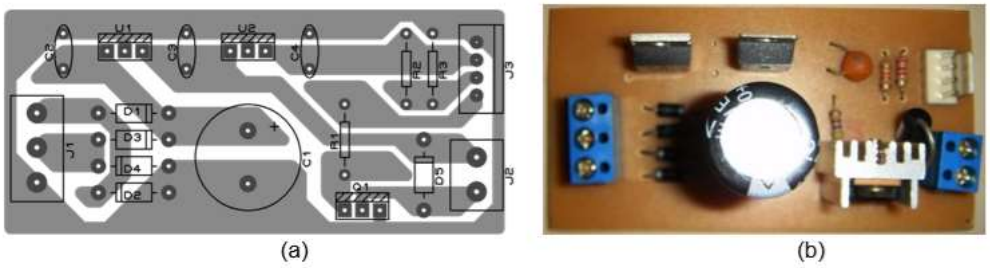
Tabela 1 – Descrição dos circuitos da placa de alimentação.

Circuito	Nome	Finalidade
1	Ponte retificadora de onda completa.	Converter o sinal de tensão alternado para em um sinal próximo ao contínuo.
2	Regulador LM7824.	Regular a tensão fornecida pela ponte retificadora a uma tensão de 24 Volts contínua.
3	Regulador LM7805.	Regular para 5 Volts a tensão fornecida pelo regulador LM7824.
4	Divisor de tensão.	Divide a tensão de 5 Volts fornecida pelo regulador LM7805 a uma tensão de 2 Volts.
5	Controle do motor.	Alimentar o motor e permitir ligá-lo e desligá-lo utilizando uma saída digital do Arduino.

Fonte: Do autor, 2011.

Com a conclusão da simulação da placa de alimentação auxiliar gerou-se o circuito impresso com o auxílio do *software* Proteus de acordo com a Figura 5 (a) e posteriormente a montagem da mesma. O resultado final da placa desenvolvida é mostrado pela Figura 5 (b).

Figura 5 – Circuito impresso e placa de alimentação.



Fonte: Do autor, 2011.

Com a placa finalizada foi preciso validar a sua funcionalidade, utilizando um osciloscópio para mensurar os níveis de tensão da placa. Para poder alimentar a placa, que necessita de uma tensão alternada de 24 Volts, usou-se umtransformador de 220 Volts para 24 Volts, podendo assim verificar a tensão fornecida pelos reguladores de tensão contidos na placa auxiliar desenvolvida.

3.2 CONTROLE DO MOTOR UTILIZANDO O ARDUINO

A validação da placa eletrônica auxiliar permitiu a alimentação do motor. Assim foi possível utilizar o Arduino para controlar o motor. Para poder controlar o motor foram realizados diversos testes a fim de aprovar a funcionalidade da placa criada e apontar possíveis erros não previstos no projeto. Cada teste possui um objetivo específico para controlar o motor e um resultado esperado.

Primeiramente o foco foi controlar o motor utilizando uma saída digital do Arduino para ligar e desligar o motor, sem utilizar qualquer sinal gerado pelo taco gerador. Assim uma saída digital foi conectada a entrada de controle do motor, que permite ligar o motor com o acionamento do transistor.

Para realizar tal função foi desenvolvido um algoritmo de controle na qual uma saída digital era acionada conforme a solicitação, permitindo controlar o motor utilizando o monitor serial que é um recurso do software do Arduino. Uma palavra específica enviada ao Arduino pelocomputador permitia ligar ou desligar o motor, assim foi desenvolvido um controle manual, porém digital em malha aberta.

No segundo momento o objetivo foi controlar a rotação do motor utilizando uma saída PWM do Arduino, mas para realizar a escolha da rotação do motor foi utilizado o monitor serial da mesma forma que no primeiro teste. Um novo algoritmo foi desenvolvido e do computador foi possível enviar o valor desejado da velocidade para o Arduino, valores entre 0 e 255, ou seja, valores de 0 a 24 Volts. Assim foi possível desligar o motor com o valor 0 e girar com a velocidade nominal utilizando ovalor 255. Conforme solicitado o motor visualmente girava rápido com valores maiores e lento com valores pequenos, parando em 0. Dessa forma foi possível controlar a rotação do motor manualmente. O Arduino controlava o motor, porém ainda permanecia um controle de malha aberta.

Após finalizado o controle em malha aberta o foco foi voltado a desenvolver um controle de malha fechada utilizando o sinal de tensão gerado pelo taco gerador, dessa forma foi utilizado o controle de rotação para verificar a mudança de tensão gerada pelo taco gerador conforme a velocidade do motor.

Assim a saída do taco gerador foi ligada a uma entrada analógica do Arduino. A entrada analógica 0 foi escolhida, mas poderia ser qualquer outra, isso permitiu visualizar a tensão captada pela entrada analógica no monitor serial. A

medida que se modificava a rotação do motor notou-se um resultado inesperado, pois a tensão gerada pelo taco gerador possuía uma enorme interferência, não foi possível visualizar um valor de tensão constante quando o motor girava em uma velocidade constante, mesmo em rotação nominal o valor variava aleatoriamente. Assim uma maneira para minimizar essa influência indesejada foi inserir um filtro passa baixa, permitindo diminuir o ruído captado pela entrada analógica, minimizando a influência das tensões de alta freqüência, resultando em um sinal de tensão mais constante. Optou-se por criar um filtro digital. Esse algoritmo foi desenvolvido no *software* do Arduino e pode ser visualizado nas linhas de código da Figura 6.

Figura 6 – Linhas de código do filtro digital.

```
byte filtro_digital(){
  float n = (float) 0.1; // potencia do filtro passa baixa
  static float filtro; // declara variavel filtro como float estatico
  filtro = n*(analogRead(0)/4) + filtro*(1-n); // equação do filtro passa baixa
  return (byte) filtro;  // retorna valor filtrado
}
```

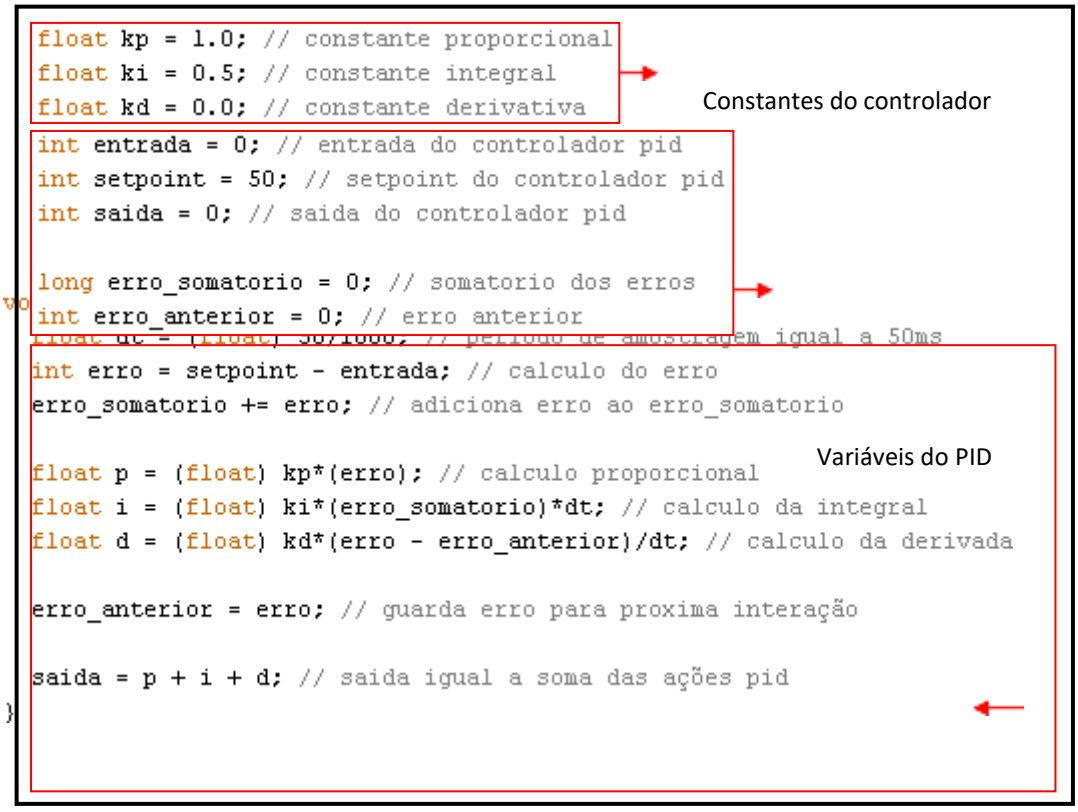
Fonte: Do autor, 2011.

A implementação do filtro digital permitiu-se uma melhora na resposta do sinal, isso possibilitou o avanço para o desenvolvimento de um controle automático, ou seja, utilizar a realimentação do sinal do taco gerador e criar um controle de malha fechada. Dessa forma o sinal captado pela entrada analógica pode ser analisado a fim de controlar a rotação do motor.

O controle de malha fechada adotado para ser utilizado no experimento foi o controle Proporcional-Integral-Derivativo (PID), para isso houve a necessidade de desenvolver um algoritmo do PID para o Arduino. Utilizando a equação 2 da fundamentação teórica foi possível implementar um algoritmo de controle PID digital.

O linhas do código implementado no *software* do Arduino podem ser visualizadas pela Figura 7.

Figura 7 – Linhas de código do algoritmo PID



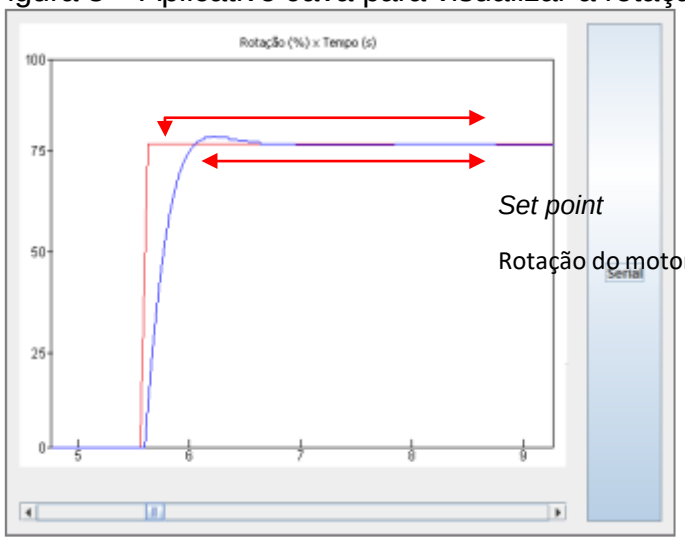
Fonte: Do autor, 2011.

Com o algoritmo do controle PID digital pronto, foi possível controlar o motor utilizando o controlador, nele os parâmetros do PID e o *setpoint* eram alterados utilizando o monitor serial. Mesmo com o controlador implementado não foipossível notar que o algoritmo realmente estava funcionando adequadamente.

Assim foi desenvolvido um aplicativo em Java para poder visualizar a rotação do motor, ou seja, mostrar em um gráfico o sinal analógico captado pelo Arduino e o *setpoint* escolhido.

A Figura 8 demonstra o aplicativo Java desenvolvido para visualizar a rotação do motor. Esse aplicativo utilizou a porta serial do Arduino para poder enviar valores referentes ao controle PID do motor e também para receber valores do *setpoint* e da entrada analógica do Arduino, mostrando-os na forma de gráfico.

Figura 8 – Aplicativo Java para visualizar a rotação do motor.



Fonte: Do autor, 2011.

Com o auxílio desse aplicativo, desenvolvido em Java, foi possível visualizar no gráfico a resposta da rotação do motor perante o controle PID implementado. Este permiti verificar no gráfico a resposta do sistema, conforme a escolha dos parâmetros do controlar digital. A finalização do controle utilizando o Arduino permitiu o avanço do projeto para a comunicação via rede.

3.3 COMUNICAÇÃO VIA INTERNET

A etapa de comunicação via internet necessitou um vasto tempo de pesquisa relacionada a biblioteca do *Ethernet Shield*, utilizada para compreender o processo de comunicação entre o Arduino e o *Shield*, pois essa possui exemplos e comentários que ajudaram a entender a o funcionamento da comunicação entre as placas.

A comunicação entre as placas Arduino Uno e *Ethernet Shield* é feita por SPI e possuem respectivamente os chips Atmega328 e ENC28J60. O chip ENC28J60 pode ser comparado com um grande buffer, ou seja, guarda pacotes recebidos da rede. Também guarda pacotes para serem enviados, assim o buffer é dividido em duas partes, uma para enviar e outra para receber dados da rede. Somente o chip *Ethernet* não faz a comunicação com a rede. É necessário um processador para analisar os pacotes e saber o que fazer com eles.

O chip Atmega328 faz o processamento dos pacotes se utilizar a biblioteca do *Ethernet* a qual possui funções específicas para diversos tipos de comunicação como TCP e UDP, permitindo identificar o protocolo e responder o pacote recebido.

Outro fator importante que limita o tamanho da página é que a biblioteca cria um buffer dentro do Arduino para guardar o pacote enviado e recebido. Nos exemplos da biblioteca é utilizado um buffer de 500 Bytes. O tamanho desse buffer é limitado pela SRAM do Atmega328 que é de 2000 Bytes, dessa forma limita o tamanho do buffer e da página *web*.

Considerando que o tamanho máximo do pacote é limitado pelo buffer, optou-se por utilizar uma estratégia diferente, ao invés de desenvolver uma página *web* para controlar o motor, foi criado um aplicativo na linguagem Java para estabelecer uma conexão com o Arduino.

O aplicativo desenvolvido estabelece uma conexão com o Arduino utilizando o protocolo de comunicação TCP, para isso foi utilizado um recurso da linguagem Java chamado *socket.* A biblioteca *socket* permiteabrir uma conexão com um determinado IP e porta de acesso, assim foi possível enviar e receber dados do Arduino pela internet.

Semelhante ao monitor serial do *software* do Arduino, esse aplicativo possui uma janela para enviar dados e outra para receber. Conforme a figura acima é destacado nas janelas: um bloco demonstra os dados enviados e o outro os dados recebidos do Arduino. As linhas do algoritmo para realizar essa função podem ser visualizada pela Figura 9.

Figura 9 – Linhas de código da comunicação pela rede

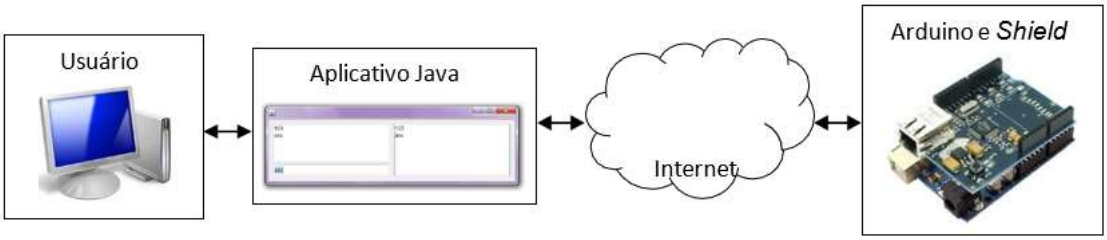
```
void setup(){
  server_init(); // inicia o chip enc28j60
}

void loop(){
  if (client_receive()){ // caso recebeu algum pacote de dados
    while(client_available()){ // enquanto tiver dados para ler
      client_write(client_read()); // le o dado e escreve no buffer
    }
    client_send(); // envia o pacote
  }
}
```

Fonte: Do autor, 2011.

O algoritmo desenvolvido tornou-se simples, pois foi criado um arquivo para simplificar a biblioteca. Esse arquivo é adicionado junto ao código principal do Arduino, que inclui a biblioteca do *Ethernet Shield* e outras funções que simplificam amesma. Todas as configurações estão presentes nesse arquivo, como configuração do MAC, IP e porta de acesso do *Shield.* Nele também pode ser modificado o tamanho do buffer do Arduino, que está configurado para 500 Bytes. O arquivo pode ser visualizado no Anexo A.

A implementação da comunicação entre o Arduino e o aplicativo Java permitiu o envio de mensagens específicas ao Arduino via internet. A Figura 10 demonstra a comunicação pela internet entre união das placas do Arduino e o aplicativo de comunicação TCP, que permite o envio de mensagens pelo usuário ao Arduino via internet. Possibilitando que o usuário troque informações com o Arduino a fim de controlar os dispositivos ligados ao Arduino.

Figura 10 – Diagrama de blocos da comunicação pela internet.



Fonte: Do autor, 2011.

Conforme a figura acima o aplicativo faz a ligação entre o usuário e o experimento utilizando a internet para ligá-los. Dessa forma é possível criar um aplicativo específico, sendo esse utilizado para controlar o experimento remoto. Assim qualquer interface do aplicativo Java poderia enviar uma mensagem específica para o Arduino. Um botão, por exemplo, permite enviar essa mensagem, que para o Arduino significaria ligar o motor. Portanto, o usuário desconhece os comandos que são válidos no código do Arduino, tornando o sistema seguro.

3.4 DESENVOLVIMENTO DO AMBIENTE VIRTUAL

Finalizada a comunicação do Arduino via internet os esforços foram voltados à criação de um ambiente de interação entre o usuário e o experimento

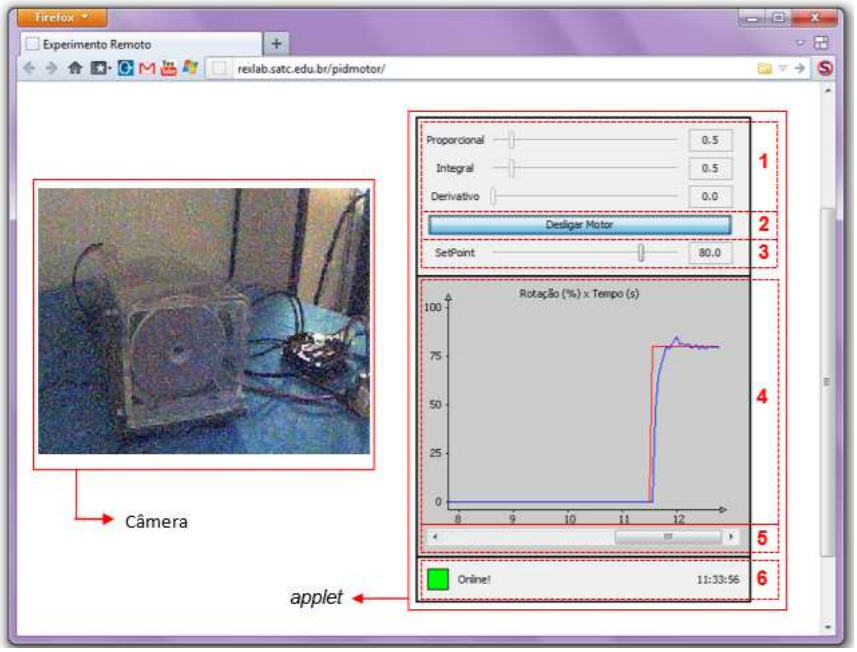
remoto.

O ambiente virtual foi desenvolvido na linguagem Java, sendo esse o local utilizado para criar uma conexão entre o usuário e o experimento remoto. O usuário poderá interagir com o experimento utilizando comandos disponíveis no aplicativo, permitindo realizar a experiência no ambiente virtual utilizando a internet como meio de comunicação para receber informações do experimento e controlá-lo.

O ambiente desenvolvido em Java será utilizado especificamente para comunicação com o experimento remoto via internet ou rede local. Esse aplicativo Java é um *applet*, e permite ser hospedado em uma página *web* similar a uma figura. Assim esse aplicativo se auto-instala no computador do usuário, permitindo utilizar todos os recursos do aplicativo dentro do próprio navegador.

O aplicativo *applet* desenvolvido foi inserido em uma página conforme a Figura 11. Nessa página será possível visualizar o motor, pois será utilizado uma *webcam* que permite o usuário visualizar a rotação do motor conforme é solicitada ao aplicativo. Esse permite ligar e desligar o motor, bem como modificar os parâmetros do controlador PID e visualizar no gráfico a rotação do motor.

Figura 11 – Página com o aplicativo *applet*



Fonte: Do autor, 2011.

A página *web* acima demonstra o local onde será possível visualizar o motor, a fim de verificar se realmente o motor gira. Ao lado o aplicativo *applet* que

proporciona ao usuário visualizar no gráfico a rotação do motor e controlar os parâmetros do controlador PID. Esse aplicativo possui diversos mecanismos para interagir com o usuário, as funcionalidades de cada item apontado na Figura 11 são descritas na Tabela 2.

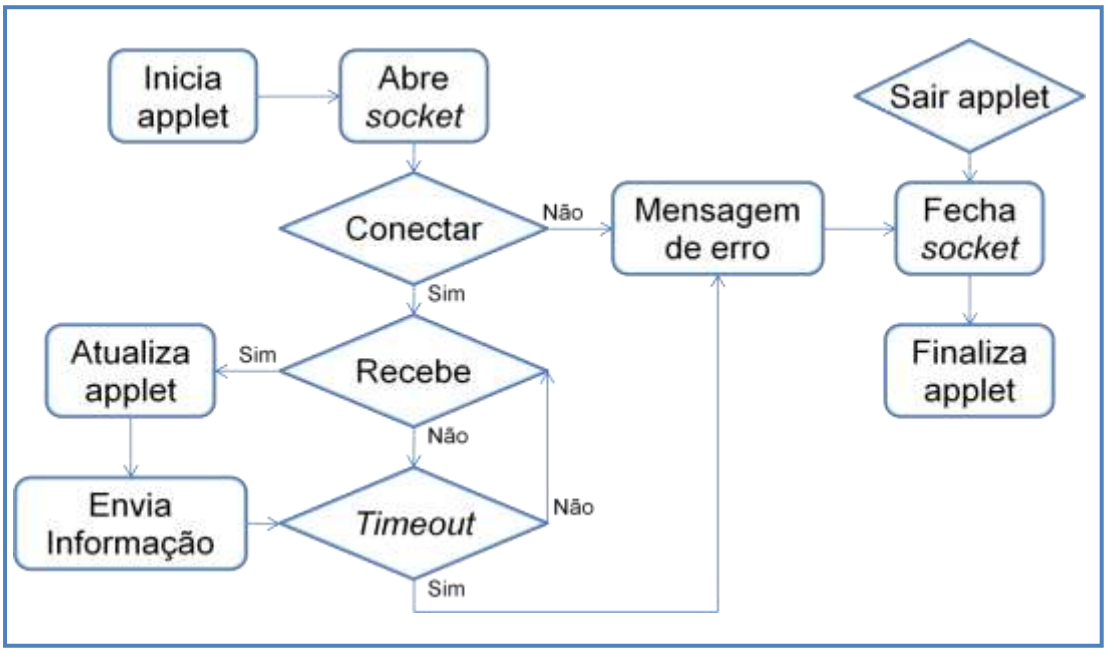
O *applet* desenvolvido possui a habilidade de se comunicar com o experimento pela rede, a interação entre o algoritmo do Arduino e o aplicativo permite controlar o motor e receber informações do mesmo. O aplicativodesenvolvido possui uma lógica de controle que é demonstrada pelo fluxograma da Figura 12.

Tabela 2 – Descrição dos blocos do ambiente desenvolvido.

Bloco	Nome	Finalidade
1	Parâmetros do controlador PID	Selecionar os valores dos parâmetros do controlador PID, ou seja, os valores das constantes, proporcional, integral, e derivativa.
2	Botão liga desliga	Ligar ou desligar o motor.
3	Seleção do *setpoint*	Escolher o valor desejado da rotação do motor onde o controlador deve atuar.
4	Gráfico da rotação	Visualizar os valores de rotação pelo tempo do motor e do *setpoint*, identificados respectivamente pela cor azul e vermelha.
5	Barra do gráfico	Paralisar o gráfico para visualizar valores anteriores de rotação.
6	*Status* do aplicativo	Mostrar o horário e mensagens do funcionamento do aplicativo.

Fonte: Do autor, 2011

Figura 12 – Fluxograma do aplicativo *applet*.



Fonte: Do autor, 2011

O fluxograma da Figura 12 demonstra o funcionamento do *applet.* Ao entrar na página *web* o aplicativo é executado no computador do usuário e inicia os parâmetros e variáveis do mesmo, abre uma conexão TCP utilizando o método *Socket* da linguagem Java, conforme o Anexo B. Caso a conexão não seja estabelecida uma mensagem de erro é demonstrada no bloco de *Status* do aplicativo e o mesmo é finalizado.

Com a conexão estabelecida com o Arduino o *applet* entra em um *loop,* se receber alguma informação do Arduino o aplicativo é atualizado, principalmente o gráfico e são enviadas informações a placa. O *timeout* é um tempo definido utilizado para desconectar e finalizar o aplicativo quando este ficar um tempo sem receber mensagem da rede, caso ocorra, outra mensagem de erro é exibida no bloco de *Status*. Caso o usuário saia da pagina que contem o aplicativo, esse é finalizado permitindo fechar a conexão TCP.

4 ANÁLISE DOS RESULTADOS

Com o experimento remoto finalizado foi possível visualizar no gráfico os valores da rotação do motor a partir da mudança dos valores dos ganhos do controlador PID e *setpoint*. A Tabela 3 demonstra os valores dos parâmetros

utilizados no aplicativo Java para realizar testes no experimento remoto.

Tabela 3 – Testes realizados no experimento.

Setpoint	60%			Figura
Controlador	Kp	Ki	Kd	
P	1.0			13a
	5.0			13b
PI	5.0	0.5		14a
	5.0	2.0		14b
PID	5.0	0.5	0.5	15a
	5.0	2.0	0.5	15b

Fonte: Do autor, 2011.

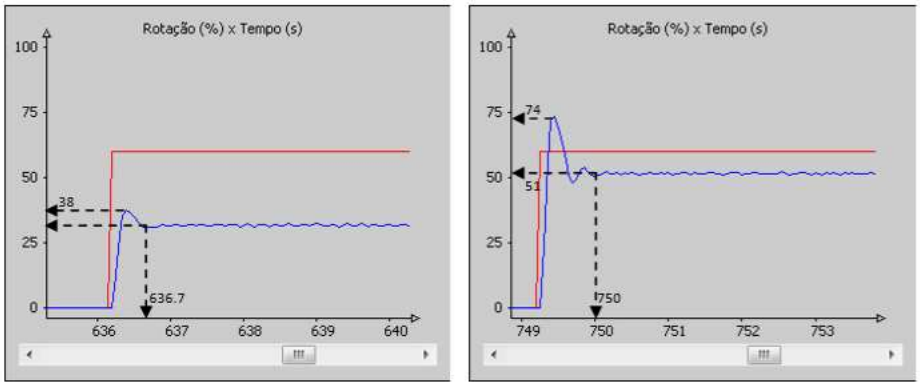
Os diversos testes realizados conforme a tabela acima permitiu verificar a resposta da rotação do motor. A seguir são mostradas as figuras de cada teste realizado relacionando os valores dos ganhos do controlador PID com as figuras na referida tabela.

Com os devidos testes realizados foi possível verificar o funcionamento do controlador, permitindo assim o funcionamento adequado da ação de controle proporcional, integral e derivativo. Posteriormente será utilizado o *software* MATLAB para realizar simulações, comparando os gráficos gerados pelo aplicativo e os gráficos simulados.

A seguir a Figura 13 demonstra a resposta do motor quando utilizado um controlador proporcional puro, com valores diferentes de ganho proporcional e de *setpoint*.

A Figura 13 abaixo apresenta um controlador proporcional, conforme a Tabela 3 com alterações no valor do ganho proporcional e do *setpoint.* Por ser um controlador somente proporcional, não anula o erro em regime permanente, mantendo a dinâmica descrita na teoria de controle. Ou seja, possui uma diferença visível entre o valor da rotação do motor e do *setpoint*.

Figura 13 – Aplicativo Java com controlador P

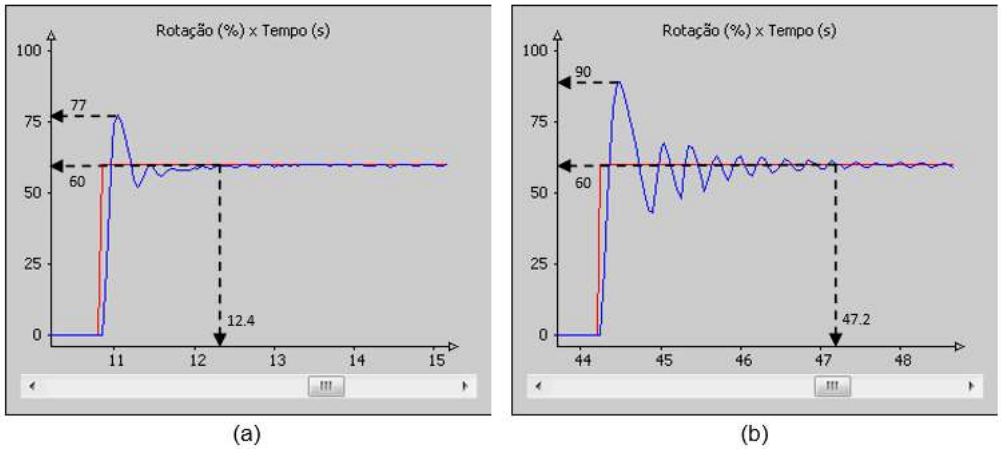


Fonte: Do autor, 2011.

A seguir são mostrados gráficos do aplicativo utilizando um controlador proporcional e integral, podendo demonstrar o desempenho desse controlador na Figura 14.

Com a inclusão do controlador integral, visualizado pela Figura 14, é possível verificar a sua atuação no controle da rotação do motor. O ganhoproporcional permanece constante, somente os valores do setpoint e do ganho integral são modificados. É visível que a adição do controlador integral ao proporcional permite levar o erro em regime permanente ao nulo, porém um ganho elevado da constante integral aumenta o valor de sobre sinal e a oscilação do sistema. O aumento do setpoint faz com que o sistema fique menos instável, diminuindo a oscilação no regime transitório.

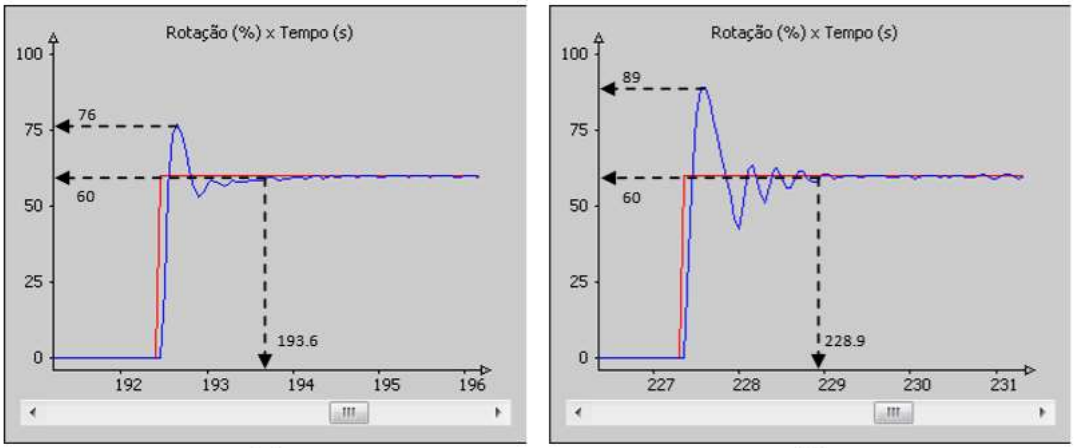
Figura 14 – Aplicativo Java com controlador PI



Fonte: Do autor, 2011.

A seguir os gráficos do controlador PID são demonstrados pela Figura 15, utilizando um valor de *setpoint* de 60%. A Figura 15 demonstra o controlador PID quando utilizado um valor de *setpoint* de 60%, nesses gráficos são alterados os valores do ganho integral e derivativo,enquanto o valor do ganho proporcional permanece constante. A atuação do controlador derivativo permite diminuir a oscilação do sistema e com o ganho integralelevado o sistema tende a se tornar instável.

Figura 15 – Aplicativo Java com controlador PID – 60%

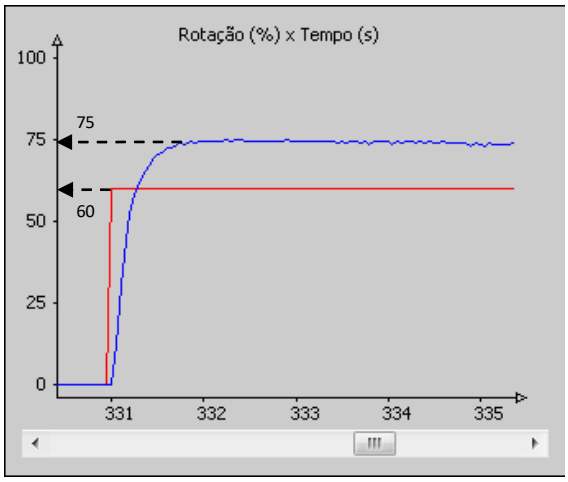


Fonte: Do autor, 2011.

4.1 SIMULAÇÕES NO MATLAB

Para poder realizar simulações foi utilizado o *software* MATLAB. Para isso foi preciso identificar a função de transferência do experimento, utilizando assim o aplicativo Java, que possibilitou monitorar a resposta da rotação do motor em malha aberta, permitindo a utilização de métodos para modelar a função de transferência do sistema a partir do gráfico. A Figura 16 a seguir demonstra a resposta do sistema em malha aberta.

Figura 16 – Aplicativo Java com controlador em malha aberta.



Fonte: Do autor, 2011.

Com o gráfico obtido da rotação do motor referente a uma entrada de 60% da rotação máxima do motor, utilizou-se análise do gráfico para encontrar os modelos matemáticos que representem a resposta do sistema, permitindo identificar as respectivas funções de transferência. O método de Hägglund encontrado no Anexo D, modela uma função de transferência de primeira ordem e o método de Mollenkamp encontrado no Anexo E, uma função de segunda ordem.

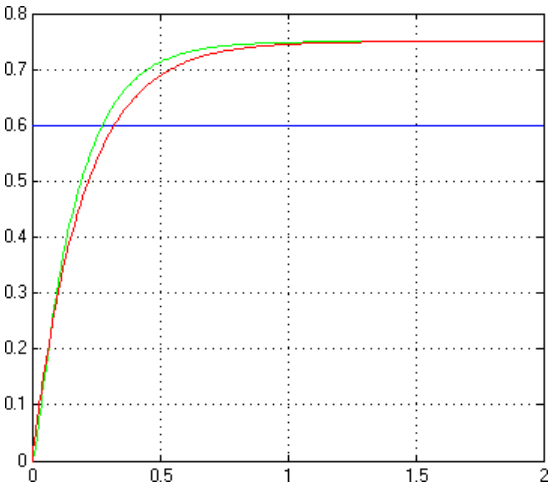
As funções de transferência modeladas a partir do gráfico da Figura 16 são mostradas a seguir, representadas pela função de primeira ordem (H1) e de segunda ordem (H2).

$$H1(s) = \frac{1.25}{0.2 * s + 1}$$

$$H2(s) = \frac{1.25}{0.00216 * s^2 + 0.1753 * s + 1}$$

Com o auxílio do *software* MATLAB foi simulado as duas funções de transferência simultaneamente, assim foi possível simular diversos tipos de controladores. A seguir a Figura 17 demonstra a simulação das funções de transferência, nela é possível visualizar o *setpoint,* H1 e H2, respectivamente pelas cores azul, vermelho e verde.

Figura 17 – Simulação das funções de transferência em malha aberta.



Fonte: Do autor, 2011.

Para comparar o sistema real com o modelado foi simulado diversos controladores utilizando as funções de transferência obtidas a partir do gráfico em malha aberta.

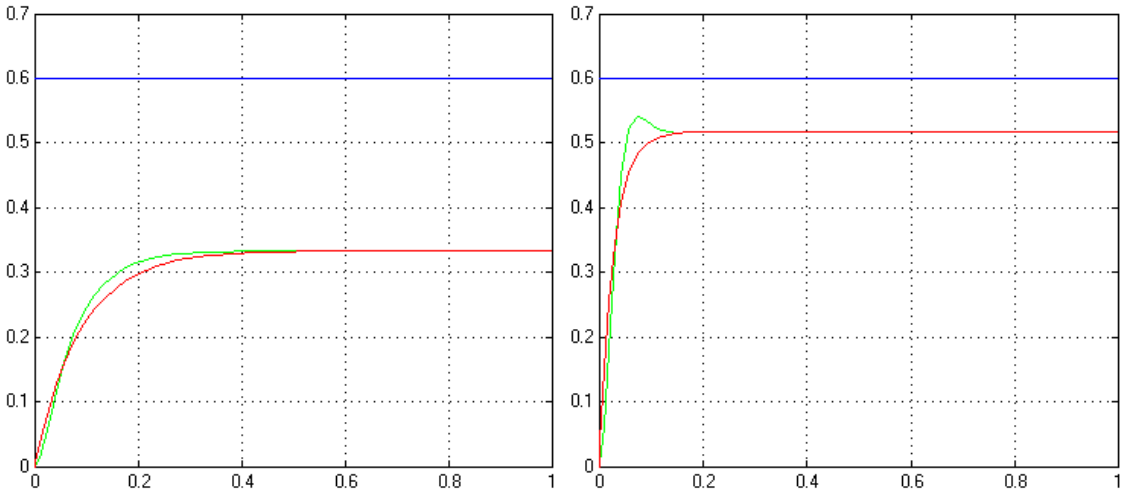
A Figura 17, Figura 18 e Figura 19 são gráficos gerados pela simulação dos modelos matemáticos e representam sistemas de malha fechada com um controlador PID. Os parâmetros do controlador e o *setpoint* podem ser visualizados pela Tabela 4 a seguir.

Tabela 4 – Testes simulados.

Setpoint	60%			Figura
Controlador	Kp	Ki	Kd	
P	1.0			17a
	5.0			17b
PI	5.0	0.5		18a
	5.0	2.0		18b
PID	5.0	0.5	0.5	19a
	5.0	2.0	0.5	19b
	5.0	0.5	2.0	19c
	5.0	2.0	2.0	19d

Fonte: Do autor, 2011.

Figura 17 – Simulação do controlador P

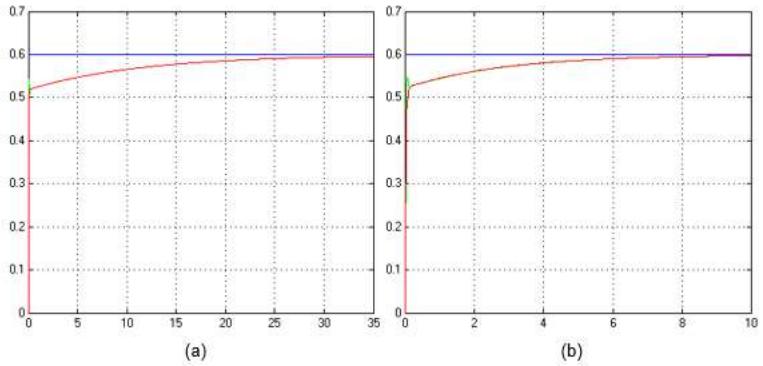


Fonte: Do autor, 2011.

A Figura 17 demonstra um controlador proporcional, nele é possível visualizar a diferença da função de transferência de primeira ordem e segunda ordem. Quando a constante proporcional é elevada, verifica-se o surgimento de um sobre sinal.

Comparando o grupo da Figura 13 e Figura 17 é possível visualizar que a resposta real possui um sobre sinal semelhante a função de segunda ordem, assim essa função se assemelha mais com a resposta real da rotação do motor, mesmo que as duas funções de transferência em regime permanente resultem no mesmo valor de rotação. A seguir o grupo da Figura 18 representa a simulação de um controlador proporcional e integral.

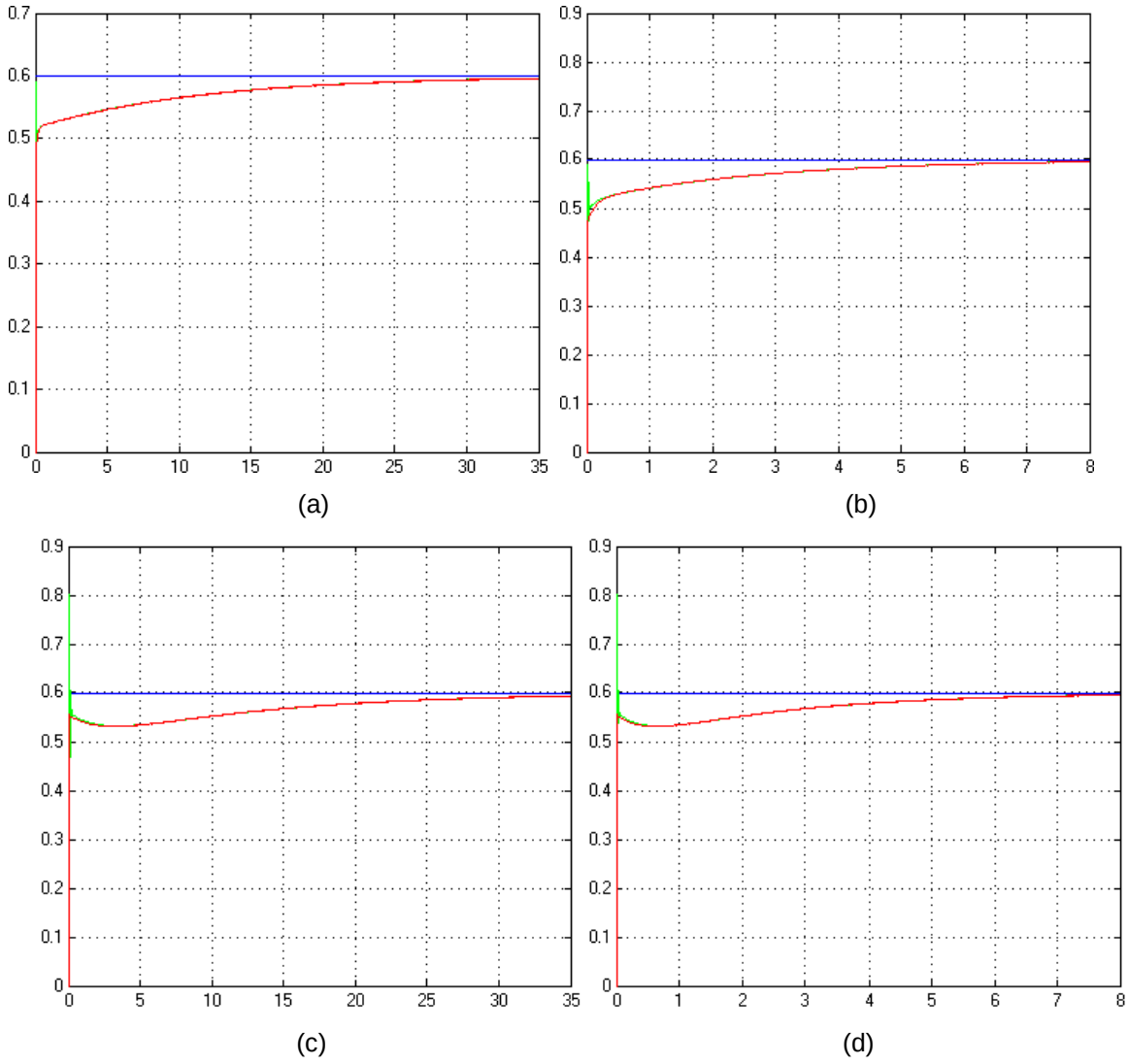
Figura 18 – Simulação do controlador PI



Fonte: Do autor, 2011.

A adição do controlador integral pode ser visualizada pela Figura 18. A inclusão do controlador integral fez com que o erro em regime permanente se anule. Essa figura mostra o controlador PI simulado modificando o valor do ganho integral epermitindo que o sistema se estabilize mais rapidamente com o aumento desse ganho. Conforme visto na Figura 14 o sistema real também elimina o erro em regime permanente com a inclusão do controlador integral. O controlador PID simulado é demonstrado pela Figura 19, 60% da rotação.

Figura 19 – Simulação do controlador PID – 60%.



Fonte: Do autor, 2011.

Quando inserido o controlador derivativo no simulador surge o sobre sinal, mais visível quando o valor ganho derivativo é maior. O controlador PID do sistema real (Figura 15) e o simulado (Figura 19) são similares, pois anulam o erro em regime permanente, porém o sistema real possui uma resposta rápida comparado com o sistema simulado.

5 CONCLUSÃO

O controlador PID empregado no experimento resultou no controle adequado da rotação do motor, conforme os ganhos utilizados e a resposta do sistema atendeu o esperado. Com as simulações feitas a partir das funções de transferência obtidas por análise gráfica foi observado a convergência das respostas do sistema físico implementado com as teóricas de sistemas de controle.

Na implementação do protótipo do sistema de controle foram aproveitados componentes já existentes na instituição. A integração da instrumentação com os elementos de potência foi feita utilizando-se um microcontrolador, que por meio de um *software* possibilita ao usuário observar o desempenho e interagir com o sistema, assim atuando sobre o controle da rotação do motor.

A implementação do software de interação e do hardware realizados permitiu que o controle fosse acessado de locais externos a rede da instituição de ensino, em diferentes horários, demonstrando o funcionamento remoto adequado.

No desenvolvimento de experimentos remotos aplicados na área da educação, em específico a sistemas de controle, observou-se algumas dificuldades relacionadas ao material bibliográfico para subsidiar os conceitos necessários ao desenvolvimento de experimentos referentes a sistemas de controle. Dentro deste contexto podemos citar os poucos livros para auxiliar na compreensão dos *softwares* utilizados para implementar a proposta.

Este trabalho pode ser aplicado em estudos futuros como um incentivo a desenvolver novos experimentos remotos, utilizando-se das ferramentas de *hardware* e *software* contidas neste. Os arquivos desenvolvidos no ambiente Arduino serão de grande ajuda para futuros desenvolvedores de experimentos remotos utilizando essa plataforma.

REFERÊNCIAS

Arduino. Disponível em: <http://arduino.cc/>. Acesso em: 30 ago. 2011.

FONSECA, Erika Guimarães Pereira da; BEPPU, Mathyan Motta. Arduino**.**Niterói: Universidade Federal Fluminense, 2010. 23 p.

Arduino Uno. Disponível em: <http://arduino.cc/en/Main/ArduinoBoardUno>.Acesso em: 30 ago. 2011.

Arduino Ethernet Shield. Disponível em: <http://arduino.cc/en/Main/ ArduinoEthernetShield>. Acesso em: 28 jul. 2011.

BRUGNARI, Arthur; MAESTRELLI, Luiz Henrique Mussi. AUTOMAÇÃO RESIDENCIAL via WEB. Curitiba: Pontifícia Universidade Católica Do Paraná, 2010. 36 p.

BOLTON, W. Engenharia de Controle. São Paulo: Makron Books do BrasilEditora Ltda, 1995.

BRISTOT, Vilmar Menegon. CONTROLE DE TEMPERATURA DE SECADORES DE REVESTIMENTOS CERÂMICOS ALIMENTADOS COM GÁS NATURAL. Florianópolis: Universidade Federal De Santa Catarina, 2002. 109 p.

BRUCIAPAGLIA, A. H.; 1992. Desenvolvimento do Controlador PID – AA/UFSC: Processos com Atraso de Transporte Dominante. Universidade Federal deSanta Catarina, Florianópolis - Brasil.

POSSER, M.S.; PID – Toolbox : Uma Ferramenta para o Ensino e Ajuste de Controladores PID's. Departamento de Engenharia Química, Universidade Federal do Rio Grande do Sul.

NORMEY-RICO, J. E.; CAMACHO, E. F.; GOMEZ-ORTEGA, Juan; 1998. Robustez em Controladores Preditivos Generalizados. In: XII CONGRESSO BRASILEIRO DE AUTOMÁTICA (Set. 1998 : Uberlândia, Minas Gerais). Anais.Minas Gerais, 1998. p.157-162.

DE CARVALHO, J.L. Martins. Sistemas de Controle Automático. Livros Técnicose Científicos Editora, Rio de Janeiro, 2000.

CAMPOS, Mário César M. Massa de; TEIXEIRA, Herbert C. G. ControlesTípicos de Equipamentos e Processos Industriais. São Paulo: Editora Edgard Blucher Ltda., 2006.

COELHO, Antônio Augusto Rodrigues; COELHO, Leandro Dos Santos, 2004. "Identificação de Sistemas Dinâmicos Lineares", Editora UFSC, Santa Catarina.