UNIVERSIDADE FEDERAL DE MINAS GERAIS

TRABALHO PRÁTICO 3:

CONTROLE

Introdução à Robótica
Professor: Mário Fernando Montenegro Campos

Grupo 5 – Gigante Guerreiro Daileon:
Leonardo Palhares
Moisés Lisboa
Rafael Pissolato
Tiago Amadeu

Objetivo:

Familiarizar o aluno com sensores ópticos e técnicas básicas de controle.

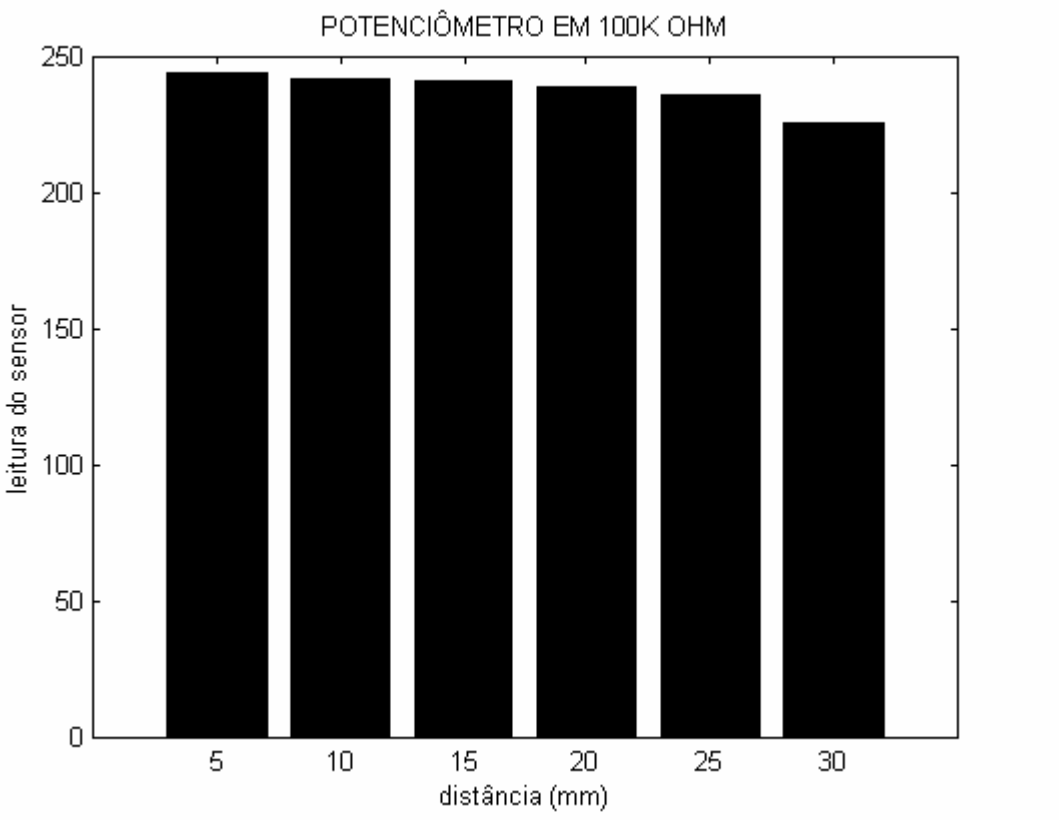
1. Utilizando sensores ópticos:

Exercício 1:

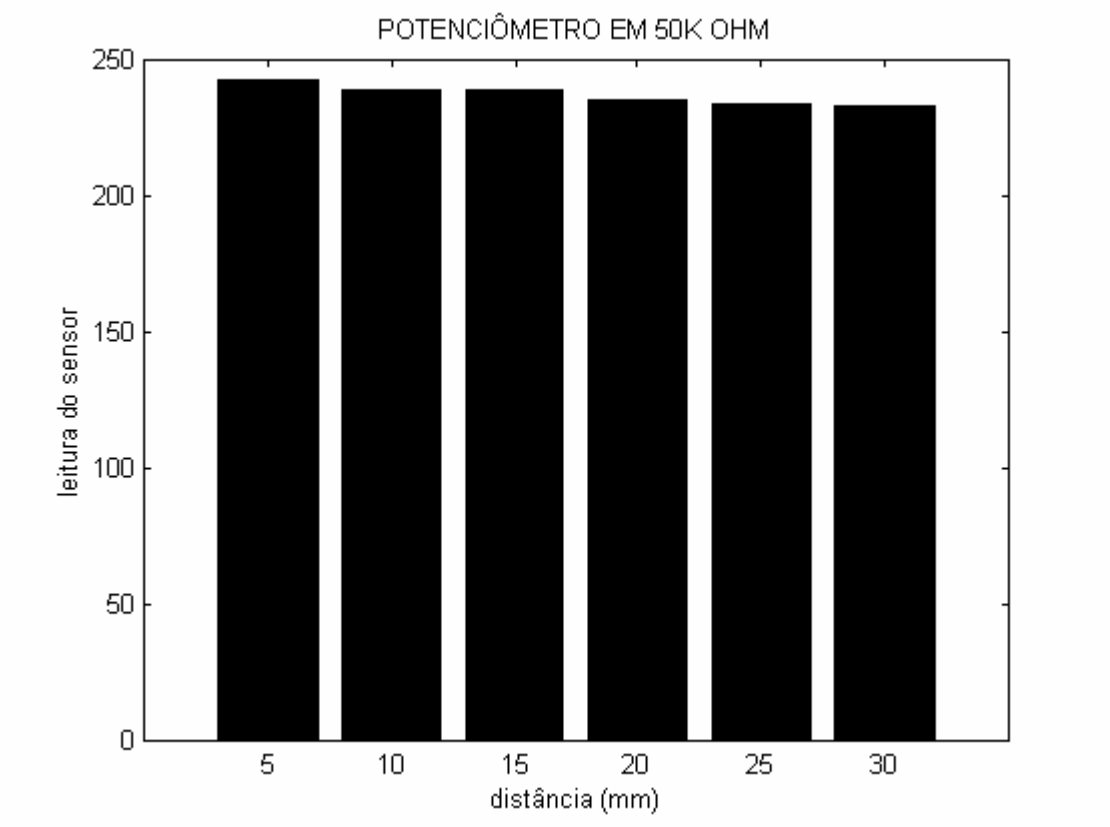
Usando um potenciômetro em série com o fototransistor do sensor óptico (e ligando-o ao +5V da Handy Board), o objetivo aqui é definir o valor da resistência que nos fornece uma melhor performance das leituras desse sensor. É importante lembrar que a Handy Board já possui uma resistência de sinal interna de 47K ohm e, realizando o procedimento atual, estamos montando o potenciômetro em paralelo a essa resistência interna.

O procedimento que utilizamos para obter a melhor performance do sensor foi utilizando dois potenciômetros de 100K ohm cada em série, obtendo assim um potenciômetro de 200K ohm. Então, escolhíamos um valor de resistência no potenciômetro e fazíamos medidas do sensor óptico para várias distâncias diferentes do sensor em relação à um objeto. O objeto que escolhemos para fazer as medidas foi um bloco de isopor amarelo, assim como o utilizado no TP anterior. Portanto, nosso procedimento foi baseado na experimentação, construindo as curvas dos resultados que obtivemos, e então analisando essas curvas de forma a determinar o valor de resistência que fornece a melhor performance do sensor óptico.

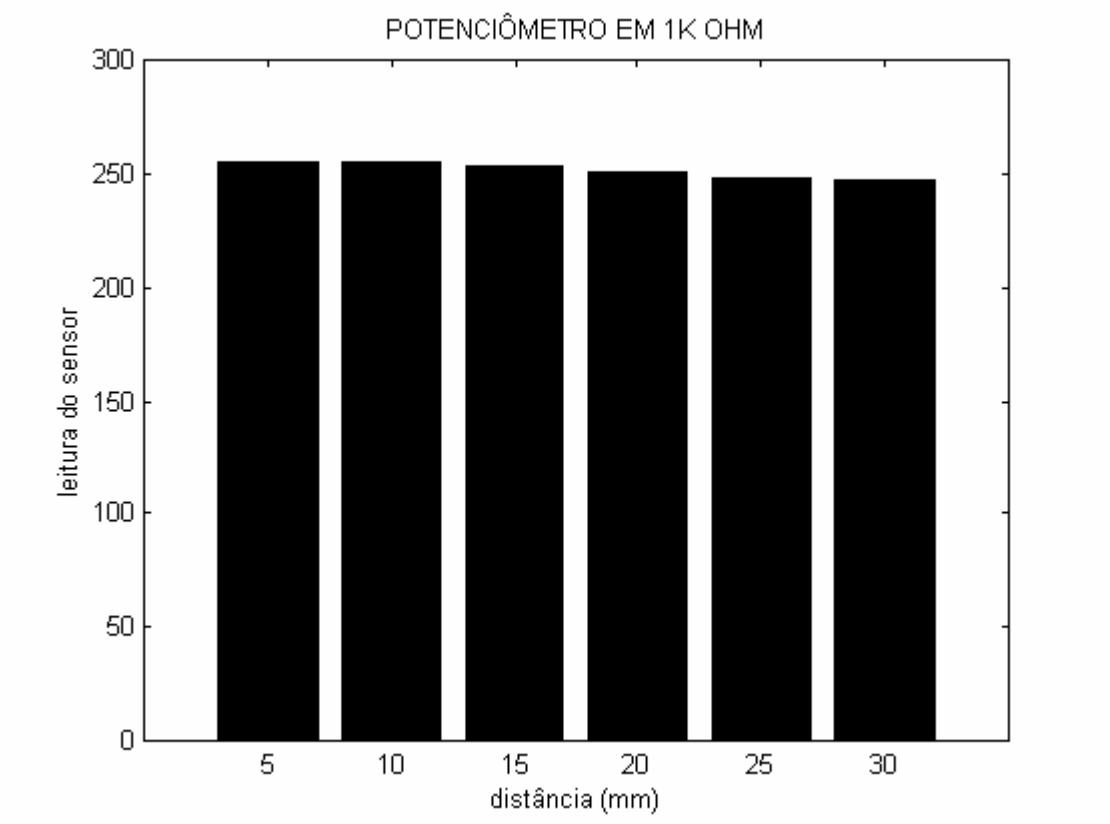
Começamos com o potenciômetro em 100K ohm. Os resultados estão resumidos na curva abaixo:



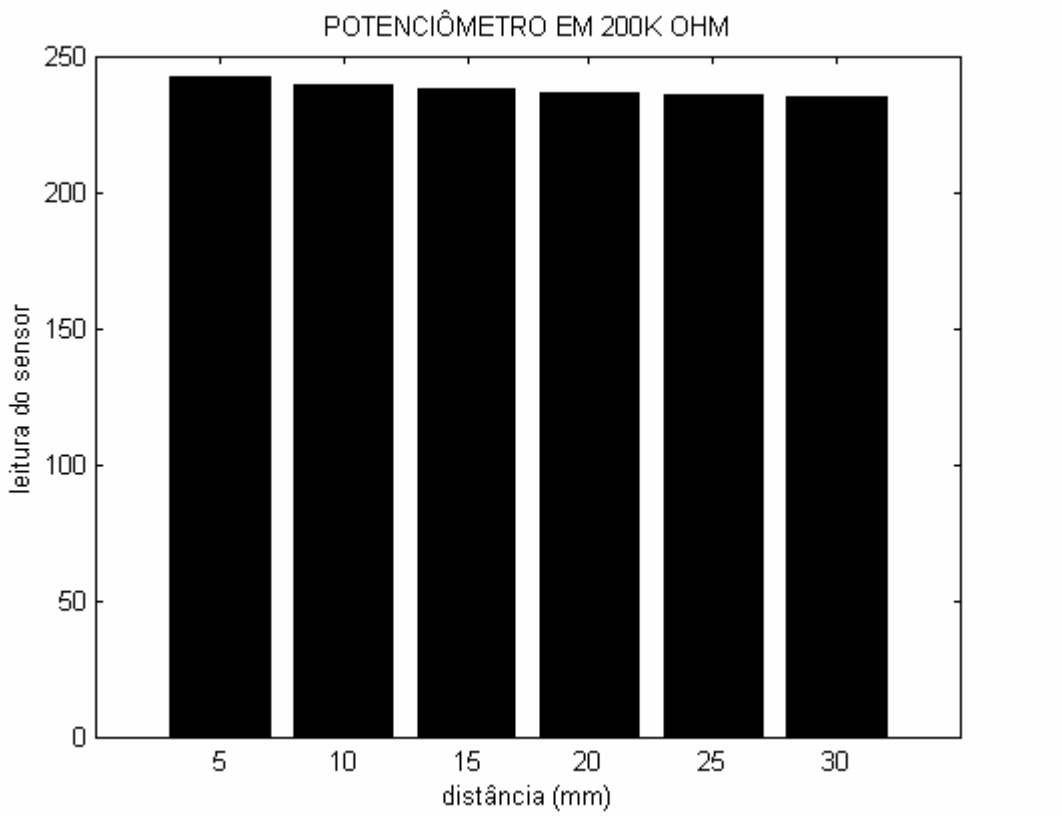
A seguir, baixamos o potenciômetro para 50K ohm. Abaixo os resultados:



Como as curvas não apresentaram muita diferença, colocamos o potenciômetro em 1K ohm, obtendo a seguinte resposta do sensor óptico:



Como pode-se perceber claramente, essa faixa de valores de resistência não nos atende, pois o sensor responde numa faixa de valores muito estreita. Então, colocamos o potenciômetro em 200K ohm, obtendo o seguinte:



Essa curva também não atende à nossos propósitos, pois também apresenta uma faixa de valores muito estreita. Levando-se em consideração uma certa linearidade do processo, seria desperdício fazer mais medidas dentro dessa faixa de 1K – 200K ohm, pois não conseguiríamos melhores resultados.

Concluímos então que colocar uma resistência em série com o fototransistor do sensor óptico que temos em mão só piora as medidas desse sensor, e o melhor a se fazer é ligá-lo diretamente nas entradas analógicas da Handy Board. Como justificativa, apresentamos o argumento de que o sensor óptico que temos já é um conjunto, constituído de um diodo emissor e um fototransistor, que já foram projetados de forma otimizada. E muito provavelmente, a resistência em série com o fototransistor já foi escolhida no projeto e implementada no sensor, de forma que seu ajuste se torna útil apenas em casos especiais.

Exercício 2:

a) Assumindo que o LED do sensor óptico tem uma queda de 2,0V e que ele esteja ligado em série a um resistor de 330 ohm e uma fonte de +5V, calculamos a corrente que flui pelo circuito:

$$I = (5V - 2V) / 330\ ohm = 9{,}1mA$$

b) Para múltiplos sensores (múltiplos diodos), ao invés de reservar uma entrada exclusiva pra cada diodo, podemos conectar vários à mesma entrada, economizando as portas da Handy Board. Para isso, conectamos os conjuntos diodo-resistor em paralelo, todos recebendo alimentação de +5V. Supondo que a corrente máxima que a Handy Board pode fornecer é de 500mA (esse dado não pôde ser confirmado, já que não encontramos essa especificação no manual da Handy Board), a quantidade máxima de LEDs que podemos conectar em paralelo são:

$$nº\ LEDs = 500mA / 9{,}1mA = 54\ Leds$$

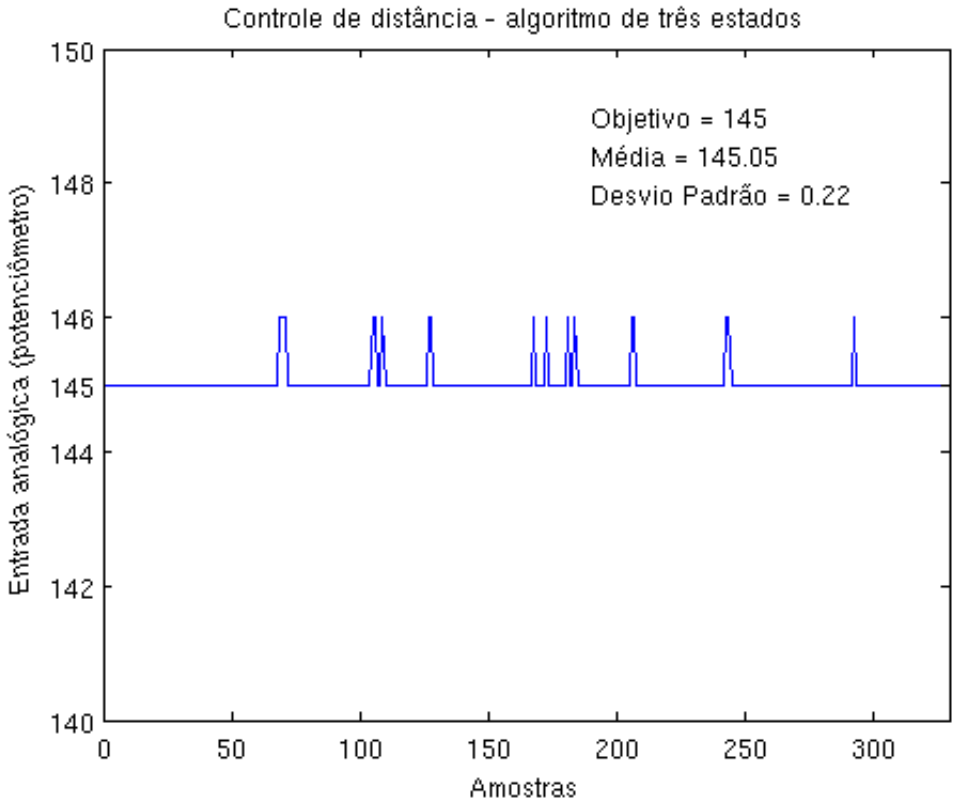
2. Controle simples com realimentação:

Ao invés de utilizarmos o “bend-sensor” ou o sensor óptico para “medir” a distância do robô à parede, com o intuito de fazê-lo seguir a parede em linha reta, construímos um sensor baseado em um potenciômetro, uma

mola e uma superfície plana rígida e leve que fica em contato com a parede. Com a variação da resistência do potenciômetro conforme o robô se distancia ou aproxima da parede, temos o nosso sensor de distância, o qual utilizamos para realizar os exercícios a seguir:

Exercício 1:

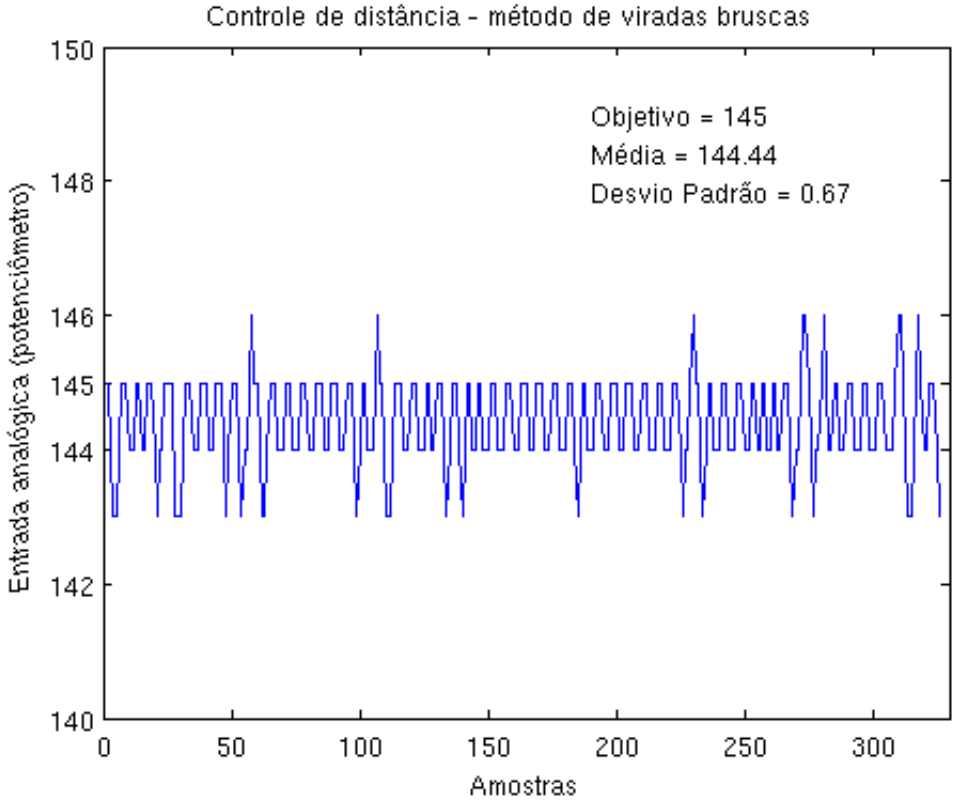
Usando um algoritmo de três estados, construímos o código em IC que se encontra na página do grupo. Nesse algoritmo, as três ações possíveis do robô são baseadas em dois valores de referência. Quando acima do valor máximo, o robô vira com potência total em um sentido. Quando abaixo do valor mínimo, o robô vira com potência máxima no outro sentido. Quando entre os valores de referência, o robô permanece em linha reta. Graficamente, obtivemos o seguinte resultado:

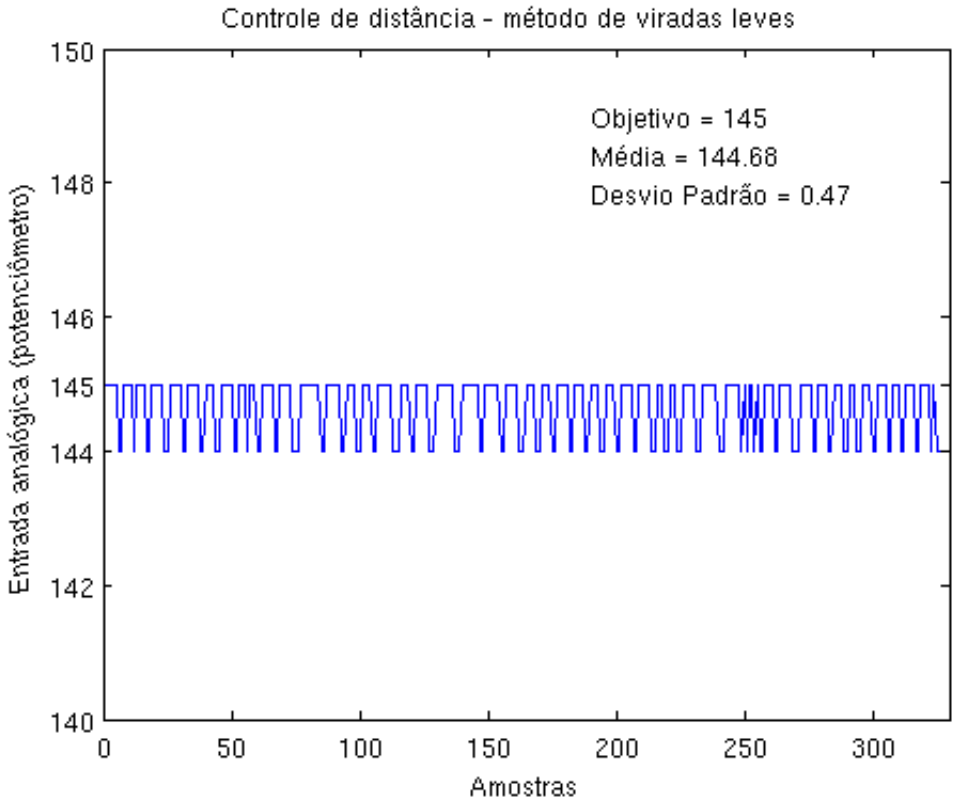


Comparando os resultados que obtivemos com os relatados no livro, observamos algumas diferenças. A primeira é devida a leitura do sensor, que no nosso caso, não ficava o tempo todo encostado na parede como deveria, porque a resposta do robô é lenta. Daí tem-se a impressão que o robô ficou bastante tempo na posição correta, e de repente saía. Porém, observando a distância entre os picos do gráfico, podemos concluir que o método de três estados funcionou muito bem, pois o robô teve de fazer menos correções. Com relação ao desvio padrão e à média, não é possível tirar muitas conclusões, pois o "bend-sensor" não estava encostado todo o tempo na parede.

Exercício 2:

Para o algoritmo de seguir parede, comparamos os métodos de virar bruscamente e virar levemente quando o robô estiver entrando ou saindo de uma curva, ao invés de seguir uma parede em linha reta. Nesses algoritmos (implementados em IC e disponíveis na página do grupo), só existe um valor de referência, e o robô sempre está virando para um lado ou para o outro. No método de virar bruscamente, o robô sempre vira com potência máxima, já no método de virar levemente o robô vira com metade da potência. Graficamente, abaixo apresentamos os resultados das duas implementações, para o robô seguindo uma parede sem curva:





Comparando os resultados obtidos, observamos que o método de viradas leves responde melhor, permanecendo mais próximo da distância desejada. Os valores do desvio padrão de cada curva também denunciam o melhor resultado do método de viradas leves do que o método de viradas bruscas, no caso estudado.

Obs.: Devido a problemas de construção do robô, não foi possível realizar o teste dos algoritmos de virada leve e brusca para seguir paredes em curvas. Porém, visto que no ambiente onde vamos realizar a apresentação não existe curvas suaves, mas somente quinas de 90º, a ausência desse experimento não afetará demasiadamente nossa apresentação.

Exercício 3:

Nesse exercício, teríamos de testar o algoritmo de três estados quando o robô tem de fazer curvas suaves. Porém, pelos mesmos motivos anteriores, esse teste não foi realizado, o que também não prejudicará de forma relevante nossa apresentação.

Exercício 4:

Generalizamos o algoritmo de três estados para fazer curvas mais fortes à medida que a distância de referência à parede aumenta. Aqui, estamos dando início ao controle proporcional do robô na tarefa de seguir parede.

Devido à escassez de tempo, esse exercício não foi efetivado pelo grupo. Porém, a idéia de controle proporcional foi implementada mais à frente, num exercício sobre percorrer uma determinada distância em linha reta.

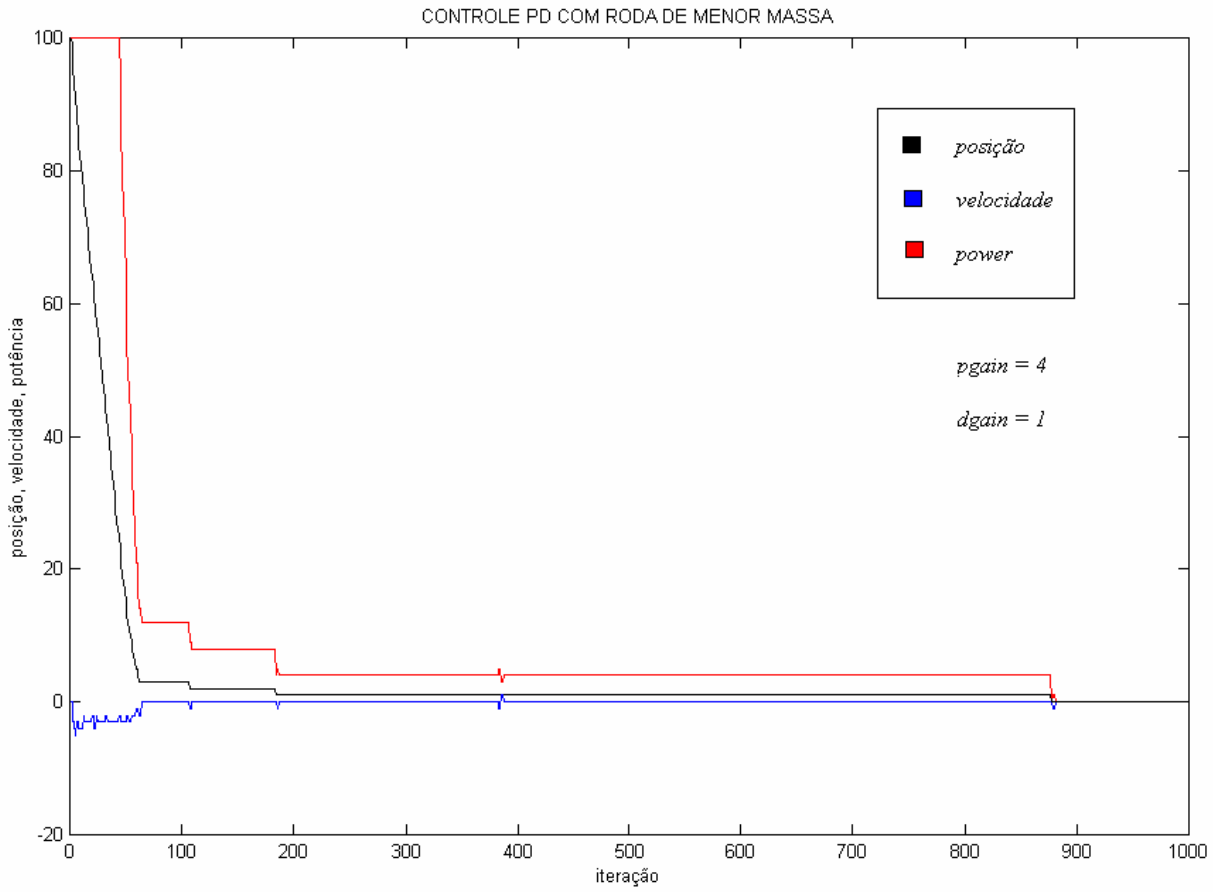
3. Controle PD:

Exercício 1:

Construímos um protótipo de testes idêntico ao da figura 5.9 do livro. Montamos dois sensores break-beam em posições no eixo da roda que nos permitissem fazer o controle PD da posição da roda baseado em algoritmos de shaft encoder de quadratura. Ou seja, usando dois break-beams posicionados estrategicamente, conseguimos obter o sentido de rotação da roda.

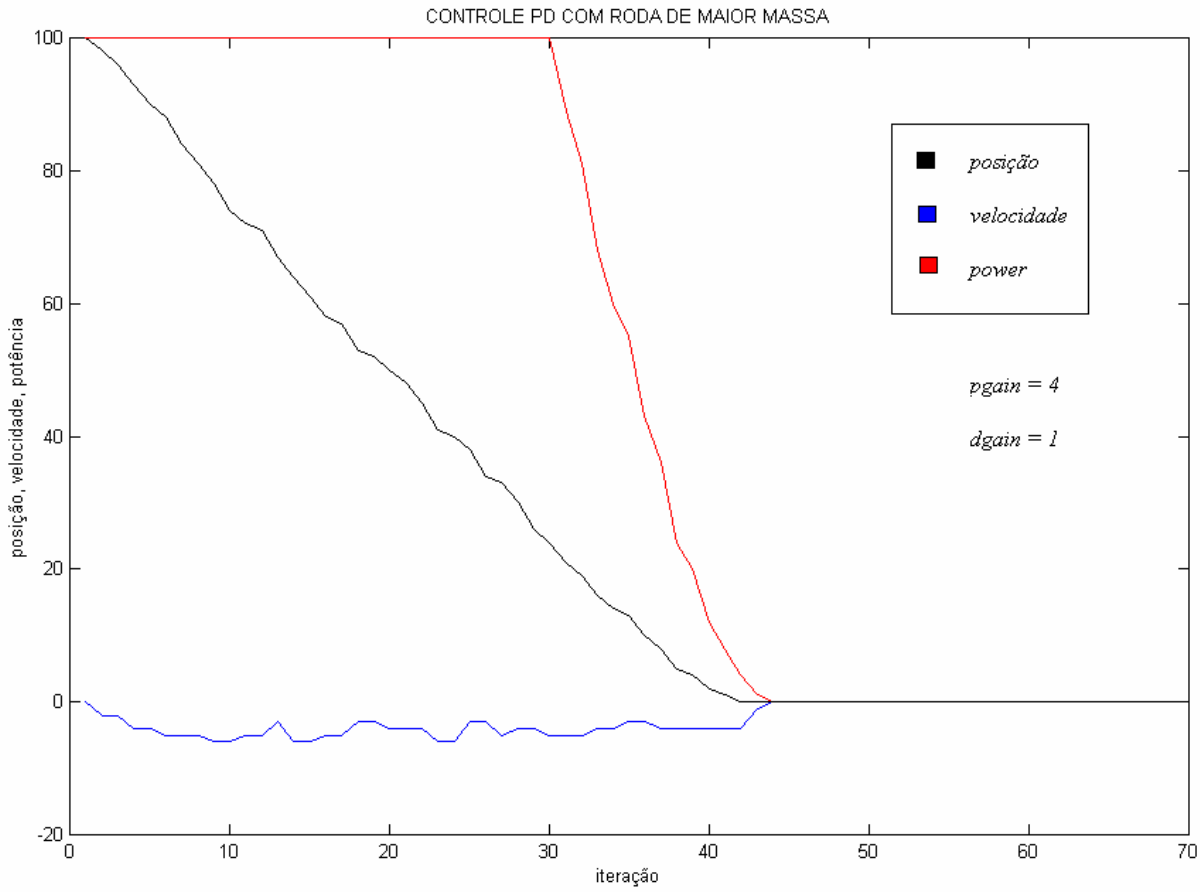
O programa que utilizamos para fazer o controle PD se encontra na página do grupo. Para fazer a leitura dos sensores break-beam, usamos uma rotina que roda diretamente no processador, chamada quaden10.icb. Encontramos o código dessa rotina na Internet. E, diga-se de passagem, foi muito difícil encontrar esse arquivo. Tínhamos em mãos um arquivo em assembly, porém o compilador para gerar um arquivo no formato .icb é bem mais difícil de se encontrar.

Para obter um bom resultado do controle PD, os ganhos proporcional e derivativo foram determinados por experimentação. Por coincidência, ou talvez pelo fato de nosso protótipo de testes ser muito idêntico ao usado como exemplo no livro, os ganhos que deram melhor resultado foram os mesmos do exemplo do livro, ou seja, *pgain* = 4 e *dgain* = 1. A curva plotada com os dados obtidos da posição, velocidade e potência do motor em função do tempo (medido em unidades de iteração), para o teste com uma roda de menor massa é mostrado a seguir:



Analisando o gráfico, percebemos o bom funcionamento da implementação. O fato da roda ter demorado tanto a atingir a posição zero nesse caso, permanecendo numa mesma posição (em torno da posição 4) por um longo período, se deve ao fato do motor utilizado não responder bem à baixa potência. Fora esse aspecto, o controle PD se mostrou muito eficiente.

Ao substituirmos a roda anterior por uma de maior massa, o resultado que obtivemos foi o seguinte:



Analisando o gráfico, percebemos claramente que a resposta do sistema nesse caso foi muito melhor do que à anterior. Usando o mesmo ganho utilizado no experimento anterior, a resposta do sistema utilizando-se uma roda de maior massa foi mais rápida, atingindo a posição zero em um menor tempo. Concluímos que isso se deve ao fato da maior inércia da roda de maior massa, que acaba por reduzir os efeitos da péssima resposta do motor à baixa potência.

Exercício 4:

Equipamos o nosso robô com um encoder simples em cada roda, com o objetivo de fazê-lo deslocar-se por uma dada distância em linha reta. O ideal seria que usássemos encoders de quadratura, porém não foi possível adquirir um, e nem foi possível implementar um usando dois sensores simples, por questão de tempo.

Outro problema encontrado foi que, como não nos foi fornecido tais sensores extras, nós acabamos por comprar dois, mas muito em cima da hora. Ao testarmos, percebemos que os sensores não funcionavam bem, e o arquivo binário que se encarregaria de coletar os dados do encoder não funcionou para os nossos sensores. Com isso, tivemos de implementar um programa em IC (disponível na página do grupo) para realizar a coleta de dados dos encoders, e ainda arcar com o prejuízo de não ter como calcular a velocidade como derivada da posição.

Por tudo isso, e mais o fato de que nossos motores são muito diferentes um do outro (para andar em linha reta, o motor mais forte tem de estar a 65% da potência total enquanto o motor fraco está a 100%, ale´m de responderem muito diferentemente à baixas potências), optamos por uma estratégia de emergência: usar só um encoder em uma das rodas (pois não tínhamos como controlar a velocidade), inferir a potência do motor dessa roda usando apenas um controle proporcional (pois, da mesma forma, não tínhamos a velocidade), e com base nessa potência, calcular a potência no outro motor pra que o robô siga em linha reta. Obviamente, não funcionou, pois a relação de rotação dos motores também varia com a potência!

Logo depois, em novos testes, percebemos que tínhamos cometido um engano a respeito da leitura dos sensores, descobrindo que eles funcionavam muito bem! Então, refizemos nosso algoritmo (também presente na página do grupo) usando as rotinas binárias para tratamento das leituras dos encoders, chamadas sencdr10.bin e sencdr11.bin, sendo cada uma designada para sua respectiva porta digital. Com isso, foi bem mais fácil fazer o robô andar em linha reta usando-se encoders e controle proporcional. O único problema que permaneceu foi uma ligeira curva que o robô fazia ao aproximar-se do destino final, pois como já mencionado, os motores respondem muito diferentemente em baixas potências.

Com isso, concluímos que a estratégia de emergência foi um fiasco, e que sem encoders de qualidade e com motores totalmente diferentes um do outro, tentar fazer o robô seguir em linha reta é uma tarefa impraticável. Porém, quando temos bons encoders, andar em linha reta não é mais um problema, porém os motores ruins ainda prejudicam um pouco a tarefa se andar por uma determinada distância.

4. Controle deliberativo:

Aqui, nossa tarefa é implementar o algoritmo Wavefront para deslocamento do robô. Usando controle deliberativo, ou seja, fornecido ao robô um mapa de sua posição atual, localização de obstáculos, e posição final, esse algoritmo calcula uma rota livre que o robô deve seguir. O mapa fornecido é discretizado, sendo o tamanho dos grids dependente do nível de detalhes desejado.

Para deslocar-se, nosso robô usa basicamente dois tipos de movimento: rotação de 90° e translação de 30cm (tamanho do grid; dimensão do piso paviflex). Esses movimentos de rotação e translação foram calibrados no próprio ambiente de apresentação, para minimizar os erros. A calibração foi feita utilizando-se tempo, ao invés dos encoders, pois no momento nos parecia a alternativa mais confiável. Também não utilizamos controle realimentado, pois os resultados obtidos anteriormente no exercício de deslocar-se em linha reta por uma determinada distância nos desencorajaram a continuar utilizando tal técnica de controle.

Os resultados obtidos foram muito satisfatórios, e conseguimos uma boa calibração de rotação e translação do robô. O algoritmo se mostrou bastante funcional durante todo o tempo em várias circunstâncias diferentes que testamos.

Conclusão:

O controle PD é realmente muito eficaz. Quando trabalhamos com sensores break beam de boa qualidade, e com isso podermos usar os códigos binários para leitura e tratamento dos dados desses sensores, trabalhar com nossa base de testes foi bem agradável e interessante, visto que podíamos verificar a resposta do sistema para várias combinações de ganhos sob cargas diferentes. Sem dúvida, foi um dos momentos mais interessantes neste trabalho.

Os testes para seguir parede também foram muito interessantes, apresentados resultados muito bons. Podemos verificar claramente a resposta do robô para os diferentes algoritmos implementados. Os experimentos nessa parte do trabalho só não foram melhores devido aos aspectos de construção do robô, que impediram que nosso "bend-sensor" pudesse abranger uma maior faixa de distância do robô à parede. Mas, mesmo com essa limitação, no geral a prática foi muito proveitosa.

Na tarefa de percorrer uma determinada distância, o uso do controle proporcional não se mostrou muito eficiente no nosso caso, que possuímos motores bem diferentes. Ao executar o algoritmo Wavefront usando temporização e sem nenhum controle, percebemos que, pelo menos pra pequenas distâncias, um deslocamento baseado em tempo, e não em algum tipo de controle, pode ser bem mais eficiente.

O controle é um aspecto importantíssimo na robótica, tornando as pequenas ações, como deslocamento e manter a velocidade constante, muito precisas e confiáveis. Dessa forma, podemos trabalhar num nível mais

complexo com grande tranqüilidade. Porém, são necessários bons equipamentos e muito trabalho, acompanhado de muitos testes, para que o controle seja realmente de qualidade.