Resolução dos Exercícios

Item 1 – Seção 3.6.5

Exercício 1

Neste exercício, observamos dois aspectos relacionados ao resistor R_{Load}. Caso o seu valor seja muito pequeno comparado com o de $47k\Omega$, a corrente de coletor seria mais elevada, porém a queda de tensão em R_{Load} seria menor devido à sua baixa resistência. Caso seu valor seja grande, a queda de tensão será grande, mas a corrente que circulará será pequena.

Para este exercício, vamos considerar $V_{CEsat} = 0$ (tensão de saturação coletor-emissor) e o ganho do transistor β.

Sabendo que:

$i_E = \beta \times i_B$ (corrente do emissor é igual ao ganho do transistor multiplicado pela corrente da base) – (1)

$i_E = i_B + i_C$ (corrente do emissor é a soma das correntes da base e do coletor) – (2)

Teremos que:

$\beta \times i_B = i_B + i_C$ - (3)

$i_C = i_B(\beta - 1)$ - (4)

A resistência equivalente R_T será:

$$\frac{1}{R_T} = \frac{1}{47k\Omega} + \frac{1}{R_{Load}} \Rightarrow R_T = \frac{47k\Omega \times R_{Load}}{47k\Omega + R_{Load}} \text{ - (5)}$$

Para ajustarmos R_{Load} adequadamente queremos que a queda de tensão nos resistores seja 5V quando i_B possuir seu valor máximo.

$$V = \frac{47k\Omega \times R_{Load}}{47k\Omega + R_{Load}} \times i_{B\max}(\beta - 1) = 5V \text{ - (6)}$$

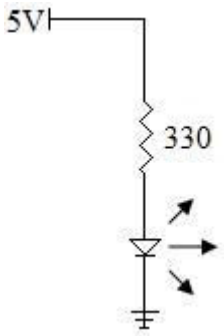
Rearranjando os termos em (6) colocando R_{Load} em evidência:

$$R_{Load} = \frac{5V \times 47k\Omega}{[47k\Omega \times i_{B\max}(\beta - 1) - 5V]} \text{ - (7)}$$

Este deve ser o valor de R_{Load} mais adequado.

Exercício 2

Item a)

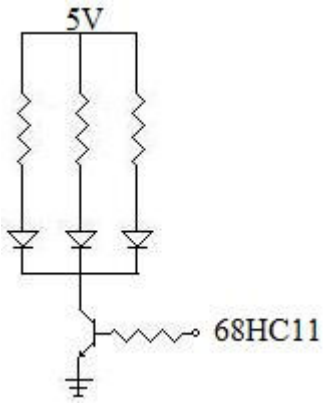


Fazendo a lei de Kirchoff da malha:

$$5V - i \times 330\Omega - 2V = 0 \Rightarrow i = \frac{3V}{330\Omega} = 9mA$$

Observamos que em geral, uma boa corrente para LED's é 20mA.

Item b)



Vamos considerar o limite de 100mA de corrente para o transistor e um consumo de 9mA por conjunto de LED. Além disso, vamos considerar V_{CEsat} igual a 0 (transistor ideal). Assim, podemos observar que até 11 LED's podem ser conectados.

Item 2 – Seção 5.1.3

Exercício 1)

Implementamos o algoritmo de "three-state" para fazer o robô seguir a parede. Infelizmente não tínhamos disponível um dispositivo para fazer a aquisição de dados da trajetória do robô, porém por inspeção visual, foi possível perceber que a trajetória foi muito parecida com a do gráfico da figura 5.8 do livro.

Exercício 2)

O robô funciona melhor com o algoritmo "gentle-turn" tanto ao andar em linha reta quanto em esquinas. Em linha reta, o robô anda com menos oscilações, visto que ao invés de virar bruscamente em direção a parede e muitas vezes chegar muito perto dela, ele anda em direção a ela mais suavemente. Ao chegar em uma esquina, ele também funciona melhor com o "gentle-turn". No algoritmo "hard-turn", assim que o sensor chega na parede, o robô começa a virar bruscamente com uma roda parada e outra com potência máxima. Isso acaba fazendo com que o robô colida com a parede, o que o impede de fazê-lo continuar em sua trajetória. Já no "gentle-turn" o robô ao virar continua se movendo também para a frente, fato que ajuda que o robô consiga voltar a detectar a parede na maioria das vezes e continuar a sua trajetória.

Exercício 3)

O algoritmo "three-state" funciona melhor que o "gentle-turn" para fazer o robô virar uma esquina. No primeiro algoritmo, o robô tem um comportamento mais suave que no segundo, fato que acarreta uma melhor percepção da parede quando esta está se aproximando. Com isso, utilizando o algoritmo "three-state" o robô conseguiu fazer mais curvas que o outro e o contorno foi mais suave, ou seja, quando voltou a detectar a parede, o robô estava em uma posição mais paralela a esta do que com o "gentle-turn".

Exercício 4)

Neste exercício nos foi pedido a implementação de um controlador proporcional para melhorar o desempenho do robô seguindo a parede. Ou seja, devemos mudar a direção do robô com uma potência proporcional à diferença (erro) entre a referência (distância que ele deveria estar da parede) e a distância medida. O resultado percebido na trajetória foi que o robô teve oscilações de menor amplitude do que quando sem o ganho proporcional.

Item 3 – Seção 5.2.3

Exercício 1)

Para controlar o sistema, utilizamos como variável controlada a diferença entre o número de pulsos contados no shaft enconder de cada roda e como variável manipulada a tensão aplicada em um dos motores. Ou seja, em um dos motores a tensão é constante e no outro a tensão é variada de forma a manter a velocidade de rotação a mesma em ambas as rodas. Dessa forma, a referência é a velocidade medida pelo encoder na roda cujo motor é movido com tensão constante e o erro é a diferença entre as velocidades medidas em cada roda. Assim, o sistema de controle vai atuar na variável manipulada de forma a manter a trajetória do robô em linha reta. Quanto ao controle, utilizamos um controlador PD (proporcional-derivativo). O termo proporcional multiplica o erro por um ganho e o termo derivativo obtém a derivada do erro, ou seja, a variação que ele está tendo. Se o erro estiver diminuindo, o termo derivativo terá uma saída negativa e se estiver aumentando, terá sua saída positiva. Com isso, ele diminui a atuação do controlador na variável manipulada quando o erro vai chegando em zero e aumenta quando vai se afastando. A saída do controlador será a soma dos dois termos.

Os ganhos encontrados para os controladores proporcional e derivativo foram, respectivamente, 10 e 0.05. O processo utilizado para obtê-los foi tentativa e erro. Fizemos vários testes e esses valores de ganho foi o que nos pareceu mais adequado para fazer o que foi proposto.

Exercício 2)

Os ganhos não continuaram ideais quando adicionamos massa ao robô. Como aumentamos a inércia do sistema, elevamos um pouco o ganho derivativo (de 0.05 para 0.15) para que o sistema voltasse a ter uma resposta aceitável. O valor do termo proporcional não foi alterado.

Exercício 3)

Podemos citar como sistemas que devem ser controlados com controladores PD: controle de posição de tornos, controle de temperatura de fornos e controle de posição de braços robóticos. Isso é justificado pelo fato destes sistemas não admitirem overshoot em suas repostas, e uma das características de controladores proporcionais-derivativos é não permitir este tipo de comportamento. O termo proporcional multiplica o erro por um ganho e o termo derivativo obtém a derivada do erro. Se o erro estiver diminuindo, o termo derivativo terá uma saída negativa e se estiver aumentando, terá sua saída positiva. Com isso, ele diminui a atuação do controlador na variável manipulada quando o erro vai chegando em zero e aumenta quando vai se afastando. Ou seja, o termo derivativo atenua a atuação do controlador a medida que a variável controlada chega perto da referência, funcionando neste caso como um “atrito” no sistema.

Exercício 4)

Foi implementado no robô uma função que utiliza um controle PD para fazê-lo seguir uma distância determinada em linha reta. Esta distância pode ser regulada através do knob da Handyboard e pode variar de 20cm a 45cm. Uma rotina nesta função faz a transformação da distância escolhida para o número de rotações que o shaft enconder deve contar. Dois problemas foram encontrados neste procedimento. O primeiro foi que o robô quando vai começar a percorrer a distância, dá uma leve guinada. Acreditamos que este fato se deve pelo fato de o comando de ligar os motores não seja pontual, ou seja, primeiro a handyboard liga um e depois o outro. O segundo problema foi um erro sistemático de aproximadamente 5cm a menos na distância escolhida. Isto provavelmente se deve a erro de medida no raio da roda, visto que esta se deforma, e erro na contagem feita pelo shaft enconder.

Item 4 - Algoritmo Wavefront

O algoritmo Wavefront utiliza uma representação do mundo real onde há apenas duas possibilidades, espaço vazio ou obstáculo. Ele, a partir das coordenadas iniciais do robô e das coordenadas do alvo, tem o objetivo de ir de um ponto ao outro percorrendo o menor caminho possível e evitando todos os obstáculos. Para isso, o mundo real é modelado como uma matriz de números inteiros, deixando números reservados para representar os obstáculos e o alvo. Os outros elementos da matriz são preenchidos na varredura lógica inicial feita pelo robô.

Com o algoritmo implementado e com as coordenadas inicial e alvo dadas, o robô faz inicialmente uma varredura lógica do mapa (matriz de n x m de números inteiros), na qual coloca os obstáculos com valor igual a MAX (um valor suficientemente grande pré-definido) e o ponto alvo com valor igual a 1. Em seguida, preenche as outras células (elementos da matriz) do mapa de forma que as células adjacentes à célula-alvo recebam o valor 2, as adjacentes às de valor 2 recebam o valor 3 e assim por diante. Isto é feito até que se chegue na posição ocupada pelo robô, que ganha o valor mais alto. Dessa forma, para encontrar um caminho, basta limitar o robô a se mover apenas para células que tem valor menor ou igual ao valor da célula que ele está naquele momento e proibí-lo de passar por células de valor igual a MAX, que são obstáculos. É opção do programador deixar que o robô se mova na diagonal ou não. No nosso caso, consideramos vizinhança 4x4 de forma que as diagonais não são consideradas.