Universidade Federal de Campina Grande

Centro de Engenharia Elétrica e Informática

Curso de Graduação em Engenharia Elétrica

RAJIV ALBINO TORREÃO MOTA

PROJETO E IMPLEMENTAÇÃO DE PLATAFORMA EXPERIMENTAL PARA O LABORATÓRIO DE CONTROLE DIGITAL BASEADA EM ARDUINO

Campina Grande, Paraíba
Dezembro de 2015

RAJIV ALBINO TORREÃO MOTA

PROJETO E IMPLEMENTAÇÃO DE PLATAFORMA EXPERIMENTAL PARA O LABORATÓRIO DE CONTROLE DIGITAL BASEADA EM ARDUINO

Trabalho de Conclusão de Curso submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Controle e Automação

Orientador:
Professor João Batista Morais dos Santos, D. Sc.

Campina Grande, Paraíba
Dezembro de 2015

RAJIV ALBINO TORREÃO MOTA

PROJETO E IMPLEMENTAÇÃO DE PLATAFORMA EXPERIMENTAL PARA O LABORATÓRIO DE CONTROLE DIGITAL BASEADA EM ARDUINO

Trabalho de Conclusão de Curso submetido à Unidade Acadêmica de Engenharia Elétrica da Universidade Federal de Campina Grande como parte dos requisitos necessários para a obtenção do grau de Bacharel em Ciências no Domínio da Engenharia Elétrica.

Área de Concentração: Controle e Automação

Aprovado em / /

Professor Avaliador
Universidade Federal de Campina Grande
Avaliador

Professor João Batista Morais dos Santos, D. Sc.
Universidade Federal de Campina Grande
Orientador, UFCG

Dedico este trabalho aos meus pais.

AGRADECIMENTOS

Agradeço primeiramente aos meus pais, Carlos Augusto e Maria do Socorro, por terem me proporcionado boa educação.

Agradeço à minha família pelo constante apoio às minhas aventuras ao longo do curso.

Agradeço à minha namorada Aline pelo carinho, companheirismo e amor.

Agradeço aos meus amigos de curso por terem feito do quiosque da universidade uma referência em bem estar e improdutividade.

Agradeço ao professor João Batista pela paciência e orientação neste trabalho.

"A simplicidade é a máxima sofisticação."

Leonardo da Vinci.

RESUMO

Este trabalho propõe um experimento prático para o Laboratório de Controle Digital. Visando melhorar a compreensão dos alunos a respeito das diferenças entre controladores *on/off*, proporcional e proporcional-derivativo, é proposto que o aluno projete-os para fazer um robô seguir uma linha preta em um fundo branco. Atrelando melhorias expressivas no desempenho do robô com monitoramento de variáveis de interesse do sistema, o aluno poderá enxergar o que cada parte do controlador proporciona para o funcionamento do sistema como um todo.

Palavras-chave: controle digital, controladores, seguidor de linha, experimento.

ABSTRACT

This paper proposes a practical experiment to the Digital Control Laboratory. To improve students' understanding about the differences between on/off, proportional and proportional-derivative controllers, it is proposed that the student design them to make a robot follow a black line on a white background. Mixing significant improvements in performance and monitoring system state variables, the student can see what each part of the controller provides to the operation of the whole system.

Keywords: digital control, controllers, line follower, experiment.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

CC	Corrente Contínua
P	Proporcional
PD	Proporcional-Derivativo
PWM	*Pulse Width Modulation* (Modulação em Largura de Pulso)
SISO	*Single-Input, Single-Output* (Entrada Única, Saída Única)

SUMÁRIO

1 INTRODUÇÃO

Nos dias atuais os educadores vêm buscando alternativas ao processo tradicional de ensino centrado em memorização e aplicação de conceitos e equações para resolução de exercícios. De acordo com (Moratori, 2003), atividades lúdicas aliam o lazer ao desafio, estimulando o interesse dos alunos em solucionar os problemas propostos, mesmo em nível de ensino superior.

No laboratório, o aluno põe em prática o assunto estudado nas aulas de teoria. Atualmente, os experimentos do Laboratório de Controle Digital da UFCG são feitos em simulações em computador que, mesmo muito importantes para o desenvolvimento do aluno, não ajudam a concretizar tão bem o aprendizado quanto um experimento prático o faria, onde o aluno precisa lidar com problemas que vão além do software e recebe como recompensa ver em funcionamento o sistema que ele projetou, tornando o aprendizado uma atividade empolgante.

O experimento proposto por este trabalho é fornecer ao aluno um sistema completo - parte mecânica, sensores, atuadores, microcontrolador, o modelo matemático e o esqueleto do código a ser completado -, para que o aluno seja capaz de aplicar os conceitos de controle no projeto dos controladores e observar o comportamento do sistema de acordo com a mudança da estratégia de controle e constantes. A Figura 1 mostra o robô montado:

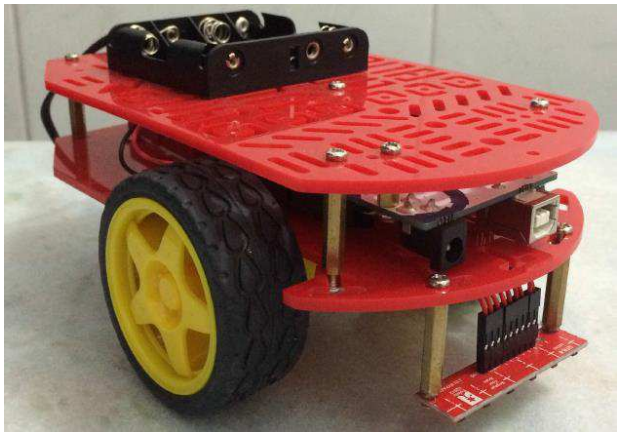

Figura 1. Robô completo.

1.1 OBJETIVOS

O objetivo a ser alcançado pelo aluno é projetar o controlador para que o robô siga uma linha preta em um terreno branco, inicialmente com um controle simples que apenas posicione o robô sobre a linha, chamado de Controlador On/Off, onde será possível ver que o carro completa o percurso, porém em baixa velocidade e com muita ondulação em torno da linha, às vezes até perdendo-a. Após completar esta primeira parte, o aluno projetará um controlador proporcional, ou Controlador P, que já mostrará uma melhoria expressiva de velocidade. Na terceira etapa, o aluno projetará um controlador proporcional-derivativo, ou simplesmente Controlador PD, que melhorará a performance do robô na execução da tarefa, vendo-o atingir velocidades ainda maiores e com ondulações drasticamente menores, devido ao fato deste controlador ser sensível a variações bruscas graças ao termo derivativo (Franklin, Powell e Emami-Naeini, 2006), que o capacita a rastrear bem a linha mesmo em curvas fechadas.

1.2 ESTRUTURAÇÃO

No Capítulo 2, estão listados os componentes de hardware e no Apêndice B encontra-se o esquemático das conexões, facilitando assim futuras manutenções pelo professor ou pelos monitores da disciplina de Laboratório de Controle Digital.

O software completo foi detalhado no Capítulo 3.

Para permitir o cálculo dos ganhos, o sistema foi modelado e será fornecido ao aluno, facilitando a ligação entre o mundo físico e o modelo matemático e ensinando-o a encontrar funções de transferência de plantas com entradas e saídas incomuns como é o caso. A entrada do sistema é a tensão diferencial entre os motores e a saída é o ângulo de orientação do robô em relação à linha. Mais detalhes sobre a modelagem serão apresentadas no Capítulo 4.

Os controladores são melhor apresentados ao longo do Capítulo 5.

2 HARDWARE

O hardware é composto por uma plataforma robótica com dois motores acoplados a caixas de redução, um microcontrolador, um sensor de refletância, um circuito de ponte H e baterias.

2.1 PLATAFORMA ROBÓTICA E MOTORES

A plataforma robótica utilizada foi o Magician Chassis, do fabricante Dagu Robot. Foi escolhido por ser uma plataforma completa com rodas, motores e caixas de redução fabricadas sob medida. Com o auxílio do guia de montagem o responsável pelo experimento poderá facilmente efetuar pequenos reparos nos circuitos internos à plataforma. O modelo pode ser visto na Figura 2.

Figura 2. Chassi.

O robô contém ainda duas rodas tracionadas na parte dianteira e uma esfera metálica na parte traseira para sustentação. Acoplado a cada roda há um motor *DG01D*, de tensão nominal 6V CC, com uma caixa de redução 48:1. É preciso dois motores para que o robô faça curvas, controlando a velocidade de cada motor separadamente. Na Figura 3, vemos como os motores estão acoplados às rodas.

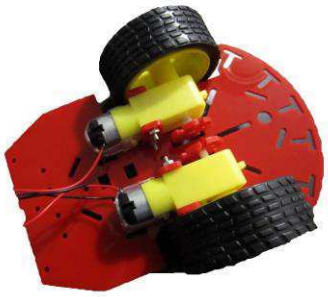

Figura 3. Motores e rodas.

2.2 SENSOR DE REFLETÂNCIA

Para o reconhecimento da posição da linha foi explorado a diferença de reflexão de raios luminosos entre as cores preto e branco. A cor branca reflete os raios luminosos que a atingem, já a cor preta absorve-os, (Costa e Okamoto Jr, 2011). Baseado neste comportamento, os sensores de refletância da família QTR da fabricante Pololu emitem raios infravermelhos em direção à superfície e medem a quantidade refletida. Caso haja muita reflexão, a superfície é considerada branca, caso contrário é considerada preta. Dispondo de seis pares LED-fototransistor, mostrados na Figura 4, é possível determinar com boa precisão a posição da linha abaixo do robô. O modelo escolhido para a aplicação foi o QTR-8RC, que utiliza um circuito de descarga de capacitores que fornece ao microncontrolador esta leitura de forma digital, mesmo sendo uma grandeza analógica, medindo o tempo que a tensão de saída leva para decair devido ao fototransistor. Esta forma de leitura dispensa o uso de conversores A/D, acelerando o procedimento e economizando recursos do sistema.

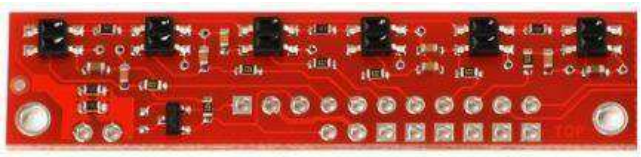

Figura 4. Sensor de refletância.

2.3 MICROCONTROLADOR

O microcontrolador utilizado é o ATmega328, presente na placa Arduino Uno Rev3, bastante conhecido no meio acadêmico. Apesar de o Arduino não ser explicitamente apresentado aos alunos de graduação em Engenharia Elétrica da UFCG

durante o curso, muitos têm contato com ele em projetos de iniciação científica e/ou pessoais, além de ser uma plataforma de fácil aprendizagem para os estudantes. Este controlador é programável em linguagem C++, que é abordada no início do curso. A programação do Arduino será o meio de interação do aluno com o experimento. A Figura 5 mostra o modelo utilizado.

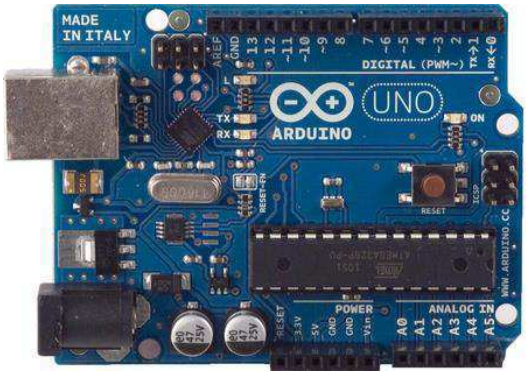

Figura 5. Arduino Uno.

Com base nas leituras do sensor de refletância, o microcontrolador calcula a velocidade de cada motor de acordo com a estratégia de controle e controla-os através de dois sinais PWM.

2.4 CIRCUITO DE POTÊNCIA

De acordo com o *datasheet* fornecido pelo fabricante, a corrente de saída máxima por pino do Arduino UNO é 40 mA. Portanto, para fornecer a corrente requerida para a operação é necessário o uso de um circuito eletrônico de potência entre o controlador e os motores. Como sugerido em (Warren, Adams e Molle, 2011), o circuito utilizado foi uma Ponte H, esboçado na Figura 6.

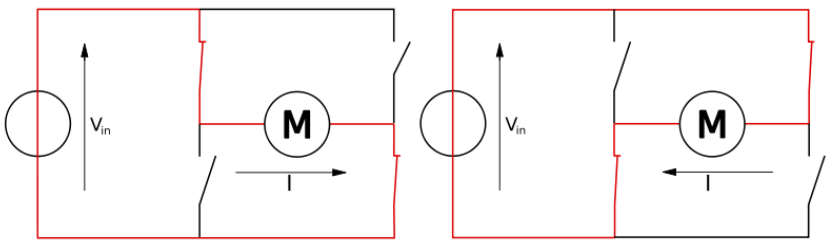

Figura 6. Esboço do funcionamento da Ponte H.

Para esta implementação foi usado o circuito integrado L293D que contém duas pontes H duplas, bem adequado ao caso pois há dois motores a serem acionados.

2.5 FONTES DE ENERGIA

A alimentação do sistema é feita com quatro pilhas AA de 1,5V, totalizando 6V para os motores, e uma bateria de 9V para o controlador.

Os componentes foram montados no chassi de forma a deixar a porta USB do controlador acessível para constantes reprogramações e as baterias colocadas na parte de cima, facilitando a troca, como mostrado na Figura 1.

Na Tabela 1 estão dispostos os componentes de hardware necessários para a reprodução deste experimento, junto ao custo de cada um deles em valores atuais (2015):

Tabela 1. Preço dos componentes.

Componente	**Preço**
Plataforma Magician Chassis	R$ 45,00
Arduino UNO rev3	R$ 60,00
Sensor QTR-8RC	R$ 28,00
CI L293D	R$ 10,00
Fios e conectores	R$ 6,00
Pilhas e Bateria	R$ 15,00
Total	R$ 164,00

3 SOFTWARE

Neste capítulo será tratado o procedimento executado pelo software do projeto. Pode-se dividir o algoritmo em quatro etapas: leitura do sensor, processamento da orientação, determinação da velocidade dos motores e atuador. Após este ciclo, o sistema recomeça o processo após as alterações no ambiente (linha).

O fluxograma da Figura 7 resume o procedimento:

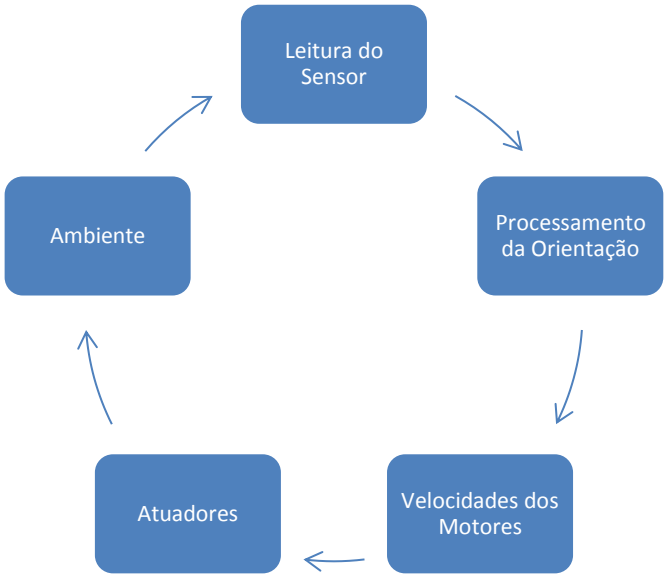

Figura 7. Fluxograma do software.

3.1 LEITURA DO SENSOR

A leitura do sensor é realizada pela função *readLine()*, presente na biblioteca *QTRSensors.h* fornecida pela fabricante do sensor. A função ativa os emissores infravermelhos e retorna uma estimativa da posição da linha.

3.2 PROCESSAMENTO DA ORIENTAÇÃO

Conforme a Equação (1), a estimativa é feita usando uma média ponderada do índice dos sensores multiplicados por 1000. Dessa forma, o valor 0 indica que a linha

está logo abaixo do sensor 0, o valor 1000 indica que a linha está logo abaixo do sensor 1, da mesma forma para os demais sensores, se estiver entre os sensores 0 e 1 a função retorna 500. Além disso, caso a linha tenha sido perdida (todos os sensores indiquem a cor branca) esta função continua retornando o valor anterior, o que é ideal para o controle, pois caso o sensor 5 seja o da direita e o robô se posicione completamente à esquerda da linha, esta função continuará a retornar 5000. Em suma, a equação é a seguinte:

$$L = \frac{0 * v_0 + 1000 * v_1 + 2000 * v_2 + \cdots}{v_0 + v_1 + v_2 + \cdots}, \quad (1)$$

em que L é a estimativa da posição da linha e v_i é o valor lido pelo sensor correspondente ao seu índice.

Isto representa uma medida da orientação do robô em relação à linha, com os valores dos ângulos mapeados entre 0 e 5000. Portanto, para que o robô permaneça no centro do percurso (diferença de orientação zero), sob os sensores 2 e 3, a referência deve ser 2500.

3.3 CÁLCULO DA VELOCIDADE DOS MOTORES

A ideia básica para executar curvas nesse robô é alterar a velocidade das rodas separadamente. Quando a velocidade da roda direita for maior do que a esquerda o robô desviará para a esquerda, caso contrário desviará para a direita.

Neste caso, a velocidade do motor é um valor entre *0* e *255* que indica o ciclo de trabalho (*duty cycle*) do PWM que controla o motor.

A determinação destas velocidades depende da estratégia de controle utilizada e será abordada no Capítulo 4.

3.4 ATUADORES

Os atuadores do sistema são os motores acoplados a cada roda. As velocidades calculadas no item anterior são um valor entre 0 e 255 que determinam o *duty cycle* do PWM que controla os motores. O sinal de PWM sai dos pinos 5 (motor direito) e 6 (motor esquerdo) e é aplicado na entrada da Ponte H.

4 MODELAGEM

Primeiramente é preciso definir as variáveis de entrada e de saída do sistema, após isso encontrar equações que as relacionem e finalmente determinar a função de transferência.

4.1 VARIÁVEIS DE ENTRADA E SAÍDA

Analisando o funcionamento do robô vemos que o meio de atuação do controlador é aplicar tensões nos dois motores. Os pinos apresentam apenas saídas digitais mas fornecem PWMs que permitem aplicar valores médios de tensão entre zero e cinco volts. Como o movimento do robô depende de ambos os valores, é plausível admitir como variável de entrada a Tensão Diferencial $\boldsymbol{v_e(t)}$ entre os dois motores, (O'Sullivan, 2013). Esta tensão é mostrada na Figura 8.

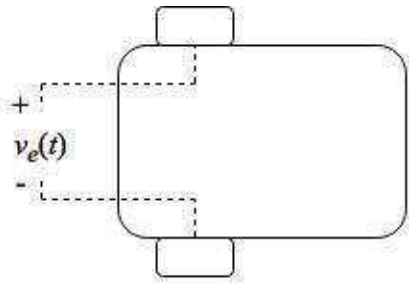

Figura 8. Tensão Diferencial.

O objetivo do controle é fazer com que o carro permaneça no centro da linha, para isso pode-se definir a variável *Orientação*, tanto para a linha quanto para o carro, e medir a diferença entre as orientações como variável de saída, objetivando reduzir esta diferença a cada ciclo do controle, (O'Sullivan, 2013). A Figura 9 ilustra este conceito, onde a variável *Diferença de Orientação* $\boldsymbol{\theta(t)}$ está explicitada:

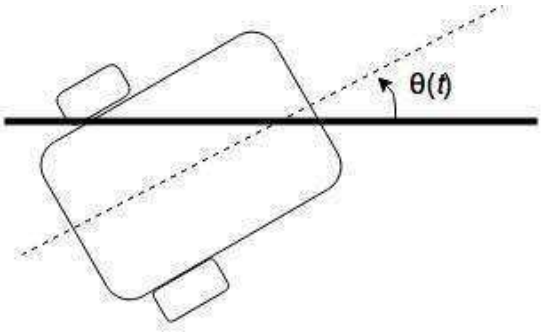

Figura 9. Orientação.

4.2 FUNÇÃO DE TRANSFERÊNCIA

Definidas as variáveis de entrada e saída, a função de transferência deve relacioná-las, como pode ser visto na Figura 10:

Figura 10. Bloco Simplificado da Planta.

Agora é preciso encontrar equações que as relacionem. O torque pode ser definido de duas formas distintas, conforme as seguintes equações:

$$\tau = J\frac{d\omega(t)}{dt}, \tag{2}$$

$$\tau = K_m\left[v_e(t) - v_{fcem}(t)\right], \tag{3}$$

em que τ é o torque, J é o momento de inércia, ω é a velocidade angular, K_m é uma constante do motor, v_e é a tensão diferencial entre os motores e v_{fcem} é a força contraeletromotriz. Além disso, a força contraeletromotriz pode ser definida como:

$$v_{fcem}(\mathrm{t}) = K_b\omega(t), \tag{4}$$

onde K_b é outra constante do motor. Igualando as equações (2) e (3) e aplicando (4), obtém-se:

$$\frac{d\omega(t)}{dt} = \frac{K_m}{J} v_e(t) - \frac{K_m K_b}{J} \omega(t). \quad (5)$$

A velocidade angular é definida como:

$$\omega(t) = \frac{d\theta(t)}{dt}, \quad (6)$$

em que θ é orientação angular. Aplicando (6) em (5), tem-se:

$$\frac{d^2\theta(t)}{dt^2} = \frac{K_m}{J} v_e(t) - \frac{K_m K_b}{J} \frac{d\theta(t)}{dt}. \quad (7)$$

Aplicando a Transformada de Laplace em (7) e rearranjando os termos, obtém-se a seguinte função de transferência:

$$\frac{\Theta(s)}{V_e(s)} = \frac{\frac{K_m}{J}}{s^2 + \frac{K_m K_b}{J} s}. \quad (8)$$

De acordo com (Parikh, Shah e Sheth, 2012), motores deste porte, potência e tensão apresentam aproximadamente as seguintes constantes dispostas na Tabela 2:

Tabela 2. Constantes da Função de Transferência.

Constante	**Valor**
K_m	0,1 N·m/V
K_b	1,5 V·s/rad
J	0,01 Kg·m^2

Finalmente, aplicando os valores na equação (8), a função de transferência aproximada do sistema é:

$$\frac{\Theta(s)}{V_e(s)} = \frac{10}{s^2 + 15s}. \quad (9)$$

5 CONTROLADORES

O controle em malha aberta é impraticável nesta situação, visto que o sistema precisa medir a posição da linha para segui-la.

Com o modelo da planta pode-se montar o diagrama de blocos em malha fechada do sistema, adicionando a realimentação e o controlador. A Figura 11 mostra o diagrama, desprezando a função de transferência do sensor:

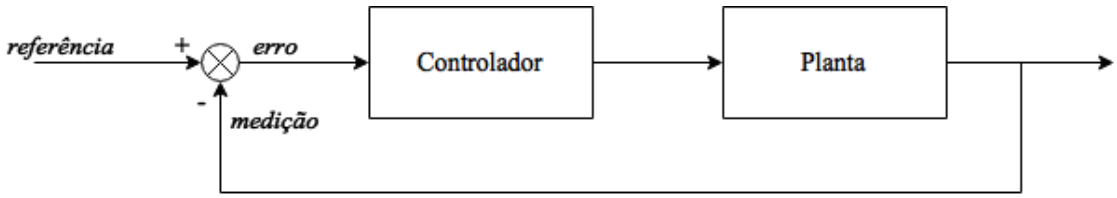

Figura 11. Diagrama de blocos.

5.1 CONTROLE ON/OFF

Esta é a estratégia mais simples utilizada no experimento. Tem-se apenas dois possíveis casos a serem tratados: se o valor da posição estiver acima da referência, mova o robô para trazê-lo para baixo; se o valor for abaixo da referência, leve-o para cima. Ou seja, se a linha estiver à direita do centro o robô é levado para a esquerda, desligando a roda esquerda e ligando a roda direita, tal como para o caso contrário.

A nível de implementação, este controlador é feito apenas com uma condição *if/else*. No cálculo do *erro*, subtraindo a medição da referência tem-se um resultado positivo quando a medição for maior que a referência e um resultado negativo, caso contrário. Usando a variável *erro* na condição, é determinado se o robô deve virar à esquerda ou à direita. A Figura 12 mostra o bloco do Controlador para este caso:

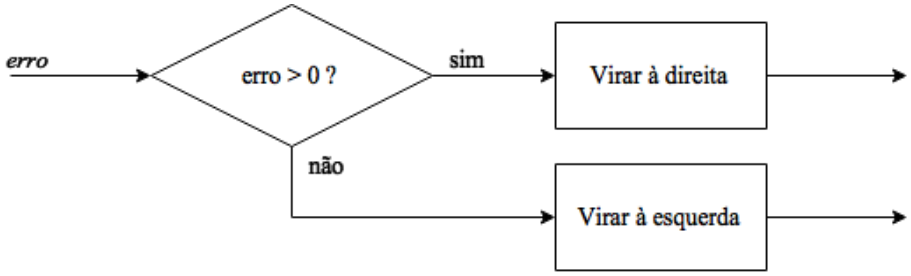

Figura 12. Controlador On/Off.

Como indicado na Figura 12, o *threshold* definido foi *0*, visto que *erro* maior que zero significa que o carro está à esquerda da linha, e menor que zero indica que o carro está à direita da linha.

Outro parâmetro relevante para a estratégia de controle é a velocidade máxima dos motores. Esta foi definida empiricamente baseada em sucessivos testes como sendo a maior velocidade com que o robô consegue completar o circuito sem perder a linha nenhuma vez. Para o controlador On/Off, a velocidade máxima encontrada foi *200* (como explicado anteriormente, este número representa o ciclo de trabalho do PWM, entre *0* e *255*).

5.2 CONTROLE PROPORCIONAL

Esta estratégia de controle já é mais sofisticada que a anterior. Primeiramente é preciso calcular a variável *erro*, diferença entre a posição medida e a referência. Após isso é calculada a variável *controle*, que é o *erro* multiplicado pela constante K_p do controlador. A velocidade do motor direito será uma velocidade base mais o controle e a velocidade do motor esquerdo será uma velocidade base menos o controle.

A Figura 13 ilustra o bloco do controlador:

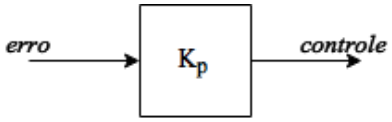

Figura 13. Controlador P.

O ganho do controlador foi calculado utilizando a ferramenta *pidtune()*do Matlab, que é explicada no Apêndice A, obtendo-se o valor $K_p = 15$.

A velocidade máxima dos motores para esta estratégia foi empiricamente fixada em *210*, e a velocidade base (que o controlador tende a manter) foi fixada em *180*.

5.3 CONTROLE PROPORCIONAL-DERIVATIVO

Esta estratégia de controle é semelhante à anterior, mas neste caso é adicionado o termo derivativo ao cálculo do controle. É preciso calcular a derivada da variável *erro* fazendo a subtração do erro atual pelo erro anterior. Agora o controle é calculado pela soma dos dois termos, o *erro* vezes o K_p e a *derivada do erro* vezes o K_d.

A Figura 14 mostra este controlador em diagrama de blocos:

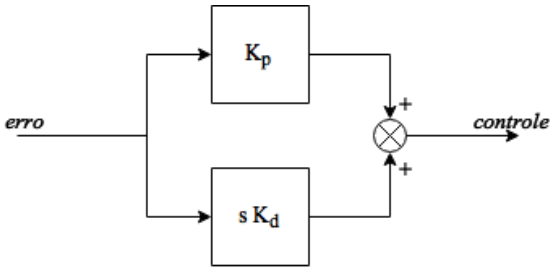

Figura 14. Controlador PD.

Os ganhos também foram calculado utilizando a ferramenta *pidtune()* do Matlab, resultando nos valores $K_p = 36,1$ e $K_d = 0,71$.

Por ser uma estratégia mais complexa que as anteriores, se alcançou velocidades maiores sem ocorrer perda da linha. A velocidade máxima foi fixada em *255* e a velocidade base foi fixada em *220*.

6 APRESENTAÇÃO DOS RESULTADOS

Nos tópicos seguintes será definido o bloco do controlador conforme a estratégia de controle, bem como o gráfico *erro* × *tempo*.

Os três controladores foram testados na mesma trilha, mostrada na Figura 15, que apresenta uma reta, uma curva suave, uma curva acentuada e ondulações.

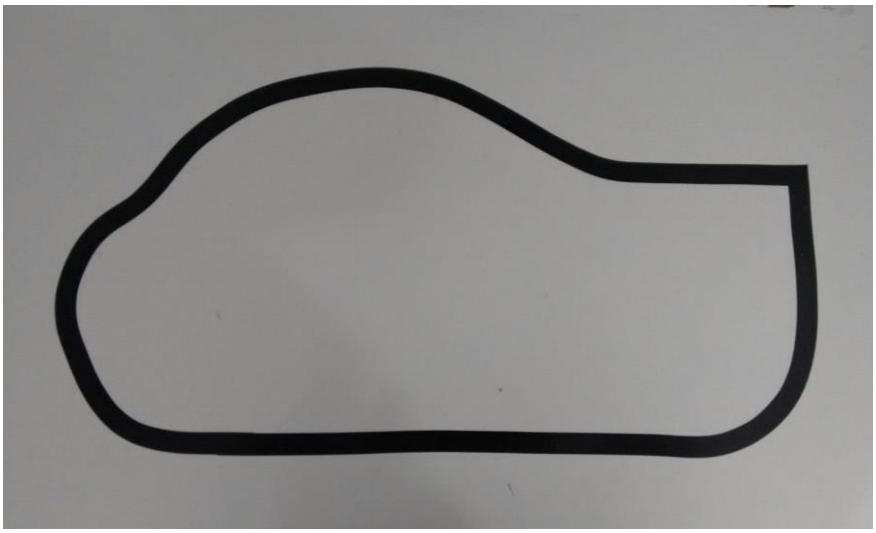

Figura 15. Trilha de testes.

Os testes consistiram em medir o tempo de uma volta, utilizando um contador de tempo interno com um *flag* de parada atribuído a um ponto de referência, e monitorar a variação do *erro*, medindo-a a cada três milisegundos através de interrupções, visando obter maior precisão nas medições. Os resultados estão dispostos nas subseções seguintes.

6.1 RESULTADOS DO CONTROLADOR ON/OFF

Com este tipo de controle, o robô completou uma volta no circuito em *9,294 segundos*. A Figura 16 mostra o gráfico *erro* × *tempo(amostra)*:

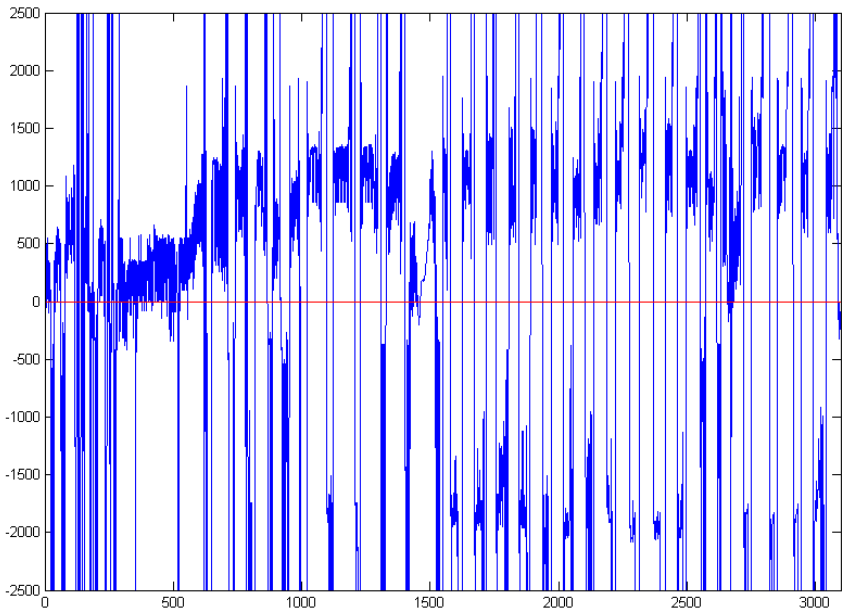

Figura 16. Monitoramento do erro no controlador On/Off.

6.2 RESULTADOS DO CONTROLADOR P

Com o ganho calculado no capítulo anterior, o tempo de volta foi de *4,506 segundos*. Novamente, a Figura 17 mostra o gráfico *erro* × *tempo(amostra)*:

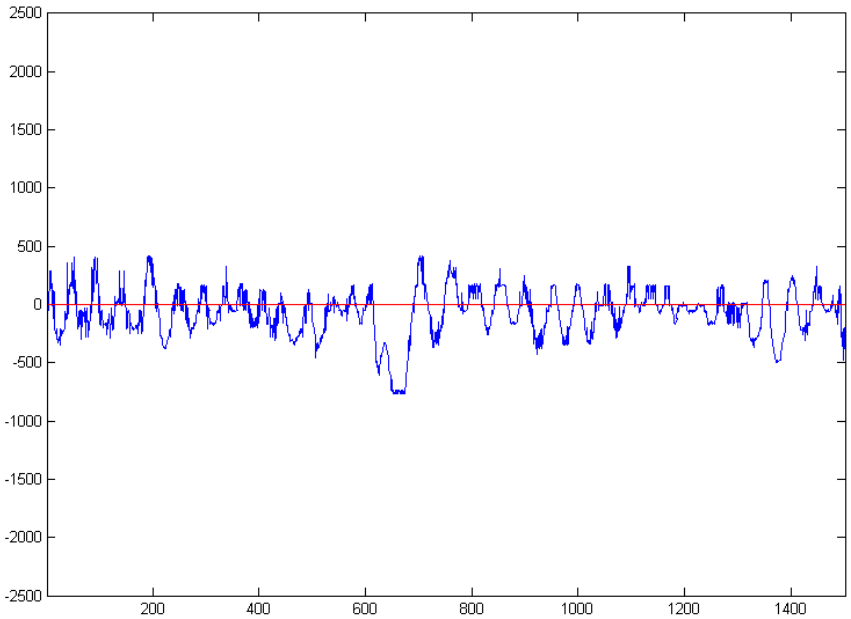

Figura 17. Monitoramento do erro no controlador P.

Como pode ser observado, há grandes diferenças em relação ao gráfico anterior, a oscilação é visivelmente menor e o tempo de volta é menor do que a metade.

6.3 RESULTADOS DO CONTROLADOR PD

Com os ganhos calculados no Capítulo 5, o tempo de volta foi de *3,393 segundos*. Mais uma vez, o gráfico *erro* × *tempo(amostra)* (Figura 18):

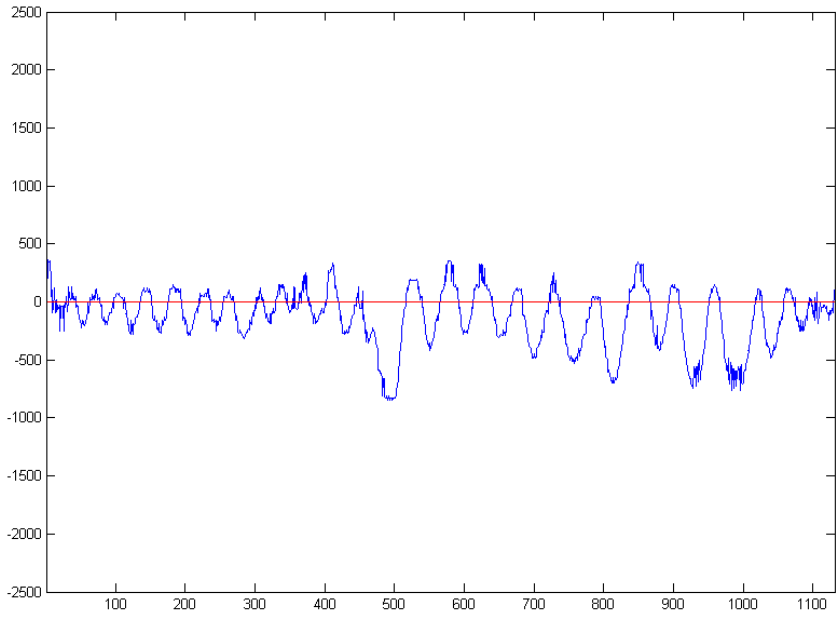

Figura 18. Monitoramento do erro no controlador PD.

Novamente houve uma melhora na performance com um tempo de volta menor e erro com picos menores, mais rapidamente corrigidos devido ao termo derivativo no controlador.

6.4 QUADRO COMPARATIVO

Visando comparar de uma forma mais direta as três estratégias de controle, a Tabela 3 exibe o cálculo dos valores médios do módulo e quadráticos das três curvas anteriormente apresentadas.

Tabela 3. Quadro comparativo entre os controladores.

Controle	**Tempo de Volta (s)**	**Média do |*erro*|**	**Média do *erro²***
On/Off	9,294	1391,4	2678200
P	4,506	156,6032	45358
PD	3,393	208,1459	77594

7 CONCLUSÃO

Ficou evidenciado que o trabalho atingiu as expectativas, visto que há uma melhora expressiva no desempenho do protótipo de acordo com a estratégia de controle utilizada, portanto, para um aluno ficará mais intuitivo entender a necessidade de aplicar as técnicas de projeto de controladores para obter sistemas mais robustos e otimizados.

Além disso, os gráficos do controlador P indicam medidas de erro menores, em relação ao controlador PD, embora o robô complete o circuito em mais tempo. Isto acontece devido ao fator derivativo fornecer uma rápida recuperação ante a distúrbios (neste caso, curvas), permitindo que o carro atinga velocidades maiores.

Diversas disciplinas foram importantes para a realização deste trabalho. Claramente, Controle Analógico e Controle Digital forneceram a base teórica aplicada no projeto de controladores. Eletrônica, Eletrônica de Potência, Arquitetura de Sistemas Digitais, Processamento Digital de Sinais, Instrumentação Eletrônica e Sistemas de Aquisição de Dados e Interface foram muito importantes pois ensinaram a trabalhar com circuitos, sensores, atuadores e microcontroladores. Introdução e Técnicas de Programação foram importantes para apresentar, logo no início do curso, as bases da programação.

BIBLIOGRAFIA

COSTA, A. H. R.; OKAMOTO JR, J. Interação de robô no ambiente, UNICAMP, 2011.

FRANKLIN, G. F.; POWELL, J. D.; EMAMI-NAEINI, A. **Feedback control of dynamics systems**. [S.l.]: Prentice Hall Inc, 2006.

MORATORI, P. B. **Por que utilizar jogos educativos no processo de ensino aprendizagem?** UFRJ. Rio de Janeiro. 2003.

NATH, A. S.; KUMAR, A.; MALIK, T. Implementation of PID control to reduce wobbling in a line following robot. **International Journal of Research in Engineering and Technology**, Kancheepuram, v. 02, n. 10, p. 531-535, October 2013.

OGATA, K. **Engenharia de Controle Moderno**. 5a. ed. [S.l.]: Pearson Prentice Hall, 2011.

O'SULLIVAN, R. Line Following Robot. **Manufacturing Design Engineering**, 2013. Disponivel em: <https://manufacturingdesignengineer.wordpress.com/>. Acesso em: 11 jun. 2015.

PARIKH, P. A.; SHAH, H. B.; SHETH, S. **A Mechatronics Design of a Line Tracker Robot Using Ziegler-Nichols Control Technique For P, PI and PID Controllers**. G. H. Patel College of Engineering & Technology. Gujarat. 2012.

WARREN, J.-D.; ADAMS, J.; MOLLE, H. **Arduino for Robotics**. [S.l.]: Apress, 2011.

APÊNDICE A – FUNÇÃO *PIDTUNE( )* DO MATLAB

C = pidtune(*sys*, *type*) projeta um controlador PID do tipo *type* para a planta *sys*. Se *type* especifica um controlador PID de um grau de liberdade, então o controlador é projetado para uma malha de realimentação unitária, como ilustrado:

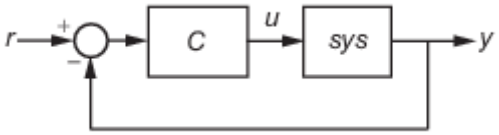

Figura 19. Sistema com realimentação unitária.

Pidtune sintoniza os parâmetros do controlador *C* para balancear desempenho (tempo de resposta) e robustez (margens de estabilidade).

Os argumentos de entrada e saída são descritos a seguir:

i. *sys*: modelo dinâmico de sistema SISO (Entrada Única, Saída Única, do inglês *Single-Input, Single-Output*) da planta para o projeto do controlador;

ii. *type*: tipo do controle para o projeto do controlador, especificado como string. O termo "tipo do controlador" refere-se a quais termos estão presentes na ação de controle. Por exemplo, um controlador PD tem apenas os termos proporcional e derivativo, enquanto um controlador PIDF contém os termos proporcional integral, derivativo e derivativo filtrado. A palavra do *type* pode ser uma das seguintes: 'P', 'I', 'PI', 'PD', 'PDF', 'PID', 'PIDF';

iii. *C*: controlador projetado para *sys*. Se *sys* é um conjunto de modelos lineares, *pidtune* projeta um controlador para cada modelo linear e retorna um conjunto de controladores PID.

Os objetivos da sintonia de controladores PID incluem:

- Estabilidade em malha fechada: o sistema em malha fechada permanece limitado para entradas limitadas;

- Desempenho: o sistema em malha fechada segue mudanças de referência e atenua distúrbios o mais rápido possível. Quanto maior a largura de banda da malha, mais rápido o controlador responde a mudanças na referência ou distúrbios na malha;
- Robustez: o projeto da malha tem margem de ganho e margem de fase suficiente para permitir erros de modelagem e variações na dinâmica do sistema.

O algoritmo da MathWorks para sintonia de controladores PID segue esses preceitos sintonizando os ganhos do PID para alcançar um bom balanceamento entre performance e robustez. Por padrão, o algoritmo escolhe uma frequência de *crossover* baseado na dinâmica da planta, e projeta para um ganho de margem de 60º. Quando é mudado o tempo de resposta, largura de banda, resposta transitória ou margem de fase usando o *PID Tuner Interface*, o algoritmo computa novos ganhos.

Para uma dada robustez (mínima margem de fase), o algoritmo de sintonia escolhe um projeto de controlador que balanceia as duas medidas de desempenho, acompanhamento da referência e rejeição de distúrbios. É possível mudar o foco do projeto para favorecer uma dessas duas medidas de performance. Para isto, é preciso usar a opção *DesignFocus* do *pidtune* na linha de comando.

Quando o foco do projeto é mudado, o algoritmo procura ajustar os ganhos para favorecer o foco escolhido, enquanto mantém a mínima margem de fase. Quanto mais parâmetros sintonizáveis há no sistema, mais chances tem o controlador de alcançar o foco desejado sem sacrificar robustez.

APÊNDICE B – ESQUEMÁTICO DE CONEXÕES

Na Figura 20 estão mostradas as conexões entre os componentes do sistema necessárias para o funcionamento do código:

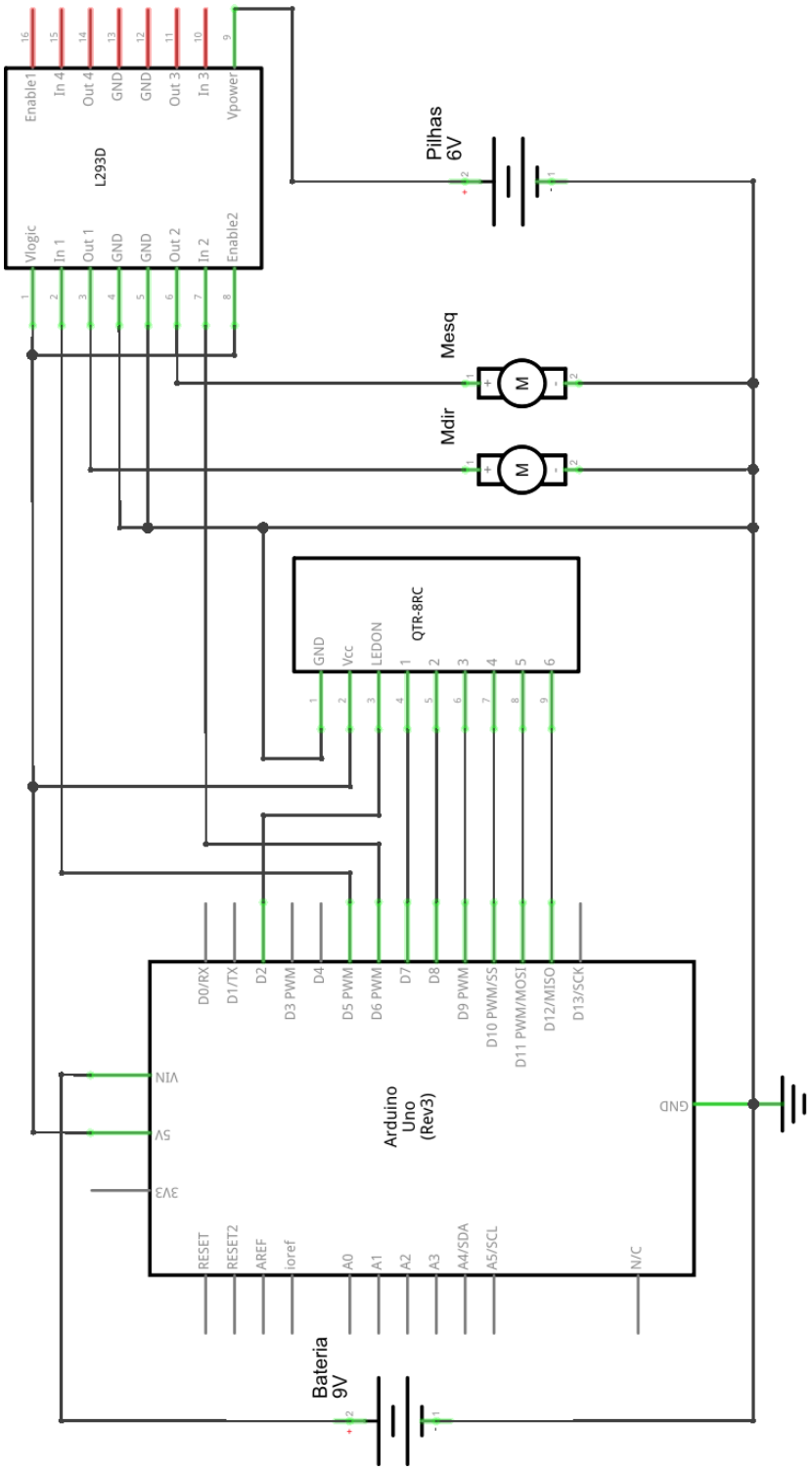

Figura 20. Esquemático.

APÊNDICE C – CÓDIGO DO CONTROLADOR ON/OFF

```
#include <QTRSensors.h>

#define NUM_SENSORES 6
#define MOT_DIREITO  5
#define MOT_ESQUERDO 6
#define VEL_MAXIMA   200
#define REFERENCIA   2500

// Instancia de um objeto para o sensor
QTRSensorRC qtr8rc((unsigned char[]){14,15,16,17,18,19}, NUM_SENSORES);

// Funcao de inicializacao, chamada uma vez no inicio da execucao
void setup()
{
  for (int i=0; i<100; i++)
  {
    // Calibracao dos sensores
    qtr8rc.calibrate();
    delay(20);
  }

  // Pino 3 sempre em HIGH para funcionar como pino Vcc
  pinMode(3,OUTPUT);
  digitalWrite(3,HIGH);

  // Delay para posicionar o carro
  delay(1000);

}

// Funcao de loop infinito
void loop()
{
  // Leitura dos sensores
  unsigned int sensores[NUM_SENSORES];
  int posicao = qtr8rc.readLine(sensores);

  // Calculo do erro
  int erro = posicao - REFERENCIA;

  // Controle
```

```
  int velocDireito = VEL_MAXIMA;
  int velocEsquerdo = VEL_MAXIMA;

  if (erro > 0)
    velocDireito = 0;
  else
    velocEsquerdo = 0;

  analogWrite(MOT_DIREITO, velocDireito);
  analogWrite(MOT_ESQUERDO, velocEsquerdo);
  delay(3);
}
```

APÊNDICE D – CÓDIGO DOS CONTROLADORES P E PD

```
#include <QTRSensors.h>

#define Kp 15.0
#define Kd 0.0
//#define Kp 36.1
//#define Kd 0.71

#define VEL_BASE     180
#define VEL_MAXIMA   210
//#define VEL_BASE     220
//#define VEL_MAXIMA   255

#define NUM_SENSORES 6
#define MOT_DIREITO  5
#define MOT_ESQUERDO 6
#define REFERENCIA   2500

// Instancia de um objeto para o sensor
QTRSensorRC qtr8rc((unsigned char[]){14,15,16,17,18,19}, NUM_SENSORES);

int erroAnterior = 0;

// Funcao de inicializacao, chamada uma vez no inicio da execucao
void setup()
{
  for (int i=0; i<100; i++)
  {
    // Calibracao dos sensores
    qtr8rc.calibrate();
    delay(20);
  }

  // Pino 3 sempre em HIGH para funcionar como pino Vcc
  pinMode(3,OUTPUT);
  digitalWrite(3,HIGH);

  // Delay para posicionar o carro
  delay(1000);
```

```
}

// Funcao de loop infinito
void loop()
{
  // Leitura dos sensores
  unsigned int sensores[NUM_SENSORES];
  int posicao = qtr8rc.readLine(sensores);

  // Calculo do erro
  int erro = posicao - REFERENCIA;
  int deriv = erro - erroAnterior;
  erroAnterior = erro;

  // Controle
  int controle = erro*Kp + deriv*Kd;

  int velocDireito = VEL_BASE + controle;
  int velocEsquerdo = VEL_BASE - controle;

  // Controle de limites
  if (velocDireito > VEL_MAXIMA)
    velocDireito = VEL_MAXIMA;

  if (velocEsquerdo > VEL_MAXIMA)
    velocEsquerdo = VEL_MAXIMA;

  if (velocDireito < 0)
    velocDireito = 0;

  if (velocEsquerdo < 0)
    velocEsquerdo = 0;

  analogWrite(MOT_DIREITO, velocDireito);
  analogWrite(MOT_ESQUERDO, velocEsquerdo);
  delay(3);
}
```