
**UNIVERSIDADE ESTADUAL DO SUDOESTE DA BAHIA - UESB**

**DEPARTAMENTO DE CIÊNCIAS EXATAS**

# ROBÔ SEGUIDOR DE LINHA AUTÔNOMO UTILIZANDO O CONTROLADOR PROPORCIONAL-DERIVATIVO EM UMA PLATAFORMA DE *HARDWARE* / *SOFTWARE* LIVRE

**ANDRIQUE FIGUEIRÊDO AMORIM**

**VITÓRIA DA CONQUISTA - BAHIA**

**2011**


**UNIVERSIDADE ESTADUAL DO SUDOESTE DA BAHIA - UESB**

**DEPARTAMENTO DE CIÊNCIAS EXATAS**

# ROBÔ SEGUIDOR DE LINHA AUTÔNOMO UTILIZANDO O CONTROLADOR PROPORCIONAL-DERIVATIVO EM UMA PLATAFORMA DE *HARDWARE* / *SOFTWARE* LIVRE

**ANDRIQUE FIGUEIRÊDO AMORIM**

Monografia apresentada ao Departamento de Ciências Exatas da Universidade Estadual do Sudoeste da Bahia como parte dos requisitos para a conclusão do curso de Ciência da Computação.

Orientador: Prof. DSc. Roque Mendes Trindade

Co-orientador: Prof. MSc. Camilo Carvalho

**VITÓRIA DA CONQUISTA - BAHIA**

**2011**

*Aos meus pais, Yvone e Wilde,*
*incansáveis incentivadores da minha*
*formação profissional, exemplo de*
*luta e determinação*

*Dedico!*

## AGRADECIMENTOS

À ***Deus*** pela vida e por todas as conquistas!

Aos meus filhos ***David*** e ***Andressa*** e**,** minha esposa ***Bárbara*** pelo amor, paciência e incentivo.

Ao meu irmão ***Wlimar*** pelo companheirismo.

À toda minha ***família*** pelo carinho.

.Ao professor **Roque** pela orientação e contribuição intelectual.

Ao professor **Camilo** pela co-orientação, confiança e apoio.

## LISTA DE TABELAS

## LISTA DE FIGURAS

SUMÁRIO

## RESUMO

Cada vez mais os robôs estão fazendo parte do cotidiano das pessoas, seja em pequenos dispositivos eletromecânicos, seja em tarefas industriais funcionando 24 horas por dia continuamente sem sentir cansaço, stress ou outras limitações dos seres humanas. O custo para a instalação de mecanismos robotizados tornou-se bem mais acessível possibilitando a inserção de robôs na educação, no meio empresarial e até na vida doméstica. Entretanto, na realidade, não há nenhum robô capaz de funcionar perfeitamente e ainda não cometer erros.

Este projeto implementou um controlador proporcional-derivativo (PD) em um robô móvel autônomo construído dentro de uma plataforma de software e hardware livre (*open source*), que melhora a performance de execução do robô reduzindo erros sobre um caminho feito com uma fita preta em uma superfície branca e lisa. O robô foi capaz de seguir a linha preta de forma eficaz deslocando-se ao longo do caminho suavemente sem nenhum problema.

## OBJETIVO GERAL

O objetivo principal deste projeto é desenvolver um robô seguidor de linha autônomo, com um controlador proporcional-derivativo (PD) que otimize o trajeto do robô sobre uma linha preta reduzindo erros no percurso, construído em uma plataforma de hardware e de software livre (*open source*).

## OBJETIVOS ESPECÍFICOS

- Compreender a teoria e a aplicação de ações de controles industriais do tipo proporcional, integral e derivativo (PID) em robôs de pequeno porte;
- Aplicar a tecnologia PWM (modulação por largura de pulso) no controle de potências de motores DC;
- Apresentar uma solução eficaz de fácil implementação para robôs seguidores de linha;
- Difundir o microcontrolador Arduino como plataforma de hardware Open Source para o uso de aplicações em robótica.

## JUSTIFICATIVA

Robôs seguidores são máquinas móveis programadas para detectar e seguir uma linha. Esse caminho pode ser uma linha preta sobre uma superfície branca ou o inverso de forma que o robô realize movimentos.

## INTRODUÇÃO

Há muito tempo o homem vem idealizando reproduzir a si próprio, por meios mecânicos, criando máquinas capazes de executar e suprir com propriedade as tarefas humanas. Grandes pensadores historicamente conhecidos como Leonardo da Vinci e Nicola Tesla costumavam dedicar grande parte do tempo em estudos direcionados à projeção da construção de mecanismos apropriados para copiar ou simular algumas características da capacidade humana. Pode-se dizer que estes inventos já continham embrionariamente a idéia da Robótica. Mas é no princípio do século XX, com a Revolução Industrial, que a Robótica aparece com maior força, devido à necessidade de intensificar a produção e melhorar a qualidade dos produtos. Surgem, neste período, os primeiros robôs com aplicações industriais, repetindo infinitamente e com precisão milimétrica, uma série de operações previamente programadas (ZILLI, 2004; SILVA, 2009; BURLAMAQUI, 2010).

Os termos Robótica e robô estão intimamente associados. Enquanto o primeiro refere-se à área do conhecimento, o segundo diz respeito ao produto desta ciência. Robô vem do tcheco “robota” e significa “trabalho forçado”. Foi usado pela primeira vez em 1921 por Karel Capek em seu romance Rossum’s Universal Robots”. Já o termo Robótica foi criado em 1948 pelo escritor de Ficção Cientifica, Isaac Asimov, em seu romance "I, Robot" (Eu, Robô). Os robôs de Capek eram máquinas de trabalho incansáveis, de aspecto humano, com capacidades avançadas mesmo para os robôs atuais (ZILLI, 2004).

A Robótica é uma área de estudo multidisciplinar que se apoia nos conhecimentos de mecânica, eletrônica, física, matemática, biologia, informática e inteligência artificial. Tem por objetivo “automatizar tarefas através de técnicas de programação e algoritmos orientados à construção de robôs” (SILVA, 2009).

Nos dias atuais, a Robótica se configura numa área de crescente desenvolvimento. Isto decorre dos inúmeros recursos que os sistemas de

microcomputadores vem oferecendo. Grandes centros de tecnologia se ocupam incansavelmente da criação de novas tecnologias, criando robôs capazes de lidar com tomadas de decisão cada vez mais complexas.

## MICROCONTROLADORES

São dispositivos de tamanho reduzido, capazes de realizar controle de máquinas e equipamentos eletro-eletrônicos através de programas. São dispositivos que reúnem, em um único circuito integrado, diversos componentes de um sistema computacional simplificado. Em outras palavras, podemos afirmar que um microcontrolador é um pequeno microcomputador integrado em um único chip. Por se tratar de um componente programável, é bem versátil, podendo ser empregado em aplicações das mais diversas. (CORTELETTI, 2006) .

Os microcontroladores representam um grande avanço da tecnologia e atualmente estão fazendo parte da vida cotidiana das pessoas, sendo encontrados na maioria dos equipamentos eletrônicos modernos, como: celulares, televisões, DVDs, fornos de microondas, aparelhos de CD, câmaras digitais, filmadoras etc. Esta aplicabilidade dos microcontroladores se deve ao fato dele ser mais versátil e funcional, já que, num minúsculo chip, dispõe de todos os elementos necessários para fazer uma máquina funcionar (ZELENOVSKY e MENDONÇA, 2010).

A versatilidade, funcionalidade e aplicabilidade dos microcontroladores, o tornam categoricamente diferente dos microprocessadores. Enquanto que, para ser usado, o microprocessador necessita da adição de outros componentes, como memória e periféricos, o microcontrolador não requer componentes externos. Isto agrega ao microcontrolador uma vantagem importante, pois, à medida que todos os seus componentes estão contidos no mesmo encapsulamento, a sua dimensão passa a ser infinitamente menor, conforme as Figuras 1 e 2, exigindo, para o seu funcionamento, um consumo de energia também menor, o que reduz o custo de sua produção. Por este

motivo, os microcontroladores se tornaram bem mais utilizados para projetos de sistemas embarcados.

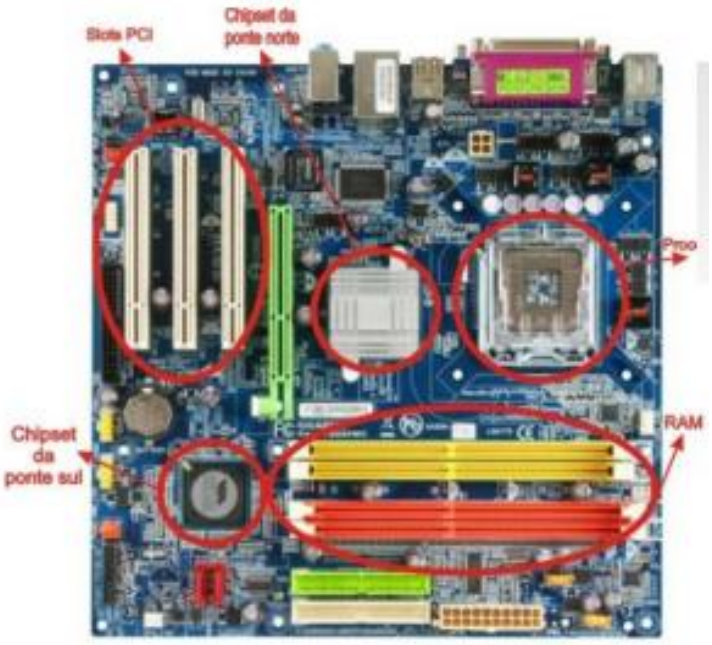


Figura 1. Placa mãe de um PC

Figura 2. Microcontrolador

## PLATAFORMA ARDUINO

A Plataforma Arduino teve sua origem em Ivrea, na Itália, em 2005. O seu surgimento ocorreu quando um grupo de acadêmicos da área de computação, formado por Massimo Banzi, Tom Igoe, Gianluca Martino e David Mellis, insatisfeitos com a complexidade com que os pacotes de computação

física, oferecidos no mercado, se apresentavam, desenvolveram um novo projeto, com plataforma de *software* e *hardware* mais simples e com um nível menor de exigência de conhecimentos de engenharia da computação. Este projeto obteve sucesso no âmbito acadêmico, atraindo a atenção do engenheiro espanhol, David Cuartielles detentor de grande experiência em microcontroladores, que se interessou pelo desenvolvimento da plataforma, em caráter comercial (ARDUINO, 2010).

O propósito desta equipe foi o de criar um ambiente de programação e uma placa padrão que fosse capaz de suportar um microcontrolador, com acesso ilimitado e livre, possibilitando o uso e manuseio tanto do hardware como do software. A plataforma criada foi validada na própria universidade, sendo prontamente aprovada pelos estudantes que, mesmo não possuindo conhecimentos profundos em computação física e robótica, puderam executar projetos mais complexos (ARDUINO, 2010).

Arduino é uma plataforma livre e flexível, fácil de usar tanto o *software* quanto o *hardware* e, por isso, pode ser usada por aqueles que se interessam em criar objetos ou ambientes interativos, apresenta algumas características que a distingue das demais. A mais importante diz respeito à concepção *open-source* que atribui a ela duas importantes peculiaridades: a primeira é de possibilitar a versatilidade e a segunda, é de ser colaborativa. Isto porque sendo *open-source* ela disponibiliza não só a documentação do projeto original da plataforma, mas também as placas e códigos de aplicação desenvolvidos a partir dela, através da comunidade virtual criada para este fim, o www.arduino.cc. Atualmente este sítio se constitui como a grande referência acerca do Arduino para todo o mundo (Figura 3).

Outra característica refere-se à placa mãe. Esta é composta por um microcontrolador (cérebro), desenvolvido pela empresa Atmel. Qualquer pessoa com algum conhecimento em eletrônica pode adquirir este cérebro por um preço accessível nas lojas especializadas ou na grande rede mundial ou construir sua própria placa sem pagar por direitos autorais (ARDUINO, 2010).

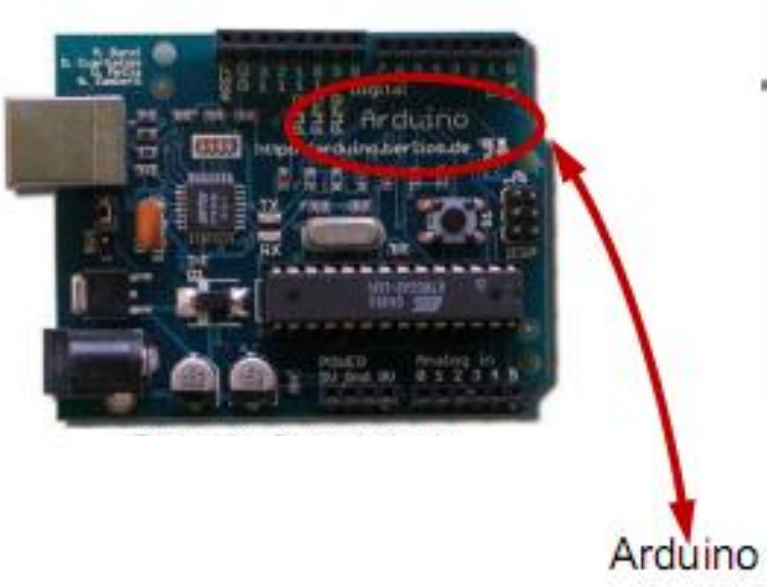


Figura 3. Placa Arduino

Fonte: ROBOTSHOP (2010)

O ambiente e a linguagem de programação do Arduino também merecem destaque. A linguagem facilita a interação com os periféricos, tais como: sensores de luz, de som, ultrassom, infravermelho e diversos tipos de motores. E o ambiente faz menção ao seu IDE (*Integrated Development Environment* ou Ambiente Integrado de Desenvolvimento) que é implementado na linguagem Java, funcionando em vários sistemas operacionais, tais como: Windows, Linux e MacOS (ARDUINO, 2010).

Arduino é uma plataforma de *hardware open source*, baseada em um microcontrolador da empresa Atmel com suporte de conexão via serial ou USB com o computador para enviar ou receber dados e/ou programas. Significa um *hardware* genérico o suficiente para permitir a criatividade dos desenvolvedores em elaborar aplicações que permitam interação com o mundo real, ou seja, computação física, através de diversos sensores e atuadores. As ferramentas que dão suporte a programação de um microcontrolador são coletivamente denominada de *software* básico. Este é usado continuamente e, portanto, deve ter eficiência na execução. Desta forma, uma linguagem para tal aplicação provê uma execução rápida. Além disso, deve ter recursos de baixo nível que

permitam ao *software* fazer interface como os dispositivos externos como: atuadores e sensores. A linguagem C oferece todas essas ferramentas, uma vez que possibilita o acesso a dispositivos de *hardware*. Sua execução é eficiente e não sobrecarrega o usuário com muitas restrições de linguagem. A plataforma Arduino aproveita todas as funcionalidades desta linguagem e ainda acrescenta uma dezena de funções e métodos para facilitar a programação de dispositivos físicos.

## MOTOR DE CORRENTE CONTÍNUA

Motores elétricos convertem energia elétrica em mecânica. Existem motores de corrente contínua (CC) (Figura 4) e de corrente alternada (CA), cada um com diversas variações. Motores de corrente alternada são geralmente usados para máquinas grandes e recebem energia diretamente da rede de distribuição de energia. Segundo Jones *et al.* (1999), robôs móveis usam tipicamente corrente contínua, pois sua fonte de energia é uma bateria.

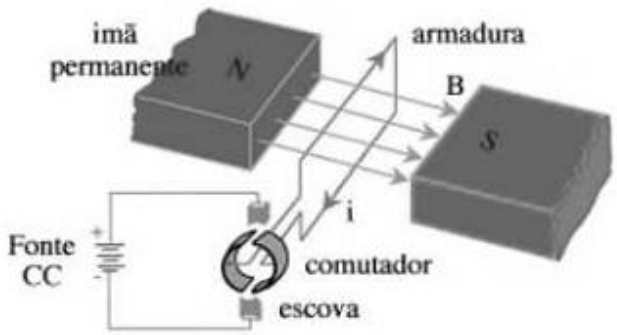


Figura 4: Estrutura de um motor de corrente contínua

Fonte: ANTUNES (2009)

Para as necessidades de locomocão de um robô o motor gira em uma velocidade muito alta e com um torque muito baixo. Para inverter essa relação o motor deve ser ligado a uma caixa de reducão que produz uma nova saída que gira mais devagar, porém com um torque maior. Muitos motores CC são

vendidos com a caixa de redução já instalada, como é o caso do adquirido para este projeto.

Os motores de corrente contínua tem pelo menos dois terminais que ao se aplicar uma tensão nestes terminais o motor gira em um sentido e invertendo a polaridade da entrada o motor gira no sentido contrário.

## SISTEMA DE CONTROLE

Um Sistema de Controle é um conjunto de dispositivos que mantém o valor de uma ou mais grandezas físicas dentro de um determinado limite, de acordo com o valor programado em sua entrada. Este sistema é composto de um processo a ser controlado, um atuador, que é o dispositivo capaz de alterar o valor físico da grandeza, um controlador, responsável pela parte “inteligente” do sistema, calculando quanta energia o atuador deverá entregar ao sistema para mantê-lo estável. Estes elementos podem ser eletro-eletrônicos, mecânicos e até mesmo o próprio ser humano. Por exemplo, no controle de velocidade de um automóvel, o atuador poderia ser considerado como o pé do motorista sobre o acelerador, o sensor seria composto pelos seus olhos observando o velocímetro e o controlador seu próprio cérebro. Se o ponto de controle fosse 60 km/h e a velocidade observada no velocímetro fosse 55 km/h, o cérebro processaria a informação vinda dos olhos e mandaria acionar o pé sobre o acelerador, aumentando a velocidade e quando esta estivesse mais próxima do valor programado seria dado o comando de diminuir a pressão sobre o acelerador até que a velocidade se estabilizasse próxima do ponto de controle.

O Sistema de Controle pode atuar sobre praticamente qualquer grandeza, desde que haja a necessidade de controlá-la, porém as mais comuns são: temperatura, pressão, vazão, nível de líqüidos ou sólidos, velocidade e posicionamento linear ou angular.

**Tipos de sistemas de controle**

Os Sistemas de Controle são classificados quanto à sua estrutura ou desenho como sendo em malha aberta ou malha fechada, e ainda quanto ao tratamento que as informações sofrerão como analógicos e digitais. O Sistema em Malha Aberta é composto pela entrada, controlador e atuador, sem a utilização de realimentação do sistema por meio de um sensor, ou seja, após a atuação do controle não é feita a verificação de como esta atuação influenciou o sistema, nem mesmo se o ponto de controle foi atingido, pois, sem um sensor em contato com o processo torna-se impossível esta determinação. A estrutura de um sistema em malha aberta (Figura 5) apresenta o seguinte diagrama:

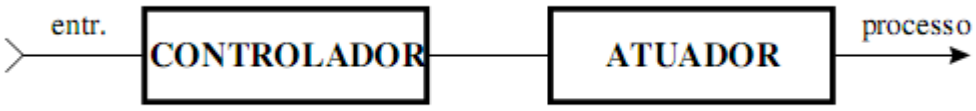


Figura 5: Sistema de malha aberta

Os Sistemas de malha fechada (Figura 6) apresentam a característica de utilizar um sensor como sinal de realimentação, ou seja, qualquer distúrbio no processo é “sentido” por este que envia um sinal compatível com o controlador que, por sua vez, pode levar esta situação em consideração na sua atuação. Nos Sistemas de Controle industriais, a maioria é feita em malha fechada.

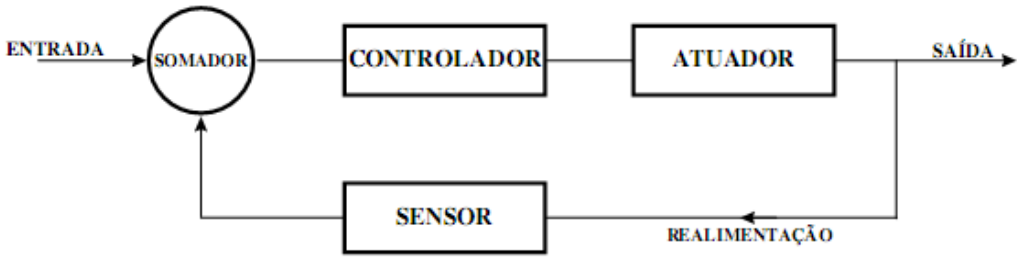


Figura 6: Sistema de malha fechada

Os sistemas analógicos ou digitais, dizem respeito à forma matemática como as variáveis são tratadas, internamente ao sistema, bem como aos componentes deste. Num sistema analógico, por exemplo, o controle é feito por um circuito composto por componentes elétricos e eletrônicos que tratam sinais elétricos proporcionais à grandeza e demais variáveis do sistema. Num sistema digital, o controle é feito por um programa, auxiliado por conversores do tipo analógico-digital e digital-analógico, que permitem transformar variações físicas, por natureza analógicas, em sinais compreensíveis pelo programa e seu resultado, por natureza digital, em sinais novamente compatíveis com as grandezas físicas, ou seja, as grandezas e variáveis externas ao sistema são analógicas e internamente são tratadas de forma digital, através dos conversores. A tecnologia digital é mais atual e traz algumas vantagens, pois, qualquer alteração no sistema pode ser feita apenas mudando um programa, dispensando a necessidade de mudanças em componentes e configurações físicas, bem como as demais vantagens de sistemas microprocessados.

## AÇOES DE CONTROLE

### *Controle liga-desliga:*

Também chamado *On-Off*. O controlador compara o sinal de entrada, que indica o valor atual da grandeza (PV ), com o valor determinado como ponto de controle, conhecido como Set Point ( SP ), se o valor atual supera o Set Point, desliga o atuador, se o Set Point supera o valor atual, liga o atuador, considerando, por exemplo, um controle de aquecimento.

As vantagens deste controlador são a simplicidade e o baixo custo, a desvantagem é a oscilação da saída em torno do *Set Point* do controlador, a chamada histerese, não garantindo precisão, apenas uma proximidade do valor do processo ao valor programado, o que pode causar desgaste do atuador, devido aos acionamentos freqüentes.

**Controle proporcional - P**

O controlador entrega energia ao processo com valor proporcional à diferença entre o *Set Point* e o valor atual da grandeza (PV), onde esta diferença é chamada de erro. Em outras palavras, se não houver diferença entre SP e PV, ou seja, erro nulo, significa que a grandeza a ser controlada está estabilizada, não necessitando de nenhuma atuação do controlador, e a partir do momento que houver diferença, o controlador atua, com valor tanto maior quanto maior for o erro. Este tipo de controlador é, ainda, relativamente simples e de baixo custo, porém em alguns casos, dependendo do processo a ser controlado, pode não atingir a estabilidade desejada, inclusive gerando oscilação permanente, conforme ajuste de seu ganho.

Mesmo quando atingido o ponto de estabilidade, ou seja, quando o valor atual se iguala ao Set Point, este tipo de controlador pode apresentar o chamado erro de regime permanente, com a tendência do valor permanecer pouco abaixo do ponto de controle, prejudicando a precisão desta estratégia.

Ex.: Esta estratégia de controle é utilizada em alguns sistemas de controle de velocidade de motores.

**Controle Integral - I**

O controlador integral produz uma saída cujo valor é o somatório do sinal de erro em um determinado instante de tempo, ou seja, a integral do erro. O valor de saída do controlador aumenta enquanto o erro existir, até atingir o valor máximo na saída, e quanto maior o erro mais rápido será este aumento.

Como a saída depende do tempo durante o qual houve erro, este controlador elimina o erro de regime constante, porém apresenta a característica de resposta lenta.

**Controle derivativo - D**

No controle derivativo a saída do controlador é proporcional à taxa de variação do sinal de erro, ou seja, a derivada do erro. De forma simplificada,

esta pode ser entendida como a taxa de variação do erro em relação ao tempo. Se o erro cresce rapidamente, sua saída será um valor grande; se o erro cresce lentamente, sua saída apresentará um valor menor. Este resultado indica a tendência de variação da grandeza a ser controlada. Tem a característica de responder rapidamente a qualquer variação da grandeza em relação ao *Set Point*.

Esta estratégia é muito utilizada em sistemas de posicionamento, movimentação de ferramentas e controle de rotação onde são exigidas ações rápidas frente às modificações nas condições de operação ou perturbações.

O controle apenas derivativo não seria viável na prática, pois, está ligado à tendência de variação e não ao valor absoluto do erro.

**Controle proporcional e derivativo - PD**

Este controle é a combinação entre o controle proporcional e o derivativo.

A ação do controle derivativo somada ao controle proporcional resulta num controlador com resposta rápida no momento do surgimento do erro, entregando um valor alto de energia ao processo, o qual se reduz à medida que o erro se reduz, ou seja, o valor da grandeza controlada se aproxima do *Set Point*.

A ação derivativa quando combinada com a ação proporcional tem justamente a função de "antecipar" a ação de controle a fim de que o processo reaja mais rápido. Neste caso, o sinal de controle a ser aplicado é proporcional a uma predição da saída do processo.

Se este tipo de controle for aplicado a um processo cuja resposta natural é rápida, o sistema poderá ficar instável, por exemplo, nos sistemas de controle de pressão que usualmente apresenta resposta rápida com a ação do atuador.

**Modulação por Largura de Pulso (PWM)**

De acordo com Braga (2002) PWM é uma tecnologia de controle de potência muito eficiente usada para obter um sinal analógico a partir de um digital, e pode ser aplicada em diversos tipos de motores, além de outros tipos de carga como lâmpadas, aquecedores elétricos, solenóides, etc.

O método PWM consiste em um padrão de rápida alternância entre liga e desliga de um período, na qual, parte do tempo ela estará em estado ativo e parte do tempo em estado desativado (Figura 7 ).

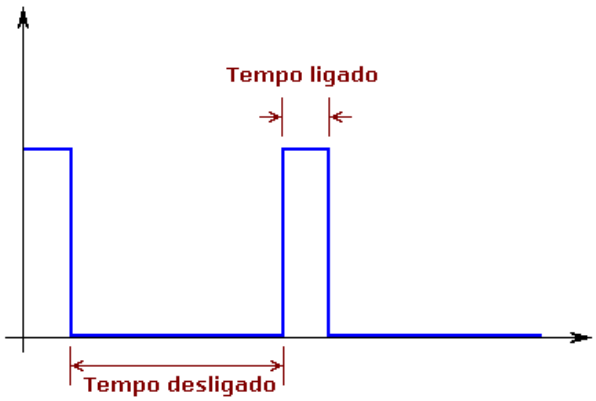


Figura 7: Atuação do PWM

A quantidade de potência entregue à carga depende da duração de cada ciclo ativo (*duty cycle)* do sinal, chamado também de pulso. Se a duração do pulso for a mesma que o intervalo entre os pulsos (*duty cycle* = 50 %) a média da potência aplicada na carga é de 50 %. À medida em que se estende a duração do pulso a média da potência aplicada à carga também aumenta na mesma proporção. Esse processo de controlar a duração do pulso se chama modulação.

**FOTOSENSOR**

O fotosensor ou fototransistor é um dispositivo que funciona baseado no fenômeno da fotocondutividade. Ele pode, ao mesmo tempo, detectar a incidência de luz e fornecer um ganho dentro de um único componente. Como o transistor convencional, o fototransistor é uma combinação de dois diodos de

junção, porém, associado ao efeito transistor aparece o efeito fotoelétrico. Em geral, possui apenas dois terminais acessíveis, o coletor e o emissor, sendo a base incluída apenas para eventual polarização ou controle elétrico.

A incidência de luz (fótons) provoca o surgimento de lacunas na vizinhança da junção base-coletor. Esta tensão conduzirá as lacunas para o emissor, enquanto os elétrons passam do emissor para a base. Isso provocará um aumento da corrente de base, o que por conseqüência implicará numa variação da corrente de coletor “n” vezes maior, sendo essa variação proporcional à intensidade da luz incidente.

Como a base está normalmente desconectada, a corrente que circula por ela dependerá apenas do fluxo luminoso incidente. Assim, na ausência de luz, a corrente de base será zero e o fototransistor estará cortado, resultando na tensão do coletor igual à tensão de polarização. Quando há luz incidindo, a tensão no coletor irá diminuir devido ao aumento da corrente.

## DESENVOLVIMENTO PRÁTICO

### ESTRUTURA FÍSICA

O corpo do robô é constituído de duas placas de policarbonato (Figura 8) um material de altíssima resistência ao impacto, com ótima transparência e moldável quando aquecido, fornecendo grande facilidade de manuseio.

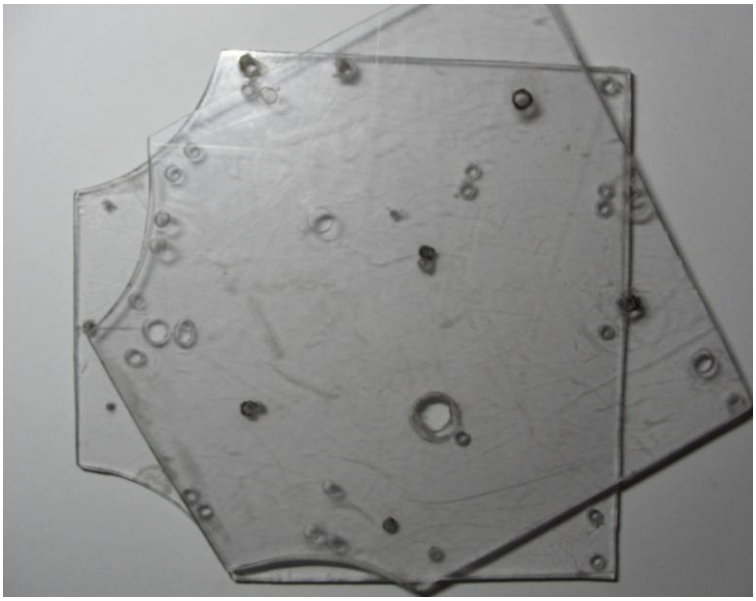

Figura 08: Placas de policarbonato

As placas de policarbonato possuem dimensões 16 cm de comprimento por 14 cm de largura, fixas umas às outras através de parafusos longos e porcas (Figura 9) que além de sustentar a estrutura física do robô, possibilitam a regulagem da altura para melhorar a disposição dos componentes eletrônicos sobre elas.

Figura 9: Parafusos e porcas de sustentação das placas

A placa inferior (Figura 10) abriga os microcontroladores, a bateria, os dois motores, a roda passiva e a matriz de sensores, todos fixos por meio de parafusos e porcas.

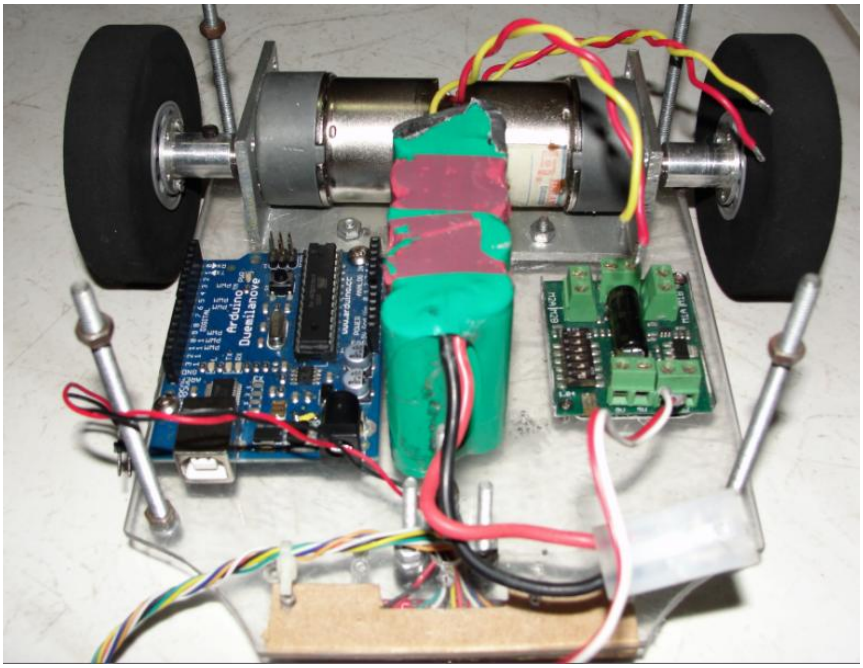

Figura 10: Placa inferior com os componentes eletrônicos dispostos

A placa superior serve como suporte da chave liga/desliga do robô e também como estrutura de proteção dos componentes eletrônicos localizados na placa inferior.

**MÓDULO DE SENSOR**

A matriz de sensores de reflexão QTR-8RC (Figura 11) é utilizada como um sensor de proximidade ou como um sensor de reflexão proporcionando um ótimo foco de aproximação na detecção de bordas e aplicações de seguidores de linha.

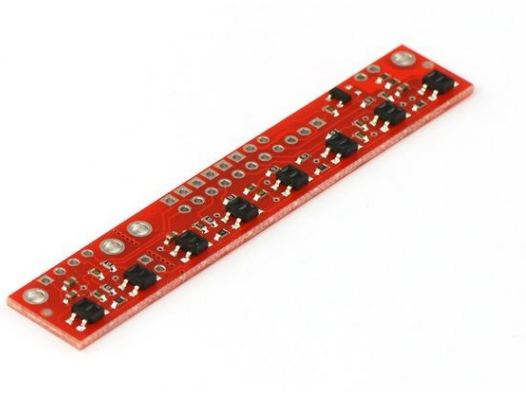

Figura 11: Matriz de sensores de reflexão QTR-8RC

O módulo é um portador conveniente para oito emissores de infra-vermelhos e receptores (foto transistores) agrupados em pares e espaçados em intervalos de 0,375 polegadas (9,525 milímetros) podendo ser ligado a uma voltagem de 3,3v ou 5v controlados por um regulador de tensão interna.

Localizada na parte dianteira do robô, a matriz de sensores fica a uma altura de sete milímetros com os sensores apontados para o chão de forma a fornecer uma leitura precisa da reflexão da luz sobre a superfície (Figura 12).

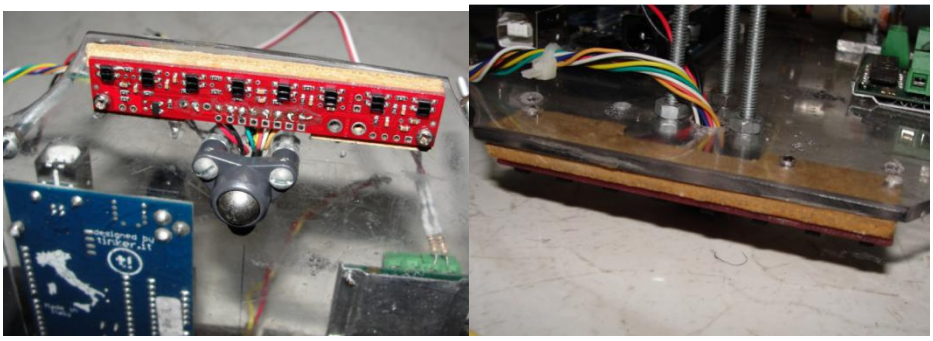

Figura 12: Disposição da matriz de sensores

As oito saídas do módulo QTR-8RC são independentes e não necessitam de um conversor analógico-digital, sendo estas algumas das principais vantagens deste dispositivo.

**Especificações**

- Dimensões: 2,95 "x 0,5" x 0,125 "(sem os pinos de cabeçalho instalados)
- Tensão de funcionamento: 3,3-5,0 V

- Corrente: 100 mA
- Formato de saída: digital I / O compatíveis
- Mínima distância de detecção: 0,125 "(3 mm)
- Máxima recomendada distância de detecção: 0,375 "(9,5 mm)
- Peso sem pinos cabeçalho: 3,09 g

**MÓDULO DE DESLOCAMENTO**

O robô possui dois motores DC do tipo ML-30:1 (Figura 13) dispostos na parte traseira e responsáveis pelo deslocamento de toda a estrutura física.

Cada motor pesa aproximadamente 206 g e é alimentado por uma bateria externa com uma voltagem de 12 v podendo atingir até 200 rotações por minuto a depender do nível de energia fornecido.


Figura 13: Motor DC ML-30:1

Mesmo sendo um motor pesado para a estrutura física do robô, seu torque e principalmente a velocidade atingida são satisfatórios para a utilização de robôs seguidores de linha.

A tabela 1 especifica mais detalhadamente o motor DC utilizado.

Tabela 1. Especificação dos motores ML-30:1

| DESCRIÇÃO | ML-30 |
|---|---|
| | 12V |
| Nenhuma velocidade da carga | 200 rpm |
| Amps nominal | 700mA |
| Corrente sem carga | 115 mA |
| Stall atual | 3.3A |
| binário de bloqueio | 194 oz-in (13.969 g cm) |
| Tensão nominal | 12V (pode ser executado mais) |
| Max Eficiência | 50 oz rpm em (g cm-3600) / 161 |
| Relação da engrenagem | 30:1 |
| Peso | 205.8g |
| Diâmetro | 1.46 "(37mm) |
| Comprimento | 2,24 "(56,9 milímetros) |
| Diâmetro do eixo | 6 mm - com plano |
| Comprimento do eixo | 0,7 "(18 mm) |

As rodas utilizadas são de um composto de borracha e espuma, um material leve que oferece ótima tração e durabilidade, estando conectadas aos motores por meio de um *hub* de alumínio (Figura 14).


Figura 14: Rodas e *hub*

Para fixar os motores à placa inferior de policarbonato foram utilizados parafusos e duas esquadrias de alumínio em formato “L”, cortadas e perfuradas dentro das medidas dos motores (Figura 15 ).


Figura 15: Estrutura de sustentação dos motores

Uma roda passiva (Figura 16) da marca TAMIYA foi colocada na parte dianteira do robô funcionando como uma terceira roda. Essa roda passiva possui uma estrutura plástica com uma esfera de metal em seu interior proporcionando ao robô uma reação rápida aos movimentos gerados pelos motores e um ótimo deslizamento sobre a superfície.

Figura 16: Roda passiva

Três parafusos e seis porcas fazem a conexão da roda passiva com a estrutura física do robô garantindo uma regulagem de altura da parte frontal caso seja necessário.

**MODULO DE POTÊNCIA**

O controle dos motores DC é realizado por um microcontrolador da marca *Sabertooth 2X5* (Figura 17) fabricado pela *DIMENSION ENGINEERRING*. Este *driver* de motor é destinado a pequenos robôs e carros de até 2kg, e possui alta flexibilidade, fácil configuração, podendo controlar dois motores com modos de funcionamento independente da velocidade e direção, tornando-o ideal para projetos de robôs móveis.

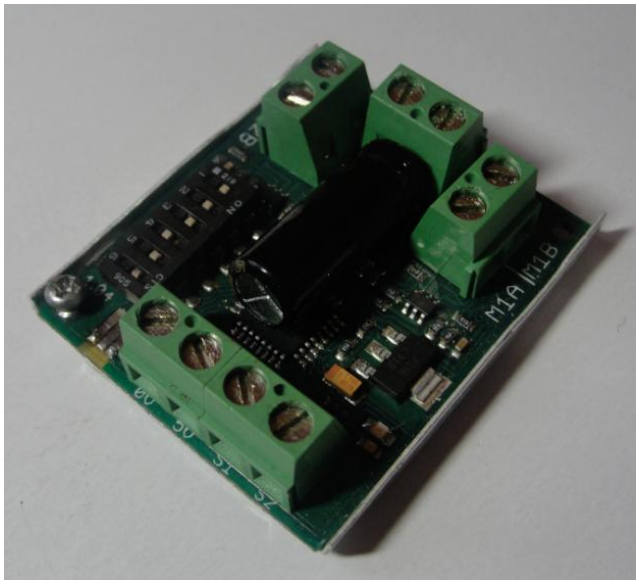


Figura 17: Sabertooh 2x5

O *Sabertooth 2x5* possui dois canais para a conexão dos fios de cada motor DC (M1A e M1B para o primeiro motor; M2A e M2B para o segundo motor), duas entradas S1 e S2 usadas para controlar os motores (conectadas às saídas PWM do microcontrolador) e um conector (positivo e negativo) para a alimentação do *driver* por meio de uma bateria externa de voltagem igual a 12 volts (Figura 18).

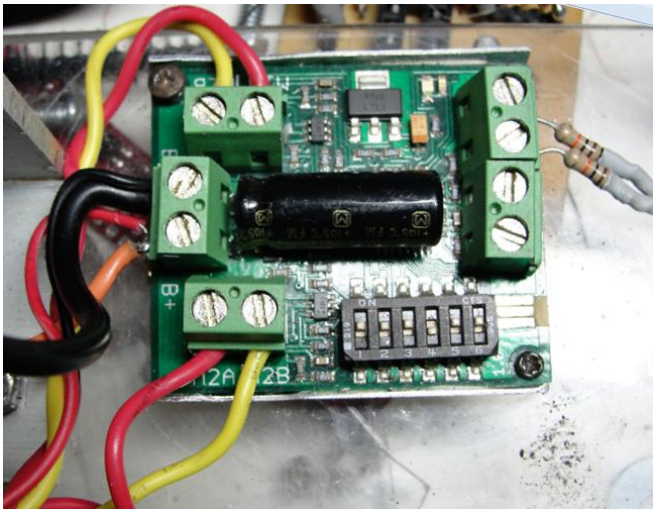


Figura 18 : Conexão dos motores, energia e saídas do Sabertooth 2x5

Foi necessária a utilização de dois resistores em cada entrada S1 e S2 para estabilizar o fluxo do sinal PWM entre o microcontrolador e o controlador dos motores.

**Especificações:**

- Peso: 19g
- Tensão de entrada: 6V a 18V,6-12 células de NiMH ou NiCd,2s-4s LiPo, 6V ou 12V chumbo-ácido (não de chumbo 18 + V!)
- 5A contínuos por canal, o pico de 10A
- Unidade síncrona regenerativa
- Proteção térmica e de sobrecarga
- Tamanho: 45 x 40 x 13 mm

**MÓDULO DE ENERGIA**

O *Lynxmotion* 12.0vdc Ni-MH 1600mAh (Figura 19) é feito de dez pilhas AA de alta capacidade e cabos multi-condutores para uma melhor transferência de energia possível. O cabeamento facilita a reabilitação destas baterias em projetos de robótica e eletrônica. Este pacote pesa 300g, cerca de metade do peso de uma bateria Ni-Cad equivalente.

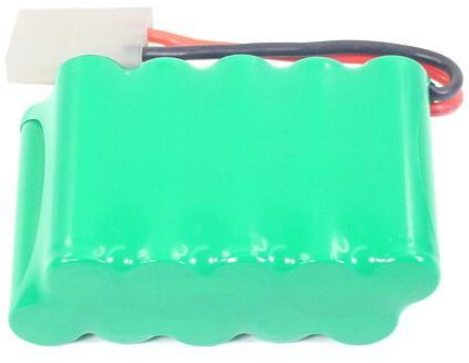

Figura 19: *Lynxmotion* 12.0 vdc Ni-MH 1600 mAh

**Especificações:**

Pacote de medidas 1,92 x 3,05 cm x 1,05 "

- 14.5vdc tensão nominal
- Capacidade nominal: 1600mAh
- Max corrente de descarga: 10 ~ 16A
- Rapid carga: 1600mA / 1,2 horas
- Peso: 300g

A bateria faz a alimentação de todos os dispositivos eletrônicos utilizados no projeto (motores, microcontroladores e sensores).

Um interruptor liga / desliga de dois polos (Figura 20) realiza o fornecimento ou corte da energia da bateria para os componentes eletrônicos do robô.

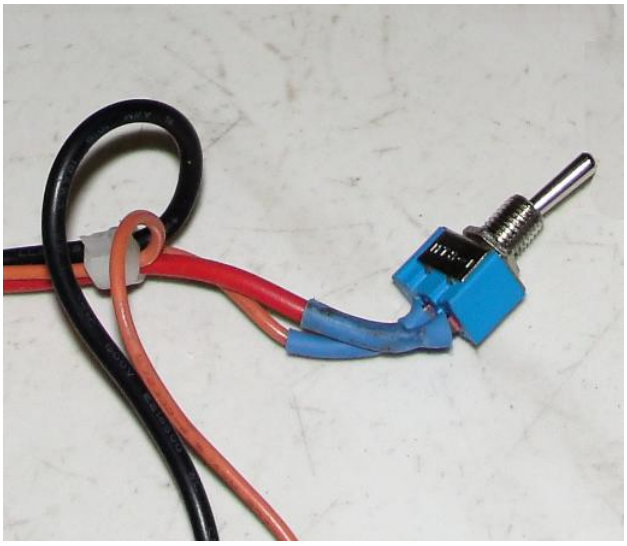

Figura 20: Interruptor liga/desliga

**MODULO CONTROLADOR**

O microcontrolador utilizado foi um *Arduino Duemilanove* (Figura 21) baseado no ATmega328.

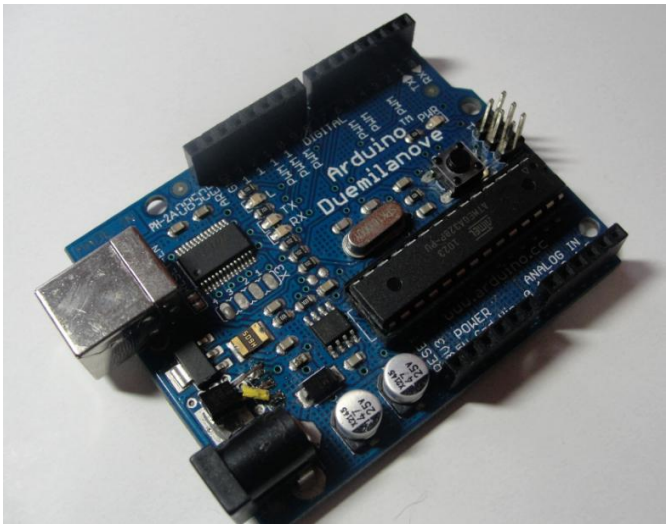


Figura 21 : Microcontrolador Arduino Duemilanove

Tabela 2. Principais características do microcontrolador:

| | |
|---|---|
| Microcontrolador | ATmega328 |
| Voltagem de operação | 5V |
| Voltagem de entrada (recomendável) | 7-12V |
| Voltagem de entrada (limite) | 6-20V |
| Pinos digitais de entrada/saída | 14 (dos quais 6 oferecem saída PWM) |
| Pinos analógicos | 6 |
| Corrente DC por pino de E/S | 40 mA |
| Corrente DC para pinos 3.3V | 50 mA |
| Memória flash | 32 KB (ATmega328) dos quais 2 KB usados pelo *bootloader* |
| SRAM | 2 KB |
| EEPROM | 1 KB |
| Velocidade do clock | 16 MHz |

O Arduino oferece um ótimo custo/benefício sendo um microcontrolador de fácil manuseio que não requer conhecimentos avançados em eletrônica, proporcionando agilidade na conexão e manipulação de dispositivos de entrada/saída.

O ATmega328 tem o papel de processar as informações oriundas dos sensores (matriz de oito sensores), tomar decisões com base no seu código armazenado (software) e enviar ações aos atuadores (*driver* dos motores).

Oito pinos digitais (2, 3, 4, 5, 6, 7, 8 e 9) foram usados para a conexão da matriz de sensores e dois pinos digitais PWM (10 e 11) para o controle da potência dos motores (esquerda e direita) conforme Figura 22.

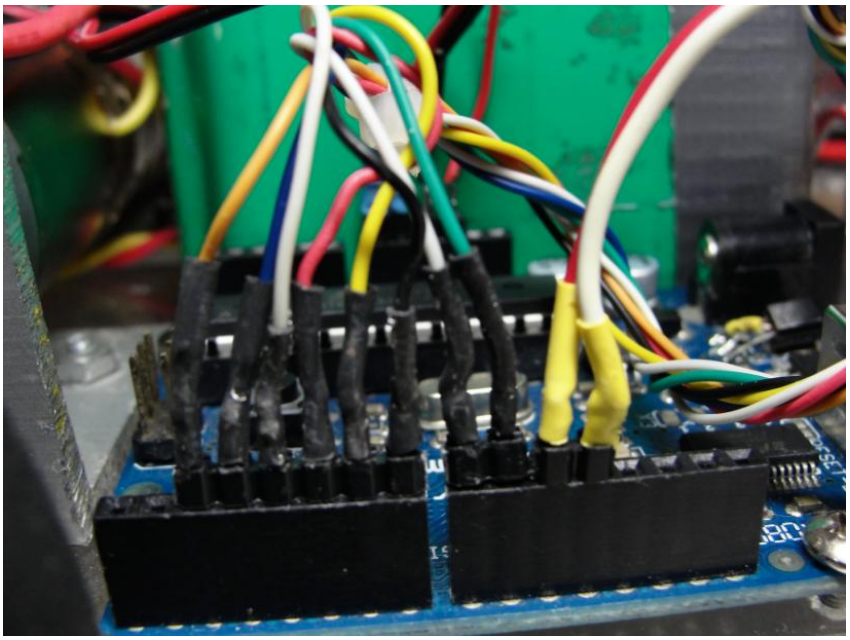

Figura 22: Conexão da matriz de sensores e dos motores na placa do Arduino

O microcontrolador é alimentado com uma voltagem de 12 volts interligado na mesma entrada de alimentação do controlador dos motores.

**O ROBÔ**

O robô após montado (Figura 23) possui 20 cm de largura, 17,5 cm de comprimento, 9 cm de altura e peso igual a xx kg.

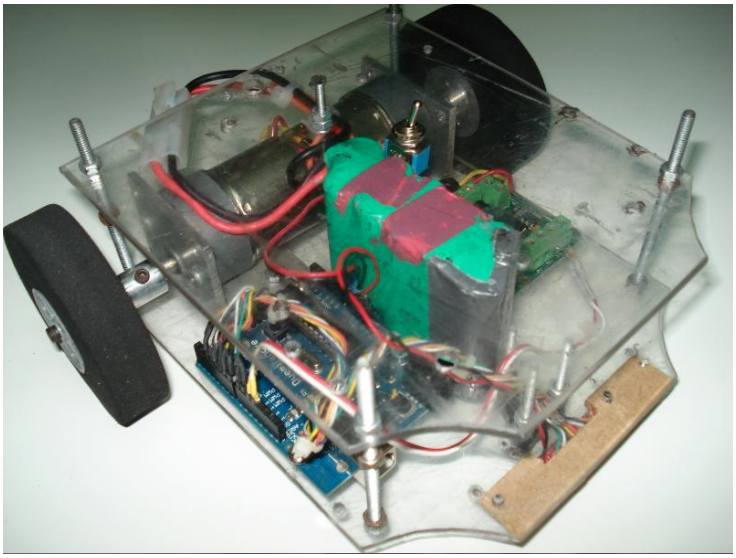

Figura 23: Robô

***SOFTWARE***

Toda a programação do robô foi realizada no ambiente de desenvolvimento do Arduino (Figura 24) que recebe comandos em C/C++ permitindo criar com facilidade muitas operações de entrada/saída.

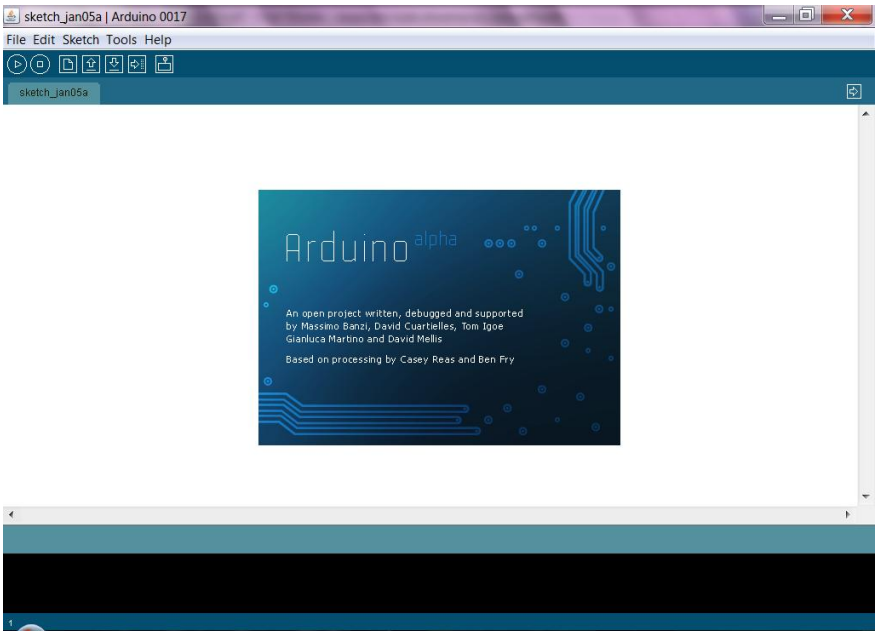


Figura 24: Ambiente de programação do Arduino

O *software* deve ser capaz de fornecer ao robô autonomia suficiente para percorrer um caminho seguindo a linha preta sem perdê-la, passando por curvas abertas ou fechadas com uma velocidade satisfatória e com uma movimentação suave obtida através da aplicação do controle proporcional e derivativo - PD.

O bloco inicial do programa define a declaração de algumas variáveis utilizadas no robô como velocidades dos motores, declaração dos pinos onde estão conectados os motores na placa do Arduino além das variáveis e constantes que compõem a lógica do controlador PD.

A velocidade aplicada aos motores recebeu o seu valor máximo igual a 255, passada como parâmetro para a instrução que usa o PWM como saída.

A manipulação via software da matriz de oito sensores é feita através de funções e métodos contidos em classes da biblioteca *PololuQTRSensors.h* (inserida no início do programa) fornecida pelo fabricante do dispositivo para a comunicação do módulo QTR-8RC com o microcontrolador Arduino.

Um objeto denominado **matrizSensores** é instanciado informando em quais pinos do Arduino estão conectados cada sensor da matriz de sensores (pino 2, 3, 4, 5, 6, 7, 8 e 9) bem como a quantidade de sensores usados no projeto (8 sensores) conforme mostrado no trecho de código a seguir.

```
PololuQTRSensorsRC matrizSensores((unsigned char[]) {2,3,4,5,6,7,8,9}, 8);
```

O segundo bloco descrito no Arduino como *void setup()*, é a área destinada à especificação de quais pinos definidos anteriormente são entradas (*input*) ou saídas (*output*) e algumas configurações que o programa precise antes de iniciar definitivamente. Neste caso, os dois motores que estão nos pinos 10 e 11 são definidos como saída.

```
pinMode(motorEsquerda,OUTPUT);
pinMode(motorDireita,OUTPUT);
```

Com o objeto **matrizSensores** instanciado na área de declaração, uma rotina de calibragem é feita visando expor cada um dos oito sensores à superfície por onde o robô irá percorrer captando a refletância da luz sobre

essa superfície. O método *calibrate()* é invocado dentro de um *for* por um período de cinco segundos, fazendo o robô girar e deslizar no seu eixo central em sentido horário passando sobre a linha preta e sobre a área branca diversas vezes, realizando assim a calibragem dos sensores. Esse giro consiste basicamente na movimentação do motor esquerdo para frente e do motor esquerdo para trás numa mesma velocidade.

O terceiro e último bloco do programa, o *void loop()*, contém instruções para o robô se deslocar sobre a linha preta aplicando o controle PD, sendo executado de forma cíclica até que o microcontrolador seja desligado ou reiniciado.

Dentro do escopo do *void loop()* o método *readLine()* faz a leitura de todos os sensores da matriz, armazenando em uma variável definida como ***leituraSensores*** um número inteiro que indica a localização da linha preta sob os sensores.

```
unsigned int arraySensores[totalSensores];
int leituraSensores = matrizSensores.readLine(arraySensores);
```

Cada sensor individual retorna valores entre 0 e 1000 onde o 0 (zero) corresponde à reflexão máxima da superfície encontrada durante a calibração (neste caso o branco máximo), e o 1000 como menor reflexão encontrada (preto máximo). Como a matriz é composta de oito sensores numerados de zero a sete (Figura 25), o valor total retornado pelo método *readLine()* varia de 0 a 7000.


Figura 25: Numeração dos sensores na matriz

A posição ideal do robô sobre a linha preta (Figura 26) é dada pelo posicionamento dos sensores centrais da matriz (sensor 3 e sensor 4) sobre o preto, e os demais sensores sobre a parte branca sendo os sensores 0, 1 e 2 à esquerda da linha preta e os sensores 5, 6 e 7 à direta da linha preta.

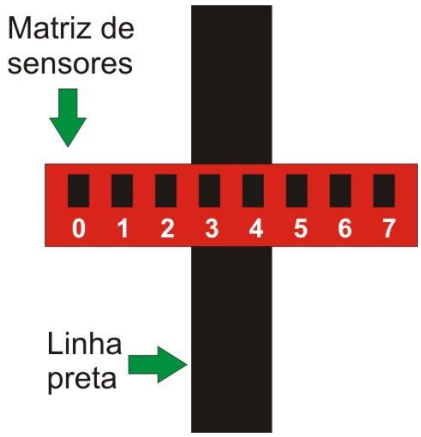


Figura 26: Ponto ideal (*setpoint*) do robô sobre a linha preta

Nessa configuração ideal, também chamada de ponto de ajuste ou *setpoint*, o número inteiro retornado pelo método *readLine()* corresponde a 3500, ou seja, o robô está simetricamente sobre a linha preta e portanto, a velocidade aplicada aos dois motores por meio do PWM deve ser a mesma para que ele se desloque em linha reta para a frente.

Os valores menores que 3.500 tendendo a zero, significam que a linha preta está sob os sensores 0, 1, 2 ou 3 com o robô deslocando-se para o lado direito (Figura 27-a). Neste caso, a velocidade do motor direito deverá ser maior que o motor esquerdo a fim de fazer o robô retornar ao ponto ideal.

Para valores maiores que 3.500 e menores que 7.000 o robô tende para o lado esquerdo com a linha preta sob os sensores 5, 6 ou 7 (Figura 27-b). Nesta situação, a velocidade do motor esquerdo deverá ser maior que a velocidade do motor direito.

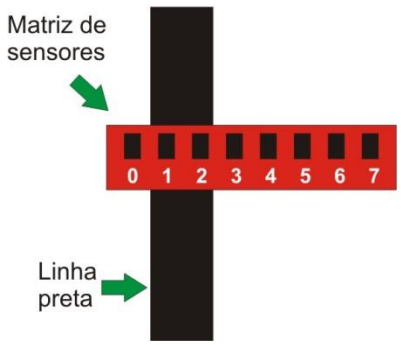


Figura 27a : Robô tendendo para a direita

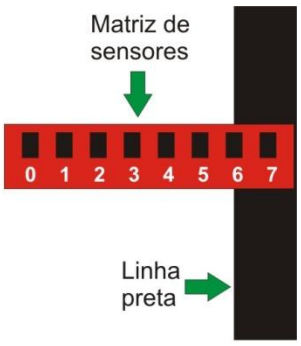


Figura 27b: Robô tendendo para a esquerda

Dessa forma, podem-se estabelecer as seguintes relações:

| Leitura dos sensores | Localização do robô | Velocidade dos motores |
|---|---|---|
| 0 > leitura < 3.500 | Lado direito da linha | motor direito > motor esquerdo |
| leitura = 3.500 | Sobre a linha | motor direito = motor esquerdo |
| 3.500 > leitura < 7.000 | Lado esquerdo da linha | motor direito < motor esquerdo |

Os valores diferentes de 3.500 são considerados como **erros de deslocamento** que podem ser pequenos (próximos de 3.500) ou grandes (próximos ou iguais a zero ou a 7.000).

No programa, esse erro é calculado subtraindo a leitura atual dos sensores pelo ponto de ajuste (definido como 3500), sendo aplicado posteriormente na fórmula do PD.

```
int erro = leituraSensores - pontoAjuste;
```

A fórmula do controle proporcional/derivativo realiza uma compensação da velocidade de cada motor que pode ser brusca ou discreta a depender do valor do erro. Essa compensação, declarada no programa como uma variável inteira chamada *sinalControlador*, é justamente o cálculo resultante das constantes proporcional e derivativa em função do erro obtido em cada leitura dos sensores sobre a linha preta.

```
// CÁLCULO DO PD

int erro = leituraSensores - 3500;
int sinalControlador = KP * erro + KD * (erro - erroAnterior);
erroAnterior = erro;
int novaVelocidadeMotor1 = velocidadeMotor1 + sinalControlador;
int novaVelocidadeMotor2 = velocidadeMotor2 - sinalControlador;
```

Antes de enviar as novas velocidades para cada motor foi necessário realizar uma correção desses valores, uma vez que o PWM que controla a velocidade dos motores varia de 127 a 255 (onde 127 os motores ficam

parados e 255 em velocidade máxima), e o resultado calculado pelo sinal controlador pode gera valores fora deste intervalo.

Por fim, as novas velocidades são aplicadas a cada motor e todo o código contido no *void Loop()* é executado diversas vezes fazendo o robô seguir a linha preta pelo ambiente.

**AJUSTE DO CONTROLADOR PROPORCIONAL E DERIVATIVO (PD)**

O controle PD funciona por meio do ajuste das duas constantes definidas no bloco de declaração do programa como Kp (constante proporcional) e Kd (constante derivativa).

A configuração do controlador PD foi realizada de forma heurística que se baseia em determinar os melhores ganhos das constantes Kp e Kd através de tentativa e erro verificando o desempenho do processo até que o desempenho desejado seja obtido - um robô seguidor de linha com velocidade e estabilidade satisfatória.

Foi aplicada a velocidade máxima que os motores podem oferecer através do PWM ajustado com valores iguais a 255 para cada saída dos motores.

Inicialmente as duas constantes Kp e Kd foram zeradas e somente a constante proporcional Kp foi sendo gradativamente ajustada (Tabela 3).

Tabela 3: Ajuste da constante proporcional

| VALORES DA CONSTANTE PROPORCIONAL (Kp) | RESULTADO OBTIDO |
|---|---|
| Kp = 0.0 | O robô não segue a linha |
| Kp = 0.05 | O robô segue apenas em linha reta não faz curvas |
| Kp = 0.1 | O robô segue a linha parcialmente, mas instável em curvas |
| Kp = 0.2 | O robô segue a linha parcialmente, mas ainda instável em curvas |
| Kp = 0.3 | Robô 100% estável na linha e em curvas |
| Kp = 0.4 | Robô segue a linha, porém apresenta variações discretas em zig-zag |
| Kp = 0.5 | Robô segue a linha, porém apresenta variações suaves em zig-zag |
| Kp = 0.8 | Robô segue a linha, porém apresenta variações intensas em zig-zag |
| Kp = 2.0 | Robô segue a linha, porém apresenta fortes variações em zig-zag |

O melhor resultado obtido para o controle proporcional foi Kp=0.3, deixando o robô preciso durante o seu percurso sobre a linha preta tanto em retas quanto em curvas.

O próximo passo foi ajustar o valor da constante derivativa Kd (Tabela 4).

Tabela 4: Ajuste da constante derivativa

| VALORES DA CONSTANTE DERIVATIVA (Kd) | RESULTADO OBTIDO |
|---|---|
| Kd = 0 | Nenhuma interferência |
| Kd = 5 | Boa suavização |
| Kd = 7 | Ótima estabilidade do robô |
| Kd = 10 | Boa suavização com discreta pertubação |
| Kd = 20 | Pequenas oscilações do robô |
| Kd = 40 | Fortes oscilações gerando perda da linha |

Um valor satisfatório foi obtido com a constante Kd=7, tendo como conseqüência uma suavização do deslocamento do robô sobre a linha e uma ótima estabilidade.

## CONCLUSÃO

A ação de controle proporcional-derivativo (PD) foi capaz de estabilizar a movimentação do robô sobre a linha, fazendo-o percorrer o caminho de forma suave e com velocidade satisfatória.

O controle integral (I) foi desprezível uma vez que o robô atingiu o seu objetivo somente com as constantes proporcional e derivativa.

Observou-se também a eficiência do microcontrolador Arduino e dos demais componentes eletrônicos na leitura e processamento das informações atuando precisamente sobre todo o sistema.

## REFERÊNCIAS BIBLIOGRÁFICAS


1. ARDUINO, 2010. Disponível em <http://www.arduino.cc>, Acesso em 17.11.2010.
2. BRAGA, Newton C. Eletrônica básica para mecatrônica. São Paulo: Editora Saber, 2005.
3. BURLAMAQUI, A. et AL., Introdução a Robótica. 2010. Disponível em:<http://aquilesburlamaqui.wdfiles.com/localfiles/apresentacoes/IntroducaoARobotica.pdf>, Acesso em 20.08.2010.
4. CORTELETTI, D. Dossiê Técnico: introdução à programação de microcontroladores Microchip PIC. SENAI-RS. Centro Tecnológico de Mecatrônica. 2006.
5. SILVA, A. F. da RoboEduc: uma metodologia de aprendizado com robótica educacional.2009. Tese (Doutorado). Universidade Federal do Rio Grande do Norte. Natal.
6. ZILLI, S. do R. A Robótica Educacional no Ensino Fundamental: Perspectivas e Prática. 2004. 89 f. Dissertação (Mestrado em Engenharia de Produção). Programa de Pós-Graduação em Engenharia de Produção, UFSC, Florianópolis.

## ANEXO

## CÓDIGO FONTE DO ROBÔ

```
#include <PololuQTRSensors.h>

int motorEsquerda = 11;
int motorDireita = 10;
float KP = 0.3;
float KD = 7;
int erroAnterior = 0;
int velocidadeMotor1 = 255;
int velocidadeMotor2 = 255;
PololuQTRSensorsRC matrizSensores((unsigned char[])
{2,3,4,5,6,7,8,9}, 8);  // 0-1-2-3-4-5-6-7

void setup()
{
  pinMode(motorEsquerda,OUTPUT);
  pinMode(motorDireita,OUTPUT);

  analogWrite(motorEsquerda,127);
  analogWrite(motorDireita,127);
  delay(2000);

  analogWrite(motorEsquerda,175);
  analogWrite(motorDireita,50);

  // CALIBRAÇÃO DOS SENSORES - 5 SEGUNDOS
```

```
        int i;
        for (i = 0; i < 250; i++)
        {
          matrizSensores.calibrate();
          delay(20);
        }
        analogWrite(motorEsquerda,127);
        analogWrite(motorDireita,127);
        delay(2000);

      }
      ////////////////////////// FIM DO SETUP
/////////////////////

      void loop()
      {
        // LEITURA DOS SENSORES
        unsigned int arraySensores[8];
        int leituraSensores =
matrizSensores.readLine(arraySensores);

        // CÁLCULO PD
        int erro = leituraSensores - 3500;
        int sinalControlador = KP * erro + KD * (erro -
erroAnterior);
        erroAnterior = erro;
        int novaVelocidadeMotor1 = velocidadeMotor1 +
sinalControlador;
        int novaVelocidadeMotor2 = velocidadeMotor2 -
sinalControlador;
```

```
//// CORREÇÃO DOS VALORES EXTREMOS DOS MOTORES
  if (novaVelocidadeMotor1 < 127)
    novaVelocidadeMotor1 = 127;
  if (novaVelocidadeMotor2 < 127)
    novaVelocidadeMotor2 = 127;

  if (novaVelocidadeMotor1 > 255)
    novaVelocidadeMotor1 = 255;
  if (novaVelocidadeMotor2 > 255)
    novaVelocidadeMotor2 = 255;

  // APLICAÇÃO DAS NOVAS VELOCIDADES DOS MOTORES VIA PWM
  analogWrite(motorEsquerda,novaVelocidadeMotor1);
  analogWrite(motorDireita,novaVelocidadeMotor2);

} // FIM DO VOID LOOP
```