Universidade Estadual do Sudoeste da Bahia – UESB

Departamento de Ciências Exatas

Marcos Pereira dos Santos

Robótica Educacional:

desenvolvimento de um robô móvel de baixo custo

Orientador:
Prof. Ms. Adilson de Lima Pereira

Vitória da Conquista – BA
Novembro de 2010

Universidade Estadual do Sudoeste da Bahia – UESB

Departamento de Ciências Exatas

Marcos Pereira dos Santos

Robótica Educacional:

desenvolvimento de um robô móvel de baixo custo

Monografia de conclusão de curso apresentada ao Departamento de Ciências Exatas – DCE – da Universidade Estadual do Sudoeste da Bahia – UESB – como requisito para obtenção do título de Bacharel em Ciência da Computação.

Área de Concentração: Robótica, Informática na Educação

Orientador:
Prof. Ms. Adilson de Lima Pereira

Vitória da Conquista – BA
Novembro de 2010

Universidade Estadual do Sudoeste da Bahia – UESB

Departamento de Ciências Exatas

Marcos Pereira dos Santos

Robótica Educacional:
desenvolvimento de um robô móvel de baixo custo

Monografia de conclusão de curso apresentada ao Departamento de Ciências Exatas – DCE – da Universidade Estadual do Sudoeste da Bahia – UESB – como requisito para obtenção do título de Bacharel em Ciência da Computação.

Versão final aprovada pela Banca Examinadora em 16.12.2010.

BANCA EXAMINADORA:

_____

Prof. Ms. Adilson de Lima Pereira - UESB
(Orientador)

_____

Prof. Ms. Cláudio Rodolfo Sousa de Oliveira - UESB

_____

Prof. Alexandro da Silva Santos – IFBA/Vitória da Conquista

Vitória da Conquista – BA
Junho de 2010

Dedicatória

Dedico a Vana pelo seu profundo amor,
apoio eterno e infinita paciência,
sempre lhe serei grato.
Aos meus magníficos filhos,
que assistiram e compartilharam,
com humor e encorajamento.

Agradecimentos

Agradeço a Deus
por ter colocado em minha vida
uma legião de anjos de luz.

SUMÁRIO

Lista de Abreviações e Siglas

AA	Ácido Alcalina
Ah	Ampère Hora
AVR	*Advanced Virtual* RISC
CA	Construção de Argumentação
CD	*Compact Disc*
CF	Compreensão dos Fenômenos
DC	Corrente Contínua
DL	Domínio das Linguagens
DVD	*Digital Video Disc* ou *Digital Versatile Disc*
EEPROM	*Electrically-Erasable Programmable Read-Only Memory*
EP	Elaboração de Propostas
FEI	Fundação Educacional Imanciana
FTDI	*Future Technology Devices International*
FURG	Universidade Federal do Rio Grande
I/O	*Input / Output*
IBM	*International Business Machines*
ICSP	*In-Circuit Serial Programming*
IDE	*Integrated Development Environment* (Ambiente Integrado de Desenvolvimento)
ITA	Instituto Tecnológico de Aeronáutica
JIRA	*Japan Industrial Robot Association*
LCD	*Liquid Crystal Display*
MHz	*megahertz*
MIP	Milhões de Instruções por Segundo
MIT	*Massachussets Institute Technology*
OBR	Olimpíadas Brasileira de Robótica
PC	*Personal Computer*
PIC	*Programmable Interface Controller*
PWM	*Pulse Width Modulation*
RCX	*Robotic Command Explorer*

RIA	*Robot Institute of America*
RIS	*Robotics Invention System*
rpm	Rotações por minuto
SP	Enfrentamento de Situações-Problema
SRAM	*Static Random Access Memory*
UART	*Universal Assyncrhonous Receiver and Transmiter*
TTL	*Transistor-Transistor Logic*
UFRGS	Universidade Federal do Rio Grande do Sul
UFRN	Universidade Federal do Rio Grande do Norte
UNB	Universidade de Brasília
UNESP	Universidade Estadual Paulista
USB	*Universal Serial Bus*
USART	*Universal Synchronous Asynchronous Receiver Transmitter*

Lista de Figuras

Lista de Tabelas

Resumo

Este trabalho teve por objetivo desenvolver um robô, denominado de Virgulino, como opção de kit de robótica, viável economicamente, para uso em ambiente educacional. A robótica, nos dias atuais, é considerada como uma ferramenta auxiliar de significativa importância para o estudo de conceitos teóricos de diversas disciplinas. Ela facilita a assimilação de conteúdos, já que pressupõe a participação do educando na construção do seu próprio conhecimento, servindo de estímulo à sua criatividade e inteligência. O Virgulino foi testado na OBR de 2009, apresentando desempenho satisfatório, o que o certificou como opção de kit educacional para implantação da robótica em escolas do ensino fundamental e médio. Este resultado se deu graças à utilização da plataforma Arduino, que contribuiu para o seu baixo custo e caráter *open-source*.

Palavras-Chave: Robótica Educacional, Plataforma Arduino, kit Educacional

Abstract

The main goal of this paper is to develop a robot, known by Virgulino, in order to form an economical suitable robotic equipment kit to be applied to the educational environment. Robotics, nowadays, is considered an auxiliary tool of significant importance for the study of theoretical concepts over several disciplines. It makes the assimilation of content to become easier, since it presupposes the participation of the learner in the process of building his own knowledge, as much as the increase of his creativity and wisdom. The Virgulino was tested in the OBR (Brazilian Robotics Olympics) 2009, presenting a satisfaction performance, which certified this robot as an option for deployment of robotics education in elementary schools and high schools. This result came through the use of the Arduino platform, which contributed to its low cost and open-source approach.

Related Words: Educational Robotics, Arduino platform, educational kit.

Introdução

Atualmente, a Robótica é vista como uma realidade que cresce dia após dia na vida do ser humano. É comum se observar o uso de robôs em diversos segmentos da sociedade, realizando tarefas que vão desde a limpeza do pó nos dutos de ar condicionados, como em tarefas mais complexas, tipo a limpeza de lixo tóxico, exploração subaquática e espacial, cirurgias médicas, mineração, busca de minas terrestres, montadoras de automóveis entre outras. A robótica traz soluções rápidas e eficientes, proporcionando, a longo prazo, um retorno rentável ainda que possa, inicialmente, apresentar um custo relativamente alto para sua implementação.

Pode-se dizer que um dos grandiosos passos da Robótica foi a sua inserção no meio educacional, a chamada Robótica Educacional. No Brasil, ela vem sendo implementada, principalmente a partir da década de 90, se configurando como uma ferramenta de auxílio para a assimilação de conceitos teóricos de diversas disciplinas. Tem-se buscado caminhos, através da Robótica Educacional, para oferecer ao cidadão, na fase escolar, um melhor preparo para enfrentar a competitividade do mundo globalizado. O intuito da inserção dela na educação é facilitar a construção do conhecimento, através de experiências concretas e práticas, estimulando a criatividade e inteligência dos educandos.

Para contribuir com a inserção da Robótica Educacional nas instituições de ensino médio e fundamental, foi criada a OBR (Olimpíadas Brasileira de Robótica). Uma competição nacional onde o objetivo principal é promover o conhecimento da Robótica através de competições e despertar nos participantes o interesse em seguir carreiras científico-tecnológicas, usando como artifícios a realização de provas em âmbito nacional, realização de torneios locais, parcerias com universidades, disponibilização de *web site* com material de Robótica Educacional, fóruns para troca de experiência entre educadores.

Os robôs que participam destas competições são desenvolvidos com kits de Robótica de grandes empresas existentes no mercado, como por exemplo: a Lego *Mindstorns* e a *Vex Robotics*. Todos eles são soluções de qualidade

indiscutível, entretanto, possuem um elevado custo para implantação em instituições da rede de ensino pública e privada.

Com a perspectiva de se desenvolver um *kit* educacional de baixo custo, que possibilitasse a democratização da Robótica, principalmente em escolas públicas, é que foi desenvolvido este estudo. Assim, o robô Virgulino foi projetado para se tornar uma solução viável economicamente e ser o*pen-source*. A sua aplicação na OBR de 2009 demonstrou que ele pode ser usado como *kit* educacional para o ensino da Robótica, podendo ser capaz de desenvolver robôs em condições de participar de competições desta natureza com alta qualidade e bom desempenho, obtendo resultados que o colocam em pé de igualdade em relação aos *kits* já disponíveis.

O estudo foi estruturado em quatro capítulos. No primeiro, foi feita uma abordagem histórica acerca da Robótica, introduzindo informações sobre a Robótica Educacional, enquanto proposta pedagógica, alicerçada no construcionismo de *Seymour Papert*, que se fundamenta na concepção de que o conhecimento se constrói a partir de situações concretas, levando o educando a ser participante ativo desta construção. A Robótica Educacional foi apresentada como ferramenta auxiliar na educação, tendo um caráter integrador de diversas disciplinas. Ainda neste capítulo foi trabalhado o conceito de robô, referindo sobre seus componentes essenciais: *hardware* e *software*, enfatizado a importância dos microcontroladores no contexto atual. No segundo capítulo foi falado sobre a plataforma usada no robô Virgulino, o Arduino, suas características e ferramentas de desenvolvimento. Mas somente no terceiro capítulo é que o robô Virgulino foi descrito com detalhes, sendo apresentado o seu projeto físico e eletrônico, além do *software* para ele desenvolvido. No último capítulo foi abordado sobre os testes realizados com o robô Virgulino e os resultados alcançados com a sua aplicação na OBR de 2009. O estudo foi finalizado com a análise do robô Virgulino como uma opção de *kit* educacional para o desenvolvimento da Robótica em escolas do ensino fundamental e médio.

Capítulo I

Robótica

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 apóia 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, s/d).

Nos dias atuais, a Robótica se configura numa área de crescente desenvolvimento. Isto decorre dos inúmeros recursos que os sistemas de microcomputadores vêm oferecendo. Grandes centros de tecnologia se ocupam

incansavelmente a criação de novas tecnologias, criando robôs capazes de lidar com tomadas de decisão cada vez mais complexas.

1. Leis da Robótica

Isaac Asimov (REZENDE, 1992), criador do termo “Robótica” elaborou três leis que até hoje permanecem vivas nas regras da comunidade científica. São elas:

Primeira Lei: Um robô não pode ferir um ser humano ou, por omissão, permitir que um ser humano sofra algum mal.

Segunda Lei: Um robô deve obedecer às ordens que lhe sejam dadas por seres humanos, exceto nos casos em que tais ordens contrariem a Primeira Lei.

Terceira Lei: Um robô deve proteger sua própria existência desde que tal proteção não entre em conflito com a Primeira e Segunda Leis.

Para *Asimov*, a razão em criar essas leis era a de incentivar a tornar-se possível a existência de robôs inteligentes, mas que nunca ultrapassassem limites a ponto de se contraporem ao domínio do homem. Com isto, *Asimov* pretende ressaltar que um robô não pode fazer mal à humanidade e nem permitir que ela sofra algum mal, prezando desta forma, o bem da humanidade como algo primordial aos indivíduos.

2. Robótica Educacional

A Robótica Educacional, também chamada de Robótica Pedagógica, é um assunto abordado no Brasil desde a década de 50 (PIEROTTI, 2007). Mas, somente a partir de meados dos anos 1980, começou a ser reconhecida mais amplamente como um importante recurso pedagógico. Ela pode ser considerada como uma ferramenta auxiliar, utilizada pelo educador, para a melhor e mais fácil assimilação de conceitos teóricos das diversas disciplinas estudadas em sala de aula (ZANELATTO, 2004), possibilitando, portanto, a interdisciplinaridade. Representa ainda uma alternativa de resposta às exigências impostas à educação, neste momento histórico de grandes mudanças societárias,

proporcionando condições para que a escola se renove e possa continuar atendendo ao seu propósito de formação e informação. Sua utilização no dia a dia escolar favorece um ambiente estimulante e motivador, auxiliando o processo ensino-aprendizagem e também possibilitando a construção de novas relações entre educador-educando e educando-conhecimento. Este, o conhecimento, passa a ser construído a partir de experiências práticas, tornando-se significativo e contextualizado para ambos (ZILLI, 2004: p.18-39).

A Robótica Educacional, enquanto proposta pedagógica, foi desenvolvida por *Seymour Papert*, psicólogo do Laboratório de Inteligência Artificial do MIT (*Massachussets Institute Technology*), que, a partir dos princípios do construtivismo piagetiano[1], elaborou o que chamou de "construcionismo" (ZILLI, 2004: 32-36), adotando o computador como uma ferramenta para a construção do conhecimento e desenvolvimento do aluno" (ALMEIDA, 2000).

Para *Papert*, o "construcionismo" é a sua reconstrução pessoal do "construtivismo", possuindo, deste modo, características diferentes em suas concepções. Enquanto no construtivismo é enfatizado o modo pelo qual o conhecimento é processado internamente, a depender dos diferentes níveis de desenvolvimento cognitivo, no construcionismo, a ênfase esta na praticidade com que o conhecimento é construído, a partir de experiências concretas, possibilitando que o aprendizado ocorra à medida em o conhecimento prévio é projetado, materializando as idéias mentais, de modo a aprimorá-las. *Papert* considera que o educando é quem constrói o seu próprio conhecimento, tendo o educador como facilitador e o computador como recurso pedagógico, e o ato de aprender se dá pela motivação e interesse despertados. A aprendizagem ocorre, segundo ele, pelo estabelecimento de conexões entre o novo conhecimento em construção e os conceitos que já são de domínio do educando nas diversas disciplinas, estando ele a todo momento desafiado a observar, abstrair e inventar. Assim, o conhecimento é construído a partir de situações concretas que motivam

[1] Construtivismo Piagetiano é uma teoria desenvolvida por *Jean Piaget* que busca explicar como a inteligência humana se desenvolve partindo do princípio de que o desenvolvimento da inteligência é determinado pelas ações mútuas entre o indivíduo e o meio. A idéia desenvolvida por ele é que o homem não nasce inteligente, mas também não é passivo sob a influência do meio, isto é, ele responde aos estímulos externos agindo sobre eles para construir e organizar o seu próprio conhecimento, de forma cada vez mais elaborada. (NITZKE, CAMPOS e LIMA, 2010).

o educando a formular hipóteses, testá-las e implementá-las, simulando, desta forma, o método científico (ZILLI, 2004: p.35-39).

3. Conceito de “Robô”

Genericamente, pode-se dizer que robô é uma máquina que realiza determinadas tarefas em substituição ao ser humano. Contudo, alguns estudiosos tem se dedicado ao estudo da Robótica, buscando um maior aprofundamento quanto ao seu conceito.

Para o *Randon House* Dicionário (ZILLI, 2004: p.38), “robô é qualquer máquina ou equipamento mecânico que funcione automaticamente, simulando habilidades humanas”. Segundo *Marsh* (apud ZILLI, 2004: p.38), robô é “uma máquina que, na aparência ou comportamento, imita uma pessoa ou uma ação específica daquela pessoa como um movimento de seu corpo”. Por sua vez, o JIRA *(Japan Industrial Robot Association)* defende que robô é “um sistema mecânico que possui movimentos flexíveis análogos aos movimentos orgânicos, e combina esses movimentos com funções inteligentes e ações semelhantes as do ser humano” (JÁCOBO, 2001: p.7). O JIRA nesta definição compreende função inteligente como aquela que envolve: decisão, reconhecimento, adaptação e\ou aprendizagem. Já o RIA *(Robot Institute of America)* utiliza o seguinte conceito: “robô é um manipulador, re-programável e multi-funcional, projetado para mover materiais, partes, ferramentas ou dispositivos especializados através de movimentos variáveis, podendo ser programados para desempenhar uma variedade de tarefas (SILVA, 2009: p.28).

De acordo às diversas definições acerca do termo “robô”, percebe-se que, embora não se tenha um consenso, pode-se observar que para uma máquina ser considerada um robô, é indispensável a existência de alguns elementos, como: sensores, processadores e atuadores. Sendo assim, adota-se, neste estudo, o conceito dado pelo pesquisador *Ronald Arkin*, do Instituto de Tecnologia da Geórgia (Estados Unidos): “um robô é uma máquina capaz de extrair informações do ambiente e usar conhecimento sobre o mundo de modo a se mover com segurança e com um propósito" (RIGHETTI, 2005).

3.1. Robô Móvel

Segundo *Muir* (1988: p.13) robô móvel é “um robô capaz de se locomover sobre uma superfície somente através da atuação de rodas montadas no robô e em contato com a superfície”. Ou seja, por intermédio da rotação das rodas se obtém um movimento relativo entre o eixo do robô e a superfície de contato. Isto pode ser facilitado com o uso de sensores para que o robô tenha a capacidade de perceber a existência de obstáculos e possíveis fenômenos físicos, oferecendo mais informações para a tomada de decisão, sem a intervenção do homem (NETO, 2007: p.13).

Para este autor, os processos de percepção e decisão podem ser considerados de três formas:

Estratégia Reativa: o comportamento do robô é determinado pelos estímulos exteriores vindos do ambiente obtidos por sensores;

Estratégia deliberada: as informações do ambiente são previamente processadas e a tarefa e o comportamento do robô é previamente determinado;

Estratégia híbrida: junção das duas anteriores.

3.2. Composição de um Robô

Ao longo dos anos a arquitetura dos robôs vem sofrendo alterações, acompanhando as descobertas da tecnologia. É evidente que um robô construído na época da Revolução Industrial, por exemplo, não é o mesmo robô existente nos dias atuais. Se naquele momento havia a necessidade de robôs estáticos, que executassem ininterruptamente as mesmas ações, hoje, os robôs são dotados de percepção sensorial, controle motor, capacidade de locomoção e até comportamento inteligente.

Um robô se compõe necessariamente por dois componentes essenciais, que mantém entre si uma relação de interdependência: *software* e *hardware*. Um não funciona sem o outro, sendo os dois igualmente importantes. No caso específico de um robô, pode-se afirmar que sua existência está condicionada a perfeita interação entre o *software* e *hardware*.

3.2.1. *Hardware*

O *hardware* de um robô é a sua estrutura física, um conjunto de componentes eletrônicos, circuitos integrados, placas, meios de locomoção, chassis etc. Dentre os seus componentes eletrônicos, o microcontrolador tem uma importância preponderante, já que ele é tido como o "cérebro" do robô, sendo responsável pelo processamento e controle de todo o sistema.

Microcontroladores

Daniel *Corteletti* assim conceitua os 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[2], 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. *(*2006: p.3-4)

Os microcontroladores representam um grande avanço da tecnologia e, contemporaneamente, 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 torna 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

---

[2] "circuito integrado são circuitos eletrônicos funcionais, constituídos por um conjunto de transistores, díodos, resistências e condensadores, fabricados num mesmo processo, sobre uma substância comum semicondutora de silício que se designa vulgarmente por *chip*" (ARAÙJO, 2010).

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, conseguintemente, reduz o custo de sua produção. Por este motivo, os microcontroladores se tornaram, hoje em dia, bem mais utilizados para projetos de sistemas embarcados[3].

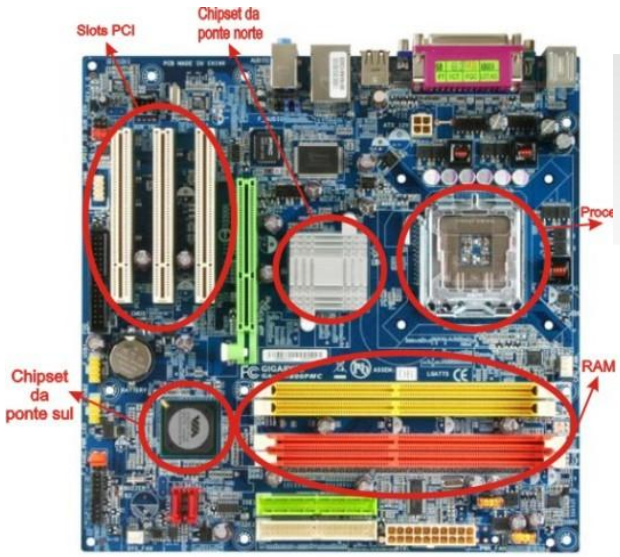


Figura 1 - Placa mãe de um PC

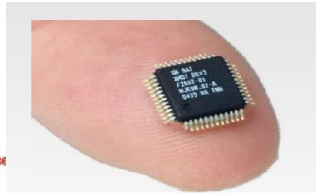


Figura 2 - Microcontrolador

3.2.2. Software

O software se refere à parte lógica, sendo o conjunto de comandos e dados processados pelos circuitos eletrônicos do hardware, ou seja, “inclui o sistema operacional, os programas e todas as informações armazenadas” (MARIMOTO, 2007). É uma aplicação, um programa computacional que possibilita a execução de uma determinada tarefa. Assim, o robô ganha funcionalidade através da decodificação das instruções criadas pelo programa, sendo capaz de, por exemplo, executar a leitura de sensores, controlar motores, definir que direção seguir, qual o deslocamento efetuar.

[3] Sistema embarcado (ou sistema embutido) é um sistema microprocessado no qual o computador é completamente encapsulado ou dedicado ao dispositivo ou sistema que ele controla. Diferente de computadores de propósito geral, como o computador pessoal, um sistema embarcado realiza um conjunto de tarefas predefinidas, geralmente com requisitos específicos (SILVEIRA, s/d).

Os programas são criados a partir de linguagens de programação, podendo estas serem entendidas como um conjunto de regras sintáticas e semânticas usadas para a constituição de um código fonte que, através de um processador, é traduzido em códigos de máquina.

As linguagens de programação, com o passar do tempo, foram evoluindo. As primeiras eram compostas por complexos dígitos binários, sendo substituídas pela linguagem de Montagem (*Assembly*), que utiliza mnemônicos[4] para simplificar os comandos, considerada de baixo nível. Embora tenha facilitado o processo de programação, este tipo de linguagem se configurou em programas extensos, ainda complexos, de difícil interpretação. Mais tarde, surgiram as linguagens de alto nível, que se aproximam mais do pensamento humano. A pioneira foi a Fortran, "uma linguagem voltada para a análise e resolução de problemas matemáticos, criada na metade da década de 50, por pesquisadores da IBM" (PEREIRA, 2003a: p.16), sendo sucedida por várias outras, como a C, por exemplo.

A linguagem C utiliza a programação estruturada, dividindo os programas em módulos ou estruturas independentes entre si, com o propósito de realizar determinadas tarefas. Atualmente, a linguagem C é a mais utilizada para a programação de microcontroladores (PEREIRA, 2003a).

4. *Kits* Educacionais

O Dicionário Interativo da Educação Brasileira conceitua Robótica Educacional como o:

> "Termo utilizado para caracterizar ambientes de aprendizagem que reúnem materiais de sucata ou *kits* de montagem compostos por peças diversas, motores e sensores controláveis por computador e *softwares*, permitindo programar, de alguma forma, o funcionamento de modelos." (MENEZES e SANTOS: 2002)

A implementação da proposta pedagógica da Robótica requer a utilização de *kits* educacionais. Em geral, estes *kits* possuem linguagem própria

[4] "Mnemônicos são abreviações de termos usuais que descrevem a operação efetuada pelo comando em código de máquina (...). Assim, em vez de se escrever o comando em código de máquina 0011000010001100, pode-se utilizar o comando MOVLW 0x8c para realizar a mesma tarefa (definição do autor)

de programação e são compostos por sensores, motores, engrenagens, processadores, transmissores e interfaces de interação, componentes ideais para a construção de protótipos[5]**.** Atualmente, há uma variedade deles disponível no mercado.

A grande vantagem do uso destes *kits* é que eles não exigem um conhecimento aprofundado de eletrônica, sendo viável a sua manipulação por educadores e educandos em geral. Entretanto, embora se reconheça as vantagens da utilização destes *kits* para a implementação da Robótica nas escolas, o seu acesso não tem sido facilitado, vez que normalmente possuem um elevado custo. Uma alternativa encontrada é a utilização de *kits* compostos por sucatas de equipamentos eletrônicos em substituição aos produtos comerciais e a utilização de *software* livres.

Dentre os *kits* mais usados contemporaneamente para a aplicação da Robótica no contexto escolar, merecem destaque o LEGO *Mindstorms* e o VEX *Robotics*.

4.1. LEGO *Mindstorms*

O *Kit* Lego *Mindstorms*, ou melhor, o *Mindstorms* RIS (*Robotics Invention System*) foi desenvolvido pela Lego e lançado oficialmente em agosto de 1998. A Lego é uma empresa dinamarquesa, que existe desde 1942, cujo propósito original era o de desenvolver brinquedos de montar para crianças. Em 1980, a Lego criou um setor interno chamado de Lego *Educational Division* com a finalidade de desenvolver *kits* de Robótica destinados ao público escolar. Com isto, a Lego pretendeu tornar a tecnologia acessível e significativa aos seus usuários, preparando-os para serem capazes de investigar, criar e solucionar problemas (BAGNALL, 2007: P.2).

Em 2006, a Lego lançou um *kit* chamado de NXT, após anos de estudo. Anteriormente, desde 1996, ela comercializava o *kit* RCX. O novo *kit* trouxe ao universo da Robótica Educacional um grande impacto, favorecendo a

[5] Protótipos são produtos que ainda não foram comercializados, mas estão em fase de testes ou de planejamento, e, no caso específico, refere-se a projetos para o desenvolvimento de robôs (definição do autor).

montagem dos ambientes de aprendizagem (BAGNALL, 2007: p.16), tendo em vista a confiabilidade já conquistada pela Lego na fabricação de brinquedos educativos durante os longos anos de atuação no mercado.

O *hardware* da Lego se compõe de peças moldadas com plástico rígido, totalizando 577, dentre elas, baterias, saídas de áudio, portas de entrada e de saída, cabos, *displays*, *bluetooth* e USB, motores, sensores de luz, de toque, de som, sensor ultrassônico, bloco inteligente, rodas, garras, guindastes, que, usadas com o intuito educacional, podem se transformar em ágeis robôs, conforme a *Figura 3* (MINDSTORNS, 2010), dotados de capacidade para movimentar-se sozinhos, detectando o meio ambiente e desviando-se de obstáculos, executar tarefas, desde as mais simples às mais complexas, resolver problemas e decidir sobre o que fazer diante de um determinado problema (BAGNALL, 2007: p.15).

Figura 3 - Peças do *Kit* NXT

O monitoramento destes robôs é feito através de um minúsculo computador acoplado à sua estrutura, que, ao processar o programa operacional, age como um “cérebro”, controlando movimentos, através de informações obtidas pelos sensores.

O *Kit* NXT tem como software um ambiente de programação gráfica desenvolvida pela *LabVIEW*, conhecida como Linguagem NXT-G. Esta linguagem possui uma interface gráfica para o desenvolvimento de códigos, que são chamados de blocos, conforme a *Figura 4*. Cada um deles representa uma rotina

ou ação que poderá vir a ser executada pelo robô. O robô é programado a partir de uma sequência lógica de blocos (BAGNALL, 2007: p.16).

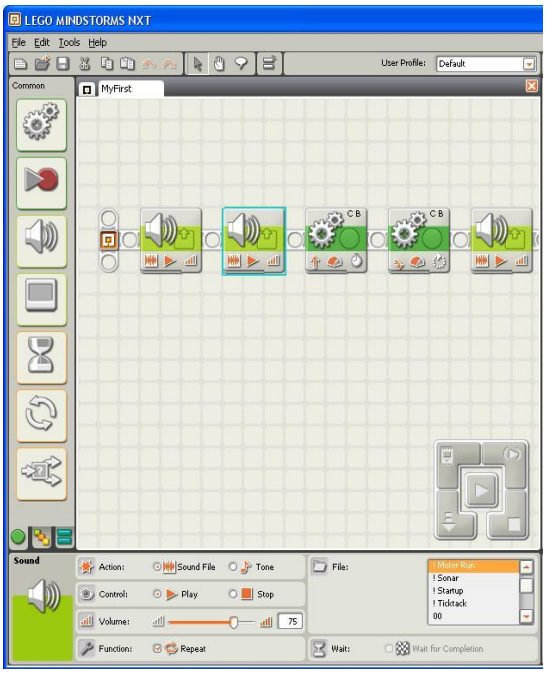


Figura 4 – Interface Gráfica

Segundo Bagnall (2007), é a versatilidade dos *Mindstorms* que os diferem dos demais brinquedos. Ele argumenta: “um brinquedo normal executa tarefas repetitivas, enquanto um *Mindstorms* sempre inova. É um brinquedo que nunca envelhece”.

Os *kits* criados pela Lego passaram a ser uma ferramenta de engenharia que propicia a implementação de ambientes de Robótica nas escolas e universidades, atendendo a uma diversificada faixa etária.

4.2. VEX *Robotics*

O *kit* da VEX foi desenvolvido pela empresa americana *Innovation First International* e distribuído por VEX *Robotics Inc., RackSolutions Inc. and Innovation First Labs Inc*, com o nome comercial de “Vex *Robotics Design System*”. A pretensão da *Innovation First* foi a de oferecer um *kit* educacional voltado ao Ensino Fundamental II e ao Ensino Médio. Ele possui vários pacotes, sendo o “VEX *Classroom Lab Kits*”, considerado o básico. Este subsidia a introdução dos conhecimentos elementares de Robótica, articulando conteúdos

de Física, Matemática e Programação na construção de protótipos robotizados (VEXROBOTICS, 2010).

O hardware do VEX constitui-se de quase 500 peças, entre rodas, placas metálicas, engrenagens, parafusos, porcas, motores, sensores, baterias, rádio controle, esteiras e eixos de vários tamanhos, conforme a *Figura 5,* sendo possível a aquisição de peças e sensores complementares caso se queira construir robôs mais possantes (VEXROBOTICS, 2010).

Figura 5 – *Kit Robotics Design System*

Como *software*, os ambientes de programação mais usados, oficialmente pela Vex *Design Robotics* são: o *EasyC* e o *RobotC*. O primeiro, *EasyC,* conforme a *Figura 6*, é a opção ideal para iniciantes em programação. Este ambiente aceita programação gráfica simultaneamente com a linguagem de código em C, o que facilita, ao usuário, o processo de criação de seu projeto. O segundo, *RobotC,* conforme a *Figura 7*, é considerado como uma solução intermediária. Possui um poderoso depurador, compilador e editor de código-fonte. Tem disponível o método avançado - MPLAB IDE o que possibilita programar diretamente o microcontrolador PIC que está embutido na plataforma Vex (ROBOSHOP, 2010b).

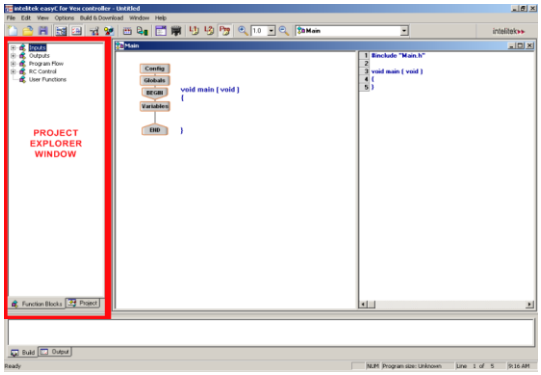


Figura 6 – Ambiente de Programação EasyC

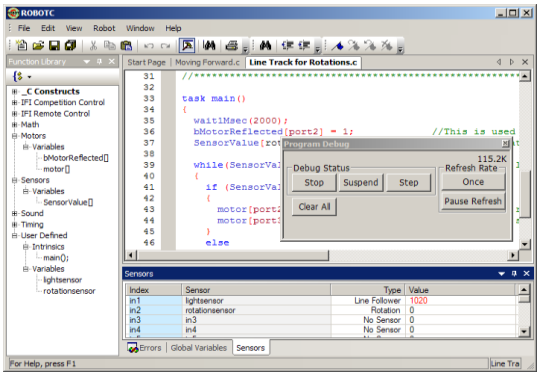


Figura 7 – Ambiente de Programação RoboticC

Capítulo II

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, 2010a).

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, 2010a).

1. Características da Plataforma Arduino

Arduino é uma plataforma livre e flexível, fácil de usar tanto o software quanto o hardware e, por isto, pode ser usada por aqueles que se interessam em criar objetos ou ambientes interativos (ARDUINO, 2010a).

A Plataforma Arduino apresenta algumas características que a distingue das demais. A mais importante diz respeito à concepção *open-source*[6],

[6] O termo *Open-Source* tem origem inglesa e significa Código Aberto, ou seja, *software* livre. A concepção *Open Source* determina que um programa de código aberto deve garantir: distribuição livre, acesso ao código fonte, criação de trabalhos derivados, integridade do autor do código fonte, não discriminação contra pessoas ou grupos ou áreas de atuação, distribuição da licença

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, que é o www.arduino.cc. (ARDUINO, 2010a). Atualmente este sítio se constitui como a grande referência acerca do Arduino para todo o mundo, conforme as *Figuras 8* (ARDUINO, 2010d) *e 9* (ROBOTSHOP, 2010c).

Figura 8 – Placa Arduino

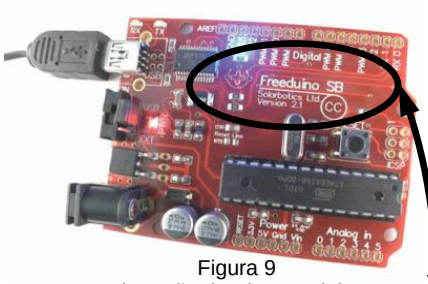


Figura 9
Alteração da Placa Arduino

Freeduino

Arduino

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, 2010c).

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, 2010c).

---

permitindo que os direitos associados ao programa possam ser aplicáveis para todos aqueles cujo programa é redistribuído, sem a necessidade da execução de uma licença adicional para estas partes (ARDUINO, 2010a).

2. *Hardware*

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. Ela 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 atuares (ARDUINO, 2010a).

2.1. Microcontrolador *Atmel* AVR

O microcontrolador usado na plataforma Arduino é da família AVR, produzido pela *Atmel,* empresa que possui grande participação no mercado de embarcados e tecnologias que envolvam semicondutores. Possui arquitetura de *Harvard*, com freqüência de operação em torno de 16MHz e a grande maioria de suas instruções é executada em um ciclo de clock, obtendo 1MIP[7] por MHZ. O AVR (*Advanced Virtual RISC*) tem foco na relação desempenho e consumo de energia (SCHUNK, 2001).

A plataforma Arduino utiliza os modelos *Atmega*168, *Atmega*328 e *Atmega*1280 de microntroladores, cujas características e tipos de memória serão descritos a seguir, conforme a *Tabela1* (ARDUINO, 2010e):

Tabela 1 - Plataforma Arduino

Modelos	Características Gerais						
	Flash	SRAM	EPROM	Clock	ADC	Pinos	Recursos auxiliares
Atmega168	16KB	1KB	512bytes	20Mhz	10bit	14	6 PWM SPI RS232 I^2C
Atmega328	32KB	2KB	1KB	20Mhz	14bit	14	6 PWM SPI RS232 I^2C
Atmega1280	128KB	8KB	8KB	16Mhz	54bit	54	14 PWM SPI RS232 I^2C

[7] MIP – milhões de instrução por segundos

2.2. Placa Física

Ela possui duas versões: Arduino *Duemilanove* e Arduino *Mega,* conforme as *Figuras 10* e *11* (SOLARBOTICS, 2010), respectivamente. Distinguem-se entre si apenas pela quantidade de memória, quantidade de pinos de entrada e saída (analógicos e digitais) e pela dimensão física de suas placas, sendo a versão mais usada nos protótipos, a *Duemilanove.*

Figura 10
Arduino *Duemilanove*

Figura 11
Arduino *Mega*

O *Duemilanove* Arduino é uma placa baseada no microcontrolador *ATmega*168 ou *ATmega*328. Tem 14 pinos de entrada e saída digital (dos quais 6 podem ser utilizados como saídas PWM), 6 entradas analógicas, 16 MHz oscilador de cristal, uma conexão USB, um conector de alimentação, um cabeçalho ICSP e um botão de *reset* (ARDUINO, 2010d). Ele contém todo o aparato necessário para que se possa programar sem dificuldades de configurações, conforme a *Tabela 2* (ARDUINO, 2010d).

Tabela 2: Especificações do Microcontrolador ATmega168

Tensão	5V
Tensão de entrada (recomendado)	7- 12V
Tensão de entrada (limites)	6- 20V
Digital I / O Pinos	14 (dos quais 6 oferecem saída PWM)
Analógico Input Pinos	6
Memória Flash	16 KB (ATmega168) Ou 32 KB (ATmega328) , dos quais 2 KB utilizado pelo bootloader
SRAM	1 KB (ATmega168) Ou 2 (KBATmega328)
EEPROM	512 bytes (ATmega168) Ou 1 KB (ATmega328)
A velocidade do clock	16 MHz

Alimentação do Sistema - A fonte de energia da plataforma Arduino é selecionada automaticamente, podendo ser através de conexão USB ou com fonte de alimentação externa. Se for externa não haverá conexão com o computador, podendo ser uma fonte de corrente contínua ou com 4 baterias AA. Pode também operar com uma fonte externa de 6 a 20 *volts*. Se fornecido menos de 7V, o circuito pode se tornar instável. Se usado mais do que 12V, o regulador de voltagem pode superaquecer e danificar a placa. Por isto a faixa recomendada é de 7-12 *volts* (ARDUINO, 2010a).

Memória - A plataforma Arduino dispõe de dois modelos de microcontroladores: o *ATmega*168 e o *ATmega*328. O primeiro tem 16 KB de memória *flash* para armazenar o código (*bootloader* - de 2 KB que é usado para o gerenciador de inicialização), 1 KB de SRAM e 512 *bytes* de EEPROM, enquanto que o segundo possui 32 KB (também com 2 KB utilizado para o *bootloader),* 2 KB de SRAM e 1 KB de EEPROM. Os dois modelos podem ser usados na placa do *Duemilanove* (ARDUINO, 2010a).

Entrada e Saída - No Arduino as entradas e saídas tem funções semelhantes para o *Duemilanove* e o *Mega*, diferenciando apenas pela quantidade de pinos de cada um. O primeiro possui 14 pinos dos quais 6 provém saída PWM e, o segundo, possui 54 pinos sendo 14 PWM e 16 analógicos. Cada um dos pinos, tanto digital como analógico, podem ser usados como entrada ou saída, bastando apenas usar os comandos para configurar. Eles operam em 5 volts. Cada pino pode fornecer ou receber um máximo de 40 mA. O *Duemilanove* tem 6 entradas analógicas e cada uma fornece 10 *bits* de resolução (ou seja, 1.024 valores diferentes) (ARDUINO, 2010a).

Comunicação Serial - A comunicação serial possibilita o Arduino se comunicar com um computador ou com outros dispositivos. Todas as placas Arduino possuem pelo menos uma porta serial (também conhecida como UART ou USART): **Serial**. A comunicação ocorre através dos pinos digitais 0 (RX) e 1 (TX), assim como uma conexão USB. Desta maneira, caso se utilize esta

funcionalidade, os pinos 0 e 1 não poderão ser utilizados como entrada ou saída digital (ARDUINO, 2010a).

O *ATmega*168 e *ATmega*328 fornecem UART TTL (5V) de comunicação serial, que está disponível nos pinos digitais 0 (RX) e 1 (TX). Há na placa um circuito eletrônico, chamado de FTDI FT232RL, que possibilita a comunicação com o PC através da porta USB, conforme a *Figura 12*.

Alguns pinos têm funções específicas:

- **Interrupções Externas:** 2 e 3. Estes pinos podem ser configurados para disparar uma interrupção em um valor baixo, um aumento ou diminuição de ponta, ou uma alteração no valor.

- **PWM** : 3, 5 , 6, 9 , 10 e 11. Assegurar a saída PWM de 8 *bits* (ARDUINO, 2010a).

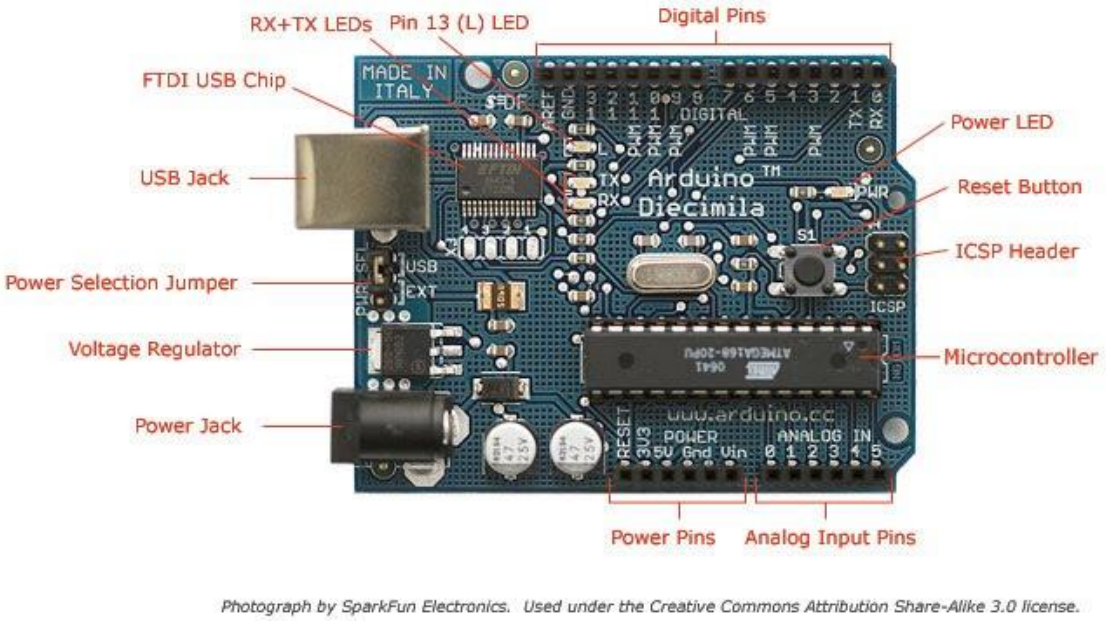


Figura 12 – Diagrama dos componentes do Arduino

3. *Software*

As ferramentas que dão suporte a programação a 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.

3.1. Diagrama do Ciclo de Desenvolvimento

Entende-se por ciclo de desenvolvimento a síntese dos passos efetuados para a instalação, configuração e utilização, após a instalação do *software*. Ele pode ser representado pelo seguinte diagrama, conforme a *Figura 13*.

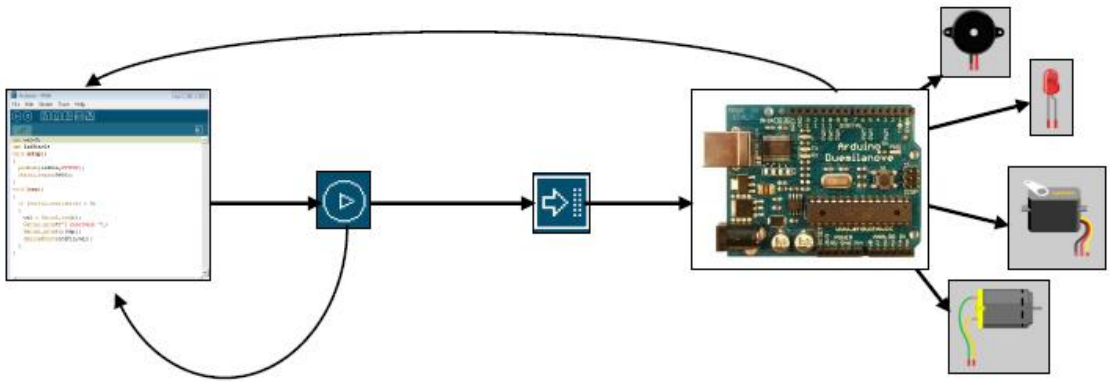


Figura 13 – Diagrama do Ciclo de Desenvolvimento

3.2. Ambiente de Desenvolvimento

No ambiente de desenvolvimento da plataforma Arduino são disponibilizadas as seguintes ferramentas: editor de texto, compilador e gravador de programas. Ele é composto pela Barra de *Menu* e Barra de Ferramentas, conforme a *Figura 14*. O primeiro diz respeito às funcionalidades disponíveis e, o

segundo, refere-se aos ícones localizados abaixo da Barra de *Menu* que permite ao usuário um acesso rápido às principais funções do programa.

Figura 14 – Ambiente de desenvolvimento da plataforma Arduino

Além das funcionalidades comuns a qualquer ambiente de programação, a Barra de Ferramentas da plataforma Arduino oferece outras funcionalidades específicas para microcontroladores (*tools*) que são: escolher o modelo da placa (*board*), configurar a porta para que o sistema operacional se comunique com a placa (serial *port*) e uma janela que permite mostrar a troca de comunicação entre o computador e a placa (serial *menu*), conforme a *Figura 15*.

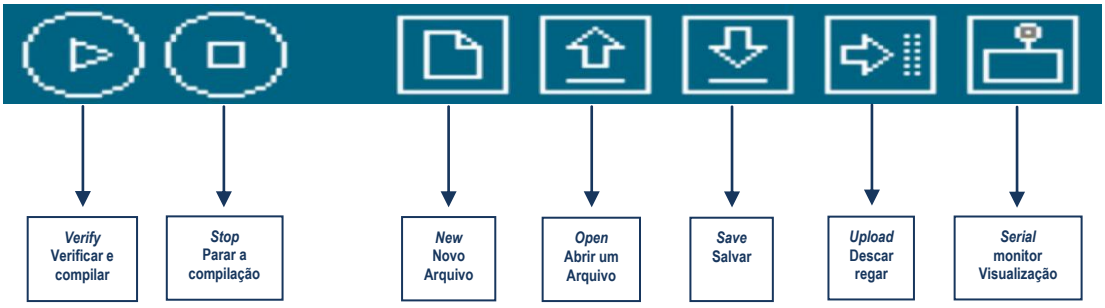


Figura 15 – Barra de Ferramentas da plataforma Arduino

3.3. Características da Linguagem

Para que tenha uma maior funcionalidade, o microprocessador da plataforma Arduino necessita de uma linguagem de programação que disponha de características, tais como: portabilidade, modularidade, recursos de baixo nível, geração de código eficiente, simplicidade, facilidade de uso. Estas características estão presentes na Linguagem de Alto Nível, como a C, por exemplo, a qual a plataforma Arduino tem como base. Além disto, a linguagem C "permite uma grande velocidade na criação de novos projetos devido às facilidades de programação e também à sua portabilidade, o que permite adaptar programas de um sistema operacional com um mínimo de esforço" (PEREIRA, 2003, p.18). Há de se considerar ainda que a linguagem C possui a eficiência ideal para gerar códigos menores e mais rápidos.

3.3.1. Funções Básicas da Linguagem

A plataforma de programação Arduino, além de se beneficiar das funcionalidades da linguagem C, introduziu diversas funções e bibliotecas específicas para interagir com as funcionalidades dos seus microcontroladores e outros dispositivos de hardware, como: motores, sensores e memórias externas etc (ARDUINO, 2010b).

***void setup*() -** Esta função é executada somente uma vez e é normalmente utilizada para inicializar variáveis, utilização de bibliotecas, definição dos pinos como entrada ou saída, início do uso de comunicação série, entre outros. Esta função volta a ser executada novamente quando se desliga a energia ou reinicializa-se a placa (*reset*).

***void loop*() -** Esta função faz um *"loop"* sucessivo, ou seja, todos os comandos existentes no interior desta função são sucessivamente repetidos, o que pode permitir a leitura sucessiva de portas, a leitura do estado dos sensores externos e a movimentação dos atuadores de acordo com a lógica do algoritmo, entre muitas outras aplicações. Ela dá a dinâmica a toda programação do sistema, colocando todo o código que envolve esta função em um ciclo infinito.

Os procedimentos “*void setup*()” e “*void loop*()” são de caráter obrigatório, ou seja, mesmo que a sua utilização não seja necessária, deverá constar no código utilizado. E apenas serão chamadas funções externas as que constem na função “void loop()”.

3.3.2. Outras Funções

***Input/Output digital - pinMode*() -** Ao recorrer a esta instrução, é possível configurar o modo de comportamento de um determinado pino, possibilitando assim defini-lo como entrada (*input)* ou saída (*output)*. Esta definição normalmente é efetuada dentro da função “*void setup*()”.

Sintaxe:

pinMode (Número do pino, Modo);

O “Modo” acima descrito pode ser definido da seguinte forma:

“INPUT”

“OUTPUT”

***digitalWrite*() –** esta função possibilita que os pinos configurados como *output*, através da instrução “*pinMode*”, estabeleça a saída dos respectivos pinos com o valor lógico 1 (*HIGH* – 5 V) ou com o valor lógico 0 (*LOW* – 0V)

Sintaxe:

DigitalWrite (Número do pino, Modo);

O “Modo” acima descrito, pode ser definido como:

“HIGH”

“LOW”

***digitalRead*() -** Possibilita a leitura de uma entrada digital específica, retornando um valor do tipo inteiro. Se receber um valor de retorno igual a “1”, apresenta uma leitura do tipo “*HIGH*” (valor lógico 1). Se for o contrário, sendo o valor de retorno igual a “0”, apresenta leitura do tipo “*LOW*” (valor lógico 0).

Sintaxe:

Variável do tipo *integer* = *digitalRead*(Número do pino);

***analogRead*() -** Possibilita a leitura do valor analógico do pino especificado, com um conversor A/D, possuindo uma resolução de 10 *bits*. Isto faz com que um valor compreendido entre 0V e 5 V, esteja entre os valores inteiros (int) 0 e 1023.

Sintaxe:

Variável do tipo *integer* = *analogRead*(Número do pino);

***analogWrite*() -** Possibilita a utilização dos pinos PWM (*Pulse Width Modulation*) da placa Arduino, sendo mantido até que uma outra instrução o modifique.

Sintaxe:

analogWrite(Número do pino, valor);

O "valor" referido anteriormente varia entre 0 (sempre desligado), até ao valor 255 (que representa um sinal de 5 V constante).

***Time - millis*() -** Possibilita o retorno da quantidade de tempo, em milisegundos na forma de uma variável do tipo "*unsigned long*". O valor retornado representa o tempo que passou desde que o programa atual começou a ser executado. O *overflow* (voltar ao início, valor zero) do contador ocorre passado um tempo de ≅ 50 dias.

Sintaxe:

unsigned long tempo = millis();

***micros*() -** Possibilita o retorno da quantidade de tempo em microsegundos na forma de uma variável do tipo "*unsigned long*". O valor retornado representa o tempo que passou desde que o programa atual começou a ser executado. O *overflow* (voltar ao início, valor zero) do contador ocorre passado um tempo de ≅ 70 minutos.

Sintaxe:

unsigned long tempo = micros();

***delay* (milisegundos) -** Possibilita efetuar uma pausa ao programa em execução, por uma quantidade de milisegundos especificada. Útil para manter um estado durante uma certa quantidade de tempo.

Sintaxe:

delay(tempo que deseja efetuar a pausa – ms);

delayMicroseconds **(microsegundos) -** Possibilita efetuar uma pausa ao programa em execução, por uma quantidade de microsegundos especificada.

Sintaxe:

delayMicroseconds(tempo que deseja efetuar a pausa – microsegundos);

Capítulo III

Robô Virgulino

O robô desenvolvido neste trabalho foi denominado como robô Virgulino, sendo definido como um dispositivo robótico autônomo, terrestre, móvel, que se locomove através de rodas.

Ele é apresentado neste capítulo como uma opção viável de kit educacional de baixo custo e com características open-source, direcionado ao ensino da Robótica Educacional, podendo ser considerado como uma alternativa, principalmente para as escolas públicas do Ensino Fundamental e Médio.

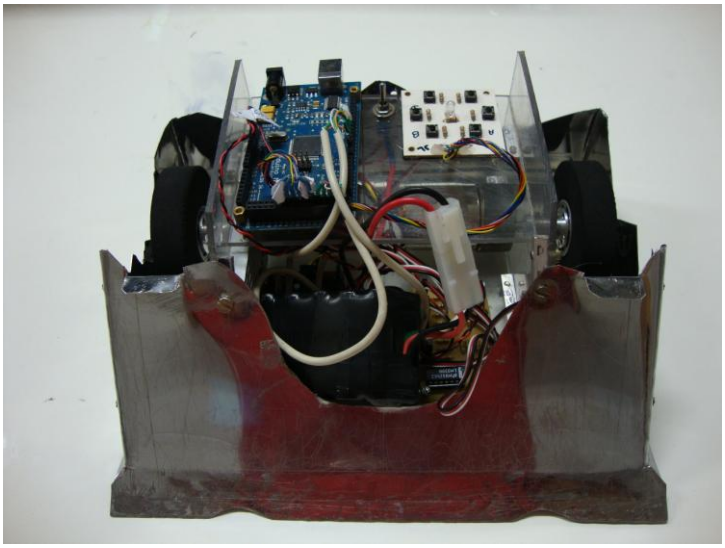
Para o desenvolvimento do projeto, alguns requisitos[8] foram definidos, de acordo a descrição abaixo:

- Ser um robô móvel;
- Ter capacidade de se locomover com velocidade máxima de 80 cm por segundo.
- Possuir rodas com tração suficiente para se locomover em diversas superfícies;
- Ser capaz de empurrar 3 kilogramas;
- Suportar choques mecânicos, sendo capaz de resistir a impactos sem se danificar;
- Não possuir ligações físicas com outros dispositivos ou objetos como fios ou cabos, sendo totalmente autônomo;
- Ser capaz de obter informações a respeito do ambiente;
- Ser capaz de repetir seqüências de comandos e permitir criação de novos comandos;
- Possuir interface de comunicação para comandos e carregamento de programas mediante operações simplificadas;
- Possuir o menor peso possível para simplificação da dinâmica, aumento da autonomia energética e diminuição de custos com atuadores;

[8] Requisitos são aqui definidos como uma condição ou uma capacidade com o qual o sistema deve estar de acordo.

- Possuir rigidez suficiente para manter suas placas de circuito impresso de forma estática;
- Ter autonomia de funcionamento de pelo menos uma hora;
- Possuir baixo custo para produção;
- Ser destinado ao uso educacional direcionado aos educandos com faixa etária entre 12 a 17.

1. Dimensões do Robô Virgulino



As dimensões do robô Virgulino são (veja a *Figura 16*):

- Comprimento: 21 cm
- Largura: 26 cm
- Altura: 10,2 cm
- Massa: 1,450 kg

Figura 16 - Robô Virgulino

2. Composição do Robô Virgulino

Conforme já mencionado no primeiro capítulo, um robô se compõe de dois módulos essenciais que mantém estreita relação entre si. Deste modo, o Robô Virgulino tem a seguinte composição: os projetos físico e eletrônico (*hardware*), e o programa que simula luta de sumô[9] (*software*).

2.1. *Hardware*

Para melhor organização didática, a descrição do hardware do Virgulino será feita, como aludido anteriormente, a partir dos projetos físico e

[9] Sumô, segundo o Dicionário Aurélio, "é uma luta tradicional japonesa, em que um dos contendores perde quando é empurrado para fora do ringue ou quando qualquer parte do seu corpo, exceto as solas dos pés, toca o chão" (AURÉLIO, 2010).

eletrônico. No primeiro será abordado sobre suas partes interna e externa, sistema de direção, sistema de locomoção e sistema de alimentação, e, no segundo, serão apresentadas suas placas: Arduino, de controle dos motores, de sensor de luz e de interface.

2.1.1. Projeto Físico

O sistema físico do robô Virgulino, além de alojar todos os componentes e determinar o esqueleto do dispositivo, possui características que habilitam sua interação com o ambiente no qual está inserido. Sua estrutura física é dotada de dois tipos de matérias distintos que formam a parte interna e a externa, usando diferentes métodos para o seu manuseio.

Parte Interna

Para a constituição da parte interna do robô Virgulo foi usado um material chamado policarbonato. Este material foi escolhido por ter alta resistência, ser moldável quando aquecido e possuir várias características que facilitam sua usinagem, podendo ser cortado com serra manual, perfurado com facilidade, desbastado sem formar fissuras. Nos dias atuais, ele é frequentemente utilizado para fabricar viseiras de capacetes de fórmula 1 e mídias de DVDs.

Cabe ressaltar que a placa de policarbonato utilizada neste projeto foi doada por uma empresa de publicidade que fabrica placas luminosas de identificação, o que contribui ainda mais para minimizar seus custos.

A montagem da parte interna do Virgulino obedeceu às seguintes etapas:

1º. Etapa - Foi construído um molde de papelão do que seriam as placas de policarbonato para melhor visualização do projeto, conforme a *Figura 17*.

Figura 17 – Molde das placas

2º. Etapa - As peças foram cortadas seguindo os moldes de papelão com utilização de serra manual, conforme a *Figura 18*.

Figura 18 – Corte das placas

3º. Etapa - As placas foram perfuradas com furadeira elétrica para receberem os parafusos a fim de dar sustentação á carroceria, os motores, as placas eletrônicas e fixar a parte externa, conforme a *Figura 19*.

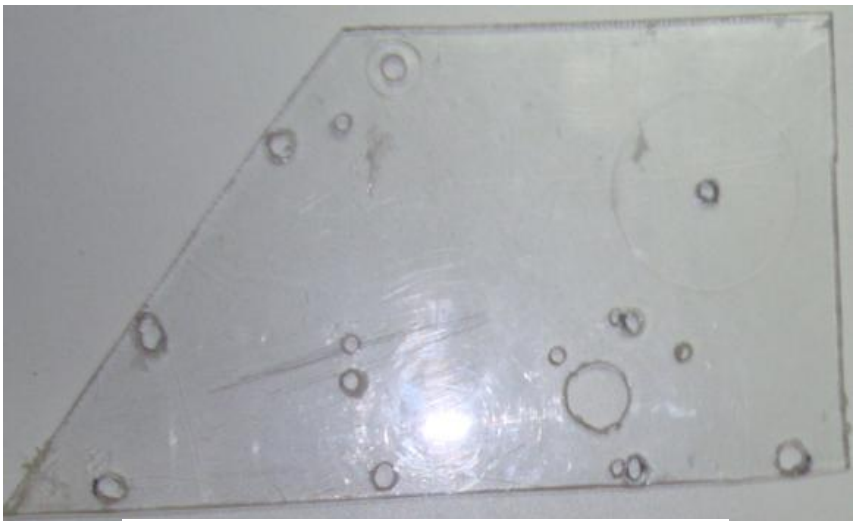


Figura 19 – Placa lateral

4ª. Etapa - A solução encontrada para unir as placas foi utilizar barras de alumínio que foram adquiridas em lojas de esquadria de alumínio. Estas barras podem ser perfuradas para receber parafusos, conforme as *Figuras 20*.

Figura 20 – Placas de alumínio

A parte interna do robô Virgulino ficou com boa sustentação e pronta para receber as placas externas, os pneus e a parte elétrica, conforme a *Figura 21*.

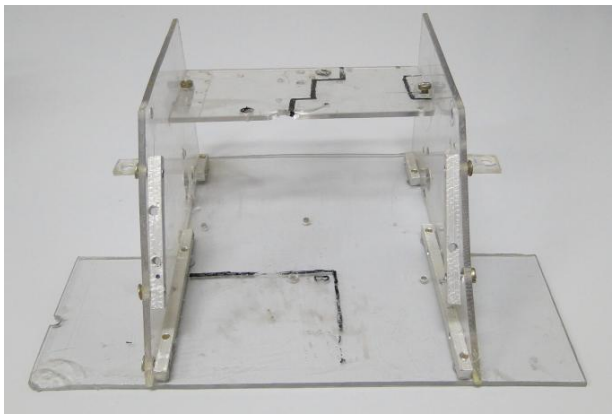


Figura 21 – Placa Interna

Parte Externa

Para estrutura externa do robô Virgulino foi escolhido o aço inox, devido ao fato dele possuir resistência mecânica, resistência a corrosão e uma boa usinagem, propriedades essenciais para formar a sua armadura, conforme a *Figura 22*.

Figura 22 – Parte Externa

Observou-se que o material escolhido para a montagem da parte externa, o aço inoxidável, de fato, possibilitou uma estrutura resistente, com adequada sustentação, pesando 1,4 kilogramas. Na parte frontal foi feita uma inclinação de 75 graus visando facilitar o movimento de levantar os oponentes e empurrá-los para fora do ringue.

Sistema de Direção

O sistema de direção escolhido foi o do tipo tanque. Nele todo o lado esquerdo é acionando de forma independente do lado direito. E, para que o robô pudesse andar em linha reta, foi necessário programar ambos os lados para terem a mesma velocidade. Para se conseguir a execução desta tarefa complexa, o robô Virgulino foi projetado para que as curvas a serem executadas variassem as velocidades dos motores, ou seja, pretendendo direcionar-se para a esquerda, foi acionada mais velocidade neste lado e, querendo ir para o lado oposto, foi colocada mais velocidade do lado direito. Para dar um giro no próprio eixo, projetou-se o Virgulino para executar velocidades contrárias nos motores, mas de mesma intensidade, conforme a *Figura 23.*

Figura 23 – Sistema de Direção

Sistema de Locomoção

O sistema de locomoção do robô Virgulino foi composto por atuadores, rodas e pneus, conforme especificações a seguir:

Atuadores - Foram definidos dois atuadores de corrente contínua, com tensão - 12V, 200 RPM (rotações por minuto), torque de 620g.cm, tamanho do eixo 6mm,

amperagem mínima de 113mA, amperagem máxima de 223mA, caixa de redução 30:1 e peso de 152 gramas.

Os dois motores DC possibilitaram uma velocidade de aproximadamente 82cm/s, sendo capaz de deslocar objetos de até 2,5kg. Sua caixa de redução foi formada por engrenagens de metal fornecendo ao eixo uma viável relação entre velocidade e torque.

Embora o peso do sistema de locomoção tenha sido responsável por 25% do peso total do protótipo desenvolvido, sendo, portanto, um aspecto negativo, manteve-se esta opção devido a força e resistência alcançadas, já que os atuadores resistiram a 25 horas de teste sem apresentar nenhum superaquecimento, nem sintomas de fadiga. Eles foram obtidos em sucata de oficinas elétricas de automóveis (BRAGA, 2005).

Rodas e pneus - As rodas utilizadas possuem plástico resistente montada em um pneu feito de espuma de borracha que garante estabilidade em vários tipos de piso. Os pneus medem 8 cm de diâmetro e 3 cm de espessura, o que oferece boa aderência e boa tração.

Para fixar as rodas ao eixo do motor foi necessário utilizar um componente chamado de flange. Este foi fixado com parafusos sem cabeça. O conjunto destes componentes foi adquirido em sites de automodelismo, conforme as *Figuras 24 e 25.*

Figura 24 - Rodas, Pneus, Parafusos e Flange

Figura 25 - Rodas e Pneus fixados ao eixo do motor

Sistema de Alimentação

A fonte de alimentação usada no Virgulino foram baterias de Níquel-hidreto metálico (veja a *Figura 26*) devido à maior relação custo benefício. Isto porque elas são leves, com peso de 100g, fornecem energia suficiente para vários testes seguidos, não oferecem risco de manuseio, são degradáveis, recarregáveis, podendo ser carregadas em no máximo 02 horas e são de fácil aquisição. O conjunto utilizado no protótipo possui 6 baterias, envolvidas em plástico resistente e fornecem 12V e 1,6Ah (*Ampère* Hora).

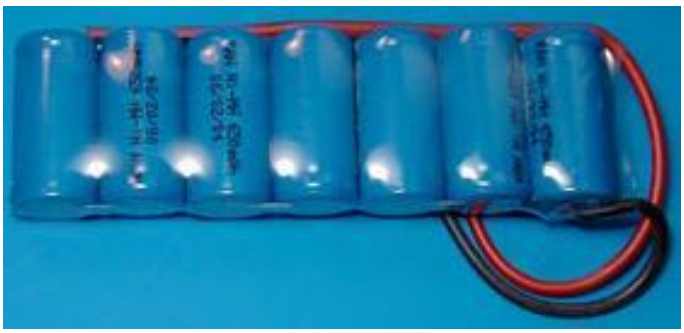


Figura 26 – Baterias de Níquel-hidreto metálico

2.1.2. Projeto Eletrônico

Antes de desenvolver o projeto eletrônico, fez-se necessário estabelecer alguns requisitos para que tal projeto pudesse atender a contento às particularidades do robô Virgulino.

Os requisitos exigidos levaram em conta:

- Facilidade de aprendizado;
- Material de apoio pedagógico;
- Facilidade de aquisição no mercado;
- Custo do "*Kit* de Desenvolvimento";
- Número de I/Os necessários para o projeto;
- Quantidade de memória;
- Versatilidade da plataforma;
- "*Hardware*" interno disponível, como àqueles que possuem uma UART (*Universal Assyncrhonous Receiver and Transmiter*) interna para rotinas de comunicação serial;
- Custo final do produto.

Considerando tais exigências, após uma exaustiva busca, escolheu-se a Plataforma Arduino, sendo a placa Arduino *Mega* a mais adequada devido às suas especificações técnicas, principalmente devido à quantidade de pinos de entrada e saída, conforme a *Figura 27* (ARDUINO, 2010e). Esta placa é desenvolvida com base no *ATmega*1280 da empresa *Atmel*, possuindo 54 pinos de entrada digital de saída, dos quais 14 podem ser utilizados como saídas PWM, 16 entradas analógicas, 4 UARTs (portas seriais de *hardware*), 16 MHz Oscilador de cristal, 128Kb de memória *flash*, 8Kb de SRAM, 4Kb de EEPROM, uma conexão USB para fazer a troca de dados e alimentação e um plugue para fazer a alimentação externa.

Figura 27 - Placa Arduino *Mega*

Placa de Controle dos Motores

A placa utilizada foi a de Motores DC, fabricada pela empresa *Solarbotics*. Ela possui características de um acionador robusto, com dimensões reduzidas e baixo custo. O circuito integrado principal que constitui a ponte H chama-se L298 da *National Semiconductor* que pode receber tensões contínuas de 6V a 26V, podendo operar com motores DC dos mais variados tipos e com carga de consumo entre 0,3 a 4A de consumo. O usuário pode controlar o sentido de rotação dos motores através do uso de chaves ou computadores ou qualquer equipamento capaz de gerar os níveis lógicos de entrada, conforme as *Tabelas 3* e *4*).

TABELA 3 – Tabela Lógica de Controle do Motor A			
Pino	Pino L1	Pino L2	Estado
PWM	0	0	Parado
PWM	0	1	Frente
PWM	1	0	Recuar
PWM	1	1	Parado

TABELA 4 - Tabela Lógica de Controle do Motor B			
Pino	Pino L3	Pino L4	Estado
PWM	0	0	Parado
PWM	0	1	Frente
PWM	1	0	Recuar
PWM	1	1	Parado

O circuito foi montado em uma placa compacta de circuito impresso de dupla face. Foi usado ainda um conjunto de soquetes para: interligação dos motores, conexão de baterias e barra de pinos para interligação do circuito de controle. Utilizou-se também um regulador de tensão cujo modelo foi o LM237, fornecendo tensão de 5V, conforme a *Figura 28.*

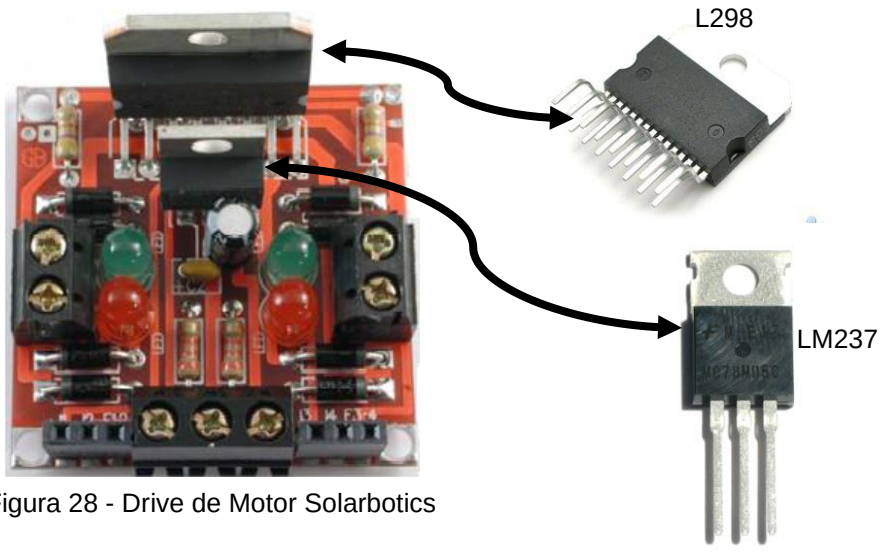


Figura 28 - Drive de Motor Solarbotics

Sensor de Luz

Foram utilizados, no robô Virgulino, 4 sensores de luz, conforme a *Figura 29,* fixados um em cada canto de sua estrutura interna para que fosse possível fazer o monitoramento da parte frontal e traseira do agente de sumô.

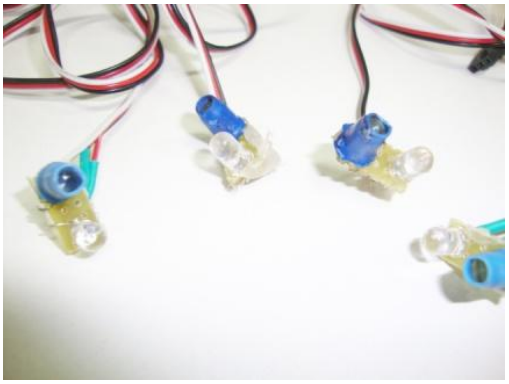
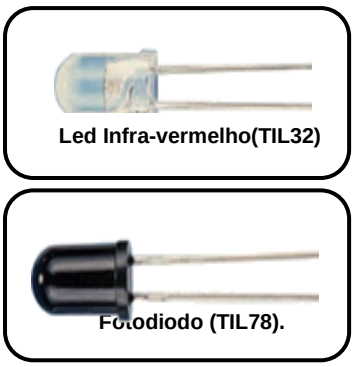


Figura 29 - Sensores do robô Virgulino

Escolheu-se o fototransistor, atuando como foto-receptor do tipo TIL78, devido sua velocidade de resposta e seu baixo consumo de energia. Como foto-emissor utilizou-se o componente TIL32 que emite luz infra-vermelho a fim de estimular a base do fototransistor. Assim, quanto maior a incidência de luz na base deste sensor de luz, maior a corrente reversamente polarizada, conforme a *Figura 30*. O componente receptor, o TIL78, é composto de silício e apresenta as seguintes especificações técnicas:

• Comprimento de onda da luz para ganho máximo: 890 nm;
• Corrente máxima causada pela luz: 28,5μA;
• Corrente mínima (ausência de luz): igual a 60 nA;
• Tempo de resposta de 5 ns;
• Potência máxima dissipada: 100 mW;
• Ângulo de aceitação de luz: ±60º;
• Temperatura de operação: -40ºC a +100ºC.



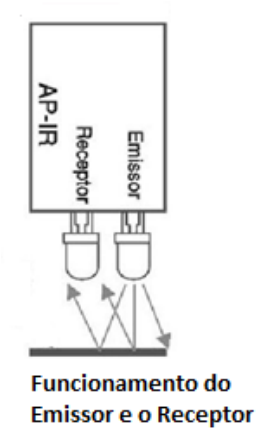


Figura 30 -
Led Infra-vermelho(TIL32)
Fotodiodo (TIL78)

Tendo em vista de que fototransistores não possuem tensão alta que seja capaz de acionar a entrada de um microcontrolador, no caso do Virgulino, foi necessário adicionar um amplificador para filtrar e amplificar o sinal emanado pelo fotodiodo. Neste caso, para solucionar a limitação do fototransistor, optou-se por um amplificador operacional típico LM339 (veja a *Figura 31*). Este componente eletrônico funcionou no circuito receptor do sensor, operando como um comparador de tensão simples, já que o potenciômetro tem a função de ajustar a sensibilidade do sensor, conforme a *Figura 32*.

Com esta adaptação foi possível minimizar a interferência da luz ambiente, configurando a calibração com o padrão de cores e obtendo respostas favoráveis como: estabilidade dos sinais tanto no claro quanto no escuro, distância de operação considerada adequada para manter o agente sempre na arena sem ter problemas com a iluminação e calibração do sistema de acordo com as condições de iluminação apresentadas momentos antes do combate, conforme a *Figura 33*.

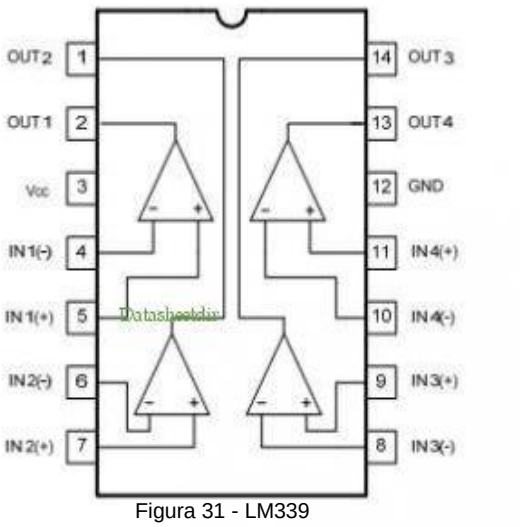


Figura 31 - LM339

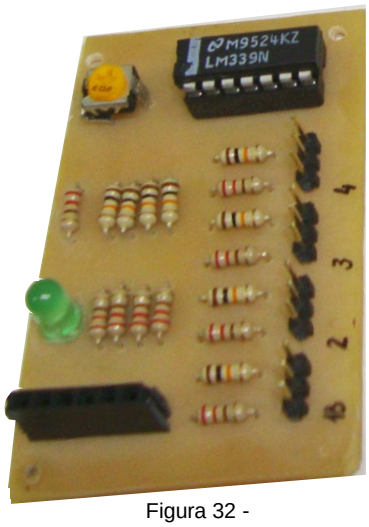


Figura 32 - Placa Impressa do Sistema de Sensores do Robô Virgulino

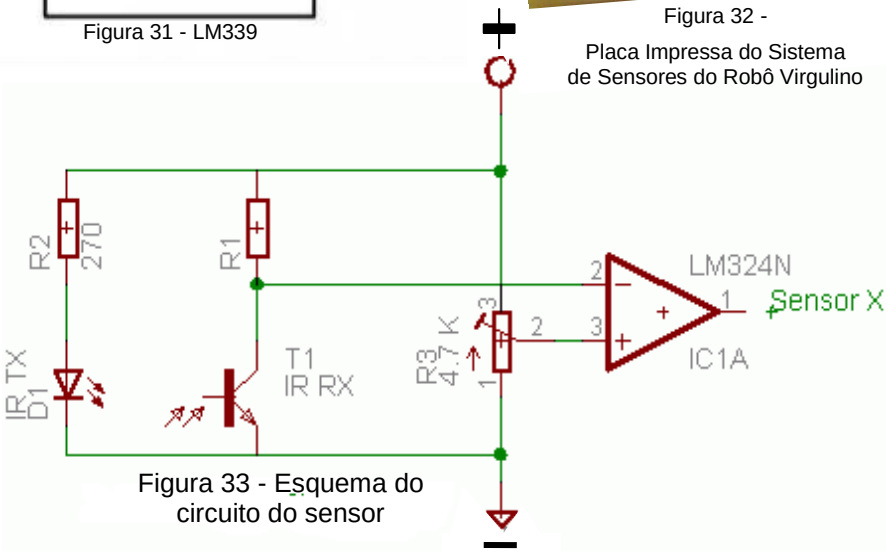


Figura 33 - Esquema do circuito do sensor

Placa de Interface

Segundo FORESTI (2006: p.52), dispositivos robóticos devem possuir uma interface para interagir como o sistema de processamento a fim de receber comandos de um operador com o objetivo de executar as missões predefinidas em um algoritmo. De acordo com o nível de complexidade da programação, os agentes robóticos podem executar diferentes missões. A partir destas referências, o robô Virgulino foi projetado para variar suas ações conforme a sua interação com o painel de botões, contidos na sua placa de interface, conforme as *Figuras 34 e 35*, ou seja, ao alimentar o sistema de controle, ele executa um algoritmo, na expectativa que uns dos cinco botões sejam acionados para executar as tarefas que estão atreladas a estes botões. Funciona da seguinte forma: ao pressionar o botão A, o robô executa a ação de ir para frente até o sensor de luz encontrar a borda de uma arena, por exemplo. Já o botão B, simplesmente vira 90 graus para esquerda. Objetivamente, a funcionalidade do sistema de interface, no tocante a facilidade de interação, e a variação das tarefas possíveis de serem efetuadas possibilitaram ao Virgulino uma satisfatória interface entre homem e máquina.

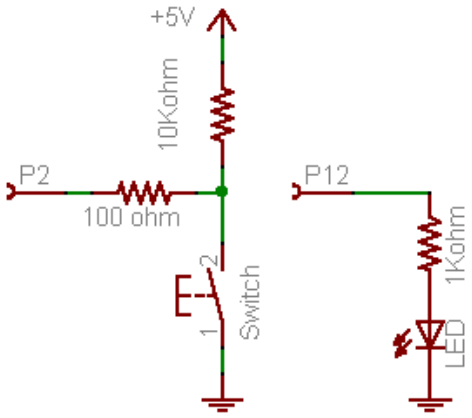


Figura 34
Diagrama do circuito eletrônico

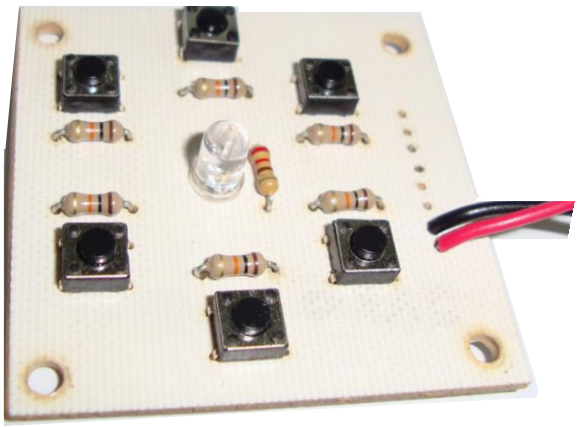


Figura 35 - Placa de interface do Virgulino

2.2. *Software*

O robô Virgulino foi desenvolvido para participar da OBR (Olímpiada Brasileira de Robô) de 2009 que teve como desafio o sumô de robôs[10].

Considerando a simplicidade exigida para o comportamento de um robô em luta de sumô, a programação feita para o robô Virgulino se baseou nas seguintes tarefas: detectar a claridade da superfície, girar os motores no sentido horário e anti-horário, variar as velocidades dos dois motores, parando um e girando o outro para formar ângulos variados. No caso específico do Virgulino, sua programação foi modulada para executar quatro procedimentos a partir do acionamento dos botões localizados na parte de cima do robô.

O fluxograma abaixo, conforme a *Figura 36,* exprime de forma sucinta o comportamento autônomo do Virgulino.

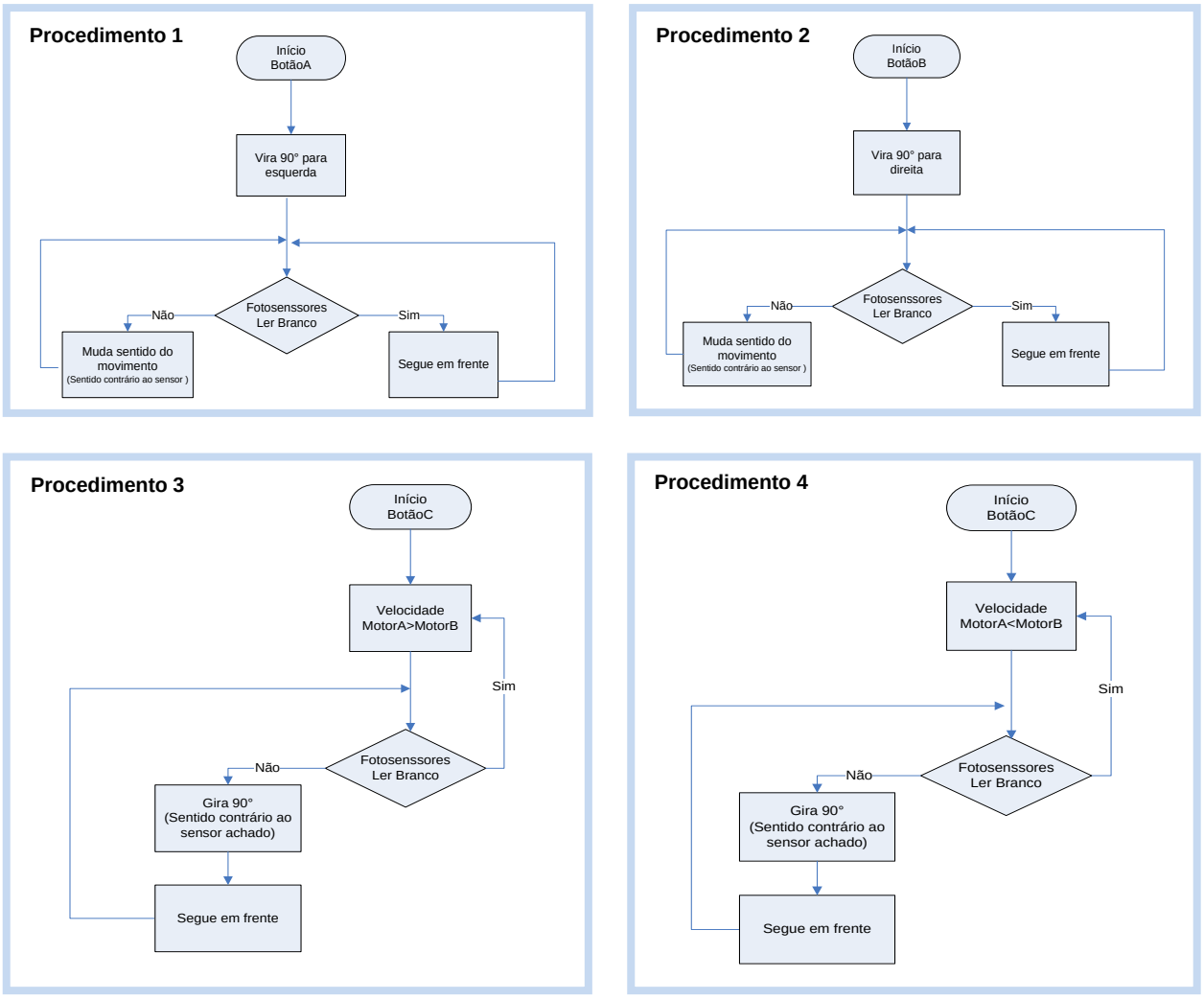


Figura 36 – Fluxograma do Robô Virgulino

[10] **Sumô de robôs** "é um modelo padrão para o desenvolvimento de robôs autônomos direcionado à iniciação de estudantes na robótica (OBR, 2010b).

Capítulo IV

Testes e Resultados

Como já exposto no capítulo anterior, o robô desenvolvido neste trabalho teve por finalidade a participação na Olímpíada Brasileira de Robótica de 2009.

1. Olimpíada Brasileira de Robótica - OBR

A OBR é uma iniciativa pública, gratuita e sem fins lucrativos, totalmente dedicada às escolas, professores e jovens brasileiros (com ou sem conhecimento prévio em robótica) vinculados ao ensino fundamental, médio ou técnico. Foi criada em 2007 por pesquisadores e professores de várias universidades, como: ITA, UFRN, UFRGS, FURG, UNESP e FEI. Tem como matriz de referência os seguintes eixos cognitivos: I. Domínio das linguagens/DL, II. Compreensão dos fenômenos/CF, III. Enfrentamento de situações-problema/SP, IV. Construção de argumentação/CA, V. Elaboração de propostas/EP (OBR, 2010a).

Ela tem por objetivos: despertar e estimular o interesse pela Robótica, áreas afins e a Ciência em geral; promover a difusão de conhecimentos básicos sobre Robótica de forma lúdica e cooperativa; promover a introdução da robótica nas escolas de ensino médio e fundamental; proporcionar desafios aos estudantes; aproximar a universidade das escolas; identificar talentos e vocações em Robótica; colaborar para o desenvolvimento e aperfeiçoamento dos professores. A cada ano, ela propõe desafios aos participantes. Em 2009, teve como desafio, na modalidade prática, um torneio de sumô de robôs (OBR, 2010b).

Segundo o regulamento da modalidade prática, nível 1, 2009, o objetivo da competição "Sumô de Robôs[11]" foi:

[11] O Sumô de Robôs "é um modelo padrão para o desenvolvimento de robôs autônomos direcionado à iniciação de estudantes na robótica. Nesta competição, a intenção não é agredir (destruir) o adversário, mas sim empurrá-lo para fora do *Dohyo* (arena) e se manter na arena durante o maior tempo possível" (OBR, 2010b).

> “Proporcionar ao aluno uma iniciação na área de Robótica, aprendendo como construir e programar robôs para agirem de forma autônoma. Dois robôs competem com o intuito de um tirar o outro da arena. Aquele que conseguir tirar o adversário da arena primeiro ou fizer a maior pontuação na rodada é o vencedor” (OBR, 2009: p.1)

O “Sumô de Robôs” consiste numa modalidade de competição inspirada no esporte Sumô, originado no Japão há milênios, cujo propósito se assemelha ao sumô humano. Porém, no lugar de pessoas, os competidores são robôs e, assim como no sumô humano, o sumô de robôs não tem fins destrutivos. O jogo consiste, apenas, na retirada, de forma inteligente, do adversário de dentro da arena. Esta arena, por sua vez, chamada de *“Dohyo”* (veja a *Figura 37*), nada mais é que uma plataforma circular com 100cm de diâmetro e 3cm de altura, com margens delimitadas por uma faixa preta de 2cm de largura, em cujo espaço são dispostos os robôs, um ao lado do outro, em direções contrárias, conforme a *Figura 38*.

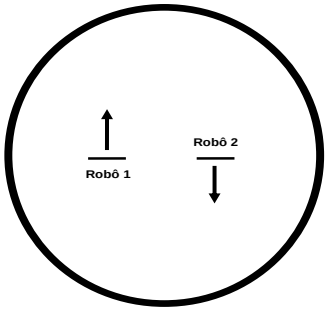


Figura 37
Arena de Sumô
da OBR - 2009

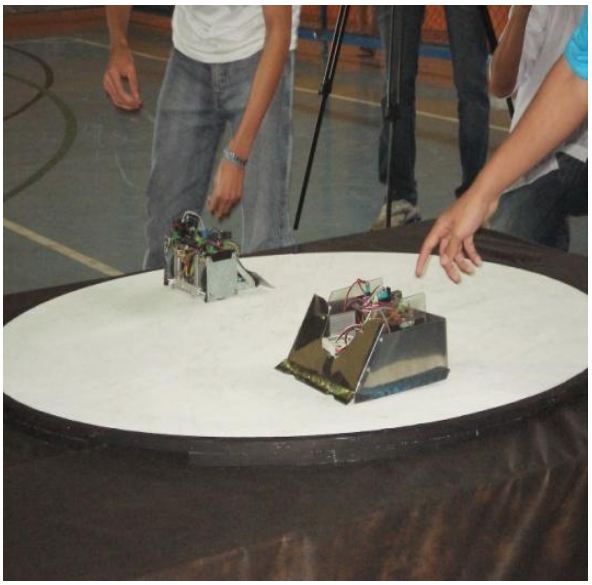


Figura 38 -
Disposição dos robôs na arena - OBR 2009

Em 2009, a OBR teve como desafio a construção de um robô completamente autônomo, tendo que realizar a “identificação do espaço (arena) e do adversário para compor a sua estratégia de vitória” (OBR, 2009: p.1). Para a competição, a OBR exigiu a formação de equipes composta por no mínimo 2 ou no máximo 4 alunos do ensino fundamental, e um professor, atuando como orientador, ocorrendo em duas etapas: uma regional, e outra, nacional.

O robô Virgulino, alvo deste trabalho, participou das duas etapas, tendo conquistado a 3ª. colocação a nível nacional.

1.1. Etapa Regional

O robô Virgulino teve seu primeiro teste na etapa regional que aconteceu em Vitória da Conquista, no Colégio Nossa Senhora de Fátima, no dia 27 de setembro. Ele foi defendido por uma equipe composta por 4 estudantes da 6ª. série do ensino fundamental, tendo um professor de robótica, como orientador.

Disputou 5 partidas, ganhando todas as lutas com apenas uma estratégia: girar 90 graus e empurrar o adversário até encontrar a borda da arena. O seu bom desempenho se deu devido aos motores utilizados que foram capazes de executar uma velocidade média de 50cm/s, enquanto que seus adversários não ultrapassaram a velocidade média de 17cm/s.

Nesta etapa, ele se consagrou como campeão baiano, no nível I, ganhando inclusive do robô vice-campeão nacional do ano de 2008.

1.2. Etapa Nacional

A etapa nacional ocorreu em Brasília-DF, na UNB, onde participaram 12 equipes, representando os seguintes estados: Alagoas, Amazonas, Bahia, Distrito Federal, Espírito Santo, Goiás, Paraíba, Pernambuco, Rio de Janeiro, Rio Grande do Norte, Rio Grande do Sul e São Paulo.

O robô Virgulino, nesta fase, manteve a sua estratégia de luta da etapa anterior, com uma única diferença: inclinação da rampa do robô, deixando-o mais rente à superfície.

No momento inicial da competição, o robô Virgulino concorreu com 4 equipes, obtendo vitória absoluta. Este resultado possibilitou a ele se classificar para as quartas de final, onde disputou com a equipe de São Paulo. O robô adversário, nesta fase, conseguiu vencê-lo, atingindo o seu flanco esquerdo, o que o tirou da arena. Este desempenho não o eliminou da competição. Encarou mais um adversário, a equipe do Rio Grande do Norte, tendo ganhado a luta. Conseguiu chegar a semi-final, considerando-se a pontuação obtida, tendo classificado em terceiro lugar no resultado final.

Estes resultados alcançados constatam a eficiência e eficácia do robô aqui apresentado.

Considerações Finais

Pode-se afirmar, a partir deste trabalho, que a robótica hoje é uma ferramenta importante, e porque não dizer, indispensável à escola que pretende ser atual e conectada à mudança dos tempos e inovações desta era. A experiência de construção do robô Virgulino, como um robô móvel e autônomo, empregando a plataforma Arduino e o uso de componentes de baixo custo ou mesmo a reutilização de sucata eletrônica, pode ser considerada como uma opção viável e acessível à realidade das escolas brasileiras, para o incremento de ferramentas pedagógicas em situações de ensino-aprendizagem através da robótica educacional. Por meio dela se possibilita à escola a contextualização do conhecimento, vez que através dela, o conhecimento se constrói concretamente, passando a ter significado e, ao mesmo tempo, ser significante para o educando. E ainda, através dos desafios que a robótica educacional propõe, se vivencia um ambiente inovador, estimulante e motivador, reafirmando de forma mais criativa a função formativa e informativa da escola.

No decorrer do estudo e, mais especificamente, durante a construção do robô Virgulino observou-se concretamente o uso de várias áreas do conhecimento, exigindo dos educandos a aplicação prática de conceitos empregados na Matemática e Física, por exemplo, antes conceitos puramente abstratos. Isto reafirmou o caráter interdisciplinar da robótica educacional, facilitando a introdução do conhecimento sobre microcontroladores, componentes eletrônicos, linguagem de programação, funcionamento de atuadores e sensores. Foi possível observar também o quanto a robótica estimula, nos educandos, o raciocínio lógico, a concentração, a disciplina, a responsabilidade com o cumprimento de prazos, o uso prático da língua inglesa, o desenvolvimento de habilidades manuais, além de exercitar a troca de conhecimento entre eles e o estabelecimento de relações interpessoais saudáveis, cooperativas e solidárias.

O robô Virgulino e o seu processo de criação desencadearam ainda, nos educandos, o empenho e compromisso, além de um desejo crescente de participarem da competição proposta pela OBR. Representou uma motivação a

mais, a medida em que a Olímpiada se constituiu numa realidade concreta, inclusive, reconhecendo o mérito do robô e do trabalho desenvolvido pela equipe.

Para o educador, a aplicação da robótica educacional como uma ação pioneira na escola, exigiu-lhe o domínio de conhecimentos técnicos específicos e pedagógicos, e o estudo permanente, estimulando-o a buscar e pesquisar incansavelmente material e metodologia apropriados. Apesar de haver atualmente uma quantidade importante de material disponível na rede mundial, a dificuldade encontrada foi à adequação deste à realidade local. Isto exigiu dedicação, sistematização de apostilas e exercícios teóricos para a introdução dos conceitos específicos da robótica.

Com relação à plataforma utilizada para o desenvolvimento do robô Virgulino, o Arduino, percebeu-se que, embora neste estudo, ela tenha se direcionado a uma proposta específica, o “Sumô de Robôs”, a versatilidade da plataforma Arduino proporciona o uso em várias outras atividades pedagógicas como investigações científicas e automação de tarefas.

Finalmente, pode-se afirmar que o robô Virgulino apresentado neste trabalho, pode se configurar numa solução viável como *kit* educacional de baixo custo, por utilizar uma plataforma *Open-Source* e componentes eletrônicos acessíveis, permitindo a criação de robôs de boa qualidade e desempenho satisfatório, e ao mesmo tempo, uma opção para o ensino da robotica educacional, democratizando-a e tornando-a viável em instituições de ensino em todo o território nacional, particularmente para as escolas públicas, já que as privadas possuem, inegavelmente, mais condições de tornar a robótica uma realidade em seus currículos.

1. Trabalhos Futuros

Embora tenha se comprovado que o robô Virgulino atendeu às expectativas inicialmente projetadas, o seu desenvolvimento e a experiência obtida na aplicação dele, possibilitou a percepção de que o seu desempenho poderá ainda ser melhor, mediante algumas modificações que serão descritas a seguir:

- Implementar uma interface gráfica que provê comandos de envio, gerenciamento e criação de comandos, pois com essa funcionalidade pode-se oportunizar a inclusão de educandos da educação infantil;
- Acrescentar sensores de toque, acelerômetro, infra-vermelho, ultrassom e giroscópio;
- Equipar o agente com baterias de lítio, a fim de diminuir o peso e o espaço ocupado;
- Substituir os motores por outros de menor peso e maior torque;
- Colocar comunicação via *Bluetooth* com o objetivo de transferir arquivos sem a necessidade de usar cabos de transmissão;
- Organizar os cabos de forma que diminua a sua quantidade e facilite o manuseio;
- Adicionar um *display* de LCD para facilitar a interface entre homem e máquina.

Referências Bibliográficas

1. ALMEIDA, M. E. **Informática e formação de professores**. Volume 1. Brasília: Editora Parma, 2000.

2. ARAÚJO, 2010. Disponível em: <http://www.esev.ipv.pt/tear/Recursos/Hits.ASP?URL=28%2FCircuitos+integrados.ppt%26CodRecurso%3D177>, Acesso em 13.07.2010.

3. ARDUINO, 2010a. Disponível em <http://www.arduino.cc>, Acesso em 17.11.2010.

4. ARDUINO, 2010b. Disponível em: <http://www.arduino.cc/en/Reference/HomePage>, Acesso em 17.11.2010

5. ARDUINO, 2010c. Disponível em: <http://www.arduino.cc/en/Guide/Environment>. Acesso em 28.11.2010.

6. ARDUINO, 2010d. Disponível em: <http://www.arduino.cc/en/Main/ArduinoBoardDuemilanove> Acesso em 17.11.2010

7. ARDUINO, 2010e. Disponível em: <www.arduino.cc/en/Main/ArduinoBoardMega>, Acesso em 28.11.2010.

8. AURÉLIO, 2010. Disponível em: <http://www.dicionariodoaurelio.com/Sumo>, Acesso em 23.11.2010.

9. BAGNALL, Brian. ***Maximum*** **Lego NXT**: *building robots with Java brains. Winnipeg: Variant Press*, 2007.

10. BRAGA, Newton C. **Eletrônica básica para mecatrônica**. São Paulo: Editora Saber, 2005.

11. BURLAMAQUI, A. et AL., **Introdução a Robótica**. 2010. Disponível em:<http://aquilesburlamaqui.wdfiles.com/local--files/apresentacoes/IntroducaoARobotica.pdf>, Acesso em 20.08.2010.

12. CORTELETTI, D. **Dossiê Técnico**: introdução à programação de microcontroladores *Microchip* PIC. SENAI-RS. Centro Tecnológico de Mecatrônica. 2006.

13. FORESTI, H. B. **Desenvolvimento de um robô bípede autônomo.** Dissertação de Mestrado, Departamento de Engenharia Mecânica. UFPE, 2006.

14. http://www.arduino.cc/en/Main/ArduinoBoardDuemilanove

15. JÁCOBO, J. E. A. **Desenvolvimento de um robô autônomo versátil utilizando arquitetura subsumption**, Dissertação de Mestrado, Departamento de Mecânica Computacional. Faculdade de Engenharia Mecânica. Universidade Estadual de Campinas, Campinas, SP: 2001.

16. LEGO, 2010. Disponível em: <http://www.lego.com/>, Acesso em 16.09.2010.

17. MARIMOTO, C.E. ***Hardware*, o guia definitivo**. GDH Press e Sul Editores. São Paulo: 2007, 848p.

18. MENEZES, E. T. de; SANTOS, T .H. dos. ”Robótica educacional” (verbete). **Dicionário Interativo da Educação Brasileira** – EducaBrasil. São Paulo: Midiamix Editora, 2002. Disponível em: <50TTP://www.educabrasil.com.br/eb/dic/dicionario.asp?id=49>, acesso em: 16/7/2010.

19. MUIR, P.F. ***Modelling and Controlo f wheeled móbile robots***. 1998. 335p. Tese (PhD.) *Department of Eletric and Computer Engineering and Robotics Institute, Canegie Mellon University, Pittsburgh*, PA, USA.

20. NETO, G. M. **Desenvolvimento de um robô móvel versátil**. 2007. Monografia (Graduação em Engenharia Mecatrônica). Escola Superior de Tecnologia – EST, Universidade do Estado do Amazonas – UEA. Manaus.

21. NITZKE, CAMPOS e LIMA, 2010. Disponível em: <http://penta.ufrgs.br/~marcia/constru1.htm>, Acesso em 18.11.2010.

22. OBR, 2010a. Disponível em: <http://obr.ic.unicamp.br/Anteriores/obr2009/MatrizReferencia>, Acesso em 08.11.2010.

23. OBR, 2010b. Disponível em: <http://obr.ic.unicamp.br/Anteriores/obr2009>, Acesso em 08.11.2010.

24. OBR. **Regulamento da modalidade prática nível I**, 2009 (Disponível em: <http://obr.ic.unicamp.br/Anteriores/obr2009/regulamentoPratica>, acesso em 20/11/2010.

25. PEREIRA, F. **Microcontroladores PIC:** programação em C. São Paulo: Érica, 2003a.

26. PIEROTTI, T. M. **Ensino de inteligência artificial através de robôs móveis inteligentes**. 2007. Monografia (Graduação em Ciência da Computação). Universidade Estadual de Londrina. Sabe o departamento?

27. REZENDE, M. F. de. **Desenvolvimento de um Robô Móvel Autônomo Inteligente Utilizando a Arquitetura de Assunção**. 1992. 102p. Dissertação (Mestrado). Centro de Ciências Exatas e Tecnologia, Universidade Federal de Uberlândia. Uberlândia – MG

28. RIGHETTI, 2005. Disponível em: <http://www.comciencia.br/reportagens/2005/10/02_impr.shtml>, Acesso em 28.11.2010.

29. ROBOTSHOP, 2010. Disponível em: <http://robotshop.ca/>, Acesso em 02.10.2010.

30. ROBOTSHOP, 2010b. Disponível em: < http://www.robotshop.ca/vex-easyc-2-programming-software.html>, Acesso em 28.11.2010.

31. ROBOTSHOP, 2010c. *Disponível em:* <www.robotshop.ca/.../solarbotics-hvwtech-freeduino-usb-microcontroller.html>, Acesso em 28.11.2010.

32. SABERELETRÔNICA, 2010. Disponível em: <www.sabereletronica.comEE.br/secoes/leitura/210/imprimir:yes>, Acesso em 28.11.2010.

33. SCHUNK, L. M. **Microcontroladores AVR**: teoria e aplicações práticas. São Paulo: Érica, 2001.

34. SILVA, A. F. da **RoboEduc**: uma metodologia de aprendizado com robótica educacional.2009. Tese (Doutorado). Universidade Federal do Rio Grande do Norte. Natal.

35. SILVA, R. C. B. et al. **ROBÓTICA:** Construção e análise do comportamento de robôs móveis aplicados na solução do problema do labirinto. s/d. Artigo (Graduação em Ciência da Computação). Faculdade Ruy Barbosa.

36. SILVEIRA, J. **Sistemas em tempo real**. UFC\DETI. s/d. (apresentação aula sobre circuitos integrados).

37. *SOLARBOTICS*, 2010a. Disponível em 2010: <http://solarbotics.com/>, Acesso em 10.10.2010.

38. SOLARBOTICS, 2010b. Disponível em: <http:///www.solarbotics.com/freeduino_arduino/arduino_boards/>, Acesso em 28.11.2010.

39. VEXROBOTICS, 2010. Disponível em <http://www.vexrobotics.com>, Acesso em 25.09.2010).

40. ZANELATTO, M. S. **Robótica Educacional nos Cursos de Ciência da Computação**. 2004. Monografia (Bacharel em Ciência da Computação) – Universidade Estadual de Londrina.

41. ZELENOVSKY e MENDONÇA, 2010. Disponível em: <http://www.mzeditora.com.br/artigos/mic_modernos.htm>, Acesso em 19.11.2010.

42. 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.

Código Fonte do Robô Virgulino

```
int led13 = 13;
int sensorLuzE = 28;
int sensorLuzD = 30;
int sensorLuzTraseiroE = 24;
int sensorLuzTraseiroD = 26;
int valorSensorLuzE = 1;
int valorSensorLuzD = 1;
int valorSensorLuzTraseiroE = 1;
int valorSensorLuzTraseiroD = 1;
int direcaoMotorEquerdo1 = 3;
int direcaoMotorEquerdo2 = 2;
int pwnMotorEsquerdo = 5;
int direcaoMotorDireito1 = 6;
int direcaoMotorDireito2 = 4;
int pwnMotorDireito = 7;
int botaoA = 32;
int botaoB = 42;
int botaoC = 34;
int botaoD= 40;
int botaoE = 36;
int botaoF= 38;
int valorBotaoA=0;
int valorBotaoB=0;
int valorBotaoC=0;
int valorBotaoD=0;
int valorBotaoE=0;
int valorBotaoF=0;
```

```
int contador=0;
char sentido = 'h';
int ligado = 0;

void setup()
{
  pinMode(direcaoMotorEquerdo1,OUTPUT);
  pinMode(direcaoMotorEquerdo2,OUTPUT);
  pinMode(pwnMotorEsquerdo,OUTPUT);
  pinMode(direcaoMotorDireito1,OUTPUT);
  pinMode(direcaoMotorDireito2,OUTPUT);
  pinMode(pwnMotorDireito,OUTPUT);
  pinMode(sensorLuzE,INPUT);
  pinMode(sensorLuzD,INPUT);
}

void loop()
{
valorBotaoA=digitalRead(botaoA);
valorBotaoB=digitalRead(botaoB);
valorSensorLuzE=digitalRead(sensorLuzE);
valorSensorLuzD=digitalRead(sensorLuzD);
valorSensorLuzTraseiroE=digitalRead(sensorLuzTraseiroE);
valorSensorLuzTraseiroD=digitalRead(sensorLuzTraseiroD);

    if(valorBotaoA == 1)
      {
      girarEixoHorario();
      delay(700);
      paraFrente();
      }

      else if ((valorSensorLuzE == 0) || (valorSensorLuzD == 0)  )
```

```
    {
     paraTras();
     delay(10);
    }

    else if ((valorSensorLuzTraseiroE == 0) || (valorSensorLuzTraseiroD == 0))
    {
     paraFrente();
     delay(10);
    }

if(valorBotaoB == 1)
    {
    circular();



    }

    else if ((valorSensorLuzE == 0) )
    {
     girarEixoHorario();
      delay(10);
      fazerCurvaD();
      delay(50);
      paraFrente();

    }

    else if  (valorSensorLuzTraseiroD == 0)
    {
     paraFrente();
     delay(10);
    }
```

```
}

void parar() {
  digitalWrite(direcaoMotorEquerdo1,LOW);
  digitalWrite(direcaoMotorEquerdo2,LOW);
  analogWrite(pwnMotorEsquerdo,0);
  digitalWrite(direcaoMotorDireito1,LOW);
  digitalWrite(direcaoMotorDireito2,LOW);
  analogWrite(pwnMotorDireito,0);
}
void paraFrente()
 {
  digitalWrite(direcaoMotorEquerdo1,LOW);
  digitalWrite(direcaoMotorEquerdo2,HIGH);
  analogWrite(pwnMotorEsquerdo,200);

  digitalWrite(direcaoMotorDireito1,LOW);
  digitalWrite(direcaoMotorDireito2,HIGH);
  analogWrite(pwnMotorDireito,200);
}

void paraFrenteA() {
  digitalWrite(direcaoMotorEquerdo1,HIGH);
  digitalWrite(direcaoMotorEquerdo2,LOW);
  analogWrite(pwnMotorEsquerdo,200);
  digitalWrite(direcaoMotorDireito1,HIGH);
  digitalWrite(direcaoMotorDireito2,LOW);
  analogWrite(pwnMotorDireito,200);
}

void paraFrenteB() {
  digitalWrite(direcaoMotorEquerdo1,LOW);
  digitalWrite(direcaoMotorEquerdo2,HIGH);
```

```
  analogWrite(pwnMotorEsquerdo,180);
  digitalWrite(direcaoMotorDireito1,LOW);
  digitalWrite(direcaoMotorDireito2,HIGH);
  analogWrite(pwnMotorDireito,230);
}

void paraTras()
{
  digitalWrite(direcaoMotorEquerdo1,HIGH);
  digitalWrite(direcaoMotorEquerdo2,LOW);
  analogWrite(pwnMotorEsquerdo,200);
  digitalWrite(direcaoMotorDireito1,HIGH);
  digitalWrite(direcaoMotorDireito2,LOW);
  analogWrite(pwnMotorDireito,200);
}
void girarEixoHorario()
{
  digitalWrite(direcaoMotorEquerdo1,HIGH);
  digitalWrite(direcaoMotorEquerdo2,LOW);
  analogWrite(pwnMotorEsquerdo,100);
 digitalWrite(direcaoMotorDireito1,LOW);
  digitalWrite(direcaoMotorDireito2,HIGH);
  analogWrite(pwnMotorDireito,100);
  }

void girarEixoAnti()
{
  digitalWrite(direcaoMotorEquerdo1,LOW);
  digitalWrite(direcaoMotorEquerdo2,HIGH);
  analogWrite(pwnMotorEsquerdo,100);  //Tinha o valor 0
  digitalWrite(direcaoMotorDireito1,HIGH);
  digitalWrite(direcaoMotorDireito2,LOW);
  analogWrite(pwnMotorDireito,100);
```

```
}


void rodar() {

  digitalWrite(direcaoMotorEquerdo1,LOW);
  digitalWrite(direcaoMotorEquerdo2,HIGH);
  analogWrite(pwnMotorEsquerdo,255);
  digitalWrite(direcaoMotorDireito1,LOW);
  digitalWrite(direcaoMotorDireito2,HIGH);
  analogWrite(pwnMotorDireito,200);
}

void fazerCurvaD() {
  digitalWrite(direcaoMotorEquerdo1,LOW);
  digitalWrite(direcaoMotorEquerdo2,HIGH);
  analogWrite(pwnMotorEsquerdo,200);
  digitalWrite(direcaoMotorDireito1,LOW);
  digitalWrite(direcaoMotorDireito2,HIGH);
  analogWrite(pwnMotorDireito,255);
}
void fazerCurvaE() {
  digitalWrite(direcaoMotorEquerdo1,LOW);
  digitalWrite(direcaoMotorEquerdo2,HIGH);
  analogWrite(pwnMotorEsquerdo,255);
  digitalWrite(direcaoMotorDireito1,LOW);
  digitalWrite(direcaoMotorDireito2,HIGH);
  analogWrite(pwnMotorDireito,80);
}


void fazerCurvonaD() {
  digitalWrite(direcaoMotorEquerdo1,HIGH);
```

```
  digitalWrite(direcaoMotorEquerdo2,LOW);
  analogWrite(pwnMotorEsquerdo,255);
  digitalWrite(direcaoMotorDireito1,LOW);
  digitalWrite(direcaoMotorDireito2,HIGH);
  analogWrite(pwnMotorDireito,255);
}

void fazerCurvonaE()
{
  digitalWrite(direcaoMotorEquerdo1,LOW);
  digitalWrite(direcaoMotorEquerdo2,HIGH);
  analogWrite(pwnMotorEsquerdo,255);
  digitalWrite(direcaoMotorDireito1,HIGH);
  digitalWrite(direcaoMotorDireito2,LOW);
  analogWrite(pwnMotorDireito,255);
 }
void circular()
{
  digitalWrite(direcaoMotorEquerdo1,LOW);
  digitalWrite(direcaoMotorEquerdo2,HIGH);
  analogWrite(pwnMotorEsquerdo,70);
  digitalWrite(direcaoMotorDireito1,LOW);
  digitalWrite(direcaoMotorDireito2,HIGH);
  analogWrite(pwnMotorDireito,180);  }
```