Universidade Federal do Rio de Janeiro

Escola Politécnica

Departamento de Eletrônica e de Computação

RoboLogo: Sistema Móvel Controlado Remotamente pela Linguagem Logo para Aplicações Educacionais

Autor:

Délio Silva Nunes

Orientador:

Prof. Joarez Bastos Monteiro, D. Sc.

Examinador:

Prof. Antônio Cláudio Gómez de Sousa, M. Sc.

Examinador:

Prof. José Gabriel Rodriguez Carneiro Gomes, Ph. D.

DEL

Setembro de 2010

DEDICATÓRIA

Aos meus pais, familiares, amigos e professores, que tanto me ajudaram durante toda minha graduação, contribuindo de forma significativa para a construção do meu caráter como pessoa e engenheiro.

AGRADECIMENTO

Em primeiro lugar agradeço aos meus pais pelo apoio incondicional. Por sempre estarem ao meu lado e por serem parte fundamental da pessoa em que me tornei.

À toda minha família por entender minhas ausências durante minha graduação.

Agradeço ao meu orientador, professor Joarez Monteiro, que me ajudou indo além de sua competência como professor e que dividiu comigo um pouco da sua experiência através de suas histórias e conselhos. Também não poderia deixar de agradecer por sua compreensão, disponibilidade e conhecimentos fundamentais para conclusão deste trabalho.

Aos professores do departamento por todo conhecimento compartilhado, por me ajudarem a concretizar esse sonho.

À UFRJ pelo investimento com a bolsa PIBEX que a mim foi pleiteada, e a FINEP que através dos recursos do projeto PROMOVE me forneceu boas condições para realização deste projeto.

Aos meus amigos de faculdade, que trilharam comigo este longo caminho, com quem também pude compartilhar muitas histórias e pelas noites de estudo nas bibliotecas.

Um agradecimento especial aos amigos Raphael Sadao, Roberto Dias, Pedro Coelho e Alexandre Guazzi sem os quais, não só a graduação, mas também a conclusão deste trabalho, teriam sido muito mais difíceis.

RESUMO

Nos dias atuais existe um grande volume de informação nos atingindo diariamente. Foi percebido que ensiná-lo aos alunos pode se tornar uma tarefa infindável e, por isso, percebeu-se que ensinar o aluno a aprender é extremamente importante, isso quer dizer, estimular a curiosidade e desenvolver um raciocínio lógico solucionador de problemas. Este é o intuito do pensamento construtivista e com esse princípio nasceu a Linguagem LOGO.

O Construtivismo (concepção voltada à educação) é uma das correntes teóricas que diz que o desenvolvimento da inteligência é determinado pelas ações mútuas entre o indivíduo e o meio. Isso quer dizer: 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, modelando, o tempo todo, suas ações e operações conceituais com base nas suas experiências [1].

O LOGO é uma linguagem de programação interpretada, utilizada com grande sucesso como ferramenta de apoio ao ensino regular e por aprendizes em programação de computadores. Ela implementa, em certos aspectos, a filosofia construtivista, segundo a interpretação de Seymour Papert, co-criador da linguagem junto com Wally Feurzeig [2].

Uma vez que a linguagem é interpretada e interativa, o resultado é mostrado imediatamente após digitar-se o comando – incentivando o aprendizado. Nela, o aluno aprende com seus erros, vivenciando e tendo que repassar este conhecimento para o LOGO. Se algo está errado em seu raciocínio, isto é claramente percebido e demonstrado na tela, fazendo com que o aluno pense sobre o que poderia estar errado e tente, a partir dos erros vistos, encontrar soluções corretas para os problemas. A maioria dos comandos, pelo menos nas versões mais antigas, refere-se a desenhar e pintar. Mas em versões mais atuais, como o AF LOGO, podem ser muito mais abrangentes, trabalhando com textos, fórmulas e até IA (Inteligência Artificial), servindo como excelente ferramenta para o ensino regular.

Existem, também, comandos para se manipular a porta paralela do computador, o que permite desenvolver projetos de robótica utilizando o LOGO, que pode passar a controlar robôs e mecanismos de desenho, gerando uma interação entre o conhecimento

adquirido e o "mundo físico", entre outras coisas, e é neste contexto que o projeto está inserido.

Nesta linguagem de programação o cursor na tela, tradicionalmente representado por uma tartaruga, realiza diversos trajetos desenhando figuras. O objetivo é construir um sistema real/físico que reproduza os mesmos movimentos que esse cursor executa na tela através de comandos do LOGO e, assim, ajude no raciocínio matemático/físico e estimule maior interesse pela área da engenharia, área esta que vem sendo pouco procurada e foi identificada pelo Ministério de Ciência e Tecnologia como de vital importância para o desenvolvimento do país [3].

Palavras-Chave: LOGO, sensor de deslocamento, *PWM*, Arduino, *driver* Ponte-H, transmissor e receptor RF.

ABSTRACT

Nowadays there is a large amount of information available to us every day and it was found out that teaching students can become an endless task. Therefore, teaching students how to learn is extremely important, encouraging their curiosity and developing their logical thinking. This is the goal of constructivist thought and according to this principle the LOGO language was born.

The constructivism (a concept related to education) is one of the theoretical perspectives which says that the development of intelligence is determined by mutual actions between the individual and the environment. This means that human being are not intelligent by the time of birth, but they are not passive to the influence of the environment either. They respond to external stimulation using it to build and organize their own knowledge in a more and more elaborate way, modeling all the time their actions and conceptual operations based on their experiences [1].

LOGO is an interpreted programming language used with great success as a tool to support regular education and learners in computer programming. It implements, in some respects, the constructivist philosophy, as interpreted by Seymour Papert, co-creator of the language along with Wally Feurzeig [2].

Since the language is interpreted and interactive, the result is displayed immediately after the command is typed, encouraging the learning. With this language the students learn from their own mistakes, experiencing and applying this knowledge to the LOGO. If something is wrong with a idea, it is clearly understood and shown on the screen, making the student think about what could be wrong and try, to find the right solutions for problems from the errors. Most of the commands, at least in older versions, refer to drawing and painting. In later versions such as AF LOGO however, commands can be much more comprehensive, working with text, formulas, and even AI (Artificial Intelligence), serving as an excellent tool for education.

There are also commands to control the computer's parallel and serial ports, which allows the development of robotics projects using the LOGO. It is possible to control robots and drawing mechanisms, thereby creating an interaction between the acquired knowledge and the "physical world" as well as other functions. This project is inserted in this context.

In this programming language the cursor on the screen, traditionally represented by a turtle, performs a variety of paths to draw pictures. The goal is to build a

real/physical system that reproduces through LOGO commands the same moves performed by the cursor on the screen. This system helps to improve the mathematical/physical thinking and fosters a greater interest in the field of engineering, which was identified by the Ministry of Science and Technology as of vital importance for the development of our country [3].

Keywords: LOGO, displacement sensor, PWM, Arduino, H-Bridge Driver, transmitter and RF receiver.

SIGLAS

IA – Inteligencia Artificial

UFRJ – Universidade Federal do Rio de Janeiro

PWM – "*Pulse Width Modulation*". Modulação por largura de pulso

RF – Radio Frequência.

TX– Transmissor.

RX – Receptor.

CI – Circuito Integrado.

DC – "*Direct current".* Corrente contínua

PC – "*Personal Computer*". Computador pessoal.

LIpE – Laboratório de Informática para Educação

MOSFET – *Metal Oxide Semiconductor Field Effect Transistor*

CD-Rom – *Compact disc read-only memory.*

HP – *Hewlett-Packard*

FSK – Frequency shift keying.

IDE – Integrated Development Environment

USB – Universal Serial Bus

MSB – Most Significant Bit

LSB – Least Significant Bit

H_i-z – High Impedance (Alta Impedância)

Sumário

Lista de Figuras

Lista de Tabelas

Capítulo 1

Introdução

1.1 – Tema

A proposta deste trabalho consiste em desenvolver um periférico móvel e autônomo para o LOGO. Tal *hardware* terá finalidade educativa, auxiliando o aprendizado em sala de aula e desenvolvendo o raciocínio lógico em alunos dos níveis fundamental e médio. Como houve a preocupação de tornar seu custo baixo, tentou-se reutilizar a vasta gama de "lixo" eletrônico que se encontra hoje em dia, abordando assim um outro tema também importante que é a reciclagem dos chamados "e-lixos", que tem se tornado uma nova ameaça ao nosso ambiente.

1.2 – Objetivos

O objetivo é criar um dispositivo real/físico, controlado pelo software LOGO, que se locomova em cima de uma superfície, desenhando ou não, e agindo de forma sincronizada com o cursor na tela.

Para uma maior praticidade, esse dispositivo tem que ser capaz de se comunicar, sem fio, com o computador; deve dispor de um sensor para que se tenha informação sobre o caminho que percorre e, assim, corrigir eventuais erros, causados por derrapagem ou outros fatores; deve ter, também, um sistema de controle para as rodas e para o sistema de ativação da caneta. Para permitir o seu controle, deve-se criar um conjunto de comandos específicos que, acrescentados ao conjunto de instruções do LOGO, permitirão acionar os atuadores de movimento e escrita do dispositivo móvel. Também, deve-se criar um conjunto de atividades para execução com alunos em sala de aula.

1.3 – Motivação

O LOGO deixa transparecer que "ideias poderosas" surgem espontaneamente da atividade do aluno ao programar - com o auxílio do professor, ensinando a sintaxe do LOGO e lançando desafios. Um aspecto importante, nas concepções de Papert [2], é o fato de, no LOGO, considerar-se o erro como um importante fator de aprendizagem, o que oferece oportunidades para que o aluno entenda porque errou e busque uma nova solução para o problema, investigando, explorando, descobrindo por si próprio, ou seja, a aprendizagem pela descoberta.

Por exemplo, a criança não aprende apenas pelo ensino formal e deliberado, ela é uma aprendiz nata, mesmo antes de chegar à escola apresenta conhecimentos adquiridos por meio de uma aprendizagem natural, espontânea e intuitiva, que se dá através da exploração, da busca e da investigação, a qual pode ser caracterizada como uma real auto-aprendizagem.

Esse aprendizado tido através da prática, além de mais prazeroso, é muito mais efetivo, pois sua aplicação está sujeita às condições reais e variáveis, ao contrário da teoria pura.

Segundo o filósofo e professor norte-americano John Dewey (1859-1952) e de seu seguidor (orientando em seu doutoramento) Donald Schön (1930-1997) em o "Professor Reflexivo", o aprendizado só ocorre quando há uma situação de problema real para se resolver e com base nos conhecimentos teóricos e na experiência prática, é possível solucioná-lo [4].

Sendo uma ferramenta de programação de fácil manuseio, o LOGO permite que os alunos criem, através de comandos simples, figuras complexas; e promove o desenvolvimento do raciocínio lógico e o aprendizado/consolidação de conceitos matemáticos. Neste sentido o presente projeto é uma complementação ao LOGO, funcionando como um elemento incentivador, pois reproduz, no campo real, os movimentos do cursor na tela. E ainda, como os comandos do *software* descrevem distâncias, ângulos e coordenadas – assuntos tratados em sala de aula de forma, às vezes, abstrata – esse projeto se torna uma ótima forma para trazer estes temas ao mundo real e ver a sua importância, já que a realização de uma figura e de um trajeto que o sistema realizará depende destes valores. Por se tratar de um elemento real, conceitos físicos poderão ser abordados.

1.4 – Metodologia

No LIpE, laboratório de extensão do qual sou integrante, trabalha-se com a linguagem LOGO no ensino da informática, matemática e raciocínio lógico, tanto com alunos, como professores de escolas públicas de ensino médio e fundamental. Essa experiência deu base e conhecimento sobre a teoria por trás do LOGO e, estudando a teoria construtivista, percebe-se a importância desse tipo de ferramenta para a construção/consolidação do aprendizado [5].

Como o resultado deste projeto é um *hardware* móvel e autônomo, sua construção precisará de todo um conjunto de peças para que ele possa se locomover, como rodas e motores, assemelhando-se a um carrinho, com um adicional de que o mesmo possa desenhar, ou não, por onde passa.

Durante o processo de escolha das peças houve preocupações em utilizar aquelas que poderíamos obter através de reciclagem de materiais eletrônicos, as que tivessem menor consumo, para dar maior autonomia ao RoboLogo, e as de menor preço, pois deve se ter em mente que este será um projeto com o objetivo de trabalhar e reproduzir em escolas públicas, as quais não possuem grandes recursos financeiros. Seguindo esta linha de escolhas, o *software* não poderia deixar de ser grátis e de código aberto.

Como o RoboLogo deve traçar as linhas com uma certa precisão para formar figuras como triângulos, quadrados, círculos e etc, além de interpretar os comandos para os atuadores de movimento e escrita, ele deve analisar a sua trajetória, em tempo real, para corrigir os erros de deslocamento. Assim para processar todas essas informações que chegam, traduzindo-as em sinais de controles, foi colocado um microcontrolador, dando "inteligência" ao sistema, e abrindo, também, mais possibilidades para projetos futuros. A utilização de um enlace de rádio, torna o dispositivo projetado autônomo e móvel como desejado.

1.5 – Delimitação

O projeto envolve o desenvolvimento de diversos itens, como mecânica, eletrônica de controle, eletrônica do sensor de deslocamento, comunicação sem fio,

programação do *firmware* e programação para a interação do usuário com o RoboLogo. O diagrama de blocos da Figura 1.1 mostra a interação entre as partes.

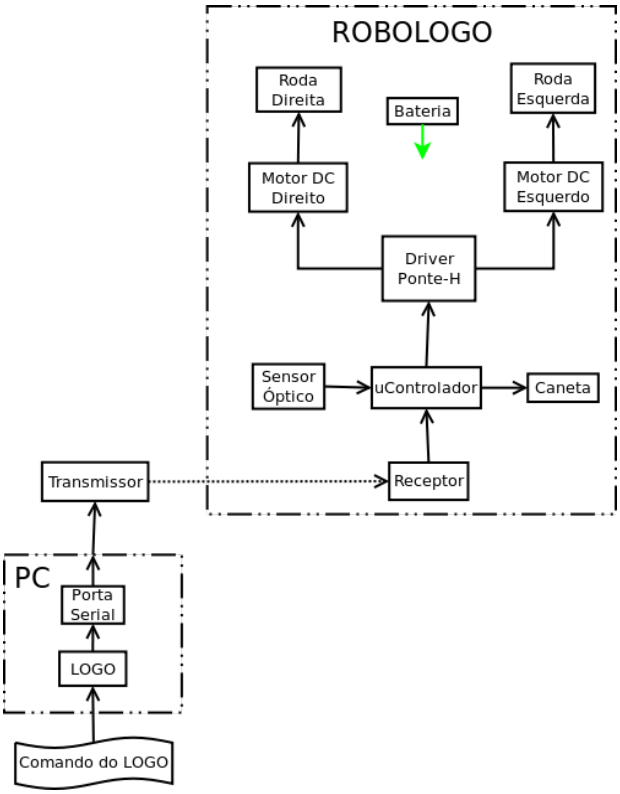


Figura 1.1 – Diagrama em blocos do Projeto

A mecânica será semelhante a de um carrinho e fará utilização de peças como motor, rodas, caixa de redução, driver de corrente e de uma base onde todas as peças serão fixadas, assim como a bateria e os circuitos.

A parte eletrônica do projeto fará utilização de alguns módulos prontos, como IDE microcontrolada, transmissor e receptor RF e o circuito do sensor para realimentação do sistema e correção de trajetória.

O *firmware* será programado com recursos da IDE, utilizando funções e bibliotecas que possam ajudar com controles dos motores e caneta, com a comunicação e com a utilização do sensor.

Os novos comandos criados para que o usuário interaja com o RoboLogo serão feitos utilizando funções da própria linguagem LOGO.

1.6 – Descrição

No Capítulo 2 será abordado todo o desenvolvimento do *hardware*, citando as principais características de cada módulo e componente, organizados em seções. Os módulos se dividem em outras subseções que informam quais os seus respectivos objetivos, motivos para sua escolha, quais suas características e funcionamento, entre outras informações importantes.

O Capítulo 3 apresenta o desenvolvimento do *software*, onde teremos duas seções importantes, que são identificadas como *Firmware* e *Software* SuperLogo. A primeira diz respeito ao algorítimo usado na programação do microcontrolador e o segundo ao *script* de interface do SuperLogo com o usuário final.

No Capítulo 4 temos os resultados obtidos pelo projeto assim como alguns testes importantes realizados.

No Capítulo 5 a conclusão com críticas e sugestões para projetos futuros.

Capítulo 2

Desenvolvimento do *Hardware*

2.1 – Motor

2.1.1 – Objetivo

Gerar torque nas rodas para que o conjunto passe a se deslocar sobre uma superfície plana e horizontal.

2.1.2 – Características desejadas

O motor necessita ter um torque suficientemente grande para carregar todas as peças e ainda assim um consumo baixo para que não diminua muito a autonomia da bateria.

2.1.3 – Escolha

Para que o RoboLogo tenha maior mobilidade, podendo executar um maior número de manobras como girar em torno de seu próprio eixo sem que saia do lugar, algo que é necessário para fazer vértices de figuras geométricas, foram escolhidos dois motores, um para cada roda, com controles independentes.

A existência de uma grande quantidade de sucata de impressoras, *drivers* de cd-roms e disquetes disponíveis hoje, fornece a possibilidade de escolha dos mais diferentes motores. A escolha inicial foi por motores de passos de 5 fios pela facilidade de acionamento.

Na tentativa de melhorar o torque foram escolhidos motores de passo mais forte (da impressora HP 690C) e foi projetada uma redução para eles, o que melhorou o torque, mas a velocidade caiu muito e o consumo aumentou, exigindo a construção de *driver* usando *MOSFET*, pois os CIs (Circuitos Integrados) disponíveis no mercado vão

até 500 mA. No circuito, as medições acusavam 800 mA, em passo simples. Usar meio passo dobraria este consumo. Essas medições foram feitas em apenas um motor.

Logo este sistema iria necessitar de uma bateria mais potente aumentando ainda mais o peso e o comprometendo o movimento devido a sobrecarga.

Devido ao elevado consumo, a solução usando motores de passo foi abandonada, optando-se por usar motores DC. A escolha deste tipo de motor implica em algumas vantagem como a de ter maior torque e velocidade com baixo consumo. A sua desvantagem é a necessidade de um sistema de controle mais sofisticado [6], que, na essência, consiste em um sensor de deslocamento e um sinal *PWM*. O sinal carrega a informação de sentido de rotação e velocidade, e o sensor a informação da posição.

Na procura destes motores muitas peças importantes para o projeto foram encontradas. Os motores escolhidos foram motores de *driver* de *cd-rom*, e nesses mesmos *drivers* foram encontrados eixos e suportes para as rodas, caixa de redução com polias e suas correias que já se encaixavam perfeitamente.

2.2 – Driver ponte-H

2.2.1 – Objetivo

Fornecer corrente para o funcionamento dos motores DC segundo o circuito de controle e permitir a alternância de sentido da corrente nos motores, fazendo assim com que ele possa girar para ambos os lados.

2.2.2 – Funcionamento

O driver de corrente usado foi o CI LM293D, por ter duas pontes-H. O funcionamento está descrito em seu *datasheet*, mas sua tabela de funcionamento se encontra na Tabela 2.1.

EN	A	B	Função
H	L	H	Motor p/ direita
H	H	L	Motor p/ Esquerda
H	L	L	Parada Rápida
H	H	H	Parada Rápida
L	L	L	Parada Rápida

Tabela 2.1 – Tabela de Funcionamento do CI L293D

Para que se pudesse utilizar o CI com máxima eficiência, permitindo escolher os sentidos com apenas um pulso e garantir a simultaneidade do acionamento, foi montado um circuito lógico com um inversor, conforme a Figura 2.1, transformando assim o controle do funcionamento como está mostrado na nova tabela de funcionamento Tabela 2.2. Também pode se perceber na Figura 2.1 que foram colocados diodos *schottky* para suprimir o transiente do motor, protegendo o circuito.

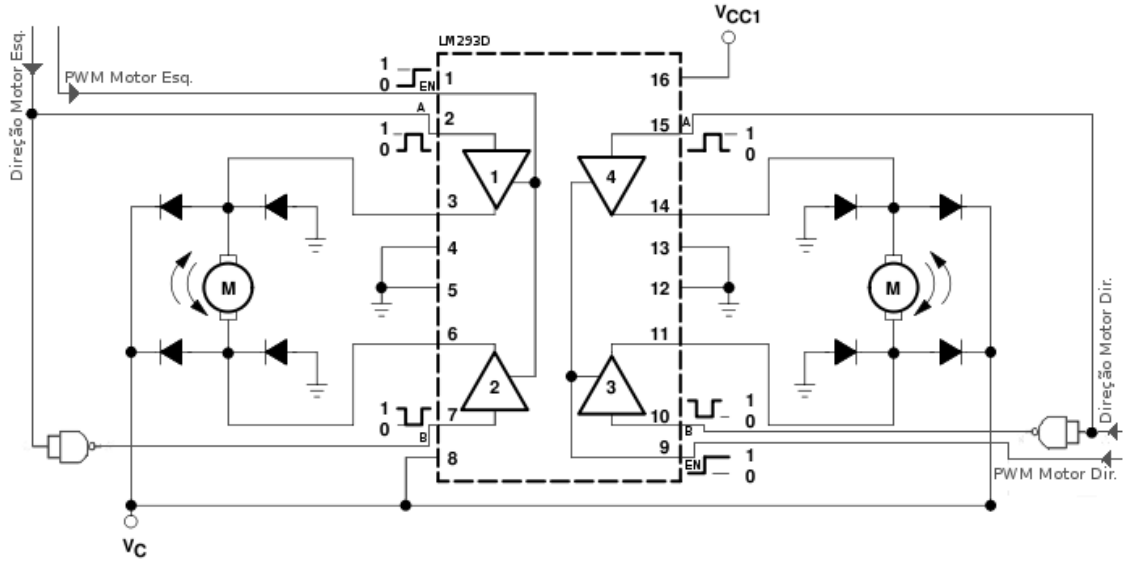


Figura 2.1 – Circuito de driver dos motores

PWM	Direção Motor	Função
L	L	Parada Rápida
L	H	Parada Rápida
H	L	Motor p/ direita
H	H	Motor p/ Esquerda

Tabela 2.2 – Tabela de Funcionamento do circuito da Figura 2.1

2.3 – Sensor

2.3.1 – Objetivo

Fornecer ao microcontrolador informação sobre o deslocamento do RoboLogo e assim corrigir eventuais erros que possam acontecer durante a execução de um trajeto, como por exemplo derrapagem das rodas sobre a superfície.

2.3.2 – Escolha

Com o objetivo de usar as peças acessíveis, pesquisou-se informação sobre o sensor do *mouse* óptico, e por meio desta investigação foi encontrada uma biblioteca para uma plataforma de desenvolvimento chamada de Arduino [7] que utiliza o sensor do *mouse* com o sensor de deslocamento.

Um *mouse* descartado compatível com a biblioteca OptiMouse foi encontrado para ser utilizado, o que não foi difícil, pois a maioria dos *mouses* antigos utilizam o CI PAN3101 ou equivalentes, que são os compatíveis com ela.

2.3.3 – Funcionamento

Para a utilização foram retirados do *mouse* componentes desnecessários aproveitando somente o CI PAN3101 e os componentes referentes ao seu funcionamento. Como o sensor necessita estar próximo à superfície em que se desloca, foi feita uma adaptação para que ficasse abaixo da base do RoboLogo, sem alterar as disposições dos componentes ópticos para não descalibrá-los, tornando-se uma parte da base que desliza sobre a superfície.

Para o seu funcionamento é necessário ligar apenas quatro fios aos pinos do CI, identificados na Figura 2.2.

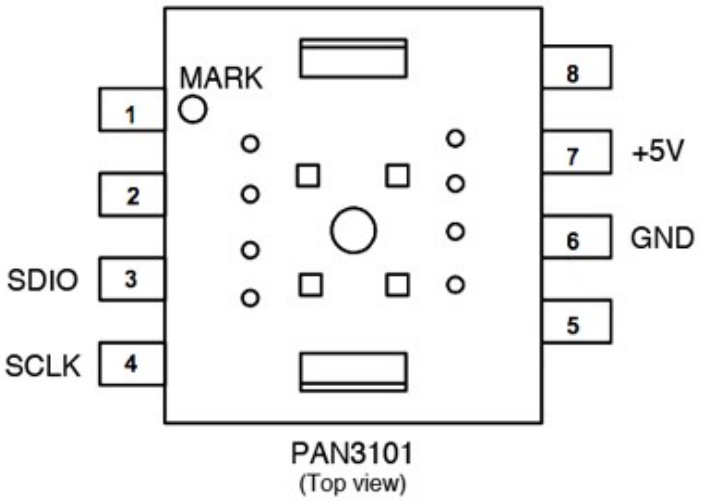


Figura 2.2 – Visão do Sensor [8]

- Pino 3 – SDIO, s*erial interface bi-direction data;*
- Pino 4 – SCLK, s*erial interface clock;*
- Pino 7 – VCC, alimentação +5V;
- Pino 6 – *GND*, *ground,* terra;
- Demais pinos não são utilizados neste projeto.

Os pinos 7 e 6 do CI são os pinos de alimentação do mesmo, o SDIO (s*erial interface bi-direction data*) é responsável por nos dar a informação do deslocamento e o SCLK (s*erial interface clock*) é o *clock*. O sensor funciona como uma micro câmera que tira fotos de 16x16 *pixels* em alta velocidade de acordo com o *clock* nele estabelecido. A partir da comparação dessas fotos, através de processamento das imagens, calcula e informa quantos pontos andou e em que direção. Esta informação já vem em um formato de dados pelo pino 3 (SDIO), 1 *byte* para valores de coordenada "x" e 1 *byte* para "y".

Toda a comunicação de dados sobre SDIO é sincronizada por SCLK. SDIO é alterado na queda de nível de tensão do sinal SCLK e lido em cada subida do SCLK, como pode ser verificado no diagrama de tempo na Figura 2.3. O micro-controlador sempre inicia a comunicação e o sensor responde com a transferência de dados.

Através do SDIO podemos escrever ou ler os registradores do PAN3101. Os registradores de escrita servem para configurar algumas características de operação

como por exemplo a resolução e o modo *sleep*. Os registradores de leitura trazem a informação com descrição do PAN3101 e os dados do “dx” e “dy”. Essas operações de leitura e escrita são compostas de 2 *bytes*, sendo o primeiro *bit* responsável por identificar se a operação é de leitura ou escrita, os 7 *bits* seguintes o endereço dos registradores e o último *byte* o dado que será escrito ou lido do registrador desejado.

A Figura 2.4 exemplifica esse protocolo de transmissão, realizando uma leitura do registrador. Basicamente esta é a única ação realizada no sensor pelo microcontrolador, onde ocorre apenas a modificação do primeiro *byte* correspondente aos registradores dx e dy, que no caso do PAN3101 tem como endereços 0x03 e 0x04, respectivamente. Deve-se lembrar que o primeiro *bit* (0) desse *byte* corresponde à ação de leitura. Após o envio deste *byte* o microcontrolador entra em alta impedância *(H_i-z)* esperando os dados do registrador requisitado.

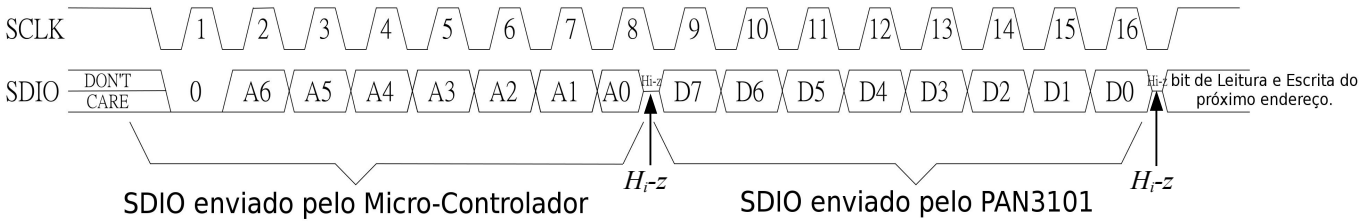


Figura 2.3 – Diagrama de Tempo

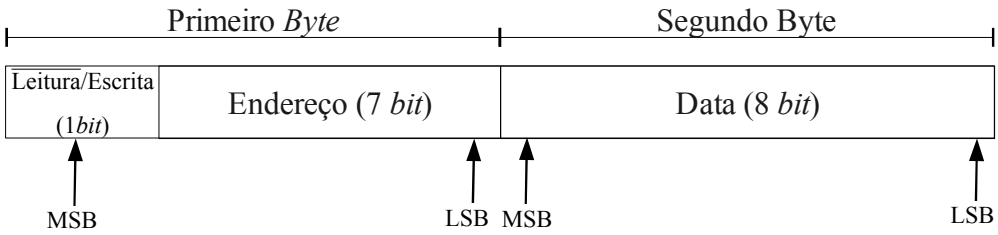


Figura 2.4 – Protocolo de Transmissão

Os dados deste sensor têm a informação da variação entre uma foto e outra, mas não a de posição, sendo que somando estas informações a cada instante tem-se o deslocamento total.

2.3.4 – Resolução

Os valores “x” e “y” obtidos pelo sensor do *mouse* têm suas unidades em pontos. Para obter a distância deslocada em medidas métricas usamos sua informação de resolução. Por exemplo, o *mouse* utilizado neste projeto tem uma resolução máxima de 800 dpi (*dots per inch*), ou seja, tem 800 pontos em uma polegada, logo cada ponto representa um deslocamento de 0,03175 mm.

A resolução do sensor e a velocidade em que os dados são apresentados no pino SDIO dependem do valor do *clock* inserido no pino SCLK. A escolha desse *clock* depende do valor do cristal conectado entre os pinos 1 e 2, e deve ter a frequência 4 vezes menor que a do cristal, segundo o *datasheet* do PAN3101. O circuito para o funcionamento do sensor pode ser visto na Figura 2.5. Este circuito foi obtido através da reutilização da placa do *mouse*. A placa apresentava um cristal de 24,58 MHz, logo o microcontrolador usado deve fornecer um *clock* de 6,1 MHz no pino SCLK, fazendo o sensor assumir uma resolução de aproximadamente 400 dpi, ou seja, 0,06 mm por ponto. A Tabela 2.3 apresenta dados importantes sobre as condições de operação deste CI.

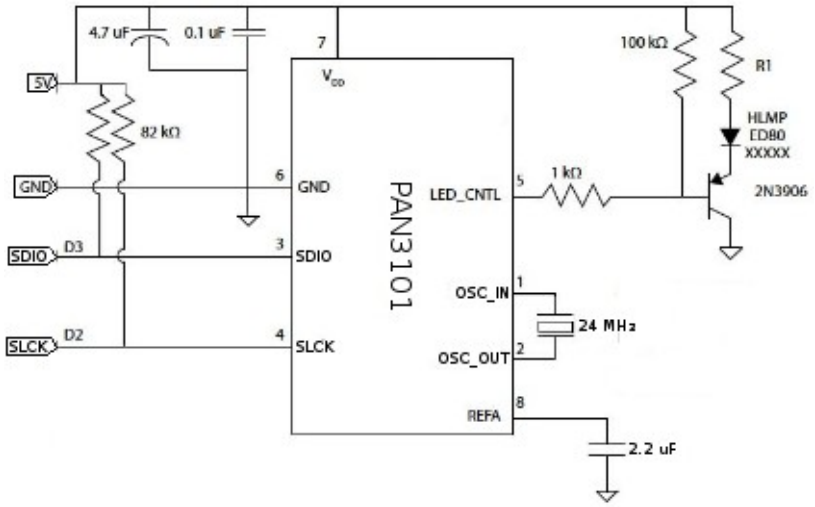


Figura 2.5 – Circuito para a utilização do sensor

Simbolo	Parâmetros	Min.	Típico	Máx.	Unidade
T_A	Temp. de Operação	0		40	°C
F_{CLK}	Frequência de *clock*		18,43	24,58	MHz
FR	Frame Rate		3000	4000	Frames/s
CLK	Serial Port Clock			$F_{CLK}/4$	MHz
S	Speed	0	16		inches/sec
A	Acceleration			3,9	g
R	Resolution		200	400	DPI

Tabela 2.3 – Condições de Operação Recomendadas

2.3.5 – Posicionamento

Na hora de fixar o sensor uma questão importante foi a posição em que este deveria ficar no RoboLogo. Ele deveria estar alinhado com a caneta e obter leituras de todos os movimentos executados.

Para a leitura de quando são executados os comandos para frente ou para atrás, não existe muita necessidade de uma posição específica para fixar, pois todos os pontos do RoboLogo se deslocam igualmente. Se o deslocamento é para frente ou para atrás 100 mm todos os pontos da superfície do RoboLogo irão 100 mm para frente ou para atrás. Logo o sensor iria ter a mesma leitura independente de onde fosse posicionado.

Mas, nos comandos que executam giros tanto para esquerda quanto para direita os pontos executam deslocamentos diferentes, que se traduzem em diferentes arcos com os seus centros localizados no ponto médio entre as rodas.

Como o sensor tem que obter leituras de todos os movimentos ele não pode estar no ponto médio entre as rodas (ponto A), que é o ponto que não se desloca quando uma roda gira para frente e a outra para atrás com a mesma velocidade, fazendo o RoboLogo girar em torno deste ponto. A Figura 2.6 ilustra tais movimentos.

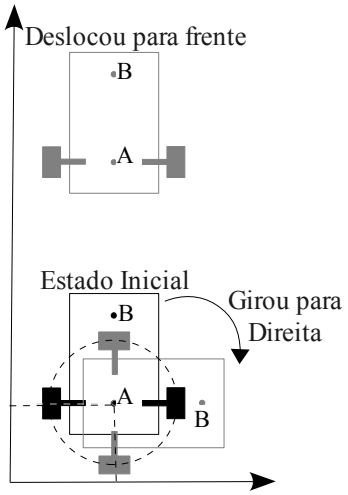


Figura 2.6 – Ilustração com os movimentos para frente e giro para direita

Ainda na Figura 2.6, pode-se observar que o ponto B, no mesmo alinhamento do ponto A, é sensível a todos os movimentos que o RoboLogo pode realizar. Deve-se observar que quanto maior o afastamento do ponto B, maiores serão os deslocamentos de giro ou de erro de trajetória, o que facilita a detecção pelo sensor e melhora a sensibilidade. A definição deste afastamento será vista na Seção 2.5.3. Esta distância será, também, levada em consideração nos cálculos de deslocamento, conforme será explicado no Capítulo 3.

2.4 – Caneta

2.4.1 – Objetivo

Permitir ao usuário riscar ou não a superfície enquanto o RoboLogo se desloca, segundo um comando.

2.4.2 – Funcionamento

Foi construído um sistema de acionamento utilizando um eletroímã que, ao ser alimentado, puxa uma alavanca metálica conectada a uma linha que levanta a caneta da superfície, como mostra a Figura 2.7.

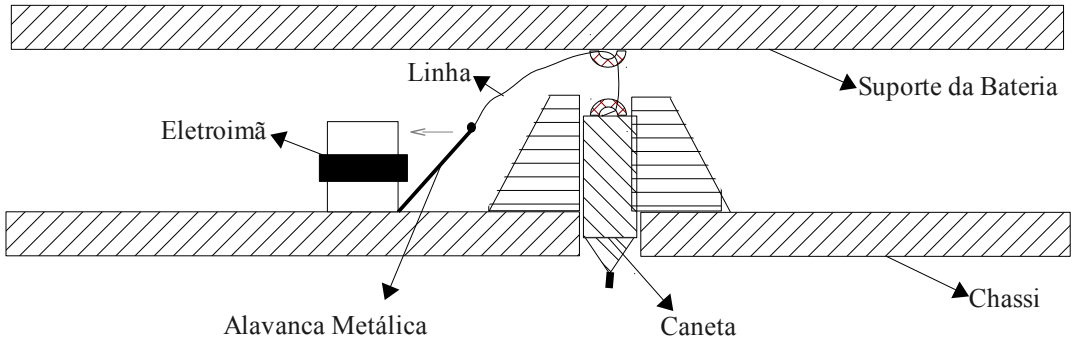


Figura 2.7 – Esquema do sistema acionador de caneta

Este eletroímã é acionado por um MOSFET que conduz corrente diretamente da bateria, quando recebe, do Arduino, o pulso que indica o recolhimento da caneta.

2.4.3 – Posicionamento

A caneta precisa riscar a superfície quando o RoboLogo se desloca para frente e para trás, mas quando executa os comandos que o giram para direita ou para esquerda ela não deve riscar, pois este movimento de girar em torno da caneta, sem que a mesma risque, é necessário para fazer vértices de figuras geométricas. Sendo assim o único ponto que atende a esta característica é o ponto A, conforme visto na Seção 2.3.5. É o ponto médio entre os eixos da roda, que não se desloca durante o movimento de giro, como está indicado na Figura 2.6.

2.5 – Base

2.5.1 – Objetivo

Servir de chassi para o RoboLogo onde os circuitos, motores, caixa de redução e bateria serão fixados.

2.5.2 – Características desejadas

A base do RoboLogo deve ser plana, leve e forte, de forma a não se deformar com o peso, mas nem tanto a ponto de dificultar o trabalho de furar e fresar. Foram encontradas tais características nas carcaças de fontes de computadores, feitas de uma liga metálica fina e forte.

2.5.3 – Mecânica

Pode-se observar na Figura 2.8, que quanto maior a distância (R) entre o sensor de deslocamento (B) e o ponto médio entre os eixos (A), maior será a sensibilidade do sensor, pois para um mesmo ângulo, maior será a distância percorrida. Logo, o sensor foi fixado na maior distância possível dos motores, limitando-se ao tamanho da base, que foi definida de forma a não tornar o RoboLogo de difícil transporte.

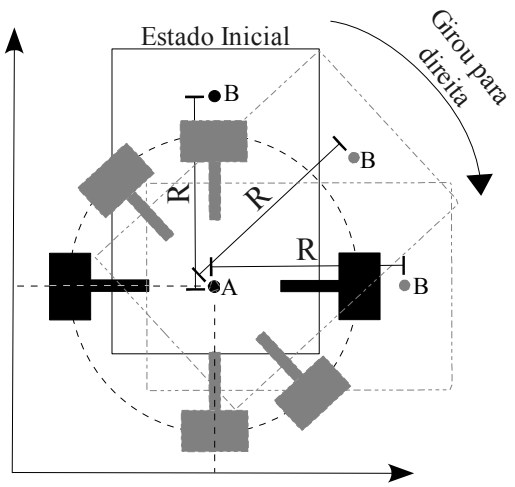


Figura 2.8 – Posicionamento do sensor

Como pode ser visto na Figura 2.9, o sensor ficou a uma distância de 52,8 mm da caneta. Isso proporciona um arco de 0,921 mm a cada grau de giro em torno da caneta. Esta distância, convertida para número de pontos que o sensor deverá ler para completar este o descrito pela movimentação do sensor (trajetória ponto B da Figura 2.8), fica em 15 pontos. Ou seja, sua sensibilidade é suficiente para corrigir eventuais erros nesta ordem de grandeza.

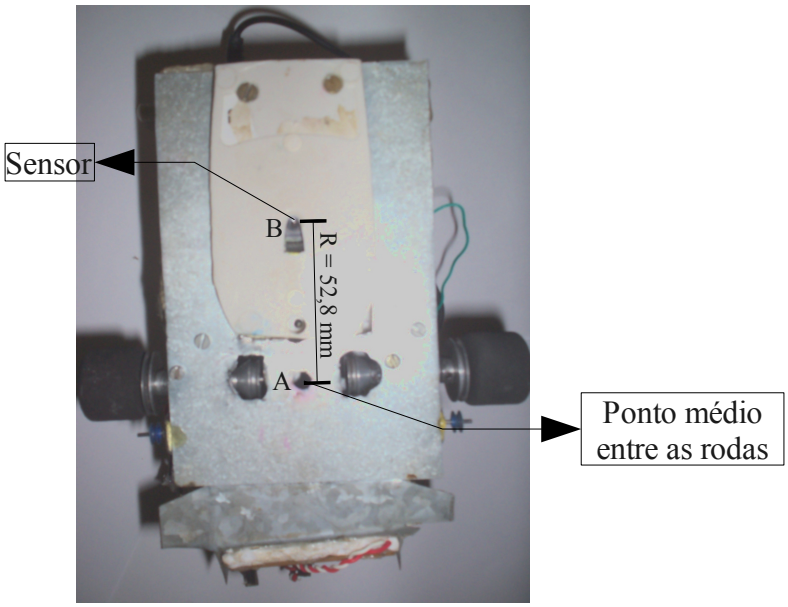


Figura 2.9 – Vista Inferior do RoboLogo

Outra necessidade importante é a de que o sensor deve ficar próximo à superfície em que se desloca. Para atender a essa necessidade fixou-se o sensor na parte inferior da base fazendo nela um corte para que os componentes do circuito do sensor pudessem atravessá-la.

Tendo fixado os motores e o sensor, as partes restantes já ficaram definidas, como por exemplo os eixos das rodas, que devem ficar ao lado dos motores ligados por uma correia, e o furo da caneta que deve ficar no ponto médio entre os eixos como citado na Seção 2.4.3.

Para a fixação dos diversos módulos na base foram feitas furações e cortes adequados as suas necessidades, como pode ser visto na Figura 2.10.

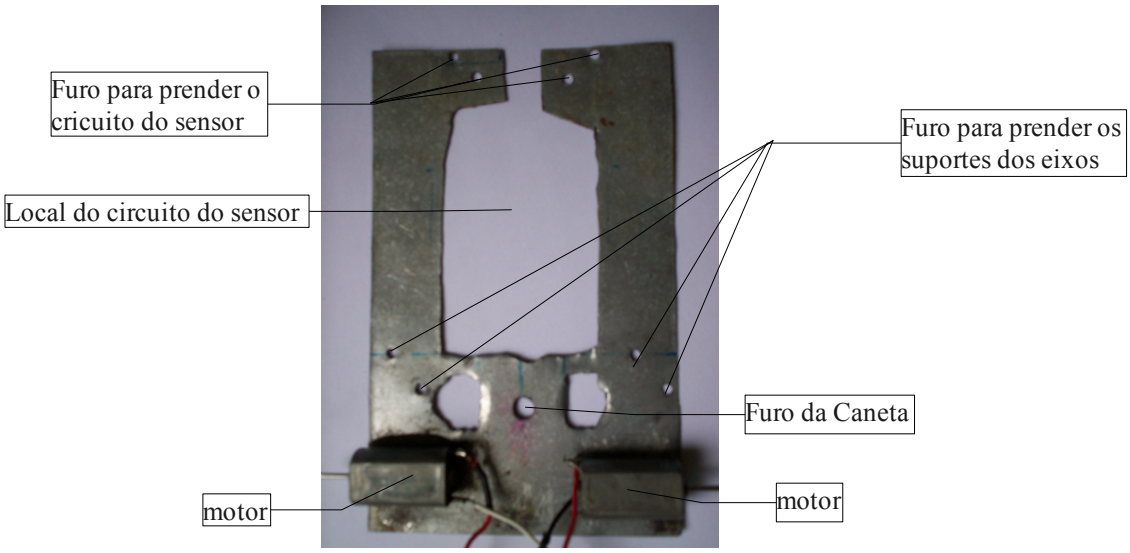


Figura 2.10 – Foto da Base do RoboLogo

2.6 – Comunicação RF

2.6.1 – Objetivo

Transmitir os comandos feitos pelo usuário no computador, para serem processados remotamente, sem a necessidade de fio, conferindo maior mobilidade ao RoboLogo.

2.6.2 – Funcionamento

Para realizar a transmissão RF foram adquiridos 2 transceptores, *Wireless RF transciver 915Mhz* da *Sure Electronics Co.* [9]. Eles trabalham com modulação em FSK (*Frequency Shift Keying*) na frequência de 915MHz e utilizam o protocolo RS232. Estes transceptores funcionam numa taxa de 19.200 bps, logo deve-se configurar todos os dispositivos para a comunicação serial nesta taxa.

A utilização deste transceptor é muito simples. Pode ser visto na Figura 2.11 que, apesar de serem transceptores, o que está conectado ao computador é apenas transmissor apenas e o conectado ao RoboLogo atua como receptor.

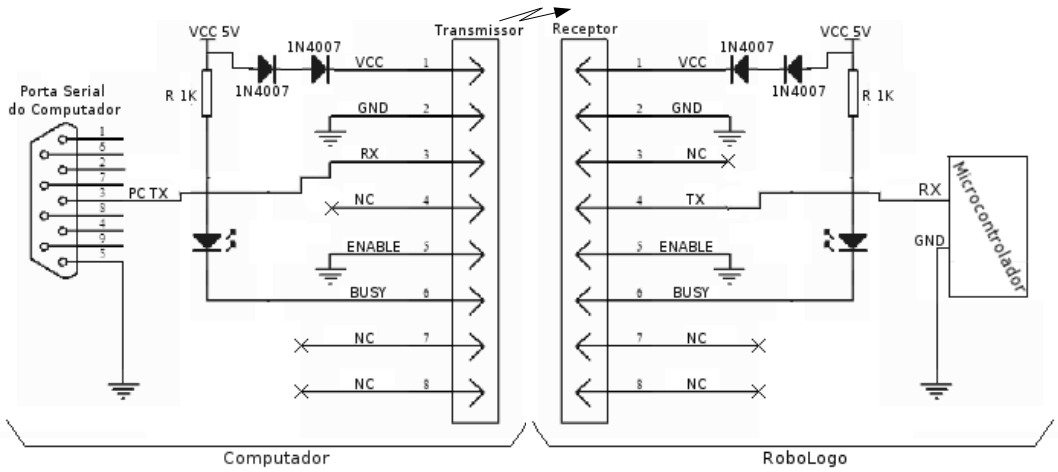


Figura 2.11 – Circuito dos módulos transmissor e receptor

2.7 – Microcontrolador

2.7.1 – Objetivo

É a peça central. Tem ligação com todos os módulos e é responsável pela "inteligência" do sistema. Ele é quem controla os motores DC via *PWM* interfaceado por um *driver* ponte-H, recebendo e processando os dados do sensor de deslocamento e do receptor RF (Radio Frequência).

2.7.2 – Escolha

Durante a investigação de qual sensor utilizar, cheguei a uma plataforma de desenvolvimento chamada de Arduino [7]. Ela é *open-source*, de fácil confecção e barata. Para montá-la se gasta em torno de R$25,00, e na internet tem uma comunidade bastante ativa [7], disponibilizando diversas bibliotecas para poder trabalhar com motores DC (*PWM*) e o sensor do *mouse* PAN101.

Sua programação é feita em C++ e as funções são de fácil uso, apesar de, às vezes, limitadas. Para gravar não é necessário circuito extra, é só conectar a placa na porta serial do computador e pronto.

Todos esses fatores foram importantes para a escolha desta *IDE (Integrated Development Environment)*, que facilitou bastante o desenvolvimento do projeto.

2.7.3 – Arduino

O Arduino é um projeto de código aberto, com circuito baseado em microprocessador Atmel, contendo uma placa física baseada em um circuito de entradas/saídas simples e um *software* próprio que funciona como ambiente de desenvolvimento para seu *firmware*.

O Arduino pode ser usado para desenvolver objetos interativos, recebendo entradas de várias chaves ou sensores, e controlando uma variedade de luzes, motores, mecanismos, entre outras saídas. Os projetos do Arduino podem tanto ser autônomos, como se comunicar com outros *softwares*.

As placas podem ser montadas à mão ou compradas; o código fonte e a *IDE* (Integrated Development Environment) são livres, podendo ser obtidos no próprio site do Arduino [7].

Como o Arduino é um produto de código aberto, as empresas de desenvolvimento de *hardware* muitas vezes desenvolvem as suas próprias plataformas de desenvolvimento, por exemplo, o Tatuino, Severino, Freeduino, etc, mas todos são compatíveis entre si. Na Figura 2.12 é apresentada a foto de uma placa comercial.

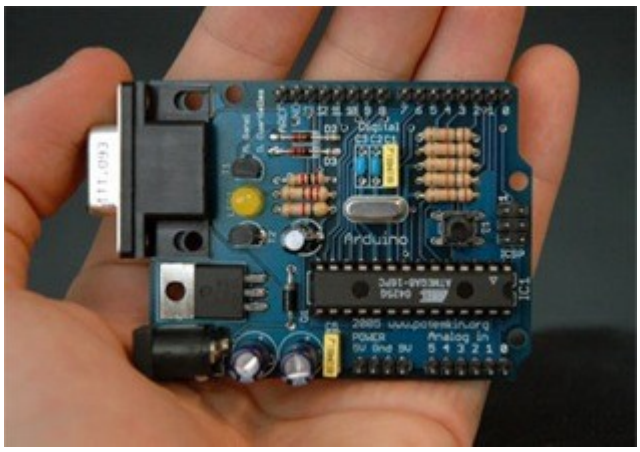


Figura 2.12 – Foto do Arduino [7] (Severino).

O Arduino é uma excelente placa de desenvolvimento para quem precisa fazer projetos de hardware. Já existe na internet uma série de bibliotecas (C++) que podem ser incluídas no código fonte para as mais variadas funções, por exemplo, Serial, Interrupção de Timer, Manchester (RF), etc. O Arduino utilizado neste projeto segue as especificações:

- Microcontrolador Atmega168;
- Tensão de Alimentação = 5V;
- Tensão de Entrada (recomendada) = 7-12 V;
- Tensão de Entrada (limite) = 6-20 V;
- 14 Pinos Digitais de I/O (6 deles são saídas PWM);
- 6 Pinos de Entradas Analógicas;
- Corrente Contínua de Pinos I/O = 40 mA;
- Corrente Contínua de Pino 3.3V = 50 mA;

- Memória Flash 16 KB (2 KB usado pelo *bootloader*);
- SRAM 1 KB;
- EEPROM 512 *bytes*;
- Velocidade do Clock 16 MHz.

Interessante é que o Arduino não necessita de nenhum *hardware* adicional para fazer a gravação no microcontrolador. Ele já vem com um *BootLoader*, capaz de receber todo o seu programa via serial.

Esta placa foi confeccionada através de instruções encontradas na página www.arduino.cc, lugar onde também podem ser encontradas diversas bibliotecas e exemplos. Foi nesta mesma página da internet onde foram encontradas referências de como utilizar o sensor do *mouse*.

No Arduino serão tratados diversos módulos do projeto: o controle dos motores através de sinais *PWM*, a leitura do sensor de deslocamento, e o processamento dos comandos.

2.7.4 – Funcionamento

O Arduino conecta-se aos diversos módulos através dos seus pinos de saída/entrada digital, que, ao todo, somam-se 14 pinos, nomeados de 0 a 13, sendo que alguns têm funções pré-determinadas. A Figura 2.13 é o diagrama em blocos do RoboLogo indicando as conexões entre os módulos do projeto com definições de entrada e saída.

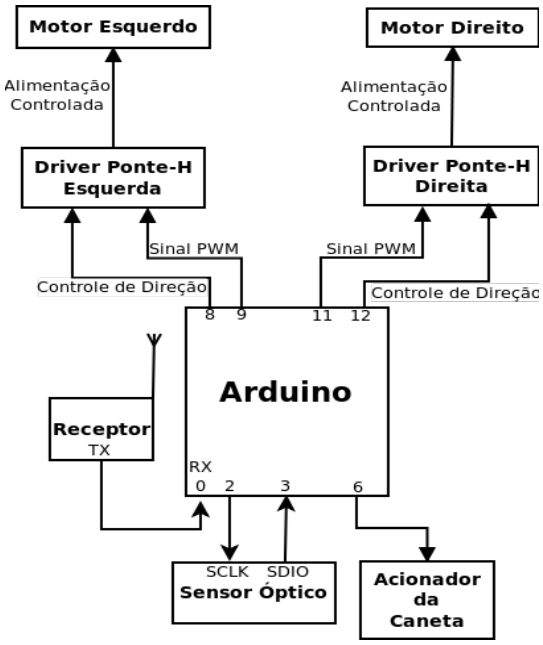


Figura 2.13 – Croqui das conexões entre os módulos

O pino "0" do Arduino pode ser entrada ou saída digital ou entrada serial RX padrão RS232. Estas opções são selecionadas através de um *jumper* localizado na placa.

A escolha da comunicação serial se justifica pelos seguintes motivos: a porta paralela é uma tecnologia obsoleta, que está cada vez mais rara nos computadores novos; a existência de diversos adaptadores serial para *USB* que são encontrados no mercado e o fato de já existir, no Arduino, um *buffer* para o protocolo RS232, que permite o argumento de cada novo comando enquanto espera a sua vez de ser executado. Assim, são evitados problemas de fluxo de informação, caso chegue um novo comando ao RoboLogo, antes do término da execução do comando anterior. O *buffer* do Arduino armazena até 128 bytes, o que é o suficiente para o presente projeto.

Os pinos do Arduino que podem trabalhar com a função geradora de *PWM* são: 9, 10 e 11. Para fazer cada motor funcionar é necessário um pino gerador de sinal *PWM* e outro para indicar o sentido do motor. Foi feita uma escolha de modo que estes pinos fossem adjacentes para cada motor, apenas para facilitar a sua localização e documentação, logo não poderia ser escolhido o pino 10 já que seus adjacentes são os outros pinos que trabalham com o sinal *PWM*. Escolheu-se os pinos 9 e 8, para serem respectivamente o sinal de controle *PWM*, e controle de direção do motor esquerdo, e os pinos 11 e 12 para serem respectivamente o sinal de controle *PWM*, e controle de direção do motor direito.

Os pinos 2 e 3 foram ligados ao sensor do *mouse*, respectivamente nos pinos SCLK e SDIO do CI. Poderiam ser escolhidos quaisquer outros pinos digitais livres, mas foram escolhidos os pinos 2 e 3, porque se localizavam distantes daqueles que controlam os motores, e assim não ficam tantos fios concentrados em um só lugar, o que facilita na hora de manuseá-los.

O pino 6 foi utilizado para acionar a caneta, pois estava a uma distância boa dos pinos de controle dos motores e do sensor do *mouse*, e assim, diminuindo a possibilidade de, ao manusear os outros pinos, o pino 6 se solte acidentalmente.

Todos os demais pinos de sinais digitais que não foram utilizados (pinos 1, 4, 5, 7 e 13) ficam livres para implementações futuras, assim como os 6 analógicos que se encontram na IDE Arduino.

Capítulo 3

Desenvolvimento do *Software*

3.1 – Firmware

3.1.1 – Objetivo

É o *software* embarcado, que roda no Arduino, que define a lógica e o algoritmos dos processamentos. Ele é responsável pela interpretação dos comandos e a correções dos erros.

3.1.2 – Funções

Para a programação do *firmware* foi necessário criar novas funções e utilizar funções específicas da plataforma Arduino. Para maior entendimento, construiu-se o digrama estruturado de função, representado na Figura 3.1, e uma lista de funções utilizadas. Nesta lista, foram explicados o funcionamento das funções específicas do Arduino e, nas funções criadas para o presente projeto, além da explicação, foi feita uma descrição sucinta da lógica de seu algoritmo.

O código deste *firmware* devidamente comentado segue no Apêndice A deste projeto.

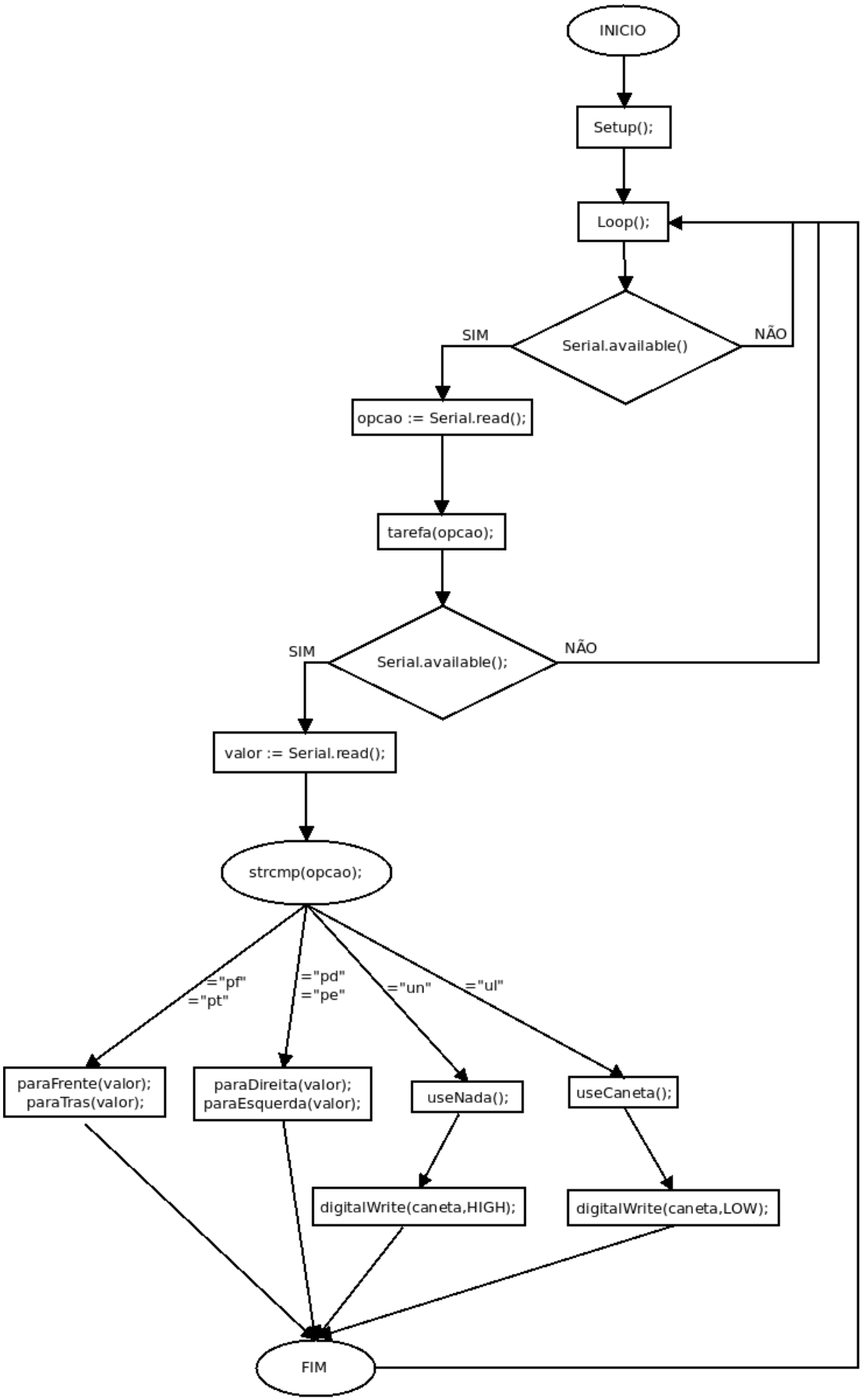


Figura 3.1 – Principal diagrama de função do *firmware*

Funções do Arduino utilizadas no projeto:

A função *setup*() é exigida para o funcionamento do Arduino, e todo o seu conteúdo será executado apenas uma vez, quando o Arduino é ligado. É considerada como "programa de configuração". E é nela que se configura a velocidade de comunicação serial, os modos dos pinos e se inicializa o sensor e as variáveis.

A função *Serial.begin*(*int*) configura a comunicação serial para a velocidade de 19.200 bps. Esta velocidade foi escolhida por ser compatível com o receptor RF.

A função *Optical1.begin*() inicializa o sensor do *mouse*.

A função *pinMode(pino, mode)* configurados os "modos" dos pinos a serem utilizados pelo programa.

As funções *digitalWrite(pino, valor)* servem para enviar sinal para o pino designado, sendo o valor definido como HIGH ou LOW, que caracterizam 5 Volts ou zero respectivamente.

A função *analogWrite(pino, valor)* gera sinal PWM de frequência 480Hz no pino escolhido. O Arduino só suporta sinais PWM nos pinos 9, 10 e 11, e o valor do argumento varia de 0 à 255, que corresponde ao *duty cycle,* variando a tensão média de 0 a 5 volts, como pode ser observado na Figura 3.2.

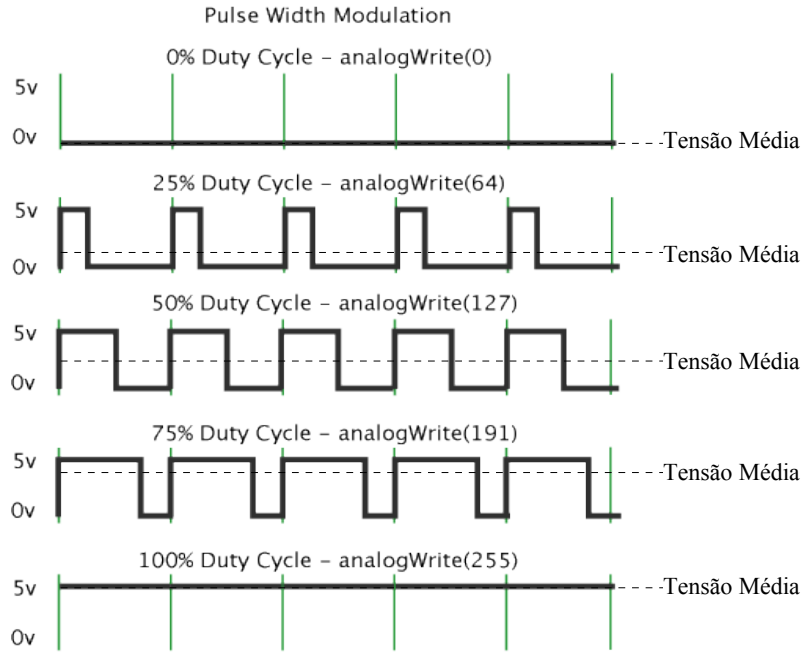


Figura 3.2 – Sinal gerado pela função *analogWrite*();

A função *loop(),* exigida pelo Arduino*,* é responsável por todas as modificações e comandos em tempo real do RoboLogo, e como o próprio nome diz é um laço que fica repetindo o tempo todo, considerado como o corpo principal do programa.

A função Serial.available() é responsável por verificar se existe algum dado na pilha serial do Arduino. Os dados ali contidos são os recebidos pelo transmissor RF.

A função Serial.read() faz a leitura da porta serial *byte* a *byte*, depois de constatar que existe algum dado na pilha.

A função *Optical1.updateStatus() é* responsável por atualizar os registradores "dx" e "dy" do sensor.

As funções *Optical1.dx()* e *Optical1.dy()* retornam os valores dos registradores "dx" e "dy" do sensor.

Funções criadas

A função *tarefa*() pega os dados recebido pelo pela pilha serial que é passado para a variável chamada de *opcao* e, dependendo dela, chama os respectivos comandos, que são representados pelas funções paraFrente(), paraTras(), paraDireita(), paraEsquerda(), useCaneta() e useNada().

A Função paraFrente() (ver diagrama da Figura 3.3) recebe como argumento um número inteiro representando quantos pontos o RoboLogo deve caminhar para frente. O usuário não necessita fazer a conversão de pontos para distância em mm, essa conversão é feita automaticamente quando a função tarefa() lê os valores dos argumentos das funções. Essa conversão não é complexa, basta pegar o valor que o usuário digitou em mm e multiplicá-lo por um fator de 0.06, pois como vimos na Seção 2.3.4 a resolução do sensor é de 0.06 mm por ponto. Como o sistema pode já ter caminhado por algum comando anterior, esse valor é somado com o valor inicial do sistema, tendo assim sempre armazenado o valor absoluto do deslocamento total que indicará o ponto de parada para este comando. Em seguida, é dado o comando para os motores girarem ambos no sentido para frente com uma velocidade determinada pelo PWM.

Durante todo o deslocamento para frente, é feita uma comparação do número de pontos que deve ser deslocado (armazenado), com os valores acumulados lidos pelo sensor. Essa leitura é feita atualizando os registradores "dx" e "dy" que representam o quanto se deslocou a cada instante no eixo x e y respectivamente, depois esses valores são acumulados em uma variável, pois o importante sempre é o total deslocado até o momento.

Nesta função paraFrente() a comparação feita com o valor deslocado em y é para identificar se o RoboLogo chegou ao seu destino. Se não chegou ainda ele continua se deslocando para frente, se chegou ele para, aplicando valor zero no PWM, se passar ele volta, invertendo os motores. A comparação feita com o valor deslocado em x é para correção de seu trajeto, pois se o RoboLogo vai se deslocar em linha reta para frente, apenas os valores de y devem mudar. Se houver alguma mudança nos valores de x maior que a resolução do sensor, ou seja, acima de 0.06 mm é feita uma correção alterando os valores de PWM de cada motor, aumentando o de um e diminuindo o do outro, de forma que ele volte para o trajeto original. Essas alterações nos valores de PWM tem que respeitar os valores máximos e mínimos que a função *analogWrite(pino, valor)* aceita.

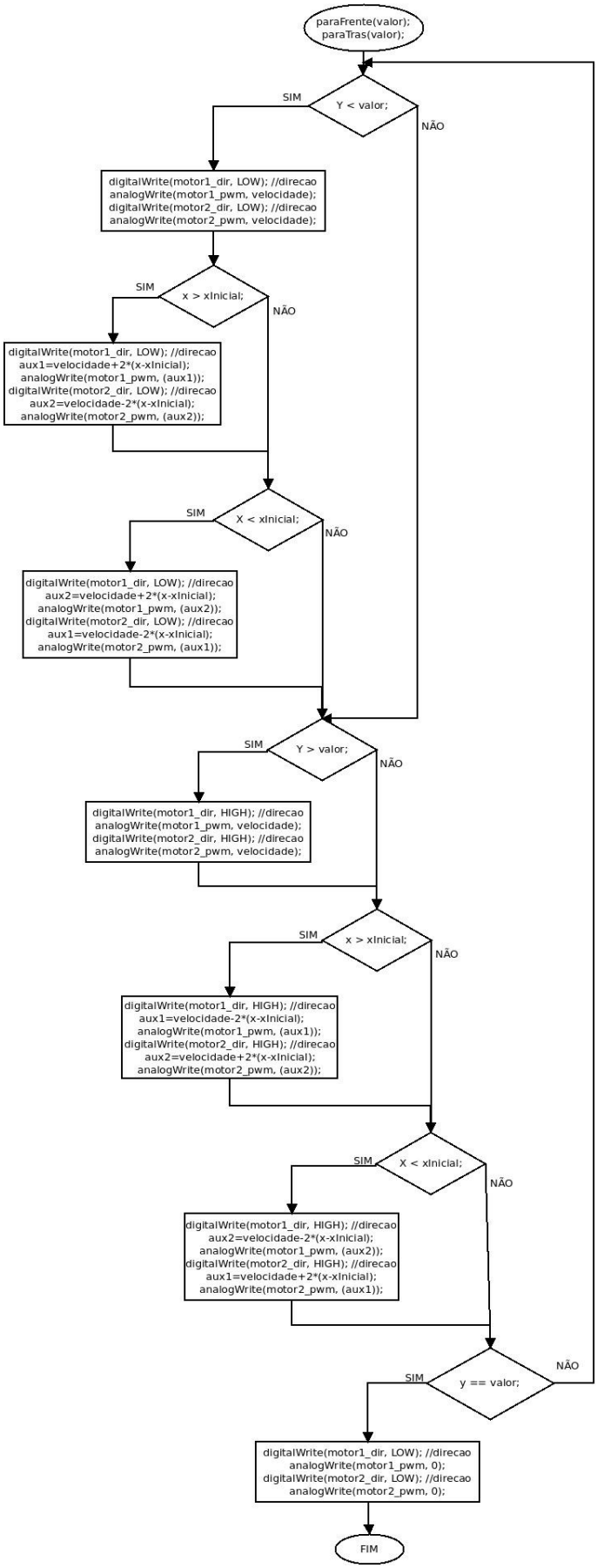


Figura 3.3 – Diagrama da função paraFrente();

A função paraTras(), segue o mesmo algorítimo da função paraFrente(), no entanto em vez do deslocamento ser somado ao valor de y inicial, ele será subtraído e a comparação que comanda a parada do movimento muda o sinal.

A função parDireita(), recebe como argumento um número inteiro representando quantos graus o RoboLogo deve rotacionar para direita. Este movimento é percebido pela variável x, pois o sensor está sempre se deslocando neste sentido, sem ir para frente descrevendo uma circunferência, o que para o RoboLogo é uma rotação entorno do ponto médio entre os eixos.

A Figura 3.4 exemplifica o movimento do sensor girando para direita 90 graus, evidenciando o estado inicial, quando chega a metade do percurso e, finalmente, quando chega ao seu destino. Percebe-se que o sensor se desloca sempre para direita (vista superior), mantendo a mesma distância (R) do centro dos eixos, local onde a caneta está fixada. O eixo dy não apresenta nenhuma variação, apenas o eixo dx. Para calcular o quanto ele se desloca até chegar a 90 graus basta fazer a seguinte regra de três: 360º está para 2πR (R =52,8 mm) assim como 90º está para o valor dx que o sensor deve ter.

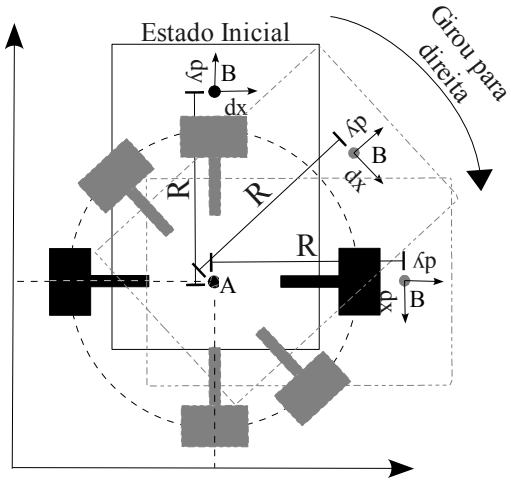


Figura 3.4 – RoboLogo girando entorno do ponto médio entre os eixos.

Dessa maneira o valores de graus são convertidos em pontos no eixo x, e somados ao um valor inicial, pois o sistema pode já ter caminhado por algum comando anterior, tendo assim sempre armazenado o valor absoluto do deslocamento total que indicará o ponto de parada para este comando. Em seguida, é dado o comando para o motor da esquerda girar para frente e o da direita para atrás com uma velocidade determinada pelo PWM.

Assim como a função paraFrente(), durante todo o deslocamento para direita do sistema, é feita uma comparação do número de pontos que deve ser deslocado (armazenado), com os valores acumulados, lidos pelo sensor, sendo que nesta função a comparação feita para identificar se o RoboLogo chegou ao seu destino é feita com a variável x. Se não chegou ainda, ele continua mandando ir para direita, se chegou ele para, aplicando valor zero no valor do PWM, se passar ele volta, invertendo o sentido de cada um dos motores.

A comparação feita com o valor deslocado em y é para a correção de seu trajeto, pois se o RoboLogo vai se deslocar em linha reta para direita, apenas os valores de x devem mudar, se houver alguma mudança nos valores de y durante o trajeto é feita uma correção alterando os valores de PWM de cada motor, aumentando o de um e diminuindo o do outro, de forma que ele volte para o trajeto original. Essas alterações nos valores de PWM tem que respeitar os valores máximos e mínimos.

A função paraEsquerda(), segue o mesmo algorítimo da função paraDireita(), no entanto em vez de ser somado o deslocamento ao valor de x inicial, será subtraído e a comparação, que comanda a parada do movimento, muda o sinal. Na Figura 3.5 pode ser visto o seu diagrama.

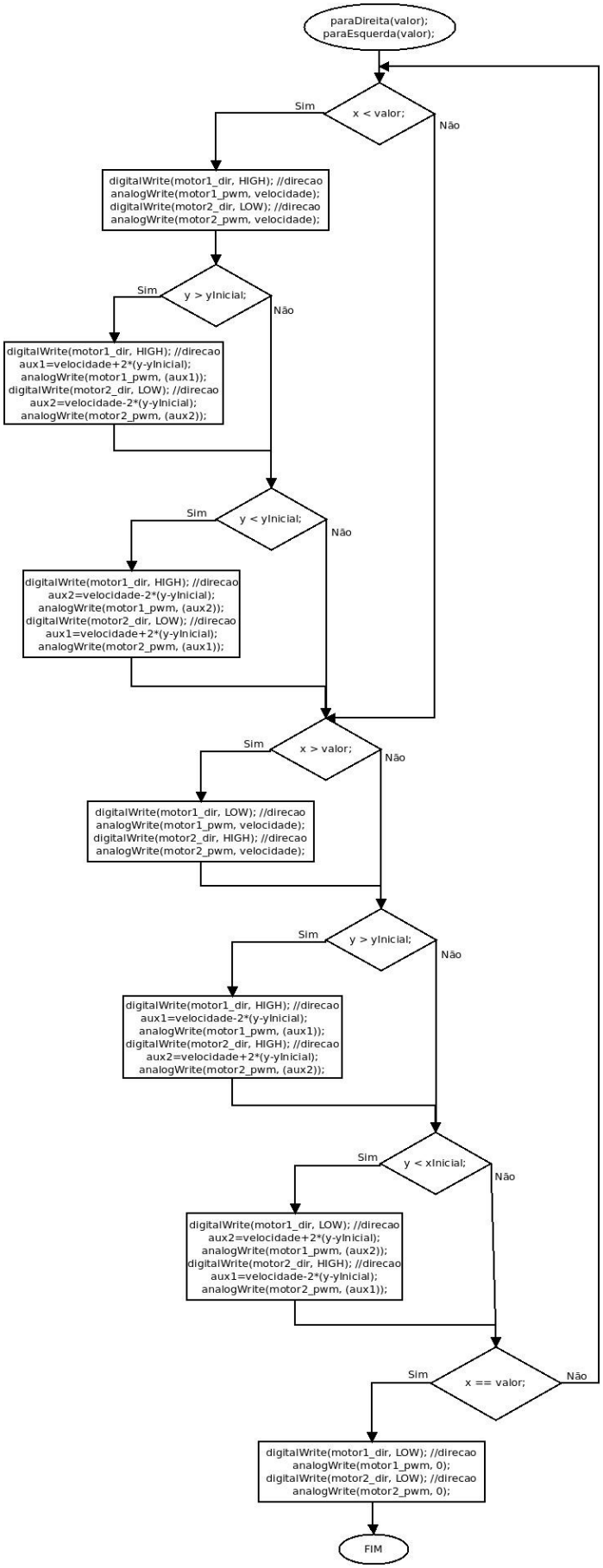


Figura 3.5 – Diagrama da função paraDireita();

3.2 – Software SuperLogo

3.2.1 – Objetivo

O SuperLogo [10] fornece ao usuário um ambiente amigável em que ele pode executar os comandos que serão transmitidas ao RoboLogo via porta serial.

3.2.2 – Escolha

Essa plataforma foi escolhida, porque possui diversas funções para se trabalhar com automação. Ela também é utilizada pela LEGO [11] para controlar seus kits robóticos, demonstrando a existência de um certo reconhecimento para essa aplicação. O fato de ser uma ferramenta gratuita e traduzida para o português também são fatores importantes.

3.2.3 – Funcionamento

Foi criado um *script* para interpretar seis novos comandos: pfr, ptr, pdr, per, unr e ulr, que na realidade são os comandos existentes do programa acrescentado da letra "r" (de robô), eles representam as funções paraFrente(), paraTras(), paraDireita(), paraEsquerda(), useNada() e useCaneta().

Ao iniciar o programa devemos abrir a porta serial para transmitir os dados, para isso usamos um comando do SuperLogo chamado "abraporta" seguido da porta serial que esteja sendo utilizada, exemplo: abraporta "com5. Que significa que a porta serial utilizada é a porta de comunicação COM 5.

Após esse procedimento pode-se utilizar os comandos. Eles enviam, para essa porta, dados na forma de caracteres que representam as funções utilizadas pelo *firmware* e assim, como seus devidos argumentos, neles também é feita uma sincronia para que o desenho que é apresentado na tela seja correspondente ao que é feito pelo RoboLogo. Todo os comandos respeitam a lógica da linguagem LOGO, podendo utilizar laços de repetição, operadores condicionais e outros recursos da linguagem.

Essa programação do *script* que interpreta os novos comandos está disponível no Apêndice B deste documento.

3.2.3 – Atividades para o LOGO

Como exemplos de atividades para utilização do programa podemos citar duas figuras; o quadrado e o círculo, que apesar de simples, trabalham com problemas de ângulos, raio, circunferências e etc.

O quadrado é bem simples, para reproduzí-lo basta mandar o RoboLogo ir para frente e girar 90º para um dos lados 4 vezes. Podemos criar a figura de duas formas, uma repetindo esses comandos manualmente e a outra usando um laço de repetição. Essa última forma pode ser pedida aos alunos como desafio, após apresentar a primeira. O código para as duas atividades são respectivamente "$ pfr 100 pdr 90 pfr 100 pdr 90 pfr 100 pdr 90 pfr 100 pdr 90" e "$ repita 4 [pfr 100 pdr 90]".

O simbolo "$" representa o campo de execução dos comandos. Basta digitar todos esses comandos na mesma linha na área chamada "Janela de Comandos" do programa SuperLogo e pressionar "ENTER".

Uma outra atividade é a de traçar uma circunferência e para isso optamos por aproximá-la por um polígono de 36 lados. Pois, para ângulos pequenos, o lado do polígono se confundiria com o arco da circunferência. Assim, basta mandar o RoboLogo ir para frente um valor e girar para direita um pequeno ângulo repetidas vezes. Por exemplo, para um ângulo de 10º o arco correspondente pode ser calculado através de uma simples regra de três. Sabemos que um círculo tem circunferência de 2*π*R, o valor de R é o raio da circunferência que se deseja traçar, e isso está relacionado a um ângulo de 360º, logo para 10º o arco será (2*π*R)/36. Então, para construir uma circunferência de raio R devemos andar (2*π*R)/36 para frente e realizar um giro de 10º para direita, isso repetido 36 vezes, o que resultará em 360º ao todo. O código fica "$ repita 36 [pfr (2*pi*R)/36 pdr 10]".

Com essa atividade podemos abordar diversos temas relacionado a círculos, ângulos, arcos, etc. Existem outros comandos e recursos da linguagem SuperLogo que possibilitam a manipulação de varáveis, tornando o *software* bem interativo, como por exemplo, fazer o programa perguntar qual o raio que se deseja inserir. Este e outros recursos que são inerentes da linguagem também funcionam no RoboLogo.

Capítulo 4

Resultados

4.1 – Testes

Durante o processo de desenvolvimento muitos testes foram realizados para determinar os padrões de funcionamento do RoboLogo.

Os itens a seguir exemplificam alguns desses testes considerados críticos e que deram resultados importantes para compor a lista de características do RoboLogo.

4.1.1 – Teste de superfície

Um teste de grande importância, foi o de identificar, para qual superfície o sensor responderia melhor. Nos testes foram executados comandos no RoboLogo e através de um cabo ligado ao computador foram verificados os dados do sensor. Executando o mesmo comando duas ou mais vezes no RoboLogo, ele deveria responder da mesma forma a cada um deles.

O resultado destes testes indicaram que superfícies brilhosas, como por exemplo a lousa branca para canetas hidrocor, atrapalham a leitura do sensor, fazendo com que o sistema respondesse de forma errônea, e já papéis do tipo cartolinas, por normalmente serem guardadas enroladas, suas superfícies apresentam dobras e desníveis que também deixam todo o sistema suscetível a erros.

A melhor superfície para trabalhar foi a placa de divisória branca, pois é fosca e rugosa, também é leve e fácil de apagar para sua reutilização.

4.1.2 – Teste de torque

Outro importante fator a ser escolhido que influencia diretamente o torque e a velocidade, é o PWM. Para escolher qual valor, entre 0 e 255, usar, foi feita uma série de testes para balancear o fator velocidade, que não pode ser alto, pois influencia

diretamente no erro de leitura do sensor, e o torque que tem que ser grande o suficiente para vencer a inércia dos movimentos.

Os resultados deste teste são os valores de PWM. Estes valores são carregadas na inicialização do RoboLogo. Foi percebido, também, que para a execução de trajetos retilíneos e giros, os valores de PWM devem que ser diferentes pois, para realizar giros seu momento é menor, necessitando de menos torque para ser realizado. Utilizando o mesmo valor de PWM para os dois tipos de movimento, ou um não era realizado ou simplesmente o outro era muito rápido. Através de teste empírico definiu-se para execução de trajeto retilíneos valores de PWM entre 190 e 240 e para execução de giro valores entre 190 e 240.

4.1.3 – Centralização da caneta

Na realização de giros, o risco de caneta na superfície deveria ser um ponto. Entretanto isso só ocorre se a caneta estiver corretamente centralizada entre os eixos das rodas, como a furação do chassi do RoboLogo apresenta uma pequena folga, permite pequenos deslocamentos da caneta, acarretando erros no desenho.

Outro fator que atrapalha, na solução adotada, é a largura das rodas do RoboLogo, que torna difícil determinar onde é o centro real dos movimentos de giro, pois este centro depende do ponto de contato da roda com a superfície. Como os pneus apresentam ressaltos, o ponto central acaba sendo variável. Estes erros são vistos nos riscos dos movimentos.

4.1.4 – Teste de pilha serial

Como citado no Capítulo 3, os comandos executados pelo SuperLogo são enviados para a pilha do Arduino, e o RoboLogo vai executando cada um deles lendo a pilha. Foi então executado um comando em que o RoboLogo iria demorar a executar, como por exemplo andar uma distância grande, e seguido de uma sequência de outros comandos, no intuito de ver quando iria estourar a pilha.

Quando os dados passaram de 126 bytes houve o estouro de pilha e o sistema parou antes de terminar o resto dos comandos, ou seja, os dados após o estouro da pilha

são perdidos. Este problema foi solucionado colocado um tempo de espera no script do SuperLogo, antes da transmissão, dando o tempo suficiente para evitar o estouro da pilha. Deve-se observar que esta solução não elimina a possibilidade de estouro, que ocorrerá somente se o RoboLogo executar um trajeto muito grande, maior que a superfície definida, que está limitada em 1 m x 1 m para não dificultar sua portabilidade.

4.2 – Resultado Final

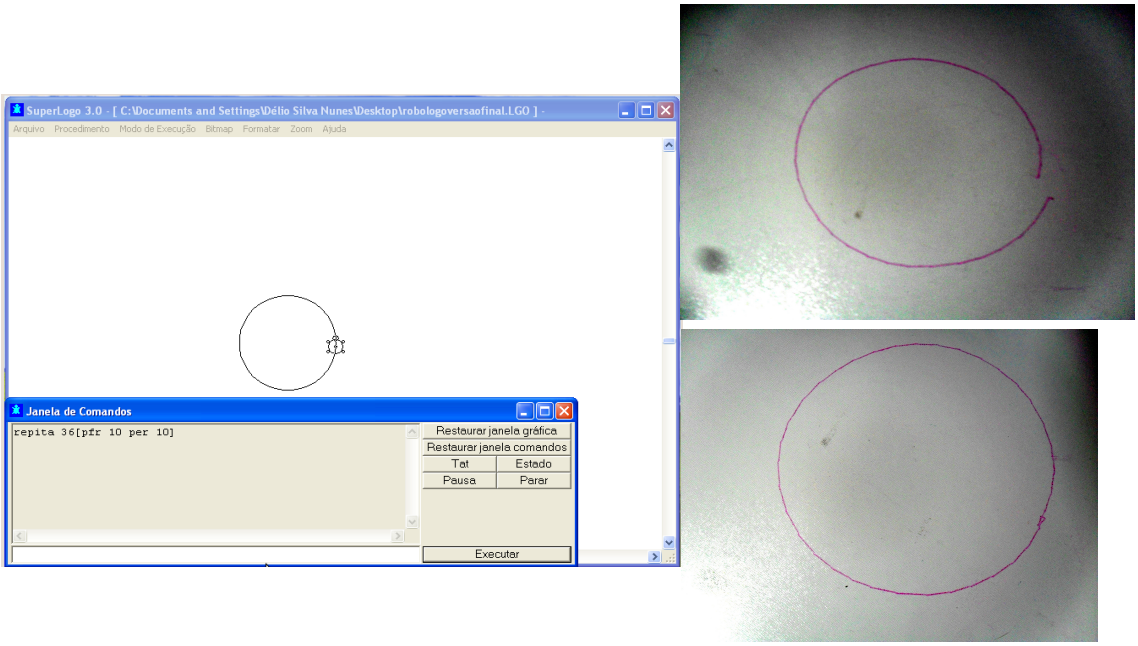
Como produtos deste projeto foram desenvolvidos quatro artefatos sendo dois *hardwares*, o RoboLogo e o transmissor que liga ao computador, e dois *softwares*, o programa de interface do SuperLogo e o *firmware.*

O RoboLogo responde a todos os comandos propostos no projeto de forma satisfatória, representando a figura que também é desenhada na tela do programa SuperLogo. Esses comandos respeitam a sintaxe do LOGO e possuem validações em suas entradas.

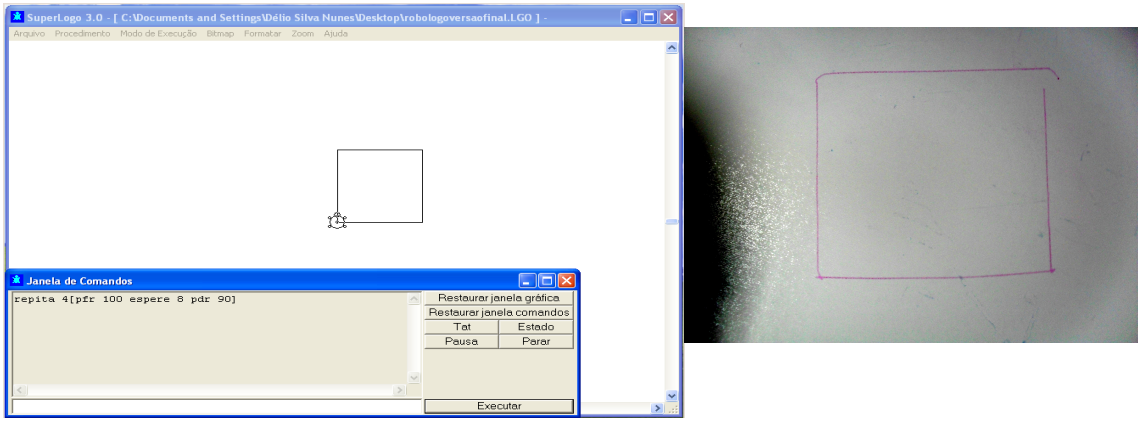
A transmissão sem fio não apresenta erros até um raio de 10 metros, o que é suficiente para ser utilizado em uma sala de aula.

O sistema de correção de erro funciona com uma boa precisão, tanto nas execução de trajetos retilíneos como curvos. O sistema acionador da caneta escreve ou não na superfície, conforme o comando executado.

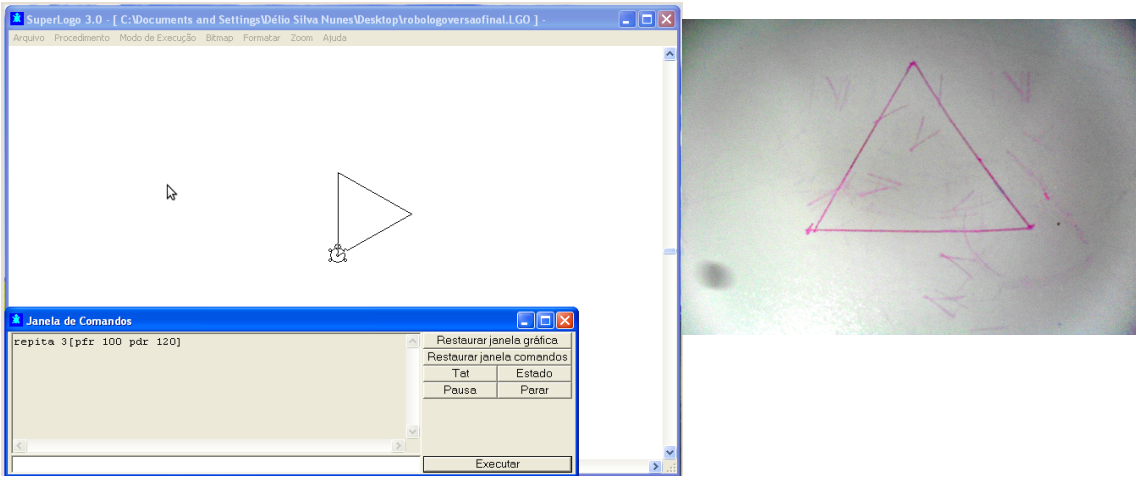
Pode-se observar na Figura 4.1 que o mesmo comando executado ora responde corretamente, ora não. Estes erros são associados à falta de robustez do projeto mecânico, como a descentralização da caneta e a variação do ponto de apoio das rodas. As Figuras 4.2 e 4.3 exemplificam outros comandos executados pelo RoboLogo.



Figuras 4.1 – Execução de um círculo com o comando $repita 36[pfr 10 pdr 10]



Figuras 4.2 – Execução de um quadrado com o comando $repita 4[pfr 100 pdr 90]



Figuras 4.3 – Execução de um triângulo com o comando $repita 3[pfr 100 pdr 120]

Capítulo 5

Conclusão

Apesar do RoboLogo cumprir o seu papel principal que é executar os comandos enviados pelo SuperLogo, diversos problemas associados à mecânica surgiram durante os testes.

A mecânica foi feita de forma artesanal e intuitiva, e a influência dela sobre o projeto foi subestimada, pois foi interpretado que todos os problemas poderiam ser resolvidos através do processamento da leitura do sensor, o que não foi verdade, pois como o posicionamento preciso da caneta e do sensor são importantes para a precisão do desenho, qualquer variação durante o movimento pode acarretar em leituras erradas e desenhos distorcidos.

Todos esses erros influenciaram na precisão do desenho, incluindo o inerente à leitura do sensor, como pode ser observado nas Figuras 4.1 a 4.3.

O projeto apresenta graves problemas na estabilidade mecânica, com peças muito frágeis e móveis. O sistema de acionamento da caneta está confinado em uma região de difícil acesso, o que dificulta a troca da mesma. Caso haja uma tentativa de reprodução em maior escala toda a mecânica deve ser reprojetada.

O código fonte do *firmware* é outra parte do projeto que pode evoluir. Durante a confecção deste documento foi percebido que existe muita informação redundante. A otimização do código poderia torná-lo mais simples de ser entendido por qualquer usuário que queira reproduzir o projeto ou contribuir para sua evolução.

A ideia de utilizar o sensor do *mouse* como sensor de deslocamento facilitou muito o trabalho, mas restringe muito a superfície que deve ser utilizada, pois a mesma tem que ser lisa e opaca, pois a sensibilidade do *mouse* varia muito dependendo destas características. O fato deste sensor precisar estar bem próximo à superfície em que se desloca, fez com seu suporte tivesse que encostar na superfície aumentando o atrito, algo que poderia ser contornado por um projeto mecânico que levasse esta característica em consideração.

O fato de dar prioridade à utilização de material eletrônico descartado mostrou ser um desafio, pois era necessário um conhecimento prévio de onde achar as peças, e

algumas vezes foi preciso gastar tempo para desmontar e procurar onde tinham certos tipos de materiais. Muitos testes foram realizados para cada peça encontrada, pois a mesma poderia estar avariada ou até mesmo não se adequar por alguma caraterística desejada.

Como o sistema é composto por dois *transceivers* RF bem robustos e um micro controlador, ele pode receber muito mais funcionalidades, para utilizar os demais pinos que restam no Arduino, sendo viável colocar uma câmera e receber imagens.

Fica claro que apesar de todos os problemas, o RoboLogo é uma ferramenta livre utilizada para fins educativos. Age como um catalizador de ideias, pode crescer em funcionalidade e melhorar seu desempenho, ou seja, serve como parâmetro para projetos futuros numa área tão pouco explorada pela comunidade científica, voltada ao aprimoramento do aprendizado em sala de aula.

Apêndice A

Código fonte do firmware do Microcontrolador

Código completo incluído no CD anexo.

Apêndice B

Código dos script do SuperLogo

Código completo incluído no CD anexo.

Bibliografia

[1] PEREIRA, L. P., "O Desenvolvimento Humano na Perspectiva de um Educador", http://www.iesb.br/psicologiaiesb/jan_2009/10_psicologiaiesb_pereira_2009_1.htm 2009, (Acesso em 15 Dezembro de 2009).

[2] PAPERT, S. LOGO: Computadores e Educação. São Paulo, Brasiliense, 1985.

[3] FINEP., "Chamada Pública MCT/FINEP/CT-PETRO – PROMOVE – 01/2008", http://www.finep.gov.br/fundos_setoriais/ct_petro/editais/PROMOVE %202008_versao_final.pdf, 2008, (Acesso em 15 Dezembro de 2009).

[4] MENDONÇA, M. L., CARVALHO, M. A., A Construção da Identidade do Professor como Profissional Reflexivo, http://www.ufpi.br/mesteduc/eventos/ii encontro/GT-1/GT-01-23.htm#_ftn1, 2007, (Acesso em 15 Dezembro de 2009).

[5] LAJONQUIÈRE, L. Piaget : Notes for a Constructivist Theory of Intelligence. Psicologia USP, São Paulo, v.8, n.1, p.131-142, 1997.

[6] Controle de um Motor DC pelo PWM do PIC, http://www.mecatronicaatual.com.br/ secoes/leitura/125/imprimir:yes, 2008, (Acesso em 15 Dezembro de 2009).

[7] ARDUINO, http://www.arduino.cc/, 2009, (Acesso em 15 Dezembro de 2009).

[8] MARTIJN, http://www.martijnthe.nl/2009/07/interfacing-an-optical-*mouse*-sensor-to-your-arduino/, 2009, (Acesso em 15 Dezembro de 2009).

[9] SURE ELECTRONICS CO., http://www.sure-electronics.com/, 2009, (Acesso em 15 Dezembro de 2009).

[10] NIED, Universidade Estadual de Campinas, http://www.nied.unicamp.br/softwares /software_detalhes.php?id=33, (Acesso em 20 Janeiro de 2010).

[11] ALMEIDA, Maria E., Lego-Logo e Interdisciplinariedade, http://www.casadaciencia.ufrj.br/Publicacoes/Artigos/EduBytes95/LegoLogo.htm, 2007, (Acesso em 20 Janeiro de 2010).