PROGRAMA INTEGRADO DE INICIAÇÃO CIENTÍFICA (PIC)

RELATÓRIO FINAL DO PERÍODO DE AGOSTO/2006 A JULHO/2007

PLANO DE TRABALHO DE INICIAÇÃO CIENTÍFICA

Leandro Augusto Fogolin Pereira, RA: 05356787

Faculdade de Engenharia de Computação

Aluno de iniciação científica: ( ) PIBIC/CNPq

(x) FAPIC/Reitoria

( ) VIC

Plano de trabalho: Estudo e Desenvolvimento de uma Biblioteca para Tratamento de Sinais de Sensores de Robôs Móveis

Prof. Dr. Juan Manuel Adán Coello

Faculdade de Análise de Sistemas

Projeto de pesquisa: Aprendizado com Classes Desbalanceadas: Tratamentos e Possíveis Soluções.
Grupo de pesquisa: Sistemas inteligentes

Linha de pesquisa: Inteligência Artificial

Líder do Grupo de Pesquisa: (X) Sim

1) Inserção do aluno de IC no Grupo de Pesquisa:

1. Plano de Trabalho

Este projeto faz parte de um maior com o intuito de desenvolver um robô auto-suficiente, capaz de mapear um ambiente e se locomover sem interação humana.

2. Participação do Aluno

Responsável pela parte mecânica e eletrônica do projeto, bem como co-responsável pela biblioteca de leitura e tratamento de sensores (e responsável pela interface gráfica de depuração do robô). No momento está projetando um sistema de controle alternativo ao do kit educacional utilizado, já que o incluso provou ser insuficiente para as necessidades do projeto. Em paralelo, está finalizando, junto com o Fernando Paolieri, a criação e integração de sensores de distância infra-vermelho e ultra-som, necessários tanto para o mapeamento e navegação.

3. Eventos: participação em comunicações e painéis

- Participação no XI Encontro de Iniciação Científica da Pontifícia Universidade Católica de Campinas (Outubro/2006)

4. Demais atividades acadêmico-científicas desenvolvidas

Nenhuma.

2. Apresentação das atividades desenvolvidas relacionadas ao Plano de Trabalho:

1. Produção científica

Nenhuma.

2. Relatório

Em anexo.

Programa Integrado de Iniciação Científica

Relatório final do período de Agosto/2006 a Julho/2007

Leandro Augusto Fogolin Pereira, RA: 05356787

Faculdade de Engenharia de Computação
Aluno de Iniciação Científica: FAPIC/Reitoria

Plano de Trabalho: Estudo e Desenvolvimento de uma Biblioteca para Tratamento de Sinais de Sensores de Robôs Móveis

Prof. Dr. Juan Manuel Adán Coello
Faculdade de Engenharia de Computação
Projeto de pesquisa: Aprendizado com Classes Desbalanceadas: Tratamentos e Possíveis Soluções
Grupo de pesquisa: Sistemas Inteligentes
Linha de pesquisa: Inteligência Artificial
Líder do Grupo de Pesquisa: sim

Sumário

1 Introdução

Robôs autônomos despertam a atenção de muitos pesquisadores há décadas. Este projeto faz parte de um maior, com o intuito de senvolver um robô autosuficiente, capaz de mapear um ambiente e locomover-se nele sem interação humana.

2 Objetivos

Construir um robô autônomo, capaz de mapear e locomover-se em um ambiente fechado.

3 Metodologia

3.1 Kit ALFA da PNCA

O *kit* ALFA da empresa brasileira PNCA[1] é orientado para o ensino de robótica em escolas de ensinos fundamental e médio. O conjunto é composto de várias peças, que vão de motores de corrente contínua com caixa de redução, passando por chapas de alumínio previamente perfuradas e os respectivos materiais de afixação, alguns sensores, e um módulo de controle denominado MC2 (seção 3.2).

3.1.1 Montando um robô simples

Para efeito de reconhecimento do *kit* citado, foi montado o robô que consta em sua documentação. Denominado "Robô Zero" (Figura 1), é praticamente o que se consegue de mais simples com as partes oferecidas.

Figura 1: Robô Zero (Cortesia PNCA)

3.1.2 Programando o robô

O robô zero foi programado utilizando a linguagem Legal, padrão do *software* fornecido pela PNCA. A linguagem é extremamente simples, dado o foco do produto.

A primeira aplicação permitia que o robô seguisse uma faixa com cor contrastante (no caso, uma fita isolante preta colada no tampo de uma mesa cinza-claro). Essa aplicação foi feita de duas formas: utilizando o método embutido na linguagem (`SIGA FAIXA`), e utilizando outras construções, como decisões, leitura dos sensores e atuação dos motores.

Pela simplicidade, a linguagem Legal não foi suficiente para implementar a lógica do robô: não há vetores ou matrizes, nem a possibilidade da criação de variáveis; além disso, há limitações na quantidade de funções que podem ser criadas. Por estes motivos, a idéia original era substituir o *firmware* do módulo de controle, mas não foi possível obter nem a documentação, nem o código fonte juntamente com a PNCA. Por causa disso, utilizamos um outro modo de funcionamento do MC2, que captura os sinais dos sensores e permite a atuação dos motores, de forma interativa.

3.1.3 Descobrindo o protocolo de comunicação

As informações do modo de captura dos sensores são enviadas ao computador por meio de uma porta serial, comunicando a 9600 *bauds*, e interpretadas pelo *software* para PC fornecido pela PNCA. O protocolo não possui documentação disponível para usuários do *kit*, portanto foi necessário fazer engenharia reversa e escrever sua documentação baseada em nossas observações. Ver Apêndice 7.1 para detalhes.

Para descobrir o protocolo, foram utilizados basicamente dois *softwares*: o `ser232mon`[9], *sniffer* para interface serial, e um emulador de terminal serial (`putty`[10]).

O *sniffer* permite observar o que é lido e recebido da porta RS-232. Baseado nas ações enviadas pelo *software* da PNCA, e das reações do MC2,

foi possível mapear todos os comandos que acionassem os motores e lessem os sensores, além de outras funcionalidades, como reprodução de sons e leitura da carga das baterias.

O emulador de terminal serial permite enviar e receber informações através de comunicação serial. Foi utilizado tanto para confirmar as anotações antes de transformá-las em documentação do protocolo, quanto para utilizar o MC2 em ambiente Linux, durante a escrita da biblioteca de comunicação (seção 3.1.4).

3.1.4 Escrevendo a biblioteca de comunicação

Com base na documentação escrita através da engenharia reversa, uma biblioteca escrita na linguagem de programação Python[3] foi desenvolvida juntamente com o colega de pesquisa Fernando Paolieri Neto. Esta biblioteca permite, de forma fácil, utilizar todas as funcionalidades obtidas através do processo de *sniffing*.

As funções dessa biblioteca utilizam o pacote `pyserial`[6], o que a torna portável pelo menos entre ambientes *Unix-like* e *Win32*-compatíveis.

3.1.5 Escrevendo a interface gráfica de controle

Para facilitar os testes, foi escrita, também em linguagem Python, uma interface gráfica (Figura 2) para controlar o módulo MC2. A interface foi desenvolvida utilizando o *toolkit* GTK+[8], com o módulo para *Python* `pygtk`.

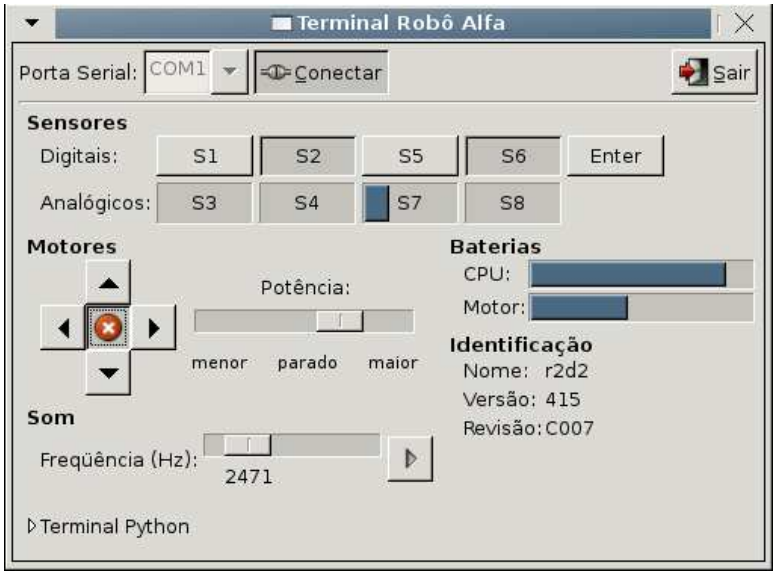

Figura 2: Interface Desenvolvida

A interface gráfica também pode ser executada em mais de um ambiente e foi testada em Linux (ambiente "nativo" ao desenvolvimento) e Windows.

No canto superior esquerdo, pode-se escolher a porta serial na qual o MC2 está conectado; pressionando o botão "Conectar", o *software* conecta-se ao módulo de controle e verifica se há reposta (ver Tabela 2 no Apêndice 7.1). O módulo respondendo a um comando `ping`, a parte inferior da janela fica disponível para uso.

Pode-se observar o valor dos sensores digitais e analógicos, controlar os motores de tração (ordenando o robô para movimentar-se para as quatro direções, com uma velocidade configurável), e os servo-motores (escolhendo para qual ângulo o eixo do motor deve girar). Além disso, existe a medição da carga das baterias da CPU e dos motores, identificação do MC2 (nome, versão, revisão do *firmware*), e há a possibilidade, também, de ordenar o MC2 emitir som de uma dada freqüência.

Além disso, há um terminal no canto inferior da janela, que permite executar, interativamente, quaisquer comandos da linguagem Python, e chamar os métodos da biblioteca.

É uma ferramenta útil para efetuar pequenos testes dos sensores e motores, sem ter que recorrer ao *software* fornecido pela PNCA, que não foi feito para este propósito.

3.2 Limitações da CPU do Kit Alfa

Embora a biblioteca escrita possa efetuar a leitura dos sensores e atuar os motores, há alguns problemas que precisam ser contornados.

O maior problema é durante a frenagem dos motores: o comando demora um pouco para ser consumido pela CPU do MC2, e só é possível frenar um motor por vez; por causa disso, o robô, que utiliza um sistema de direção com uma roda-boba e dois motores, acaba virando ao parar. Outro problema, de igual importância, também ocorre com os motores: ao ativá-los, um é ativado alguns instantes antes do outro, o suficiente para que o robô desvie do caminho em linha reta.

Sem o código fonte do módulo de controle, a solução do problema seria difícil demais. Por causa disso, um *hardware* "clone" do módulo foi desenvolvido, contendo apenas as funcionalidades básicas de leitura de sensores e atuação de motores necessárias ao projeto.

3.2.1 Krone

O "krone", desenvolvido apenas com base na engenharia reversa do protocolo de comunicação serial, é dotado de um microcontrolador RISC de 8-bit[2], e um *driver* de motores (ponte-H)[5].

Um esquema de blocos é mostrado na Figura 3, o fluxograma do *firmware* é mostrado na Figura 4 e o esquemático completo do circuito é mostrado na

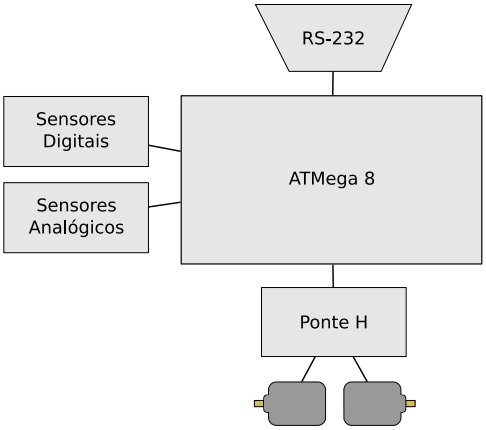

Figura 3: Diagrama de Blocos do Krone

Figura 5.

Este dispositivo foi desenvolvido utilizando o *software* Proteus Isis[7], versão 7.0, que permite efetuar simulação do circuito eletrônico, mesmo que este contenha um microprocessador. O *firmware* foi desenvolvido utilizando compilação cruzada com o porte do GCC[4] para a linha AVR[2] e tenta, sempre que possível, colocar o processador em modo de espera para economizar energia.

Como o clone foi desenvolvido apenas levando em consideração o modo de captura de sinais do MC2, não possui a máquina virtual capaz de executar o código da linguagem Legal, ou até mesmo uma linguagem mais poderosa, necessária para aplicação de mapeamento.

O projeto utiliza Redes Neurais Artificiais para filtragem do sinal dos sensores. Embora de implementação simples, a CPU utilizada não é dotada de unidade de ponto flutuante (o que agilizaria consideravelmente os cálculos – embora seja possível implementar a rede utilizando apenas números inteiros, perde-se em precisão por se tratar de uma CPU com registradores de 8-bit), nem há memória suficiente para armazenamento das matrizes contendo os pesos sinápticos.

Falta de memória também é um problema para a geração dos mapas, já que estes são representados como grandes matrizes.

3.2.2 CPU mais poderosa

Foi feita uma pesquisa procurando uma CPU que contivesse uma quantidade maior de memória e que pudesse executar o código da biblioteca desenvolvida, além de poder interagir com o clone. Foram consideradas várias alternativas, incluindo placas compatíveis com PC para automação industrial – que foram logo descartadas, dado o alto custo relativo às funcionalidades.

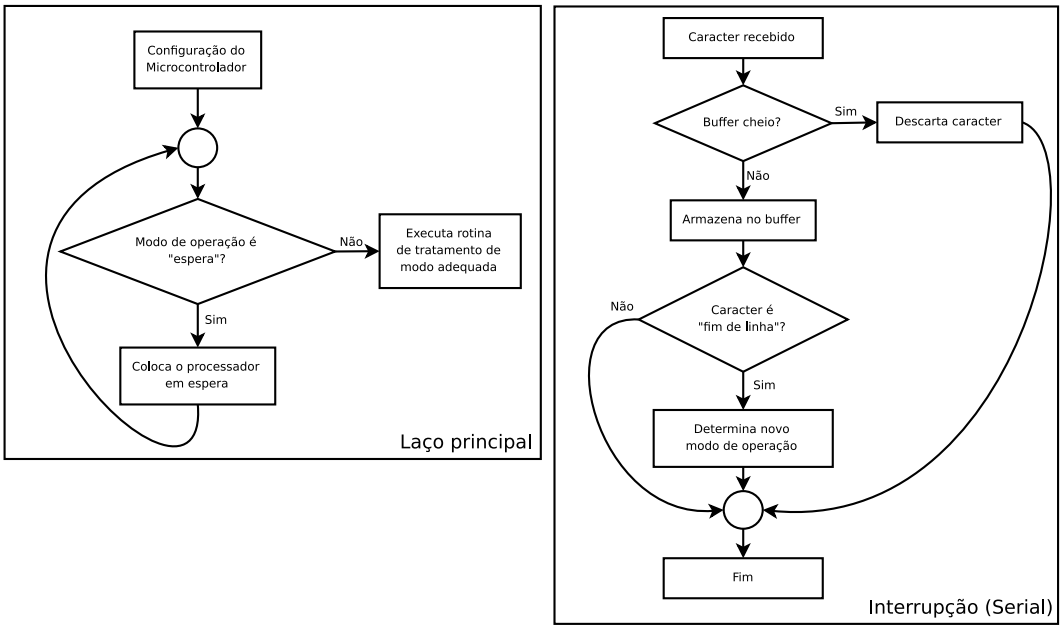

Figura 4: Fluxograma do *firmware* do Clone

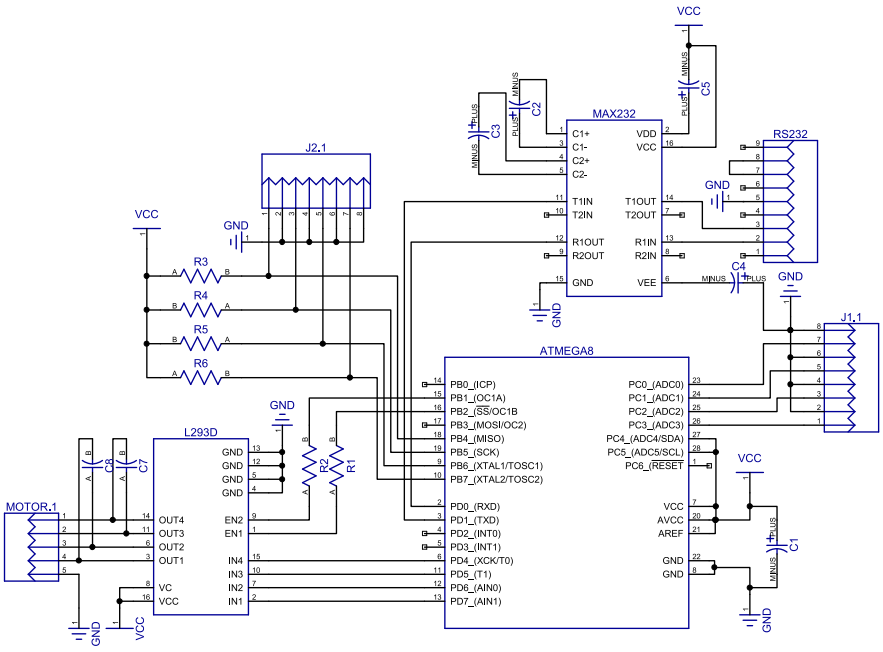

Figura 5: Esquemático do Krone

4 Discussão dos Resultados

O plano de trabalho não foi seguido a risca, já que muitos obstáculos foram subestimados durante o desenrolar do projeto. Sub-projetos como o Krone não estavam previstos, tampouco a utilização de um processador mais poderoso para conter a lógica de mapeamento e navegação do robô.

Os resultados finais, entretanto, foram satisfatórios. Assuntos não estudados em aula até então foram pesquisados e implementados; destaque para o Krone, com a programação e desenvolvimento de uma plataforma embarcada, e para a engenharia reversa do protocolo de comunicação (juntamente com a biblioteca que faz uso das informações obtidas pela respectiva documentação).

5 Perspectivas de Continuação

O projeto como um todo agora tem uma boa base para ser efetivamente iniciado. Mas há, ainda, problemas a serem resolvidos:

- Falta adicionar controle de servomotores (por exemplo, para movimentação dos sensores) e emissão sonora no Krone (para depuração ou outros avisos)
- Um mecanismo que meça o quanto o robô já se movimentou deve ser desenvolvido
- A alimentação dos motores e dos circuitos de controle deve ser reprojetada, levando em consideração o alto consumo do roteador
- Seria interessante que o robô procurasse uma fonte de energia para recarregar suas baterias

6 Considerações Finais

Embora o trabalho tenha sido proveitoso, em termos de experiência e aprimoração dos conhecimentos, trabalhar com pouco ou nenhum incentivo da instituição para obtenção de equipamentos e material de pesquisa fez com que grande parte do tempo investido fosse gasto com o desenvolvimento de tarefas não ligadas diretamente com a proposta de trabalho inicial.

7 Apêndice

7.1 Comunicação serial com o módulo MC2

7.1.1 Identificação

```
rr2d2\r\n    # Nome do robô (r2d2)
```

Nome	Comando	Resposta	Observações
Ping/Pong	`ping`	`pong`	
Identificação	`Mn`	Veja "Identificação"	
Leitura de Sensores	`Ms`	Veja seção 7.1.1	O robô entra em modo de leitura de sensores e só pára quando um Mf é enviado
Parar o Modo de Leitura	`Mf`	Não tem	
Motor Esquerdo	`Me num`	Não tem	`num` escolhe a potência do motor; 0-10: rotação reversa; 11: parado; 12-20: rotação normal
Motor Direito	`Md num`	Não tem	`num` escolhe a potência do motor; 0-10: rotação reversa; 11: parado; 12-20: rotação normal
Servo O	`Mo num`	Não tem	Veja tabela 3
Servo P	`Mp num`	Não tem	Veja tabela 3
Servo Q	`Mq num`	Não tem	Veja tabela 3
Servo R	`Mr num`	Não tem	Veja tabela 3
Som	`MM num`	Não tem	`num` é a freqüência em Hertz
Pára Som	`Mm`	Não tem	

Tabela 1: Comandos do MC2 obtidos por engenharia reversa

```
t415\r\n     # Versão (415)
sC007\r\n    # Revisão (C007)
```

7.1.2 Leitura de Sensores

- Os sensores digitais ficam em 1 quando não há sinal, e em 0 quando há
- Os analógicos ficam em 0 quando não há nada conectado
- As respostas são no formato `LetraNúmero`

7.2 Controle de Versão

Todo o código do projeto foi desenvolvido utilizando o sistema de controle de versões *Subversion*. Durante o desenvolvimento, existiu a necessidade de reverter alguma modificação, rever o código escrito pelo outro colega, ou até

Letra	Significado
a	Sensor digital (porta S1)
b	Sensor digital (porta S2)
c	Sensor analógico (porta S3; 0-1024)
d	Sensor analógico (porta S4; 0-1024)
e	Sensor digital (porta S5)
f	Sensor digital (porta S6)
g	Sensor analógico (porta S7; 0-1024)
h	Sensor analógico (porta S8; 0-1024)
i	Bateria da CPU (0-1024)
j	Bateria dos Motores (0-1024)
k	1 se estiver tudo bem; 0 se os motores estiverem consumindo muita corrente
l	0 se o botão enter estiver pressionado

Tabela 2: Comandos do MC2 obtidos por engenharia reversa

mesmo trabalhar fora do ambiente universitário, sem que isso atrapalhasse o sincronismo do código durante a fase de desenvolvimento.

7.3 Wiki

A documentação escrita para o projeto durante a fase de pesquisa e desenvolvimento foi gerenciada com um *software* Wiki (ver Figuras 6 e 7), hospedado no servidor Web de um dos colegas de pesquisa. O Wiki foi de suma importância para centralizar a documentação, tanto na hora de escrita, quanto para deixar recados e na hora de consultar para o desenvolvimento do clone.

Referências

[1] PNCA robótica e eletrônica. http://www.pnca.com.br.

[2] Atmel Corporation. Atmega8 - 8-bit risc microcontroller. http://www.atmel.com/dyn/products/product_card.asp?part_id=2004.

[3] Python Software Foundation. Python programming language. http://www.python.org.

[4] Free Software Foundation Incorporated. Gcc, the gnu c compiler collection. http://gcc.gnu.org.

[5] Texas Instruments Incorporated. Quadruple half-h drivers. http://focus.ti.com/docs/prod/folders/print/sn754410.html.

[6] Chris Liechi. pyserial. http://pyserial.sf.net.

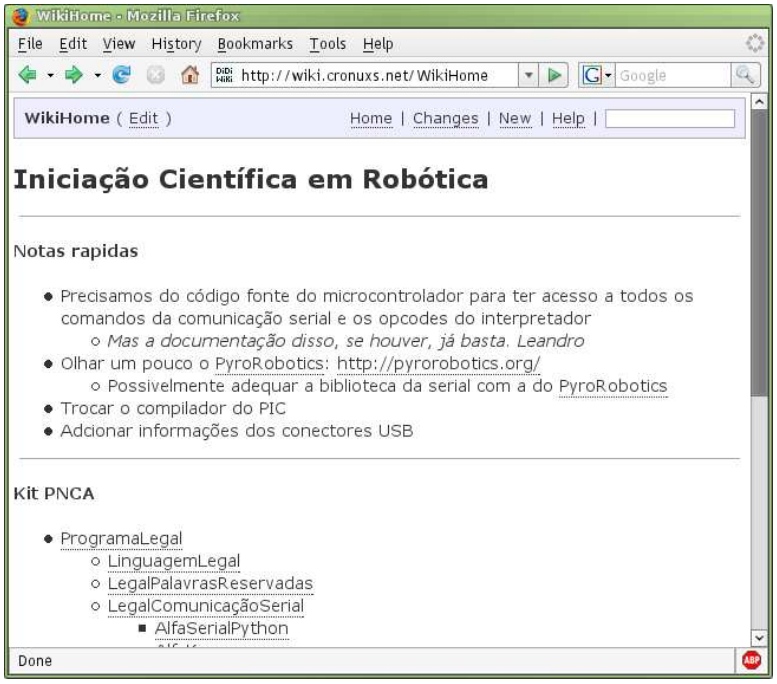

Figura 6: Captura de tela do Wiki (página principal)

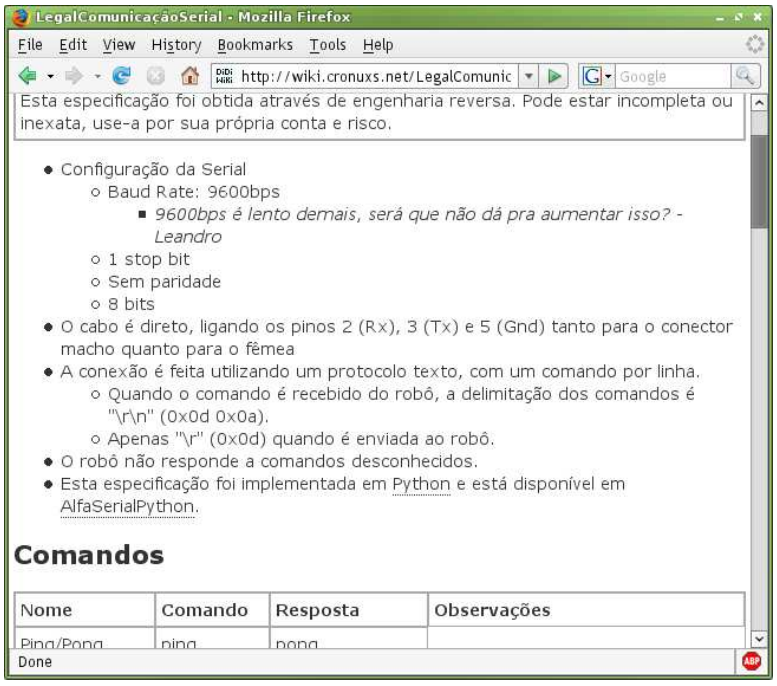

Figura 7: Captura de tela do Wiki (documentação da comunicação serial)

num	Ângulo
15	0
21	15
27	30
33	45
39	60
45	75
49	90
57	105
63	120
69	135
75	150
78	165
81	180

Tabela 3: Relação num-ângulo do servo motor

[7] Labcenter Electronics Ltd. Proteus isis schematic capture. http://www.labcenter.co.uk/products/schematic.cfm.

[8] Peter Mattis, Spencer Kimball, and Josh MacDonald. GTK+: The GIMP toolkit. http://www.gtk.org.

[9] Yippee Soft. ser232mon 1.01. http://www.shengfang.org/blog/p/Ser232Mon.php.

[10] Simon Tatham. PuTTY: a free telnet/ssh client. http://www.chiark.greenend.org.uk/~sgtatham/putty/.