TRATAMENTO DE SINAIS PARA ROBÔS – IC PUC-CAMPINAS

Leandro Augusto Fogolin Pereira
Engenharia de Computação
CEATEC
email leandro@tia.mat.br

Juan Manuel Adán Coello
Engenharia de Computação
CEATEC
email juan@puc-campinas.edu.br

Resumo: *Criação de uma biblioteca para capturar sinais de sensores e acionar motores de um robô comercialmente disponível. Envolveu engenharia reversa para a descoberta do protocolo, criação de um "hardware" compatível com o mesmo e a criação de uma interface gráfica para analisar os resultados e controlar o robô utilizando um microcomputador.*

Palavras-chave: *robótica, sensores, microcontrolador.*

Área do Conhecimento: *Eletrônica Industrial, Sistemas e Controles Eletrônicos*

– Automação Eletrônica de Processos Elétricos e Industriais

1. INTRODUÇÃO

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

2. O "KIT" DE ROBÓTICA

O *hardware* do robô é baseado no *kit* ALFA da empresa nacional PNCA[1], orientado para ensino de programação de computadores e robótica em escolas de ensino fundamental e médio. O conjunto é composto de peças para construir a estrutura mecânica, dos sensores, motores e de um módulo de controle.

3. CONTORNANDO AS LIMITAÇÕES

Durante os testes iniciais, foi possível perceber que a linguagem Legal, padrão do *kit*, não era suficiente para o propósito ambicioso do projeto. Mesmo assim, o módulo de controle possui um modo de captura de sinais dos sensores, cujo protocolo foi devidamente descoberto e documentado através de engenharia reversa, e uma biblioteca escrita em parceria com o colega de iniciação Fernando Paolieri Neto na linguagem *Python*[2], foi criada para conversar com o módulo.

A Figura 1 mostra a captura de tela da interface gráfica desenvolvida com base na biblioteca. Esta interface permite, de modo simples, exibir o valor dos sensores, a carga das baterias e controlar os motores. Além disso, há um *prompt Python* interativo embutido, permitindo a realização de testes rápidos.

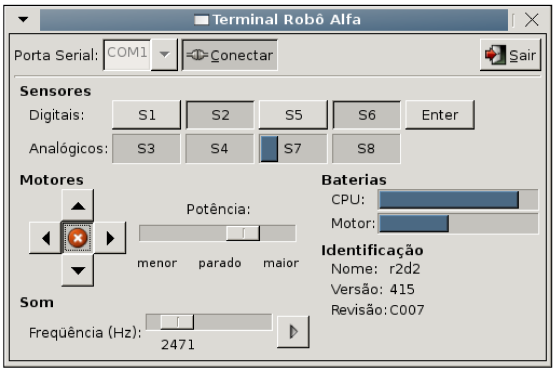

Figura 1: Captura de Tela da Interface Gráfica

Entretanto, mais limitações foram observadas depois que a biblioteca foi criada. Como o módulo de controle leva um certo tempo para consumir e realizar um comando quando no modo de leitura de sensores, frear ou acionar os motores de tração em instantes diferentes pode fazer o robô desviar da linha reta.

Contornar este problema em *software* é praticamente impossível, sem a posse do código fonte do *firmware* do módulo de controle. Por ser propriedade intelectual da PNCA, este não nos foi fornecido. Por causa disso, um *hardware* clone, baseado apenas na documentação obtida anteriormente pela engenharia reversa, foi desenvolvido.

4. O "HARDWARE" CLONE

Utilizando um microcontrolador e componentes de apoio, um módulo de controle simplificado foi desenvolvido. O esquema de blocos pode ser visto na Figura 2.

Todo o *firmware* foi escrito em linguagem C, utilizando compilação cruzada, levando em consideração o consumo de energia (desligando o microcontrolador sempre que preciso).

4. A UNIDADE DE CONTROLE “REAL”

Como há a necessidade de armazenar grandes volumes de informação (em comparação com a ínfima quantidade de memória disponível no microcontrolador), e também faz-se necessário o uso de algumas técnicas de tratamento de sinais dos sensores que fazem cálculos com números de ponto flutuante, o microcontrolador escolhido não atenderia os requisitos.

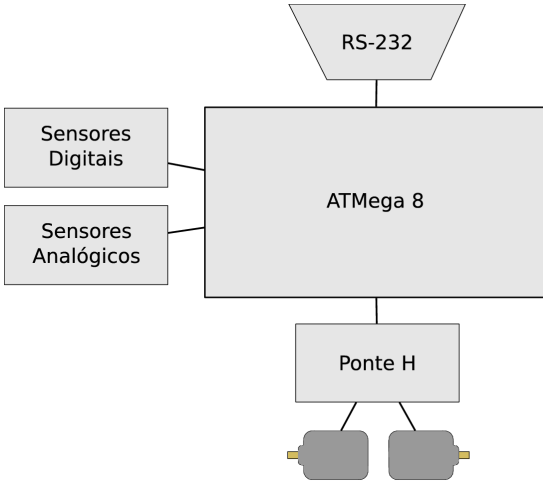

Figura 2: Diagrama de Blocos do Módulo de Controle "Clone"

Além disso, com possibilidades de, remotamente, controlar o robô e obter *streams* de imagens, o clone desenvolvido teve seu papel rebaixado de unidade de controle para apenas uma interface com os sensores e motores.

O *hardware* escolhido para a tarefa de unidade de controle foi um roteador de conexão sem fio. Executando *Linux*[3] de fábrica e com uma porta USB, o roteador foi escolhido também pelo seu atrativo preço, muito abaixo de soluções similares para automação industrial, compatíveis com a arquitetura do IBM-PC. Utilizar o sistema operacional *Linux* foi uma grande vantagem, já que a biblioteca de leitura desenvolvida pôde ser utilizada sem alteração alguma de código.

REFERÊNCIAS

[1] Kit ALFA, *PNCA*, capturado online em 30/08/2007 de <*http://www.pnca.com.br*>.

[2] Lutz, M. (2006), *Programming Python*, 3rd ed., O'Reilly Media, Sebastopool, CA.

[3] Barrett, D. J. (2005), *Linux in a Nutshell*, 5th ed., O'Reilly Media, Sebastopool, CA.