Análise da Plataforma SunSPOT para Programação de Sistemas de Controle Distribuído em Rede de Sensores sem Fio*

André Cavalcante[1], Rodrigo Allgayer[1], Ivan Müller[1], Jovani Balbinot[1], Carlos E. Pereira[1]

[1]Departamento de Engenharia Elétrica – Universidade Federal do Rio Grande do Sul (UFRGS)
Av. Osvaldo Aranha, 103 – Porto Alegre/RS – Brasil

andrecavalcante@ufam.edu.br, allgayer@ece.ufrgs.br, ivan.muller@ufrgs.br,
jovani.balbinot@gmail.com, cpereira@ece.ufrgs.br

Abstract. *This work presents a temporal analysis for a diferential control system of a mobile robot in SunSPOT platform. In such system, which is programmed in Java, each part of it is executed in distinct processor device. These devices form a wireless sensor network. The results of simulations on PC are compared with experimental data and reveal inserted errors by the communication system and by the processing load of the platform.*

Resumo. *Este trabalho apresenta uma análise temporal para um sistema de controle diferencial de um robô móvel em uma plataforma SunSPOT. Neste sistema, programado em Java, cada parte é executada em dispositivos de processamento distintos. Estes dispositivos formam uma rede de sensores sem fio. Os resultados de simulações em PC são comparados com os dados experimentais e revelam os erros introduzidos pelo sistema de comunicação e pela carga de processamento da plataforma.*

1. Introdução

Sistemas de controle distribuídos são desejáveis por diversas vantagens tais como o aumento na robustez do sistema, redundância e a migração de tarefas entre os sistemas computacionais da rede [Zampieri 2008]. Também, a carga de processamento por máquina é balanceada, não sobrecarregando somente uma máquina da rede, ao contrário de sistemas centralizados onde o servidor pode estar sobrecarregado.

Atualmente, diversas tecnologias e protocolos de comunicação possuem capacidade de proporcionar a interligação de sistemas de forma distribuída, incluindo comunicações cabeadas e comunicações sem fio. Com o avanço da comunicação sem fio, há agora a possibilidade da utilização deste tipo de tecnologia em ambientes industriais, reduzindo-se custos devido a não necessidade de uma estrutura física (cabos e barramentos) para transmissão de dados entre os dispositivos do sistema [Kawka and Alleyne 2005, Liu and Goldsmith 2004].

Por outro lado, a utilização de sistemas distribuídos demanda uma arquitetura descentralizada muito dependente do comportamento do sistema de comunicação.

Este artigo analisa o desempenho de um sistema de controle distribuído utilizando-se Rede de Sensores Sem Fio (RSSF). Como plataforma de desenvolvimento foi utilizado

*Este trabalho é parcialmente financiado pelas instituições CAPES, CNPq e FAPEAM.

nodos SunSPOTs [Microsystems 2010b]. Uma API (Application Programming Interface) que possibilita a chamada de métodos remotos também foi desenvolvida a fim de facilitar a programação de sistemas distribuídos.

O texto está dividido da seguinte forma, na Seção 2, apresenta-se as características e as dificuldades que surgem na programação de sistemas de controle distribuídos utilizando Java e SunSPOT. A seguir, na Seção 3, é descrita a API desenvolvida para chamada de métodos remotos. Na Seção 4, é apresentada a aplicação desenvolvida para posterior análise, e na Seção 5, os resultados obtidos. Por fim, na Seção 6, são colocadas as conclusões finais deste trabalho e apresentados os trabalhos futuros.

2. Programação de aplicações distribuídas

Em um ambiente industrial, é importante a realização de distribuição completa em uma rede de sensores, isto é, que todos os nodos da rede possam acessar as respectivas aplicações distribuídas em todos os demais nodos, independente de qual tipo é tal nodo. Isso significa que as aplicações que são executadas em uma rede industrial, abstraem a própria rede, de acordo com a sua aplicação. Não é interessante para o programador ter que criar código para tratar diferentes tipos de nodos (sensores, atuadores, repetidores, controladores, entre outros).

Além disso é esperado que, em uma rede industrial, as aplicações e a rede tenham certas características, como a auto-organização, onde os nodos têm a capacidade de organizar-se autonomamente na rede. Para realizar a auto-organização, o desenvolvedor deve prover a rede com a capacidade de distribuição automática de endereços e portas de comunicação, bem como a descoberta de serviços. A rede deve também ser capaz de se comunicar com dispositivos não anteriormente planejados, desde que respeitadas algumas interfaces padronizadas.

Uma arquitetura promissora que permite distribuição e possui um registro dos objetos (e serviços) disponíveis na rede é o RMI (Remote Method Invocation) [Waldo 1998]. O RMI é uma interface de programação que permite a execução de chamadas remotas no estilo RPC (*Remote Procedure Call*) em aplicações desenvolvidas em Java. Através da utilização da arquitetura RMI, é possível que um objeto ativo em uma máquina virtual Java possa interagir com objetos de outras máquinas virtuais Java, independentemente da localização dessas máquinas virtuais na rede. Dessa forma, algumas das características de RMI são desejáveis para sistemas industriais.

3. Chamadas de métodos remotos na plataforma SunSPOT

A plataforma de escolha para este trabalho foi a SunSPOT. Esta foi desenvolvida para a programação de aplicações para rede de sensores sem fio utilizando funcionalidades da linguagem Java. Possui sensores embarcados e um rádio de comunicação sem fio, representados por objetos Java cujas funcionalidades são acessadas através de métodos.

Enquanto é simples e rápido o acesso aos periféricos do SunSPOT, o mesmo apresenta algumas limitações importantes no que tange o desenvolvimento de uma rede de sensores e atuadores completamente distribuídos. Por ser baseado na CLDC 1.0 [Microsystems 2010a], o SunSPOT não possui as características de Java necessárias à chamadas de métodos remotos e também não possui nenhum tipo de serialização de objetos. Porém, tais características podem ser inseridas no sistema através da programação

de uma API. Com o objetivo de prover ao programador tanto a serialização quanto a chamada remota de métodos foi desenvolvida uma API denominada mRMI (minimum Remote Method Invocation), na tentativa da utilização de um SunSPOT em ambientes industriais.

Assim definiu-se uma interface denominada de `Externalizable`. Quando objetos de uma classe necessitarem ser serializados (externalizados, no jargão do RMI), basta que a classe implemente a interface. Assim, os objetos das classes especiais `ObjectOutputStream` e `ObjectInputStream` podem operar sobre um objeto que implemente `Externalizable`, realizando o transporte dos objetos pela rede. As classes `ObjectOutputStream` e `ObjectInputStream` também necessitaram ser criadas. Contudo cabe à classe que implementa `Externalizable` realizar a escrita e leitura adequada de seus parâmetros a partir dos *streams* fornecidos. Essa abordagem foi chamada de "manual", porque deixa ao programador toda a carga do protocolo de serialização.

Foi igualmente definida uma interface `RemoteSever` para permitir a construção de servidores que possibilitem a recepção de chamadas remotas. No lado do cliente foi criada uma interface `RemoteStub` que define o comportamento padrão para todos os "stubs" da API. Foram criadas classes com um comportamento e protocolo padrão a fim de minimizar os esforços de programação, contudo o programador deve ao menos conhecer o protocolo de rede para a utilização correta do sistema.

Como não há nenhuma implementação padrão no SunSPOT para o RMI, então não há igualmente um registro de objetos remotos (o *registry*). Isto também teve de ser criado e o gerenciamento de endereços e portas também teve de ser realizado manualmente.

4. Estudo de Caso: Simulação de um robô móvel distribuído

Como estudo de caso de um sistema de controle distribuído utilizando uma malha de controle fechada, foi escolhido um robô móvel que se movimenta no plano [Lages 2008]. O robô apresenta um acionamento diferencial que pode ser descrito pelo modelo no espaço de estados dado pela Equação 1.

$$\dot{x}(t) = \begin{bmatrix} \cos(x_3) & 0 \\ \sin(x_3) & 0 \\ 0 & 1 \end{bmatrix} u(t) \qquad y(t) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} x(t) + \begin{bmatrix} R\cos(x_3) \\ R\sin(x_3) \end{bmatrix} \tag{1}$$

onde $x(t) = [x_c\ y_c\ \theta]^T$, sendo $(x_c,\ y_c)$ a posição do centro de massa do robô e θ a sua orientação. $u(t) = [v\ \omega]^T$ é a entrada do sistema, sendo v a velocidade linear e ω a velocidade angular do robô. A saída do sistema é $y(t)$, correspondendo a frente do robô cujo diâmetro é $D = 2R = 0.6m$.

Para este sistema, tem-se:

$$\dot{y}(t) = \begin{bmatrix} \cos(x_3) & -R\sin(x_3) \\ \sin(x_3) & R\cos(x_3) \end{bmatrix} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = L(x)u(t) \tag{2}$$

e portanto, como $L(x)$ é não singular, o sistema pode ser linearizado por realimentação fazendo-se $u = L^{-1}(x)v$, e sendo v a nova entrada do sistema linearizado e desacoplado

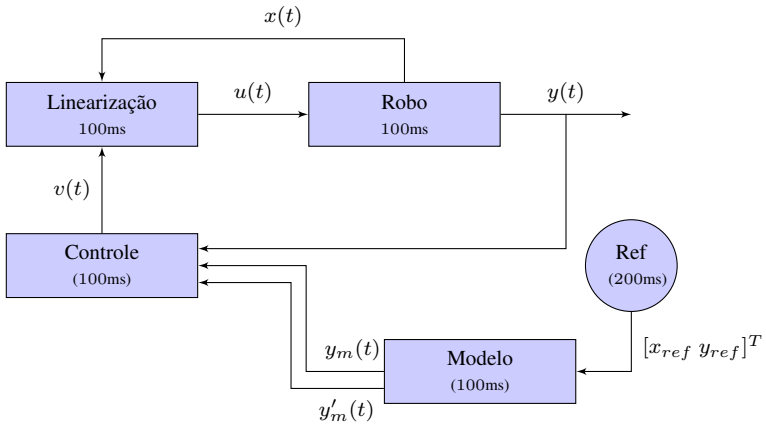


Figura 1. Diagrama em blocos para a simulação do robô.

dado por $\dot{y}(t) = v(t)$. Então, para controlar o sistema é utilizado um controlador por modelo de referência com $v(t)$ dado por:

$$v(t) = \begin{bmatrix} \dot{y}_{mx} + \alpha_1(y_m x - y1) \\ \dot{y}_{my} + \alpha_2(y_m y - y2) \end{bmatrix} \tag{3}$$

A referência é dada por:

$$\begin{cases} x_{ref}(t) = \frac{5}{\pi}\cos(0.1\pi t) \\ y_{ref}(t) = \begin{cases} \frac{5}{\pi}\sin(0.1\pi t) & \text{, para } 0 <= t < 20 \\ -\frac{5}{\pi}\sin(0.1\pi t) & \text{, para } t >= 20 \end{cases} \end{cases} \tag{4}$$

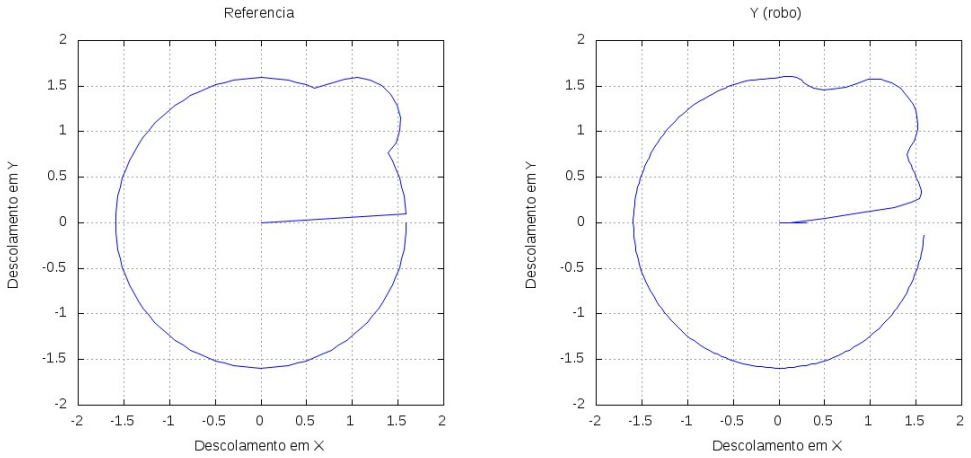
O robô descrito pela Equação 1 foi simulado no SunSPOT e visa refletir o comportamento da frente do robô (a saída do sistema) dada a entrada da próxima posição do centro de massa. A partir das equações mostradas, pode-se montar um diagrama em blocos do sistema do robô móvel controlado, sendo este representado na Figura 1.

O sistema do robô foi separado em dois nodos distintos: o nodo Robô e o nodo Referência. O nodo Robô executa as equações do robô dado pela Equação 1. Por outro lado, o nodo Referência executa os demais blocos funcionais apresentados na Figura 1.

5. Resultados

Com base no estudo de caso desenvolvido na Seção 4 e a fim de que se tenha uma visão clara da plataforma SunSPOT como nodo em uma aplicação de sistema de controle distribuído, algumas simulações foram realizadas com períodos de amostragem de 50ms, 80ms, 100ms e 200ms. Também foram realizadas simulações em que todas as *threads* do sistema executam somente em um PC, ou somente no SunSPOT, ou utilizando uma abordagem híbrida. Como análise dos resultados, neste artigo são abordados dois casos.

No primeiro caso os nodos Robô e Referência executam em uma máquina PC. Assim, a comunicação entre os nodos não apresenta interferência da rede física, e ainda ocorre em uma plataforma capaz de suportar muito processamento. Na Figura 2(a) está representada a referência gerada para o robô e, na Figura 2(b) a trajetória que o robô

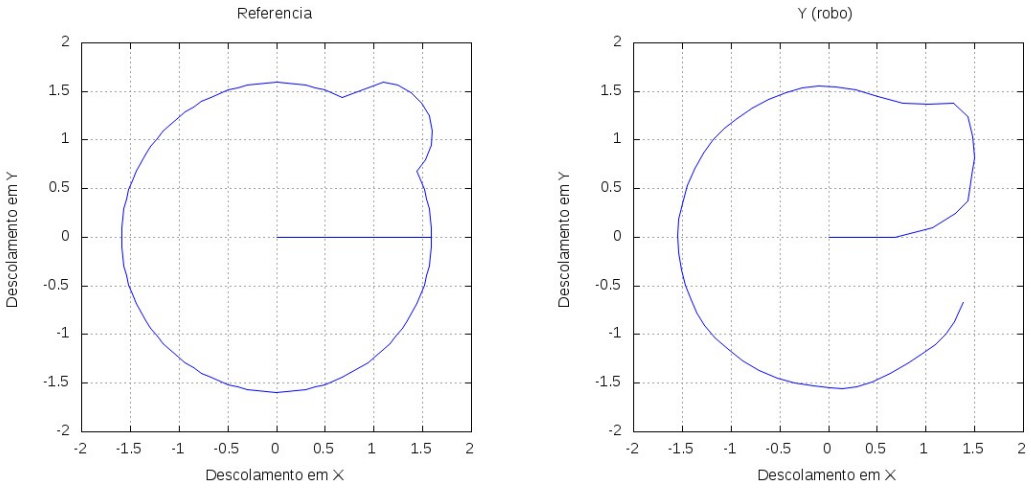


(a) Sinal de referência (b) Sinal de saída

Figura 2. Simulação em uma plataforma PC

realiza ao ser simulado o seu modelo. As figuras representam um plano (x, y) sobre o qual o robô se move.

Em outra situação, o nodo Robô é executado em um dispositivo SunSPOT, e o nodo Referência é executado em uma máquina PC comunicando-se com o nodo Robô através de rádio-frequência. O objetivo desta simulação é verificar o desempenho do sistema e o *jitter* dos períodos de execução dos blocos funcionais em um sistema de controle distribuído utilizando comunicação sem fio. Na Figura 3(a) está representada a referência gerada para o robô executado em um PC e, na Figura 3(b), a trajetória que o robô realiza ao ser simulado o seu modelo na plataforma SunSPOT.



(a) Sinal de referência (b) Sinal de saída

Figura 3. Simulação em uma plataforma SunSPOT

A análise estatística para o período de amostragem, dos dois casos, está apresentado na Tabela 5.

Tabela 1. Medidas estatísticas do jitter na simulação do robô nas plataformas.

	Simulação PC	Simulação SunSPOT
Período	0.100000s	0.100000s
Amostras	202	177
Mínimo	0.097000s	0.057000s
1o Quartil	0.099000s	0.092000s
Mediana	0.100000s	0.110000s
Máximo	0.105000s	0.180000s
Media	0.099801s	0.113744s
Desvio padrão	0.000860s	0.027153s

6. Conclusões

Comparando os gráficos da referência e aqui colocados, como era de se esperar as simulações utilizando a rede sem-fio apresentam erros maiores. As simulações com período de amostragem inferior a 100ms apresentam erros ainda maiores. As simulações no PC contudo não apresentam os mesmos erros, o que mostra que a plataforma escolhida (SunSPOT) carece de performance de processamento para este caso.

Igualmente o esforço necessário para a programação da API mRMI torna apenas mais simples a construção da aplicação em si, contudo o cuidado com o protocolo de comunicação ainda é relevante no esforço de programação. Portanto, verificou-se que o SunSPOT pode ser utilizado para suportar uma API que implemente chamada de métodos remotos, porém isto somente é possível para aplicações pequenas, onde existe um número limitado de nodos e de malhas de controle. Na mRMI, a complexidade do código aumenta em função do número de nós e da complexidade da aplicação, sendo necessário uma plataforma de hardware com maior capacidade de processamento para executar aplicações mais complexas.

Como trabalho futuro está previsto o desenvolvimento de estimadores do atraso do sistema para que o seu impacto possa ser minimizado.

Referências

Kawka, P. and Alleyne, A. (2005). Stability and feedback control of wireless networked systems. In *Proceedings of American Control Conference*.

Lages, W. F. (2008). Projeto. Technical report, Universidade Federal do Rio Grande do Sul.

Liu, X. and Goldsmith, A. (2004). Wireless network design for distributed control. In *Proceedings of 43rd IEEE Conference on Decision and Control*.

Microsystems, Sun (2010a). *Connected Limited Device Configuration - CLDC*. Sun Microsystems. Disponível em: <http://http://java.sun.com/products/cldc/>. Acesso em: março 2010.

Microsystems, Sun (2010b). *Sun Small Programmable Object Technology - Theory of Operation*. Disponível em: <http://www.sunspotworld.com/docs/index.html>. Acesso em: março 2010.

Waldo, J. (1998). Remote procedure calls and java remote method invocation. In *IEEE Concurrency*, volume 6, pages 5 – 7.

Zampieri, S. (2008). Trends in networked control systems. In *Proceedings of 17th IFAC World Congress*.