UM MIDDLEWARE COMO INTERFACE PADRÃO PARA ACESSO A DISPOSITIVOS BIOMÉDICOS

João Marcos Teixeira Lacerda
Mestrando em Engenharia Elétrica e de Computação pela Universidade Federal do Rio Grande do Norte (UFRN). E-Mail: joao.marco@deb.ufrn.br

Ricardo Alexsandro de Medeiros Valentim
Professor do Departamento de Engenharia Biomédica da Universidade Federal do Rio Grande do Norte (UFRN). E-mail: ricardo.valentim@ufrnet.br

Bruno Gomes de Araújo
Doutorando em Engenharia Elétrica e de Computação pela Universidade Federal do Rio Grande do Norte (UFRN). E-Mail: bruno.gomes@ifrn.edu.br

Gláucio Bezerra Brandão
Professor do Departamento de Engenharia Biomédica da Universidade Federal do Rio Grande do Norte (UFRN). E-mail: glaucio@dca.ufrn.br

Ana Maria Guimarães Guerreiro
Professora do Departamento de Engenharia Biomédica da Universidade Federal do Rio Grande do Norte (UFRN). E-Mail: anamaria@dca.ufrn.br.

Francis Solange Vieira Tourinho
Professora do Departamento de Enfermagem da Universidade Federal do Rio Grande do Norte (UFRN). E-Mail: francistourinho@ufrnet.br

José Diniz Júnior
Professor do Departamento de Medicina Clinica da Universidade Federal do Rio Grande do Norte (UFRN). E-Mail: diniz@ufrnet.br

RESUMO
A grande diversidade na arquitetura de dispositivos de hardware, aliada aos seus diferentes protocolos de comunicação, tem dificultado a implementação de sistemas que necessitam realizar o acesso a esses dispositivos. Diante dessas diferenças, surge a necessidade de prover o acesso a esses dispositivos de forma transparente. Neste sentido, o presente trabalho propõe um middleware mult entrada e saída para acesso a dispositivos, como forma de abstrair o mecanismo de escrita e leitura de dados em dispositivos de hardware, contribuindo desta forma, para o aumento na produtividade dos sistemas, uma vez que os desenvolvedores estão focados apenas nos seus requisitos funcionais.

PALAVRAS-CHAVE: Middleware, Dispositivos biomédicos, Entrada e saída, Arquitetura Orientada a Serviço.

A MIDDLEWARE AS DEFAULT INTERFACE FOR ACCESS TO BIOMEDICAL DEVICES

ABSTRACT

The great diversity in the architecture of hardware devices, allied to the different communication protocols, has been hindering the implementation of systems that need to accomplish the access of these devices. Before these differences, it appears the need of providing the access of these devices in a transparent way. In this sense, the present work proposes a middleware, mult input and output for access the devices, as form of abstracting the writing and reading data mechanisms in hardware devices, contributing this way, for the increase in the productivity of the systems, once the developers are just focused in their functional requirements.

KEYWORDS: Middleware, Biomedical Devices, input and output, Software Oriented Architecture.

UM MIDDLEWARE COMO INTERFACE PADRÃO PARA ACESSO A DISPOSITIVOS BIOMÉDICOS

INTRODUÇÃO

A rápida disseminação de novos dispositivos de hardware no mercado tem aumentado a complexidade na comunicação entre esses. Segundo Valentim (2008), muitos dispositivos de hardware disponibilizam suas funcionalidades através de protocolos proprietários, de drivers que são dependentes de uma plataforma de sistema operacional. Diante desse ambiente fechado e heterogêneo de hardware, uma questão precisa ser discutida, a interoperabilidade. Isso porque, quanto mais divergentes forem, mais difícil e complexo serão os mecanismos de integração. Esse aspecto torna-se ainda mais forte quando se trata de arquiteturas proprietárias. Essa problemática está ainda mais presente na área médica, na qual muitos dispositivos biomédicos são proprietários e, portanto, dificultam o acesso e conseqüentemente o desenvolvimento de aplicações, como por exemplo, a monitoragem de pacientes, a aquisição de sinais biomédicos, entre outras.

Neste contexto, têm surgido propostas de arquitetura orientada a serviço (Service Oriented Architecture – SOA) para dispositivos de Hardware (SODA – Software Oriented Device Architecture), como mecanismo para construção de interfaces abertas na forma de acesso (I/O – Input and output) aos dispositivos de qualquer natureza, Scott, et. al, (2006).
Este trabalho tem como objeto de estudo, a proposta de um modelo de arquitetura orientada a serviço para acessar dispositivos de hardware, aqui intitulado de Mult-I/O. O Mut-I/O, implementa uma visão baseada em uma interface aberta, a qual permite manipular elementos de entrada e saída de um hardware de forma transparente. Com vista, para validar o Mult-I/O foi realizado um estudo de caso dirigido a dispositivos biomédicos, uma vez que estes equipamentos utilizam arquitetura e protocolos proprietários para comunicação.
Nesta perspectiva, o presente trabalho contribui para que o acesso aos dispositivos de hardware seja independente de drivers, plataforma de desenvolvimento e sistema operacional. Outra contribuição é o encapsulamento provido pelo modelo, que abstrai do cliete o mecanismo de acesso ao dispositivo (I/O), bem como, os protocolos de acesso. Neste sentido, permitindo que o desenvolvimento de aplicações seja focado nas regras de negócio, possibilitando então uma

maior produtividade. Esse fator é substancial em se tratando da área médica, pois permitirá, por exemplo, que a equipe de tecnologia da informação de um hospital, possa desenvolver aplicações de monitoramento de pacientes sem precisar conhecer como funcionam os dispositivos biomédicos relativos a está função, contribuindo dessa forma, para a redução dos custos e riscos e para o aumento da produtividade.

O Mult-I/O atua como um middleware de aplicações clientes para acesso a dispositivos de hardware, seja por uma porta USB, serial, paralela, interface de rede, entre outras. Neste sentido, a principal função do modelo proposto será a de abstrair (tornar transparente) os mecanismos de escrita e leitura das aplicações clientes no acesso aos dispositivos de hardware, neste caso, os dispositivos biomédicos.

A proposta de uma arquitetura orientada a serviço como um middleware, aliada ao paradigma de dispositivos eletrônicos não é nova. Sales (2005) usa SOA no intuito de monitorar processos industriais de forma distribuída e independente de plataforma, abstraindo o mecanismo de acesso dos sistemas de supervisão aos CLPs (CLP – Controlador Lógico Programável). Esta é uma proposta eficiente, pois aumenta a produtividade dos sistemas supervisores no âmbito industrial. No entanto, por ser específica aos CLPs, não tem uma abordagem generalista que permita abstrair o acesso aos dispositivos de tipos diversos, tal qual o Mult-I/O.

Neste contexto, o conceito de SODA, proposto por Deugd et. S. (2006) foi concebido como uma adaptação de SOA para a aplicação em dispositivos. O principio básico da proposta SODA, portanto, é o de reunir padrões existentes tanto no domínio de TI (Tecnologia da Informação) como o usado em dispositivos, cujo objetivo é reduzir a complexidade e os custos gerados na integração destes.

Como descrito acima, é possível modelar um dispositivo como se fosse um serviço, para tanto, é definida uma interface que descreve os seus mecanismos de acesso, que geralmente é executado através de um modelo de requisição e resposta (arquitetura cliente/servidor). Todavia, apesar de ser uma boa proposta, SODA é uma tecnologia muito nova e que ainda não foi validada. Nesta perspectiva, a proposta Mult-I/O contribui também para demonstrar a viabilidade SODA através de testes experimentais.

2 MULT-I/O

Um desenvolvedor, ao implementar uma aplicação que precisa acessar um determinado dispositivo de hardware, se depara com alguns problemas:

- Mecanismos de acesso do dispositivo;
- Implementação dos métodos de acesso ao dispositivo;
- Lógica de negócio da aplicação passa a ficar em segundo plano.

Esses problemas serviram de motivação para o desenvolvimento da proposta Mult-I/O, que efetivamente é uma camada de software distribuída que faz o intermédio entre as requisições e respostas direcionadas aos dispositivos de hardware.

O middleware em questão pode ser contextualizado em um ambiente corporativo, pois geralmente é bastante heterogêneo. A Figura 1 ilustra o Mult-I/O inserido nesse ambiente, sobretudo ilustrando a heterogeneidade, tanto nas aplicações que acessam os dispositivos de hardware, quanto nos próprios dispositivos, ou seja, o Mult-I/O age como elemento de

acesso entre as aplicações e os dispositivos. Deste modo, tornando homogêneo o acesso aos dispositivos.

Desenvolvedores de aplicações que utilizam linguagens de programação como Java, C# ou Python, e qualquer outra que forneça suporte a SOA, poderão acessar dispositivos de hardware sem a necessidade de ter a implementação do acesso a esses dispositivos dentro de suas aplicações.

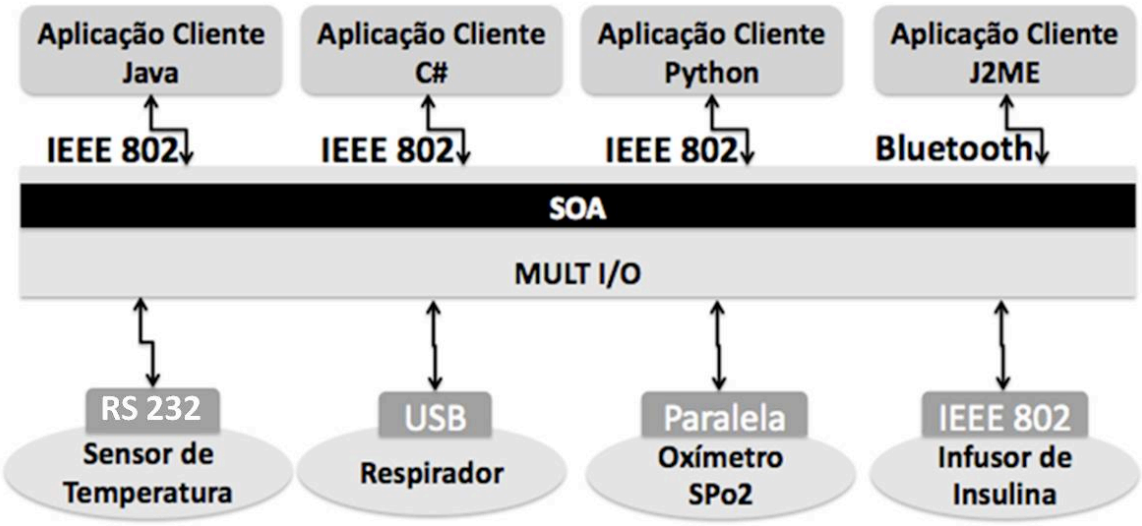

Figura 1. O ambiente do Mult-I/O

A Figura 1 apresenta requisições sendo realizadas por aplicações clientes de forma remota, através de uma arquitetura orientada a serviço, que no Mult-I/O é implementada através de Web Services (Stal, 2002). Nessa arquitetura, o conceito de marshalling e unmarshalling (Hamilton, et. al, 1993), são implementadas através da tecnologia XMLRPC utilizada pelos Web Services. Basicamente, são requisições feitas na linguagem de programação nativa da aplicação, que necessita acessar um dispositivo qualquer. Essas requisições operam sobre o padrão W3C XML (W3C, 2010). O código XML que chega ao Mult-I/O é processado, e posteriormente, com base no processamento, o Mult-I/O realiza o acesso ao dispositivo de destino que aplicação cliente requisitou.

A comunicação realizada sobre o Mult-I/O ocorre através de um repositório de dispositivos, o qual permite a ele identificar exatamente o dispositivo requisitado, provendo deste modo, acesso transparente às aplicações clientes.

2.1 ARQUITETURA

Conforme ilustrado na Figura 2, há dois componentes principais na arquitetura do Mult-I/O, a implementação das interfaces de comunicação (API Mult-I/O) e a implementação de um serviço baseado em SOA (Componente SOA).

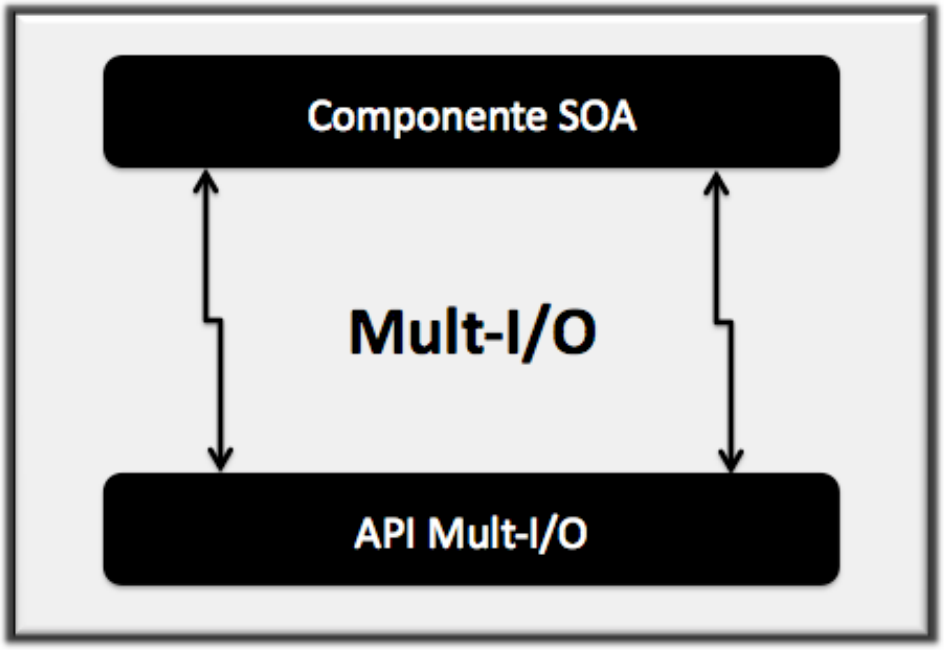

Figura 2. Principais elementos da arquitetura Mult-I/O

A API Mult-I/O (Figura 3) integra as diferentes implementações de acesso a dispositivos, tais como USB, Serial, Paralela, IEEE 802. Essa API provê uma interface remota para a aplicação que servirá de base para as requisições de I/O. As diferentes implementações são definidas de acordo com a interface de comunicação que o dispositivo implementa. Durante o processo que estabelece a comunicação, o middleware Mult-I/O utiliza o padrão de projeto Factory Method (GoF, 1998), para criar uma interface adequada ao dispositivo. Para isso é realizada uma consulta ao repositório de dispositivos que contém as informações pertinentes dos dispositivos integrados ao Mult-I/O.

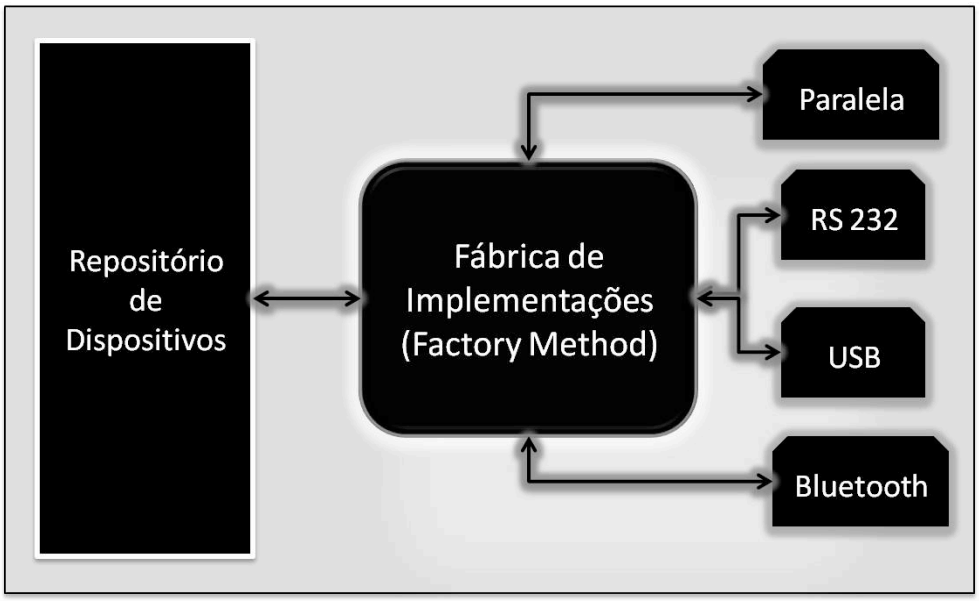

Figura 3. A API do Mult-I/O

O padrão de projeto Factory Method tem uma importância fundamental na API Mult-I/O, no sentido de torná-la dinâmica em relação às implementações das interfaces de comunicação. Para isso, será dada uma interface para a aplicação cliente, no qual conhecerá apenas a assinatura dos métodos, sem detalhes de implementação. A alternância na implementação das interfaces de comunicação dependerá de uma demanda da aplicação cliente em relação ao dispositivo. Ou seja, o cliente não saberá se o dispositivo se comunica via porta USB, paralela ou RS-232, ele apenas o escolherá para trocar informações. O código executado estará atrelado ao Repositório de dispositivos (Figura 3).

O componente SOA terá como base o framework SOAP, que é a implementação do protocolo SOAP (W3C, 2010) direcionada a certa linguagem de programação, ou seja, ele sofrerá variação de acordo com a linguagem implementada. Por exemplo, o Apache Axis (Apache, 2010) para a linguagem Java ou Nusoap (Microsoft, 2010) para a linguagem PHP. O protocolo SOAP (W3C, 2010) é responsável pelo transporte das requisições e respostas (mensagens em XML) pela rede. O Framework SOAP atua como uma ferramenta de conversão entre as requisições feita pelas aplicações clientes em XML, para a linguagem nativa da implementação da interface à qual pertence o dispositivo. Também recebe as respostas dos dispositivos na linguagem nativa e as converte para XML. O padrão W3C WSDL (W3C, 2010) descreve em serviços os recursos disponibilizados pela implementação da interface de comunicação. A Figura 4 apresenta os elementos do Componente SOA e a Figura 5 exemplifica o WSDL no contexto do Mult – I/O.

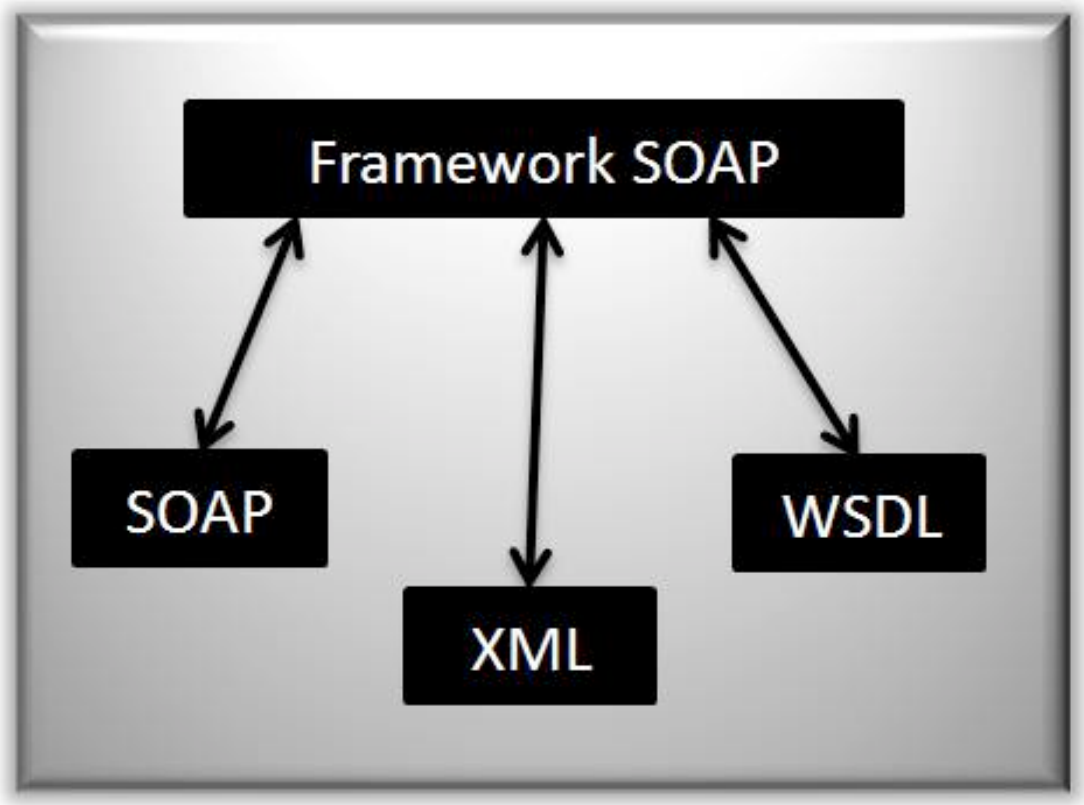

Figura 4. Descrição do Componente SOA

```
<element name="lerDispositivo">
    <complexType>
     <sequence>
      <element name="dispositivo" type="xsd:int"/>
     </sequence>
    </complexType>
   </element>
   <element name="lerDispositivoResponse">
    <complexType/>
   </element>
   <element name="lerDados">
    <complexType/>
   </element>
   <element name="lerDadosResponse">
    <complexType>
     <sequence>
      <element name="lerDadosReturn" type="xsd:string"/>
     </sequence>
    </complexType>
   </element>
<element name="escreverDispositivo">
    <complexType>
     <sequence>
      <element name="msg" type="xsd:string"/>
      <element name="dispositivo" type="xsd:int"/>
     </sequence>
    </complexType>
   </element>
   <element name="escreverDispositivoResponse">
    <complexType/>
   </element>
```

Figura 5. Trecho do WSDL no contexto do Mult-I/O

O trecho do WSDL da Figura 5, é a interface na qual as aplicações clientes dos dispositivos implementarão para acessá-los de forma transparente. Esse exemplo mostra duas operações disponibilizadas: A leitura (“lerDispositivo”) de dados do dispositivo e a escrita (“escreverDispositivo”) no dispositivo.

2.2 QUALIDADES DO MULT-I/O

Por ser um middleware, o Mult-I/O provê qualidades inerentes a este, dentre as mais importantes estão interoperabilidade, integração e encapsulamento das especificidades de comunicação.

A interoperabilidade provida pelo Mult-I/O é justificada pelo fato de qualquer aplicação, independente de sistema operacional, poder acessar um determinado dispositivo em um

repositório de dispositivos (Como visto na Figura 1) de maneira transparente, seja qual for o sistema operacional onde estiver executando o dispositivo, ou mesmo se esse estiver em um ambiente embarcado.

O encapsulamento provido pelo Mult-I/O é explicado por que o desenvolvedor da aplicação não tem a necessidade de conhecer o protocolo de comunicação ao qual o dispositivo que se deseja acessar implementa, ele precisar apenas conhecer interfaces padrões que descrevem o modo como tal dispositivo funciona.

3 TESTES E RESULTADOS EXPERIMENTAIS

A implementação de um sistema distribuído ge-ra alguns desafios, segundo Shimitd (1995). Tais desafios são: Concorrência, tratamento de eventos, acesso e configuração dos serviços e sincronização. Devido a essa complexidade, para o que o Mult-I/O seja validado em um primeiro passo, o ambiente no qual ele executará foi simplificado (Figura 6).

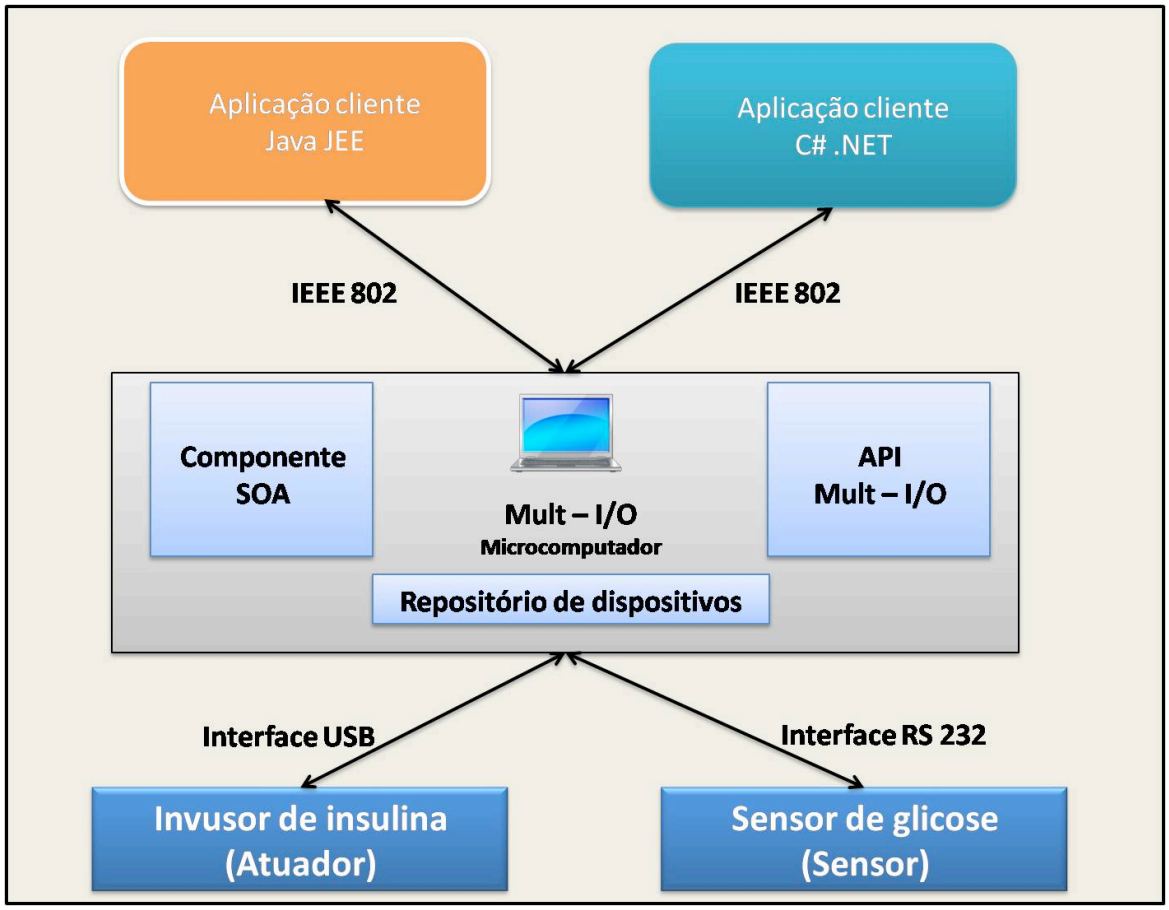

Figura 6. O middleware no ambiente de testes

Nesse ambiente da Figura 6, as requisições aos dispositivos foram realizadas por aplicações com duas implementadas em duas linguagens de programação distintas: Java JEE e C# .NET. Essas aplicações se comportarão da mesma maneira que na Figura 1. As diferenças estarão no ambiente ao qual o Mult-I/ O executa. Ao invés de um ambiente distribuído, foi utilizado um microcomputador com a seguinte configuração: Processador Pentium Dual Core com 2GB de memória RAM e sistema operacional Windows XP. Toda

a implementação do Mult-I/O foi realizada com a linguagem Java. No microcomputador do middleware, estarão conectados os dispositivos que as aplicações desejam acessar (Nas suas diferentes interfaces de I/O), tal como o repositório de dispositivos. Esse repositório é um banco de dados com registros referentes aos dispositivos biomédicos, como nome e interface de comunicação. A diferença entre os ambientes, é que na Figura 1, os dispositivos são acessados pelo Mult-I/ O via protocolo IEEE 802 e, na Figura 6, os dispositivos são conectados localmente.

A aplicação cliente que acessou o Mult-I/O, foi implementada na linguagem de programação Java. Essa aplicação estava localizada em outro micromputador com a seguinte configuração: Processador Athlon 4600+ com 2GB de memória RAM. O acesso ao middleware foi realizado através do padrão IEEE 802 em uma rede residencial local, por um roteador de 54 Mbps, ligado através de cabo.

A fim de calcular o impacto causado pela adição de mais uma camada de software (o Mult-I/O) no acesso aos dispositivos biomédicos, foram realizados testes na presença e na ausência do middleware. A Tabela 1 apresenta os resultados na ausência do Mult-I/O, com uma taxa de amostragem de 10 requisições:

Tabela 1. Acesso ao dispositivo na ausência do Mult-I/O

Dispositivo	Tempo de acesso (Milissegundos)
Invusor de insulina (atuador)	**203**
Sensor de glicose (sensor)	**2875**

Uma requisição é o tempo em que um dado é lido ou escrito no dispositivo e o Tempo de acesso, é o tempo médio calculado em cima das requisições.

A Tabela 2 apresenta os resultados na presença do Mult-I/O, com uma taxa de amostragem de 10 requisições:

Tabela 2. Acesso ao dispositivo na presença do Mult-I/O

Dispositivo	Tempo de acesso (Milissegundos)
Invusor de insulina (atuador)	**430**
Sensor de glicose (sensor)	**3650**

Neste caso da Tabela 2, o Tempo de acesso é o tempo de processamento de uma requisição ao dispositivo + o tempo de comunicação pela rede.

4 CONCLUSÕES

Desenvolver aplicações para acessar dispositivos de hardware não é uma tarefa trivial. Alguns obstáculos aparecem quando está se desenvolvendo, como permissões do SO (Sistema Operacional), requisitos de comunicação entre o SO e a aplicação, ou até mesmo problemas referentes à localização e reconhecimento das bibliotecas de acesso aos dispositivos. Esses problemas podem tirar o desenvolvedor do foco principal: Analisar e codificar o acesso aos dispositivos.

Quesitos como interoperabilidade, integração e encapsulamento das especificidades de comunicação, fornecidos pelo Mult-I/O, contribuem para a aproximação do desenvolvedor com esse foco principal.

Através do uso deste middleware, o custo no desenvolvimento e manutenção de sistemas corporativos pode ser reduzido, evitando que desenvolvedores trabalhem com diversas APIs voltadas para dispositivos.

AGRADECIMENTOS

Este trabalho contou com o apoio do Laboratório de Automação Hospitalar e Bioengenharia (LAHB) do Departamento de Engenharia de Computação e Automação da Universidade Federal do Rio Grande do Norte e do Laboratório de Inovação Tecnológica em Saúde do Hospital Universitário Onofre Lopes da Universidade Federal do Rio Grande do Norte.

REFERÊNCIAS BIBLIOGRÁFICAS

1. Apache. Apache Axis. Disponível em: http://ws.apache.org/axis/. Acesso em: 03 de março de 2010.
2. Deugd, Scott et AL. SODA: Service-Oriented Device Architecture, submetido no IEEE ComSoc and IEEE CS. University of Florida. 2006.
3. Hamilton, G., Mitchell, J. G., and Powell, M. L. 1993 Subcontract: a Flexible Base for Distributed Programming. Technical Report. UMI Order Number: TR-93-13., Sun Microsystems, Inc.
4. GoF. GAMMA, Erich, HELM, Richard, JOHNSON, Ralph, VLISSIDES, John. Design Patterns: Elements of Reusable Software. 18. ed. Reading: Addison Wesley Longman, Inc. 1998
5. Microsoft. Consuming MapPoint Web Service in PHP. Disponível em: http://msdn.microsoft.com/en-us/library/ms980207.aspx. Acesso em: 06 de março de 2010.
6. Sales, F. L. F. Gerência de Informação de Processos Industriais Usando Web Service.Trabalho de conclusão de curso. Centro Federal de Educação Tecnológica do Rio Grande do Norte. 2005.
7. Schmidt, D. and Suda, T. "An Object-Oriented Framework for Dynamically Configuring Extensible Distributed Systems".Distributed Systems Engineering Journal, Special Issue on Configurable Distributed Systems, 1995.

8. Scott de Deugd, Randy Carroll, Kevin Kelly, Bill Millett, Jeffrey Ricker, "SODA: Service Oriented Device Architecture," IEEE Pervasive Computing, vol. 5, no. 3, pp. 94-96, c3, July-Sept. 2006, doi:10.1109/MPRV.2006.59.

9. Stal, M. 2002. Web services: beyond component-based computing. Commun. ACM 45, 10 (Oct. 2002), 71-76. DOI= http://doi.acm.org/10.1145/570907.570934.

10. Valentim, R. A. M. et al. Um Modelo para Acesso a Dispositivos Biomédicos Dirigido a uma Arquitetura Orientada a Serviços. SHEWC'2008 - Safety, Health and Environmental World Congress. Rio de Janeiro, July 20 to 23, 2008.

11. W3C. Extensible Markup Language (XML). Disponível em: www.w3.org/XML/. Acesso em: 07 de março de 2010.

12. W3C. Simple Object Access Protocol (SOAP). Disponível em: http://www.w3.org/TR/soap/. Acesso em: 02 de março de 2010.

13. W3C. Web Services Description Language (WSDL). Disponível em: http://www.w3.org/TR/wsdl. Acesso em: 10 de março de 2010.