Research on Biomedical Engineering
rbejournal.org
Volume 33, Number 2, p. 166-172, 2017

Technical Communication
DOI: http://dx.doi.org/10.1590/2446-4740.09716

SOA-BD: Service Oriented Architecture for Biomedical Devices

João Marcos Teixeira Lacerda[1,2]*, Jailton Carlos de Paiva[1], Diego Rodrigues de Carvalho[1], Philippi Sedir Grilo de Morais[1], Yáskara Ygara Menescal Pinto Fernandes[1], Ricardo Alexsandro de Medeiros Valentim[1]

[1] *Laboratory for Technological Innovation in Healthcare, Federal University of Rio Grande do Norte, Natal, RN Brazil.*
[2] *Federal Institute of Rio Grande do Norte, Ceará Mirim, RN, Brazil.*

Abstract **Introduction:** The communication of information systems with biomedical devices has become complex not only due to the existence of several private communication protocols, but also to the immutable way that software is embedded into these devices. In this sense, this paper proposes a service-oriented architecture to access biomedical devices as a way to abstract the mechanisms of writing and reading data from these devices, thus contributing to enable the focus of the development team of biomedical software to be intended for its functional requirements, i.e. business rules relevant to the problem domain. **Methods:** The SOA-BD architecture consists of five main components: A Web Service for transport and conversion of the device data, Communication Protocols to access the devices, Data Parsers to preprocess data, a Device Repository to store data and transmitted information and Error handling, for error handling of these information. For the development of SOA-BD, technologies such as the XML language and the Java programming language were used. Besides, Software Engineering concepts such as Design Patterns were also used. For the validation of this work, data has been collected from vital sign monitors in an Intensive Care Unit using HL7 standards. **Results:** The tests obtained a difference of about only 1 second in terms of response time with the use of SOA-BD. **Conclusion:** SOA-BD achieves important results such as the reduction on the access protocol complexity, the opportunity for treating patients over long distances, allowing easier development of monitoring applications and interoperability with biomedical devices from diverse manufacturers.

Keywords SOA-BD, Telehealth, Biomedical devices communication, Service Oriented Architecture, HL7, Intensive care unit.

Introduction

The rapid spread of new hardware devices on the market has increased the complexity of communication between them (Deugd et al., 2006). Many hardware devices provide their functionality through proprietary protocols, that is, drivers that are dependent on a specific operating system platform (Valentim et al., 2008). Given this closed and heterogeneous hardware environment, an issue needs to be discussed, which is interoperability. This problem is also present in the biomedical field, because many of the biomedical devices have embedded software that doesn't change during the lifetime of the device, and the data streams are proprietary to the device vendor (Degaspari, 2012).

Furthermore, developing applications to access biomedical devices remotely is not a trivial task. Some obstacles appear during the development, such as Operating System permissions, requirements for communication between the OS and the application, or even issues relating to the location and recognition of libraries to access the devices. These problems can take away the developer's main focus, which is to create an application to access biomedical devices.

The goal of this work is to present an architecture, called SOA-BD, to abstract the complexity in the access to biomedical devices so that the devices can be viewed as a network service, which can be accessed through standard interfaces provided in SOA (Software Oriented Architecture), using remote calls (RPC - *Remote Procedure Call),* in order to enable a medical systems development team to focus only on their functional requirements instead of device specific characteristics.

How to cite this article: Lacerda JMT, Paiva JC, Carvalho DR, Morais PSG, Fernandes YYMP, Valentim RAM. SOA-BD: Service Oriented Architecture for Biomedical Devices. Res Biomed Eng. 2017; 33(2):166-172. DOI: 10.1590/2446-4740.09716

**Corresponding author:* Instituto Federal de Educação Ciência e Tecnologia do Rio Grande do Norte, Campus Ceará Mirim, BR-406, Km 145, Bairro Planalto, CEP 59570-000, Ceará Mirim, RN, Brazil. E-mail: joao.lacerda@ifrn.edu.br

Received: 08 December 2016 / *Accepted:* 21 April 2017

There are similar works in the literature, as in Gregorczyk et al. (2012) that uses SOA to integrate medical devices and systems in operating rooms, or Thelen et al. (2015) that uses medical standards to integrate emergency medical devices. However, this work is different from the ones mentioned above because it addresses implementation issues not addressed in these articles, such as details on the protocol of the exchanged messages by the architecture or the way in which data is stored, which may make it easier to use the architecture. Besides, only Thelen et al. (2015) presents quantitative validation data.

Methods

Architecture

SOA-BD was designed considering a heterogeneous hospital environment, both in relation to applications that perform access to biomedical devices, and the devices themselves, as illustrated in Figure 1. The purpose was to specify the architecture of SOA-BD as an element of access between applications and devices in order to simplify the communication process among them. Thus, hospital application developers using programming languages such as Java, C# .NET, Python or any other language that supports SOA (Software Oriented Architecture), can access the hardware without the need to know specific technical details of biomedical devices.

The SOA-BD architecture consists of five main components: Web Service (based on SOA), Communication Interfaces (called Mult I/O API), Data Parsers (Mult Parser API), a Device Repository and Error handling. The objective of the Web Service is to transform the client's information (that is coded in XML) into the default codification of the architecture. This codification is based on Unicode (UTF-8). The main components of the Web Service are: The SOAP protocol (*Simple Object Access Protocol*) and WSDL interfaces (*Web Services Description Language*), both based in SOA, formatted on XML language, deployed by the Web Services technology (Shall et al., 2006). The Figure 2 illustrates a request and response message in the SOAP protocol on SOA-BD.

The Mult I/O API has the function of converting the information from the default coding of the architecture to the communication protocol used by biomedical devices. The devices normally use one of the following: standard industrial communication, such as USB and IEEE 802.x, or biomedical standards, such as HL7 (Health..., 2016) and DICOM (Digital..., 2016). To permute the different communication protocols, two Design Patterns

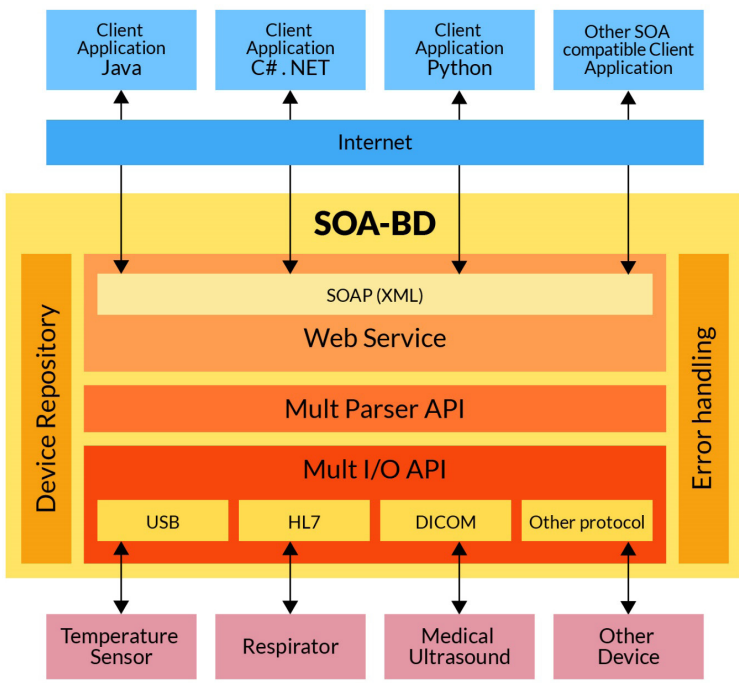

Figure 1. SOA-BD architecture. The SOA-BD architecture as an element of access between applications with different program languages (like Java, C# .NET, Python or others program languages that supports SOA – Software Oriented Architecture) and devices (Temperature Sensor, Respirator, Medical Ultrasound or other device) through different communication protocols in order to make the communication process heterogeneous between them. The SOA-BD architecture consists of five main components: Web Service – transport and conversion of devices data, Communication Protocols (Mult I/O API) – access the devices, Data Parsers (Mult Parser API) – To process RAW data, Device Repository – store data and transmitted information and Error handling – error treatment on transmission of these information.

```
SOAP Request Envelope:

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:q0="http://soadb.lais.ufrn.br" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soapenv:Body>
    <q0:readDevice>
      <q0:device>4</q0:device>
    </q0:readDevice>
  </soapenv:Body>
</soapenv:Envelope>

SOAP Response Envelope:

<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/">
  <soapenv:Body>
    <ns:readDeviceResponse xmlns:ns="http://soadb.lais.ufrn.br">
      <ns:return>MSH|^~\&|EVOL|DIXTAL|||[redacted]||UDM^Q05|DIX123456|PA|2.3;
URD|[redacted]|R||RES||D|T;
DSP|||\H\\F\       \F\ FC     \F\ SpO2   \F\ NIBPs  \F\ NIBPd  \F\ TA     \F\ \N\;
DSP|||\F\ 07/03 \F\ 23:00 \F\92      \F\ 97      \F\ 102     \F\ 60      \F\ 36.6    \F\;
DSP|||\F\ 07/03 \F\ 22:00 \F\0       \F\ 100     \F\ 95      \F\ 59      \F\ 36.7    \F\;
DSP|||\F\ 07/03 \F\ 21:00 \F\130     \F\         \F\ 105     \F\ 61      \F\ 36.8    \F\
</ns:return>
    </ns:readDeviceResponse>
  </soapenv:Body>
</soapenv:Envelope>
```

Figure 2. Request and response message in the SOAP protocol in SOA-BD. The utilized method on request (SOAP Request Envelope) has the "readDevice". The desired device for access has tag "4". The response message (SOAP Response Envelope) returned the data from a vital sign monitor on HL7 2.3 standard. Black stripes are not to expose patient information.

were applied, Factory Method and Abstract Factory (Gamma et al., 1995), which provide a high degree of decoupling. This way, for a new communication protocol to be integrated on SOA-BD, it needs to implement a Programming Interface (Oracle..., 2016). In this perspective, the architecture was specified to create, during the process of establishing a communication, instances of library communications used by biomedical devices, making use of the mentioned design patterns at runtime.

The Mult Parser API is an optional component that allows the use of preprocessed data. This component provides pluggable implementations of parsers (RAW data converter in preprocessed data) according to the communication protocol used by the medical device. It is used in case the medical system developer that uses SOA-BD wants to exchange messages with parsed data instead of RAW Data (shown in Figure 2). For each protocol used by a medical device there may be a Data Parser API. The Parsers exchange is performed similarly to the protocol exchange of the Mult I/O API.

The Device Repository has the function of storing device information (transmitted data and its metadata). This metadata matches device identification, communication ports, communication address and protocol. To manage the metadata an Administration Application is available with the following operations: register, list, delete and update. The Figure 3 illustrates the Device Repository.

The Error handling component is responsible for informing the client application that errors occurred in communication with other devices. These errors can happen due to device mal functioning or transmission errors. The error detection is performed by the communication protocols of the devices. The error treatment of the architecture is accomplished by the methods that capture exceptions.

Access and device mapping

Medical devices are previously registered and configured in SOA-BD by the Administration Application. Access takes place through any client application that supports the consumption of Web Services, which can be, for example, in a computer or smartphone. SOA-BD services are accessed through Universal Description, Discovery and Integration - UDDI (Advancing..., 2017), which enables a dynamic scenario. It means that even in case in which the client device is moved, access to SOA-BD can still be performed.

In order to make the device unique, the physical address (MAC Address) has been included as the device ID. That way, even if the medical device is moved somewhere else, it remains unique for both SOA-BD and client applications that consume its services. Medical device information is provided to client applications through the "returnDevices" service, and through this service, the client application can choose one or more devices to gain access.

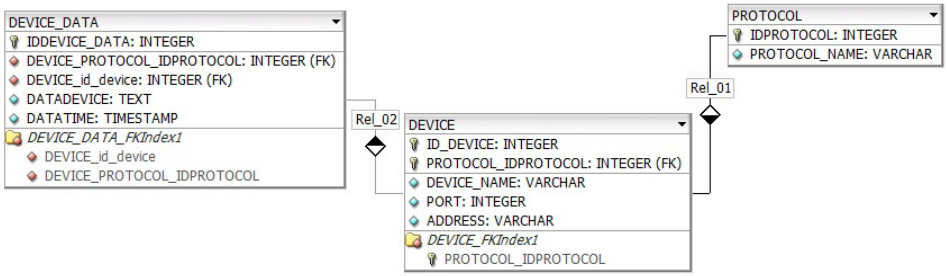

Figure 3. The Device Repository illustrates the data model of Repository of Devices used in SOA-BD (above) to store information regarding the devices. Tables are DEVICE, PROTOCOL and DEVICE_DATA. The tables DEVICE and PROTOCOL store device's metadata (identification, port, address and protocol) and table DEVICE_DATA store transmitted data. The fields of DEVICE_DATA are: DATADEVICE to store the data content and DATATIME to store the exact time in which data has been inserted. The ID_DEVICE field always refers to the MAC address of the device.

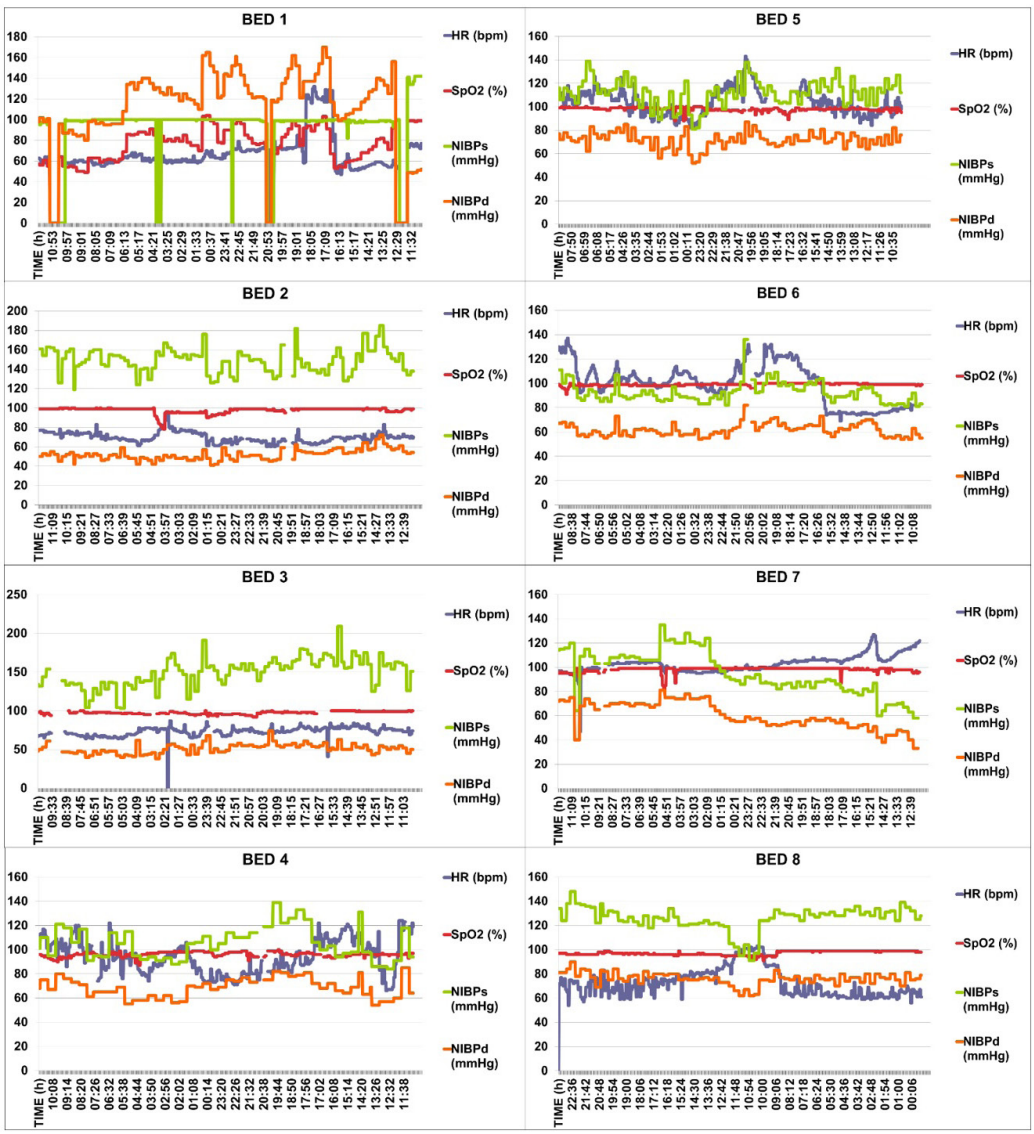

Figure 4. Collected data by the 8 vital sign monitors chosen on the ICU (Intense Care Unit) beds for validation of SOA-BD. Each framework answers a bed from the ICU. The null or zero values characterize in its great majority the patient hygiene or some other event that forced the removal of the patient's sensor. The collected vital signs were the Heart Rate (HR), Pulse Oximetry (SpO2), Non-Invasive Blood Pressure Systolic and Diastolic (NIBPs and NIBPd).

Implementation

The architecture of SOA-BD has been implemented in the Java programming language. The Web Service component was developed based on the framework SOAP Axis 2 (Apache..., 2016a), which can implement the SOAP protocol and execute under the Web Apache Tomcat Server (Apache..., 2016b). Tomcat has the function of answering the client application requests. The whole conception was addressed to ensure a low coupling between the medical devices and the client applications. For the client applications to have access to SOA-BD, a class has been made to implement the mentioned programming interface and convert it into WSDL by Axis 2.

Validation

The data used for validation of SOA-BD was collected from within an Intensive Care Unit (ICU) at Natal-RN, Brazil. This data is comprised of vital signs from DIXTAL monitors, model 2020, placed at the beds of the ICU. This data was transmitted from monitors, on HL7 communication standards. 8 beds were used, which represent nearly 52% of respective ICU beds. The transmitted and collected parameters were Heart Rate (HR), Pulse Oximetry (Sp02), Non-Invasive Blood Pressure Systolic and Diastolic (NIBPs and NIBPd). The reading delay of these customized parameters were 1 minute in each monitor, in a total period of nearly 24 hours. The Figure 4 illustrates these data.

Two scenarios were used for the accomplishment of the data receiving. The first, a conventional environment, by a monitoring central (Compatible with HL7 standards) that reads transmitted data by the monitors. The monitoring central used was a computer with an Intel Core i7 processor, 16 GB of RAM, and running the Microsoft Windows 10-64 bits operating system. The second scenario drew on the same structure of scenario 1, with the implementation of SOA-BD. Both scenarios used the HL7 2.x standards. For data receiving on scenario 2, eight mobile devices with Android 6.0 were used. The Figure 5 illustrates the scenarios.

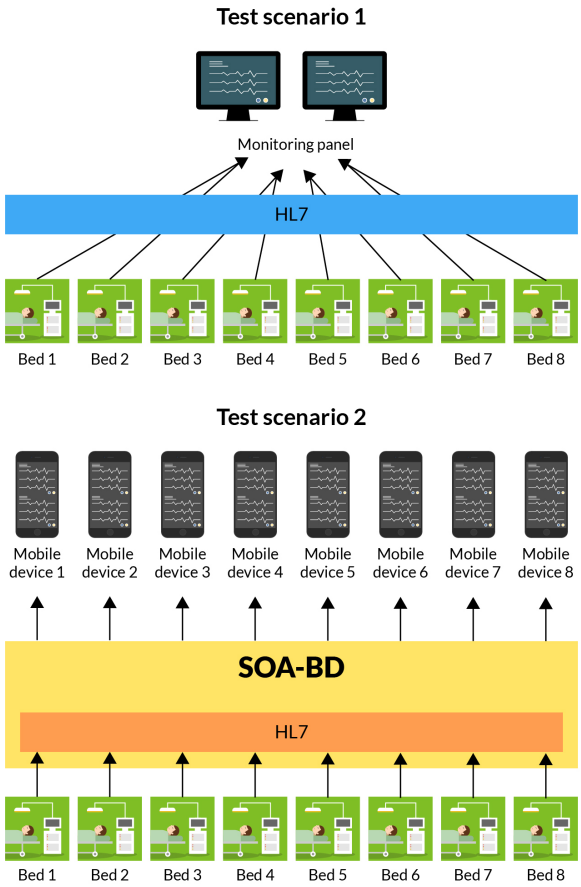

Figure 5. Illustration of the scenarios on this work tests. On scenario 1, the vital data from the patient's beds were received by a monitoring central compatible with the HL7 standard. On scenario 2, SOA-BD is utilized to receive the data from beds from Intensive Care Unit, with the HL7 standard built-in. For each, a mobile device was used to access the data transmitted by the bed monitors of the Intensive Care Unit.

Table 1. Average response times regarding scenario 1 and scenario 2 and their respective standard deviations, in addition to the difference in the average response time between scenario 2 and scenario 1.

	Scenario 1 (Local)		Scenario 2 (SOA-BD)		Scenario 1 x Scenario 2
	Average response time (seconds)	Standard deviation	Average response time (seconds)	Standard deviation	Average response time difference (seconds)
Bed 1	1.54	0.73	2.09	0.70	0.55
Bed 2	1.59	0.76	2.16	0.81	0.57
Bed 3	1.66	0.65	2.27	0.58	0.61
Bed 4	1.67	0.65	2.26	0.55	0.59
Bed 5	1.68	0.69	2.60	0.84	0.92
Bed 6	1.69	0.67	2.40	0.66	0.71
Bed 7	1.68	0.67	2.39	0.69	0.71
Bed 8	1.62	0.70	2.20	0.66	0.58

Results

Table 1 exemplifies the average response time regarding scenario 1 and scenario 2 and its respective standard deviations, in addition to the average response time between scenario 2 in relation to scenario 1. The objective was to estimate the impact of SOA-BD usage on access to vital sign monitors. The response time regarding scenario 1 is the time between data request from the monitoring central to the vital sign monitor, processing and response. The response time regarding scenario 2 is the request time by the mobile application of the health professional to the SOA-BD plus the request time of SOA-BD to the vital sign monitors, processing and response.

Discussion

According to the results, the addition of SOA-BD on access to the vital signs monitor generated a difference in the range of 0.5 seconds to less than 1 second in terms of response time. These values should not be significant in many scenarios, which makes SOA-BD little costly to use. Besides little costly, the usage of the architecture resulted in advantages, such as: reduction on access protocol complexity, allowing easier development of monitoring applications, interoperability with diverse manufacturers devices and the possibility of treating patients over long distances. The main contribution of this work is to reduce the complexity in the access of biomedical devices that may contribute to increased quality in patient monitoring applications and decrease in maintenance of biomedical data monitoring systems. For more information about the SOA-BD project, access its homepage (Universidade..., 2017).

References

Advancing Open Standards for the Information Society. OASIS UDDI specification TC [Internet]. Burlington: OASIS; 2017. [cited 2017 Mar 3]. Available from: https://www.oasis-open.org/committees/uddi-spec/faq.php

Apache Software Foundation. Welcome to Apache Axis2/Java [Internet]. Wakefield: Apache Software Foundation; 2016a. [cited 2016 Nov 22] Available from: https://axis.apache.org/axis2/java/core/

Apache Software Foundation. Apache Tomcat® - Welcome! [Internet]. Wakefield: Apache Software Foundation; 2016b. [cited 2016 Nov 22]. Available from: http://tomcat.apache.org/

Degaspari J. Device connectivity. Virtua links disparate biomedical devices to its enterprise-wide EMRs. Healthcare informatics: the Business Magazine for Information and Communication Systems. 2012;29(5):43-4. PMID: 22655446.

Deugd S, Carroll R, Kelly K, Millett B, Ricker J. SODA: Service Oriented Device Architecture. IEEE Pervasive Computing. 2006; 5(3):94-6. http://dx.doi.org/10.1109/MPRV.2006.59.

Digital Imaging and Communications in Medicine. DICOM Homepage [Internet]. Rosslyn: DICOM; 2016. [cited 2016 Nov 8]. Available from: http://dicom.nema.org/

Gamma E, Helm R, Johnson R, Vlissides J. Design patterns: elements of reusable object-oriented software. Boston: Addison-Wesley; 1995.

Gregorczyk D, Bußhaus T, Fischer S. A proof of concept for medical device integration using web services. In: Proceedings of the 9th International Multi-Conference on Systems, Signals Devices; 2012 Mar 20-23; Chemnitz, Germany. New York: IEEE; 2012. http://dx.doi.org/10.1109/SSD.2012.6198124.

Health Level Seven International. Health Level Seven International – Homepage [Internet]. Ann Arbor: HL7; 2016. [cited 2016 Nov 8]. Available from: http://www.hl7.org

Oracle Corporation. Interfaces [Internet]. Redwood City: Oracle Corporation; 2016. [cited 2016 Nov 08]. Available from: https://docs.oracle.com/javase/tutorial/java/IandI/createinterface.html

Schall D, Aiello M, Dustdar S. Web services on embedded devices. International Journal of Web Information Systems. 2006; 1(2):45-50. http://dx.doi.org/10.1108/17440080680000100.

Thelen S, Czaplik M, Meisen P, Schilberg D, Jeschke S. Using off-the-shelf medical devices for biomedical signal monitoring in a telemedicine system for emergency medical services. IEEE Journal of Biomedical and Health Informatics. 2015; 19(1):117-23. PMid:25312967. http://dx.doi.org/10.1109/JBHI.2014.2361775.

Universidade Federal do Rio Grande do Norte. Hospital Universitário Onofre Lopes. Laboratório de Inovação Tecnológica em Saúde. Projeto SOA-BD [Internet]. Natal: LAIS; 2016. [cited 2017 Mar 10]. Available from: http://www.lais.huol.ufrn.br/index.php/component/k2/item/162?show=show

Valentim RAM, Morais AHF, Brandão GB, Guerreiro AMG, Xavier MA, Araújo CAP. MP-HA: Multicycles Protocol for Hospital Automation over multicast with IEEE 802.3. In: Proceedings of the IEEE 6th International Conference on Industrial Informatics; 13-16 July 2008; Daejeon, Korea. New York: IEEE; 2008. p. 979-84.