Using Brazilian Digital TV to Integrate Health Care Services Embedded in Medical Commercial Devices

Vandermi Silva, Ricardo Erikson Veras De Sena Rosa
and Vicente Ferreira De Lucena Jr.
*Federal University of Amazonas, UFAM,
Brazil*

1. Introduction

The Digital TV (DTV) System is getting a stimulus within industry and in Brazil that stimulus is based on the belief that this new system will be successful. Several proposals for extending the potential of interactivity, for providing innovative services are available today. In this context, health care is a concept being studied. The health care concept is relative to care about health, in preventing diseases, quality of life and with applications to hospitals, elderly care centres and so on. Integrating health care technology with home devices is a new trend. Thus this chapter shows how the software programmer can integrating devices and sensors with the architecture based in filters presented here.

Our solution is a software platform based on the home gateway, which is in charge of all medical devices interconnecting with the DTV. As proof of the concept, we will build two implementation scenarios. One for measuring heart beat frequency and the other for measuring blood pressure. All the procedures related to the integration are detailed. We believe this work will be useful in this new field, bringing alternative ways for using common devices to provide quality of life to the home user.

The chapter is divided into sections containing foundations, the DTV architecture of health care, construction on a prototype of health care and tests and results, to present the Brazilian standard for digital TV applications and the concepts of health. An overview of the integrated system for medical devices in the architecture section will show an architecture based on filters through all the details and technologies used. In the section on building the prototype we will present how to configure a development environment and how to implement a simple example using the Java programming language and the language Lua. Finally, in the tests and results section illustrations of a system running on a TV will be presented with the Brazilian digital TV standard installed.

1.1 Foundations

The technology improvements in wireless communications and the increasing usage of commercial medical devices to monitor patients have contributed to the development of health care systems targeted to home usage. Indeed, the actual state-of-the-art of these technologies is sufficient to construct highly integrated systems for measuring and collecting patients' data aimed at assisting in diagnoses and prevention of diseases.

In Brazil, according to the Brazilian Institute of Geography and Statistics (IBGE), over 97% of the population living in urban areas have access to television and certainly that percentage should be reflected in the usage of the new digital TV system as soon as it is implemented all over Brazil. Similar studies show that the number of active mobile phones is almost double that of the Brazilian population. This means that almost every family has more than one mobile device available at home.

The IBGE statistics also show that the number people aged 60 years and over in Brazil in 2009 accounted for more than 19 million people. This is 11% of the Brazilian population and the number continues to grow. This study also shows that this population is predicted to reach 30 million over the next 20 years or almost 15% of the population at the end of this period. This leads us to believe that the demand for specialized health services will grow in proportion to the growth of the elderly population, affecting health care in hospitals and nursing homes.

In fact, there is a technological opportunity here. On one hand the market needs applications that automate the processes of monitoring patients remotely to allow for better care quality, reduced queues and costs at the clinics, and on the other hand the Brazilian population has access to new technologies able to construct new equipment and systems that may help solve known problems.

1.1.1 Brazilian DTV

DTV technology is composed of a set of elements comprising signal reception and transmission of the audio and video with digital modulation, a software layer for integrating between the application layer and the hardware, called middleware, and the set of applications. The advance of new algorithms for encoding and decoding audio and video, the transformation of the analogue signal to digital and interactivity via a return channel, allow the viewer a new perspective which will now interact with the DTV system and its programming.

The specification of ISDB-TB standards was based on the Japanese Integrated Services Digital Broadcasting Terrestrial (ISDB-T), following a decision by the Brazilian Digital TV Forum, created by a group led by the National Telecommunications Agency (ANATEL) and the Ministry of Communications. The Brazilian system has been specified with system changes in the compression of audio and video that will now come with encoding H264 and HE-AAC v.2, with speeds up to 30 frames per second.

After defining the standards for all architecture levels of the STB, the Brazilian consortium took charge of the technology to specify the basis of interactivity. In generic architecture of digital TV receivers, the middleware is represented by a software stack that provides an interactive interface for applications to access resources, system services and iDTV. In this regard, the Forum established the Brazilian Ginga as an open standard for the production of interactive programmes and its implementation has been divided into two parts: Ginga-NCL and Ginga-J.

Figure 1 illustrates the Brazilian middleware inserted into the architecture of the STB with ISDB-TB. The subsystem Ginga-J supports Java-based programming, already the Ginga-NCL is focused on the temporal synchronization of media with interactivity. Ginga-NCL follows the model of the Brazilian language NCL (Nested Context Language) that allows the manipulation of media properties to be displayed, event handling, access to information services, etc. In addition to the NCL, the declarative environment of middleware includes another Brazilian programme language called Lua. This subsystem is frequently used and recently Ginga-NCL was recognized as a standard ITU for IPTV.

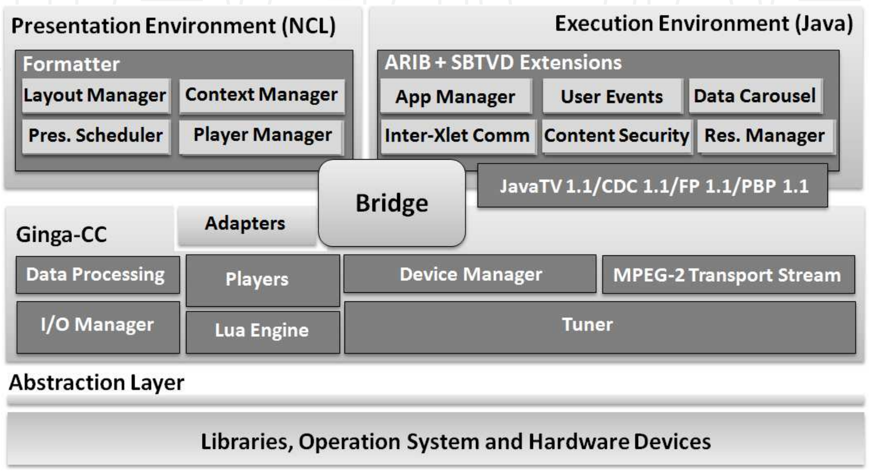


Fig. 1. Ginga middleware architecture.

From the point of view of procedural middleware, Ginga-J is modelled similarly to existing standards, such as JavaTV. The first subsystem specifications included in the software stack of Ginga-J are a set of APIs (Application Programming Interface) for event handling, access to information services, media control, graphical user interface, etc. However, some components of the APIs' platform have been replaced by third-party royalty-free. The specification of the new platform, released in 2009, is based on JavaDTV technology, derived from the Java ME specification for portable consumer devices and modified to suit the peculiarities of the Brazilian digital TV platform.

Currently, the Ginga-J is normalized by the ABNT standard. Two important elements stand out in the Brazilian middleware compared to others in existence: the bridge Ginga-NCL/Ginga-J integration and support communication with multiple devices. The first point refers to a mechanism that allows applications to produce hybrid Ginga-J/Ginga-NCL taking advantage of NCL extensions that allow the application to pass control to other components, that can be written in Java. This enables us to create complete applications that exploit the potential of the platform in its entirety. This mechanism is called common core present in the Ginga, Ginga Common Core or Ginga-CC (see Figure 1): a stack of components that perform functions common to both subsystems.

The second issue is an extension of NCL to enable interaction with other TV devices, such as mobile phones or smartphones. This feature reflects the current trend in digital TV toward connectivity with other devices and can be seen as a potential for the use of other appliances in conjunction with the TV to promote innovative services for users, for example a health care service that integrates mobile systems, medical devices and DTV.

1.1.2 Residential gateway

The Residential Gateway (RG) is a device that interconnects local area network devices with the external environment. The RG focuses management of various services of the residents, providing a single interface to access all devices. In a home network, a given service provider linked through communications technology delivers users functionality implemented by the device, this includes four categories of services that can be distributed on a home network: entertainment (home entertainment), communication (home communications), computer (home computing) and management and monitoring (home monitoring and management). Figure 2 illustrates the role of GR in a home network.

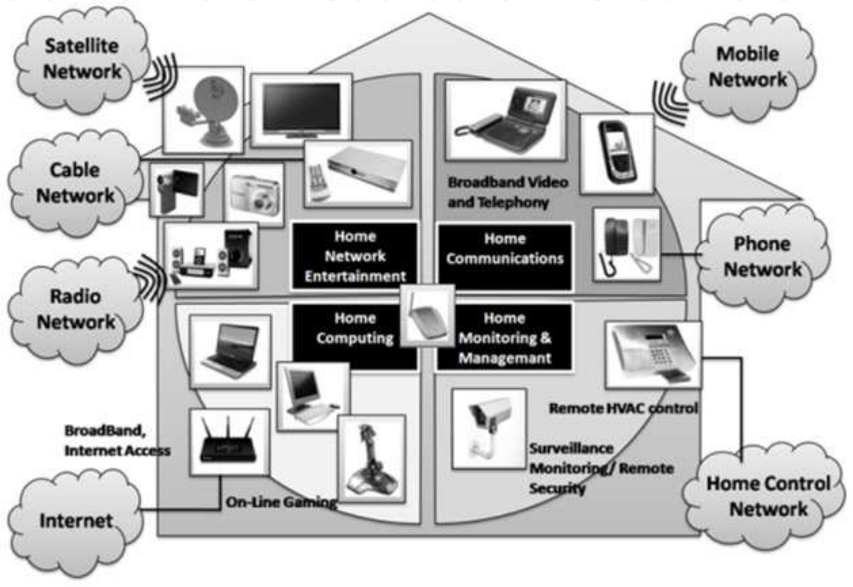


Fig. 2. Network services centralized in the RG. Adapted from O. B. Maia.

To centre on RGfor all types of services, considering the particularities of each, it is necessary to have a software platform that meets the needs of the home environment. Some studies in the literature do an analysis of existing technologies, among which are: Jini, UPnP (Universal Plug and Play) and OSGi (Open Services Gateway Initiative).

To facilitate the integration with medical devices and scenarios for the RG health care is essential. Moreover, as an extension of interactive digital TV, a similar proposal is currently being studied with regard to RG , that being can services be accessed from all devices in the home through DTV.

1.1.3 Medical devices

Currently there are a range of medical devices on the market that can be integrated with systems for the diagnosis and monitoring of chronic diseases. Among these devices are automatic blood pressure meters and pulseoximeters. The combined use of medical devices in a home network for treatment and prevention of disease is already quite common in developed countries like the US and EU countries. In these countries, it is possible to monitor a patient in his own residence. In Brazil, the initiative at the State University of Campinas (Unicamp) which is implementing the concept of a digital city where a backbone will connect all essential services, such as rescue and police, in a short period of time will also be a reality. Some of the most widely used medical devices are shown in Figure 3. There are data collection devices for systolic and diastolic pressure, metering devices of heartbeat and of insulin in the literature able to be integrated with mobile phones or even with small devices, such as a watch.

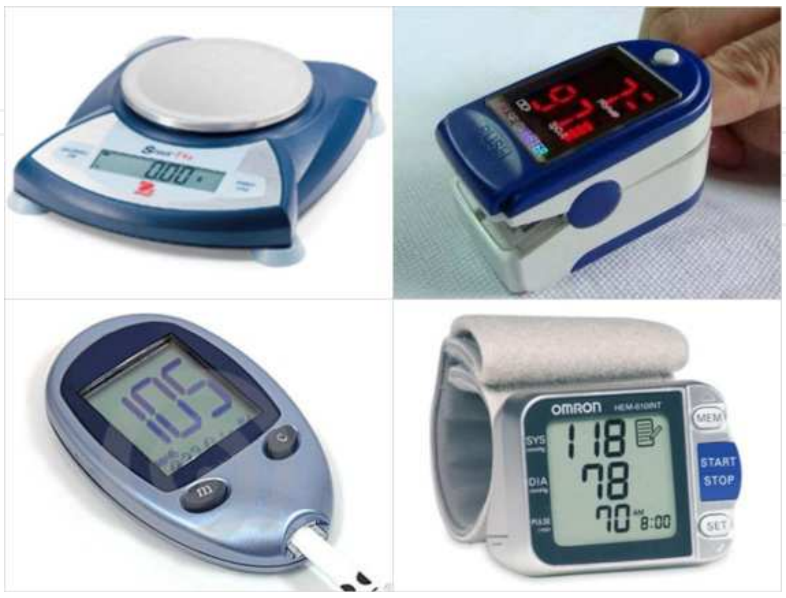


Fig. 3. Example of medical devices.

1.2 DTV Health care architecture

The architectural model of this paper is based on the pipes and filters architecture, which consists of treating the data by applying filters at various levels and transforming the raw data collected from the devices in readable information to the user. Pipes and filters consider the existence of a network through which data flows from one end to another source and the destination data stream undergoes transformations by means of filters. In this work, pipes and filters are unidirectional, leading and treating the data between the source (sensors for medical devices) and destination (RG and DTV).

From the architectural design of the model presented in Figure 4 the system uses two data filters who work in the transformation of the data received from the devices and communication protocols defining a parser that converts the data format to another representation. After the work of these filters, a common document is generated, validated and stored in the database.

The data flow that goes from sensors to the database first passes through a filter of protocols that treat the raw data to then process it into a sub-parser module that generates the document containing the sensor readings. The standard representation generated by the parser is checked by sub-module validator and the result is stored in the database.

From the generation of a data document that can be accessed by all modules of the validator and confirmation as to the origin and formatting of the document, the second filter data using the module action queries the document to verify that the values in pressure and heart rate are consistent with those stipulated by the health professional.

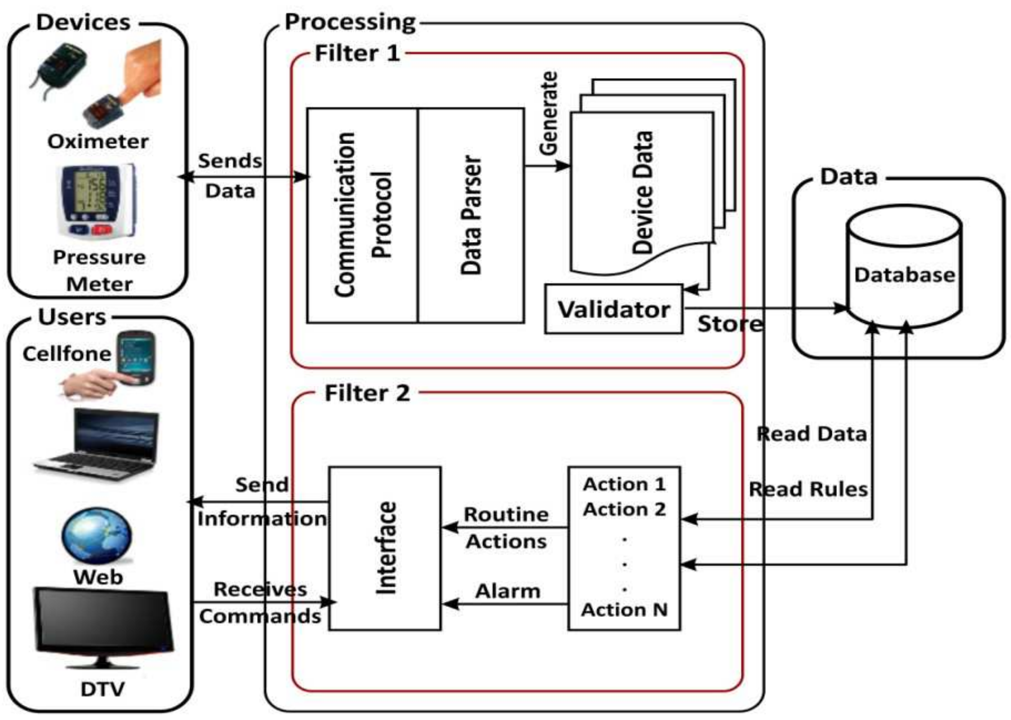


Fig. 4. Proposed architecture based in filters.

Through an interface with the devices that display data to the user, such as phones, computers connected to the Web and STB platforms with standard Brazilian DTV, the messaging subsystem, through action routines, performs the task of sending SMS messages (or prompts for TVDi).

The action module needs to check a table of rules for sending messages to the cell and the DTV. Therefore, based on the literature found in the area of health, a document containing such rules should be developed and stored in the database. Two examples of such data can be seen in Tables 1 and 2, respectively, showing the maximum range of the expected heart rate for an individual exercise and classification of blood pressure in adults over 18 years.

To illustrate the process, assume a scenario of monitoring the physical activity of a person on a treadmill. The user while performing the activity with a medical device (heart rate meter) attached generates raw data that are passed to the controller (a filter in Figure 4), enabling the execution of the components of the protocol level, which will address the data and store the useful information in the database. Then, obeying the rule base previously stored, the system (two components of the filter in Figure 4) checks the new data, sending the inferences made from the base of rules applied to the corresponding information. The result could be a message that appears on the TVDi, such as: "slow down the activity you are doing is too much physical effort."

The main challenges to implementing the solution are the integration of conventional medical devices with DTV, the synchronization between the devices when accessing the database and the implementation of common rules for the preparation of messages to be

Age	Max Freq.	Ideal Freq. 75%	Target zone between 70% and 80%
20	200	150	140 and 170
25	195	146	137 and 166
30	190	142	133 and 162
35	185	139	130 and 157
40	180	135	126 and 153
45	175	131	123 and 149
50	170	127	119 and 145
60	160	120	112 and 136
65	155	116	109 and 132
70	150	116	105 and 128

Table 1. Maximum heart rate range. [29]

Systolic	Diastolic	Category
greater than 130	less than 85	Normal
130-139	85-89	Normal Height
140-159	90-99	Mild hypertension (Stage 1)
160-179	100-119	Moderate hypertension (Stage 2)
180-209	110-119	Severe hypertension (Stage 3)
equal to the greater 210	equal to the greater 210	Hypertension very severe (Stage 4)

Table 2. Classification of blood pressure in adults aged less than 18 years. [28]

posted on the display devices. This is partly because there is no standardization of data sent by the devices, because each manufacturer uses its own standards and protocols for generation and transmission of sensor data.

1.2.1 Scenarios for architecture implementation

One scenario that can be used for implementation is the use of a mobile phone to send alert messages to users of a health care system. Thus, a module can send messages using a messaging library that can be integrated into the architecture of GR as a return channel of the application. In this study we used an API written in Java that can be found at http://smslib.org/. In the section on construction of the prototype we will demonstrate how to use the API.

The architecture scenario of the module Short Message System (SMS) is illustrated in Figure 5 and runs on the layer of JVM (Java Virtual Machine) installed on GR. Thus, a text message can be triggered for devices registered with the GR, using the infrastructure network of Global System for Mobile Communications (GSM) and its communication protocols.

In this scenario presented in Figure 5, two mobile devices must be present: a cell associated with the GR and another target, whose number is stored in the database. If any medical device presents a reading in a range of risk, considering the rules of the tables on module action, the GR uses a cellular network as an output for sending data over the Internet. The

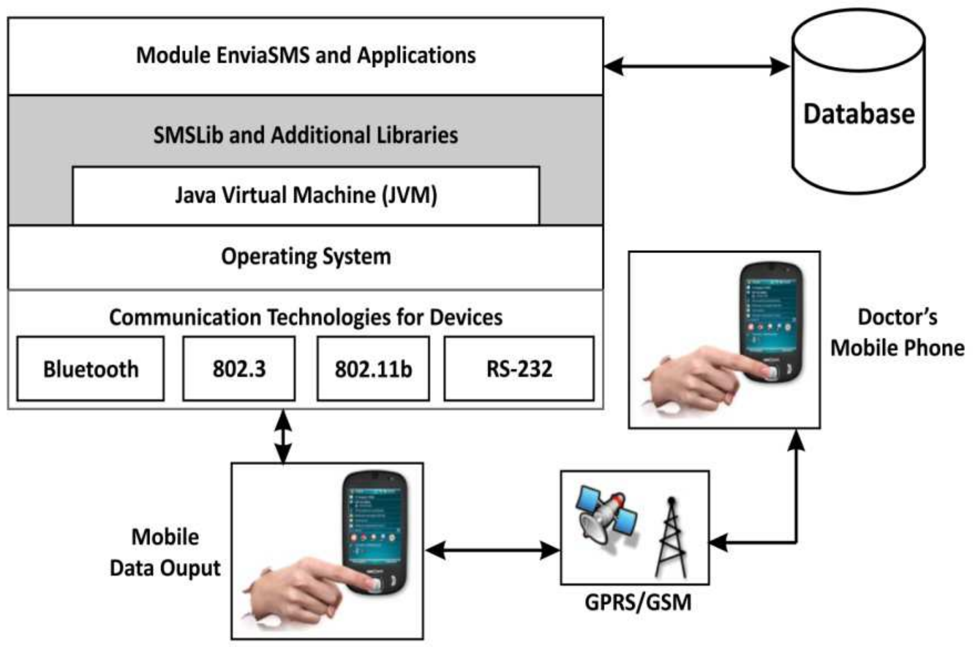

Module EnviaSMS and Applications
SMSLib and Additional Libraries
Java Virtual Machine (JVM)
Operating System
Communication Technologies for Devices
Bluetooth
802.3
802.11b
RS-232
Database
Doctor's Mobile Phone
Mobile Data Ouput
GPRS/GSM

Fig. 5. Module for sending SMS messages.

SMSLib enable communication through this module EnviaSMS. Once connected to the cell output, the GR sends the alert message (blood pressure or heart rate) for the device registered in the database, which can be the phone of the doctor or person responsible for the monitored patient.

For the presentation of messages on a mobile device, Micro-edition Java (J2ME) can be used, which has a Java virtual machine for devices with low processing power (KVM). From the application layer there is access to layers of Profile and Mobile Information Device Profile (MIDP) that allow direct access to the layer Connected Limited Device Configuration (CLDC), responsible for device configurations accessible to the Kilobyte Virtual Machine (KVM), a custom Java virtual machine to the limitations of devices with low processing power.

The application layer is limited to the use of CLDC with MIDP, to reduce the cost of memory and processing power, enabling low-cost handsets to be used in the implementation of the prototypes. Figure 6 illustrates the modules that make up the JME platform, inserted into the overall system architecture.

Finally, a presentation module of the data in DTV can be built using the architecture shown in Figure 7. Within this module is an interface to an application using DTV Brazilian Ginga middleware technologies. For the two sub-specifications of middleware, applications were built corresponding: to the Java part, considering also the unavailability of an open platform 100% compatible with the Ginga-J, we adopted a subset of the MHP middleware API. For the environment we created a declarative interface for data visualization in NCLua. The data is accessed via APIs specific to each platform.

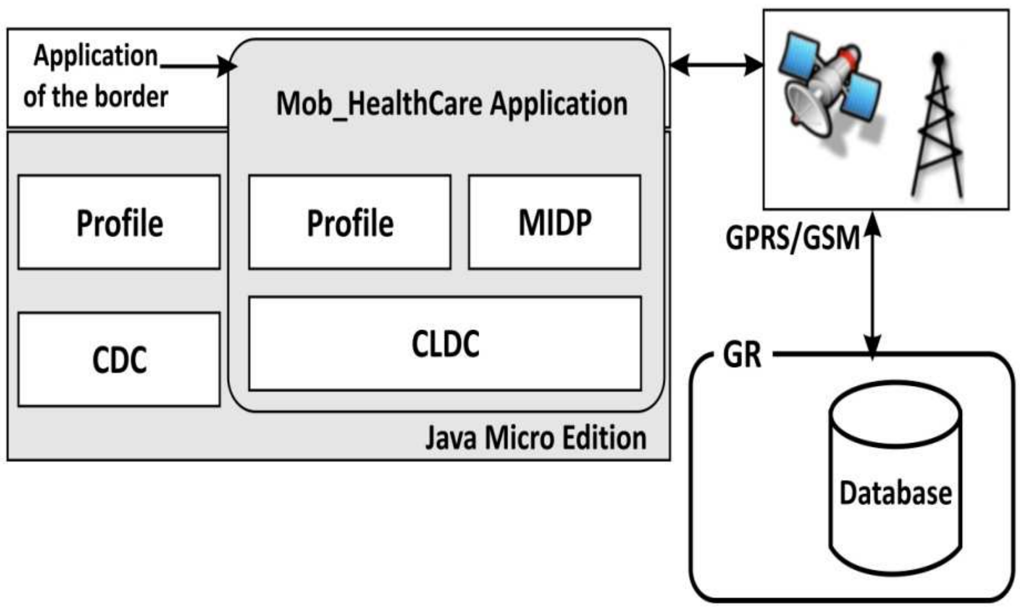


Fig. 6. Module of the mobile device architecture J2ME.

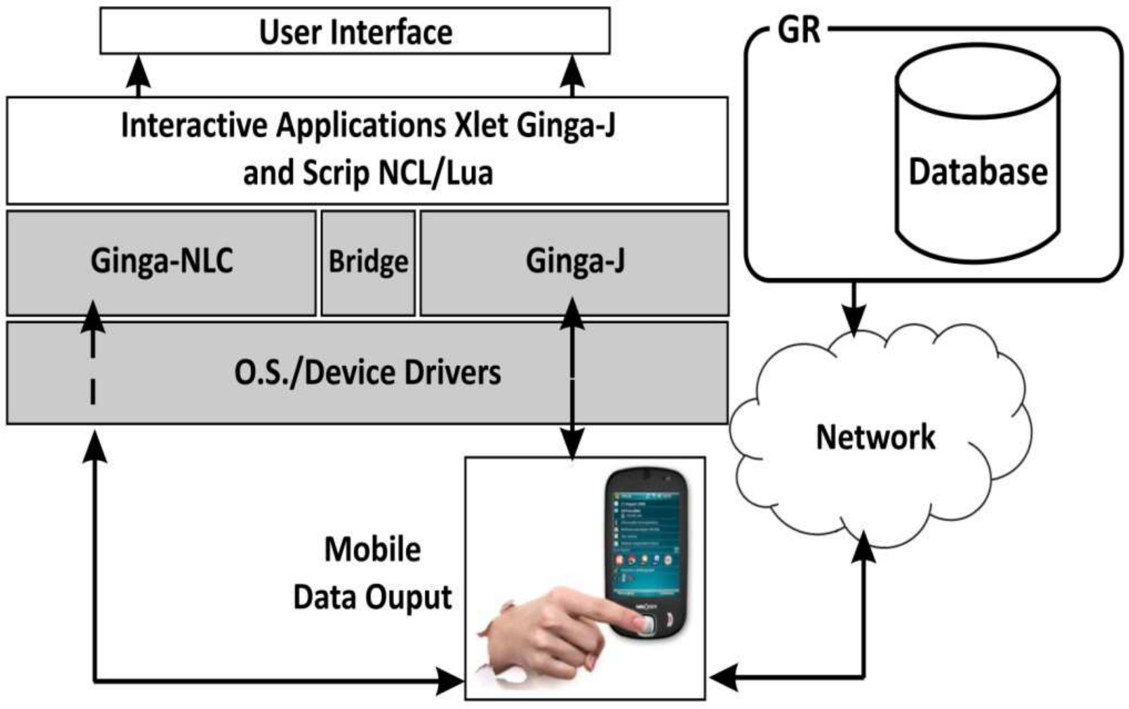


Fig. 7. Architecture for DTV module.

The alert module receives the data via the network through the reading of the GR database and then displays that data on the screen of the DTV. On submission of alerts, the GR first queries the database to see if it is a new alert, then sends it to the DTV module that displays it on the screen. The user, through the remote control, can access information on devices and decide to perform a procedure on the patient, depending on what is recommended by the system.

1.3 Building your health care software prototype

To assemble the test environment of the proposed model we incorporated technologies on the market for each of the components of the modules of Figures 5, 6 and 7. Among the solutions used were the XBee as a means of communication between the GR and the oximeter, GSM networks for the transmission of information relating to medical devices and XML to build a model for the standardization of information.

The hardware of the GR is designed on the x86 architecture, comprising a dual-core Atom processor, motherboard model A945GC with gigabit Ethernet interfaces, sound, video, 1GB of RAM and at least 8GB for storage, where we set up the System Linux operating system and related applications.

1.3.1 Installing and configuring the gateway

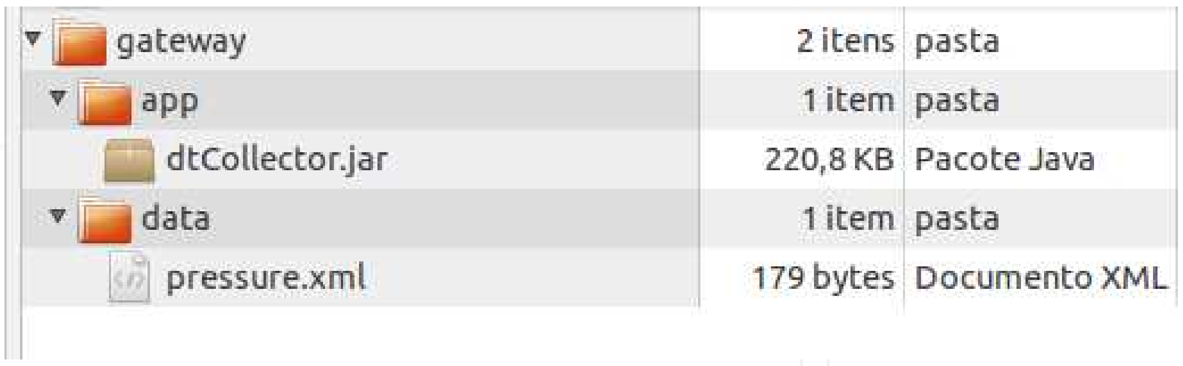
The operating system used at the time this chapter was written was the Linux Ubuntu 10.10, but new versions of Linux based on Debian will also work. To install Ubuntu 10.10 it is necessary to start the gateway computer via CD and run the installation from it. A tutorial that explains step by step installation of Ubuntu can be found at the link https: //help.ubuntu.com/10.10/installation-guide/. After installing the operating system, you must install the Java development tools, it is enough to open the terminal command as the Linux root user and run the command add-apt-repository "deb http://archive. canonical.com/ Lucidpartner " and then apt-get install sun-java6-jre sun-java6-jdk.

After the configuration of Java, you must install the RXTXLinux API that will allow access to data from the serial port device connected to the gateway. Simply download the API directly from wiki http://rtx.qbang.org/wiki. To Linux systems, you can insert the lib RXTcomm.jar in the folder /jre/lib/ext and insert the librxtxSerial.so too. Make sure the user is in group lock or uucp so lockfiles work.

1.3.2 Application configurations

Before starting the development it is ideal to set up a directory structure similar to that shown in Figure 8, to separate the data collected from sensors and devices, and other parts of the implementation.

The folder includes the gateway application app folder in which should be installed the data collector and folder date, responsible for storing data generated in the XML file. To generate the XML API you can use any Java-compatible, however in the example shown we used the API JDOM parser which was already integrated and easy to use.

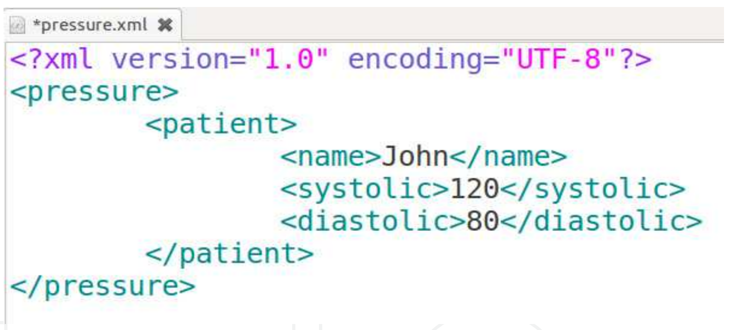


▼ gateway	2 itens	pasta
▼ app	1 item	pasta
dtCollector.jar	220,8 KB	Pacote Java
▼ data	1 item	pasta
pressure.xml	179 bytes	Documento XML

Fig. 8. Folder structure of the application.

1.3.3 Encoding an example using a pressure meter

We will now present an example coding of data collection of a blood pressure meter. For this example we used the measuring device shown in Figure 3 that can be easily found on the market. This device was connected to the gateway via a USB cable. Then the software collector runs, reads the USB port, collects data and stores them in an XML file. An example of the XML file stored is shown in Figure 9, where we can see the storage of data from the pressure meter already treated and ready to be presented in DTV.

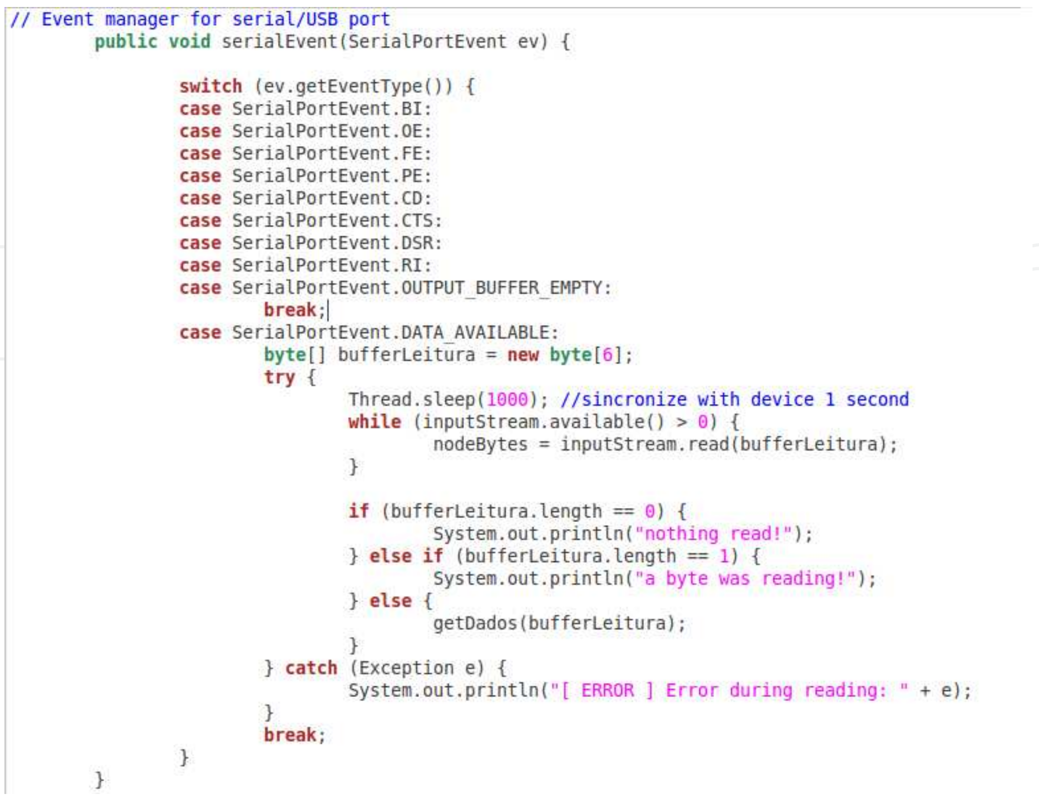


```
<?xml version="1.0" encoding="UTF-8"?>
<pressure>
        <patient>
                <name>John</name>
                <systolic>120</systolic>
                <diastolic>80</diastolic>
        </patient>
</pressure>
```

Fig. 9. XML data collecting by application.

To generate the XML file shown, it is necessary to codify the first reading from the USB port, so that a class written in Java was developed. This class has methods for reading the specific USB port, decoding of bytes received on the port, the transformation of information using substrings and finally creating the data file into XML using JDOM API. Parts of the source code showing the constructed methods are presented in Figure 10 and then its operation is explained.

The SerialEvent method shown in Figure 10 checks the bytes being passed into the USB port, synchronizes with the device using a second, and checks if the buffer has received some information. If you have an error while reading an exception is thrown and the method aborts execution. This method treats the events in serial or USB port connected on the device and uses the rxtx API for this.



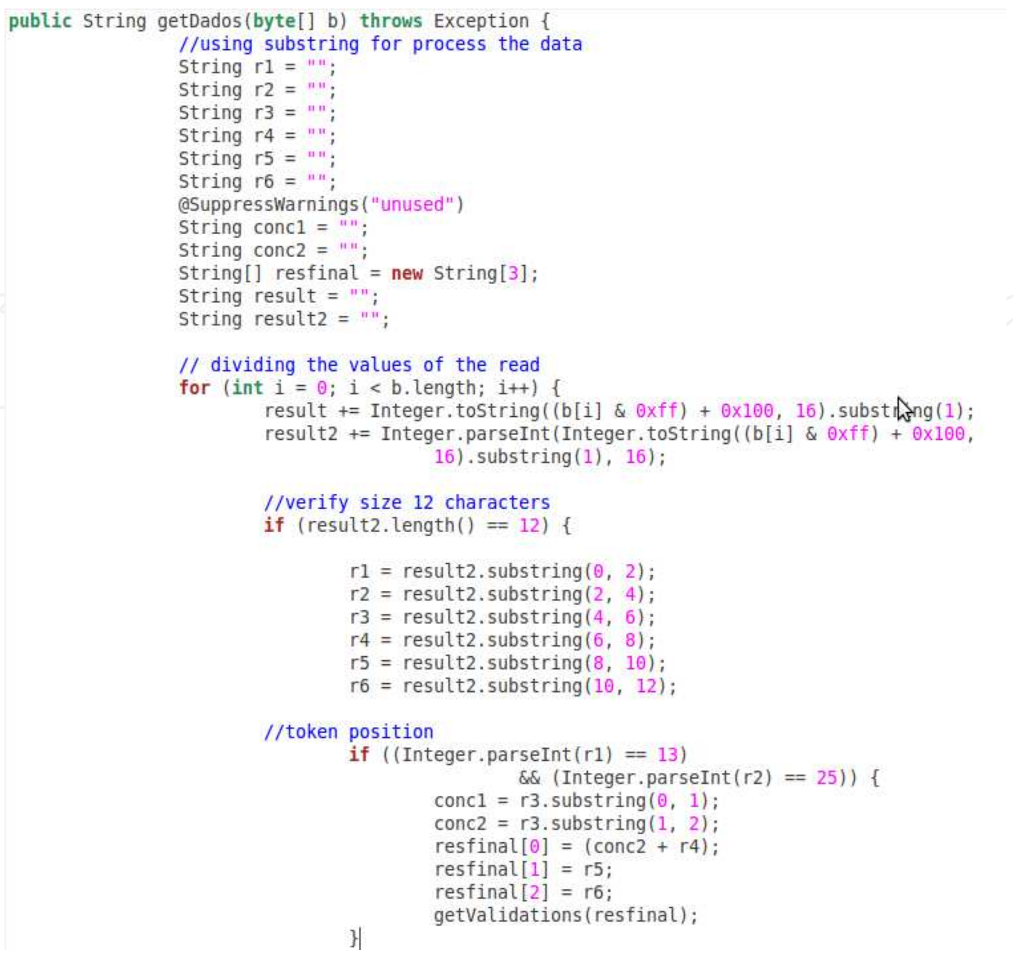
```
// Event manager for serial/USB port
        public void serialEvent(SerialPortEvent ev) {

                switch (ev.getEventType()) {
                case SerialPortEvent.BI:
                case SerialPortEvent.OE:
                case SerialPortEvent.FE:
                case SerialPortEvent.PE:
                case SerialPortEvent.CD:
                case SerialPortEvent.CTS:
                case SerialPortEvent.DSR:
                case SerialPortEvent.RI:
                case SerialPortEvent.OUTPUT_BUFFER_EMPTY:
                        break;
                case SerialPortEvent.DATA_AVAILABLE:
                        byte[] bufferLeitura = new byte[6];
                        try {
                                Thread.sleep(1000); //sincronize with device 1 second
                                while (inputStream.available() > 0) {
                                        nodeBytes = inputStream.read(bufferLeitura);
                                }

                                if (bufferLeitura.length == 0) {
                                        System.out.println("nothing read!");
                                } else if (bufferLeitura.length == 1) {
                                        System.out.println("a byte was reading!");
                                } else {
                                        getDados(bufferLeitura);
                                }
                        } catch (Exception e) {
                                System.out.println("[ ERROR ] Error during reading: " + e);
                        }
                        break;
                }
        }
```

Fig. 10. Event Manager for serial/USB ports.

Another part of the source of importance and worth noting is the method getDados. This method works with substrings to separate the information and extract only the necessary data. This is because the pressure measuring device sends a set of bits via serial port that include heart rate, systolic pressure, diastolic pressure and time of data collection. This example was not required to collect all information and thus we concentrated our efforts on separating only the systolic and diastolic blood pressure to generate the document data.

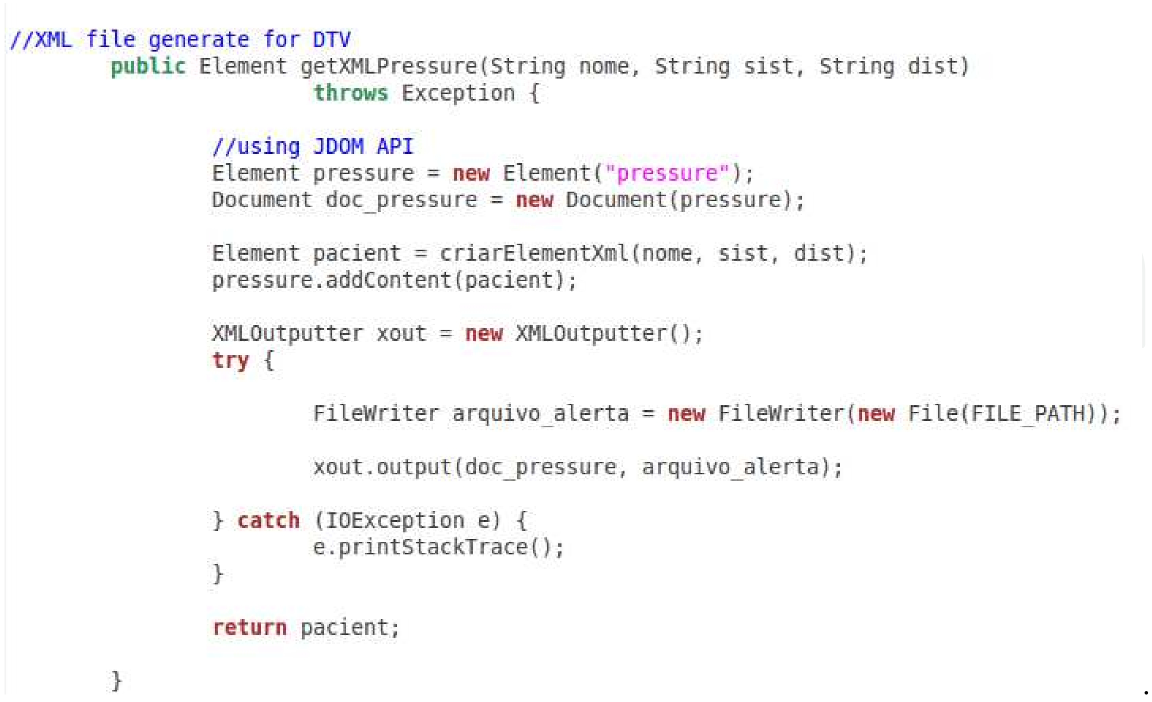
Figure 11 shows part of the method getDados and the logic as separate substrings. First, the method converts the information received in bytes, for strings, then counts the characters and stores them into substrings. Then it finds the position of the token and appends the information to finally send to the method that generates the data file. This process is repeated until all data is collected and treated. In the next step, the data file in XML format is created in the folder "data" of the gateway, which can be accessed via DTV using a wireless network connection. Part of the source code that generates the XML file is shown in Figure 12.

Looking at the source code in Figure 12, it is clear that data handled are passed as parameters to be inserted in the XML file using the method FileWriter. The JDOM API methods create the tags in accordance with the custom configuration for the client. For example, if the client wants to enter the patient's name, it is passed as a parameter to the application that generates the tag name.



```
public String getDados(byte[] b) throws Exception {
                //using substring for process the data
                String r1 = "";
                String r2 = "";
                String r3 = "";
                String r4 = "";
                String r5 = "";
                String r6 = "";
                @SuppressWarnings("unused")
                String conc1 = "";
                String conc2 = "";
                String[] resfinal = new String[3];
                String result = "";
                String result2 = "";

                // dividing the values of the read
                for (int i = 0; i < b.length; i++) {
                        result += Integer.toString((b[i] & 0xff) + 0x100, 16).substring(1);
                        result2 += Integer.parseInt(Integer.toString((b[i] & 0xff) + 0x100,
                                        16).substring(1), 16);

                        //verify size 12 characters
                        if (result2.length() == 12) {

                                r1 = result2.substring(0, 2);
                                r2 = result2.substring(2, 4);
                                r3 = result2.substring(4, 6);
                                r4 = result2.substring(6, 8);
                                r5 = result2.substring(8, 10);
                                r6 = result2.substring(10, 12);

                        //token position
                                if ((Integer.parseInt(r1) == 13)
                                                && (Integer.parseInt(r2) == 25)) {
                                        conc1 = r3.substring(0, 1);
                                        conc2 = r3.substring(1, 2);
                                        resfinal[0] = (conc2 + r4);
                                        resfinal[1] = r5;
                                        resfinal[2] = r6;
                                        getValidations(resfinal);
                                }
```

Fig. 11. Part of the source code of the method getDados.



```
//XML file generate for DTV
        public Element getXMLPressure(String nome, String sist, String dist)
                        throws Exception {

                //using JDOM API
                Element pressure = new Element("pressure");
                Document doc_pressure = new Document(pressure);

                Element pacient = criarElementXml(nome, sist, dist);
                pressure.addContent(pacient);

                XMLOutputter xout = new XMLOutputter();
                try {

                        FileWriter arquivo_alerta = new FileWriter(new File(FILE_PATH));

                        xout.output(doc_pressure, arquivo_alerta);

                } catch (IOException e) {
                        e.printStackTrace();
                }

                return pacient;

        }
```

Fig. 12. getXMLPressure method.

1.4 Results presented at the DTV

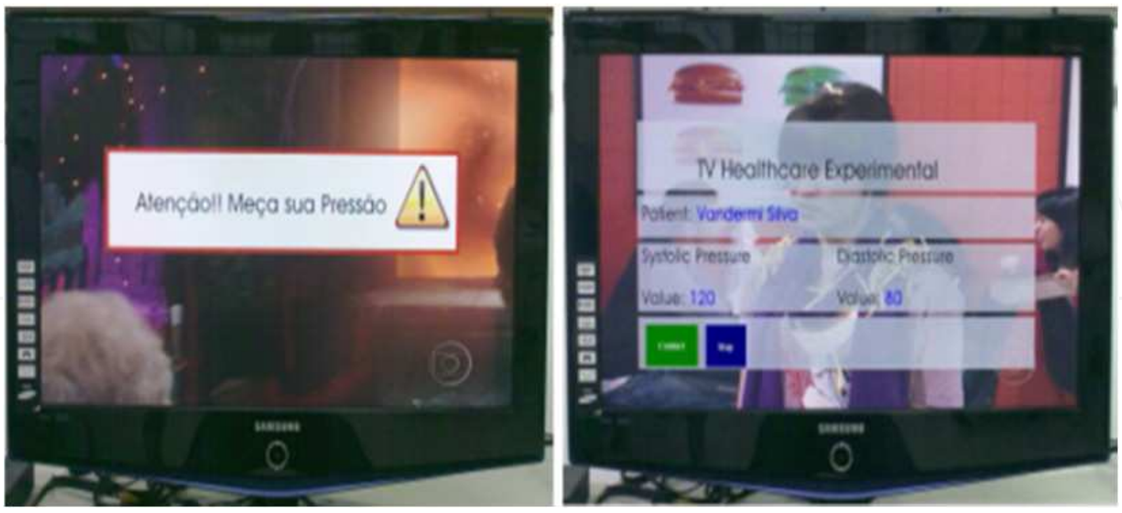
To display the monitoring data was used in a DTV STBs, with a Brazilian middleware Ginga-NCL that supports interactive applications that can run an external storage device or transmitted directly to the STB via broadcast by the broadcaster.

For the data presentation using the specification in DTV Ginga-NCL there were certain changes in the architecture, because the programming structure of NCL/Lua and JavaTV differs from the API. Currently in Brazil, most STBs compatible with the national standard have not yet run code written in Java. However, some companies have their own versions of middleware that enable execution of applications written in NCL and Lua. According to the programming model to the Ginga-NCL, an application to access the XML data from GR by extending the functionality of the application through the NCL script Lua has been developed.

In NCL are defined items responsible for the presentation of media (images, videos, text) and user interaction with the remote control. Through a link element control of the application for a Lua script can pass which performs a specific function and returns control to the NCL.

In this case, with regard to the health care application system, the Lua script is responsible for establishing a TCP / IP with GR, retrieving and processing the data in XML, and then formatting them for display on screen TVDi.

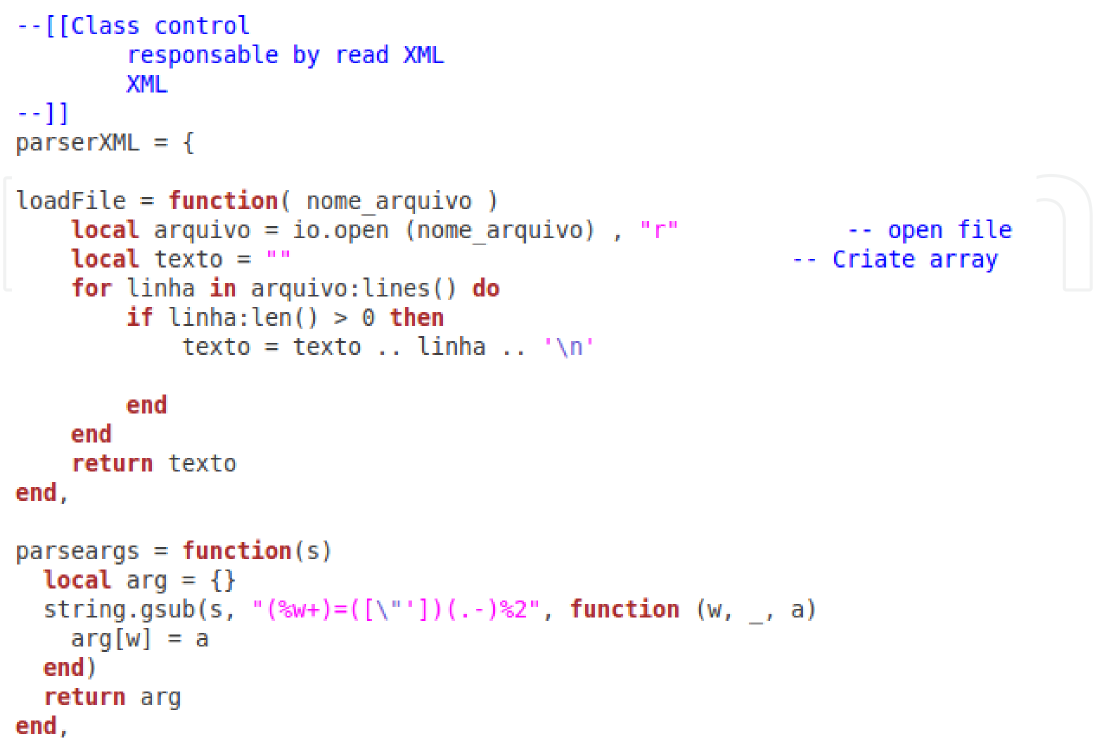
Figures 13 illustrates, respectively, a warning to the interface pressure measurement and presentation of data collected from GR, after you connect medical devices to the system. The interface was built in NCL / Moon in a real STB.


Atenção!! Meça sua Pressão

TV Healthcare Experimental
Patient: Vandermi Silva
Systolic Pressure
Value: 120
Diastolic Pressure
Value: 80

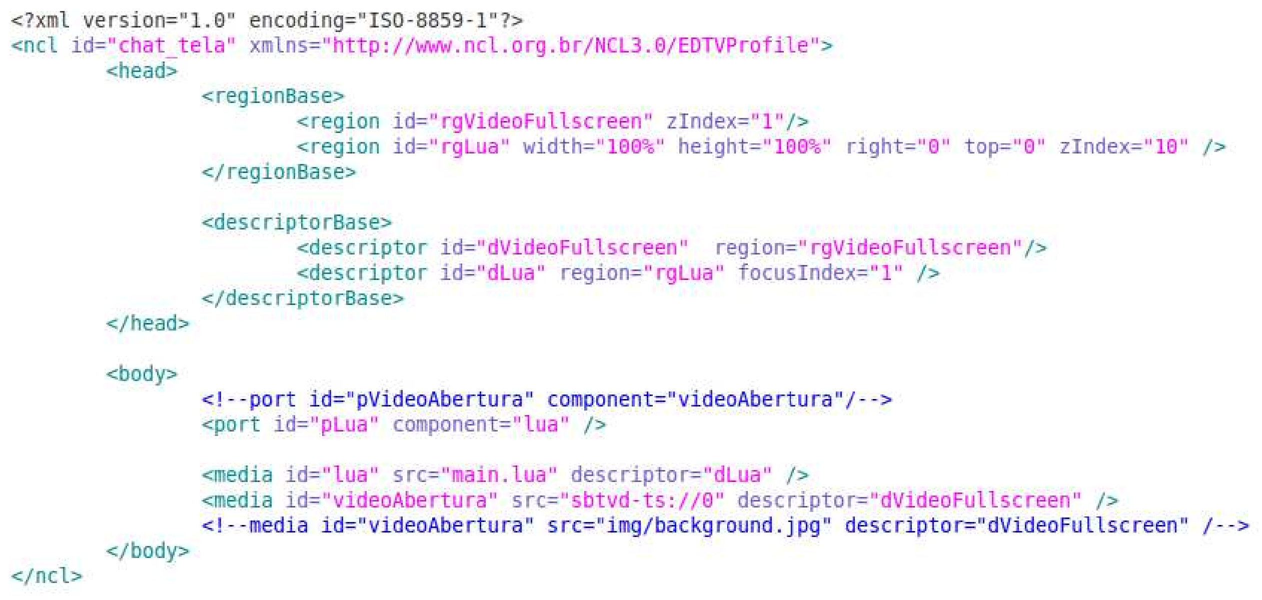
Fig. 13. Data presented at the DTV.

Part of the source code responsible for reading the XML and display of the DTV is shown in Figure 14.



```
--[[Class control
        responsable by read XML
        XML
--]]
parserXML = {

loadFile = function( nome_arquivo )
    local arquivo = io.open (nome_arquivo) , "r"            -- open file
    local texto = ""                                     -- Criate array
    for linha in arquivo:lines() do
        if linha:len() > 0 then
            texto = texto .. linha .. '\n'

        end
    end
    return texto
end,

parseargs = function(s)
  local arg = {}
  string.gsub(s, "(%w+)=([\"'])(.-)%2", function (w, _, a)
    arg[w] = a
  end)
  return arg
end,
```

Fig. 14. Source code in Lua.



```
<?xml version="1.0" encoding="ISO-8859-1"?>
<ncl id="chat_tela" xmlns="http://www.ncl.org.br/NCL3.0/EDTVProfile">
        <head>
                <regionBase>
                        <region id="rgVideoFullscreen" zIndex="1"/>
                        <region id="rgLua" width="100%" height="100%" right="0" top="0" zIndex="10" />
                </regionBase>

                <descriptorBase>
                        <descriptor id="dVideoFullscreen"  region="rgVideoFullscreen"/>
                        <descriptor id="dLua" region="rgLua" focusIndex="1" />
                </descriptorBase>
        </head>

        <body>
                <!--port id="pVideoAbertura" component="videoAbertura"/-->
                <port id="pLua" component="lua" />

                <media id="lua" src="main.lua" descriptor="dLua" />
                <media id="videoAbertura" src="sbtvd-ts://0" descriptor="dVideoFullscreen" />
                <!--media id="videoAbertura" src="img/background.jpg" descriptor="dVideoFullscreen" /-->
        </body>
</ncl>
```

Fig. 15. Source code in NCL.

The script uses a Lua function that captures the XML file stored on the server through the LAN and shares your information to further the application developed in NCL graphically presented in DTV. This is possible because the Ginga-NCL enables integration between Lua and NCL, which allows working static and dynamic content on DTV.

We hope this chapter serves as a first step to assist readers who want to better understand how technologies can be integrated to develop applications for day to day users. The work is hard, but certainly rewarding.

2. References

[1] IBGE. Instituto Brasileiro de Geografia e Estatística. "Pesquisa Nacional por Amostragem de Domicílios". http://www.ibge.gov.br. Acessado em 01/06/2011.

[2] CNDH. Coordenação Nacional de Hipertensão e Diabetes. http://dab.saude.gov.br/cnhd. Acessado em 01/06/2011

[3] SBTVD. Fórum Brasileiro de TV Digital. http://www.forumsbtvd.org.br/. Acessado em: 01/06/2011.

[4] S. Morris, A. Smith-Chaigneau. Interactive TV Standards A Guide to MHP, OCAP and Java TV. Burlington, MA, USA: Elsevier, 608p, 2005.

[5] ABNT. Associação Brasileira de Normas Técnicas. "NBR 15606-2:2007. Televisão Digital Terrestre – Codificação de dados e Especificações de Transmissão para Radiodifusão Digital – Parte 2: Ginga-NCL para Receptores Fixos e Móveis – Linguagem de aplicação XML para codificação de aplicações".

[6] L. F. G. Soares, M. F. Moreno, C. S. Neto. "Ginga-NCL: Declarative Middleware for Multimedia IPTV Services". *IEEE. Communications Magazine*, Vol. 48, No. 6, p. 74-81, 2010.

[7] R. Ierusalimschy Programming in Lua. Rio de Janeiro: Editora da Pontifícia Universidade Católica do Rio de Janeiro (PUC-RIO), 327p, 2006.

[8] M. F. Moreno, C. E. C. F. Batista, L. F. G. Soares. "NCL and ITU-T's Standardization Effort on Multimedia Application Frameworks for IPTV". *Brazilian Symposium on Multimedia and Web*, Vol. 1, No. 1, p. 1-6, 2010.

[9] M. Legally, J. Pätzold. "New TV Standard for Digital TV in Brazil". http://download.java.net/mobileembedded/developerdays/2009/TS-5-v2.pdf. Acessado em 01/06/2011.

[10] M. F. Moreno, L. F. G. Soares. "Resilient Hypermedia Presentations". *IEEE Workshop on Software Aging and Rejuvenation (WoSAR)*, Vol. 1, No. 1, p. 1-6, 2011.

[11] G. L. de Souza Filho, L. E. C. Leite, C. E. Batista. "Ginga-J: The procedural middleware for the Brazilian digital TV system". *Journal of the Brazilian Computer Society*, Vol. 13, No. 1, p. 47-56, 2007.

[12] ABNT. Associação Brasileira de Normas Técnicas. ABNT NBR 15606-4. Part 4: Ginga-J – The Environment for the Execution of Procedural Applications.

[13] L. F. G. Soares, R. M. Costa, M. F. Moreno. "Multiple exhibition Devices in DTV systems". *MM'09 Proceedings of the 17th ACM International Conference on Multimedia*, Vol. 1, No. 1, p. 281-290, 2009.

[14] O. B. Maia, N. S. Viana, V. F. de Lucena Jr. "Using the iDTV as a Center of a Ubiquitous Computing Environment". *InTech Open Publisher Access*, Vol. 1, No. 1, p. 225–248, 2011.

[15] N. S. Viana; V. F. de Lucena Jr., "iDTV Home Gateway Convergence: an Open Software Model Integrating the Ginga Middleware and the OSGi Framework". *Multimedia Systems*, Vol.16, No.5, p. 1-15, 2010.

[16] S. Dixit, R. Prasad. "Technologies for Home Networking". New York: John Wiley & Sons, 218p, 2008.

[17] V. F. de Lucena Jr., J. E. C. Filho, N. S. Viana, O. B. Maia. "A Home Automation Proposal Built on the Ginga Digital TV Middleware and the OSGi Framework". *IEEE Transactions on Consumer Electronics*, Vol. 55, No. 3, p. 1254-1262, 2009.

[18] W. –W Lin, Yu-Hsiang-Sheng. "Using OSGi, UPnP and Zigbee to Provide a Wireless Ubiquitous Home Healthcare Environment". *IEEE Computer Society*, Vol. 1, No. 1, p. 268–272, 2008.

[19] R. C. Augusto, J. Carlos, S. Daniel. "Ambient Intelligence — The Next Step for Artificial Intelligence". *IEEE Intelligent Systems*, Vol. 23, No. 2, p. 15–18, 2008. ISSN 1541-1672.

[20] J. Corchado, J. Bajo, J. Abraham. "Healthcare Delivery IN Geriatric Residences". *IEEE Intelligent Systems*, Vol. 23, No. 2, p. 19-25, 2008.

[21] M. Valero, L. Vadillo. "An Implementation Framework for Smart Home Telecare Services". *Future Generation Communication and Networking (FCGN),* Vol. 2, No. 1, p. 60–65, 2007.

[22] PROTÉGÉ. Overview. http://protege.stanford.edu/overview/.
Acessado em 01/06/2011.

[23] D. S. G. Pekhteryev, H. Z. Sahinoglu, C. Challa. "Cam N. Real-Time and Secure Wireless Health Monitoring". Int. J. Telemedicine Appl. Computer, Vol. 1, No. 1, p. 1–10, 2008.

[24] V. Becker, L. F. Soares. "Viva Mais Alimentação Saudável".
http://clube.ncl.org.br/node/29. Acessado em 01/06/2011.

[25] V. Becker, L. F. Soares. "Viva Mais Peso Ideal".
http://clube.ncl.org.br/node/15. Acessado em 01/06/2011.

[26] Portal do Software Público Brasileiro. Governo Federal.
http://www.softwarepublico.gov.br/. Acessado em 01/06/2011.

[27] A. Mendes. Arquitetura de Software: Desenvolvimento Orientado para a Arquitetura. Rio de Janeiro, RJ, Brasil: Campus, 212p, 2002.

[28] F. de C. Branco, J. M. Viana, J. R. P. Lima. "Freqüência Cardíaca na Prescrição de Treinamento de Corredores de Fundo". *R. Bras. Ci. e Mov.*, Vol. 12, No. 2, p. 75-79, 2004.

[29] Chobanian, A. V. et al. Seventh report of the joint national committee on prevention, detection, evaluation, and treatment of high blood pressure. the American Heart Association, v. 1, n. 1, p. 1221–1222, December 2003.