Prototyping of a Remote Monitoring System for a medical Personal Area Network using Python

M. J. Morón, J. R. Luque, A. Gómez-Jaime, E. Casilari and A. Díaz-Estrella

Dpto. Tecnología Electrónica, University of Málaga, Spain

Abstract— **This paper presents a prototype developed in Python of a pervasive mobile health system aimed at monitoring a patient in indoor and outdoor environments continuously. The system is based on a Bluetooth PAN (Personal Area Network), worn by the patient, whose master node, a smartphone, collects information about patient's location and health status and detects emergency situations. These data are sent to a central server through Wi-Fi or GPRS/UMTS, which allows physicians to get access to patient data and configure the PAN sensors remotely using a conventional web browser.**

Keywords— **Bluetooth, Smartphone, pulse-oximeter, PAN, python, m-health.**

I. Introduction

The concept of m-health (or Mobile-Health) combines mobile computation technologies, wireless communications and smart sensors to provide remote healthcare services. The purpose of m-health applications is to improve the quality of the medical assistance and patient comfort while minimizing the cost of the services by maximizing the efficiency of the resource management [1].

Previously to m-health emergence, the expansion of Internet led to the appearance of e-health systems, intended for applying Internet an Information and Communications Technologies to healthcare services. Thus, the first prototypes of e-health system were based on personal computers and fixed communication networks, such as ISDN (Integrated Services Digital Network) y DSL (Digital Subscriber Line) [2].

Nowadays, the low cost, variety and popularity of handheld mobile devices (such as PDA or Personal Digital Assistants, cell phones, blackberries, etc) as well as the development software platforms specifically created for these devices (Palm, Pocket PC/Windows CE, Symbian, Embedded Linux, etc) have contributed to evolve from the general paradigm of e-health to m-health. Additionally, the fact that wearable smart biosensors integrate short-range wireless communication technologies, such as Bluetooth or ZigBee, has eased the development of telemonitoring prototypes for medical W-BAN (Wireless Body Area Network) or W-PAN (Wireless Personal Area Networks). In most cases, the PAN/BAN is coordinated by a node which in turn may retransmit the signals to a remote central monitoring unit [3]. Thus, the general architecture of an m-health PAN/BAN considers three components [4]:

- A network of attachable or wearable biosensors (which can be completed with movement and positioning sensors), equipped with a low power, short range wireless interfaces, mainly based on Bluetooth or ZigBee technologies.
- A network (PAN/BAN) coordinator which is responsible for centralising and managing the communications with the biosensors. This coordinator acts as an Internet gateway to other access networks (GSM/GPRS/UMTS or WLAN) in order to transmit sensors biosignals (or medical alarms) to a remote monitoring point and to receive control information.
- A central node (or a distributed central system) in charge of storing the sensors signals, detecting possible alarms and distributing the patients' information (e.g: via Internet, SMS, e-mail, etc) to the medical staff. The access to the information can be in turn accessed from wired or wireless terminals (for example, in [2] a Pocket PC/Palm is used to permit the medical staff accessing remotely to the Hospital information system).

There are many works [5-19] in the literature that follow (to same extent) this architecture to deploy applications of medical telemonitoring. From these experiences, we can remark the growing use of 'general purpose' mobile devices (mainly PDAs or Smartphones) to build the PAN node which acts as the gateway between the wireless sensors and the final remote control node. This can be justified by the universalised use of cell phones in developed countries. Thus, the implementation of the medical BAN central node can be performed without introducing any new wearable hardware and just with a re-configuration of a familiar and quotidian element in the everyday life of the patients (the mobile phone).

As it refers to the programming language that is normally chosen to build the software in the phones (or PDAs), initial architectures utilised C or C++, which permitted an optimised design for the real time processing of biosignals and a

Digital Object Identifier: 10.4108/ICST.PERVASIVEHEALTH2009.5854
http://dx.doi.org/10.4108/ICST.PERVASIVEHEALTH2009.5854

better interaction with the underlying operative system (e.g.: Symbian in the case of most phones). However, the emergence of scripting support in the S60 developer platform for Symbian OS through Python, enables the adoption of this designing tool in the field of pervasive health systems, because of its benefits for the rapid prototyping of applications.

This work presents an architecture that defines a smartphone-based Personal Area Network (PAN) of Bluetooth biosensors and a central node whose main task is to distribute the captured biosignals to the medical staff. The goal of this architecture is to evaluate both the performance of smartphones when employed as gateway/master in a PAN of Bluetooth sensors and to propose a prototype of a web-based central server for the remote access of sensors data. The evaluation specially will take into account the limitations of the Python language as an alternative developing tool for the implementation of the local application running on the smartphone and the server-side web application.

Python is an object-oriented scripting language. According with [20] the main benefit of this type of programming philosophy is that the development time of applications is radically decreased. Furthermore, scripting languages are easier to learn, which allows that casual programmers, as the hospital technicians or even the same medical staff, can participate in the development and adaption of the healthcare application to their particular needs.

The rest of the work is structured as follows: Section 2 briefly describes the general structure of the proposed ongoing prototype. Section 3 comments in more detail the implementation. Finally section 4 presents some preliminary conclusions and project's current status.

II. System Architecture

The goal of the proposal is to carry out a smart tracking of patients requiring a continuous monitoring. With this purpose, the system defines an architecture with the following components:

1. A PAN carried by the patient to be remotely monitored.
2. A Central Control Server (CCS), which centralises the data of patients.
3. A Mobile Control and Monitoring Unit (MCMU) carried by the physicians.

The PAN prototype is based on Bluetooth technology. Currently the PAN includes two commercial medical sensors (a pulse-oximeter and an ECG sensor), a GPS device and a smartphone performing as the network coordinator or *Intelligent Node* (IN). The smartphone incorporates two wireless communications interfaces, 802.11 and Bluetooth, as well as a GPRS/UMTS connection to cellular network. Bluetooth is the technology employed for the communications between the sensors and the IN, which plays the master role in the Bluetooth piconet. Wi-Fi and/or GPRS/UMTS are employed to send information from the IN to the CCS while the MCMUs access the patient information by an authenticated connection to the CCS via Internet.

The next devices have been considered in the PAN prototype: (1) Smartphones N95 and E61 as hardware and software platforms for the intelligent node (IN). The software developer platform are 3rd Ed FP 1 for the N95 and 3rd Ed (initial release) for E61, with operating System Symbian v9.2 and v9.1, respectively; (2) a Bluetooth v1.1 pulse-oximeter Nonin 4100, from Nonin Medical Inc. [21] and a CorBelt Blueetooth ECG sensor from CorScience [22]; and (3) a Bluetooth v 1.2 GPS receiver with chipset SirfStarIII. All the Bluetooth devices employ Serial Port Profile (SPP)

A. Intelligent Node (IN)

A Python application has been developed for the deployment of the IN software in the Nokia S60 smartphones (N95 and E61 models). In particular Python for S60 3rd Edition v1.4.1 based on Python 2.2.2 has been utilised. Different Python APIs (Application Program Interfaces) have been employed, including: S60 extension socket.py for managing BT connections; built-in extension e32, for the special Symbian OS services based on active objects; built-in extension appuifw, an UI application framework; or Python standard API threading.py for concurrent programming with threads.

B. Central Control Server (CCS)

As it is shown in Figure 1, the CCS is integrated by three components:

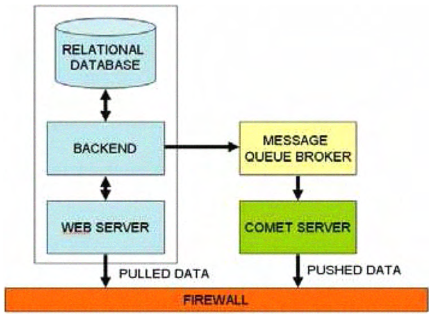


Fig.1. Structure of the Central Control Server

Digital Object Identifier: 10.4108/ICST.PERVASIVEHEALTH2009.5854
http://dx.doi.org/10.4108/ICST.PERVASIVEHEALTH2009.5854

1. **A traditional web server** to implement a web application.

The web application or backend of the web server has been structured according to the design pattern MVC (Model-View-Controller). Besides, the next Python web frameworks have been used: (a) SQLObject, an Object Relational Mapper (ORM) to access relational databases as MySQL conforming to object-oriented model; (b) Cheetah, a template engine to generate dynamically web documents; (c) CherryPy, an object-oriented HTTP framework to connect the web service to the Python controller code. Specifically, at the present time, the web server employed is that provided by CherryPy. Nevertheless, Apache web server could be integrated.

2. **A Comet server** to asynchronously communicate the events generated from the server to the clients.

Comet refers to a new paradigm to develop web applications, which utilizes persistent HTTP connections in order to allow a web server to send data to a web browser without explicit data request (in a asynchronous way). In the system described here, Orbited server has been used as Comet server. Orbited, based on Twisted, is a Python event-driven networking framework, currently supported by the most popular web browsers, such as Mirosoft IE, Firefox, Safari y Google Chrome.

3. **A Message Queue Broker (MQB)** to integrate components (1) and (2).

In the system, Apache Active MQ has been used as MQB. The native implementation of Apache Active MQ employs Java Message Service API (JMS) and provides connectivity with STOMP (Streaming Text Orientated Messaging Protocol) clients. In fact, an interesting feature of Orbited is that it offers a Javascript implementation of a STOMP client which allows the connection to any Stomp MQB. Specifically, with the Active MQ supplies, MCMUs are enabled to subscribe to the different channels or queues where data of monitored patients, which the web server receives from the PANs and resend to the MQB, are temporally saved. In this sense, the implementation permits that several MCMUs can simultaneously subscribe to a channel, so that the Active MQ will resend the data to all of them (broadcast o multicast).

III. Implementation

A. Exchange Data Format

JSON (JavaScript Object Notation, RFC 4627) has been used to encode the patient data exchanged by HTTP 1.1 (RFC 2616), between the architecture components (IN, CCS and MCMU). The main advantage of JSON is its simplicity. JSON encoded data are plainly discernible by the humans. Besides, most programming languages include libraries to encode and decode this format.

B. Interaction IN-CCS

The IN of the PAN sends the biosignals to the CCS periodically, by means of HTTP POST requests, which encapsulate the sensor and GPS data. The transmission period can be configured by physicians through the MCMUs. For the CCS, request pipelining is used instead of HTTP POST. This method consists in sending multiple requests through a TCP connection without waiting for the response before continuing with the next transmission. In any case, in parallel with the data transmissions, HTTP GET requests are sent periodically (HTTP polling) in order to allow that the CCS can encapsulate a command received from any MCMU in the HTTP response.

C. Interaction CCS-MCMU

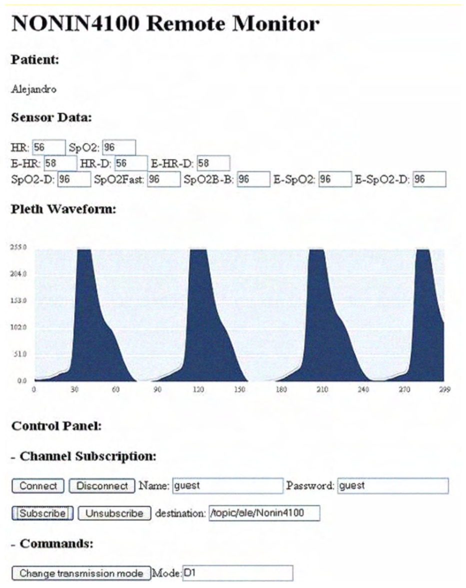
As it refers to the communication between the CCS and the MCMUs HTTP streaming and AJAX requests have been used for data packets and control commands transmission, respectively. In both cases, XML HTTP Requests have been employed because they allow to issue asynchronous HTTP requests without blocking the web browser. However, to enable continuous data reception a persistent connection is kept, whereas a new connection is established for every command transmission, which is closed after the response from the CCS is received.
Besides of these connections, once the patient and sensor have been selected, a MCMU has to submit some HTTP requests to download a HTML document and several Javascript libraries: a) json2.js to decode and encode JSON text; (b) MochiKit to easy processing of the HTML document (DOM API); (c) Stomp.js, that implements a STOMP client to connect and subscribe to the corresponding channel of the MQB, through the Comet server.

D. Web Application on MCMUs

When the physicians wish to monitor a patient, they open the start page in the web browser to select the patient and

Digital Object Identifier: 10.4108/ICST.PERVASIVEHEALTH2009.5854
http://dx.doi.org/10.4108/ICST.PERVASIVEHEALTH2009.5854

the sensor whose data are desired to be displayed. Then a dynamic web page is downloaded in the browser which is organised in three main parts as shown the figure 2: in the uppermost part or header, the patient name who is being monitored is shown; in the middle or body, the sensor data is displayed; and finally, in the footer we can find the control panel, in which the physician can connect and subscribe to the selected sensor channel in order to start the remote monitoring, and to execute several sensor commands to set several parameters such as the transmission mode, the thresholds values used to detect an emergency or even the transmission period.


NONIN4100 Remote Monitor

Patient:

Alejandro

Sensor Data:

HR: 56 SpO2: 96
E-HR: 58 HR-D: 56 E-HR-D: 58
SpO2-D: 96 SpO2Fast: 96 SpO2B-B: 96 E-SpO2: 96 E-SpO2-D: 96

Pleth Waveform:

Control Panel:

- Channel Subscription:

Name: guest Password: guest

destination: /topic/ale/Nonin4100

- Commands:

Mode: D1

Fig. 2: Web application for remote monitoring of pulse-oximeter data

Technological evaluation

Before the adoption of Python, other versions written in J2ME and Symbian C++ were developed for the implementation of the software running in the IN. The functionality achieved using Python has been very similar but the development time (and the number of lines of code) were considerably reduced. Relative to the capability of getting near real-time transmission of the data received from medical sensors, one remarkable design decision has been the evolution from a traditional synchronous request/response mode of HTTP to pipelining request mode using persistent TCP connections. Pipelining enables a higher output transfer rate and saves the computational resources required by the web server to reopen the TCP connections. On the other hand, at the MCMU side, the overall system has been evaluated considering different popular web browsers. FireFox 3, IE7, Safari 3, Chrome 0.2 have shown to be fully compatible while this has not been the case of Opera 7, Opera 8, (Symbian) and OSS Browser (Symbian).

VI. Conclusions

This paper is focused on the adoption of Python for the implementation of a system for remote monitoring of biosignals because of the benefits in productivity coming from scripting programming languages. The proposed architecture is based on W-BAN, whose master/gateway node is a smartphone with Wi-Fi and GPRS/UMTS capabilities. The current prototype has an implementation of the application located in a smartphone using the S60 developer platform and Python support. Furthermore, the proposed architecture also includes a central web server connecting different Python web frameworks according to the MCV pattern. Finally, the sending of patient's data to the medical staff has been solved applying a new model of web programming called Comet and a Message Queue Broker for the integration between a traditional Web server and a new Comet server. This solution allows quasi-real time transmission of sensors data to the physicians and the transmission of data over the HTTP protocol, so it is easy to traverse firewalls with a restrictive security policy.

Acknowledgments

This work has been supported by project TEC2006-12211-C02-01/TCM.

References

1. Istepanian R S, Jovanov E, Zhang Y T (2004) Introduction to the Special Section on M-Health: Beyond Seamless Mobility and Global Wireless Health-Care Connectivity, *IEEE Trans. on Inf. Tech. in Biomedicine*, vol.8, pp405–414, Dec. 2004.
2. Güler N F and Übeyli E D (2002), Theory and applications of telemedicine. *J. Med. Syst.* 26(3), pp. 199-220.
3. Hung K, Zhang, Y, Tai B (2004) Wearable medical devices for telehome healthcare, in *Proc. of Conf. EMBC 2004*, vol.7, pp.5384–5387.
4. Altieri R, Incardona F, Kirkilis H and Ricci R. (2006) Mobi-dev: Mobile devices for healthcare applications. *M-Health* pp. 163-175.
5. Jovanov E, Milenkovic A, Otto C et al. (2005) A wireless body area network of intelligent motion sensors for computer assisted physical rehabilitation, *Jour. of Neuroengineering Rehabilitation*, vol.2, no.6.
6. Krco S (2003) Implementation solutions and issues in building a personal sensor network for health care monitoring, in *4th International IEEE EMBS Special Topic Conference on Inf. Tech. Applications in Biomedicine, 2003.*, pp 350–353.

7. Malan A D, Fulford-Jones T, Welsh M (2004) CodeBlue: An Ad Hoc Sensor Network Infraestructure for Emergency Medical Care, in *Proc. of MobiSys WAMES 2004, Boston, MA*, pp 12–14.
8. Warren S, Yao J, Schmitz R et al. (2003) Wearable telemonitoring systems designed with interoperability in mind, in *Proc. of IEEE Eng. in Medicine and Biology Society, 2003*, vol. 4, pp 3736– 3739.
9. Hung K, Zhang Y (2002) Usage of Bluetooth in wireless sensors for tele-healthcare, in *EMBS/BMES Conference, 2002. Proc. of the Second Joint*, vol.3, pp 1881–1882.
10. Dong J, Zhu H (2004) Mobile ECG detector through GPRS/Internet, in *Proc.of CBMS 2004.*, pp. 485– 489.
11. Khoor S, Nieberl K, Fugedi K et al. (2001) Telemedicine ECG-telemetry with Bluetooth technology, *Computers in Cardiology 2001*, pp. 585–588.
12. Liszka K, Mackin M, Lichter (2004) Keeping a beat on the heart, in *Pervasive Computing, IEEE*, vol. 3, pp.42–49.
13. Lee R G, Hsiao C C, Chen C C et al. (2006) A mobile-care system integrated with bluetooth blood pressure and pulse monitor, and cellular phone, *IEICE Trans. on Information and Systems*, vol. E89-D, pp.1702–1711.
14. Krco S, Kostic S, Sakac D et al. (2005) mSens mobile health monitoring system, Proc of. EUROCON 2005, vol. 1, pp 80–83.
15. Jones V, Halteren A V, Widya I, Dokovsky N, Koprinkov G, Bults R, Konstantas D and Herzog R. (2006), Mobihealth: Mobile health services based on body area networks. *M-Health* pp. 219-236.
16. Krco S, Delic V (2003) Personal wireless sensor network for mobile health care monitoring, in *Proc. of TELSIKS 2003*, vol. 2, pp 471–474.
17. Wang D, Lu Y, Zhang H et al. (2005) A wireless sensor network based on Bluetooth for telemedicine monitoring system, in *Microwave, Antenna, Propagation and EMC Technologies for Wireless Communications*, vol. 2, pp 1361– 1364.
18. Warren S, Lebak J, Yao J et al. (2005) Interoperability and Security in Wireless Body Area Network Infrastructures, in Proc. of *IEEE-EMBS 2005. 27th Annual International Conference of the*, pp 3837–3840.
19. Park D, Kang S (2004) Development of reusable and expandable communication for wearable medical sensor network, in *Proc. of EMBC 2004*, vol. 7, pp 5380– 5383.
20. Ousterhout J K (1998), Scripting: Higher-level programming for the 21st century. *Computer 31(3),* pp. 23-30.
21. Nonin Medical Inc. at http://www.nonin.com
22. Corscience. at http://www.corscience.de/de/corscience/home.html