Hindawi Publishing Corporation
EURASIP Journal on Wireless Communications and Networking
Volume 2011, Article ID 629526, 10 pages
doi:10.1155/2011/629526

Research Article

Development and Evaluation of a Python Telecare System Based on a Bluetooth Body Area Network

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

Departamento de Tecnología Electrónica, ETS de Ingeniería de Telecomunicación, Universidad de Málaga, Spain

Correspondence should be addressed to M. J. Morón, mjmoron@uma.es

Received 31 October 2010; Revised 28 December 2010; Accepted 31 December 2010

Academic Editor: Arie Reichman

This paper presents a prototype of a telemonitoring system, based on a BAN (Body Area Network) that is integrated by a Bluetooth (BT) pulse oximeter, a GPS (Global Positioning System) unit, and a smartphone. The smartphone is the hardware platform for running a Python software that manages the Bluetooth piconet formed by the sensors. Thus the smartphone forwards the data received from the Bluetooth devices, encoded into JSON (JavaScript Object Notation), to a central server. This server provides universal access to the information of the patient's location and health status through a web application based on AJAX (Asynchronous JavaScript and XML) technology. Additionally, for the described prototype, the study presents some performance analyses about several topics that are of great interest for the applicability of the prototype: (i) the technique used to forward the patient's location and health status, (ii) the power consumption of the smartphone (which is compared with the measurements of an equivalent software developed for Java Micro Edition platform), and (iii) the web browser compatibility of the web application developed for the control and monitoring of the patients.

1. Introduction

In the context of e-Health (i.e., the application of information and communication technologies to the health area), remote monitoring is one of the most representative applications and one of the e-Health services which implies more technologic and logistical challenges.

The term "chronic diseases" (applied to disorders such as diabetes, asthma, cardiovascular diseases, cancer, or depression) is employed to refer to health problems that, while not being transmissible diseases, persist over time and require some degree of care. Epidemiological data from 2000 indicate that, globally, nontransmissible diseases and mental disorders have entailed a mortality rate of 59% and 46% of the total morbidity [1]. Additionally, there are predictions that, for 2020, both types of conditions will lead to a 78% of the global morbidity in developed countries [1]. Moreover, it cannot be neglected that these diseases impact not only on the health of the population but also on the economic resources of the citizens and states. For example, in the United States where health cost per capita is higher than the average of other developed countries [2], the total expenditure on health due to chronic diseases increased from 78% in 2002 up to 84% in 2009 [3, 4].

The efficient management of chronic diseases represents a challenge due to the significant impact on the population health (quantified in terms of morbidity and mortality rates) and on health expenditure. In this scope, both European countries and the United States have deployed new and more efficient health care models, focused on the prevention of chronic diseases and their consequences. Although there are different methods, specifically conceived to manage every kind of disease, all of them require the active involvement of the patient for the control of the symptoms and the corresponding therapy as well as for the tracking of the evolution accomplished by the physicians (normally during a lengthy period). This vision of the health care implies a close communication with the patient and, therefore, a greater number of home visits, which obviously increase the cost of the sanitary system. However, the application of information and communication technologies, and specially home telecare or home telemonitoring, allows extending the health care outside the hospital by virtual medical visits, combining the tracking of the patients with a cost reduction.

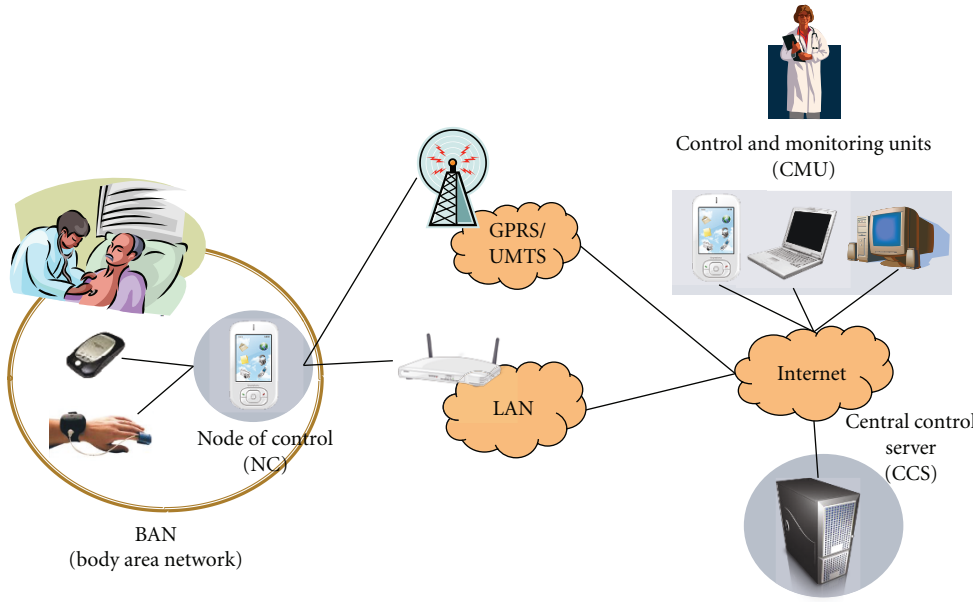

Figure 1: Architecture of monitoring system.

Besides, the information retrieved from the continuous monitoring of patients during long periods represents a key tool for the advances in the diagnosing of a disease, for the description of its evolution and for the forecast of possible complications, including the early prevention of the occurrence of severe events that may require the hospitalization [5]. At last, the implantation of this new model of care would bring about the following benefits: an improved quality of life of the patients, a decrease in the rate of hospitalizations, fewer outpatient visits, and an increased patient satisfaction [6].

Up to now, the advantages of telemonitoring for chronic diseases care have been exposed. Nevertheless, the usefulness of telemonitoring is more evident when mobile communications and wireless technologies are combined. Traditionally, the medical telemonitoring systems have consisted of home telecare units that send biosignals or medical alarms to the medical premises from a PC connected to the fixed telephony network [7]. However, the progress in the field of smart medical sensors together with the expansion of wireless and mobile communications, has enabled the creation of new monitoring systems which allows increasing the mobility and the comfort of the patients.

These systems are normally based on the usage of wireless sensors and a smartphone or PDA (Personal Digital Assistant), which is in charge of forwarding the data received by the sensors, to the monitoring point [8–10]. The application of these technologies to telemonitoring is known as m-Health (mobile health) or pervasive-Health [11, 12], in a process in constant evolution that leads to the concept of u-Health or ubiquitous-Health [13]. The ubiquitous telemonitoring involves a significant improvement in the management of chronic conditions, allowing continuous and real-time monitoring during the normal patient activity. The aim under the u-Health concept is to provide the telemonitoring service from any place and in any time, without limiting it to the home environment. Certainly, as noted in [14], the chronic care must be provided in these conditions.

The scientific literature has paid a lot of attention to m-Health systems. The research jobs presented in [10, 15–20], focus on the telemonitoring of patients with chronic diseases, by employing wireless technologies. From these and other existing works we can point out that remote monitoring systems represent one of the most promising technological research areas in the health context, especially because its application to the management of chronic diseases may have a significant economic impact. However, the results shown in several systematic reviews of published articles (evidence-based studies) conclude that, although the telemonitoring of patients represents a promising solution for the management of chronic diseases, more trials are needed to assess its economic viability and its applicability in real scenarios [21–23]. In fact, besides the need for empirical studies to appraise the cost-effectiveness (as in the case of the work presented in [22]), the testing with actual patients (see the study in [24]) and the assessment of the functionalities of the involved technologies [25] should not be disregarded. This paper focuses on this assessment. Specifically, the paper presents and evaluates a prototype of a monitoring system based on a Body Area Network (BAN) worn by the patient. This BAN integrates Bluetooth sensors, managed by a Python [26] application that runs on a smartphone. The patient is monitored through a web application, based on AJAX (Asynchronous JavaScript and XML) [27] technology.

The paper is structured as follows. Section 2 summarizes the objectives of the proposed architecture. The explanation of the developed prototype is detailed on Section 3. Section 4 includes the tests which have been executed to

evaluate the power consumption, under different conditions of monitoring, and the web browser compatibility of the web application intended for remote monitoring. Finally, Section 5 recapitulates the conclusions and presents the current research lines of the on-going work.

2. Objectives

One of the architectures usually employed for monitoring systems is based on wireless short range networks: BANs (Body Area Networks) or PANs (Personal Area Networks). Bluetooth [28] and 802.15.4/ZigBee [29] are the most widely used standards for the deployment of these types of networks. In the typical topology defined for short-range wireless networks, a central element (coordinator, master etc.) is normally in charge of coordinating the network and forwarding the information sent by the sensors to the remote monitoring point. Additionally, in the monitoring center, the data received from the BAN or PAN networks are stored and processed in order to detect and notify risk events (e.g., medical alarms) [30]. Some examples of research jobs and even commercial products, which employ telemonitoring applications based on short-range wireless networks, are presented in [14, 31–34].

In this paper, we describe and analyze a prototype of a biomedical monitoring system with an architecture based on a Bluetooth BAN. The system is deployed without requiring the development of any specific hardware, just combining commercial Bluetooth vital parameter sensors and a conventional smartphone.

In contrast with other studies describing similar experiences, we pay special attention to the evaluation of specific issues which may impact on the system operation. In particular we focus our analysis on (i) the performance evaluation of the technique used by the network to forward the patient's location and health status, (ii) the power consumption of the smartphone that is used to forward the information received from a Bluetooth sensor, and (iii) the browser compatibility of the web application developed for the remote and real-time tracking of the patient.

3. System Description

The developed prototype of the monitoring system is integrated by the next subsystems, as shown in Figure 1.

(i) A Bluetooth BAN which is formed by a pulse-oximeter, a GPS receiver, and a smartphone. The smartphone along with the Python-developed control application acts as the Node of Control (NC) or master node (coordinator) of the Bluetooth piconet.

(ii) A Central Control Server (CCS) consisting in a web server with Python support, which centralizes the monitoring information received from the NC.

(iii) The remote Control and Monitoring Units (CMUs): a web application in charge of the control and monitoring of the BAN network. The developed web application should run in any conventional computer or mobile device with an Internet connection and a web browser.

All these components of the system and the communications interfaces between them are described in the following sections.

3.1. Architecture

3.1.1. BAN. The Body Area Network is integrated by the following components.

(i) A Nonin 4100 [35] pulse-oximeter and a GPS receiver, both provided with Bluetooth interfaces.

(ii) The Nonin 4100 pulse-oximeter measures several vital parameters, such as the heart rate (HR), the saturation of peripheral oxygen (SPO2), and the perfusion level. The pulse-oximeter supports two operational modes: (i) under simple mode 1, the device sends only 3 octets per second with basic information about the health status (basically the SPO2 value and the heart rate); (2) under the verbose mode 2, the device transmits three packets per second. Every packet includes 25 five-octet frames, which encapsulate the information of the plethysmogram, two different averaged estimations of the HR and the SPO2, and the battery status. On the other hand, the GPS receiver has been included considering the application of the system in a real scenario. The GPS would allow the emergency team to locate the patient in case of detecting alarm conditions.

(iii) A Nokia smartphone with Symbian OS S60 [36] (Series 60 User Interface), Python support, and Bluetooth and Wi-Fi [37] interfaces has been employed as the hardware platform for the NC component. The main reason to select a smartphone is the wide diffusion of these handheld devices in the market of consumer electronics. In fact, the total sales of smartphones in 2009 have attained a 36.4% of the global sales of mobile phones [38]. Besides, another advantage of using smartphones is the familiarity of general users with these electronic gadgets as well as the quick and easy installation of applications, such as what is concluded in [14].

For the NC component a Python application, whose graphical interface is shown in Figure 2, has been developed. This application provides the following functionalities:

(i) local configuration of the BAN (selection of sensors to be monitored),

(ii) establishment and management of Bluetooth communications with the BT devices of the BAN,

(iii) local monitoring: For this mode of operation, the application only shows a screen depicting the data received from the sensors,

(vii) remote monitoring: the application connects with the CCS server in order to forward the data measured by the sensors,

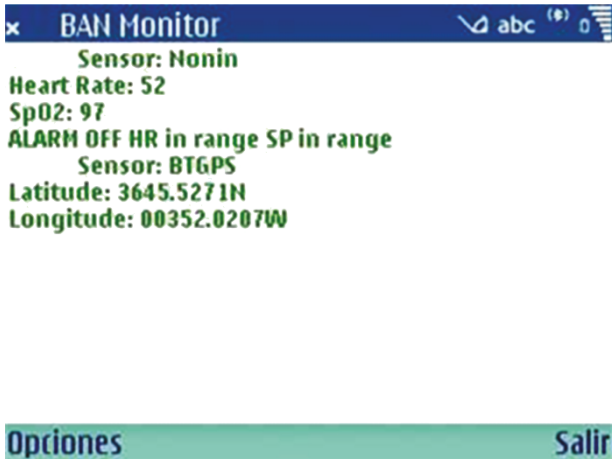

Figure 2: View of the Python application developed for NC (node of control) component.

(iv) management and execution of the configuration commands which are received from the CMU unit through the CCS,

(v) detection and notifications of events that occur when the heart rate (HR) and saturation of peripheral oxygen (SPO2) are out of the normality range (whose values can be remotely programmed).

3.1.2. Central Control Server (CCS). The internal structure of the CCS server is shown in Figure 3. The main component is the PyMHealth application, developed with Python programming language. This application has been deployed on CherryPy [39], an object-oriented HTTP (Hyper Text Transfer Protocol) [40] framework encoded in Python, which provides both a web server and an application server. Besides, as persistent layer, SQLObject [41] has been employed as it supplies a common object-oriented interface for different database management systems (DBMSs): PostgreSQL, MS SQL Server, SQLite, Sybase, and MySQL. Particularly, in a first prototype, MySQL [42] has been selected as DBMS, due to its simplicity and because its source code is available under the terms of the GNU General Public License [43].

The other components in the CCS are the following:

(i) An open source MQB (Message Queue Broker) that fully implements the Java Message Service 1.1 (JMS) [44], Apache ActiveMQ [45]: it has been used to forward the data of the patients to the CMUs. For the communication of the PyMHealth application with this broker the messaging protocol STOMP (Streaming Text Orientated Messaging Protocol) [46] has been chosen.

(ii) A comet server, Orbited [47]: this element allows that external applications interact with the messaging broker with HTTP protocol. Comet servers are web servers that permit sending data to a web client as soon as these data are generated, without waiting for any HTTP request from the client.

3.1.3. Control and Monitoring Units (CMUs). For the CMU component a web application, coded in JavaScript [48] and based on AJAX, has been developed. The advantage of using JavaScript is that the most popular web browsers include a native implementation of this language. Therefore, CMUs do not require the installation of any additional software or plug-in. On the other hand, AJAX technology enables CMUs to establish asynchronous HTTP connections in order to obtain data from the patients. In this way, the data visualized in the browser are automatically updated, without refreshing the web page continuously.

The CMUs retrieve the patient's information by accessing the web pages which are dynamically executed and generated by the CCS server. The main page is the DHTML (Dynamic Hyper Text Markup Language) [49] page "Monitor.htm," which is depicted in Figure 4. This page displays the monitored biosignals while it offers a simple interface to set up the configurable control parameters of the sensor. Thus, this page contains all the logic required by the communications with the CCS server for the reception and representation of the data collected by the sensors, as well as for sending configuration commands. Depending on the nature of the monitored signals, this page is automatically updated by using JavaScript routines and the DOM (Document Object Model) [50] interface. Specifically, aiming at obtaining the data corresponding to a particular sensor, the JavaScript object XMLHttpRequest [51] is employed. The execution of this object takes into account the differences between the most popular web browsers (so that the CMUs can be deployed on any common browser in a transparent way for the user). For example, the initial versions of Microsoft Internet Explorer [52] do not include a native implementation of XMLHttpRequest interface. In this case, the program automatically imports the ActiveX [53] object from the Microsoft XML Parser (MSXML) library [54].

3.2. Interfaces between Subsystems

3.2.1. Communication between the Node of Control (NC) and the Central Control Server (CCS) . For the communication between the NC and CCS components, based on HTTP Protocol, two channels are employed.

(i) *Data Channel.* The goal of this channel is to collect in the CCS the biosignal data that are forwarded by the NC. The NC can be optionally configured to transmit continuously. Thus, for any packet received from a device connected to the BAN, the NC sends to the CCS a POST HTTP message, whose body encapsulates the sensor data. Conversely, if the transmission has been set in a periodic basis (with a configurable period), all the packets received during the last period are assembled to be transmitted in a single HTTP request. In any case, the data are encoded into JSON (JavaScript Object Notation) [55], which is a text format based on a subset of JavaScript literals (Standard ECMA-262 [56]). JSON is independent from the programming language but

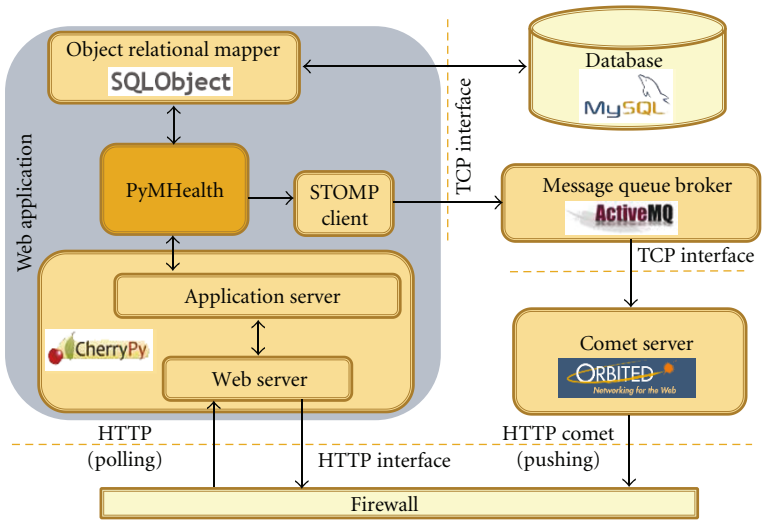

Figure 3: CCS server structure.

it shares notations with some languages, including Python. JSON has been selected instead of other formats, such as XML (eXtensible Markup Language) [57], due to its simplicity for encoding/decoding and because it is lighter in weight than XML.

(ii) *Control Channel*. This channel is conceived for the transmission/reception of configuration commands. By means of this channel the NC component periodically sends GET HTTP requests to the CCS in order to get the pending configuration commands. The frequency for this polling process is also a configurable parameter of the NC.

The way in which HTTP is utilized differs for these channels.

(i) In the case of the data channel, standard HTTP mode is used for periodic transmissions. On the other hand, for continuous transmissions, HTTP pipelining mode is employed. This nonblocking mode allows sending several consecutive requests without having to wait for the corresponding HTTP responses from the server. For this purpose, pipelining mode requires that the underlying HTTP connection operates in persistent mode. Consequently, the latency is minimized as the handshake and the overhead of establishing a new connection for each request is eliminated. Thus, the reuse of existing connections significantly improves the performance of the application.

(ii) The Control Channel utilizes the HTTP polling mode. This mode permits the client to query the server at regular intervals in order to get data which are asynchronously updated in the server.

The management of both data and control channels is centralized in a dedicated thread formed by three active objects. Two of them are responsible for the data channel: one is only in charge of sending HTTP requests, while the other one receives the responses from the server. Concurrently, the third active object periodically polls the server querying about the pending commands.

3.2.2. Communication between the Central Control Server (CCS) and the Control and Monitoring Units (CMUs). To acquire the data received by the CCS from the NC, the CMU units could periodically send HTTP requests to the server. In that case, the period of these requests should be short enough to guarantee the real-time signal monitoring. However, this conventional method could negatively impact on the system performance in several respects: less available bandwidth, more battery consumption (which would entail a serious problem for portable CMU units) and an unnecessary overload of the server. Alternatively, in order to avoid these drawbacks, the data received from NC are asynchronously forwarded by the CCS to the CMU units. For this purpose, the publish/subscribe model, also known as streaming HTTP, has been adopted. This communication model is a functionality provided by the Orbited daemon, a Comet server based on Twisted [58] (a Python event-driven framework for asynchronous communication between processes).

As it has been previously mentioned, the MQB Apache ActiveMQ and STOMP protocol are employed to forward to the CMU units the data of the sensors. Specifically, the PyMHealth application is responsible for publishing the data received from the NC at the queue of the MQB identified as */topic/ban/sensor*, where *ban* and *sensor* parameters, respectively, define the source BAN and sensor from which the data are being received. Besides, depending on the data to be monitored, every CMU has to subscribe to the corresponding queue. So, whenever a new STOMP message is received, a JavaScript function is invoked to extract data, which are encoded into JSON.

The STOMP client used by PyMHealth application to publish the data is a Pyhton client, while the client program

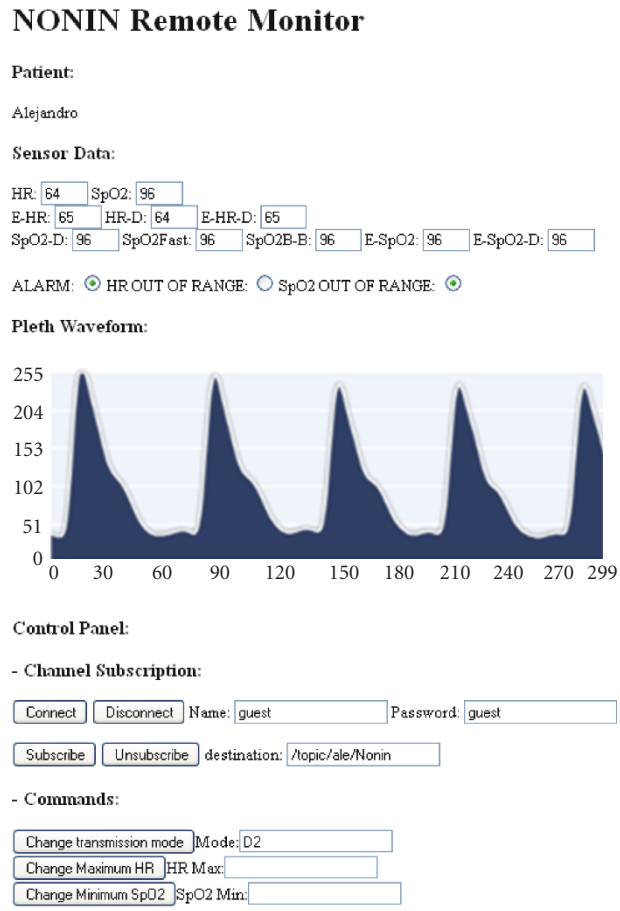

Figure 4: DHTML page "Monitor.htm".

that is executed on the CMU side is a JavaScript pull client provided by the Orbited daemon.

Additionally, three specific control commands have been contemplated to remotely configure the pulse-oximeter. These commands can be selected by the user in the DHTML page "Monitor.htm" (as it is represented in Figure 4). These commands define the operational mode of the sensor and the alarm thresholds for the HR and the SPO2 signals. Moreover, a general command has been included to configure the data transmission in a continuous or periodic basis. The CMU unit generates an HTTP POST request by using JavaScript object XMLHttpRequest. This POST HTTP request, sent to the CCS server, encapsulates the commands (which are encoded into JSON) together with the identifier of the BAN which the commands are targeted to.

4. Evaluation Tests

This section includes (i) the performance evaluation of the HTTP pipelining mode, used by the NC component to forward the patient's location and health status; (ii) the power consumption tests which are carried out for the NC component, under different monitoring conditions; (iii) the compatibility tests that have been executed for the CMU units with different browsers.

4.1. Performance Evaluation of HTTP Pipelining. The goal is to evaluate the performance of the HTTP pipelining mode, used by the NC component to forward the patient's data to the CCS. The evaluation has been carried out in terms of Round-Trip delay Time (RTT), by comparing the pipelining technique with other HTTP modes. For this test, several Python client applications have been developed for the following operating HTTP modes.

(i) *HTTP Socket*. The socket directly sends HTTP messages through the TCP interface implemented by *socket.py*.

(ii) *HTTP Nonpersistent*. This mode employs the *httplib.py* Python module for the standard distribution of information with non-persistent connections.

(iii) *HTTP Persistent*. It also uses the *httplib.py* module. However, in this case the TCP connection is not closed until the HTTP connections have concluded.

(iv) *HTTP Pipelining* (*Header*). a variant of the previous mode that applies *pipelining* to the header.

(v) *HTTP Pipelining*. It employs the *httplib.py* module, at lowlevel, to implement the pipelining mode.

Under all these modes, the client establishes a sequence of HTTP connections, measuring the resulting RTT. By averaging successive iterations of the tests, the measured average RTT for every client is graphically represented in Figure 5. The graph also shows the results obtained through a raw TCP socket (which obviously involves less overhead and less delay). As it can be observed, the pipelining technique is the most efficient method when compared with the other methods provided by the *httplib.py* module.

4.2. Power Consumption Tests. This subsection describes the analysis of power consumption that has been conducted for the NC component. In particular, the consumption tests have been executed on a Nokia N95 smartphone, acting as the NC. During all the measurements, "Nokia Energy Profiler 1.2" application [59] (a specific software for Symbian S60 3rd Edition, FP1, and later versions) has been used. This application generates an owner file, which can be exported to CSV (Comma Separated Values) format, containing information about several performance metrics of the smartphone, such as power and current consumption, average and instant battery voltage, CPU load, RAM memory usage, and downlink (download) and uplink (upload) speeds through the employed IP stack. Basing on the measurements logged in this file, the power consumption has been evaluated when the Python application of the NC component is running.

Initially, the efficiency of HTTP pipelining mode, employed by NC component in the continuous transmission, has been compared with the standard HTTP mode. From the results, it can be concluded that HTTP pipelining mode, which is the most efficient technique in terms of delay, introduces a 12% decrease of the battery autonomy.

Additionally, the results have been compared with the measurements obtained with an equivalent transmission

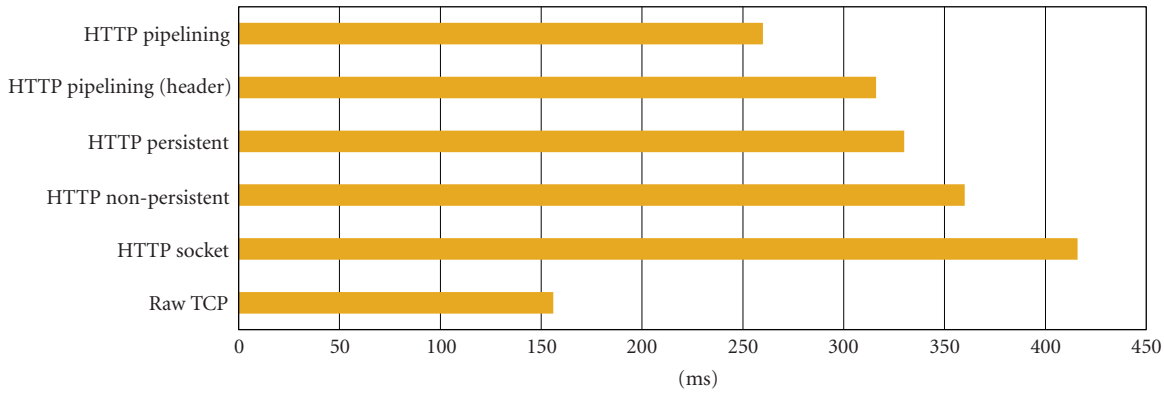

Figure 5: Average round-trip delay time (ms) for different HTTP clients.

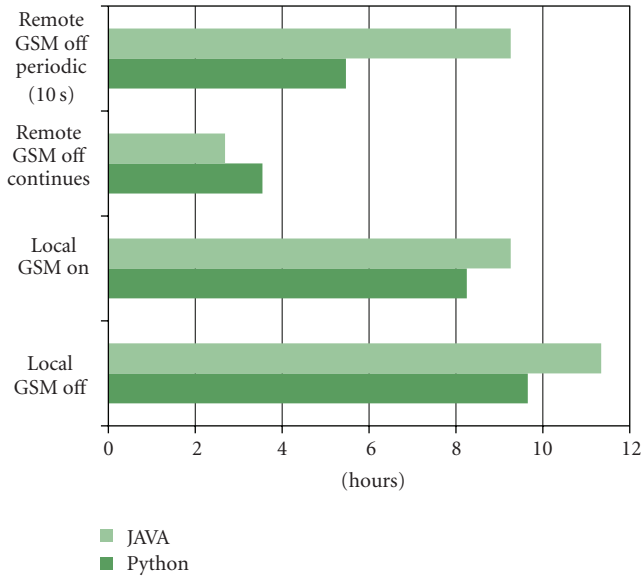

Figure 6: Results (estimation of the battery lifetime in hours) of the autonomy tests for the NC component.

software developed for Java ME (Java Micro Edition) platform [60], when the pulse-oximeter device operates in mode 2, without getting location information and in the next cases:

(i) Local monitoring:

  (a) With GSM (conventional mobile) communication enabled.

  (b) With GSM communication disabled.

(ii) Remote monitoring: with GSM communication disabled. For these tests, standard HTTP (nonpersistent) is used for both Python and Java ME versions of the program. Two cases are considered:

  (a) continuous transmissions of the sensor information,

  (b) periodic transmission: Every 10 s, an HTTP request with all packets received during this interval is sent to the CCS.

From the results which are shown in Figure 6, it can be inferred the following:

(i) Local monitoring: with the Java ME encoded client a greater autonomy is obtained in the NC, independently that GSM communication is enabled (12%) or disabled (18%). Nevertheless, the difference is not very significant if we take into account that Python is a very high-level language (VHLL), even more than Java.

(ii) Remote monitoring and continuous transmission: Python program outperforms Java ME version with a 32% increase in the autonomy.

(iii) Remote monitoring and periodic transmission: the energy efficiency which is achieved with the Java ME version is remarkably higher to that measured for the Python version (with a 69% of reduction in the battery consumption). For this case, it is worth pointing out that the method used to send the data to the CCS (with JSON encoding) prevents from improving the performance. Specifically, the efficiency loss is mainly provoked by the Wi-Fi transceiver and the additional CPU load that implies the serialization of data according to JSON format: with Java ME version, the instantaneous rate in the uplink, every 10 s, does not exceed 4.3 KB/s while the measured CPU load is 18%. Conversely, with Python version, a rate of 22 KB/s is achieved and the CPU load rises to 51%. Therefore, although JSON simplifies the encoding and decoding processing, it is less efficient than binary transmission, which is the method used by Java ME version.

4.3. Web Browser Compatibility Tests. Existing web browsers (Safari, Firefox, IE, Opera, Chrome, S60 Browser, among the most popular ones) [52, 61–65] are theoretically supposed to fulfill the web standards defined by organizations such as World Wide Web Consortium (W3C) [66]. However, in the actual implementations, there are different interpretations of the standards and compatibility problems may appear.

Table 1: Compatibility of the web application developed for the CMU units with different existing web browsers.

Web browser	Compatible
FIREFOX 3	YES
SAFARI 3	YES
IE 7	YES
CHROME 0.2	YES
OPERA 9	NO
OPERA 8 (Symbian S60)	NO
OSS Browser v1.0/v2.0 (Symbian S60)	NO

Table 2: Simultaneous persistent connections that a browser can support.

Web browser	Number of simultaneous persistent connections
FIREFOX 3	≥6
SAFARI 3	4
IE 7	2
CHROME 0.2	≥6
OPERA 9	4
OPERA 8 (Symbian S60)	2
OSS Browser v1.0/v2.0 (Symbian S60)	1

Furthermore, certain aspects of the implementation of the standards may be inefficient for mobile platforms, such as those that are based on Symbian S60. In addition, novel features, as support of XMLHttpRequest object, are only specified in draft versions of the standard [51], although this feature has been incorporated to the most popular browsers, due to the wide diffusion of AJAX technologies in the web applications context. Therefore, it is presumable that the employed web technologies will not be error-free and will not work with the same efficiency in all the typical browsers [67]. For these reasons we have considered of great interest the evaluation of the compatibility of the web application developed for the CMU units when it is run on different web browsers. In fact, in a first step, it has been verified if the connections with Orbited server through STOMPClient object can be properly established for the web browsers. The results are summarized in Table 1.

In a second stage, upon checking that the implementation of STOMP client is not compatible with all the considered browsers (problems appear with the browsers in mobile platforms), other tests have been executed in order to find out the reason which causes this malfunction. Specifically, the relationship between the behavior of the XMLHttpRequest object and the numbers of simultaneous persistent connections that a browser can support has been studied. The underlying problem is that an Orbited STOMP Client requires, at least, the usage of two simultaneous persistent HTTP connections. In order to estimate the numbers of simultaneous persistent connections that a browser can support, a specific server over CherryPy has been developed. The results are shown in Table 2. In this table it can be observed that Nokia S60 browser is not able to establish more than a single connection. So, we can conclude that this is the reason why STOMP client does not properly work. Consequently the CMU cannot be executed in a smartphone due to the limitations of current browsers for mobile platforms.

5. Conclusions

Remote monitoring systems represent one of the most promising technological research areas in the health context, especially because its application to the management of chronic diseases would have a significant economic impact. However, to ensure that the potentialities of remote health monitoring are fully developed and guaranteed, more practical trials and realistic testbeds (with real patients) are needed, especially to assess the economic viability of the monitoring applications. Nevertheless, besides the need for empirical studies to evaluate the cost-effectiveness, it cannot be neglected that the maturity of the technologies involved in the development of the applications may seriously impact on its performance. The presented paper has focused on the evaluation of different Web technologies that can be employed to deploy the system software for the remote monitoring of biosignals. Specifically, the paper has presented a prototype of a monitoring system based on BAN (Body Area Network) that is worn by the patient. This BAN integrates diverse Bluetooth sensors and a smartphone. The smartphone along with the Python-developed control application acts as the coordinator node (NC, Node of Control) of a Bluetooth piconet formed by the sensors. This component forwards the data received from the Bluetooth devices to a Central Control Server (CCS).

On the other hand, the physicians can control and monitor the patient's BAN from Remote Control and Monitoring Units (CMUs). For these units we have developed a web application, based on AJAX (Asynchronous JavaScript and XML) technology, which retrieves the patient's information through the CCS.

The presentation of the prototype is accompanied by a study on specific issues which can impact on the applicability of the system software, in particular, (i) the HTTP pipelining technique which is used by the NC component to forward the patient's location and health status; (ii) the power consumption of the NC component (smartphone), which is compared with the measured consumption when an equivalent Java ME software is employed; (iii) the web browser compatibility of the web application developed for CMU units.

From the results it can be concluded that HTTP pipelining mode, employed by NC component in the continuous transmission, is the most efficient method, although it implies a decrease in the battery autonomy with respect to the standard HTTP mode. Additionally, although the employed JSON encoding format is lighter in weight than XML, it is less efficient than the binary transmission, which is the method used by the Java ME version. The energy efficiency which is achieved with Java ME version is significantly higher than that measured with Python.

As it refers to the web browser compatibility, it has been verified that the STOMP client does not work properly for all considered browsers. The reason which causes this malfunction is the number of persistent HTTP connections that the browsers can support, as an Orbited STOMP Client requires at least two connections.

The described prototype is currently being extended to other technologies. In particular, a Java prototype for Android platform is being developed. For this new prototype other biosensors are going to be integrated. In addition, other operation modes will be implemented. For example, in order to minimize the power consumption, a surveillance mode is planned to be included. Under this mode, only severe events would be notified to the server. Additionally, the physicians will be able to remotely configure the periodicity of specific measurements, such as the blood pressure and the electrocardiogram.

Acknowledgment

This work has been supported by Spanish National Project no. TEC2009-13763-C02-01.

References

[1] S. Pruitt, S. Annandale, J. Epping-Jordan et al., *Innovative Care for Chronic Conditions: Building Blocks for Action*, World Health Organization, Geneva, Switzerland, 2002.

[2] National Centre For Chronic Disease Prevention and Health Promotion, "Power of prevention: chronic disease...the public helath challenge of the 21st century," 2009, http://www.cdc.gov/chronicdisease/pdf/2009-Power-of-Prevention.pdf.

[3] G. Anderson, *Chronic Conditions: Making the Case for Ongoing Care*, The Robert Wood Johnson Foundation, 2002, http://www.rwjf.org/pr/product.jsp?id=14197.

[4] G. Anderson, *Chronic Care: Making the Case for Ongoing Care*, The Robert Wood Johnson Foundation, 2010, http://www.rwjf.org/pr/product.jsp?id=50968.

[5] T. Penzel, K. Kesper, and H. F. Becker, "Biosignal monitoring and recording," in *Information Technology Solutions for Healthcare*, K. J. Hannah, M. J. Ball, K. Zielinski, M. Duplaga, and D. Ingram, Eds., Health Informatics, pp. 288–301, Springer, London, UK, 2006.

[6] M. Duplaga and O. M. Winnem, "Model of chronic care enabled with information technology," in *Information Technology Solutions for Healthcare*, K. J. Hannah, M. J. Ball, K. Zielinski, M. Duplaga, and D. Ingram, Eds., Health Informatics, pp. 248–270, Springer, London, UK, 2006.

[7] N. F. Güler and E. D. Übeyli, "Theory and applications of telemedicine," *Journal of Medical Systems*, vol. 26, no. 3, pp. 199–220, 2002.

[8] M. Tounsi and B. Qureshi, "A Bluetooth-enabled mobile intelligent remote healthcare monitoring system: analysis and design issues," *International Journal of Healthcare Technology and Management*, vol. 9, no. 5-6, pp. 473–484, 2008.

[9] S. Neubert, D. Arndt, K. Thurow, and R. Stoll, "Mobile real-time data acquisition system for application in preventive medicine," *Telemedicine and e-Health*, vol. 16, no. 4, pp. 504–509, 2010.

[10] S. Winkler, M. Schieber, S. Lücke et al., "A new telemonitoring system intended for chronic heart failure patients using mobile telephone technology—feasibility study," *International Journal of Cardiology*. In press.

[11] R. S. H. Istepanian, E. Jovanov, and Y. T. Zhang, "Introduction to the special section on m-Health: beyond seamless mobility and global wireless health-care connectivity," *IEEE Transactions on Information Technology in Biomedicine*, vol. 8, no. 4, pp. 405–414, 2004.

[12] U. Varshney, "Pervasive healthcare and wireless health monitoring," *Mobile Networks and Applications*, vol. 12, no. 2-3, pp. 113–127, 2007.

[13] K. Jeong, E. Y. Jung, and D. K. Park, "Trend of wireless u-health," in *Proceedings of the 9th International Symposium on Communications and Information Technology (ISCIT '09)*, pp. 829–833, September 2009.

[14] F. del Pozo, P. de Toledo, S. Jiménez, M. E. Hernando, and E. J. Gómez, "Chronic patient's management: the Copd example," in *M-Health: Emerging Mobile Health Systems*, Topics in Biomedical Engineering, pp. 575–585, Springer, London, UK, 2006.

[15] A. Tura, L. Quareni, D. Longo, C. Condoluci, A. van Rijn, and G. Albertini, "Wireless home monitoring and health care activity management through the Internet in patients with chronic diseases," *Medical Informatics and the Internet in Medicine*, vol. 30, no. 4, pp. 241–253, 2005.

[16] R. Ciorap, D. Zaharia, C. Corciovă, M. Ungureanu, R. Lupu, and A. Stan, "Wireless device for monitoring the patients with chronic disease," *Revista Medico-Chirurgicala a Societatii de Medici si Naturalisti din Lasi*, vol. 112, no. 4, pp. 1115–1119, 2008.

[17] K. Perakis, M. Haritou, R. Stojanovic, B. Asanin, and D. Koutsouris, "Wireless patient monitoring for the e-inclusion of chronic patients and elderly people," in *Proceedings of the 1st International Conference on Pervasive Technologies Related to Assistive Environments (PETRA '08)*, pp. 1–4, July 2008.

[18] G. Chen, B. Yan, M. Shin, D. Kotz, and E. Berke, "MPCS: mobile-phone based patient compliance system for chronic illness care," in *Proceedings of the 6th Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous '09)*, pp. 1–7, July 2009.

[19] S. Sultan and P. Mohan, "How to interact: evaluating the interface between mobile healthcare systems and the monitoring of blood sugar and blood pressure," in *Proceedings of the 6th Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous '09)*, pp. 1–6, July 2009.

[20] D. Capozzi and G. Lanzola, "An agent-based architecture for home care monitoring and education of chronic patients," in *Proceedings of the Workshop on Complexity in Engineering (COMPENG '10)*, pp. 138–140, February 2010.

[21] M. Jaana, G. Pare, and C. Sicotte, "Home telemonitoring for respiratory conditions: a systematic review," *American Journal of Managed Care*, vol. 15, no. 5, pp. 313–320, 2009.

[22] G. Paré, M. Jaana, and C. Sicotte, "Systematic review of home telemonitoring for chronic diseases: the evidence base," *Journal of the American Medical Informatics Association*, vol. 14, no. 3, pp. 269–277, 2007.

[23] J. G. F. Cleland, C. Lewinter, and K. M. Goode, "Telemonitoring for heart failure: the only feasible option for good universal care?" *European Journal of Heart Failure*, vol. 11, no. 3, pp. 227–228, 2009.

[24] R. S. H. Istepanian, K. Zitouni, D. Harry et al., "Evaluation of a mobile phone telemonitoring system for glycaemic control in patients with diabetes," *Journal of Telemedicine and Telecare*, vol. 15, no. 3, pp. 125–128, 2009.

[25] A. B. Cunha, M. A. M. Capretz, and L. Raptopoulos, "Support systems for Telehealth services: critical operational and ICT complementary assets for large-scale provisioning," in *Proceedings of the IEEE Toronto International Conference in Science and Technology for Humanity (TIC-STH '09)*, pp. 340–345, September 2009.

[26] Python, Python Programming Language, http://www.python.org/.

[27] J. J. Garrett, "Ajax: A New Approach to Web Applications," February 2005, http://www.adaptivepath.com/ideas/essays/archives/000385.php.

[28] Bluetooth SIG, Bluetooth, http://www.bluetooth.com.

[29] ZigBee Alliance, http://www.zigbee.org.

[30] M. J. Morón, J. R. Luque, A. Gómez-Jaime, E. Casilari, and A. Díaz-Estrella, "Prototyping of a remote monitoring system for a medical personal area network using python," in *Proceedings of the 3rd International Conference on Pervasive Computing Technologies for Healthcare—Pervasive Health (PCTHealth '09)*, pp. 1–5, April 2009.

[31] MobiHealth project, http://www.mobihealth.org.

[32] N. Oliver and F. Flores-Mangas, "HealthGear: a real-time wearable system for monitoring and analyzing physiological signals," in *Proceedings of the International Workshop on Wearable and Implantable Body Sensor Networks (BSN '06)*, pp. 61–64, April 2006.

[33] Y. Zhang and H. Xiao, "Bluetooth-based sensor networks for remotely monitoring the physiological signals of a patient," *IEEE Transactions on Information Technology in Biomedicine*, vol. 13, no. 6, pp. 1040–1048, 2009.

[34] Healthcare@Home, Healthcare@Home: Patient-Centered Grid Based e-Healthcare, http://www.healthcareathome.info/index.html.

[35] Nonin Medical, Nonin 4100 Bluetooth, Wireless Pulse Oximeter, http://www.nonin.com/OEMSolutions/4100.

[36] Nokia, Symbian S60 Platform, http://www.s60.com.

[37] Wi-Fi Alliance, http://www.wi-fi.org.

[38] Gartner Inc., "Gartner Says Worldwide Mobile Phone Sales to End Users Grew 8 Per Cent in Fourth Quarter 2009; Market Remained Flat in 2009," http://www.gartner.com/it/page.jsp?id=1306513.

[39] CherryPy, CherryPy project, http://www.cherrypy.org.

[40] W3C, HTTP—Hypertext Transfer Protocol, http://www.w3.org/Protocols.

[41] SQLObject, SQLObject Project, http://www.sqlobject.org.

[42] MySQL, MySQL Comunity Server, http://www.mysql.com/downloads/mysql.

[43] GNU, GNU General Public License, http://www.gnu.org/licenses/gpl.html.

[44] Oracle, Java Message Service (JMS) API, http://www.oracle.com/technetwork/java/index-jsp-142945.html.

[45] Apache Software Foundation, Project Apache ActiveMQ, http://activemq.apache.org.

[46] STOMP Project, Streaming Text Orientated Messaging Protocol, http://stomp.codehaus.org.

[47] The Orbited Project, Orbited: Real-Time Communication for the Browser, http://orbited.org.

[48] Mozilla Foundation, Mozilla Developer Network. JavaScript, https://developer.mozilla.org/en/JavaScript.

[49] Mozilla Foundation, Mozilla Developer Network, DHTML, https://developer.mozilla.org/en/JavaScript.

[50] W3C, HTML DOM Specification, http://www.w3.org/DOM.

[51] W3C, XMLHttpRequest W3C Working Draft, http://www.w3.org/TR/XMLHttpRequest.

[52] Microsoft, Windows Internet Explorer, http://www.microsoft.com/windows/internet-explorer/default.aspx.

[53] Microsoft, Introduction to ActiveX Controls, http://msdn.microsoft.com/en-us/library/aa751972%28VS.85%29.aspx.

[54] Microsoft, Microsoft XML Core Services (MSXML), http://msdn.microsoft.com/en-us/library/ms763742%28v=VS.85%29.aspx.

[55] JSON, JSON (JavaScript Object Notation) Specification, http://www.json.org.

[56] ECMA International, Standard ECMA-262. ECMAScript Language Specification, http://www.ecma-international.org/publications/standards/Ecma-262.htm.

[57] W3C, Extensible Markup Language (XML), http://www.w3.org/XML.

[58] T. M. Labs, Twisted Project, http://twistedmatrix.com/trac/wiki/TwistedProject.

[59] Nokia, Nokia E61 specifications, http://www.forum.nokia.com/devices/E61.

[60] Oracle, "Java ME—the Most Ubiquitous Application Platform for Mobile Devices," http://www.oracle.com/technetwork/java/javame/overview/index.html.

[61] Apple Computer, Sarfari Browser, http://www.apple.com/safari.

[62] Mozilla Foundation, Mozilla Firefox, http://www.mozilla.com/en-US/firefox.

[63] Opera Software, Opera Browser, http://www.opera.com.

[64] Google, Chrome Browser, http://www.google.com/chrome.

[65] Nokia, Nokia Mini Map Browser, http://www.nokia.com/microsites/s60-browser-site.

[66] W3C, World Wide Web Consortium (W3C), http://www.w3.org.

[67] Apple Developer Network, Web Page Development: Best Practices, http://developer.apple.com/internet/webcontent/bestwebdev.html.