ORIGINAL ARTICLES

Lessons learned from the development of an affordable fall detection system for mHealth

Miguel Ángel Estudillo Valderrama[1], Laura M Roa Romero[1], Luis Javier Reina Tosina[2,1], Gerardo Barbarov Rostan[1], David Naranjo Hernández[1]

1. Biomedical Engineering Group, University of Seville, CIBER-BBN, Seville, Spain. 2. Department of Signal Theory and Communications, University of Seville, Seville, Spain.

Correspondence: Miguel Ángel Estudillo Valderrama. Address: Biomedical Engineering Group, Escuela Técnica Superior de Ingenieria, University of Seville, Seville 41092, Spain. Email: mestudillo@us.es

Received: June 29, 2015 **Accepted:** August 2, 2015 **Online Published:** August 12, 2015
DOI: 10.5430/jbei.v1n1p59 **URL:** http://dx.doi.org/10.5430/jbei.v1n1p59

Abstract

This paper discusses some relevant methodological and implementation experiences acquired during the design and development of an embedded Fall Detection System (FDS), which can be of help in order to develop efficient and safe biomedical software for mobile Health (mHealth). For this purpose, an analysis of concepts like portability and iterative design, as well as some concerns about risks and safety involved, is provided in order to address some of the current challenges in embedded software, regarding the state-of-art of software development standards and mHealth technologies. This analysis is later evaluated for a custom pre-industrial prototype of the FDS, as an example of the feasibility of the approach followed. The results obtained show that a convenient methodological process can help to optimize available resources so as to provide affordable mHealth solutions.

Key words

Embedded software, mHealth, Methodologies, Sustainability, Real-time

1 Introduction

The massive use of connected devices in everyday-settings, empowered by upcoming paradigms as the Internet of Things (IoT), is a reality nowadays [1] and will become a key factor in the future of the digital society, as many predictions claim [2, 3]. This emerging ecosystem is also valid in the healthcare domain, where it is highlighted the increasing importance of unobtrusive sensing technologies for the monitoring of physiological and health conditions through Body Sensor Networks (BSN) and Medical Devices (MDs) [4]. At this point, it is necessary the participation of low-cost, low-power, small-sized, robust and portable devices, with processing and sending user-monitored information capabilities [5]. Thus, it is required a comprehensive envision of biomedical embedded software that addresses engineering challenges like iterative development, proper verification, and management of the safety and risks associated.

Software portability is a good sample, as it encompasses many of the requirements described since the very early stages of Computer Science. A precise definition of the concept can be found in the standard ISO SQUARE 25010 [6], where it is defined as the "ease with which a system or component can be transferred in an efficient and effective way from one hardware, software or operation environment to another" (sic). This main feature is in turn divided in the following sub-characteristics: adaptability, capacity to be installed and capacity to be replaced. Ref. [7] remarks that portability is the

key issue for cost reduction and sustainable development, which is a vital requirement in any engineering area [8]. Among these areas, the upcoming paradigms of mHealth [9] have shown the importance of software portability. A common approach described in Ref. [10] addresses the rapidly development of Smartphone and in general the touch-pad market, where it must be highlighted the explosion of the Android mobile operating system. Although, for certain applications, this approach may demand too much resources that may not be needed [11]. In addition, current context of economic and environmental awareness demands that the development of technologies walks in hand with their re-use. Therefore, there is an inevitable tradeoff to be tackled between pursuing global embedded software portability, and dealing with sustainable requirements, as it may be the case for the multiple mHealth applications [12]. However, it must be noticed that software portability in healthcare, as well as in other industrial contexts with strict privacy rules, must address issues like certification that difficult its exportation into different platforms without running a separated certification project.

Other important issues to tackle when developing MD software, either portable or not, are safety and risks associated whilst assuring the best Quality of Service (QoS) of communications. In this context, there are different standards and procedures that define a general iterative framework to guide the developers. In a first place, ISO 14971 [13] establishes the requirements for risk management in order to determine the safety of a MD by the manufacturer during the product life cycle. In a second place, ISO 13485 [14] includes the requirements for a comprehensive management system to design and manufacture MDs. Finally, due to the lack of specific standards for MDs, the IEC 62304 [15] initiative emerges as an integration effort so as to define requirements for the whole life cycle of medical embedded software: from the first stages of design to the final tests and verifications. Besides, IEC 62304 defines specific requirements to identify factors that may contribute to additional risks for the user derived from MD software, like the ones that may appear when porting to different platforms.

Nevertheless, despite the novelty of these efforts and to the best of authors' knowledge, there is a lack of studies in the scientific literature or the industry on how to take advantage of current procedures so as to address affordable mHealth solutions. In the meantime, vendors and manufacturers [16] are progressively paving the way to the optimization of software development through the adoption of these standards and the use of iterative processes and methodologies, although there is still room for improvement, for instance in terms of usability for software developers [17]. In fact, embedded mHealth systems are characterized by very strict operating conditions, which refer to runtimes that allow real-time responses in order to provide an advanced monitoring. Besides, the limited resources in terms of memory and processing of embedded systems are added, which require the optimization of applications through real-time Embedded Operating Systems (EOS) [7]. Hence, it is essential the knowledge for co-development at hardware (HW) and software (SW) levels. At this point, C programming language arises has a reference tool for the co-development, like in Ref. [9, 18], where code is compliant with ANSI C standard, and open-source libraries are used to minimize portability issues.

Thus, the main contribution of this paper is to share with the scientific community the lessons learned from the development of a specific mHealth system able to detect human falls, called FDS, which involves the challenges described. This paper is not intended to be a comprehensive guide for embedded software developers, but instead it details some important and useful insights about the methodological issues that may arise when developing affordable embedded biomedical software, regarding current state-of-art of mHealth standards and technologies. The manuscript is organized as follows: Section II defines the methodological approach adopted, as well as the methods and materials employed; Section III analyzes the main results obtained for the embedded software performance, and its development viability considering sustainable technologies; Finally, Section IV provides some discussion and concluding remarks.

2 Methods and materials

2.1 Fall detection system

The Fall Detection System (FDS) is defined by a BSN that consists of a single intelligent accelerometer unit (IAU), among other smart sensors, which is placed in the lumbar region of the monitored subject, and a personal server (PSE) [19]. The

FDS follows a distributed processing paradigm in real-time where the IAU performs a rough signal pre-processing, whereas the PSE is responsible for conducting further analysis to determine events of interest: the occurrence of falls [20]. Besides, the PSE is composed of a set of embedded software modules responsible for various functions, such as the internal operation of the system, the bi-directional communication with smart sensors (up-link) and service providers (down-link), or the standardization of the medical information generated in the point of care (PoC) for its inclusion in the healthcare communications infrastructure.

The core of the pre-industrial prototype of the PSE is supported by a floating point Digital Signal Processor (DSP) from Texas Instruments (TI) TMS320C6727 family. The prototype (see Figure 1) also incorporates a 1MB flash memory in order to store code for both signal processing and management of internal data. The HW of the PSE is completed with the communications module and the user interface, consisting of both visual and auditory elements that let the interaction with the server be eased. A deeper hardware analysis of the prototype can be found in Ref. [19]. By the other hand, the up-front requirements for the product development of the PSE have been defined through network QoS metrics, related to a safe data transfer rate and a minimum alarm time delay response, which were assessed during system test and verification in several settings (see Section 3). In particular, and for the sake of the evaluation of the FDS, it has been established [21] a maximum response time of 1 second for the FDS and two kinds of setting scenarios (indoor and outdoor) for the test experiments. Thus, the paper focuses on how the EOS within the PSE addresses these requirements for an application where time response is critical, as human fall detection. Besides the EOS, due to the multimodal design of the PSE must be designed to be able to manage different smart sensors in real time, and therefore it has to host a set of algorithms for different types of processing [22]. For these reasons, embedded processing modules of the PSE have strict requirements in terms of memory allocation and time response that give rise to important considerations in the software development process.

Figure 1. Preindustrial prototype of the PSE. The central part is mainly occupied by the DSP, while the lower part is reserved for the Zigbee (left) and GSM/GPRS (right) communication modules.

2.2 Software development process

The guidance for the process of software development of the PSE has been to follow the EN/IEG 62304 standard [15], which is the de-facto benchmark for management of the medical software development lifecycle. This way, a design compliant with IEG 62304 ensures that quality software is produced by means of a defined and controlled process of software development. Besides, the standard defines particular processes to cover other fundamental areas: maintenance, software configuration, problem resolution and risk management [13]. The latter is especially important to ensure patient safety, and contains a set of requirements based on the safety class of the software that is being developed. In the first place, risks can be addressed by a divide-and-conquer strategy, in a scenario where different software items are run on

different processors, each of which having a different safety classification. In the second place, the standard proposes the need of a quality management system [23] that demonstrate the capability of SW to meet user and legal requirements.

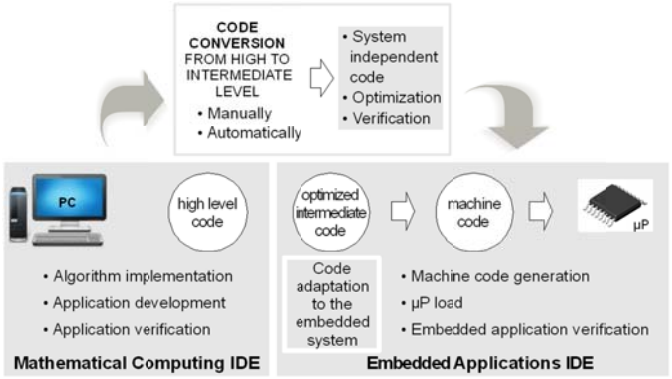

Figure 2. Functional diagram of the development procedure for EOS

On the other hand, given the strong correlation between the SW/HW resources available in the PSE, it is advisable to adopt an iterative implementation approach, rather than the traditional waterfall cycle, which is very efficient for software debugging and reuse. This iterative approach has been chosen as the best option to fulfill all the requirements of the PSE optimally [15], and has been mainly addressed by agile methodologies [23], in particular Scrum [24]. It consists in developing functional software that is progressively debugged and optimized through a set of incremental changes in short work cycles so as to meet the user design requirements and facilitate future redesigns. This way, the approach followed was to consider the risks posed by the MD as a whole, before the SW/HW division has been decided. Hardware risk analysis can then run alongside software risk analysis to define the required safety systems for the device. In order to detail the approach followed for the SW development of the PSE, the authors will mainly focus on one of the most important issues of an embedded system: the digital signal processing. This way, the authors considered two main alternatives to develop the real-time signal processing algorithms to be hosted on an embedded system: Firstly, they can be directly implemented within the CPU (DSP) of the system using low-level language (machine or assembler code), which is ceasing to be a common practice. This is explained by the evolution of development tools able to generate machine code from the compilation of high-level source code, which is normally embedded C language (see right bottom of Figure 2). Thus, the second alternative is to use integrated development environments (IDEs) that ease a two-layer software development: high-level (algorithm implementation in a controlled environment through a mathematical IDE) and low-level (on-platform algorithm verification through an embedded IDE). This alternative allows the best optimization of the algorithms developed, although the complexity of machine code and the specificity of proprietary C libraries increases the learning curve due to its heavily dependence of the particular platform.

Table 1. Main Software modules of the PSE

Module	Peripheral	DSP Interface	Safety Class
Communications	Zigbee	SPI	B (C)
	GSM/GPRS	GPIO & I2C	
User Interface	LCD	I2C	A
	Audio	SPI & I2C	
	Keypad	GPIO	
DSP and Power Mgt.	Flash	EMIF	B (C)
	Battery	I2C	

According to IEC/EN 62304, Safety Class A means “No injury or damage to health is possible”; Safety Class B means “Non serious injury is possible”; Safety Class C means “Death or serious injury is possible”.

 ISSN 2377-9381 E-ISSN 2377-939X

Nevertheless, when building the embedded application within the particular constraints of the embedded system, it is often useful to take a step back and consider the conversion to an intermediate system-independent programming language (see Figure 2 at the top). Developing an intermediate code is a powerful tool for multi-platform applications where the same software might be hosted in several devices. This way, the portability of the algorithmic software is empowered [17]. For all these reasons, the design of the PSE has followed the approach depicted in Figure 2.

3 Results

This section has four parts. The first two provide a general perspective of the PSE SW in order to study safety considerations. Later on, a deeper analysis of the approach followed for the development of the fall detection algorithm is provided, regarding current technologies. Finally, the approach proposed is evaluated from two perspectives: 1) real-time performance; 2) test-bed verification.

3.1 Embedded operating system of the PSE

In order to develop the EOS of the PSE, the main features provided by the DSP/BIOS tool [25] were used. In particular, a concurrent programming of the three main modules has been done through software tasks (TSK). Besides, a set of hardware interrupt routines (HWI) have been established to attend interruptions like those coming from user interface. Software functions (SWI) have also been used, which are triggered to release the load of HWI routines due to their smaller deadline execution. Furthermore, periodic functions (PRD) have been employed to implement the periodic modules needed. The three first columns of Table 1 lists the inter-device standards applied, either using I/O lines emulated through general purpose pins (GPIO) or by means of specific peripherals embedded in the DSP, such as the Inter-Integrated Circuit (I2C), the Serial Peripheral Interface (SPI) or the External Memory Interface (EMIF) ports. The last column refers to the safety classes assigned to the different software modules, which is discussed later on.

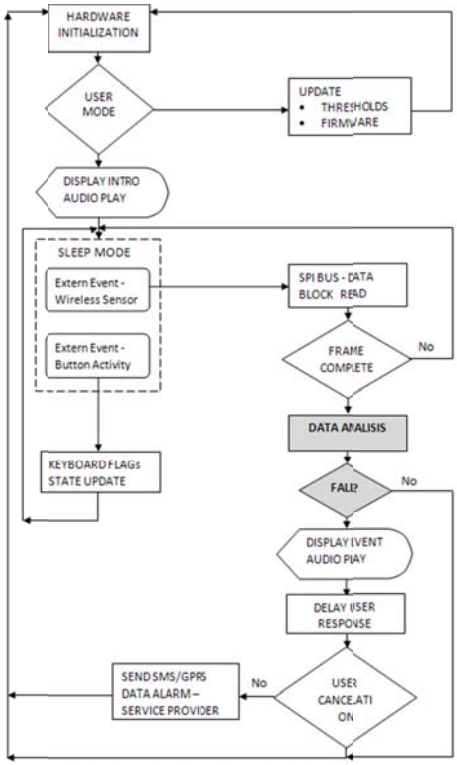

Figure 3. Flow diagram of the EOS of the PSE. Shaded blocks represent the main software modules for the fall detection

Figure 3 shows a flow diagram of the EOS of the PSE when dealing with fall events: A TSK thread devoted to communications listens to the SPI port through a HWI routine that is activated when the arrival of any sensor data from the IAU occurs. Then, the DSP executes the appropriate tasks to process the signal and if a fall event is detected, the PSE sends an alert to the service provider through the communications module. This alarm event is also triggered if the user presses a dedicated button of the PSE. When none of the threads is running or the processing thread is finished, the energy saving module is executed, resulting in a decrease of the switching frequency of the state logic CMOS circuitry of the DSP and an idle state of the CPU, waiting for an interruption to get it out of that state [26].

3.2 Safety assessment and management

The IEC/EN 62304 standard defines a safe system if it does not create a hazard that could result in an injury for the users. According to this definition, the fall detection system as a whole may be considered as Class C (see caption of Table 1) if the time response or its absence could indirectly worsen the consequences originated by a fall, in the worst case scenario.

However, the divide-and-conquer strategy followed for the safety management in the FDS has been the software classification in two categories of safety classes, as stated in the standard.

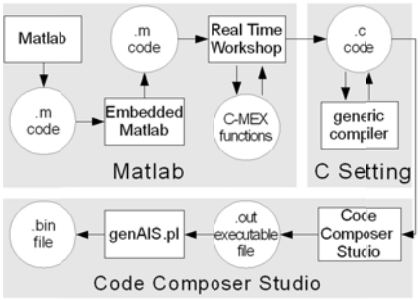

Figure 4. Proposed design procedure for the FDS

C classes are tied with their ability to detect falls (detection) and the alert of their occurrence (communication), whereas A classes are dedicated to other low risk tasks. Nevertheless, the standard allows the reclassification of C to B-class (see right column in Table 1 between parentheses), if and only if the user is provided with additional hardware resources, which are defined as hardware risk control in 62304. To that aim, the user of the FDS has always the option to press a button of the PSE in case the fall detection fails. Moreover, class-B software modules have been implemented in different microcontrollers and isolated memory allocations in order to maximize system safety and minimize risks. In particular, the communication module is executed both in the System-on-Chip (SoC) embedded in the Zigbee and the GSM/GPRS peripherals of the PSE. In the third place, the CPU and Power Management Module are controlled by the DSP, where the class-B algorithm for fall detection is executed and stored. All these technologies have been chosen because of their proven robustness and QoS. Despite this, if data connection is lost or in case of leakage of battery, which are critical issues in any mHealth system, it is followed the ISO IEC/CD 60601-1-8 recommendations [27]. Thus, it warns the user that the PSE must be connected to an appropriate power source, and/or alerts that the corresponding fall alarm will be sent as soon the connection is recovered. This standard specifies the requirements for systems and alarm signals within a "distributed alarm system" for patient care, and determines the status of an "Alarm System", distinguishing between clinical (fall event) and technical alarms (power failure). The standard also provides the guidance for the application of these alarm systems, through the definition of: alarm categories, priorities, urgency, and control states.

3.3 Fall detection algorithm

The algorithm used by the FDS has five set of functions, which perform the basic tasks of the algorithm [28]: signal preprocessing, calculation of key parameters and decision about the event occurred. The processing techniques employed have been simplified so as not to overload the processing capacity of the PSE. Besides, the algorithm has been designed to be remotely personalized through various threshold optimization techniques so as to minimize the number of false positives. For this purpose, it is employed a distributed processing architecture [20] that explicitly integrates capabilities for its continuous adaptation to the medium, the context, and the user. In addition, the development and initial evaluation of algorithm performance is significantly eased in a controlled environment like a mathematic IDE, with which preliminary bugs are quickly debugged and corrected. This way, taking into account the methodology described in Section II and the HW of the PSE, the procedure consists of the following steps (see Figure 4):

- Design and development using a mathematic IDE (*e.g.*, Matlab™).
- Adaptation of Matlab code to Embedded Matlab. Optionally, generation of intermediate code (*e.g.*, C-MEX functions) for a first evaluation of the algorithm in Matlab.
- Conversion to C-code using Real-Time Workshop (RTW).
- Debugging, optimization and verification of the generated C-code with a generic compiler.
- Adaptation of the C-code through an embedded IDE (*e.g.*, Code Composer Studio (CCS) from Texas Instruments.

The authors of the paper have chosen Matlab as Mathematical IDE due to its worldwide adoption and comprehensive set of toolboxes. Nevertheless there are other emerging options available, based on Python language, which show great

performance among other interesting features that make it a great choice for embedded software developers [29]. In any case, the methodologies proposed are independent of the solution adopted and can be easily adapted to the particularities of the IDE selected. In the particular case of Matlab, they led to a five-step approach where it is tested each task alone, in a modular basis and in conjunction, as stated in the 62304 standard. Its correct operation was firstly verified in Matlab and the obtained results were taken as a reference for the rest of the validation procedure. Anyway, the verification is implicit in every one of these stages, since it follows agile methodologies. This way, the results obtained in the successive stages are compared with those validated by simulation using Matlab environment. Besides, agile approach has speeded up the development by cyclically evaluating the results compared to the set of requirements and QoS metrics defined in the methodology section. This iterative process is easily performed considering current state-of-art of IDE, which allows a rapid development by minimizing the burden for the embedded software developer. This process has allowed the optimization of the co-development of the hardware and software of the PSE.

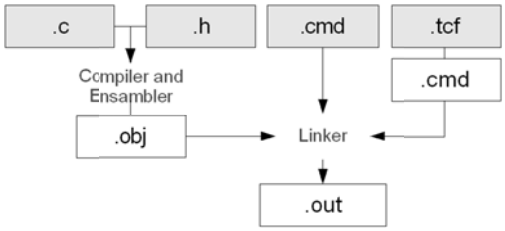

Figure 5. Generation of the executable file

In parallel, each major function has been converted independently. This way, RTW creates a specific file that contains all C header files and facilitates the debugging of each function in a platform-independent C language build environment. Therefore, portable C-code software has been obtained, so that the application can be executed in various embedded systems from lighter platforms. The final step is dedicated to algorithm adaptation to the particular characteristics of the PSE through the CCS IDE. Thus, the memory map and debugging tasks have been defined using the configuration file generated by DSP/BIOS (.tfc/.cmd in Figure 5) in order to optimize the real-time performance of the software module. Specifically, the additional configuration file (.cmd) defines memory sections, one of which has been used to place the application code. Finally, CCS allowed the compilation of source code, along with the header files and settings to generate the executable file.

3.4 Evaluation

The evaluation of the PSE has been performed from two perspectives. First, a real-time validation that evaluates FDS performance compared to other solutions available. Second, a test-bed validation that analyzes other technical issues related to wireless MDs.

3.4.1 Real-time performance

A set of experimental data captured by the IAU has been used in order to validate the real-time processing algorithm. The aim was to check that the algorithm embedded in the PSE, as well as in other platforms, maintains the sensitivity (100%) and specificity (96.15%) of the original algorithm previously simulated in the Mathematical IDE, in order to avoid any new error introduced in the process of software optimization. This way, and in order to assess the goodness and correct performance of the embedded software to different platforms, the code of the fall detection algorithm has been compiled and executed through open-source IDEs for C programming on several CPUs from the following platforms: an average notebook, an Android-based Smart TV-BOX, and an open-source hardware embedded board. The comparison of the main outcomes achieved is summarized in Table 2. In the first place, the main conclusion obtained was that the methodological development approach permitted to successfully compile and execute the embedded FDS in several platforms with different features. Besides, the needed programmer person-hours in order to fulfill these tasks were 2 hours in average for every platform, once evaluated in Matlab. This amount of time, and consequently the associated personnel costs of development, would have been significantly higher if the algorithm had been developed from scratch for each platform, thus confirming the viability of the methodological techniques used and encouraging developers to follow this approach in other mHealth projects.

Table 2. Comparative analysis of features of the fall detection algorithm embedded in different platforms

Platform	Average NoteBook	Smart TV BOX	Embedded Board	PSE
SoC/Core	Intel Core2 Duo/Dell Precision M4400	ALLWINER A107/ARM CORTEX A8	PIC32MX440F256 / MIPS M4k	TMS320 C6727 / C6000
OS	Windows XP	Android 4.1	Custom	DSP/BIOS
Clock	2.53GHz	1.2GHz	80MHz	300MHz
Compiler	Bloodshed Dev-C++	JNI-NDK GCC 4.6	GCC 4.3	CCS4
Deadline	31 ms	10 ms	11.2 s	92.85 ms
Memory	69 Kb	13,1 KB	126,12 KB	67KB
BOM	$443-$762	$65*	$7**	$45

* BOM, which refers to bill of materials, does not include a graphical user interface; ** BOM includes only the SoC and the clock

Then, the real-time performance of the embedded processing module has been estimated using the Real-Time Analysis tools of CCS. For this purpose, the event log of CCS was used to assess the expected results from the input data. As a result, the software modules required for fall detection require 27854944 CPU cycles to complete its execution, which corresponds to 92.85 ms at the DSP clock frequency (300 MHz), as shown in Table 2. The difference in the time deadline of the algorithm that is executed on a Notebook compared with the PSE is mainly due to the more limited processing and RAM memory resources of the embedded system. Nevertheless, the development process followed has allowed ensuring that the deadline of the fall detection algorithm, and in particular for the PSE, is within the a-priori set goals, which corresponds to a soft real-time embedded system. The exception is the results obtained for the Embedded Board, due to a lower frequency clock from a not-floating point processor (PIC32). Finally, the 67 KB binary file obtained represents an occupancy rate of 6.64% of a standard 1MB flash memory, like the PSE memory, thus having a 93.36% available for other processing modules hosted in the PSE.

3.4.2 Test-bed evaluation

Once the correct performance of the algorithm was successfully evaluated in the previous sub-section, the test-bed validation was performed through several sets of experiments. These were developed on thin mats under control in our lab facilities (indoor) and the surrounding area (outdoors) by 31 young and healthy males and females with ages of 28±4 years, weights of 72±14 kg, and heights of 174±8 cm. The aim of these experiments was, in a first place, to confirm the feasibility of alarm transmission and management with robust technologies like Zigbee or GSM/GPRS (see Figure 6).

The PSE, after discriminating impacts sent by the IAU carried by the user, reports potential fall events to the service provider through alarm SMS. To this end, it uses a commercial GSM/GPRS module in order to validate the communications in an outdoor scenario, where the majority of FDS alternatives cannot operate. However, for the final design of the PSE and in order to reduce to the limit the costs of its BOM, shown in the last row of Table 2, an embedded mobile transceiver and antenna must be used instead of the module. This way, and to the best of authors' knowledge, the PSE provides the most optimal and cost-efficient solution for the case of use of human fall detection. Moreover, the costs of the PSE are even lower compared to other portable options to be used in an outdoor scenario, like a Nexus One Smartphone [30]. Finally, the latency of communications was calculated for the Zigbee data-link in order to assess system QoS for the saturated ISM band. For this purpose, the physical layer of the IEEE 802.15.4 standard has been evaluated considering the continuous transmission of data sets from the IAU grouped in frames of 500 ms. Then, the time delay for up to 5 (32 ms), 10 (65 ms) and 15 (77 ms) simultaneous smart sensors attached to the same PSE has been estimated, assuming a worst case scenario of 14% transmission error rate of frames. The results obtained are in all cases lower than 1 second, even adding the processing time delay of the PSE (see "Deadline'" row in Table 2), and considering that just a single IAU is needed for fall detection. Thus, the results obtained meet the up-front design requirements defined in Section 3.A. A deeper analysis of the communication protocol falls out from the scope of this paper [31].

 ISSN 2377-9381 E-ISSN 2377-939X

Figure 6. Picture sequence describing the process of indoor fall detection and sending of the corresponding alarm (left to right; top to bottom)

4 Discussion and conclusions

This paper has discussed the increasing relevance of proper methodologies and development tools to address affordable mHealth, from the learned experiences during the design and development of a FDS. The aim is to share these experiences with the scientific community, in order to deal with these issues from a sustainable perspective regarding current standards and technologies from the IoT ecosystem. This is achieved with a two-level implementing process: first, considering from the design, the capability of porting the software applications to several platforms; second, assessing risks associated with the software development process itself. In this regard, an especially critical application has been studied that gathers the challenges analyzed, so as to assess the strategy followed: human fall detection. To accomplish this goal, agile methodologies and standards like IEC 62304 and IEC 60601 are needed in order to define an iterative software process that minimizes safety and risks involved, while addressing alarms triggered. The approach also takes advantage of available IDE tools for automatic conversion of software, which have facilitated the SW development in a high-level language that eases its installation in several platforms. Besides, the procedures and materials employed have allowed more time for low-level debugging and optimization within the FDS, regarding the specific hardware considerations of different platforms.

In the first place, the correct operation of the fall detection algorithm embedded in the FDS was assessed in terms of specificity and sensitivity, confirming the feasibility of the software development process. Then, the performance of the FDS was evaluated from two perspectives: real-time performance and test-bed verification. The results obtained confirmed that the most optimal device was the PSE in comparison to other platforms (see Table 2): the minimal occupation code of the processing module releases a high percentage of memory for processing other sensors data within the PSE. In addition, the application spends less than one tenth of a second to make a decision about the occurred event, which is a reasonable latency for this soft real-time system. Regarding the test-bed validation, it has been demonstrated the

feasibility of the FDS and the PSE with current and low-cost technologies, for a critical operation like fall detection. QoS metrics have been used so as to analyze the performance of latency communications in the 2.4-GHz ISM band in a hostile communication channel. The results obtained fulfill by far the requirement of 1 second of latency, even considering a 14% of data frames lost. Consequently, the results meet the up-front design requirements for the FDS and strengthen the approach followed for the embedded medical software.

This paper represents some of the challenges involved in the development of affordable mHealth solutions and how to tackle them with available tools and methodologies. Nevertheless, further research is needed, for instance to test the FDS in real-life scenarios with elderly participants, which is a current task of the authors to be detailed in future publications. Besides, it is needed to study how this sustainable development approach of MD may be addressed by healthcare industry with compliance of certification efforts.

Acknowledgements

The authors are grateful to A. Miquel and A. Cóbreces for their helpful assistance in the development of the PSE. This work was supported in part by the CIBER de Bioingeniería, Biomateriales y Nanomedicina (CIBER-BBN) and the intramural Grant PERSONA, DIAB-Support, PLADEBACT and NEUROMON, in part by the Instituto de Salud Carlos III under PI11/00111, and in part by the Dirección General de Investigación, Tecnología y Empresa, Government of Andalucía, under Grants P08-TIC-04069 and TIC6214. CIBER-BBN is an initiative funded by the 6th National R&D&i Plan 2008–2011, Iniciativa Ingenio 2010, Consolider Program, CIBER Actions and financed by the Instituto de Salud Carlos III with assistance from the European Regional Development Fund.

References

[1] Stankovic JA. Research Directions for the Internet of Things. IEEE Internet Things J. 2014 Feb; 1(1): 3-9. http://dx.doi.org/10.1109/JIOT.2014.2312291

[2] Gartner Says 4.9 Billion Connected [Internet]. [cited 2015 Jun 26]. Available from: http://www.gartner.com/newsroom/id/2905717

[3] Internet of Everything | ABI Research [Internet]. [cited 2014 Jan 20]. Available from: https://www.abiresearch.com/research/service/internet-of-everything/

[4] Zhang Y-T, Poon CCY. Health Informatics: Unobtrusive Physiological Measurement Technologies. IEEE J Biomed Health Inform. 2013; 17(5): 893-893. http://dx.doi.org/10.1109/JBHI.2013.2279187

[5] Brown PJ. Software Portability. Encyclopedia of Computer Science [Internet]. Chichester, UK: John Wiley and Sons Ltd.; [cited 2014 Jan 17]. p. 1633-4. Available from: http://dl.acm.org/citation.cfm?id=1074100.1074809

[6] ISO/IEC 25010:2011 - Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models [Internet]. [cited 2015 Jun 26]. Available from: http://www.iso.org/iso/catalogue_detail.htm?csnumber=35733

[7] Systems Design for Remote Healthcare. New York, NY: Springer-Verlag New York Inc.; 2014.

[8] Koltashev A. A Practical Approach to Software Portability Based on Strong Typing and Architectural Stratification. In: Böszörményi L, Schojer P, editors. Modular Programming Languages [Internet]. Springer Berlin Heidelberg; 2003 [cited 2014 Jan 17]. p. 98–101. Available from: http://link.springer.com/chapter/10.1007/978-3-540-45213-3_13. http://dx.doi.org/10.1007/978-3-540-45213-3_13

[9] Paggetti C, Barca CC, Rodríguez JM. System Integration Issues for Next-Generation Remote Healthcare System. In: Maharatna K, Bonfiglio S, editors. Systems Design for Remote Healthcare [Internet]. Springer New York; 2014 [cited 2014 Jan 20]. p. 229-49. Available from: http://link.springer.com/chapter/10.1007/978-1-4614-8842-2_8. http://dx.doi.org/10.1007/978-1-4614-8842-2_8

[10] Wolf M. Improving Clinical Engineering by Web Apps on Mobile Devices. In: Jobbágy Á, editor. 5th European Conference of the International Federation for Medical and Biological Engineering [Internet]. Springer Berlin Heidelberg; 2012 [cited 2014 Jan 17]. p. 693-4. Available from: http://link.springer.com/chapter/10.1007/978-3-642-23508-5_180

[11] Membarth R, Reiche O, Hannig F, *et al*. Code Generation for Embedded Heterogeneous Architectures on Android. Proceedings of the Conference on Design, Automation & Test in Europe [Internet]. 3001 Leuven, Belgium, Belgium: European Design and

Automation Association; 2014 [cited 2015 Jun 26]. p. 86:1-86:6. Available from: http://dl.acm.org/citation.cfm?id=2616606.2616712

[12] Membarth R, Hannig F, Teich J, *et al*. Generating Device-specific GPU Code for Local Operators in Medical Imaging. Parallel Distributed Processing Symposium (IPDPS), 2012 IEEE 26th International; 2012. p. 569-81. http://dx.doi.org/10.1109/ipdps.2012.59

[13] ISO 14971:2007 - Medical devices -- Application of risk management to medical devices [Internet]. [cited 2014 Mar 31]. Available from: http://www.iso.org/iso/catalogue_detail?csnumber=38193

[14] ISO 13485:2003 - Medical devices -- Quality management systems -- Requirements for regulatory purposes [Internet]. [cited 2014 Feb 4]. Available from: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=36786

[15] IEC 62304:2006 - Medical device software -- Software life cycle processes [Internet]. [cited 2014 Mar 31]. Available from: http://www.iso.org/iso/catalogue_detail.htm?csnumber=38421

[16] Dahnoun N, Brand J. Defining a process for rapid processor selection and algorithm development. 2011 7th International Workshop on Systems, Signal Processing and their Applications (WOSSPA); 2011. p. 259-62.

[17] Schmitt C, Kuckuk S, Kostler H, *et al*. An Evaluation of Domain-Specific Language Technologies for Code Generation. 2014 14th International Conference on Computational Science and Its Applications (ICCSA); 2014. p. 18-26.

[18] Perron M, Kamwa I, Dessaint LA. Development of a portable software tool for time domain modal analysis. 2012 11th International Conference on Information Science, Signal Processing and their Applications (ISSPA); 2012. p. 1371-6.

[19] Estudillo-Valderrama MA, Roa LM, Reina-Tosina J, *et al*. Design and Implementation of a Distributed Fall Detection System #x2014; Personal Server. IEEE Trans Inf Technol Biomed. 2009 Nov; 13(6): 874-81. PMid:19775977. http://dx.doi.org/10.1109/TITB.2009.2031316

[20] Naranjo-Hernandez D, Roa LM, Reina-Tosina J, *et al*. Personalization and adaptation to the medium and context in a fall detection system. IEEE Trans Inf Technol Biomed Publ IEEE Eng Med Biol Soc. 2012 Mar; 16(2): 264-71. PMid:22287249. http://dx.doi.org/10.1109/TITB.2012.2185851

[21] E.800: Definitions of terms related to quality of service [Internet]. [cited 2014 Feb 7]. Available from: http://www.itu.int/rec/T-REC-E.800-200809-I/en

[22] Cóbreces Á, Valderrama MAE, Roa LM, *et al*. Multi-Device Information Management for Real-Time Processing of Biomedical Signals. In: Long M, editor. World Congress on Medical Physics and Biomedical Engineering May 26-31, 2012, Beijing, China [Internet]. Springer Berlin Heidelberg; 2013 [cited 2014 Mar 31]. p. 2224-7. Available from: http://link.springer.com/chapter/10.1007/978-3-642-29305-4_584

[23] Oshana R, Kraeling M. Software Engineering for Embedded Systems: Methods, Practical Techniques, and Applications. Newnes; 2013. 1201 p.

[24] Carvalho SC, Motta Cardoso FR, Da Cunha AM, *et al*. A Comparative Research between SCRUM and RUP Using Real Time Embedded Software Development. 2013 Tenth International Conference on Information Technology: New Generations (ITNG). 2013. p. 734-5. http://dx.doi.org/10.1109/ITNG.2013.112

[25] DSP/BIOS Real-Time Operating System (RTOS) - DSPBIOS - TI Software Folder [Internet]. [cited 2014 Mar 31]. Available from: http://www.ti.com/tool/dspbios

[26] Powering the TMS320C6742, TMS320C6746, and TMS320C6748 with the TPS650061 - slva490.pdf [Internet]. [cited 2014 Mar 31]. Available from: http://www.ti.com/lit/an/slva490/slva490.pdf

[27] IEC/CD 60601-1-8 - Medical electrical equipment -- Part 1-8: General requirements for basic safety and essential performance -- Collateral standard: General requirements, tests and guidance for alarm systems in medical electrical equipment and medical electrical systems [Internet]. [cited 2014 Mar 11]. Available from: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=61754

[28] Estudillo-Valderrama MA, Roa LM, Reina-Tosina J, *et al*. A proposal of a fall detection algorithm for a Multidevice Personal Intelligent Platform. 8th IEEE International Conference on BioInformatics and BioEngineering, 2008 BIBE 2008; 2008. p. 1-4.

[29] Python *vs.* Matlab — Pyzo - Python to the people [Internet]. [cited 2015 Jul 31]. Available from: http://www.pyzo.org/python_vs_matlab.html

[30] Google Nexus One Carries $174.15 Materials Cost, iSuppli Teardown Reveals - IHS Technology [Internet]. [cited 2014 Mar 31]. Available from: https://technology.ihs.com/389033/google-nexus-one-carries-17415-materials-cost-isuppli-teardown-reveals

[31] Naranjo-Hernandez D, Roa LM, Reina-Tosina J, *et al*. SoM: A Smart Sensor for Human Activity Monitoring and Assisted Healthy Ageing. IEEE Trans Biomed Eng. 2012 Nov; 59(11): 3177-84. PMid:23086195. http://dx.doi.org/10.1109/TBME.2012.2206384