Optimization Proposal of a Standard-Based Patient Monitoring Platform for Ubiquitous Environments

I. Martínez, J. Escayola, I. Fernández de Bobadilla, M. Martínez-Espronceda, L. Serrano, Senior Member IEEE, J. Trigo, S. Led, and J. García, Member IEEE

Abstract—This article describes the optimization of a patient telemonitoring platform based on the ISO/IEEE11073 (X73) standard to enable medical device interoperability. In order to achieve this, principal advantages and remaining improvements are evaluated to include in further upgrades towards the new profile evolution, oriented to ubiquitous environments and wearable devices (Personal Health Devices, X73-PHD), and opened to additional plug-and-play features and remote management. After evaluating the possibilities, we describe the platform porting process, a required step to adapt it to the new functionalities and allowing the development of end-to-end standard based systems. The paper details the implementation of the agent-manager architecture, particularized on the X73-PHD communication protocol between a Medical Device (MD) and a central gateway (Compute Engine, CE). Lastly, the obtained results are evaluated, oriented to constitute an X73-PHD tester to prove the challenges currently under discussion in the European Standardization Committee (CEN).

ISO/IEEE11073 STANDARD BASIC NOMENCLATURE

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BER/DER</td>
<td>Basic/Distinguished Encoding Rules</td>
</tr>
<tr>
<td>PER/MDER</td>
<td>Packet/Medical Devices Encoding Rules</td>
</tr>
<tr>
<td>CE</td>
<td>Computer Engine</td>
</tr>
<tr>
<td>DIM</td>
<td>Domain Information Model</td>
</tr>
<tr>
<td>FSM</td>
<td>Finite State Machine</td>
</tr>
<tr>
<td>MD</td>
<td>Medical Device</td>
</tr>
<tr>
<td>MDBG</td>
<td>Medical Data Information Base</td>
</tr>
<tr>
<td>MDAP/MDDL</td>
<td>Medical Device Application Profile/Data Language</td>
</tr>
<tr>
<td>X73-PoC/PHD</td>
<td>X73-Point of Care/Personal Health Device</td>
</tr>
</tbody>
</table>

I. INTRODUCTION

Along the 90’s decade, telemonitoring services were aimed to patient control in hospitals, especially those whose vital signs had to be continuously under supervision and were allocated at the Intensive Care Unit (ICU) [1]. Each manufacturer developed medical systems with proprietary specifications thus complicating the incorporation of these devices in e-Health systems, which required of a homogeneous implementation to achieve a global and ubiquitous (u-Health) solution.

With this evolution towards the u-Health, the patient becomes the center of the entire medical care system. Technologies for ubiquitous computing, through Personal and Body Area Networks (PAN/BAN) emerge as a need for the new telemonitoring scenarios.

Interoperability between different medical systems, in terms of protocol standardization becomes, then, a main requirement to progress towards e-Health [3]. This is a long process that has been promoted by several organizations, like the European Standardization Committee (CEN) through its Technical Committee 251 (TC251) [4]. From CEN/TC251, new standards, the focus of our work, are being developed: ISO/IEEE11073 (X73) family of standards on its first version for medical device interoperability at the Point-Of-Care (X73-PoC) [5], as well as its most recent evolution oriented to BAN/PAN and wearable systems to be used over Personal Health Devices (X73-PHD) [6].

There are previous contributions [7], developed in the USA from the research group headed by Dr. Warren, but neither European contributions in this field nor proposed end-to-end solutions to cover new use cases for ubiquitous patient monitoring and design-oriented to be compatible with the new standard version X73-PHD, as it is introduced in this article. In this article, taking previous developments as a start point [8] in which a first end-to-end implementation based on standards (X73 and EN13606) was implemented, we show an optimization progress on that platform.

The new architecture has to provide a solution to those ubiquitous scenarios proposed by the X73-PHD standard, incorporating necessary changes at the agent-manager communication model, redefining the state machine for connection stage management, and designing new transport and physical layers. It is necessary then to develop an architecture which ensures its portability to other environments and use cases (geriatric services, athlete training monitoring, mobile scenarios, etc.), ease its remote management (medical information, alarms, operating scheme, etc.), add new technologies (Bluetooth, ZigBee, RFID), and propose new wireless devices (PDAs, SmartPhones, microcontrollers, etc.).

Section II describes the X73 evolution and the platform optimization. Section III shows the suggested design, its architecture and its implementation progress. Section IV evaluates the results and new functionality oriented to X73-PHD. Conclusions are commented in Section V.
II. X73 EVOLUTION AND PLATFORM OPTIMIZATION

The development of new personal and wearable devices have brought X73 to an optimized version, X73-PHD, which describes the landscape of transport-independent applications and information profiles for personal telehealth. These profiles define data exchange/representation and terminology for communication between Medical Devices (MDs, as glucometer, thermometer, etc.), and Compute Engines (CEs, as cell phones, personal computers, etc.). In X73-PHD communication model, MD and CE are named as agent and manager, respectively. As shown in Fig. 1, the architecture is divided into three main protocol blocks:

- **Device Specializations.** A set of model descriptions which collects the total of objects and attributes related to the device components, like an overall system’s configuration (Medical Device System, MDS), Persistent Metric (PM-Store and Segments) or Metric Specifications. New MDs are continuously being added, by developing its MDS.

- **Optimized Exchange Protocol.** The main part of the standard consists of a medical and technical terminology framework (Domain Information Model, DIM) which will be encapsulated inside the Protocol Data Unit (PDU). The communication model describes a point-to-point connection based on manager-agent architecture through a Finite State Machine (FSM). The first version of X73 defined this part as the Medical Device Data Language (MDDL). Next, a Service Model defines a set of messages and instructions to retrieve data from the agent based on the DIM. In addition to this, it provides a data conversion from an Abstract Syntax Notation (ASN.1) to a Transfer Syntax, using an optimized Encoding Rules (ER) denoted as Medical Device ER (MDER), as well as standard Binary ER (BER) and even more effective Packet ER (PER) support. Service Elements (SE) taken from the previous X73 version for this purpose are: Remote Operation (ROSE, Optimized for MDER), Association Control (ACSE) and Common Management Information (CMISE).

- **Transport Layer.** Data transmission will be held over a transport technology caused by X73-PHD identifies assumptions that require direct support by this layer, allowing various transports to be implemented (X73-PoC established higher dependency between transport and upper and lower layers). Thus, transport specifications are out of the scope of the X73-PHD standard, while other Special Interest Groups (SIG) are working towards profile definition for Bluetooth, USB, etc.

Following this evolution, migration from the previous platform [8] (known as platform1.0-α) to a new platform1.5-β was necessary. A snapshot of the overall process is shown in Fig. 2. Main objectives for these challenges are:

- Use case evolution from fixed systems to mobile, and the addition of plug-and-play or hot-swap features, forced to move from IrDA or RS-232 to new technologies like USB or Bluetooth.

- The new MDs, because of upgrading from X73-PoC to X73-PHD, required an important change in the communication scheme with the CE, which implies a new FSM design. This new FSM is somehow too complex to be implemented in the platform1.0-α due to new methods that need to be incorporated.

- The new X73-PHD base standard does not specify a concrete transport level. Nevertheless, platform1.0-α introduces a specific complex model to add modularity to the implementation. The Optimized Exchange Protocol replaces the previous X73 protocols to define PDUs.

- X73 adapter evolution towards more versatile systems able to be used in microcontroller-based systems, wireless CEs, new use cases (from fixed to ubiquitous), and multimedia environments of Graphic User Interfaces (GUIs) that require a simpler high-level encapsulation (platform1.0-α is too complex at the low level development).

From this basis, the requirements that will state the design specifications for the new implementation will be evaluated.
III. PROPOSAL OF DESIGN AND IMPLEMENTATION

After the technical analysis that justifies the platform migration; the solution architecture, design rules, and implementation development are described below.

A. Migration architecture. Design rules

The proposed new architecture has to cover a group of main commitments or design guidelines.

First, it should look for a functional design in compliance with X73. To do so, the dependence on transport technology has to be removed by looking for a generic and configurable solution: transport layer manager or handler. Thus, the acquired data are first translated to X73, and refreshed with every new measurement (but in an episodic way, not periodically as in platform1.0-α). Then, the data transmission is initiated (already X73-compliant) from the previous X73-adapter to CE, under user request. Therefore, it is up to the developer to include the files capable to give support to the appropriate transport technology for each MD.

Secondly, the design of generic X73 stack has to be kept. However, the code has to be optimized and the definitions library ASN1.C (ASNX for X73) has to be introduced in order to successfully connect in the communication link from the development environment for each resource. The abstract classes for MD and CE have also to be modified to implement the new FSM defined by CEN for X73-PHD.

Finally, since no transport technology is incorporated, the data encapsulated through the different layers become into a layout based on a structure of buffers that collect the set of PDUs of the different stack layers. These buffers have to be properly managed so that they can response to the communications protocol designed by X73.

---

B. Solution implementation

First, and due to the need of a generic transport layer, the communication between both ends at that level is up to the developer. In the platform1.5-β, the new generic layer TRANSPORTS (Fig. 3) is included. The communication with the session layer is carried out through the generic stack interface for each end (MD and CE), and with the physical layer through a system of buffers. This function is implemented by means of a generic transport layer manager interface (handler), transparently to the protocol.

Second, it is essential that the platform be X73-compliant in all its versions, following rigorously the design of the new state machines FSM at both ends of the system: agent (MD) and manager (CE). That design philosophy has been a key issue in the design, along with the implementation of the states carried out by the communication MD/CE in both stack models. A continuous loop has been created in order to design these machines. That loop includes the following states (Fig. 4): DISCONNECTED, CONNECTED, DISASSOCIATED, ASSOCIATED, CONFIGURING, and OPERATING. The operation process is detailed in [6].

Finally, as it occurred with the transport layer, it is up to the developer to implement any physical layer. This physical layer is in charge of receiving and sending the [st_buffer] structures that MD and CE send through the handler. The st_buffer structure is a data container in memory. The bits coded in each PDU of each layer follow this structure, being that way easier to manage.

From all this, platform1.5-β is obtained. This platform controls the bits on the wire (bits that form the st_buffer and that are sent by both stacks through its interface), enabling flow and error control, which is a goal pursued by the CEN and it will enable the system management.

---

Fig. 3. Transport protocol handler for the new X73-PHD platform.

Fig. 4. Generic Finite State Machine for the new X73-PHD platform.
IV. RESULTS AND OPEN POINTS

The platform1.5-β implementation is a testbed model of the X73-PHD-compliant communication. Besides, its forthcoming platform2.0-release, which will include the open points detailed in IV.B, will allow the platform to become a solution transferable to the health system.

A. X73-PHD tester

The X73-PHD tester starts asking the user which MD wants to be used from an available list. After the selection of MD, its information, the measures types that can be acquired, and a menu to control the FSM are shown. FSM goes through the ends of the X73-PHD communication (MD-CE). From here, MD is initiated and the stack layers, operating interfaces and Medical Data Information Base (MDIB) structure are created: MDS object and its sub-branches.

Later, the transport system that supports the communication is required, getting the handler ready to support corresponding protocols. Moreover, information of the execution is shown in the screen, helping the engineer to know the methods of the layers. It is also shown how the buffers send the X73-PHD information and the other configuration parameters of the events and responses interchanged between MD and CE.

After association, MD enters to the configuration point; CE sends the MDS object to the MD, without measurements yet. In CE a context of data reception is created (episodic or polling, regarding MD model). Thus, MD is now ready for the measurement acquisition (always under user request), and enters into the OPERATING state of the FSM.

By submitting data, MD updates the MDS object with the acquired measurements, and sends them to CE to be also updated. The received measurements are shown, detailing the X73-compliant identifications (in this example: 19230, 19229 and 18442; corresponding to a blood pressure device: diastolic pressure, systolic pressure and pulse, respectively), as shown in Fig. 5. Finally, it is asked if more measurements are going to be made or, on the contrary, a menu is used to dissociate MD and CE or disconnect them regarding FSM.

B. Open points: platform2.0-release

As told before, some technical advances remain opened to achieve the definitive version of the platform:

- Implementing each stack in a microcontroller-based adapter inside a MD and in a wireless CE, with their transport and physical layers technologies (TCP/IP, USB, Bluetooth, etc.) in the handler. Besides, designing private methods that strengthen the management by means of sockets and threads, and incorporate priority models to attend certain MDs (under consideration by CEN).
- Supporting the multiple MDs connection with one or multiple CEs, optimizing the creation and management of the different MDIBs, and implementing a state manager of FSM (to read MD configuration parameters and add them to a database to guarantee the plug-and-play features).
- Migrating the console interface to an interactive GUI, adaptable to the device type (miniPC, mobile, Smartphone, PDA, etc.), interface (Java, .Net, Web 2.0, etc.), or OS (Windows Mobile, Android, Symbian, etc.).

V. CONCLUSION

The evolution of X73-PoC to X73-PHD has derived to an optimization of the end-to-end platform, allowing to achieve an ubiquitous, plug-and-play and standard-based solution. Besides, it can be seen as a X73-PHD demonstrator to prove the challenges currently under discussion in the CEN: flow and errors control, errors and alarms management, multiple MD connection with one or multiple CEs, or implantation in micro-controllers, wearable and wireless devices.

ACKNOWLEDGMENT

The authors wish to thank X73-PHD Working Group, CEN, and Mr. Melvin Reynolds, convenor of the CEN TC251 WGIV, for the contributions to this research. We appreciate the contribution of Miguel Galarraga to the excellent results carried out during the last years, and also Adolfo Muñoz (Instituto de Salud Carlos III, Paula de Toledo (Univ. Carlos III), and Silvia Jiménez (Univ. Politécnica de Madrid).

REFERENCES


Fig. 5. X73-PHD demonstrator. X73 measurement sending from MD to CE
Implementing ISO/IEEE 11073: Proposal of two different strategic approaches

M. Martinez-Espronceda, L. Serrano, Senior Member IEEE, I. Martinez, J. Escayola, S. Led, J. Trigo, and J. Garcia, Member IEEE

Abstract—This paper explains the challenges encountered during the ISO/IEEE 11073 standard implementation process. The complexity of the standard and the consequent heavy requirements, which have not encouraged software engineers to adopt the standard. The developing complexity evaluation drives us to propose two possible implementation strategies that cover almost all possible use cases and cases handling the standard by non-expert users. The first one is focused on medical devices (MD) and proposes a low-memory and low-processor usage technique. It is based on message patterns that allow simple functions to generate ISO/IEEE 11073 messages and to process them easily. In this way a framework for MDs can be obtained. Second one is focused on more powerful machines such as data loggers or gateways (aka. computer engines (CE)), which do not have the MDs’ memory and processor usage constraints. For CE a more intelligent and adaptive Plug&Play (P&P) solution is provided. It consists on a general platform that can access to any device supported by the standard. Combining both strategies will cut developing time for applications based on ISO/IEEE 11073.

I. INTRODUCTION

TELEMEDICINE services improve considerably patient’s quality of life and reduce health system’s waiting lists offering a better service. The paradigm that proposes, based on Information Technologies (IT), is motivating the appearing of new health service models empowering the patient [1]. A broad field in telemedicine is telemonitorization systems, which focus mainly on elderly, chronic, under palliative care or patients that have undergone surgery. These systems’ key points are a right operation confidence, ergonomics and P&P capabilities, which must be addressed using standardization in order to be effective. Most mature standard in this way is ISO/IEEE 11073 (11073) [2-5], which was initially aimed to be implemented at Point of Care Medical Device Communications (PoCMDC) in Intensive Care Units (ICU) and is currently being the center of a continuous synergy of standardization groups, universities and manufacturers in order to develop and adapt it to the future challenging requirements. One of the most active standard’s developments is being carried out by the Personal Health Data group (PHD) [6] that aims to define a lightweight profile adapted to personal health devices compliant with 11073. Drawbacks of 11073 are its inherent complexity, the time needed by Medical Device (MD) developers to learn it, the hardware requirements imposed (which usually limit the type of platform used), the restriction to serial port (RS-232) or Infrared Data Association (IrDA) transports, and the lack of both testing and 11073-conformance tools. Efforts are being done in order to adapt 11073 to other transports [7,8], such as Bluetooth [9], Ethernet, Internet Protocol (IP), WiFi [10,11], and, in the near future, USB, and ZigBee [12].

The 11073 standard defines completely interchanged messages and sequence diagrams in a point-to-point MD communication and distinguish two roles in that communication: agent, which owns data that are to be transferred, and manager, which accepts and process the agent’s data. 11073 also defines three actors in a ICU or telemonitoring scenario: the Medical Device (MD), which collects data from the patient, the Computer Engine (CE), which receives the data from MDs and post process it, and the Monitoring Server (MS), which receives all the post processed data from all CEs. Fig. 1 shows some of these scenarios. More details can be seen in [13,14].

Our paper aims to meet the 11073 and manufacturers requirements providing two proposals that allow 11073 implementations in low-requirements and with a short software development time.

The paper is organized as follows. Section II presents an overview of the two implementation approaches as a proof of concept for obtaining a trade-off between 11073 requirements and time-to-market MDs implementation. These concepts are then applied in Section III to elaborate a framework for MDs implementation, and in Section IV to propose to a general CE implementation that could be used to implement a mobile phone gateway, a MS in a data center, or ICU central monitoring equipment. Finally, Section V concludes the paper.

II. IDENTIFICATION OF IMPLEMENTATION APPROACHES

According to 11073 implementation experiences [15,16], two implementation contexts covering most of hardware
possibilities are distinguished. These are, firstly, a simple MD system that samples data, and secondly, a CE that access to the MD’s data. A description is given as follows:

A. MD Systems

The proposed implementation strategy is carried out on a simple MD system that collects data from a patient in a telemonitoring service or an ICU scenario, and sends the results to the CE, which can be an elemental gateway or a more complex system. This device is usually mobile or wearable so its ergonomics is fundamental, which means that both size and weight must be the smallest possible, and autonomy the largest. That, in turn, implies low processor usage, and low memory capacity. Usually, typical features of a microcontroller used in medical applications are a few Kilobytes of Random Access Memory (RAM), a few tens of Kilobytes of non-volatile solid-state memory (typically flash or Read-Only Memory, ROM), and a few Megaflops processor. As far as software is concerned, this type of devices do not require a high grade of intelligence and code sources are usually written in assembler, C, or embedded C++. Operating System (OS), in case it was used, is platform dependent and its API differs strongly from some devices to others.

B. CE systems

These systems have generally more capabilities, and are used either as simple gateway machines (such as mobile phones or Personal Digital Assistant (PDA)), or more complex monitoring equipment offering real time data. The key feature required in these devices is a relatively high degree of intelligence. Usually, a unique CE must collect information from several MDs. Moreover, in a typical ICU scenario or a telemonitoring service, P&P capabilities are strongly required, so that a new MD can be connected to the CE and used by the assistance staff without having to update or manipulate the software in the CE. These devices are usually provided with an OS, such as all flavors of Windows, Symbian or Linux, and sometimes with a Virtual Machine (VM), as JAVA.

III. PROPOSED SOLUTION FOR SIMPLE MDS SYSTEMS

This proposal is based on the idea of patterns. A pattern is a model that can be used as an archetype to produce other equal. In this case this concept is applied to messages interchanged between a specific MD and its CE. Doing an exhaustive study of these messages, one can come to the conclusion that they can be grouped so that those in the same group are almost equal and only differ in a few bytes. These similarities appear because in each transaction one or more ASN1 structures are transmitted and its attributes are nearly always the same. Blocks of bytes that do not differ can be extracted for a specific type of message, following the 11073 standard. We call pattern to each of them. The resulting patterns are stored in a pattern library and duplicated ones are discharged in order to reduce memory consumption. The pattern library must be generated only once for each MD specialization.

Once patterns are determined, each of the messages interchanged between MDs and its CE can be reproduced with minimum task processing, as it is shown in Fig. 2 (a). The message of interest is filled with blocks from patterns library, and a few program variables, such as Invoke-Id or an ObservedValue (obtained, for example, from the last A/D converter sample). Messages generated in this way can be compared with received ones, or transmitted. Each of these
transactions usually entails a state change in the Finite State Machine (FSM), modifications in some of the Domain Information Model (DIM) objects, and/or execution of some actions.

A MD developer usually needs to design both hardware and software. The board is typically includes a microcontroller, a communication module, and a sensor. There are also System-on-Chip (SoC) modules that integrate all these components in an only chip [17]. Once the hardware has been selected, developers must develop the software. The software framework for such a task is commonly determined by the hardware involved. For example in the Bluetooth or ZigBee case, the framework is determined by the communications stack that usually provides a proprietary Real Time Operating System (RTOS) and a proprietary Application Programming Interface (API) to access the RTOS functions.

The variability of the MD system requires a great adaptability of the development framework in order to succeed in its adoption by MD manufacturers. Our proposal, showed in Fig. 2 (b), is a generic multiplatform architecture. It is composed by an X73 kernel and a pattern library. The X73 kernel is a small module (usually written in C language) that manages common signals required by the generic 11073 stack, and processes the patterns according to 11073 to produce the messages that, then, the X73 kernel compares with received ones or transmits. It also manages the state of the FSM, the state of objects in the DIM, and system signals. The signals managed include data sent or received signals, connection established, connection lost, timer signals for scanners (such as PeriCfgScanner), etc. Both the X73 kernel and the pattern library are coded for each pair of specific MD specialization - MD communication profile. For instance, in a weigh scale MD specialization, there should be a specific implementation for polling, baseline, and PHD profiles.

Untying these MD specialization - MD communication pairs gives great optimization in memory space. More important is that this solution requires low non-volatile memory usage with a strongly reduced consumption of volatile memory. Consequently, the low memory consumption makes possible to use the same X73-kernel and library of patterns in a big number of MDs that, at the same time, allows sharing it between different manufacturers improving interoperability, since all of them work with the same code.

From the novel 11073 implementer’s viewpoint, this framework, once developed, offers a Rapid Application Development environment (RAD) that will allow him to develop a MD with only a basic knowledge of 11073. The needed modules to be developed are the adaptation layer and the device drivers. The adaptation layer leads X73 kernel’s calls to the lower transport layer and interrupts and signals from this transport layer back to the X73-kernel. The device drivers produce signals leaded to the X73-kernel, and provide with some services such as data access. For example, in the case of a heart rate monitor, device drivers must provide with a signal that inform the X73-kernel when a new sample is ready, and a method to access that sample. In order to maximize interoperability it is preferable that all manufactures use the same framework in their final versions. Moreover, there must be only one framework implementation to be consistent and it must be shared between developers. To create this framework, the help of 11073 world experts, experiences of other implementations, and the expertise of other Special Interest Groups (SIG) is fundamental.

IV. PROPOSAL OF A SOLUTION FOR CES

This proposal has been partially implemented with success in [15]. It is based on both advanced Object Oriented Analysis and Design (OOA/D), and a dual model of information and knowledge, which separates data and its
meaning providing methods to access data in a transparent way. Fig. 3 shows the software architecture. Developers only need to develop drivers able to manage 11073 framework signals and to unleash processes in the 11073 framework. Examples of driver functionality in an ICU monitor could be to create a dialog window and show incoming data or an alarm, to create a control to change a device configuration and adjust thresholds, to show connected devices and its alarm, to create a control to change a device configuration and adjust thresholds, to show connected devices and its alarm, to create a control to change a device configuration and adjust thresholds, to show connected devices and its alarm. Drivers are also in charge of 11073 framework initialization tasks, such as establishing active interfaces, establishing the default route in a gateway, etc. This approach shows the following two main points: 1) the framework could be shared between the manufacturers with a consistent contribution to interoperability, and, also, 2) reduce time-to-market since developers only need to learn 11073 basics to be able to use 11073 framework’s API.

V. CONCLUSION

Implementing 11073 is not a trivial task. Instead, it is a complex process that requires a big effort for a novel 11073 developer. This paper could be used as a guide for those. It explains the two main implementation strategies encountered in medical monitoring systems and proposes approaches for them. The first approach is for MDs, which have a reduced capacity, but are customized for a specific functionality. A framework based on patterns is proposed. This allows the use of the same code in different platforms without modification. The second approach is for CEs, more capable devices with more intelligence, but not specialized. A dynamic, adaptable and P&P implementation framework is proposed for them. It is expected that both proposals, once implemented, reduce considerably the 11073 software development time and hence, facilitate the adoption of the standard.

ACKNOWLEDGMENT

The authors wish to thank Mr. Melvin Reynolds, convenor of the CEN TC251 WG14, for his key suggestions to this research. We also appreciate the contribution of Miguel Galarraga to the excellent results carried out during the last years.

REFERENCES