IEEE TRANSACTIONS ON

INFORMATION TECHNOLOGY IN BIOMEDICINE

A PUBLICATION OF THE IEEE ENGINEERING IN MEDICINE AND BIOLOGY SOCIETY

EMB

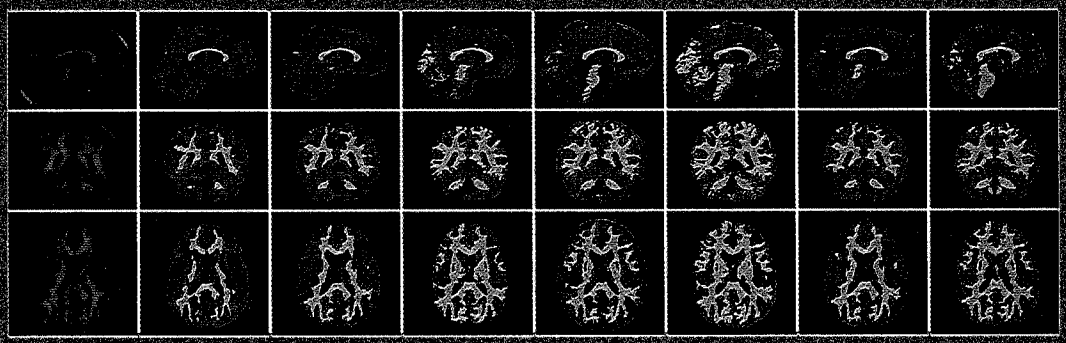
TECHNICALLY COSPONSORED BY THE IEEE COMPUTER SOCIETY

IEEE COMPUTER SOCIETY

Indexed in PubMed® and MEDLINE®, products of the United States National Library of Medicine

PubMed MEDLINE U.S. National Library of Medicine

MAY 2011 VOLUME 15 NUMBER 3 ITIBFX (ISSN 1089-7771)



High-resolution T1-weighted brain MR study with index "IBSR13" (first column), its segmentation results obtained by applying the EM algorithm (second column), VEM algorithm (third column), GA-EM algorithm (fourth column), SPM8 algorithm (fifth column), FSL algorithm (sixth column) and proposed GA-VEM algorithm (seventh column), and the ground truth (eighth column). As shown in "Hybrid Genetic and Variational Expectation-Maximization Algorithm for Gaussian-Mixture-Model-Based Brain MR Image Segmentation" by Tian *et al.*, p. 377.

IEEE

IEEE TRANSACTIONS ON

INFORMATION TECHNOLOGY IN BIOMEDICINE

MAY 2011 VOLUME 15 NUMBER 3 ITIBFX (ISSN 1089-7771)

PAPERS

IEEE

Implementation Methodology for Interoperable Personal Health Devices with Low-Voltage Low-Power Constraints

Miguel Martínez-Espronceda, *Student Member, IEEE,* Ignacio Martínez, Luis Serrano, *Senior Member, IEEE*, Santiago Led, Jesús Daniel Trigo, Asier Marzo, Javier Escayola, and José García, *Member, IEEE*

***Abstract*—Traditionally, e-Health solutions were located at the Point of Care (PoC), while the new ubiquitous user-centered paradigm draws on standard-based Personal Health Devices (PHD). Such devices place strict constraints on computation and battery efficiency, which encouraged the ISO/IEEE11073 (X73) standard for medical devices to evolve from X73PoC to X73PHD. In this context, Low-Voltage Low-Power (LV-LP) technologies meet the restrictions of X73PHD-compliant devices. Since X73PHD does not approach the software architecture, the accomplishment of an efficient design falls directly on the software developer. Therefore, computational and battery performance of such LV-LP-constrained devices can even be outperformed through an efficient X73PHD implementation design. In this context, this paper proposes a new methodology to implement X73PHD into microcontroller-based platforms with LV-LP constraints. Such implementation methodology has been developed through a patterns-based approach and applied to a number of X73PHD-compliant agents (including weighing scale, blood pressure monitor and thermometer specializations) and microprocessor architectures (8, 16 and 32 bits) as a proof-of-concept. As a reference, the results obtained in the weighing scale guarantee all features of X73PHD running over a microcontroller architecture based on ARM7TDMI requiring only 168 bytes of Random Access Memory (RAM) and 2546 bytes of flash memory.**

***Index Terms*—Interoperability, ISO/IEEE11073 standard, Low-Voltage Low-Power, microcontroller architecture, patterns-based design, Personal Health Device.**

ACRONYMS

APDU	Application Protocol Data Unit
ARM	Advanced RISC Machine
CE	Compute Engine
CFM	Core Functionality Module
DIM	Domain Information Model
FSM	Finite State Machine
HCIS	Health Care Information System
ICS	Implementation Conformance Statement
IOAL	Input Output Abstraction Layer
IPC	Inter-Process Communication
LV-LP	Low Voltage-Low Power
MDS	Medical Device System
OS	Operating System
RAM	Random Access Memory
ROM	Read Only Memory
X73	ISO/IEEE11073
X73PHD	ISO/IEEE11073 for Personal Health Devices

This research work has been partially supported by projects TIN-2009-08414 and TIN-2008-00933/TSI from Comisión Interministerial de Ciencia y Tecnología (CICYT) and European Regional Development Fund (ERDF), TSI-020100-2010-277, TSI-020302-2009-89 and TSI-020302-2009-7/Plan Avanza I+D from Ministerio de Industria, Turismo y Comercio, and FPI grant to M.Martínez-Espronceda (Res.1342/2006 Public University of Navarre).

M.Martínez-Espronceda, L. Serrano, S. Led, and A. Marzo are with the Electrical and Electronics Engineering Dept., Public University of Navarre (UPNA), Campus de Arrosadía, s/n 31006 Pamplona, Spain (corresponding e-mail: miguel.martinezdeespronceda@unavarra.es).

I. Martínez, J.D. Trigo, J. Escayola, and J. García are with the Communications Technologies Group (GTC), Aragón Institute of Engineering Research (I3A), University of Zaragoza (UZ), c/María de Luna, 1 50018 Zaragoza, Spain.

This paper has been accepted for publication in a future issue of this journal but has not been fully edited. Content may change prior to final publication.

I. INTRODUCTION

THE fast development of Information and Communication Technologies (ICT) is fostering the transformation of classic healthcare models into new patient-centered environments. Allowing the patient to monitor and report vital signs (e.g. electrocardiographic (ECG) signal, weight, blood pressure, temperature, glucose concentration in blood, etc.) is a key issue in the follow-up of some diseases such as heart failure, Chronic Obstructive Pulmonary Disease (COPD), arterial hypertension (one of the main risk factors for heart diseases, strokes and also a prognostic indicator of renal failure), or obesity, which increases the potential of suffering other illnesses such as diabetes, cholesterol-related illnesses, hypertension, joint diseases, gallbladder malfunction, metabolic syndrome, or coronary and respiratory complications, among others [1]. Such paradigm, so-called patient empowerment [2] changes the conventional role of patients and physicians into a novel scenario where patients play a greater role in their own health and wellness. Furthermore, the e-Health solutions allow the physician to follow the patient's evolution remotely, improving both the efficiency of the overall healthcare systems and the life quality of the patient.

To provide the monitoring process with ubiquity, Personal Health Devices (PHD) may be incorporated to acquire biomedical signals and measurements that can be afterwards gathered by a Compute Engine (CE) [3]. This medical information is then sent by the CE to the Healthcare Information System (HCIS) to be subsequently analyzed by the physicians. A scheme of this architecture is shown in Fig. 1. This architecture includes several PHDs such as weighing scale, blood pressure monitor, glucose meter, thermometer and ECG monitor (Holter

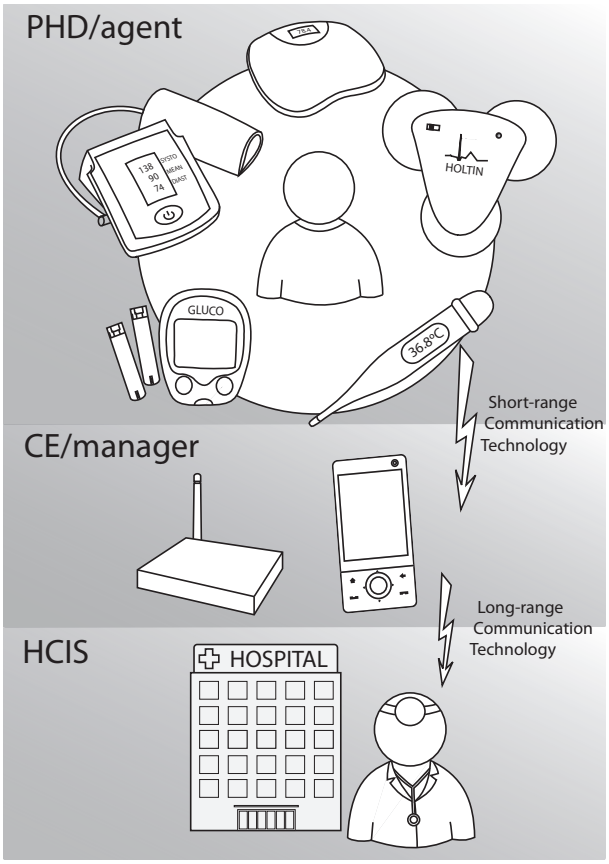


Fig. 1: Architecture scheme for e-Health solutions based on Personal Health Devices.

Intelligent, HOLTIN, for heart failure patient's follow-up [4]), and a CE that could be implemented over a cell phone, a Smartphone, a tablet PC or a set-top-box, among others devices. The communication process between a PHD and a CE follows the agent-manager model and it is supported by short-range communication technologies such as Universal Serial Bus (USB), Bluetooth and ZigBee [5], [6]. The interconnection process between the CE and the HCIS follows the client-server model and it is supported by the long-range communication technologies such as Asymmetric Digital Subscriber Line (ADSL), General Packet Radio System (GPRS) or Wireless Fidelity (WiFi).

As mentioned before, PHDs are essential to assist patients with the follow-up of their diseases in a personalized manner. Ideally, these PHDs should be wearable or portable in order to further ease personal usability and mobility. Moreover, they need to integrate communication protocols that allow the transmission of biomedical data to a CE. The high reliability, ergonomics and accessibility requirements that these PHDs must meet necessitate the optimization of computation and battery efficiency in terms of both software and hardware. Thus, the use of Low-Voltage Low-Power (LV-LP) technologies becomes essential.

Most PHDs use proprietary protocols and are therefore difficult to be integrated in several personal e-Health scenarios. Therefore, using standardization seems to be the best way to solve the problem of interoperability, as has been remarked in [7], [8]. In this context, several protocols have arisen to cover this interoperability gap. One of the most widely known is the ISO/IEEE11073 (X73) family of standards for interoperability of medical device communications. This was initially designed for the Point-of-Care of the patient (X73PoC) [9] and it has recently been extended to PHDs (X73PHD) [10], [11], [12] in order to be supported by emerging transmission technologies as well as LV-LP PHDs with limited capabilities. The use of X73PHD provides plug-and-play capabilities so that any PHD assigned to the patient can be directly replaced without technical knowledge. Thus, the CE should be able to recognize automatically the PHDs related to the patient and self-configure to operate correctly. However, there are some drawbacks in the implementation of X73PHD: its inherent complexity, the time needed to learn and implement it, its need of integration with other standards, the current lack of available tools for developers and the need for restrictive hardware to run it [13]. Furthermore, the use of X73PHD increases computing requirements, which works against the high autonomy that PHDs require. Likewise, LV-LP approaches must be considered as the basis for reducing power consumption of PHDs, both in hardware (LV-LP microcontrollers) and software designs (reduced size algorithms, high usage of processor's LP states, etc.). These software and hardware limitations together with the high degree of reliability needed for the sake of the patient lead to the need for a deeper understanding of how X73PHD works at low computational level and its implications in the workflow for its further implementation into microcontrollers. Taking advantage of this approach, a robust and efficient design for a microcontroller could improve the overall functionality of PHDs.

In this paper, starting from previous implementation results [14] and in collaboration with the European Committee of Standardization (CEN) and the International Organization for Standardization (ISO), a methodology for the design of X73PHD-compliant PHDs based on LV-LP technologies as well as several implementation techniques into microcontrollers are proposed. In Section II, X73PHD features and its specific characteristics related with the elements of the X73PHD architecture (agents and managers) are presented. Both hardware and software constraints are analyzed and the basis of the implementation methodology for X73PHD-compliant LV-LP agents is established. In Section III, the proposed methodology and an appropriate architecture for its further implementation are detailed. In Section IV, the obtained results of the application of the patterns-based methodology to several X73PHD-compliant agents as a proof-of-concept are presented and discussed. Finally, conclusions are drawn in Section V.

II. X73PHD Features and LV-LP Constraints

X73PHD defines point-to-point communications between the above mentioned agent and manager entities (PHD and CE, respectively, in Fig. 1). The agent attempts to establish an X73PHD association with the manager and, if it succeeds, they will proceed with the vital signs exchange by using different data schemes and services. To manage both device information

and vital signs acquired from the patient, X73PHD is based on an object oriented framework which guarantees scalability and reusability [10], [11], [12]. Basically, X73PHD defines three different models:

- Domain Information Model (DIM): this describes an abstract model composed of a set of object classes which are instantiated and can be referenced in an X73PHD communication, including the attributes and methods that can be executed on each one of them. DIM covers the definition of the Medical Device System (MDS) object (the root object in the agent modeling), scanner objects (for reporting of agent data), different metrics (numeric, real time sample array and enumeration objects), and Persistent Metric (PM) store and segment objects for long term data storage.
- Service model: this defines the means by which the manager can interact with the agent and distinguishes two different types of services as well: association services and object access services. Association services provide methods to negotiate and agree on a common configuration (association request and response), release associations and abort connections. Object access services provide methods that allow a manager to interact with an agent by remotely executing actions as well as allowing access object attributes through the established link. These services include:
  - *event reports* (often implemented by scanner objects and the MDS) that are initiated by the agent and used to send its configuration during the association procedure and medical or personal health data once the association has been established
  - *get* and *set* methods that allow the manager access to object attributes
  - *actions* that allow the manager to execute Remote Procedure Calls (RPCs) over agents' objects. Whilst *event reports* are initiated by agents' objects, *get*, *set* and *action* are executed over them and initiated by the manager.

  In addition, the service model provides protocol exception handling services meeting objectives such as aborting the association (*abrt*), error reporting (*roer*) and operation rejection (*rorj*).
- Communication model: this defines a Finite State Machine (FSM) for both agent and manager that controls the communication state and the transport layer. All the possible transitions in the FSM are well defined and they involve the execution of some actions internally in the agent or the manager, the reception or transmission of Application Protocol Data Units (APDUs), etc. The FSM determines the sequence diagrams in any X73PHD communication and its implementation is independent of the transport technology. It also includes different Encoding Rules (ER), such as Medical Device ER (MDER), that define the algorithms responsible for marshalling the APDUs generated by the DIM and how the abstract model given by the DIM has to be transformed into a stream of bytes.

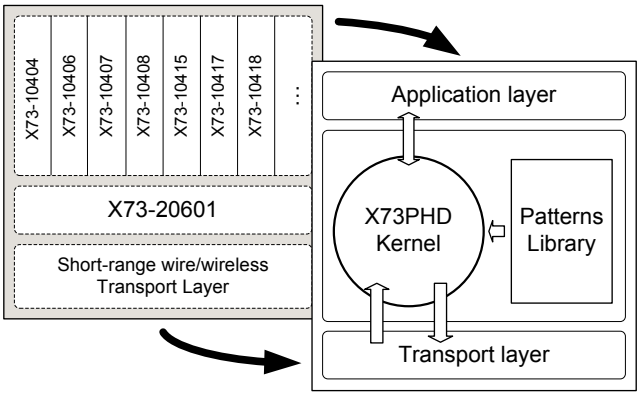


Fig. 2: Mapping of X73PHD three-layer protocol stack into the proposed patterns-based methodology.

From this three-component model, X73PHD defines the architecture for its communication protocol stack that enables the connection between agent and manager. This stack is divided into three levels (see left hand of Fig. 2):

- Device Specializations (X73-104xx) [12]. A set of model descriptions which describes all object classes and attributes representing the device components and the configuration of the MDS. New PHDs are continuously being developing, by defining their DIM through the particular X73-104xx specializations.
- Optimized Exchange Protocol (X73-20601) [10], [11]. The main part of this standard consists of a medical and technical terminology framework (DIM) which will be encapsulated inside APDUs. From the communication model definition, the point-to-point connection is developed through the FSM.
- Transport Layer. X73PHD data transmission can be held over any transport technology. The specific transport technologies supported by X73PHD are outside the scope of the standard. Several Special Interest Groups (SIG) have developed medical profiles for USB (Personal Health Device Class, USB PHDC [15]), Bluetooth (Health Device Profile, BT-HDP [16]) and ZigBee (Health Care Profile, ZHC [17]) following the recommendations of the integration initiative Continua Health Alliance [18].

The role of managers can be either as a basic gateway or as more complex monitoring equipment with added value services [19]. These devices are usually provided with an Operating System (OS, such as Windows, Windows Mobile, Symbian, Linux or Android) and sometimes with a Virtual Machine (VM, such as JAVA VM or the .Net platform for C# language). The key feature in these devices is a relatively high degree of intelligence. Usually, a single manager must recognize several agents and gather information from them. Moreover, in e-Health solutions, plug-and-play capabilities are required to facilitate the addition or replacement of agents. Therefore, PHDs can be used by the assistance staff or the patients themselves without having to configure or manipulate the manager.

Regarding agents, the ergonomics of these devices is a key point due to the fact that they are usually portable or wearable

which implies that, first, both size and weight must be kept to a minimum and, second, autonomy must be kept to a maximum. Furthermore, these devices are built on LV-LP microcontrollers with limited processing and memory features. Typical hardware features consist of 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 small Megaflops processor [20]. Both hardware and software features are usually designed ad-hoc for a specific agent. Hardware boards typically include a microcontroller, a communication module and a sensor; there are also System-on-Chip (SoC) modules that integrate all these components in a single chip. Although agents do not require a high degree of computational intelligence, the software framework is written in assembler, C or embedded C++ programming languages to increase efficiency. Moreover, software features are determined by the involved hardware characteristics because communication protocol stacks provide, in some cases, their own Real Time Operating System (RTOS) and Application Programming Interface (API). Regarding the communications protocol stack, even though X73 evolved from X73PoC to X73PHD to reduce complexity, it still involves relatively high requirements when it is investigated in the context of LV-LP. For example, in most common SoC modules used in ZigBee applications, RAM resources for the application layer are shared with the ZigBee stack and they are in the order of kilobytes. Depending on the X73PHD specialization implemented, incoming and outgoing APDUs can require up to 63 kilobytes and 8 kilobytes RAM, respectively, for buffering purposes [20]. Therefore, developers usually need to upgrade hardware requirements [21]. In these cases, the implementation of X73PHD poses a challenge for the developer.

Since X73PHD does not approach the software architecture, the accomplishment of an efficient design falls directly on the software developers. Typical implementation techniques associate the DIM with a static class model within the software solution [13]. Consequently, each object in the DIM is usually represented by an object in the targeted programming language. In this way, an increment in RAM arises due to two factors. First, a complete DIM image is created in RAM during the agent's boot-up process. Second, some additional buffers in RAM are needed in order to generate and transmit the APDUs each time the interchange of information is required. Indeed, such techniques produce implementations that turn out to be more generic and scalable but less efficient in terms of processor load and memory footprint. Nevertheless, alternative implementation approaches that outperform the APDU processing techniques are possible as according to X73PHD, there is no real need to implement the DIM in such a way. For X73PHD, agents are black boxes that must follow a communication protocol which is defined by an abstract model consisting of the DIM, the service model and the communication model. In this paper, a patterns-based methodology for X73PHD implementation is proposed in order to enhance PHD capabilities by reducing memory and processing requirements, extending device autonomy and reducing hardware costs.

III. Patterns-based Implementation Methodology

Following the previously explained X73PHD features and considering the LV-LP constraints, a patterns-based methodology is proposed in this section. The presentation of this methodology has been divided in two subsections: the first describes its basic principles and the second proposes a specific software architecture for a X73PHD-compliant LV-LP-constrained microcontroller.

A. Methodological principles

The pattern-based methodology gives great importance to the APDUs. X73PHD has been optimized considering the agent constraints. It provides mechanisms for reducing to a minimum the APDUs' overhead at byte level, thus showing low variability between fields [22]. A remarkable point is that most APDUs in X73PHD do not differ greatly. If a specific type of APDU in a given device configuration (e.g. an event report in any specialization) is taken and analyzed, the conclusion is that only a few bytes of the APDUs can change [10], [11]. Indeed, this analysis suggests that it is possible to use a template to generate APDUs [13]. Such a conclusion may have a huge impact on how to implement X73PHD minimizing the program footprint, volatile memory consumption and processor usage.

The fundamentals of this methodology are based on the concept of patterns. A pattern is a model that can be used to produce a copy of itself. In this case, the concept is applied to analyze APDUs interchanged by X73PHD agents. The idea is to build most of the structure of the APDU by using different previously-identified patterns and, thereafter, fill the non-fixed gaps, which usually represent a small percentage of the total APDU length. The similarities found among the APDUs of the same type can be explained through the highly abstract, structured, versatile and verbose schema of X73PHD. This idea is related to the concept of canned messages [13]. However, unlikely the canned messages approach, this approach do not need the use of buffers. Therefore, in addition to generating APDUs, patterns can also be used to process incoming APDUs using the same principles but back-ward.

An exhaustive study of the X73PHD APDUs is mandatory in order to obtain the patterns that are used by an agent specialization. This study must take into account all the X73PHD features that may appear in an agent-manager communication including the APDUs definition, the DIM structure of the device, the Abstract Syntax Notation One (ASN.1) definitions and the codification used (usually, MDER). Once every pattern has been determined, they are stored in the so-called Patterns Library which can be shared between and reused by devices with similar configurations, as well as interchanged between an agent and a manager with minimal processing resources, as is shown in Fig. 3. The outgoing APDUs are synthesized using the appropriate patterns from the Patterns Library along with certain variables stored in RAM, such as invoke-id and Simple-Nu-Observed-Value (e.g. a measurement obtained from the Analog-to-Digital (A/D) converter). The incoming APDUs are analyzed and correlated to the Patterns Library to identify and match suitable patterns while the non-fixed parts of the APDU

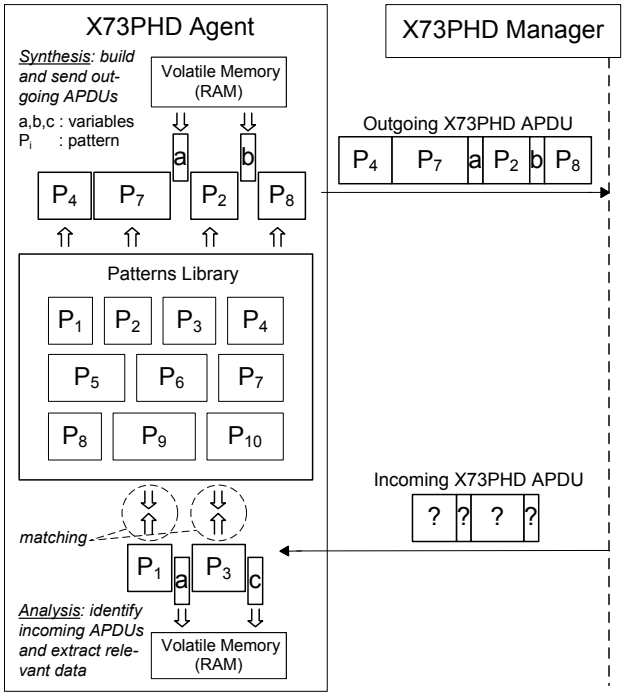


Fig. 3: Analysis and synthesis of APDUs using the principles of the patterns-based methodology.

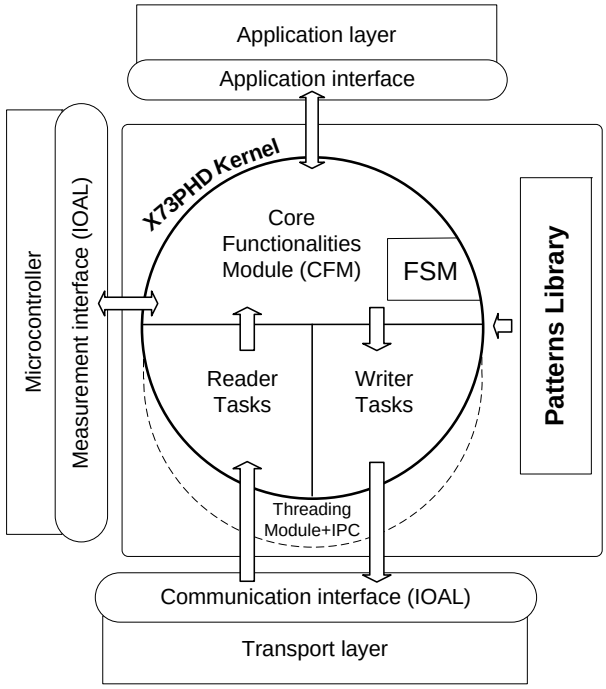


Fig. 4: Proposal of software architecture for the X73PHD implementation into a LV-LP microcontroller-based platform.

are saved in variables (e.g. the arguments of Set-Time actions). Both synthesis and analysis processes are represented in Fig. 3.

B. Proposal of software architecture for X73PHD implementation into a LV-LP microcontroller-based platform

The main components of the architecture proposed are the Patterns Library and the so-called X73PHD Kernel. The mapping between the X73PHD three-layer model to this proposed patterns-based methodology is shown in Fig. 2. The analysis and synthesis of APDUs using the aforementioned basic principles of this methodology is schematized in Fig. 3. The X73PHD Kernel manages the X73PHD FSM and the state of the DIM objects. Moreover, the X73PHD Kernel is responsible for both the analysis of incoming X73PHD APDUs, and the synthesis of outgoing X73PHD APDUs as is shown in Fig. 3. Module features have been reduced to a minimum in order to keep their footprint as small as possible while preserving relatively high execution efficiency. APDUs usually launch some processes in the agent (e.g. a change in the FSM state) according to X73PHD [10], [11], [12]. A scheme of the proposal of software architecture fully compliant with X73PHD is provided in Fig. 4. In this proposal, the X73PHD Kernel is built on top of the Threading Module, the Inter-Process Communication (IPC) module and the Input Output Abstraction Layer (IOAL). It provides X73PHD functionalities to the application layer in a transparent way. An in-depth technical description of these modules is provided as follows:

1) X73PHD Kernel: this is comprised of three modules: readers, writers and the Core Functionality Module (CFM) as is shown in Fig. 4. Readers are APDU analyzers and, reciprocally, writers are APDU synthesizers. The CFM controls the execution of readers and writers, the states of the protocols (including the FSM states), the interactions with the application layer, and the management of the IOAL.

No task is assigned to the CFM, whilst two independent tasks – corresponding to a reader and a writer – are assigned for each channel. This enables multiple channels with parallel processing and avoids the eventual block of the system (e.g., if data is not ready or transmission queues are full). The algorithms used to analyze and synthesize APDUs are based on the concepts previously described in Section III-A. Moreover, in order to reduce buffering requirements, the processing of APDUs is done byte by byte as soon as data and channels are ready.

Readers, writers and the CFM work together to provide X73PHD functionalities to the application layer in a transparent way. Readers analyze incoming APDUs continuously. Once an APDU has been analyzed, a notification containing a summary of the relevant information is sent to the CFM. The CFM takes different actions depending on the type of the incoming events, which include indications from the readers (e.g. APDU reception report), the Communication Interface (e.g. connection and disconnection reports), the Measurement Interface (e.g. new measurement available report), etc. These actions are then applied on the appropriate module: the application layer interface (e.g. indications regarding the state of the X73PHD stack), the Communication Interface (e.g. requests to establish or break a connection) or the writers (requests to send APDUs).

Following the above mentioned principles, an implementation proposal of the CFM is provided in this paper. This is approached by describing the relationship between the

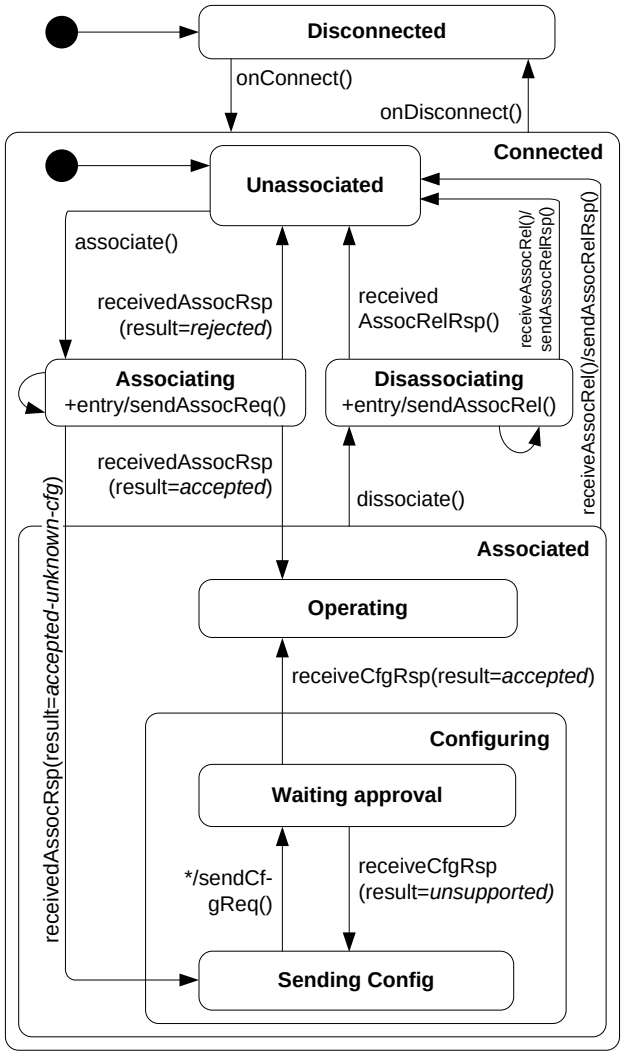


Fig. 5: Proposal of implementation of the Core Functionality Module (CFM) and its relationship with the Finite Sate Machine (FSM) of the X73PHD standard.

proposed implementation design and the X73PHD FSM. The implemented states and changes between states are shown in Fig. 5. A black circle with an arrow is used to indicate both the initial state and the initial substate inside a state. The notations <EVENT>, <EVENT>/<ACTION> and +entry/<ACTION> are used for state transitions similarly as in [10], [11]. For example, in the "Associating" state, if the association response is accepted (i.e. receivedAssocRsp(result=*accepted*)) the FSM jumps into the "Operating" state. The +entry/sendAssocReq() indicates that each time the FSM jumps into the "Associating" state, the CFM sends an association request APDU. In the "Associated" state, the receiveAssocRel()/sendAssocRelRsp() indicates that, after receiving an association release request, the CFM sends an association release response APDU. To send an APDU, the CFM sends the appropriate command to one writer through the IPC services and then the writer synthesizes and sends the corresponding APDU. Finally, a service operation example of the X73PHD Kernel is illustrated in Fig. 6. This corresponds

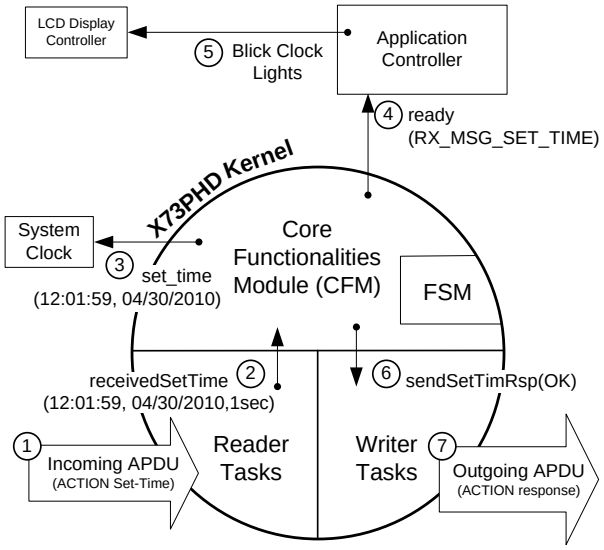


Fig. 6: Service operation example of the X73PHD Kernel for the Set-Time procedure.

to the Set-Time procedure defined in [10], [11]. Once the incoming APDU owing to the ACTION Set-Time arrives from the transport layer (1), the reader matches the corresponding APDU and fills up an extract using the Patterns Library. When the extract is concluded (2), the reader sends it to the CFM indicating in this case the Date-and-Time and the precision parameters, among others. As a result, this module starts to execute some actions. In this case, the APDU does not change the state of the FSM (it stays in the operating state) but sends other signals, such as the configuration of the system clock (3), and indicates to the application layer the event arisen (4). The application layer can provide information to the user by, for example, using a Liquid Crystal Display (LCD) controller (5). Once the core functionalities module has finished the processing, it must proceed with the confirmation of the action execution by sending a send-APDU command to one of the writers (6). Then the writer sends the outgoing APDU through the ACTION response using the Patterns Library (7).

2) The Threading Module: this provides the basic multitasking capabilities required to enable the parallel processing of APDUs, as is shown in Fig. 3. This module can be executed on top of the native OS. Traditionally, two scheduling approaches have been considered in the literature: 1) preemptive multitasking, where the time-slot allocation procedure is performed by the OS and 2) cooperative multitasking, where each task voluntarily cedes the processor to other tasks. For the development of this module, the second approach has been selected for two reasons. First, context-switching overheads are lower. Second, since the tasks usually require short time-slots – i.e. low latency –, there is no actual need for the OS to remove any task from the processor. The mathematical background of the switching algorithm implemented is explained as follows. However, some definitions are provided beforehand:

$$\{\langle R_{PR}\rangle_m\} \text{ is the set of Preserved Registers (PRs)} \quad (1)$$

R_{sp} is the stack register (2)

$MEM_{PR,i}$ is the memory in $task_i$ used to save PRs (3)

$MEM_{sp,i}$ is the memory in $task_i$ used to save R_{sp} (4)

The following switching algorithm is applied to switch the context between $task_i$ and $task_j$:

1) Save $\{\langle R_{PR}\rangle_m\}$ in $MEM_{PR,i}$
2) Save R_{sp} in $MEM_{sp,i}$
3) Load $MEM_{sp,j}$ in R_{sp}
4) Load $MEM_{PR,j}$ in $\{\langle R_{PR}\rangle_m\}$

Tasks can stay in different states: $RUNNING$ if the task is being or can be executed in the processor, $WAITING$ if the task has decided by itself to wait for any event to occur, and $BLOCKED$ if the task is waiting the conclusion of a complex operation.

3) IPC Services: this enables the processes to communicate with each other. The IPC service implements shared message queues. Each one of these queues has one server and one or more clients. Whenever the queue is empty and the server attempts to read a new message, the server is moved to the $BLOCKED$ state until a new message is available to be read. Then, the task is restarted and moved to the $RUNNING$ state. Something similar occurs when a client attempts to write a new item in a full queue. The client is moved to the $BLOCKED$ state until some memory is freed in the queue. Other IPC services can be implemented easily, but this service alone gives enough capabilities to implement versatile solutions to X73PHD implementations while not requiring considerable memory resources. Following these considerations, several access primitives have been specifically implemented for the IPC service, as it is shown in Table I.

4) IOAL: this makes the X73PHD Kernel independent of the hardware. In addition, appropriate parts of the implementation can be reused along with different hardware platforms (through the Measurement Interface) and communication technologies (through the Communication Interface). These interfaces are described below:

Communication Interface: this makes the X73PHD Kernel independent of the transport technology and the communication stack. This transport-independent interface has been defined in compliance with the requirements of X73PHD optimized exchange protocol [10], [11]. Therefore, this interface manages multiple channels that can be of two different types: reliable and best-effort. Reliable channels transmit data end-to-end using flow control and error recovery mechanisms. Therefore, APDUs are always expected to arrive in order and free of errors. In contrast, best-effort channels usually transmit data that require lower latency. In this case, packets may be delivered out of order or even lost. It is assumed by the specification that all channels are established simultaneously before the association and remain open during the entire Associated state. Thereafter, they can be used by the modules. The primitives that have been specifically implemented for this Communication Interface are shown in Table I.

Measurement Interface: this interface attempts to isolate X73PHD core functions from hardware specific issues. As opposed to the Communication Interface, the interface dealing with the acquisition of data is not approached by X73PHD.

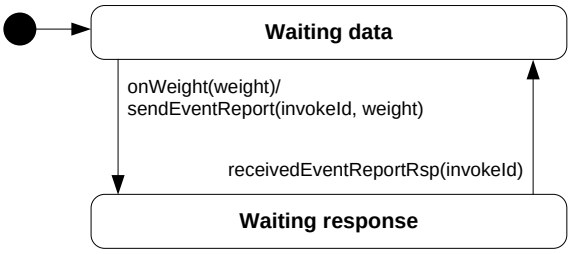


Fig. 7: Example of the state functions of the Core Functionality Module: event report for the X73PHD weighting scale specialization.

This poses a trade-off between hardware efficiency – i.e., adaptability and integration – and homogeneity, since this freedom broadens the number of different potential definitions, making it difficult to define a common interface for all specializations. This claim can be accomplished if a specific PHD is selected (the specific primitives developed for the implemented X73PHD-compliant weighing scale are defined in the next section). Therefore, the proposed Measurement Interface improves the reusability of the implementations based on Patterns Methodology.

5) Application layer interface: this makes the X73PHD Kernel independent of the user interface. The application layer interface can be used to provide X73PHD information to the application layer, for example, by using a LCD controller as is described in the following section.

IV. Results and Discussion

The patterns-based implementation methodology proposed in this paper has been applied to several X73PHD specializations as a proof-of-concept, including a weighing scale (11073-10415) [23], a blood pressure monitor (11073-10407) [24] and a thermometer (11073-10408) [25]. These implementations have been embedded in a development platform, thereby facilitating maintenance and extensibility. The weighing scale specialization, using the standard configuration (Dev-Configuration-Id=1500), has been selected to illustrate the details of this implementation, but similar results have been obtained for the other implementations (blood pressure monitor and thermometer). The software has been mostly developed using C programming language, which is the most used language for embedded devices and microcontrollers. The *x73_10415.c* source file contains the Patterns Library and the X73PHD Kernel. The *uc.s* assembler file and *vlib.c* source file correspond to the Threading Module. Finally, the *main.c* source file contains a testing module specifically implemented to check the correctness of the code operation and partially integrated in the program. The weighing scale report process, executed in the "Operating" state, is shown in Fig. 7. In this Figure, the state functions of the CFM even report are shown as well as the following specific primitives for the Measurement Interface: void onWeight(), called by the IOAL when a new measurement is available; weight_t getWeight(), used by the application to access to the measurement data; and void clrWeight(), used by the application to delete the measurement.

TABLE I: List of specifically implemented primitives for the IPC services and the IOAL Communication Interface.

Module	Type	Primitive	Function
IPC services	void	init_queue (queue_t q)	initialization of a queue indicated as q
	void	reset_queue (queue_t q)	reset of a queue indicated as q
	bool_t	push(queue_t q, byte_t b)	pushing of a byte at the end of the q queue
	bool_t	pop(queue_t q, byte_t *b, bool_t drop)	popping of a byte in the beginning of the q queue (storing its value at b)
IOAL	bool_t	newPacket(channel_t id, size_t s)	request to send a new packet of s size through the id channel
	bool_t	send(channel_t id , byte_t b)	sending of a new packet (located at the b byte) through the id channel
	bool_t	recv(channel_t id , byte_t b)	receiving of a new packet (located at the b byte) through the id channel
	void	onConnect(channel_t id)	notification of a connection through the id channel
	void	onDisconnect(channel_t id)	notification of a disconnection through the id channel
	void	onSend(channel_t id)	notification only if the previous send call returns FALSE when the transmit queue has any empty space
	void	onReceive(channel_t id)	notification only if the previous recv call returns FALSE when new data has arrived at the reception queue
	void	onPacket(channel_t id)	notification of a new packet is available

The resulting source code was compiled on different microcontroller architectures including 32-bit ARM7TDMI (LPC2294), 16-bit (MSP430) and 8-bit 8052 (ADuC841). The ARM7TDMI architecture has been selected for footprint comparison purposes, since it is widely used for 32-bit microprocessors. These results are shown in Table II. The object files *main.o*, *uc.o*, *vlib.o* and *x73_10415.o* correspond to the previously-presented *main.c*, *c.s*, *vlib.c* and *x73_10415.c* files, respectively. These numerical results can be considered as illustrative, as they may vary depending on the hardware platform, the X73PHD specialization and the compiler as well as the experience of the software developer. Nevertheless, these results highlight the reduced resources needed to implement an X73PHD agent into a microcontroller using the proposed patterns-based methodology: about 2546 bytes of flash memory and 168 bytes of RAM, meanwhile the previous prototypes over Personal Computers using Windows OS required about 1880 kilobytes (with X73PHD standard) [26] and 8450 kilobytes (with X73PoC standard) [14]. The footprint results using ADuC841 and MSP430 are of the same order of magnitude. Unfortunately, the comparison of those footprint results with other LV-LP approaches is not addressed in this paper because these data are not either of the public domain or comparable – given that the communication stack is included in the calculus – [18], [27], [28]. Nevertheless, based on our broad experience implementing X73PHD specializations, the results presented above can be considered as optimum and the patterns-based methodology very suitable for realizing X73PHD specializations in LV-LP microcontrollers.

TABLE II: Memory usage for the implemented X73PHD modules (in bytes)

Module	ro code	ro data	rw data
main.o	236		9
uc.o	84		4
vlib.o	14		
x73_10415.o	2 132	316	164
Total:	2 466	316	177

Regarding to X73PHD conformance, different levels of compliance are provided by X73PHD at the appropriate system interface and application interfaces, including both mandatory and optional clauses. The Implementation Conformance Statements (ICSs) define specifically the clauses implemented as well as the grade of conformance of an implementation to 11073-20601 and 11073-104xx which, in this case, corresponds to the weighing scale specialization 11073-10415 [10], [11], [12], [23]. In Table III, the ICSs for the implemented weighing scale specialization are shown. The first column represents specific codes GEN-x, REQ-y, POC-n, ATTR-n-a, NOTI-n-s, ACT-n-t (as defined in X73PHD), corresponding to general, minimum service support, PHD Object-and-Class (POC), POC attributes, POC notification, and POC behavior ICSs, respectively. The second column specifies the feature to be implemented. Finally the last column indicates the status of the feature (remarking if it has been implemented and the degree of implementation). In the case of static attributes, the last column also includes the implemented value.

More specifically, mention can be made of the following challenges faced during the realization of the patterns-based methodology into LV-LP microcontrollers. First, since the main aim is to implement the X73PHD standard, the natural trend of the developer is to reproduce the abstract structures of the standard in the source code. This easy-to-follow approach could be suitable in other platforms, as the X73Poc and X73PHD platforms previously implemented by our research group [14], [26], and, indeed, that was the initial approach taken by our group when attempting to implement the X73PHD standard into a LP-LV microcontroller. However, this approach was not suitable for several low-cost, LV-LP microcontrollers due to memory requirements. Therefore, an optimized implementation was pursued and,

TABLE III: ICSs of the weighing scale patterns-based implementation into an ARM7TDMI microcontroller

Index	Feature	Reference	Status, Support (Value)
GEN-1	Implementation/Description	–	Standard configuration Weighing scale. Configuration 1500
GEN-2	Std Doc Revision	–	IEEE 11073-20601-2008, IEEE 11073-20601-2010a
GEN-3	Conformance Adherence -Level 1-	–	Yes
GEN-4	Conformance Adherence -Level 2-	–	IEEE 11073-10415-2008
GEN-5	Communication Profile and hardware	–	Bluetooth/HDP
REQ-1	State Machine	–	Yes (-20601)
REQ-2	Protocol Messages	–	Yes (-20601)
REQ-3	Objects	–	Yes (-20601)
REQ-4	Encoding	–	MDER
REQ-5	Nomenclature	–	Yes (-10101)
REQ-6	Transport	–	Type-I compliance
SRV-1	GET service	8.3	Accepts command
SRV-2	SET service	8.3	Not supported
SRV-3	Confirmed SET service	8.3	Not supported
SRV-4	EVENT REPORT service	6.2	Not supported
SRV-5	Confirmed EVENT REPORT service	6.2	Sends command
SRV-6	ACTION service	8.3	Not supported
SRV-7	Confirmed ACTION service	8.3	Accepts command
POC-0	Simple_MDS:Weighing Scale	7.3.2.2	Implemented
POC-1	Numeric:Body Mass	7.3.2.2	Implemented
ATTR-0-1	Date-and-Time	7.3.2.3	Implemented/C, observational
ATTR-0-2	Dev-Configuration-Id	7.3.2.3	Implemented/M, static (1500)
ATTR-0-3	Handle	7.3.2.3	Implemented/M, static (0)
ATTR-0-4	System-Id	7.3.2.3	Implemented/M, static (11:22:33:44:55:66:77:88)
ATTR-0-5	System-Model	7.3.2.3	Implemented/M, static (U-WSBT-001)
ATTR-0-6	System-Type-Spec-List	7.3.2.3	Implemented/M, static (MDC_DEV_SPEC_PROFILE_SCALE v1)
ATTR-1-1	Absolute-Time-Stamp	7.3.3.3	Implemented/C, observational
ATTR-1-2	Attribute-Value-Map	7.3.3.3	Implemented/M, static (MDC_ATTR_NU_VAL_OBS_SIMP 4, MDS_ATTR_TIME_STAMP_ABS 8)
ATTR-1-3	Handle	7.3.2.3	Implemented/M, static (1)
ATTR-1-4	Metric-Spec-Small	7.3.3.3	Implemented/M, static (0xF040)
ATTR-1-5	Simple-Nu-Observed-Value	7.3.4.3	Implemented/C, observational
ATTR-1-6	Type	7.3.3.3	Implemented/M, static (MDC_PART_SCADA | MDC_MASS_BODY_ACTUAL)
ATTR-1-7	Unit-Code	7.3.3.3	Implemented/M, static (MDC_DIM_KILO_G)
ACT-0-1	Set-Time	7.3.2.4	Confirmed
NOTI-0-1	MDS-Configuration-Event	7.3.2.5	Confirmed
NOTI-0-2	MDS-Dynamic-Data-Update-Fixed	7.3.2.5	Confirmed

as a result, the patterns-based methodology was proposed to optimize memory consumption. Second, no complete testing or conformance tool for X73PHD is publicly available. The ValidatePDU checking tool currently provided by the National Institute of Standards and Technology (NIST) only supports X73PoC APDU analysis and it is not suitable for X73PHD conformance testing [29]. In the absence of such a tool, several alternatives have been investigated and used to check both the fulfillment of the application requirements and the conformance to X73PHD. First, a two own development tools – a test tool that validates APDU contents and sequences and the previously-implemented X73PHD platform [26] – were used. Second, the impleemnted LV-LP constrained X73PHD-compliant agents were able to successfully connect to an externally-developed X73PHD-compliant manager developed by the MORFEO OpenHealth project [30]. Despite that an exhaustive checking procedure – through an official, certified X73PHD testing tool – is unattainable at the moment of writing, evidence is provided of the compliance of the proposed implementation with the ICSs by means of these crossed-platform tests.

Regarding to reader and writer tasks (see Fig. 4), several optimizations were included to support some optional features of X73PHD. First, regarding reader tasks, several factors – such as the need of computing the length of the fields in the MDER codification or the high number of different errors that the agent has to be able to report (*roer*, *rorj* and *abrt* messages, mentioned in Section II) – increase both complexity and footprint. These factors were optimized by adding supplementary pattern-embedded information about error handling that enables the analysis procedure to easily determine by context the actions to be taken if an analysis error occurs. The pattern-embedded information consist on logical structures that contain all the information required to execute the corresponding actions. Second, regarding writer tasks, the computation time needed to synthesize variable-length APDUs can be reduced if the writer knows the length of the APDU beforehand. Some of these variable-length APDUs include the "Remote Operation Response | Get" using a variable attribute list argument and the "Remote Operation Invoke | Event | MDS-Dynamic-Data-Update-Fixed" using buffered-measurements. Specific algorithms that use microcontroller-efficient operations (mostly, addition, shift and product operations) to calculate the length of each variable-length APDU have been implemented. For

instance, in the "Remote Operation Response | Get" case, the APDU length, L, is obtained as: $L = L_h + L_{i_1} + ... + L_{i_N}$, where L_h is the header length (i.e. 18 bytes), L_{i_j}, is the length of attribute $i_j \in \{1...M\}$, M is the number of attributes of the MDS object, $j \in \{1...N\}$, N is the number of attributes in the list.

Once the final implementation for a weighing scale was achieved, the implementation and testing of a blood pressure monitor (11073-10407) and a thermometer (11073-10408) was carried out using the same procedures shown above. Only some minor changes and modifications were required to accomplish the implementation of these new specializations, compared to the initial development of the weighing scale. This design results in a reduced time to market while facilitates integration into a fully-fledged development platform.

V. Conclusion

In this paper, a patterns-based methodology and a software architecture to efficiently implement X73PHD into LV-LP agents in terms of both memory consumption and processor usage have been presented. The proposed methodology has been designed to offer an intuitive, easy-to-implement solution for LV-LP microcontrollers. The proposed software architecture splits the implementation problem in several modules that can be reused and applied to different software and hardware solutions. As a proof-of-concept, this methodology has been applied to implement a number of X73PHD-compliant agents: a weighing scale, a blood pressure monitor, and a thermometer. Quantitative results regarding X73PHD memory requirements have also been provided. These results show the reduced footprint required to implement the patterns based methodology and highlights its applicability to other X73PHD specializations.

References

[1] "Obesity: preventing and managing the global epidemic. report of a WHO consultation." *World Health Organization - Technical Report Series*, vol. 894, pp. i–xii, 1–253, 2000.

[2] J. L. Monteagudo and O. Moreno, "eHealth for patient empowerment in Europe," World Wide Web electronic publication, 2009, last access: 02/11. [Online]. Available: http://ec.europa.eu/information_society/newsroom/cf/itemdetail.cfm?item_id=3448

[3] P. Bonato, "Wearable sensors and systems," *IEEE Engineering in Medicine and Biology Magazine*, vol. 29, no. 3, pp. 25–36, 2010.

[4] M. Martinez-Espronceda, I. Martinez, S. Led, J. D. Trigo, I. Oses, J. Escayola, L. Serrano, J. Garcia, and A. Garcia, "INTENSA: heart failure patient's follow-up system using the ISO/IEEE11073 standard," in *2009 9th International Conference on Information Technology and Applications in Biomedicine (ITAB 2009)*, 5-7 Nov. 2009, p. 4. [Online]. Available: http://dx.doi.org/10.1109/ITAB.2009.5394343

[5] W. S. Wang, "Bluetooth: A new era of connectivity," *IEEE Microwave Magazine*, vol. 3, no. 3, pp. 38–42, 2002.

[6] P. Kinney, "ZigBee technology: Wireless control that simply works," *ZigBee Technology: Wireless Control that Simply Works*, 2003.

[7] C. Chronaki and F. Chiarugi, "Interoperability as a quality label for portable & wearable health monitoring systems." *Studies in health technology and informatics.*, vol. 117, pp. 108–116, 2005.

[8] M. Reynolds *et al.*, "Can telemonitoring systems interoperate? Review of the suitability of existing stantards for adaptable telecare provision," *Healthcare Comp Conf of British Comp Soc (HC)*, pp. 104–115, 2007.

[9] M. Galarraga, L. Serrano, I. Martinez, and P. de Toledo, *Standards for medical device communication: X73 PoC-MDC*, ser. Medical and Care Compunetics 3. IOS Press - "Studies in Health Technology and Informatics", pp. 242–56, 2006.

[10] "Health Informatics-Personal Health Device Communication Part 20601: Application Profile–Optimized Exchange Protocol," *IEEE Std 11073-20601-2008*, 2008. [Online]. Available: http://standards.ieee.org/findstds/standard/11073-20601-2008.html

[11] "Health Informatics–Personal Health Device Communication Part 20601: Application Profile–Optimized Exchange Protocol Amendment 1," *IEEE 11073-20601a-2010 (Amendment to IEEE Std 11073-20601-2008)*, pp. 1–119, 2011. [Online]. Available: http://standards.ieee.org/findstds/standard/11073-20601a-2010.html

[12] "Health Informatics-Personal Health Device communication Part 104xx: Device specializations," *IEEE Std 11073-104xx*. [Online]. Available: http://standards.ieee.org/findstds/standard/healthcare_it.html

[13] J. Yao and S. Warren, "Applying the ISO/IEEE 11073 standards to wearable home health monitoring systems," *Journal of Clinical Monitoring and Computing*, vol. 19, no. 6, pp. 427–436, 2005. [Online]. Available: http://dx.doi.org/10.1007/s10877-005-2033-7

[14] I. Martinez, J. Fernandez, M. Galarraga, L. Serrano, P. De Toledo, S. Jimenez-Fernandez, S. Led, M. Martinez-Espronceda, and J. Garcia, "Implementation of an end-to-end standard-based patient monitoring solution," *IET Communications*, vol. 2, no. 2, pp. 181–191, 2008. [Online]. Available: http://dx.doi.org/10.1049/iet-com:20060703

[15] "Universal Serial Bus Device Class Definition for Personal Healthcare Devices (USB PHDC) release 1.0," *USB Implementers Forum Inc.*, vol. 1st Ed., 2007.

[16] "Bluetooth Health Device Profile (BT HDP) version 1.0 revision 00 [Multi-Channel Adaptation Protocol (MCAP)][Implementation Guidance Whitepaper]," *Bluetooth Special Interest Group (SIG)*, vol. 1st Ed., 2008.

[17] "Zigbee Health Care profile (ZHC) specification version 1.0 revision 15," *ZigBee Alliance*, vol. 1st Ed., 2010.

[18] Continua Health Alliance. Last visit: 02/11. [Online]. Available: www.continuaalliance.org

[19] M. Martinez-Espronceda, L. Serrano, I. Martinez, J. Escayola, S. Led, J. Trigo, and J. Garcia, "Implementing ISO/IEEE 11073: Proposal of two different strategic approaches," in *Proceedings of the 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBS'08*, Vancouver, BC, Canada, pp. 1805–1808, 2008.

[20] "MSP430 family," last visit: 02/11. [Online]. Available: www.ti.com

[21] S. Warren, J. Lebak, and J. Yao, "Lessons learned from applying interoperability and information exchange standards to a wearable point-of-care system," Arlington, VA, United states, pp. 101–104, 2006. [Online]. Available: http://dx.doi.org/10.1109/DDHH.2006.1624807

[22] M. Clarke, D. Bogia, K. Hassing, L. Steubesand, T. Chan, and D. Ayyagari, "Developing a standard for Personal Health Devices based on 11073," in *Conf Proc IEEE Eng Med Biol Soc.*, pp. 6174–6176, 2007.

[23] "Health informatics-personal health device communication part 10415: Device specialization-weighing scale," *IEEE Std 11073-10415-2010*, dec. 2010. [Online]. Available: http://standards.ieee.org/findstds/standard/11073-10415-2010.html

[24] "ISO/IEEE Health Informatics Personal Health Device communication Part 10407: Device specialization - Blood pressure monitor," *ISO/IEEE 11073-10407:2010(E)*, pp. 1–52, 1 2010. [Online]. Available: http://standards.ieee.org/findstds/standard/11073-10407-2010.html

[25] "Health Informatics-Personal Health Device communication Part 10408: Device specialization-Thermometer," *IEEE Std 11073-10408-2010*, dec. 2010. [Online]. Available: http://standards.ieee.org/findstds/standard/11073-10408-2010.html

[26] I. Martínez, J. Escayola, M. Martínez-Espronceda, P. Muñoz, J. Trigo, A. Muñoz, S. Led, L. Serrano, and J. García, "Seamless integration of ISO/IEEE11073 personal health devices and ISO/EN13606 electronic health records into an end-to-end interoperable solution," *Telemedicine and e-Health*, vol. 16, no. 10, pp. 993–1004, 2010.

[27] "Freescale medical bus implementation," 2009, last visit: June 2010. [Online]. Available: http://www.freescale.com/webapp/sps/site/prod_summary.jsp?code=MEDICALUSB

[28] C. Park, J. H. Lim, H. Y. Jung, and S. Park, "ISO/IEEE 11073 PHD standardization of weighing scale using Nintendo's Wii balance board for healthcare services," Las Vegas, NV, United states, 2010, pp. 195–196. [Online]. Available: http://dx.doi.org/10.1109/ICCE.2010.5418874

[29] National Institute of Standards and Technology. Last visit: 02/11. [Online]. Available: www.nist.gov

[30] S. Carot-Nemesio, J. Santos-Cadenas, P. De-Las-Heras-Quirs, and J. Bustos, "OpenHealth the OpenHealth FLOSS implementation of the ISO/IEEE 11073-20601 standard," in *HEALTHINF 2010 - 3rd International Conference on Health Informatics, Proceedings*, pp. 505–511, 2010.

Miguel Martínez-Espronceda was born in Mendavia, Spain, in 1981. He received the MS degree in Telecommunication Engineering from the Public University of Navarra (UPNa), Spain, in 2005. Since then, he has developed his research within the Signal Theory and Communications Group, Department of Electrical and Electronic (UPNa). In 2009, he was a visiting student at the Department of Biomedical Engineering, University of Applied Sciences Technikum Wien, Austria. His current research interests include wearable and wireless personal health solutions based on standards and medical devices interoperability. He is Student Member of both IEEE and IEEE-EMBS, and collaborator in national and international standardization associations (AENOR/CTN139-SC4, CEN/TC251-WGIV, and PHD-WG).

Ignacio Martínez was born in Zaragoza (Spain) in 1976. He received MS (2000), DEA (2002), and PhD (2006) degrees in telecommunication engineering and bioengineering doctoral program from the Aragon Research Engineering Institute (I3A) in the University of Zaragoza (UZ), Spain. He received Best Thesis Award (2007) in Multimedia Environments from Telecommunication Engineering Official College (COIT), Spain. His research interests include telemedicine, QoS on multimedia services and interoperability & standardization, where he is coordinator of standard-based solutions for e-Health services with more than 30 published papers. He currently works in the development and implementation of ISO/IEEE11073 standard for medical devices interoperability within the Spanish Association for Standardization and Certification (AENOR/CTN139) and European Normalization Committee (CEN/TC251).

Luis Serrano (S'92, M'97, SM'06) was born in Andosilla, Spain, in 1966. He received the M.Sc. degree in Physics from the University of Zaragoza, Spain, in 1989, and Ph.D. degree in Electrical Engineering from Public University of Navarra, Spain, in 1995. From 1990 to 1993 he was a Researcher Assistant with Physics Faculty at University of Zaragoza where he worked in the field of analogue electronic design. In 1994 he moved to Public University of Navarra as Lecturer at Electrical and Electronic Engineering Faculty and worked as analogue electronic designer. Since 1997, he has been working in Bio inspired VLSI electronic systems for biomedical analogue signal processing, medical instrumentation, Wireless Communications, eHealth services and Medical Devices Standarization in the Biomedical Engineering Research Group of the Public University of Navarra.

Santiago Led holds a Telecommunication Engineering Degree from the Public University of Navarra, Pamplona (Spain) in 2003. Front 2004 to 2008 he was Teaching Assistant at the Public University of Navarra, and currently a part-time instructor at the same university. Since 2003, he has been engaged in research on analog biomedical signal acquisition and processing, wireless communications, wearable medical devices, eHealth services and medical devices standardization at the Biomedical Engineering Research Group, Public University of Navarra. He has also been researcher in several projects for deploying wireless technology in eHealth Services for Chronic Disease Management.

Jesús D. Trigo was born in Zaragoza, Spain, in 1981. He received the MS in Telecommunication Engineering from the University of Zaragoza in 2005. He is currently enrolled in a Doctoral Programme from the I3A in the Department of Electronics Engineering and Communications (CPS/UZ). He has undergone a research stage at the Biomedical Informatics Laboratory of the Foundation for Research and Technology - Hellas (FORTH) located in Heraklion, Crete (Greece). His main research interests include e-health applications and architectures, biomedical informatics or medical device interoperability and standardization among others.

Javiar Escayola received the MS (2005) in Telecommunication Engineering and Master in Biomedical Engineering (2008) from the Aragon Research Engineering Institute (I3A) in the University of Zaragoza (UZ), Spain. Since then he has been working at the I3A of the University of Zaragoza. His research interests include e-health, mobile applications, multimedia services, wireless communications, biomedical applications, and interoperability and standardisation, where he has developed several research projects within the I3A research line of telemedicine. He currently works in the development and implementation of ISO/IEEE11073 standard for medical devices interoperability within AENOR/CTN139 and CEN/TC251.

Asier Marzo was born in Pamplona, Spain, in 1986. He received the MS in Computer Engineering from the Public University of Navarra, Spain, in 2009. Since then, he has been working at Electrical and Electronic Department of Public University of Navarra as project assistant. He has worked in the software development industry and also as a teacher of programming languages in ESNE private school. He is the co-founder of InfinityK, a company aimed at software development for mobile platforms. His interests are real-time signal processing and simulations, game based learning, GCGPU and mobile e-health solutions.

José García (IEEE Member 2007) was born in Zaragoza, Spain, in 1971. He received the M.S. degree in physics and the Ph.D. degree "with Honors" from the University of Zaragoza, Spain, in 1994 and 1998, respectively. He is with the Department of Electronics Engineering and Communications in the Polytechnic Center, where he is currently the Head of the Department. He is an Associate Professor in the Telematics Engineering area and member of the Aragn Institute of Engineering Research (I3A). He is also the founder and responsible of the Telemedicine research group in the I3A. He is recipient, investigator and co-investigator of research grants in the area of telemedicine applications and networks. He has undergone several research stages on USA, Sweden and Austria and published more than 90-refereed international journal and conference papers. His research interests are in telemedicine, biomedical signal processing for transmission, wireless communications, network management and other related topics. (IEEE Member Number: 80637442).