# Solución 100% Estándar para Interoperabilidad de Dispositivos Médicos Personales ISO/IEEE 11073 sobre Perfil Médico Bluetooth HDP

A. Aragüés, J. Escayola, I. Martínez, P. del Valle, P. Muñoz, J.D. Trigo and J. García.
Aragon Institute for Engineering Research (I3A) - Univ. Zaragoza (UZ). c/María de Luna 3, 50018 Zaragoza
{aaragues, javier.escayola, imr, pdelvalle, pmg, jtrigo, jogarmo}@unizar.es

***Resumen*- Este artículo propone una solución 100% estándar basada en *software* libre (*open source*) sobre sistema operativo *Linux* para interoperabilidad de dispositivos médicos personales ISO/IEEE11073 *Personal Health Devices* (X73PHD) sobre perfil médico Bluetooth *Health Device Profile* (BT HDP). La solución se ha implementado sobre una arquitectura de capas de abstracción y se ha integrado con un servidor de Historia Clínica Electrónica (HCE) conforme a la norma internacional ISO/EN 13606. La arquitectura integra servicios en la nube y, con la incorporación de los paradigmas de e-Accesibilidad y usabilidad, constituye una propuesta completamente estándar para garantizar interoperabilidad de sistemas extremo a extremo.**

***Palabras Clave*- arquitectura de capas de abstracción, Bluetooth *Health Device Profile*, interoperabilidad, ISO/IEEE11073, *Personal Health Devices*, servicios en la nube.**

## I. Introducción. Interoperabilidad y estandarización de dispositivos médicos

La evolución más reciente de las nuevas tecnologías y las herramientas de la Sociedad de la Información ha transformado completamente el concepto de e-Salud. La Ingeniería Telemática (IT) juega aquí un papel imprescindible para abordar el reto de la integración de diferentes protocolos de comunicación en soluciones interoperables basadas en estándares. En este contexto, diversas tendencias tecnológicas están irrumpiendo con fuerza en el convulso ecosistema de los nuevos dispositivos (*Tablet* PCs, *Netbooks*, *Smartphones*) donde la aplicación de una norma que defina el intercambio de información entre dispositivos médicos se torna fundamental en un sector tan heterogéneo como la e-Salud. El estándar internacional para este propósito es ISO/IEEE 11073 *Personal Health Devices* (X73PHD) [1] y su implantación comercial viene siendo liderada por diversas iniciativas como *Continua Health Alliance* [2] o *Integrating the Healthcare Enterprise* (IHE) [3]. Estas iniciativas buscan la integración de X73PHD en los servicios sanitarios y la certificación de dispositivos X73PHD sobre los perfiles médicos adoptados para las tecnologías de transporte desde el grupo especial de trabajo *Personal Health Devices Working Group* (PHDWG) [4] para X73PHD: USB *Personal Health Device Class* (USB PHDC) [5], Bluetooth *Health Device Profile* (BT HDP) [6], y ZigBee *Health Care* (ZHC) [7].

Este es el primer paso para llegar a una plataforma totalmente interoperable pero, para lograr niveles óptimos en la calidad asistencial y continuidad de cuidado de un paciente, es necesario interactuar con la Historia Clínica Electrónica (HCE) del paciente para el seguimiento y autocontrol de su salud. El estándar internacional para lograr interoperabilidad de HCE entre sistemas sanitarios es UNE-EN/ISO 13606 [8]. Sin embargo, la existencia de normas médicas no garantiza la correcta implementación de una solución homogénea de e-Salud personal dado que la integración de las diferentes normas en soluciones extremo a extremo todavía sigue siendo una tarea compleja e intrincada. Así, y a partir de desarrollos anteriormente publicados [9]-[11], se presenta en este artículo una propuesta de arquitectura 100% estándar (denominada uz.health) para telemonitorización de pacientes integrando las normas de interoperabilidad ISO/IEEE 11073 y UNE-EN/ISO 13606 sobre sistema operativo Linux (ver Fig. 1). Esta solución se centra en entornos de e-Salud personal y utiliza la más reciente versión X73PHD. Posibilita una comunicación agente-manager entre los dispositivos médicos y un dispositivo *Tablet* PC, *Netbook*, etc. a través del interfaz de red personal (*Personal Area Networks*, PAN) y diferentes tecnologías inalámbricas (como Bluetooth). Se homogeneíza la comunicación de área extendida (*Wide Area Network*, WAN) con un servidor de HCE, mediante tecnologías *Web Services* (WS) y formato *eXchange Markup Language* (XML), que garantiza intercambio interoperable de extractos de la HCE del paciente conforme a UNE-EN/ISO 13606.

En la Sección II se describe el diseño de la solución basada en ISO/IEEE 11073, planteando las reglas en las que se sustenta, proponiendo una arquitectura de capas de abstracción y detallando el núcleo central conforme al estándar. En la Sección III se describe la implementación de la arquitectura propuesta garantizando las especificaciones de X73PHD. En la Sección IV se analiza la integración con el nuevo perfil médico BT HDP como tecnología de transporte recomendada por X73PHD. En la Sección V se discute la integración con el servidor de HCE y los servicios de usuario basados en la nube. Finalmente, en la Sección VI se discuten las conclusiones y líneas futuras de trabajo.

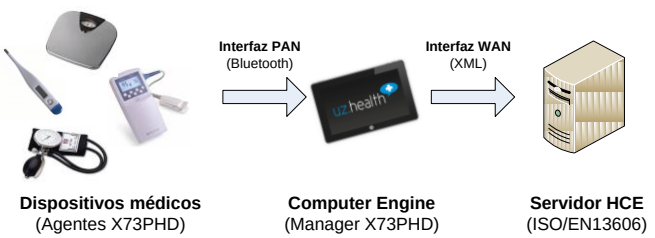



Fig. 1 Esquema de la plataforma de telemonitorización 100% estándar.

## II. Diseño de la solución basada en ISO/IEEE 11073

El primer paso en el diseño de la arquitectura es el análisis detallado de las propuestas previas [9]-[11] y la detección de sus limitaciones para, como resultado, proponer las siguientes reglas de diseño:

- **Escalabilidad**. La arquitectura es flexible para extender su margen de operaciones sin perder calidad o bien manejar el crecimiento continuo de trabajo de manera fluida. Así, añadir nuevas funcionalidades a la arquitectura propuesta es un proceso simple de modificación del código.
- **Modularidad**. La arquitectura se compone de la unión de varias partes que interactúan entre sí por un objetivo común, realizando las tareas necesarias para su consecución. El código propuesto separa módulos fundamentales como, por ejemplo, las partes que realizan tareas de comunicación del procesado de la información de la parte gráfica.
- **Portabilidad**. El *software* de la arquitectura para ejecutarse en diferentes plataformas ya que la solución es compatible con diferentes entornos y no existe una única pareja arquitectura - sistema operativo.
- **Mantenibilidad**. La arquitectura presenta, a nivel *software*, facilidad para ser modificada, corregir fallos, mejorar su funcionamiento u otros atributos, adaptarse a cambios en el entorno, etc. sin tener que rehacer prácticamente el código.
- **Robustez**. La arquitectura propuesta, además de realizar el trabajo esperado, está preparada para adaptarse a múltiples cambios en el flujo de trabajo y reaccionar apropiadamente ante condiciones excepcionales.
- **e-Accesibilidad y usabilidad**. La arquitectura evoluciona respecto a las propuestas previas ya que se soporta sobre *software* capaz de ser comprendido, usado y atractivo para el usuario. Además, incorpora aplicaciones específicas para que toda persona pueda usar el servicio, independientemente de su capacidad técnica, cognitiva o física.

Con todas las premisas anteriores, se propone un diseño basado en capas de abstracción sobre lenguaje de programación C# y plataforma de desarrollo Mono (la implementación libre de C#) como entorno multiplataforma (Windows/Linux/Mac Os). Esta arquitectura ofrece la posibilidad de crear una o diferentes interfaces de usuario totalmente independientes del núcleo e incluso del lenguaje de programación (con alguna capa de adaptación). Esto permite dotar al núcleo de capacidades de actualización automática (sin necesidad de modificar el resto de capas) o incluso proporcionar conexión y desconexión de diferentes tecnologías de transporte sin tener que modificar el núcleo principal. Este trabajo, como se ha comentado, se soporta sobre un núcleo conforme al estándar internacional 11073-20601 para interoperabilidad de dispositivos médicos, pero su diseño es compatible con otros protocolos de comunicación como HL7 [12]. En Fig. 2 se observa el diseño basado en capas de abstracción, que incluye tres capas principales:

- **Capa tecnológica**, donde se integrarán las tecnologías de transporte recomendadas desde el grupo especial de trabajo PHDWG para X73PHD: USB PHDC, BT HDP y ZHC. Esta capa presenta un interfaz homogéneo de comunicación hacia la capa superior a través de canales virtuales.
- **Capa de comunicaciones**, que presenta los diferentes protocolos de comunicaciones; en este caso, X73PHD, pero también, en un futuro, HL7 o protocolos propietarios.
- **Capa de usuario**, que incluye un interfaz gráfico (web o de escritorio), incluso con servicios de valor añadido.

En este tipo de arquitectura es crítico el diseño de la capa central, pues es el bloque que recibe los datos enviados por uno o más sistemas agente y gestiona el proceso de comunicación para asegurar que el intercambio de datos sea de acuerdo al protocolo estándar 11073-20601. Todos los protocolos definidos por X73PHD utilizan sintaxis abstracta (*Abstract Syntax Notation One*, ASN.1) para facilitar la especificación de las estructuras de datos sin hacer referencia a una tecnología de implementación concreta. ASN.1, junto con unas reglas de codificación específicas, facilitan el intercambio de estructuras de datos describiendo esas estructuras de forma independiente de la arquitectura de la máquina y el lenguaje de implementación. 11073-20601 utiliza reglas de codificación optimizadas para ASN.1 denominadas *Medical Devices Encoding Rules* (MDER). MDER fue elaborada en la norma 11073-20101 para minimizar el ancho de banda utilizado en las transferencias de datos médicos personales entre agentes y managers. La ausencia de implementaciones *open source* de estas reglas de codificación motivó la utilización y modificación de *Binary Notes*, un *framework* ASN.1 para C# y Java. Este *framework* presentaba las reglas de codificación más comunes (*Basic/Packet/Data Encoding Rules*, BER, PER o DER). Sin embargo, MDER no era soportado y tuvo que implementarse un nuevo codificador y decodificador para reglas entre dispositivos médicos. MDER es obligatorio tanto para agente como manager aunque los dispositivos pueden opcionalmente establecer otras reglas de codificación como *eXchange Encoding Rules* (XER) o PER. Este bloque central utiliza las librerías de *Binary Notes* para codificar las tramas de datos (*Applications Protocol Data Unit*, APDUs), utilizando las reglas de codificación anteriormente descritas además de las reglas MDER implementadas para esta arquitectura. Fig. 2 ilustra los componentes principales de la implementación:

- ***Domain Information Model* (DIM)**. Constituye el núcleo principal de la arquitectura y define un conjunto de clases para modelar agentes según objetos que pueden contener fuentes de datos (señales biológicas, eventos, informes de alertas, etc.), y los métodos que un manager puede usar para controlar el comportamiento de un sistema agente.
- ***Service Model***. Define el mecanismo de los servicios de intercambio de datos entre manager y agente que están mapeados en mensajes codificados usando ASN.1.
- ***Communication Model***. Se define a través de una máquina de estados finitos (*Finite State Machine*, FSM) que sincroniza los mensajes intercambiados entre manager y agente en las conexiones punto a punto.

Como se ha comentado, para mantener la independencia con la capa de transporte se ha diseñado un modelo abstracto de comunicación basado en canales virtuales. Un canal virtual puede gestionar varios canales simultáneos al mismo tiempo en cada conexión agente-manager. Además, cada uno de esos canales dentro de un canal virtual puede ser de una tecnología o perfil diferente, como Bluetooth *Serial Port Profile* (BT SPP) o BT HDP. De esta forma, se facilita un interfaz común de comunicación al manager para enviar y recibir APDUs desde o hacia el sistema agente. Para facilitar las tareas de las

capas superiores, este núcleo central publica notificaciones sobre eventos internos recibidos desde los agentes, además de facilitar un interfaz para controlar el estado de los agentes conectados al manager. Las aplicaciones que usen este núcleo central pueden suscribirse a eventos de la forma tradicional de C# pudiendo obtener notificaciones sobre eventos genéricos del manager como conexiones o desconexiones de un nuevo agente. Además, este diseño permite ampliar la arquitectura mediante *plug-ins*, que son módulos que se relacionan con una aplicación para aportarle una función nueva y generalmente muy específica. Estos complementos permiten que desarrolladores externos colaboren con la aplicación principal extendiendo sus funciones, reduciendo el tamaño de la aplicación y separando el código fuente de la aplicación por incompatibilidad de las licencias de *software*. Para ello, se ha definido una *Application Programming Interface* (API) que permita a terceros crear complementos que interactúen con el bloque principal propuesto.

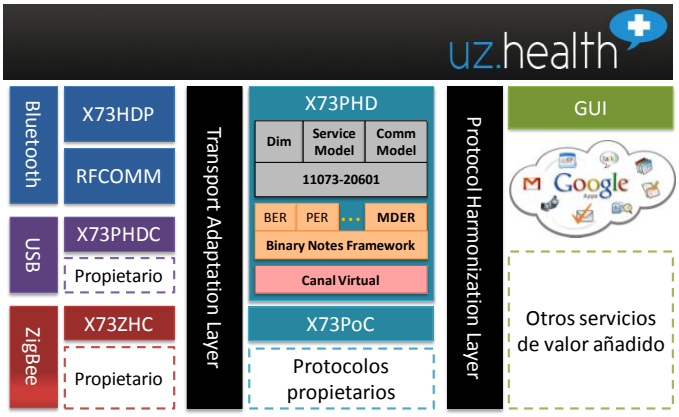



Fig.2. Propuesta de arquitectura de la solución basada en el estándar X73PHD según diseño basado en capas de abstracción.

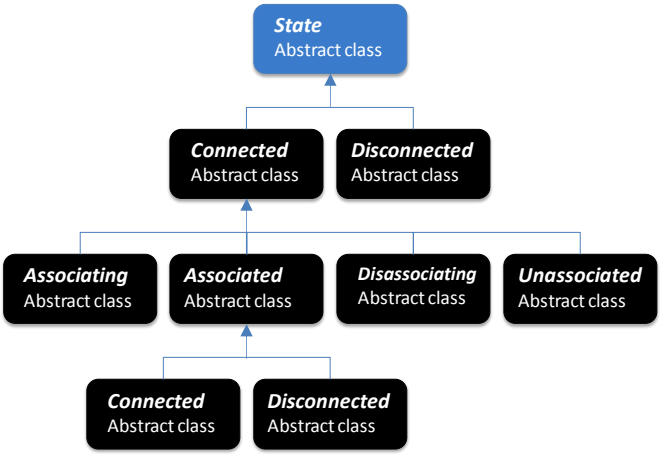



Fig. 3. Propuesta de implementación de la máquina de estados FSM X73PHD.

## III. Implementación de la Arquitectura Propuesta

La implementación de la arquitectura está pensada como un cubo de piezas donde cada usuario pueda construir un manager o un agente pieza por pieza de forma modular. Todas estas piezas se pueden obtener de la librería principal (denominada `UZ_ieee_11073`). Esta librería, desarrollada en C#, contiene todos los objetos necesarios para crear un manager X73PHD, depende únicamente de la librería *Binary Notes*, y es compatible con *framework* .NET 2.0 o superiores. Además de lo anteriormente comentado, *Binary Notes* proporciona un compilador ASN.1 (BNCompiler) que construye clases Java o C# a partir de un fichero de especificaciones ASN.1 de entrada. Sin embargo, ya que *Binary Notes* no implementa MDER, algunas clases de la librería se han modificado y creado otras conforme a la norma 11073-20101 (tipos de datos, atributos, objetos, etc.). Con casi 4000 líneas de código, la librería `UZ_ieee_11073.dll` contiene el grueso de implementación de la arquitectura propuesta. El proyecto se ha organizado intentando separar las partes en las que se divide el estándar y todas aquellas clases de apoyo que tienen funcionalidades similares. Esta separación funcional permitirá, en un futuro, modularizar todavía más la solución, moviendo cada una de las partes a librerías independientes y que puedan ser reutilizables por otras aplicaciones o códigos. A continuación se describe brevemente el contenido de cada uno de los espacios de nombres (*namespaces)* que conforman la librería:

- **Config**. Contiene las clases que gestionan la configuración de un dispositivo, como tipo de codificación, versión de protocolo utilizada, etc.
- **Core**. Contiene los objetos a más alto nivel de la librería. Los objetos agente y manager se construyen a partir de canales, objetos *Medical Device System* (MDS), etc.
- **Events**. Contiene las clases que se encargan del sistema de eventos de la implementación ya que el núcleo X73PHD publica eventos para que otras clases u objetos los reciban (nuevas medidas, conexión y desconexión de dispositivos).
- **Exceptions**. Contiene los tipos de excepciones utilizadas explícitamente para esta implementación ya que no sólo pueden generarse eventos sino también excepciones de muchos tipos durante la ejecución del núcleo.
- **Gateway**. Contiene capas de adaptación para comunicar el núcleo X73PHD con el interfaz exterior.
- **Logging**. Contiene la salida de un sistema de *log* de errores robusto, flexible y multi-hilo que genera múltiples tipos de mensajes, ya sea por consola, mensajes remotos, etc.
- **Part_10101**. Contiene, en una única clase, toda la nomenclatura del estándar X73PHD.
- **Part_104xx**. Contiene todas las especializaciones de dispositivos médicos en clases independientes (definidas en la norma 11073-104xx) aprobadas por X73PHD a fecha de redacción de este artículo.
- **Part_20601**. Contiene el grueso de la implementación, la maquina de estados FSM (ver Fig. 3), los canales virtuales, las tramas de datos APDUs, etc.
- **Utils**. Contiene clases de apoyo al resto de clases para realizar operaciones determinadas u objetos que no tienen una clasificación especial.

A continuación se detallan más en profundidad los principales *namespaces* de la librería `UZ_ieee_11073.dll`.

- **Part_10101**: En este *namespace* se definen los códigos de nomenclatura que ofrecen la posibilidad de identificar cláramente objetos y atributos en relación a su código entero *Object Identifier* (OID). Esta nomenclatura está dividida en particiones para organizar los códigos (que se definen mediante constantes estáticas que representan los atributos) dependiendo de su contenido y función.
- **Part_20601**: En este *namespace* se encuentran todos los objetos que definen el comportamiento del núcleo de la implementación según el protocolo 11073-20601. Este *namespace* se compone de otros cuatro *sub-namespaces* que engloban clases homogéneas:
  - **ASN1**. En este *namespace* aparecen los objetos ASN.1 generados mediante el compilador BNCompiler.
  - **FSM**. En este *namespace* se incluyen las clases relacionadas con la máquina de estados FSM que define el modelo de comunicación de ISO/IEEE 11073. En un primer nivel, se definen los interfaces y clases abstractas necesarias para la implementación de la máquina de estados. La implementación de los estados será diferente si el dispositivo es manager o agente. Se muestra en Fig. 3 la implementación de los diferentes estados de FSM, inspirada en el diseño del proyecto Morfeo OpenHealth [13]. Todos los estados derivan de una clase abstracta base denominada *State*. Todo estado tiene un manejador de estados (*IStateHandler*), un nombre identificador del estado y dos métodos (*Process*, para procesar las APDU provenientes de otro dispositivo, y *ProcessEvent*, para procesar diferentes eventos que puedan producirse como desconexión de otro dispositivo, problemas en la red, etc). El interface *IStateHandler* es un elemento clave para manejar los estados y permite enviar la información hacia el interfaz exterior desde cada uno de los estados. Todas las clases de la máquina de estados son clases abstractas por lo que no se pueden instanciar. A modo de ejemplo, se analiza la implementación del estado *Disconnected*. En este estado, el manager X73PHD no procesa ninguna APDU, únicamente los eventos de conexión o desconexión. Así en el método *ProcessEvent*, cuando llega un evento de conexión (*Event.Connection*), se modifica la propiedad *State* del *IStateHandler* indicándole el siguiente estado de FSM: *Unassociated*. En este estado *Unassociated* se procesan tanto eventos como APDUs. Así, si desde *Unassociated* se recibe un evento de desconexión, se mueve *IStateHandler* a estado *Disconnected*. En el caso del procesado de la APDU, si se recibe una confirmación (AARQ), se inicializa el proceso de asociación y, en caso contrario, se envía una APDU de abort por causas no definidas (*AbrtApdeuUndefined*) tal como se define en la norma 11073-20601.
  - **Messages**. En este *namespace* se construyen las APDUs que se utilizan para el intercambio de información en 11073-20601. Todas estas APDUs derivan de una clase base *ApduType* a la que se añade información. Algunos ejemplos son: *AbrtApdu* (APDU para abortar conexión en la que se indica la razón del aborto de conexión y se construye una APDU específica para esta situación), *AbrtApduBufferOverFlow* (APDU para abortar la conexión por desbordamiento de *buffer*, como caso específico heredado del anterior), o *AareRejectApdu* (clase compleja que indica el rechazo a la asociación).
  - **PHD**. En este *namespace* se definen las diferentes clases que conforman el DIM y que caracterizan un dispositivo médico. El DIM caracteriza la información de un agente como un conjunto de objetos. Cada objeto tiene uno o más atributos. Los atributos describen medidas que son comunicadas a un manager, así como elementos que controlan el comportamiento e información del estado del agente. A modo de ejemplo, destaca la clase base *metric* para todos los objetos que representan medida, estado y datos de contexto. Tal como impone X73PHD, esta clase no puede ser instanciada, por ello se trata de una clase abstracta. También destaca la clase MDS, una de las más importantes de toda la librería ya que cada agente es instanciado directamente desde una clase MDS. El objeto MDS representa la identificación y el estado de un agente a través de sus atributos. Y cabe resaltar que este *namespace* contiene la implementación del canal virtual y los diferentes tipos de canales para cada una de las tecnologías de transporte (USB PHDC, BT HDP y ZHC).

Por último, se propone en Fig. 4 un ejemplo de la implementación básica de un manager X73PHD utilizando la librería implementada *ad-hoc* `UZ_ieee_11073`. En el código propuesto, la clase *Manager* implementa el interfaz *IEventListener* que debe ser obligatoriamente implementado por cualquier clase que quiera escuchar los eventos de un agente. En el constructor de esta clase se inicializan los canales (en este caso, únicamente se ha inicializado el gestor genérico de canales para *Transmission Control Protocol* (TCP), aunque podrían haberse inicializado otros como BT SPP o BT HDP). En el momento que el manager del canal TCP dispara el evento *AgentAttached*, ya existe un agente conectado y se comienza a escuchar todos sus eventos. La modularidad de la solución propuesta permite añadir nuevos canales a la librería o incluso nuevos protocolos de comunicaciones sin afectar al resto del código.

## IV. Integración con Bluetooth HDP

Como se ha comentado, la arquitectura propuesta permite la integración de los nuevos perfiles médicos recomendados por PHDWG para las tecnologías de transporte soportadas sobre X73PHD. A partir de las experiencias anteriores que se soportaban sobre TCP/IP o BT SPP, en esta sección se detalla la integración del perfil médico BT HDP lo que constituye una solución X73PHD 100% estándar. El perfil médico BT HDP, en combinación con la especificación 11073-20601, facilita un *framework* robusto y basado en estándares que permite la interoperabilidad completa entre dispositivos de e-Salud sobre Bluetooth. A fecha de redacción de este artículo, únicamente algunas pilas Bluetooth comerciales presentan compatibilidad con el nuevo perfil médico BT HDP: *Jungo BTware* [14], diseñada para sistemas empotrados de bajo consumo; *Stollmann BlueCode+* [15], diseñada con una arquitectura independiente de la plataforma; *Toshiba* Bluetooth *Stack* [16], diseñada para Windows y certificada por *Continua Health Alliance*; y *Ethermind* Bluetooth *Stack* [17], desarrollada por la empresa *Mindtree* para sistemas empotrados y otras arquitecturas.

```
using System;
using System.Collections.Generic;
using UZ_ieee_11073.Core;
using UZ_ieee_11073.Events;
using UZ_ieee_11073.Part_20601.PHD.Channel.TCP;
namespace UZ_ieee_11073_Manager
{   public class Manager : IEventListener
    {   private TCPManagerChannel _tcpManagerChannel;
        private List<Agent> _agentsList;
        public Manager()
        {
           _agentsList = new List<Agent>();
           InitializeChannels();
        }
        private void InitializeChannels()
        {
           _tcpManagerChannel = new TCPManagerChannel();
           _tcpManagerChannel.AgentAttached +=
AgentAttached;
        }
        void AgentAttached(Agent agent)
        {
           agent.EventManager.AddEventListener(this);
        }
        public void Start()
        {
           _tcpManagerChannel.Start();
        }
        public void Stop()
        {
           Log.Debug("Stopping Manager");
           _tcpManagerChannel.Finish();
           foreach (Agent agent in _agentsList)
           {agent.SendEvent(new
Event(Event.AssociationAbortRequest));
           gent.FreeResources();
           }
           _agentsList.Clear();
        }
        #region IEventListener implementation
        public void StateChanged(Agent agent,
StateChange stateChange)
        {
           Log.Debug("Agent " + agent.SystemId + "status
has changed. " + stateChange);
        }
        public void NewMeasure(Agent agent,
List<Measure> measure)
        {
           Debug.Log("The agent " + agent.SystemId +
"has sent new measures:");
           foreach(Measure ms in measure)
                Debug.Log(ms.ToString());
        }
        public void AgentDisconnected(Agent agent)
        {
           agent.FreeResources();
           _agentsList.Remove(agent);
        }
        public string Name
        {
           get { return "UZ_ieee11073_Manager"; }
        }
        public void AgentConnected(Agent agent)
        {  if (_agentsList.Contains(agent))
           {
           Debug.Log("Agent already connected");
           return;
           }
           _agentsList.Add(agent);
        }
        #endregion
   }   }
```

Fig. 4. Ejemplo de implementación básica de un manager X73PHD.

Todas estas propuestas se apartan del camino de la interoperabilidad por ser soluciones comerciales, cerradas, que no permiten evolucionar el desarrollo con nuevas funcionalidades alejándose de los paradigmas de diseño planteados. Sin embargo, en 2009, desde la Universidad Rey Juan Carlos I, se inició un camino para integrar el perfil médico BT HDP en la pila oficial BlueZ de Linux [18] ofreciendo, así, las nuevas funcionalidades HDP a plataformas como Linux, Android o Meego, todas ellas apoyadas en el *kernel* de Linux. En los primeros pasos de esta iniciativa, la pila BlueZ no contemplaba algunos de los bloques y protocolos necesarios y específicos del perfil médico BT HDP. En Fig. 5 se muestra un diagrama de bloques de la arquitectura BlueZ, mostrando en azul aquellos bloques que no estaban previamente implementados o que necesitaron reescribir parte de su código. Estos bloques específicos del perfil médico se detallan a continuación:

- ***Multichannel Adaptation Protocol* (MCAP)**. Es un protocolo basado en *Logical Link Control and Adaptation Protocol* (L2CAP) que facilita un mecanismo sencillo para manejar múltiples canales de datos. Este protocolo soporta diferentes configuraciones de canal dependiendo de la aplicación o las necesidades de la transmisión como por ejemplo los modos *reliable* o *streaming*. La implementación de MCAP se realizó totalmente compatible con la especificación de Bluetooth *Special Interest Group* (SIG) [19], soportando todas las características obligatorias y algunas de las opcionales, como la re-conexión de canal. Como ya se ha comentado, MCAP requiere los modos de *streaming* y retransmisión mejorada de L2CAP. Estos modos no existían al comienzo del desarrollo del perfil BT HDP y se han ido construyendo paralelamente por los miembros de BlueZ.
- ***Bluetooth Health Device Protocol* (HDP)**. El perfil BT HDP permite a los dispositivos agente (conocidos en Bluetooth como *source* y asociados a los dispositivos médicos) intercambiar datos con los dispositivos manager (conocidos en Bluetooth como *sink* y asociados a móviles, portátiles, *Smartphones*, *Tablet* PCs, etc.). HDP es un perfil que simplifica el uso de MCAP y es el encargado de anunciar a otros dispositivos las características soportadas así como los canales usados por cada perfil. Para estos anuncios hace uso del *Service Discovery Application Profile* (SDP). De esta forma, los dispositivos pueden encontrar al manager y conectar con él o viceversa: el manager puede iniciar una conexión a un dispositivo. La implementación de BT HDP simplifica el uso de MCAP al usuario y registra toda la información necesaria en el registro SDP. Aunque en la primera versión de la implementación se usaban *callbacks* para notificar eventos al usuario, la actual se ha re-escrito para adaptarse a las guías de estilo de BlueZ utilizando el servicio de mensajes de sistema DBus [20], como se detallará más adelante. Esta implementación esconde por completo el mecanismo de control de los canales, facilitando un modelo de abstracción basado en un servicio de notificación de nuevos *MCAP Communication Links* (MCL) tales como nuevos dispositivos remotos. Además, esta implementación gestiona la creación de nuevos canales de datos y controla la correcta configuración de los mismos antes de ser notificados al usuario.

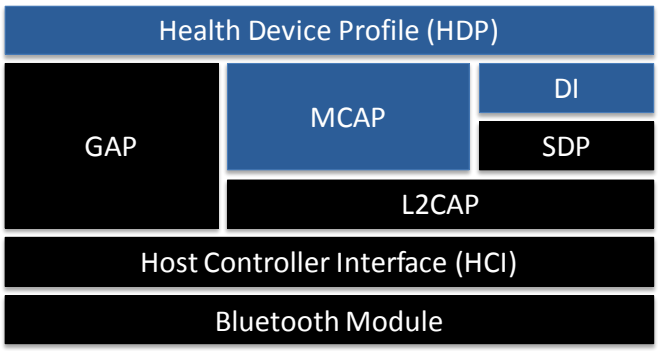



Fig.5. Pila de protocolos BlueZ.

- ***Device ID Profile* (DI)**. Especifica un método por el cual dispositivos Bluetooth comparten información que puede ser usada por otros dispositivos para encontrar imágenes o *software* asociado, como controladores específicos. Toda esta información también se publica en el registro SDP.

Como se ha adelantado, BlueZ utiliza el servicio de mensajes de sistema DBus. DBus es un protocolo de comunicación entre procesos. Ha sido diseñado para que sea ligero y fácilmente utilizado por cualquier programa. La arquitectura de DBus se compone de dos partes básicas: la librería *libdbus*, y un *daemon* que sirve como repetidor de los mensajes. La librería *libdbus* crea conexiones o canales que conectan dos aplicaciones (ver Fig. 6). Lo que hace es usar esa única conexión para conectarse al *daemon* de DBus, que se comporta como un repetidor. De esta forma, todas las aplicaciones que se conecten al *daemon* podrán contactar entre sí. La información se transmite en forma de mensajes que los hay de dos tipos: métodos (que pueden modificar el estado de un objeto o recabar información sobre el mismo) y señales (sirven para notificar un suceso de interés general). DBus es independiente del lenguaje que se use para acceder a él y hace que los objetos sean unas entidades no asociadas a ningún lenguaje concreto. Como un objeto en DBus es una ruta, dichos objetos son direccionados a través de una ruta que equivale a su nombre (un programa publica "objetos-rutas" (*ObjectPath*) a las que se puede acceder) y son equivalentes a las que se emplean en el sistema de ficheros de Linux. Cada aplicación debe tener una ruta única y no es complicado encontrar rutas como "/org/bluez/" ó "/org/freedesktop/DBus". Toda la información que circula a través de DBus puede ser depurada utilizando el comando *dbus-monitor* en una consola de texto y ver cómo se suceden las acciones. Los interfaces son conjuntos de métodos con nombres predefinidos y acciones acordadas que son conceptualmente cercanos y contienen todo lo necesario para reproducir música o buscar texto. La implementación de BT HDP en BlueZ se compone de tres interfaces DBus principales:

- **org.bluez.HealthManager**. Este es el interfaz principal a través del cual se registra una aplicación gestora de BT HDP. El método *CreateApplication* registra la nueva aplicación aceptando un objeto de tipo *Dictionary* como parámetro donde se indica el comportamiento (*sink* o *source*), el *data type*, tipo de canal (fiable o *streaming*), etc.
- **org.bluez.HealthDevice**. Representa al dispositivo médico remoto a través del cual se pueden crear o destruir canales o recibir eventos de conexión o desconexión de canal.

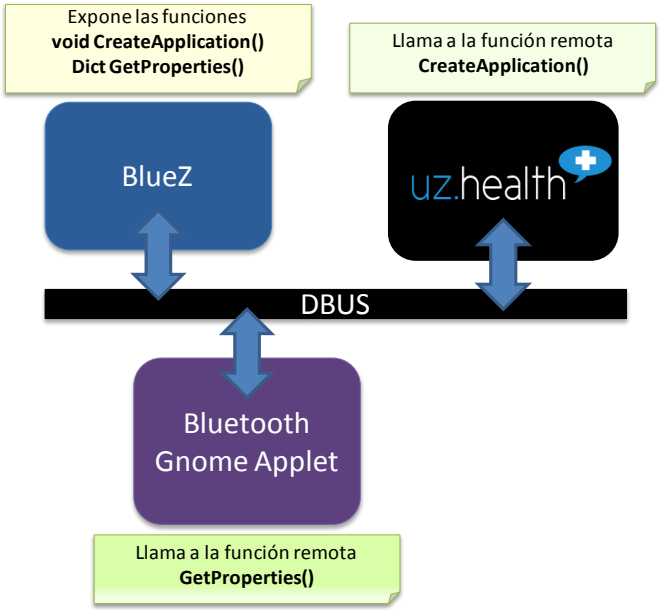



Fig. 6. Ejemplo de conexiones de DBus.

- **org.bluez.HealthChannel**. Representa un canal de datos y a través de este interfaz se obtiene el *stream* sobre el que se intercambia toda la información X73PHD.

La implementación propuesta utiliza un *wrapper* C# sobre DBus para utilizar BT HDP a través de BlueZ. El perfil médico se ha incorporado en la pila BlueZ a partir de la versión 4.71, pero es recomendable utilizar una versión más actual dado el estado semi-experimental de las primeras implementaciones. De la misma forma es necesaria, como mínimo, la versión 1.4.0 de DBus, a partir de la cual se ha incluido la característica "*Unix FD passing*" necesaria para obtener el *stream* de datos desde DBus.

## V. Integración con los servicios de usuario

Siguiendo la arquitectura de abstracción de capas, en este apartado se analiza la integración de servicios de usuario. Hasta ahora se ha descrito el núcleo de la plataforma, la capa de comunicaciones y la integración del perfil médico BT HDP. Para esta última capa de la arquitectura, se propone una prueba de concepto incluyendo servicios de valor añadido realizando un acercamiento al diseño en la nube (*cloud computing*). Con este objetivo, se han integrado diversas aplicaciones (que se detallan a continuación), así como la armonización con un servidor de HCE y el desarrollo de un interfaz gráfico básico, que reúne estos servicios y ofrece un acceso intuitivo a la plataforma de telemonitorización.

- **Servicios en la nube**. Entre las nuevas tendencias tecnológicas que están aparecido en el mercado en los últimos años, *cloud computing* ofrece un modelo de pago por uso que permite un acceso cómodo y bajo demanda a un conjunto compartido de recursos informáticos configurables como redes, servidores, almacenamiento, aplicaciones y servicios, que se puede desplegar y utilizar de forma rápida y fácil. El modelo de *cloud computing* ofrece varios tipos generales de servicios: infraestructura (disponibilidad de capacidad de almacenamiento, procesamiento y de red que se factura según el consumo), plataforma (entorno de

desarrollo y herramientas y servicios asociados que se ofrece a los clientes para crear sus propias aplicaciones), y *software* (suministro de aplicaciones que se ofrece en una red y no precisa que los usuarios lo instalen en sus propios ordenadores), entre otros. Este *software como servicio* (*Software as a Service*, SaaS) es, sin duda, el más frecuente en la nube y el que se ha integrado en la arquitectura propuesta. Este modelo de distribución de *software* proporciona a los clientes el acceso a aplicaciones a través de internet, de manera que el usuario no tiene que preocuparse de su mantenimiento. Este modelo permite, para el usuario, optimizar costes y recursos y, para el suministrador (centro de salud, hospital, etc.), implementar economías de escala optimizando costes. La arquitectura propuesta integra *Google Apps* (ver Fig. 7) para ofrecer servicios de valor añadido como alertas, calendario de eventos, recordatorio de tareas (*Calendar*) o mensajería de apoyo mediante aplicaciones de correo electrónico (*Gmail*), mensajería instantánea (*Google Talk*), documentos compartidos (*Google Docs*), gestión de información médica (*Google Health*), entre otros.

- **Servidor de HCE**. En las secciones anteriores se ha analizado X73PHD como el estándar internacional para interoperabilidad de dispositivos médicos. Este es el primer paso para llegar a una plataforma totalmente interoperable pero, para lograr niveles óptimos en la calidad asistencial y continuidad de cuidado de un paciente, es necesario interactuar con el HCE del paciente para el seguimiento y autocontrol de su salud. Así, como se adelantó en la introducción, X73PHD cubre la comunicación en el interfaz PAN y UNE-EN/ISO 13606 define el intercambio interoperable de HCE. Pero, hasta el momento, no se ha definido un estándar específico para el interfaz WAN entre el manager y el servidor de HCE. Sin embargo, se han propuesto algunas iniciativas, lideradas por *Continua Health Alliance* e IHE, basadas en los perfiles *Device to Enterprise Communication* (DEC, llamado PCD-01) [21], *Subscribe to Patient Data* (llamado PCD-02) [22], IHE *Cross-Enterprise Document Reliable Interchange* (XDR) [23] y, sobre éste último, el documento HL7 *Personal Health Monitoring* (PHM) [24]. Estas propuestas de IHE y *Continua Health Alliance* no implican la definición de nuevos estándares *ad-hoc*. Al contrario, impulsan algunos perfiles y procedimientos de otros estándares, esencialmente HL7 (IHE impulsa mensajes HL7 v2.6 y *Continua Health Alliance* hace uso del perfil HL7-PHM).

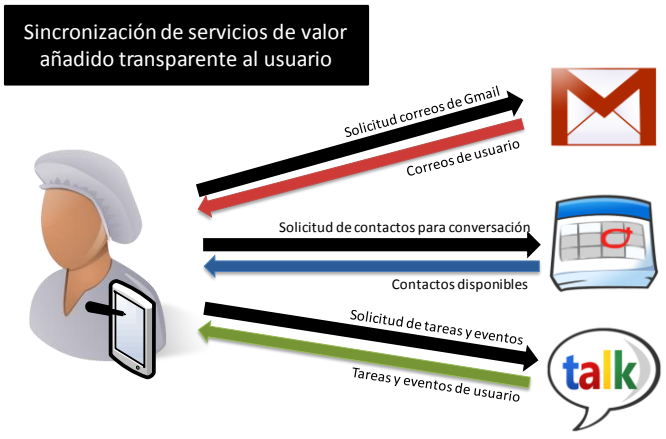



Fig. 7. Servicios en la nube integrados en la arquitectura propuesta.

En la arquitectura propuesta y debido a que la implementación está basada en X73PHD y UNE-EN/ISO 13606, estos enfoques basados en HL7 no son la opción más adecuada por lo que se ha diseñado una arquitectura WS apoyada en documentos XML. Estos documentos XML satisfacen los particulares requerimientos de las implementaciones X73PHD y UNE-EN/ISO13606 para mantener los requisitos de interoperabilidad, seguridad, fiabilidad y privacidad. El contenido y estructura XML depende de un fichero de configuración, personalizado para cada usuario, obtenido del servidor de HCE, y configurado en colaboración con especialistas médicos y a partir de análisis previos de casos de uso [25]. Como muestra Fig. 8, este XML incluye información específica de los pacientes (*idCollecter*), sus dispositivos asociados (*deviceInfo*), el procedimiento de medida (*timeStamp*), y otra información técnica como tipo de dispositivo (*mdc_ attr_id_type*), modelo (*mdc_attr_id_model*), niveles de batería (*mdc_attr_val_batt_charge*), etc.

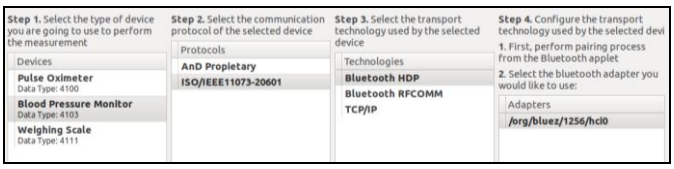
- **Aplicaciones de usuario**. La implementación se completa con el desarrollo de un interfaz como prueba de concepto (ver Fig. 9), pendiente de ser ampliado para incluir funcionalidades de usabilidad, robustez y e-accesibilidad. Se ha construido en GTK# y Mono y su uso está orientado a Linux aunque la portabilidad a entornos Windows es inmediata. En Fig. 9(a) se observa la pantalla principal, donde se presentan las opciones más habituales para el usuario. En la parte superior izquierda se localiza la zona de notificaciones donde se avisará al usuario si tiene algún correo electrónico, alguna tarea pendiente o un mensaje de *chat*.

```
<collecter>
  <idCollecter>1898721281</idCollecter>
  <soc>
    <idPatient>PacTry001</idPatient>
    <deviceReport>
      <deviceInfo>
        <attribute>
          <id>MDC_ATTR_SYS_ID</id>
          <value>00A0960D7B58</value>
        </attribute>
        <attribute>
          <id>MDC_ATTR_SYS_TYPE</id>
          <value>MDC_DEV_SPEC_PROFILE_SCALE</value>
        </attribute>
      </deviceInfo>
      <measurementInfo>
        <timeStamp>28/02/2011 14:05:28</timeStamp>
        <measurement>
          <attribute>
            <id>MDC_ATTR_ID_TYPE</id>
            <value>MDC_MASS_BODY_ACTUAL</value>
          </attribute>
          <attribute>
            <id>MDC_ATTR_NU_VAL_OBS_SIMP</id>
            <value>89</value>
          </attribute>
          <attribute>
            <id>MDC_ATTR_UNIT_CODE</id>
            <value>KILO_G</value>
          </attribute>
        </measurement>
      </measurementInfo>
    </deviceReport>
  </soc>
</collecter>
```

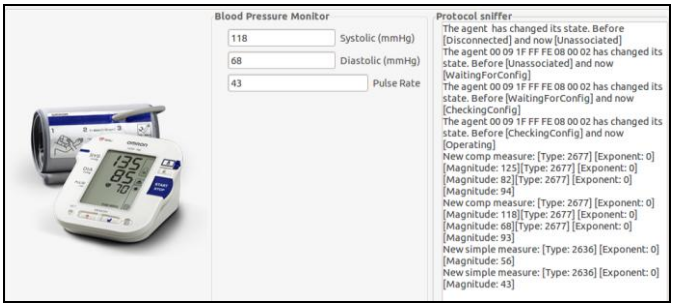
Fig. 8. Esquema XML para envío de datos X73PHD al servidor de HCE.

En la parte inferior izquierda, se localizan los botones para poder contactar con el centro de salud, consultar el HCE o realizar una medida manual que no esté planificada. Además, en la parte superior central, se han incluido una serie de pestañas bajo las que se distribuyen los servicios de usuario: tareas y eventos, correo, mensajes instantáneos y contactos. En Fig. 9(b), se presenta la pantalla de selección de parámetros para el proceso de adquisición de signos vitales desde los dispositivos médicos. Este proceso sigue cuatro pasos: seleccionar el tipo de dispositivo, seleccionar el protocolo de comunicaciones del dispositivo, seleccionar la tecnología de transporte usada por el dispositivo y, en caso de que fuera necesario, configurar los parámetros de la tecnología de transporte. En Fig. 9(b) se muestra un ejemplo de selección de un tensiómetro X73PHD desde un *Tablet* PC mediante tecnología BT HDP. Finalmente, en la última pantalla mostrada en Fig. 9(c), se observa el procedimiento completo de medida conforme al modelo de comunicación X73PHD. En la columna lateral derecha, a modo didáctico, se ha colocado un analizador de protocolo pudiendo observar los diferentes estados por los que pasa el núcleo X73PHD. Una vez completada la medida, ésta se enviará al servidor de HCE en formato el XML siguiendo el esquema detallado en el apartado anterior (ver Fig. 8).


(a) Pantalla principal de la plataforma de telemonitorización 100% estándar.




(b) Pantalla de selección de parámetros para adquisición de medidas.




(c) Ejemplo de medida de presión arterial sobre perfil médico BT HDP.

Fig. 9. Interfaz gráfico implementado como prueba de concepto.

## VI. Conclusión

En este artículo se ha propuesto el diseño de una solución de telemonitorización de pacientes 100% estándar, basada en ISO/IEEE 11073 e UNE-EN/ISO 13606, según una arquitectura de capas de abstracción. La implementación de la arquitectura incluye el nuevo perfil médico Bluetooth HDP y se ha integrado con un servidor de HCE y servicios de usuario basados en la nube. Algunos de los retos fundamentales que deben afrontar las futuras versiones serán la e-Accesibilidad y usabilidad, promoviendo su penetración en el mercado final y posibilitando su transferencia al sistema sanitario.

## Agradecimientos

Los autores quieren agradecer a Santiago Carot-Nemesio del proyecto Morfeo OpenHealth (Universidad Rey Juan Carlos I) por su asesoramiento técnico en esta contribución. Este trabajo ha sido parcialmente subvencionado por los proyectos TIN2008-00933/TSI de la Comisión Interministerial de Ciencia y Tecnología (CICYT) y Fondos Europeos para el Desarrollo Regional (FEDER), TSI-020100-2010-277 y TSI-020302-2009-7/Plan Avanza I+D del Ministerio de Industria, Turismo y Comercio, PI029/09 del Gobierno de Aragón y la Beca Programa Europa XXI de estancias de investigación (CAI-CONAI+D) a A. Aragüés (Ref. IT 23/10).

## Referencias

[1] ISO/IEEE11073 - Personal Health Devices standard (X73PHD). Health Informatics. [P11073-00103. Technical report-Overview] [P11073-104zz. Device specializations] [P11073-20601. Application profile – Optimized exchange protocol]. *http://standards.ieee.org*. [04/11].
[2] Continua Health Alliance. *www.continuaalliance.org*. [04/11].
[3] Integrating the Healthcare Enterprise (IHE). *www.ihe.net*. [04/11].
[4] Personal Health Devices Working Group (PHDWG). IEEE Standards. *http://standards.ieee.org/PHDworkgroup/*. [04/11].
[5] Universal Serial Bus Device Class (USB PHDC) release 1.0. *USB Implementers Forum Inc.http://www.usb.org*. [04/11].
[6] Bluetooth Health Device Profile (BT HDP) v1.0 rev00 Bluetooth Special interest Group (SIG). *http:// www.bluetooth.com*. [04/11].
[7] ZigBee Health Care Profile (ZHC) specification version 1.0 revision 15. *ZigBee Alliance. http://www.zigbee.org*. [04/11].
[8] ISO/EN13606 - CEN/TC251. EHR Communication Standard. Parts 1-5. *http://www.medicaltech.org*. [04/11].
[9] I. Martínez *et al*. "Implementación integrada de plataforma telemática basada en estándares para monitorización de pacientes". *Jornadas de Ingeniería Telemática* (JITEL), pp. 505-512, 2007.
[10] I. Martínez *et al*. "Optimización de una plataforma telemática para monitorización de pacientes orientada a u-Salud y basada en estándares". *Jornadas de Ingeniería Telemática*, pp. 374-381, 2008.
[11] I. Martínez *et al*. "Plataforma Telemática de Integración de Estándares End-to-End para Salud Personal". *Jornadas de Ingeniería Telemática* (JITEL), pp. 156-163, 2009.
[12] Health Level Seven (HL7). Devices Special Interest Group. *http://www.hl7. org/Special/committees/healthcaredevices/index.cfm*. [04/11].
[13] Santiago Carot-Nemesio *et al*. "The OpenHealth FLOSS Implementation of the ISO/IEEE 11073-20601 Standard", *Open Source Software Workshop OSEHC*, 2010.
[14] Jungo BTware. *http://www.jungo.com.* [04/11].
[15] Stollmann BlueCode+. *http://www.stollmann.de.* [04/11].
[16] Toshiba Bluetooth. *http://aps2.toshiba-tro.de/bluetooth.* [04/11].
[17] Ethermind Stack. *http://www.mindtree.com.* [04/11].
[18] BlueZ. *http://www.bluez.org*. [04/11].
[19] Bluetooth Special Interest Group. *http://www.bluetooth.org* [04/11].
[20] DBus. *http://dbus.freedesktop.org/*. [04/11]
[21] Device Enterprise Communication (DEC) Profile PCD-01. IHE-PCD. *http://wiki.ihe.net/index.php?title=pcd_profile_dec_overview.* [04/11].
[22] Subscribe to Patient Data (SPD) Profile PCD-02. IHE-PCD Technical Committee. *http://www.ihe.net/Technical_Framework/upload/IHE_PCD_TF_Supplement_SPD_PC_2007-07-18.pdf* . [04/11].
[23] IHE-XDR. *http://wiki.ihe.net/index.php?title=Cross enterprise_ Document_Reliable_Interchange*. [04/11].
[24] HL7 – PHM. *http://www.hl7.org/special/Committees/projman/searchable ProjectIndex.cfm?action=edit&ProjectNumber=209.* [04/11].
[25] I. Martínez *et al.*, "Recent innovative advances in telemedicine: standard-based designs for personal health", *IJBET*, 2010.