Universidad Politécnica de Madrid

Escuela Técnica Superior de Ingeniería
y Sistemas de Telecomunicación

CONTRIBUTION TOWARDS INTELLIGENT SERVICE MANAGEMENT IN WEARABLE AND UBIQUITOUS DEVICES

DOCTORAL THESIS

Pedro Castillejo Parrilla
Master in Engineering Systems and
Accessible Services for the Information Society

2015

Centro de Investigación en Tecnologías de Software y

Sistemas Multimedia para la Sostenibilidad

Escuela Técnica Superior de Ingeniería y Sistemas de Telecomunicación

CONTRIBUTION TOWARDS INTELLIGENT SERVICE MANAGEMENT IN WEARABLE AND UBIQUITOUS DEVICES

DOCTORAL THESIS

Pedro Castillejo Parrilla

Master in Engineering Systems and

Accessible Services for the Information Society

Supervisors:

Prof. PhD. José-Fernán Martínez Ortega

Prof. PhD. Lourdes López Santidrián

Universidad Politécnica de Madrid

2015

Contents

POLITÉCNICA

List of Figures

POLITÉCNICA

List of Tables

POLITÉCNICA

Acknowledgments

The thesis that I have finished today is thanks to the help and assistance of a large number of people. It is very hard to enumerate them all, but I will do my best.

First, I would like to thank Prof. Dr. José-Fernán Martinez and Prof. Dr. Lourdes López for their assistance and guidance during all my post-graduate years and the writing of this thesis. Prof. Lourdes López gave me the first opportunity to get in contact with the research world some years ago and if I have ended up writing a thesis is thanks to that first opportunity. Prof. José-Fernán encouraged me to continue my post-graduate studies enrolling in the PhD. program, and allowed me to be part of a great research group. I really appreciate the confidence that they place in my everyday work.

The results presented here have fructified thanks to the hard work of this great group. I would like to thank all the present and former members of the *Next Generation Networks and Services (GRyS)* research group. I would like to thank specially Marta Zuazúa, my first colleague in this research adventure and Jesús Rodriguez, the last but not least one. With Jesús I have shared not only work, but also friendship and hilarious discussions about life. I would also like to thank the researchers Alexandra Cuerva, Esther Moreno, Carlos Estévez, David Gómez, Néstor Lucas and my colleague Yuanjiang Huang (I will miss talking to you in the nearby desk about our thesis). I would also like to thank all the professors in the group, but especially Prof. Vicente Hernández for always being helpful, available to talk and provide advice, and Prof. Juana Sendra for the assistance with the mathematical models.

During three months I made part of my thesis work at Norway. I would like to thank everyone who helped me at NTNU University, especially Prof. Dr. Sverre Hendseth for inviting me to visit his department and Dr. Geir Mathisen for his guidance. Thanks also to Roberto F. Lopes for his support and the coffees and moments we shared talking about living abroad.

Last, but not least, I would like to thank all the people outside my research world who has helped me during this process. Thanks to my parents, my sister, my girlfriend and all my family and friends.

Thank you all.

POLITÉCNICA

Abstract

Nowadays, society is shifting towards a growing interest and concern on health care. This phenomenon can be acknowledged by two facts: first, the increasing number of people practising some kind of healthy activity (sports, balanced diet, etc.). Secondly, the growing number of commercial wearable smart devices (smartwatches or bands) able to measure physiological parameters such as heart rate, breathing rate, distance or consumed calories.

A large number of applications combining both facts are appearing. These applications are not only able to monitor the health status of the user, but also to provide recommendations about routines in order to improve the mentioned health status. In this context, wearable devices merged with the Internet of Things (IoT) paradigm enable the proliferation of new market segments for these health wearable-based applications. Furthermore, these applications can provide solutions for the elderly or baby care, in-hospital or in-home patient monitoring, security and defence fields or an unforeseen number of future applications.

The introduced IoT paradigm can be developed with the usage of existing Wireless Sensor Networks (WSNs) by connecting the novel wearable devices to them. In this way, the migration of new users and actors to the IoT environment will be eased. However, a major issue appears in this environment: heterogeneity. In fact, there is a large number of operating systems, hardware platforms, communication and application protocols or programming languages, each of them with unique features.

The main objective of this thesis is defining and implementing a solution for the intelligent service management in wearable and ubiquitous devices so as to solve the heterogeneity issues that are presented when dealing with interoperability and interconnectivity of devices and software of different nature. Additionally, a security schema based on trust domains is proposed as a solution to the privacy problems arising when private data (e.g., biomedical parameters or user identification) is broadcasted in a wireless network.

The proposal has been made after a comprehensive state-of-the-art analysis, and includes the design of a Wearable Device Service Bus (WDSB) including the technologies collected in the requirement analysis (ESB, WWBAN, WSN and IoT).

Applications are able to access the WSN services regardless of the platform and operating system where they are running. Besides, this proposal also includes the design of a Wearable Inter-Domain communication Protocols set (WIDP) which integrates lightweight protocols suitable to be used in low-capacities devices (REST, JSON, AMQP, CoAP, etc...). Furthermore, a security solution for service management based on a trustworthy domains model to deploy security services in WSNs has been designed.

Although the proposal is a generic framework for applications based on services provided by wearable devices, an application scenario for testing purposes has been included. In this validation scenario it has been presented an autonomous physical condition performance system, based on a WSN, bringing the possibility to include several elements in an IoT scenario: a smartwatch, a physiological monitoring device and a smartphone.

In summary, the general objective of this thesis is solving the heterogeneity and security challenges arising when developing applications for WSNs and wearable devices. As it has been presented in the thesis, the solution proposed has been successfully validated in a real scenario and the obtained results were satisfactory.

Resumen

Hoy en día asistimos a un creciente interés por parte de la sociedad hacia el cuidado de la salud. Esta afirmación viene apoyada por dos realidades. Por una parte, el aumento de las prácticas saludables (actividad deportiva, cuidado de la alimentación, etc.). De igual manera, el auge de los dispositivos inteligentes (relojes, móviles o pulseras) capaces de medir distintos parámetros físicos como el pulso cardíaco, el ritmo respiratorio, la distancia recorrida, las calorías consumidas, etc.

Combinando ambos factores (interés por el estado de salud y disponibilidad comercial de dispositivos inteligentes) están surgiendo multitud de aplicaciones capaces no solo de controlar el estado actual de salud, también de recomendar al usuario cambios de hábitos que lleven hacia una mejora en su condición física. En este contexto, los llamados dispositivos llevables (*weareables*) unidos al paradigma de Internet de las cosas (IoT, del inglés *Internet of Things*) permiten la aparición de nuevos nichos de mercado para aplicaciones que no solo se centran en la mejora de la condición física, ya que van más allá proponiendo soluciones para el cuidado de pacientes enfermos, la vigilancia de niños o ancianos, la defensa y la seguridad, la monitorización de agentes de riesgo (como bomberos o policías) y un largo etcétera de aplicaciones por llegar.

El paradigma de IoT se puede desarrollar basándose en las existentes redes de sensores inalámbricos (WSN, del inglés *Wireless Sensor Network*). La conexión de los ya mencionados dispositivos llevables a estas redes puede facilitar la transición de nuevos usuarios hacia aplicaciones IoT. Pero uno de los problemas intrínsecos a estas redes es su heterogeneidad. En efecto, existen multitud de sistemas operativos, protocolos de comunicación, plataformas de desarrollo, soluciones propietarias, etc.

El principal objetivo de esta tesis es realizar aportaciones significativas para solucionar no solo el problema de la heterogeneidad, sino también de dotar de mecanismos de seguridad suficientes para salvaguardad la integridad de los datos intercambiados en este tipo de aplicaciones. Algo de suma importancia ya que los datos médicos y biométricos de los usuarios están protegidos por leyes nacionales y comunitarias.

Para lograr dichos objetivos, se comenzó con la realización de un completo estudio del estado del arte en tecnologías relacionadas con el marco de investigación (plataformas y estándares para WSNs e IoT, plataformas de implementación distribuidas, dispositivos llevables y sistemas operativos y lenguajes de programación). Este estudio sirvió para tomar decisiones de diseño fundamentadas en las tres contribuciones principales de esta tesis: un bus de servicios para dispositivos llevables (WDSB, Wearable Device Service Bus) basado en tecnologías ya existentes tales como ESB, WWBAN, WSN e IoT); un protocolo de comunicaciones inter-dominio para dispositivos llevables (WIDP, Wearable Inter-Domain communication Protocol) que integra en una misma solución protocolos capaces de ser implementados en dispositivos de bajas capacidades (como lo son los dispositivos llevables y los que forman parte de WSNs); y finalmente, la tercera contribución relevante es una propuesta de seguridad para WSN basada en la aplicación de dominios de confianza.

Aunque las contribuciones aquí recogidas son de aplicación genérica, para su validación se utilizó un escenario concreto de aplicación: una solución para control de parámetros físicos en entornos deportivos, desarrollada dentro del proyecto europeo de investigación “LifeWear”. En este escenario se desplegaron todos los elementos necesarios para validar las contribuciones principales de esta tesis y, además, se realizó una aplicación para dispositivos móviles por parte de uno de los socios del proyecto (lo que contribuyó con una validación externa de la solución). En este escenario se usaron dispositivos llevables tales como un reloj inteligente, un teléfono móvil con sistema operativo Android y un medidor del ritmo cardíaco inalámbrico capaz de obtener distintos parámetros fisiológicos del deportista. Sobre este escenario se realizaron diversas pruebas de validación mediante las cuales se obtuvieron resultados satisfactorios.

1 Introduction

1.1. Motivation

A growing number of people are taking care about their health and wellness every day. The number of persons practising any kind of sport (running, cycling, or just walking on a daily basis) increases every year, and the industry of sports and wellness carries on increasing its importance. Therefore, health condition has become now one of the most important concerns for society. Mobile devices, along with an ever-increasing number of wearable devices, are being offered by manufacturers as a way of coping with this “health-fever” that nevertheless seems to go beyond just a short-lived trend and projects itself in the long term. Today, it is possible for a person to go out for running or jogging with a smartphone and/or a wearable sensor (a smartwatch, a Bluetooth device sensing the user´s heart rate or a smart-sport-shoe with embedded pace sensors). This allows the end user not only to control and monitor their exercise, but also to share their performance results with relatives or friends in their most appealing social networks, thus creating a new paradigm from the gymnasium facilities, which must adapt themselves to the challenges resulting from the ability of single persons to obtain and share data at a fast face without requiring an expensive infrastructure.

Furthermore, modern society is looking for user-friendly health aiding systems, able not only to remotely monitor the health of the elderly and people suffering from chronic diseases, but also to find a safe and efficient routine to practice some sport or specific exercises in an outdoor or indoor environment, in order to improve each person’s level of fitness and health. As it can be inferred, these systems are profuse in sensors and actuators and introduce sportsmen and women into the field of computer science, distributed systems and embedded devices. More importantly, it is in this context where wearable Internet of Things (IoT)-based systems are able to bring solutions and offer new services to their users. In practice, the extension of the Internet of Things into the area of sport and healthcare will be made possible by the usage of sensors and hardware devices that capture information with minimal to no intervention from the users, which will be left to focus on performing their own programmed routines.

One of the ways of spreading the IoT paradigm to these areas can be using Wireless Sensor Networks (WSNs) as the leading technology to acquire and manage data. Connecting other smart elements to a WSN (phones, watches, tablets, etc.) may also improve the user experience within the internet of Things paradigm, and it could act as a starting point for the use of a technology that would, in the end, be progressively accepted and utilized by people with little understanding regarding computer networking or electronics. If the smart devices are wearable, the first technology-access barrier is broken: the user has just to “wear” the technology as a daily-life garment.

The usage of wearable sensors to obtain values of useful human body parameters is nowadays a reality, partially due to the deployment of Wireless Sensor Networks as the backbone of the new emerging systems, which can be used as end-to-end deployments that rely on information collected by WSN that will be use by the high level applications. A WSN is a network composed of small autonomous devices (sensor nodes or motes) that either incorporates sensors or includes the ability to incorporate them, providing capabilities for monitoring both individually and cooperatively physical or environmental conditions (such as temperature, humidity, light or motion) in each of the possible locations of every node. Some fields of application for these networks include environmental control, industrial control, automotive, medicine, defence and security industries, home automation, and so forth. Miniaturization enhancements provide a new generation of tiny sensors that can be embedded in wearable devices to provide helpful data for a wide range of applications.

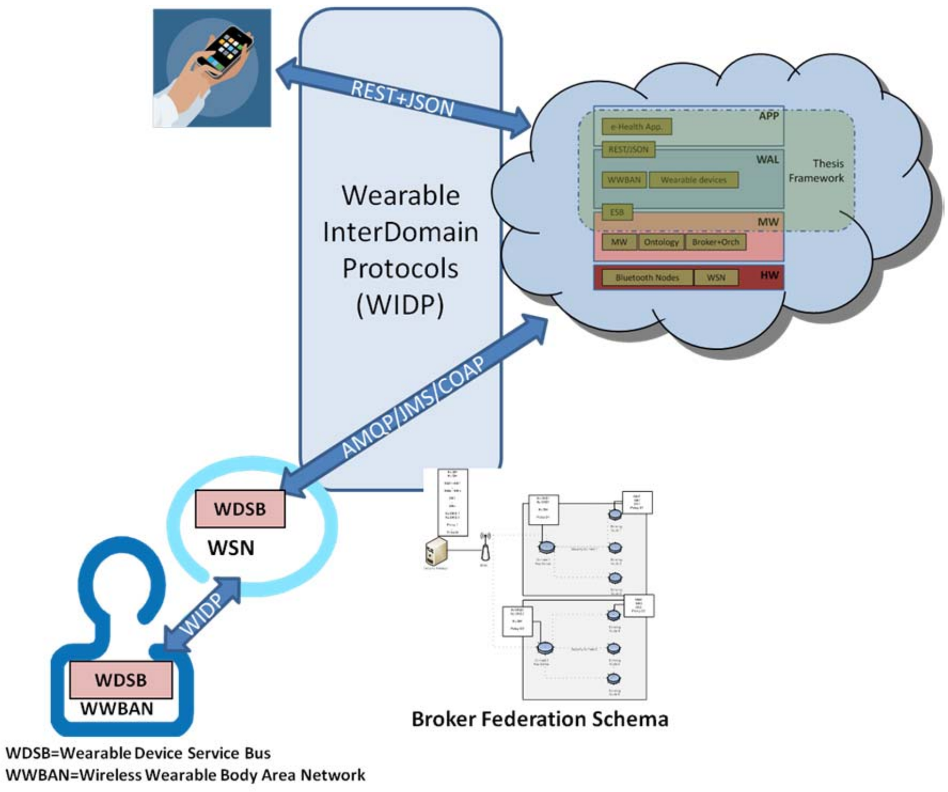


Figure 1 General overview of the thesis-related technologies

A differential value of a WSN node is the fact that any external sensor can be connected in an easy way to itself, either by using a plug on the board, soldering it or using available pins from the node pinout; additionally, the data sensing does not depend on the network management, so information can be gathered regardless of the network topology or the devices that are used to guarantee interconnectivity. For example, if an application needs biometric or human physiological parameters (such as blood pressure, heart rate, breathing rate, etc.) an external sensor can be connected to the node by means of the procedures previously described. A very easy and fast solution is using wireless communications protocols, such as Bluetooth or ZigBee. However, in this scenario a new challenge springs up: the existence of different types of devices or platforms, manufactured by vendors with different know how and strategic interests (as there is no standardization in this kind of sensors), so it is desirable to abstract the complexity and heterogeneity of hardware features and protocols from high-level layers. This can be done with an intermediate level, called *middleware*. In this way, it comes in very handy if this middleware level is able both to process environmental data and receive measurements regarding user parameters with several sensors: localization, speed, health status, preferences, etc.

High level applications using several hardware components introduce the problem of heterogeneity in the network. In order to integrate different hardware platforms as well as different middleware solutions into a single framework, the use of an Enterprise Service Bus (ESB) is put forward as a bridge for guaranteeing interoperability and integration of the different environments (as it will effectively become the cornerstone of the intermediation architecture located right between the hardware devices and network communications and the application-reliant tier), thus introducing a semantic added value needed in the world of IoT-based systems. This approach places all the data acquired (e.g., via Internet data access) at application developers disposal, opening the system to new user applications. The user can then access the data through a wide variety of devices (smartphones, tablets, computers) and Operating Systems (Android, iOS, Windows, Linux, etc.).

As a result, it can be concluded that one of the key issues arising in this field will be the management of heterogeneity in hardware devices and software modules. In fact, it is expected that there will be heterogeneity at the hardware layer (WSN nodes, wearable devices, protocols, etc.) as well as in the application layer (devices, operating systems, protocols, etc.). The contributions made in this thesis are focused on solving these heterogeneity issues and proposing integration mechanisms to ease the wearable and IoT-based application development.

Since a large number of wearable devices will be integrated in WSNs using wireless communications, security challenges are greater than in wired networks because of the open access to data transmissions and the less costly procedures required to attack a wireless scenario. By using broadcast transmissions, other devices listening in the same frequency may intercept every communication between two nodes. Another security challenge is the possibility of a node being captured and crypto-analyzed by a third party. If security keys, policies, and other cryptographic elements are accessed, a new spurious node could be introduced in the network, and several attacks can be deployed: denial-of-service, man-in-the-middle, data sniffing and/or data modification, routes spoofing, and so on.

Finally (and as a real application and validation scenario), a Wireless Sensor Network is presented in this thesis as the foundation of a sports-related scenario using wearable devices. Gathering environmental and physiological data from sportsmen/women and storing a user´s profile can be turned into an autonomous physical condition performance system, where the preferences and needs of every single user are evaluated to obtain safe and optimum exercise routines. This validation scenario has three key components. The first one deals with the integration of several wearable devices in the Internet of Things world. In order to integrate these devices (which are provided with Bluetooth-only wireless connectivity), a dual-protocol WSN/Bluetooth node is implemented, which will act as an intermediary agent between the sportsman/woman network of devices (easily becoming a Body Area Network or BAN) and the Wireless Sensor Network. In this scenario, two of these nodes are used: one is connected to the wearable health-data monitor, and another node is connected to the smartphone or smartwatch. With these nodes, all the information from/to the wearable devices can be managed in the same way as the information from other WSN nodes. Any new wearable device can be included in the system -with the only limitation of being enabled with Bluetooth protocol interfaces- and the services offered by this new device can be discovered and used as well. The second key component is an ontology, included within a service-oriented semantic middleware, to model the services provided by the WSN. With the semantically annotated services it is possible to compose new services based on the existing single services, widening the platform for future applications. The third key component is the integration of an Enterprise Service Bus (ESB) in a WSN for an IoT-based application. With the ESB, all the services published by the WSN nodes (including the services available at the wearable devices, such as heart rate, body temperature, etc.) become available for third-party applications. Should these applications connect to the ESB via REpresentational State Transfer (REST) interfaces, services and data will be able to be accessed by using and Uniform Resource identifier (URI) to interact the ESB. Consequently, if the ESB is connected to the Internet, any external application can use the RESTFul API to make requests to the published services.

1.2. Objectives

The main objective of this thesis is to define and implement a solution for the intelligent service management in wearable and ubiquitous devices so as to solve the heterogeneity issues that are presented when dealing with interoperability and interconnectivity of devices and software of different nature. To achieve this main goal, several secondary objectives have been defined:

- A comprehensive state-of-the-art analysis, including:
  - Existing solutions
  - Beyond state-of-the-art contributions selection
- Design of a Wearable Device Service Bus (WDSB) based on the following technologies:
  - ESB
  - Wearable Wireless Body Area Network (WWBAN)
  - WSN and IoT
- Design of a Wearable Inter-domain communication protocols set (WIDP)
  - Based on REST, JSON, AMQP, CoAP
  - Includes the extension of an existing service ontology to provide definitions for wearable devices and service composition.
- Proposal of a security solution for the service management
  - Including a broker federation schema for resilience
- Validation and Results, including:
  - Scenario selection
  - Tests execution
  - Results

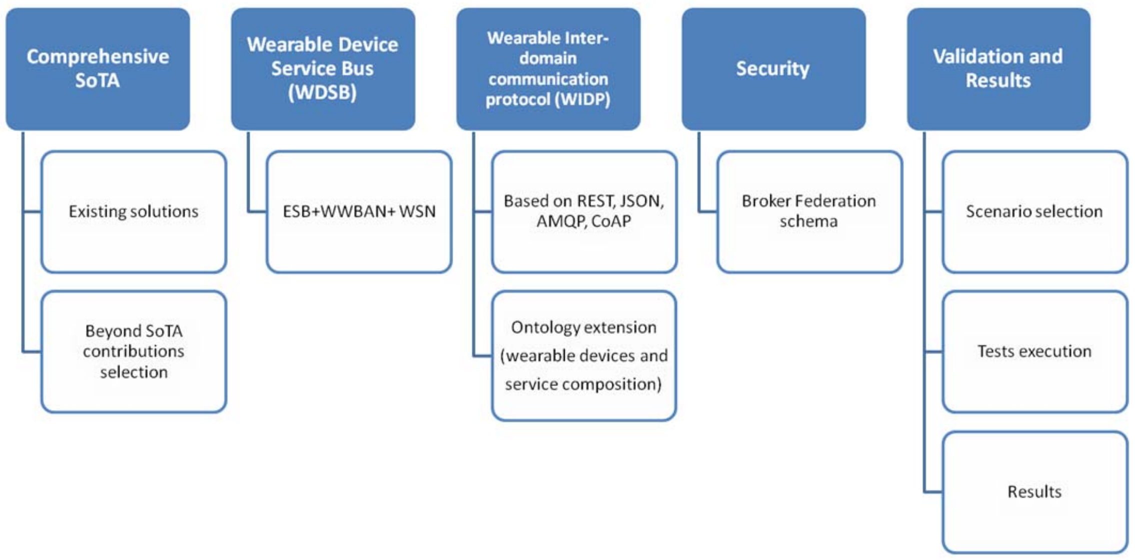

Comprehensive SoTA
Existing solutions
Beyond SoTA contributions selection
Wearable Device Service Bus (WDSB)
ESB+WWBAN+ WSN
Wearable Inter-domain communication protocol (WIDP)
Based on REST, JSON, AMQP, CoAP
Ontology extension (wearable devices and service composition)
Security
Broker Federation schema
Validation and Results
Scenario selection
Tests execution
Results

Figure 2 Thesis objectives

1.3. Thesis Framework and background

The facilities available as part of the Internet of Things -sensors, motes, Wireless Sensor Networks (WSNs), semantic middleware architecture, ontologies, etc.- have expanded significantly and, as a consequence, the number of applications based on the IoT has boomed as well. These applications have some unique characteristics: they are autonomous in their data capture patterns, have event transferring capabilities and provide strong interoperability or network connectivity. Due to these specific features (ubiquity, pervasiveness, downsizing or even miniaturization of components, etc.) researchers and engineers are constantly pushing the boundaries of technology, with applications ranging from smart vehicles using computing, communications and automation technologies to provide data to monitoring systems in transmission lines belonging to smart grids, among other applications like livestock traceability, airport anti-invasion, digital home or logistics supply chains. E-Health and environment monitoring, additionally, have achieved a high level of maturity as far as the Internet of Things is concerned; there are applications capable of using semantic engines related with medical data or fire monitoring, providing critical support in a flexible way for any kind of scenario regarding connectivity and interoperability among different actors in varied scenarios.

The possibilities that the Internet of Things developments offer are often used as feedback for previously unforeseen applications, thus making possible the design of new applications of more significant complexity and utility. Considering the present and future importance of this field, an application has been developed and conceived for its usage in scenarios where monitoring not only is applied to environmental parameters but also people. In this application the user, and the place they are performing their activities in, are seamlessly integrated as a simple unit where information can be extracted from; indeed, the end user is unaware of the separation between the different domains and their functionalities. There will be three subsystems conforming this application: one is involving the end where requests are done, and will be using to do so either a web browser or a Graphical User Interface (GUI) on one device -or several of them, if requests are done from multiple sites- capable of handling REpresentational State Transfer operations; it can be considered as the front-end of the whole deployment, the part that directly interacts with and end user or a system operator. A second subsystem is made by a Wireless Sensor Network that routes the requests and the responses done by the operator by employing a semantic middleware architecture, and a third subsystem is composed by a human being that will have several body parameters monitored by means of a Body Area Network. Figure 3 includes the boundaries of this thesis as well as the technologies involved in the validation scenario.

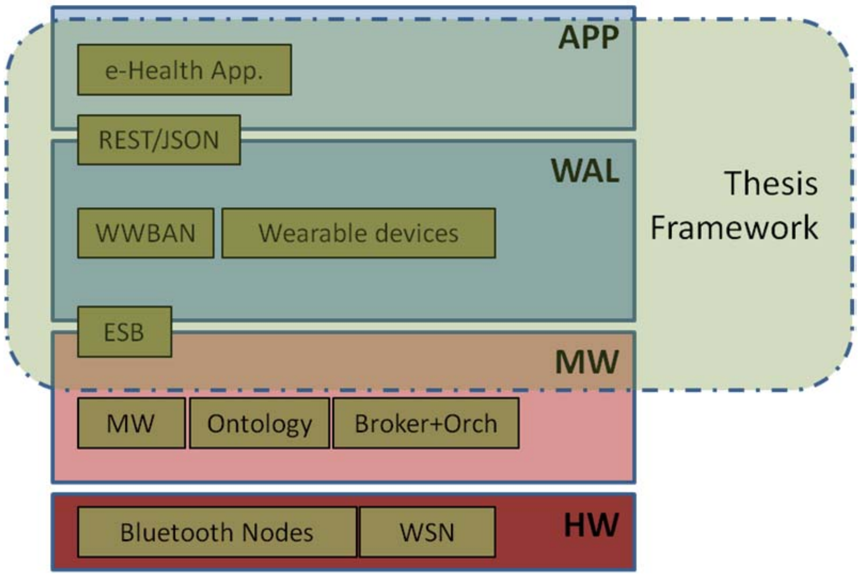


Figure 3 Thesis framework boundaries

1.4. Document structure

The first section of this document includes the introduction of the research objectives, the motivation and the framework of the thesis and the publications made by the student at indexed journals (Journal Citation Report, JCR) and international conferences.

In section two a complete State-of-the-Art analysis is presented. This study includes the most important contributions related to WSN platforms, wearable devices and security solutions.

Section three contains the dissertation and a comprehensive explanation of all the contributions made in this thesis, including: Wearable Device Service Bus, Wearable InterDomain Protocols as well as a novel security schema for WSNs.

Section four includes the validation of the contributions presented in section three. In this section, a real scenario is presented and the results of the validation are included. A security analysis is also performed in this section in order to test the strength of the security proposal.

The conclusions obtained with the work presented in this thesis are presented in section five: how the objectives of the thesis have been obtained and the overall achievements of the work done have been described in this section

Finally, section six includes references and a compendium of the acronyms used for reference of the reader.

2 State of the Art

2.1. State of the art in Wireless Sensor Networks

The ever-increasing number of both research papers and commercial solutions regarding Wireless Sensor Networks provides an overview of the real potential within these systems as part of the Internet of Things. Several years ago, Massachusetts Institute of Technology included this technology as one of the ten emerging technologies that would change the world[1].

However, while the number of commercial solutions and their applications is growing, the expansion of their usage also arise new problems related with their intrinsic limitations: power sources, limited processors, size, mobile communications, etc.

A thorough study of the recent publications and commercial solutions has been conducted, and the result is presented in the following sections.

2.1.1. Wireless communication technologies

There is a growing number of protocols and updated versions used in functional applications of Wireless Sensor Networks. The information collected by the sensors increases its value if it can be accessed outside the WSN. For this reason, connecting WSNs to Internet is a key feature. In this way, every device with an Internet connection will be able to remotely retrieve the information collected by the sensors and use these data to provide useful applications to the user, e.g. domotics, e-Health, energy saving systems, security, defence, automation, smart cities, etc.

A summary of the most relevant protocols used in these networks is presented in the next paragraphs.

2.1.1.1. Bluetooth (IEEE 802.15.1)

Bluetooth[2] is a short-range communication protocol intended in the beginning to provide wireless connectivity to cell phones in the 1990s. Although IEEE standardized the protocol (IEEE 802.15.1), currently the standard is maintained by the Bluetooth Special Interest Group, composed by more than 20.000 companies.

Bluetooth devices work in the band of 2.4GHz within the Industrial, Scientific and Medical (ISM) boundaries utilizing the frequency-hopping spread spectrum mechanism (Adaptive Frequency-Hopping performs up to 1600 hops per second). Then the band is split in channels with a bandwidth of 1 MHz (79 channels available). The first channel starts at 2402 MHz and the last finishes at 2480 MHz.

For every Bluetooth network there is a “master” device which can manage connections with up to seven “slave” devices, creating a piconet. It is possible to increase the number of devices above 8 (and also the range) by federating together several piconets with dual connected devices. This enlarged net is called a scatternet, as depicted in Figure 4.

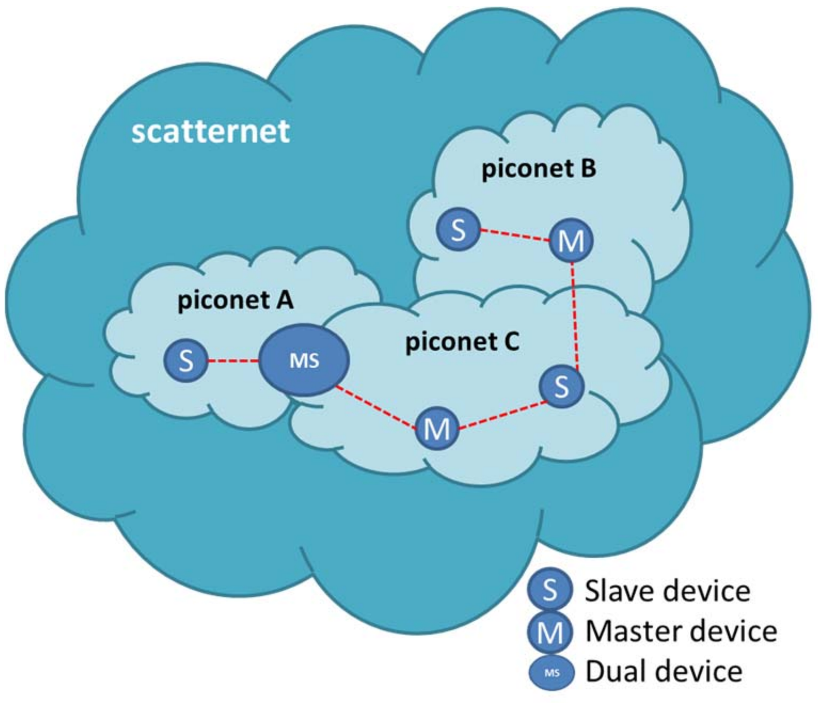


Figure 4 Bluetooth scatternet example

There is a division of Bluetooth devices based on the power transmission, as reflected in Table 1.

Table 1 Bluetooth transmitter classes

Class	Max Tx. Power in mW (dBm)	Range
Class 1	100 mW (20 dBm)	100 meters
Class 2	2.5 mW (4 dBm)	10 meters
Class 3	1 mW (0 dBm)	1 meter

Currently, Bluetooth Alliance has released 4 versions of the protocol: 1.2, 2, 3 and 4. The main features for each version are summarized in Table 2.

Table 2 Bluetooth standard versions

Version	Max. data rate	Capabilities
1.2	1 Mbit/s	Basic rate
2.0	3 Mbit/s	Enhanced Data Rate (EDR)
3.0	24 Mbit/s	EDR + High Speed (HS)
4.0	24 Mbit/s	EDR + HS + Low Energy (LE)

2.1.1.2. Wi-Fi (IEEE 802.11.a.b.h.g.n)

Wi-Fi (or WiFi) is a local area wireless technology that allows an electronic device to participate in computer networking using 2.4 GHz UHF and 5 GHz SHF ISM radio bands. The Wi-Fi Alliance defines Wi-Fi as any "wireless local area network (WLAN) product based on the IEEE 802.11 standards".

The 802.11[4] family consists of a series of half-duplex Over-The-Air modulation techniques that use the same basic protocol. Although 802.11-1997 was the first wireless networking standard in the family, 802.11b was the first widely accepted one, followed by 802.11a, 802.11g, 802.11n, and 802.11ac. Other standards in the family (c–f, h, j) are service amendments and extensions or corrections to the previous specifications.

802.11b and 802.11g use the 2.4 GHz ISM band. Because of this choice of frequency band, 802.11b and g equipment may occasionally suffer interference from microwave ovens, cordless telephones, and Bluetooth devices. 802.11b and 802.11g control their interference and susceptibility to interference by using direct-sequence spread spectrum (DSSS) and orthogonal frequency-division multiplexing (OFDM) signalling methods, respectively. 802.11a uses the 5 GHz band, which offers at least 23 non-overlapping channels rather than the 2.4 GHz ISM frequency band, where adjacent channels overlap. Better or worse performance with higher or lower frequencies (channels) may be obtained, depending on the environment.

The segment of the radio frequency spectrum used by 802.11 varies between countries. In the US, 802.11a and 802.11g devices may be operated without a license, as allowed in Part 15 of the FCC Rules and Regulations. Frequencies used by channels one through six of 802.11b and 802.11g fall within the 2.4 GHz amateur radio band. Licensed amateur radio operators may operate 802.11b/g devices under Part 97 of the FCC Rules and Regulations, allowing increased power output but not commercial content or encryption. A brief description for every version of 802.11 standard is included in Table 3.

Table 3 802.11 version features

Prot.	Release date (estim)	Fq (GHz)	BW (MHz)	Max. data rate (Mbit/s)	Modulation	Indoors range	Outdoors range
a	1999	5	20	54	OFDM	35	120
b	1999	2.4	22	11	DSSS	35	140
g	2003	2.4	20	54	OFDM, DSSS	38	140
n	2009	2.4/5	20	72.2	OFDM	70	250
ac	2013	5	20	96.3	OFDM		
ac	2013	5	20	96.3			
ad	2012	60	2,160	6.75 Gbit/s	OFDM, single carrier (SC)	60	100
ah	(2016)	0.9					
aj	(2016)	45/60					
ax	(2019)	2.4/5			OFDM		
ay	(2017)	60	8000	100Gbit	OFDM, SC	60	1000

2.1.1.3. WiMax (IEEE 802.15.16)

WiMax integrates the IEEE 802.16 family of standards and the standard HiperMan of the European standardization organism ETSI[5] . WiMAX is a wireless data transmission standard that provides concurrent access in areas of up to 50 mile radius at speeds up to 70 Mbps, supporting station mobility, MIMO technology and QoS specification for Voice over IP.

2.1.1.4. Wavenis

Wavenis[6] ultra-low-power and long-range wireless technology is currently used by millions devices around in the world, in diverse markets where communication ability and device autonomy present conflicting requirements. These markets include telemetry, industrial automation, remote utility meter monitoring, home comfort, alarms for protecting people and property, home healthcare, centralized building management, access control, cold-chain monitoring, as well as long-range UHF RFID applications for the identification, tracking, and locating of people and objects. It works in major license-free ISM bands (868, 915 and 433 MHz) and fits perfectly for low-traffic, 2-way data & M2M applications from 4.8 to 100 kbps (typically 19.2 kbps)

2.1.1.5. ZigBee (IEEE 802.15.4)

Probably the most extended protocol in WSNs is ZigBee[7]. It specifies a set of high level wireless communication protocols for reliable, cost-effective and low-power wireless networking. This specification is defined by an association of companies called the Zigbee Alliance. It is based on the IEEE 802.15.4 standard for Low Rate Wireless Personal Area Networks. The stack specification over IEEE 802.15.4 defines the network and the security layer (using 128 bit symmetric encryption keys), handling star and peer-to-peer network topologies. It also provides a framework for application programming in the application layer, connecting up to 65.000 nodes per network using the ISM 2.4 GHz unlicensed band and provides communications up to 100 meters line of sight. Zigbee protocol specifies three network topologies:

- **Star topology**: the coordinator node is located in the centre of the network.
- **Tree topology**: the coordinator node is the root of the tree.
- **Mesh topology**: where at least one of the nodes is connected with two paths.

2.1.1.6. 6LoWPan

Low-power Wireless Personal Area Networks (LoWPANs) comprise devices that conform to the IEEE 802.15.4-2003 standard. IEEE 802.15.4 devices are characterized by short range, low bit rate, low power, and low cost. Many of the devices utilising radio interfaces compliant with this standard will be limited in their computational power, memory, and/or energy availability. Key features of these networks are[3]:

1. Small packet size. Given that the larger physical layer packet possible is 127 bytes, the resulting maximum frame size at the media access control layer is 102 octets. Link-layer security imposes further overhead, which in the maximum case (21 octets of overhead in the AES-CCM-128 case, versus 9 and 13 for AES-CCM-32 and AES-CCM-64, respectively), leaves 81 octets for data packets.
2. Support for both 16-bit short or IEEE 64-bit extended media access control addresses.
3. Low bandwidth. Data rates of 250 kbps, 40 kbps, and 20 kbps for each of the currently defined physical layers (2.4 GHz, 915 MHz, and 868 MHz, respectively).
4. Topologies include star and mesh operation.

2.1.1.7. Thread

Thread[8] is an IPv6 networking protocol built on open standards, designed for low power 802.15.4 mesh networks and supported by more than 50 prominent technology companies. Existing popular application protocols and IoT platforms can run over Thread networks. Using proven standards and IPv6 technology with 6LoWPAN as its foundation, Thread offers product developers numerous technological advantages over existing wireless standards, including:

- Reliable networks: Thread offers robust self-healing mesh networks that scale to hundreds of devices with no single point of failure. Devices are ready when people need them.

- Secure networks: Thread networks feature a secure encryption mechanism. Thread closes identified security holes found in other wireless protocols and provides worry-free operating facilities.

- Simple connectivity: Thread devices are simple to install and configure using a smartphone, tablet or computer. Consumers can securely connect Thread devices in their home, to each other and to the cloud, for easy control and access from anywhere.

- Low power: Thread supports battery-operated devices as part of a home network. This allows the devices that people use every day - including thermostats, lighting controls, safety and security products - to be a part of the network without requiring constant charging or frequent battery changes.

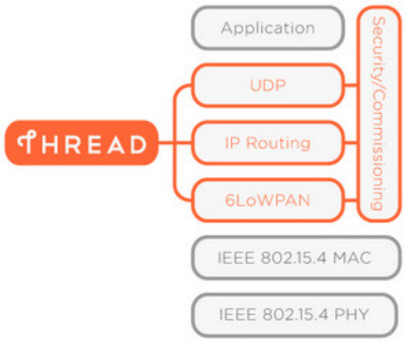


Figure 5 Thread stack. Source: Thread Group.

Table 4 includes the most relevant features for selected wireless communications technologies included in the study of the state-of-the-art, in order to ease the comparison between them. Furthermore, a growing number of routing protocols is available for WSN. Some of them are focused on energy efficiency [9-13] or high-throughput [14], while there are proposal presenting QoS-aware routing protocols [15] or position based algorithms [16].

Table 4 Most relevant features for wireless communication protocols

	Bluetooth	ZigBee (6LoWPAN)	WiFi	Wavenis
Frequency	2.4 GHz	2.4 GHz	2.4 GHz / 5.2 GHz	433/868/915MHz (2,4GHz)
Max. data transmission speed	24 Mbps	250 kbps	1 a 180 Mbps	2,4 a 250 kbps
PHY	FHSS / GFSK	DSSS	DSSS/ OFDM	FHSS / GFSK
Resilience	+++	+++	+++	+++
Low power consumption	+ BLE+++	+++	+	+++
Low cost	++	++	++	++
Indoors range	100 m.	75m	100m	up to 200m
Supported topologies	-	++	-	+++
Standardized	+++	+++	+++	++
Availability	+++	++	+++	++
Deployment	+++	+	+++	++

2.1.2. Hardware platforms

A sensor node consists of several components: a battery, a communication module, a central unit (with processor, memory, etc.) and different kinds of sensors and actuators.

Nodes can be equipped with several sensors, such as: temperature, humidity, light, pressure, chemical, biological and nuclear. One of the key features of the nodes is the possibility to upgrade them with new sensors in order to fulfil forthcoming application requirements. The main challenge when designing a WSN node is the power source limitation, since these are autonomous nodes which are not usually connected to an unlimited power source.

Figure 6 A generic WSN node

Another challenge is the existence of different types of devices and platforms: as it can be observed in the following sections, there is no standardization in this kind of sensors nodes, but it can be concluded that in general:

- these nodes are tiny devices, with limited resources
- they have a limited energy supply, although they can be powered by solar cells
- they are limited in resources regarding computational power, the amount of memory, radio bandwidth and coverage
- there are no tamper-proof zones in the commercial hardware platforms to store sensitive information (passwords, keys, identifications, etc.)
- due to exposure to adverse environmental conditions they tend to fail or lose their power, and could be removed by unauthorized personnel

A generic WSN node is composed by:

- **Power supply**: with current commercial solutions it is possible to keep alive one node up to one year long operation. Lifetime depends on the energy management as well as the workload of the node (including the number of wireless transmissions and receptions through the radio layer). Two standard AA-size batteries are able to provide enough energy to maintain a node up and running, but there are other solutions as solar power, wind, piezoelectric, etc. The power source choice will depend on the application scenario and the node requirements, but nowadays is easy to find nodes powered by ultra-capacitors, electromagnetic fields, wind, solar or kinetic (movement).

- **Microprocessor**: it is possible to find nodes equipped with CPUs running from several KHz up to MHz. The RAM memory installed can also reach up to several Mbytes.

- **Radio**: currently, several radio technologies coexist, but the most used standard among relevant suppliers is IEEE 802.15.4. Different communication bands are used, depending on the country (2.4 GHz or 868 MHz up to 250 Kbps for Europe, 915 MHz with 40 Kbps for EEUU). The range varies between 5 and 200 metres, with an average of 50 meters for commercial solutions.

- **Sensors**: nodes can be equipped with a wide range of sensors to measure a huge number of variables, such as temperature, light, humidity, chemical components, pollution, radioactivity, movement, tracking, GPS and identification (RFID).

- **Software**: nodes can be programmed using standard C language, JAVA, and proprietary languages.

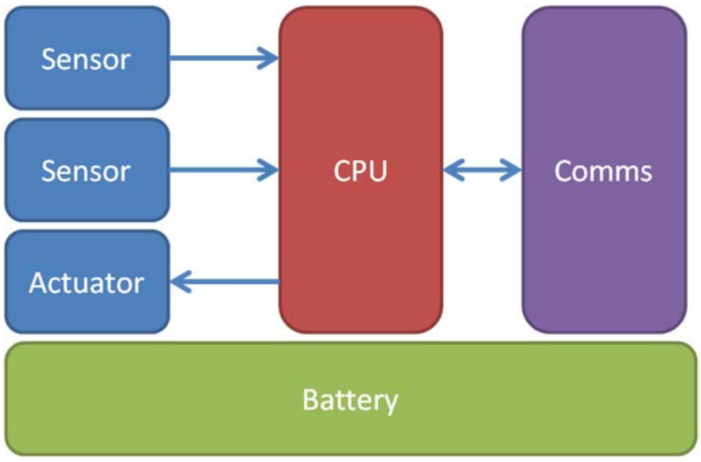


Figure 7 Generic WSN node components

The number of commercial WSN nodes is growing every year. Furthermore, the interest in developing IoT applications is leading into new devices design. The features of the platforms are being enhanced in each new version, and the prices are being reduced too.

A project to deploy a Wireless Sensor Network must be able to deal with the challenges related to the intrinsic characteristics of these types of networks. One of the most important of these challenges is energy. Nodes operate autonomously (and very often battery-powered), so it is necessary that the processes and algorithms used are efficient and energy saving.

The most relevant platforms, including key features and manufacturers will be introduced in next sections.

2.1.2.1. IMote2

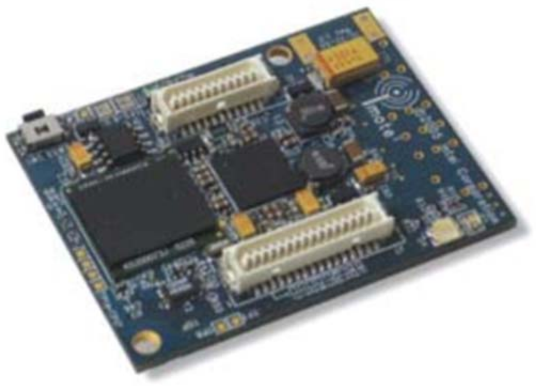


Figure 8 IMote2

Manufacturer	Crossbow
Model	IMote2 [17]
Processor	Intel PXA271 at 416 MHz
Memory	32 MB (RAM) and 32 MB (FLASH)
Radio	2.4 GHz IEEE 802.15.4 at 250kbps
Sensors	Sensorboard (temperature, light and humidity)
Power source	3xAAA supports USB charging
Typical consumption	31 mA
Max. consumption	66 mA
Sleep mode	390 uA
Size	36x48x9 mm

2.1.2.2. Tmote Sky

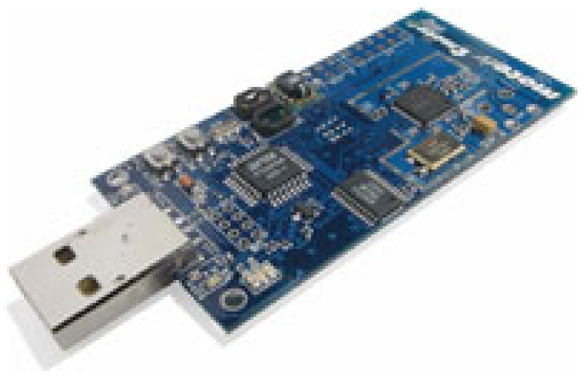


Figure 9 Tmote Sky

Manufacturer	moteIV
Model	Tmote Sky[18]
Processor	Texas Instruments MSP430 8MHz
Memory	10 Kb (RAM) and 48 Kb (FLASH)
Radio	250kbps 2.4GHz IEEE 802.15.4 (TinyOs network stack sup.)
Sensors	Humidity, temperature and light
Power source	2 x AA batteries (2000 mAh approx.)
Typical consumption	2.4 mA
Max. consumption	22 mA
Sleep mode	5.1 uA
Size	65x32x7 mm

2.1.2.3. MicaZ

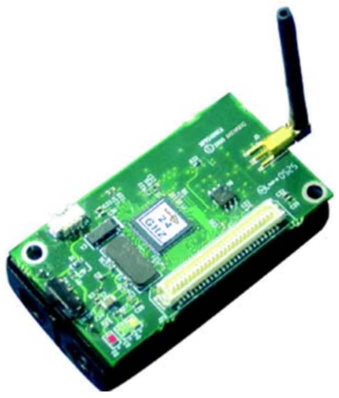


Figure 10 MicaZ

Manufacturer	Memsic
Model	MicaZ[19]
Processor	Atmel AVR at 7 MHz
Memory	4 Kb (EEPROM) and 512+128 Kb (FLASH)
Radio	IEEE 802.15.4 (Chipcon CC2420) 250 kbps
Sensors	Expansion connector for Memsic sensors (temp, light, GPS)
Power source	2 x AA batteries (2000 mAh approx.)
Typical consumption	20 uA
Max. consumption	19.7 mA
Sleep mode	1 uA
Size	58x32x7 mm

2.1.2.4. Mica2

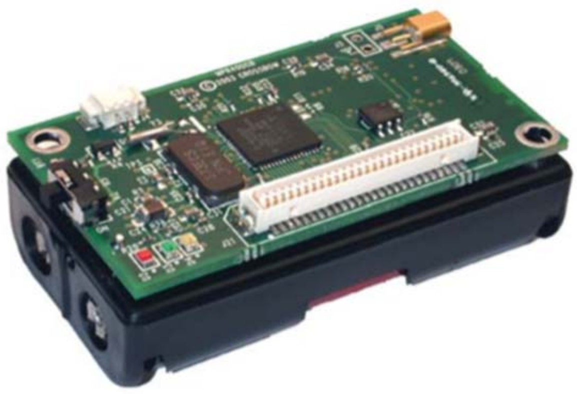


Figure 11 Mica2

Manufacturer	Crossbow
Model	Mica2[20]
Processor	Atmel ATmega 128L at 16 MHz
Memory	4 Kb (EEPROM) and 512+128 Kb (FLASH)
Radio	868/916 MHz (TinyOs network stack) at 38.4 Kbaud
Sensors	Crossbow expansion connector (temp, light, mic, accel,…)
Power source	2 x AA batteries (2000 mAh approx.)
Typical consumption	18 mA
Max. consumption	35 mA
Sleep mode	16 uA
Size	58x32x7 mm

2.1.2.5. Mica2DOT

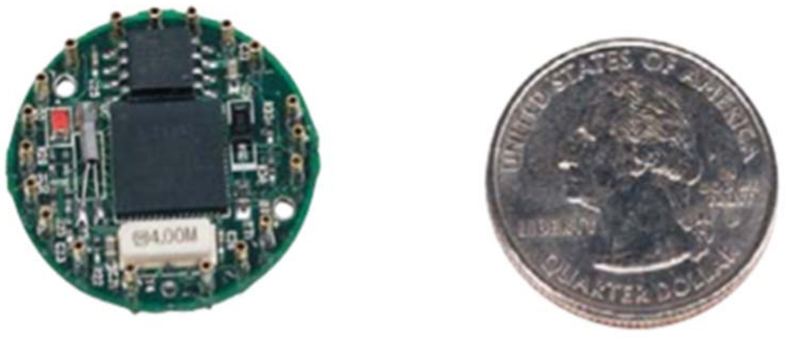


Figure 12 Mica2DOT

Manufacturer	Crossbow
Model	Mica2DOT[21]
Processor	Atmel ATmega 128L
Memory	4 Kb (EEPROM) and 512+128 Kb (FLASH)
Radio	868/916 MHz (TinyOs network stack)
Sensors	Crossbow expansion connector (temp, light, mic, accel,…)
Power source	2 x AA batteries (2000 mAh approx.)
Typical consumption	18 mA
Max. consumption	35 mA
Sleep mode	16 uA
Size	25x6 mm

2.1.2.6. TinyNode

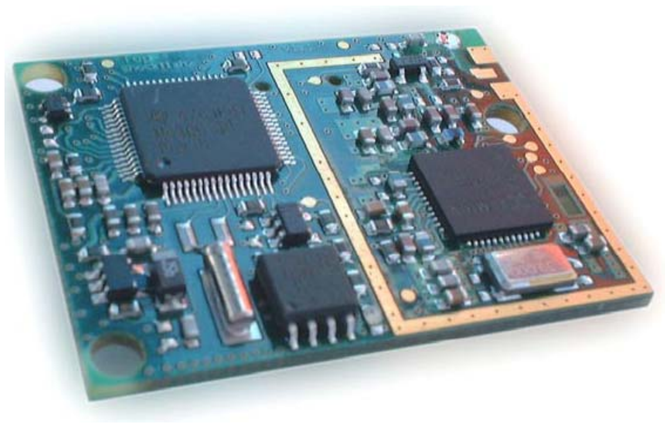


Figure 13 TinyNode

Manufacturer	ShockFish
Model	TinyNode 584 [22]
Processor	Texas Instruments MSP430 8MHz
Memory	10 Kb (RAM) and 48 Kb (FLASH)
Radio	868 MHz Xemics XE1205 (TinyOS network support)
Sensors	Humidity, temperature and light
Power source	2 x AA batteries (2000 mAh approx.)
Typical consumption	12 mA
Max. consumption	69 mA
Sleep mode	4.1 uA
Size	30x40x6 mm

2.1.2.7. Shimmer

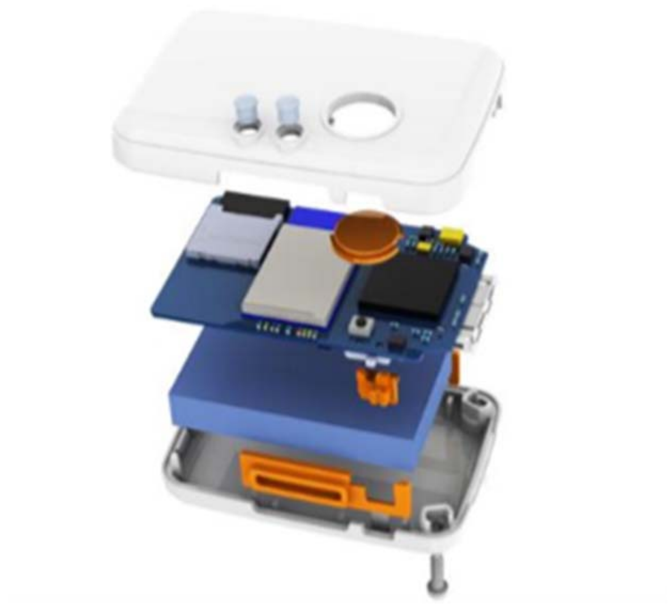


Figure 14 Shimmer

Manufacturer	Shimmer Sensing
Model	Shimmer3[23]
Processor	Texas Instruments MSP430 8MHz
Memory	16 KB (RAM) and 256 KB (FLASH)
Radio	Bluetooth and IEEE802.15.4 (TinyOS network support)
Sensors	Accel, gyro, magnetic, pressure
Power source	Li-ion battery
Typical consumption	12 mA
Max. consumption	69 mA
Sleep mode	4.1 uA
Size	51x34x14 mm

2.1.2.8. Zolertia Z1

Figure 15 Zolertia Z1

Manufacturer	Zolertia
Model	Z1[24]
Processor	Texas Instruments MSP430 16MHz
Memory	16 KB (RAM) and 256 KB (FLASH)
Radio	2.4GHz IEEE 802.15.4, 6LowPAN compliant
Sensors	Accelerometer, temperature
Power source	2xAA (or AAA) cells or 1xCR2032 cell
Typical consumption	22 mA
Max. consumption	52 mA
Sleep mode	2 uA
Size	57x34x14 mm

2.1.2.9. WaspMote

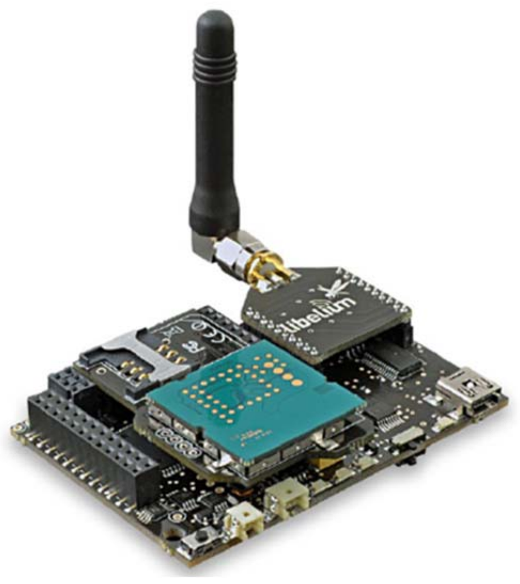


Figure 16 WaspMote

Manufacturer	Libelium
Model	WaspMote[25]
Processor	ATMega 1281 at 15 MHz
Memory	8 KB (RAM) and 128 KB (FLASH)
Radio	2.4GHz (IEEE 802.15.4 and Zigbee) and 868 MHz versions
Sensors	Accelerometer, temperature
Power source	3.3 V - 4.2V supports USB and solar panel charging
Typical consumption	55 uA (sleep)
Max. consumption	15 mA
Sleep mode	0.07 uA (hibernate)
Size	73.5x51x13 mm

2.1.2.10. SunSPOT

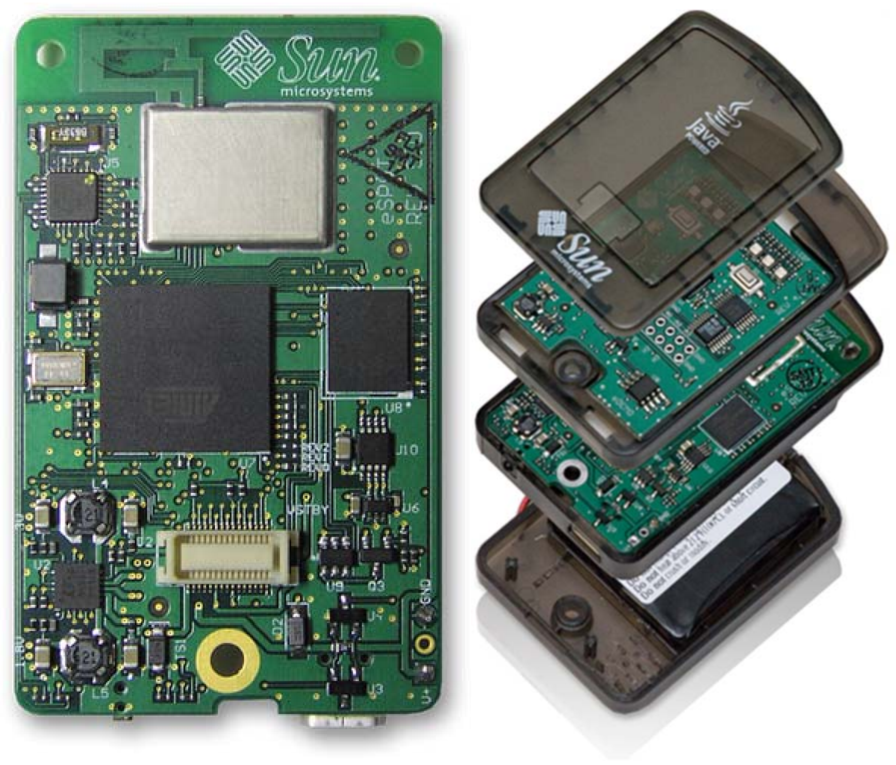


Figure 17 SunSPOT (main board—left—and stacks –right—)

Manufacturer	Oracle
Model	SunSPOT[26]
Processor	ARM 920T CPU (180MHz, 32-bits)
Memory	512Kb RAM, 4Mb FLASH
Radio	2.4 GHz IEEE 802.15.4 at 250kbps
Sensors	Sensorboard (IR, temperature, light, accel and speaker)
Power source	3.6V rechargeable 750 mAh Li-Ion battery
Typical consumption	40 mA
Max. consumption	100 mA
Sleep mode	36 uA
Size	41x70x23 mm

Table 5 includes the key features for the WSN platforms previously presented in the corresponding section.

Table 5 Key features for selected WSN commercial nodes

Platform	CPU speed	Data tx.	Comms.	Memory	Battery
IMote2	416 MHz	250 Kbps	802.15.4	32 MB (RAM) 32 MB (FLASH)	3xAAA
Tmote Sky	8 MHz	250 Kbps	802.15.4	10KB (RAM) + 48KB (FLASH)	2x AA
MicaZ	7 MHz	250 Kbps	802.15.4	4KB (RAM) + 512KB (FLASH)	2 x AA
Mica2/dot	16 MHz	38 Kbaud	TinyOs	4KB (RAM) + 512KB (FLASH)	2 x AA
TinyNode	8 MHz	152 Kbps	TinyOs	10 KB(RAM) + 48 KB(FLASH)	2 x AA
Shimmer	8 MHz	250 Kbps	BT + 802.15.4	16 KB(RAM)+256 KB(FLASH)	Li-ion
Zolertia Z1	16 MHz	250 Kbps	802.15.4	16KB(RAM)+256KB (FLASH)	2 x AA
WaspMote	15 MHz	250 Kbps	802.15.4	8 KB(RAM) + 128KB (FLASH)	
SunSPOT	180 MHz	250 Kbps	802.15.4	512KB (RAM)+4Mb (FLASH)	Li-ion

As it can be observed in Table 5, there is a wide variety of commercial platforms for WSN nodes. The speed of the processor varies from 7 MHz up to 416 MHz The most used communication protocol is the IEEE 802.15.4 but there is an increasing number of nodes implementing the Bluetooth Low Energy (BLE) standard. Depending on the application running in the nodes, the amount of memory could be a drawback (currently there are nodes with up to 32 MB RAM memory). Another important feature related with the application is the power source. There are platforms that can be powered with solar cells for unlimited battery life.

SunSPOT platform outstands as the most powerful platform, but with a large size and a low battery life as its main drawbacks. Applications for SunSPOT can be developed in Java language, re-using the libraries available for the Standard Edition, as exposed in further sections. This will reduce the effort when designing or adapting applications to this platform.

2.1.3. IoT specific platforms

IoT paradigm is currently a reality and there is a growing availability of commercial platforms intended to be used as the back-bone of IoT ecosystems. These devices are powerful enough to control a full sensor network and connect it to the Internet. In this way, applications are able to gain access to the data acquired by the sensors and command the actuators connected to the network. The four most important IoT platforms and their key features are presented in Table 6.

Table 6 IoT specific platform features

	Intel Galileo[27]	**Intel Edison**[28]	**Raspberry Pi (Model B)**[29]	**Arduino UNO**[30]
Size (cm)	10 x 7	3.5 x 2.5	9 x 6	7 x 5
Processor	Intel Quark X1000	SoC (Intel Atom+Quark)	Broadcom BCM2835	Atmel ATmega328
Architecture	Intel Pentium (32-bit)	Intel Pentium (32-bit)	ARM (32-bit)	AVR (8-bit)
Speed	400 MHz	500 MHz	700 MHz	16 MHz
Cache	16 KB	16 KB+16 KB	32KB+128KB	No
RAM	256 MB	1 GB	512 MB	2 KB
FLASH	8 MB	4 GB	No	32 KB
GPU	No	No	Broadcom	No
External Storage	Micro-SD Card (up to 32GB)	No	SD-card	SD-card (using shield)

2.1.4. WSN software development architectures

As stated in previous sections, the intrinsic requirements of the WSN nodes impose several requirements to the software developed. In brief, algorithms and protocols must provide:

- Power saving configuration
- Resilience and recovery facilities
- Self-configuration mechanisms

2.1.4.1. Software architectures for WSNs

Common WSN deployments include different kinds of applications and software modules collaborating among them. A typical example is depicted in Figure 18.

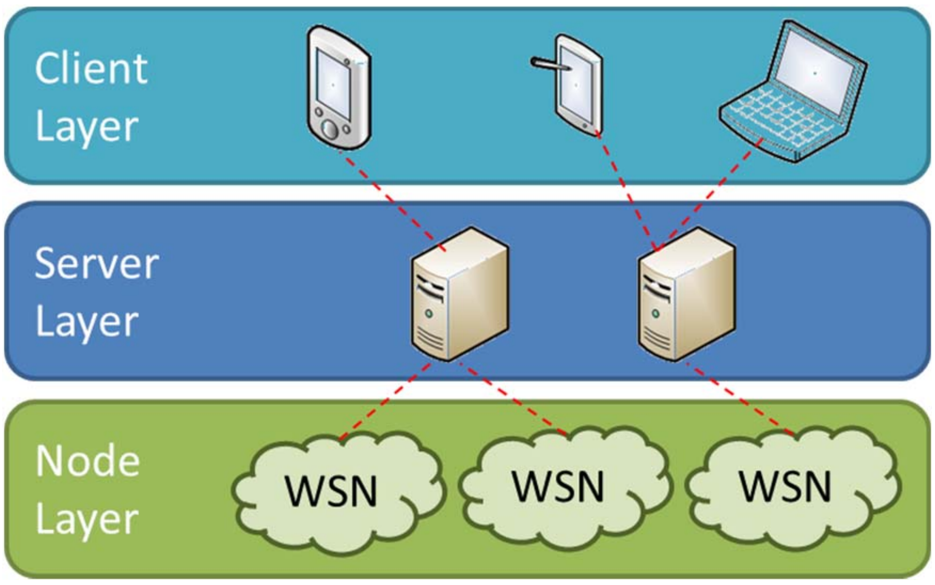


Figure 18 Software architecture for WSNs

As shown in Figure 18, a typical WSN architecture is composed by three layers:

- **Node layer** is composed by a set of nodes with their installed sensors. In this layer is included the Operating System (OS) and the service-required applications. These applications are often developed ad-hoc for a specific hardware platform.

- **Server layer** receives the information from the WSN using a set of protocols and stores it in a data base system. Furthermore, it can provide access to data through common interfaces (based on IP protocols).

- **Client layer** includes the mechanisms needed for the user to configure, manage and access the data provided by the WSN (using, for example, a Graphical User Interface).

2.1.4.2. Operating Systems

Operating systems designed for WSNs should be lighter and simpler than the general purpose ones. TinyOS is one of the first WSN specific designed OS. Other OS for WSN includes Contiki, MANTIS, NUT/OS, SOS and Nano-RK. A brief description for each system is presented below:

- **TinyOS**[31] is a light operating system intended to be used in low capability WSN nodes. TinyOS is an event-driven system where an event manager is the main actor. Both the operating system and external code are written in a specific language called nesC (which is based in ANSI C). The memory footprint of the core elements is only 400 bytes. This operating system is used in a wide range of applications and platforms, mainly because it was one of the first WSN specific operating systems. This operating system utilizes the component model, where each component is an independent entity exposing an interface. Components can be re-used in different applications, leading to a fast and easy application development.

- **Contiki**[32] is an event-driven operating system written in C and provides features like threading, scheduling, event management and IPv6 support. It has been ported to a number of architectures and the footprint is very low.

- **LiteOS**[33] provides multi-threading functionalities for low capability WSN nodes with a low memory footprint as well as Over-The-Air user application updates. It also provides dynamic memory allocation and event logging capabilities.

- **SOS**[34] is a low-capability operating system providing messaging, dynamic memory allocation, scheduling and garbage collector for memory usage optimization.

- **MANTIS**[35]is an embedded operating system designed for low-capabilities wireless sensor nodes. It provides multi-threading, real-time and low memory footprint (500 bytes) as well as on-the-fly reprogramming and remote login.

- **Nut/OS**[36]is a simple real-time operating system designed for the BTNode platform, but it has also been ported to AVR and ARM platforms implementing the Nut/Net TCP/IP stack. This operating system provides mechanisms for multithreading, event management, dynamic memory allocation and interruptions. Applications for this operating system are written in C and C++.

- **Nano-RK**[37] is a small-footprint real-time operating system with multi-hop networking support. It provides multi-tasking capabilities by means of priority and resource management. Native applications are written in C language.

- **RIOT**[38] is an open-source real-time operating system designed for sensor networks. It has been ported to 8, 16 and 32 bits set of platforms and implements IPv4, IPv6 and 6LoWPAN stacks.

- **SunSPOT**[39] platform is a Java-based and Java-enabled platform, based in the Mobile Information Device profile (MIDP) and Squawk Virtual Machine. This enables the developer to start programming WSN applications in Java language using the provided libraries for networking, sensors and memory management. The platform also provides a full-featured emulator environment (Solarium) as well as NetBeans-based IDE.

Table 7 WSN operating systems features

	Simulator Available	Multitask	Secured Implem.	Apps. Language	Real Time support
TinyOS	YES	Events and threads	TinySec	NesC	NO
Contiki	YES	Events and threads	ContikiSec	C	NO
LiteOS	YES	Events and threads	N/A	C++	NO
SOS	NO	Threads	N/A	C	NO
MANTIS	YES	Multi-threading	N/A	C	YES
Nut/OS	NO	Events and threads	N/A	C and C++	YES
Nano-RK	NO	Threads	N/A	C	YES
RIOT	NO	Multi-threading and events	N/A	C and C++	YES
SunSPOT	YES	Midlets and isolates	JCE	Java	YES

As it can be observed in Table 7 there are currently real-time and non-real-time operating systems available to be embedded in WSN nodes. The selection of one of these groups will depend on the application requirements. Most of the operating system provide multi-tasking capabilities, and C language is the most used for application development. SunSPOT platform makes possible the development of applications in Java programming language (using the development tools usually needed in Java application developments) as well as a complete simulation platform for testing the applications.

Despite the large number of platforms using C language, the resource limitations have lead into new lighter programming languages, as already mentioned nesC. New commercial platforms offer the possibility of using several languages as C++, Java, .NET, etc.).

Beyond operating systems or programming languages for WSNs there have been also proposed in academic area a group of middleware solutions specifically designed to be implemented in nodes participating in a WSN. These proposal can be generic middleware solutions [40-42], application specific (e.g. smart homes[43] or e-Health[44]) or security focused [45-47].

2.2. WSN security solutions

Since WSNs are being used in a wide variety of applications, security challenges must be addressed. Some of these emerging applications include scenarios where sensitive and personal data (medical, health-care or sports applications) are involved, and it has to be noted that these data are protected by national and international legislation.

Focusing on WSN security specific problems, it can be stated that security in WSNs must support generic applications and the basic operation of the network, taking into account the following facts (some of them have been previously mentioned):

- these nodes are tiny devices, with limited resources
- they are limited in their energy supply, although they can be powered by solar cells
- they are limited in resources regarding the computational power, the amount of memory, radio bandwidth and coverage
- there are no tampered-proof zones in the commercial hardware platforms to store sensible information (keys, identifications, etc.)
- they tend to fail due to exposure to adverse environmental conditions or lose their power, and could be removed by unauthorized personnel
- radio communications can be intercepted and crypto-analysed to extract data, keys, etc.

As a wireless system, security challenges are greater than in wired networks because of the open access to the data flow. Using broadcast transmissions, other equipment listening in the same frequency can intercept every communication between two nodes. Other WSN characteristics-related challenge is the possibility that a node is been captured and crypto-analysed by a third party. If security keys, policies, and other cryptographic elements are accessed, a new spurious node could be introduced in the network, and several attacks can be deployed: DoS, Man-In-the-Middle, data sniffing and/or modification, routes spoofing, etc.

Academic research publications regarding WSN security have increased their number substantially. The most relevant papers have been studied and the results are presented below.

The very first proposal for securing emerging WSNs was a set of security protocols based on TinyOs, called SPINS[48]. SPINS provides data confidentiality, integrity and authentication. Also, TinySec[49] provides the same security mechanisms, and both are based on symmetric cryptography without any key management system. Following with specific TinyOs proposals, TinyKey[50] addressed with key management and included mechanisms for key generation, distribution and re-keying.

A novel key pre-distribution scheme in WSNs by Subash and Divya[51] appeared with a profound work in key distribution. Their proposal is based on the establishment of two links and two keys for each communication between two nodes. If one of the communication links is compromised, there is another link available. The drawbacks are the great number of keys involved, and the problem issued when a node is captured (thus revealing all the keys contained).

SecFleck [52] proposes a platform-specific security add-on, based on an Atmel TPM (Trusted Platform Module) chip. Although this solution provides a fast and energy efficient way for the support of both symmetric and Public Key Cryptography (including a tamper-proof chip for storing keys), it is a platform-specific solution, since it has been designed and validated only in Fleck nodes.

A general overview of trust systems and a simulation-based study is presented in [53]. Karthik *et al*. selected five trust models and evaluated them using TRMSim. Results present BTRM_WSN as the best-case trust model.

TRMSim-WSN[54] is a Java-based trust and reputation models simulator aimed to provide an easy way to test a trust and/or reputation model over WSNs and to compare it against other models. Another trust calculation schema is presented by Karthik *et al*. in [55].

The first idea of a decentralized trust system was presented by Blaze, Feigenbaum, and Lacy [56]. They proposed Policymaker, a unified decentralized trust management system, based on a simple language for describing security policies, credentials and relationships.

A trust system for WSNs is presented by A. Boukerch *et al*. in [57]. They put forward an agent-based trust and reputation management scheme (ATRM) for Wireless Sensor Networks, assuming that mobile agents are resilient against the unauthorized analysis and modification of their computation logic.

In [58], Oleshchuk demonstrates how the concept of trust can be used to increase security in Wireless Sensor Networks, without using cryptography by taking into consideration trustworthiness of individual sensors and monitoring each sensor activities. Also Sarma *et al*. present a specific WSN trust system in [59].

A particular trusted routing protocol for MANETS is presented in [60]. L. Abusalah *et al*. propose a Trust-Aware Routing Protocol (TARP) for secure-trusted routing in mobile ad hoc networks. In TARP, security is inherently built into the routing protocol where each node evaluates the trust level of its neighbours based on a set of attributes and determines the route due to these attributes. Recently, there have been several contributions to trust routing in WSNs [61-65].

2.3. ESB implementations

The increasing number of web services available is easing the creation of new mash-up applications. These applications retrieve information from a collection of web services, which are developed in a wide range of programming languages, use different data formats for the interchange of information and work with different communication protocols.

A complete solution to integrate all of these new services is an Enterprise Service Bus (ESB). An ESB provides a distributed solution for integration, providing mechanisms to deploy, control and upgrade applications.

The first definition for an ESB was provided by Schulte (as presented in [66]), where service is an autonomous program able to communicate with other similar entities; bus is used as an analogy to a computer bus; and enterprise was included because at first the solution was intended to be used in large companies to reduce complexity of integration.

The ESB solution can be defined as a complete architecture to integrate different services and to facilitate the development of applications. The ESB not only provides communication for the services, but also for the high level applications. This allows applications to decouple from each other, enhancing the resilience of the full system. But there are more benefits when using an ESB:

- Scalability
- Integration
- Flexibility
- Decentralization
- Re-factoring
- Open to integrate new languages and protocols
- Re-usage of components already developed for other platforms
- Distributable architecture
- Heterogeneity solution
- Security enhancement

However, there are also some drawbacks inherent to this solution:

- Reduction of communication speed, since information needs to reach the bus
- Addition of overhead to communications
- ESB management requires time and effort
- The bus becomes a critical point to be protected in the infrastructure

There are several ESB implementations, both commercial (IBM WebSphere[67], Microsoft BizTalk Server[68], Oracle Enterprise Service Bus[69], etc.) and open-source distributions. The aim of this thesis is to use as much open-source code as possible, so the study has focused in open-source solutions. A brief summary of the most relevant implementations has been included in Table 8.

Table 8 ESB platform features

	JDK minimum version	IDE languages	Distributable	License
Apache ServiceMix[70]	JDK 1.6	X	NO	Apache License 2.0
Fuse/Jboss ESB[71]	JDK 1.6	Perl, PHP, Python, Ruby, Java, .NET, C, and C++	YES	GNU-GPL
PetalsESB[72]	JDK 1.6	X	No	GNU-GPL
WSO2[73]	JDK 1.5	Java, Ruby, Groovy	NO	Apache License 2.0
OpenESB[74]	JDK 1.6	X	YES	CDDL
MuleESB[75]	JDK 1.6	Java	YES	CPAL
UltraESB[76]	JDK 1.6	Java, scripting	YES	AGPL

2.4. Wearable devices

Smart devices (phones, watches, etc.) have become common life elements. Their capability to acquire and to process complex data expands with the inclusion of new devices and new operating systems (iOS, Android, etc.). Wireless devices can monitor not only environmental parameters (temperature, humidity, etc.) but also several human body parameters, like blood pressure, heart rate, breathing rate, body temperature, etc. The communication between these devices, conforming a Body Area Network (BAN), and Internet-capable devices (as the already mentioned smart ones) produce a wide new range of protocols and applications, as presented in several proposals [77-83].

2.4.1. Physiological monitoring

Due to the already presented characteristics of Wireless Sensor Networks, they are getting an important place in e-Health applications. WSNs are flexible to integrate into health environments, as they are not intrusive, inexpensive, small-sized and easily portable. A significant number of research works have already been presented on this topic, as presented by Yang Xiao *et al*. in [84], where they provide an extended survey on wireless telemedicine including relevant wireless technologies, applications and research issues.

In [85], the authors discuss and map the main findings resulting from the development of a series of four WSN-based health monitoring systems, under the generic name of MoteCare. The paper also presents a generic framework that can be adapted for healthcare monitoring, either at a patient’s home or in a care facility.

In [86], Cheng and Zhuang propose a Bluetooth-enabled in-home patient monitoring system, making early detection of Alzheimer's disease easier. Based on the movement pattern of a patient, a medical practitioner is able to determine whether a target patient is developing Alzheimer's disease. They have developed a study showing that the proposed in-home patient monitoring system is feasible and can be put into practice.

In [87], Lawrence *et al*. research the feasibility of using modern interactive games to help improving the quality of life of the elderly people (living in their own homes or in elderly care facilities). They intend to integrate such technologies into their prototype health monitoring system called ReMoteCare, a WSN-based system.

In [88], Chen *et al*. put forward a novel e-healthcare management system based on the introduction of encoded rules that are dynamically stored in RFID tags, and explain how it can be employed to leverage the effectiveness of existing ones.

In [89], the authors research the application of integrated IEEE 802.16/WiMAX and IEEE 802.11/WLAN broadband wireless access technologies along with the related protocol issues for telemedicine services. After reviewing IEEE 802.11/WLAN and IEEE 802.16/WiMAX technologies, applications and deployment scenarios of integrated IEEE 802.16/WiMAX and IEEE 802.11/WLAN for telemedicine services are proposed.

In [90], a novel cognitive, radio-based system for e-Health applications in a hospital is introduced, which protects medical devices from harmful interference by adapting the transmission power of wireless devices affected by Electro-Magnetic Interference (EMI) constraints. An EMI-aware handshaking protocol is proposed for channel access by two different types of applications with different priorities. They also have evaluated the performance of this cognitive radio system for e-Health applications through some simulations.

A system providing patient location, tracking and monitoring services in nursing institutes through a WSN is presented in LAURA [91]. The system is composed of three functional blocks: a location and tracking engine that performs location out of samples of the received signal strength and tracking through a particle filter; a personal monitoring module based on bi-axial accelerometers -which classifies the movements of the patients to eventually detect hazardous situations-, and a wireless communication infrastructure to deliver the information remotely. Both centralized and distributed solutions proposed for the implementation and testing of the strengths and weaknesses of the two solutions are highlighted from a system-based perspective in terms of location accuracy, energy efficiency and traffic loads. LAURA modules have been tested in a real environment using commercial hardware.

Wan-Young *et al*. presented in [92] the design and development of a wearable and ubiquitous healthcare monitoring system using non-intrusive sensors for measuring acceleration, oxygen saturation (SpO2) and electrocardiogram (ECG). Low power ECG, accelerometer and a SpO2 sensors board were integrated in a wearable device for user's health monitoring. The system transmits physiological data to a base-station connected to a computer, allowing the access to the data across external applications.

Recent applications are focused not only on specific e-Health developments, but also on improving sports routines for professional or occasional sportsmen/women. A sport-related WSN based application is REMOTE [93], providing a detailed picture of boat movement and rower individual performance. The application analyses data gathered within the network to obtain useful data about rower´s performance.

López-Matencio *et al*. presented in [94] the system architecture and implementation of an ambient intelligence assistant for runners relying on a WSN deployed over a cross-country running circuit. The heart rate of the users is monitored, and the system can select, for each user, suitable tracks where the heart rate will be in the selected range.

Another sport-related scenario for WSN is presented in [95], where the authors take a first step towards characterising wireless connectivity in the soccer field by undertaking experimental work with local soccer clubs, and assess the feasibility of real-time athlete monitoring. They have developed an empirical profile of radio signal strength in an open soccer field taking into account distance and body orientation of the players, and they have also developed practical multi-hop routing algorithms that can be tuned to achieve the right balance between the competing objectives of resource consumption and data extraction delay.

A hybrid health care and sport application is presented in [96]. Mariotti, C.*et al*. describe a health monitoring and indoor localization system based in a shoe-mounted sensor module. The shoe sole (which includes an NFC technology) measures the body temperature and also is a renewable energy scavenger, which transforms the human motion to electrical energy. The proposed platform can be extended to other sensors applications in order to monitor the sport performances of the athletes as well as to improve the rehabilitation techniques if required.

Working with all the data gathered in sports applications is also an open issue. A good approach focused on how to manage the data in e-Health is presented in SUSHI [97] (Supporting Unified access for Streaming and Historical data). SUSHI provides explicit representation of domain objects, queries over heterogeneous data sources, online access to different sources, and the inclusion of processing components such as forecasting and simulation modules.

2.4.2. Smart wearable devices

Wearable devices are called to be the new trend in smart devices, after the smartphone has become a daily use device. These wearable devices are either stand-alone or smartphone companions, and the manufactures are focusing their effort on designing new applications using them (as is can be checked in both Android and IOS app-stores, with a huge number of health and sport applications designed to be used with wearable devices). Indeed, Google has launched a specific version of Android OS (called Android Wear) to be used in wearable devices like smartwatches.

In fact, smartwatches are capturing the attention as the new trend in smart devices. These smartwatches are connected using a wireless technology to the smartphone and also incorporate several sensors (accelerometer, heart-rate sensor, GPS, compass, etc.). These features are speeding up the development of new e-Health and sports applications. In addition, less featured devices like the smartbands are appearing in the market. Compared with smartwatches, the smartbands lack of a high resolution screen or display (some devices do not even have a display, only some LEDs), their battery last longer and are lighter and easier to carry.

Finally, there is a group of devices highly related with sports and health that are also reaching the general market: the heart-rate monitoring devices. These devices are intended to be worn on the chest using the included strap. They are not all-day-long wearing devices like the smartwatches or the smartbands, since they are intended to be used only in sport practices or e-Health applications (e.g. patient monitoring or elderly care).

The most relevant features for some of these new smart devices are included in the following sections, as a reference.

In a near future, a plethora of new wearable devices will enter the market. Intel, the big company of processors, has released a platform designed to be embedded in wearable devices: Intel Edison (its key features were presented in Table 6), foreseeing the demand of such a processor for the new emerging wearable devices.

2.4.2.1. Smartwatches

Table 9 Smartwatches comparison chart

Name	**G watch [98]**	**Gear2 neo[99]**	**Moto 360 [100]**	**Watch [101]**	**Zen-Watch [102]**	**One [103]**
Brand	**LG**	**Samsung**	**Motorola**	**Apple**	**Asus**	**WIMM**
Screen Size	1.65”	1.63”	1.56”	1.5”	1.63”	1”
Screen resol.	280x280	320x320	320x290	312x390	320x320	160x160
Processor	1.2 GHz Snap-dragon 400	Exynos 3250 1GHz	TI OMAP 3	Apple S1	1.2 GHz Snap-dragon 400	667MHz ARM-11
Memory	512 MB.	512 MB.	512 MB.	512 MB.	512 MB.	256 MB.
Sensors	compass	Heart rate, compass	Heart rate, compass	Heart rate, compass	Accel, heart rate	Compass accel.
Battery	400 mAh.	300 mAh.	320 mAh.	400 mAh.	369 mAh.	230 mAh
OS	Android (Wear)	Tizen	Android (Wear)	iOS	Android (Wear)	Android (2.1)

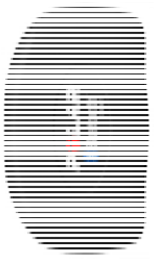
2.4.2.2. Smartbands

Table 10 Smartbands comparison chart

Name	**LifeBand [104]**	**Flex [105]**	**Fuelband SE [106]**	**MSBand [107]**	**SmartBand [108]**
Brand	**LG**	**Fitbit**	**Nike**	**Microsoft**	**Sony**
Screen	YES	NO	NO	YES	NO
Comms.	BLE	BLE	BLE	BLE, WiFi	BLE
Sensors	HR, ODO	ODO	ODO	GPS, HR, ODO	GPS, ODO
Battery	5 days	5 days	4 days	2 Days	369 mAh.
OS	iOS, Android	iOS, Android and Windows	iOS, Android, Windows	iOS, Android and Windows	Android

2.4.2.3. Heart rate monitoring devices

Table 11 Heart rate monitoring devices comparison chart

					
Name	**HxM [109]**	**Bioharness3 [110]**	**Forerunner 920XT[111]**	**H7 [112]**	**HRM-H8 [113]**
Brand	**Zephyr**	**Zephyr**	**Garmin**	**Polar**	**CooSpo**
Form	Chest strap	Chest strap	Chest strap and watch	Chest strap	Chest Strap
Comms	BLE	BLE	Bluetooth, WiFi	BLE and Polar-5KHz	BLE
Sensors	Heart Rate, speed, distance	Heart Rate, position, body temp., breathing, ECG, speed, distance	Heart rate, speed, distance, ODO, altimeter,	Heart Rate, ECG	Heart Rate
Battery	30 hours	35 hours	40 hours	200 hours (cell battery)	12 months (cell battery)
Data Storage	No	Internal (500 days)	Internal (100 hours)	No	No

2.5. Section summary

In this section, a comprehensive SotA analysis has been presented. This study has been split in three sub-sections.

The first section presents the SotA in Wireless Sensor Networks, including: the wireless communication technologies available to be used in this type of networks, the most relevant commercial platforms, the architecture schemas and the security solutions specifically designed for WSN. As a conclusion, the SunSPOT platform stands out as the most powerful node as well as a good IDE based in Java and includes an emulator solution.

The next section presents the ESB implementations (commercial and open-source) currently available. In the group of open-source solutions, the FuseESB implementation is one of the best distributable options because of the IDE languages available and the community support (including fast and often updates)

Finally, the features of selected wearable devices (smartwatches, smartbands and heart rate monitoring devices) have been presented in the last section. In the validation stage, the WIMM One smartwatch was selected as well as the Zephyr Bioharness3 monitoring belt.

3 Dissertation

3.1. Introduction

There is an increasing number of wearable devices (smartphones, tablets, watches, etc.) that should be taken into account when developing a human monitoring application based on an IoT scenario. Wearable devices may form a wireless network called Wireless Body Area Network (WBAN), which main purpose is collecting physiological data from the human body. At this point, an important issue appears when establishing communications between a Wireless Sensor Network and wearable devices: varying standards for different parts of the system.

On one hand, sensor nodes communicate employing IEEE 802.15.4 or Zigbee technologies (as presented in the Section 2). On the other, wearable devices usually work through the Bluetooth interface. This point is of critical importance if the goal is to achieve a real seamless integration of different particularities in the world of the IoT.

In addition, the network must provide standardized methods in order to allow applications to access the services provided by it. The best solution to achieve this goal is providing external interfaces, published as APIs (Application Programming Interfaces). There is a large number of proposals, based in web standards: SOAP, CoAP, REST, etc.

The importance of providing an API is not only to act as the communication end for two different systems or applications. Nowadays, APIs have become into a generic solution to provide access both to software elements (Software as a Service, SaaS), and to complete systems or hardware solutions. Currently it is possible to find a large number of hardware abstraction solutions, offering their capabilities “as a service”. The following definitions are nowadays widely used:

- **IaaS**. Infrastructure as a Service (access to storage, CPUs or networks)
- **PaaS**. Platform as a Service
- **XaaS**. Everything as a Service: security, hardware, big data, sensor, desktop, etc.

A well designed API is a good element to gain interest from third party companies in a company developed services. Developers will be interested in using the provided API and if the results are satisfactory, the will spread the benefits of that API to other developers. In fact, the idea of providing the access to the services through an API is becoming a generic and accepted solution in the market.

Another challenge to be faced is the heterogeneity. In order to solve the heterogeneity issues, an integration architecture with selected protocols (along a middleware layer and an ontology) appear as a complete solution. In the following sections the integration mechanisms will be depicted thoroughly.

A holistic view of the proposal is presented in Figure 19. This figure includes the most relevant contributions from this thesis: the Wearable Device Service Bus, the Wearable InterDomain Protocols, the distributed integration platform and the security proposal. The components will be thoroughly described in following sections.

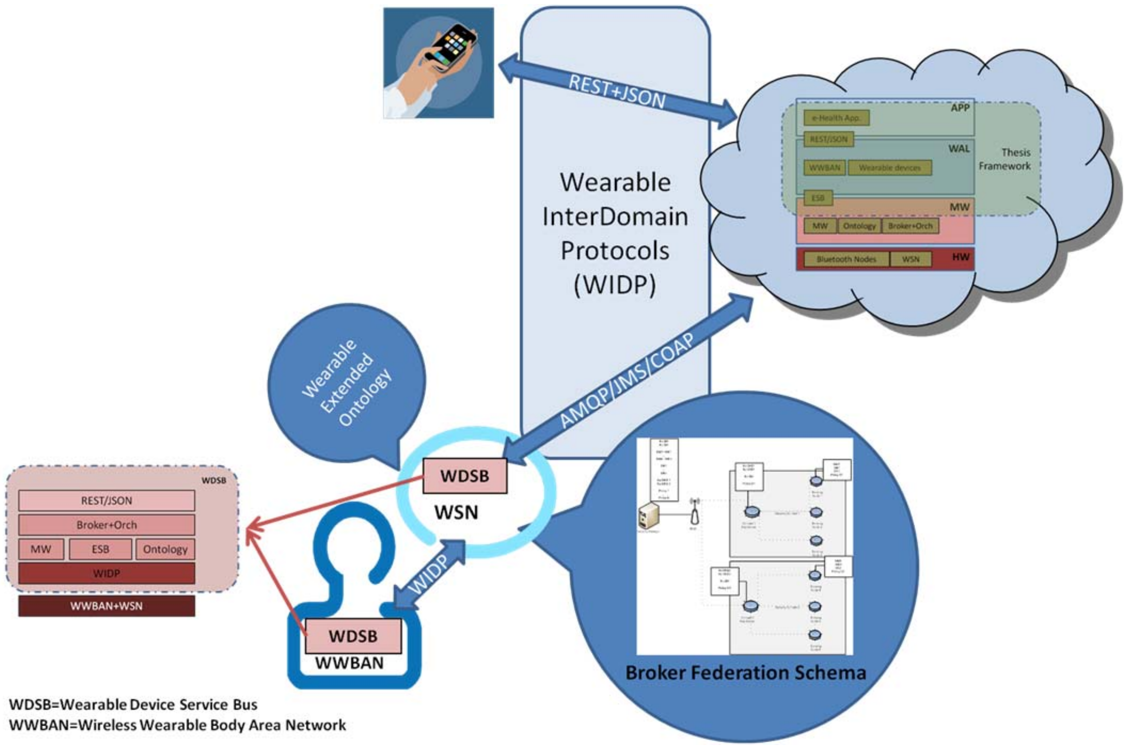


Figure 19 Holistic view of the proposal including the most relevant contributions

3.2. Wearable Device Service Bus

In order to provide an open framework for application development, it is necessary to provide application designers and programmers a suitable infrastructure with proper abstraction layers. This infrastructure should allow resolution, look-up and discovery of WSN services.

Applications should be able to access the WSN services regardless of the platform and Operating System where they are running. This requirement can be achieved using an integration layer including elements such as an Enterprise Service Bus and some web-based protocols like HTTP, REST, CoAP, etc. In the proposal, this layer is called the Wearable Device Service Bus (WDSB) and its components are represented in Figure 20.

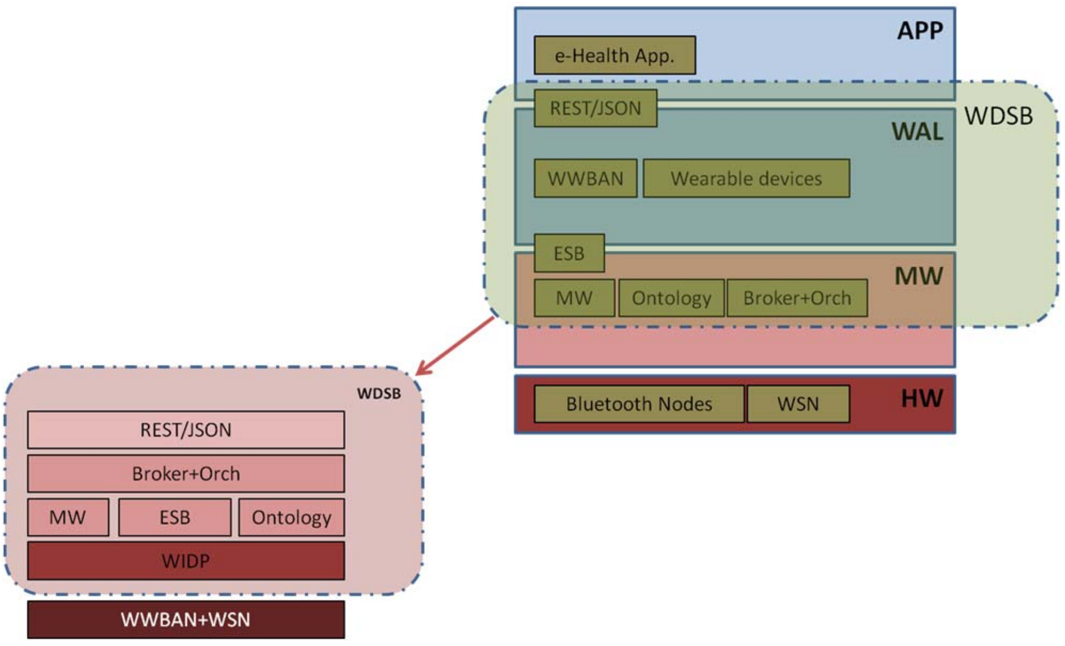


Figure 20 Wearable Device Service Bus (WDSB) components and boundaries

This WDSB should implement interoperability mechanisms in order to provide:

- Data interoperability
- Framework for application development
- Support for web-based standards
- Scalability and resilience

Related with data interoperability, the integration layer should provide mechanisms to access, translate (if necessary) and send data to applications. Data is obtained at the hardware layer and, as it will be presented in following sections, is semantically annotated. Thus, in some cases it will be necessary to translate the gathered data to the ontology used in the network (using a semantic middleware).

Supporting web-based standards will guarantee that applications have a light and standardized method to access to services provided by the middleware. Following an HTTP client/server model, WSN services would be available for applications running in low capability devices. Requests and responses are built around the transfer of resource representations (e.g., a resource could be a heart rate measuring data). HTTP methods such as GET, PUT, POST, and DELETE could be used to perform different operations. Each resource response will be coded using the same kind of widely used WEB data format like XML documents or JavaScript Object Notation (JSON).

The proposed architecture must provide updating and upgrading mechanisms in order to support future services and platforms as well as open mechanisms to update the solution with new implementations or enhancements.

The best solution to meet with the presented requirements is to include an Enterprise Service Bus (ESB). An ESB is a software architecture solution based on Service Oriented Architecture (SOA) model, used for designing and implementing the interaction and communication between different software applications. It provides an asynchronous message oriented communication between the applications inside the bus. The strength of the ESB solutions is the integration of heterogeneous solutions inside a unique communication bus.

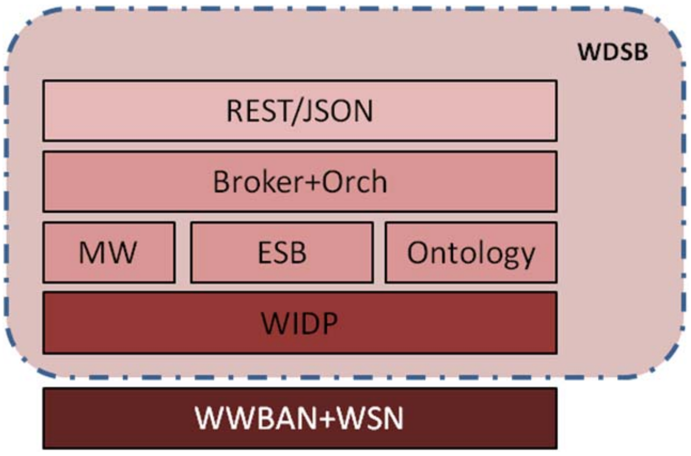


Figure 21 WDSB architecture

In a generic architecture using an ESB, an application will communicate via the bus, which acts as the single message router between applications, reducing the number of connection among the applications. Furthermore, this solution eases the upgrading and updating process in large software developments. By reducing the number of communication ends from and to a particular application, it is easier to monitor for failure and misbehaviour in highly complex systems and to allow easier changing of components. The bus can also be distributed, shifting the advantages of scalability, resilience, etc.

As depicted in Figure 21 there are several layers composing the proposed WDSB architecture:

- **REST/JSON** is the high level layer provided to the applications in order to connect to the services published by the network. REST provides a light and simple way for applications to connect using HTTP commands, in the same way as retrieving web pages. Thus, every device able to run a browser or to manage the HTTP stack will be able to connect to the WDSB (including low-capabilities devices). The possibility of using JSON messages as the language for the response to the REST queries is useful because of its language-independent data format, being lighter than XML responses.

- **Middleware layer** includes the middleware (or middleware deployments), the integration platform (ESB) and the services ontology.

- **WIDP** is the wearable inter-domain protocols with its corresponding messages to be interchanged between the different intermediation layers. The selection of the protocols has been conducted taken into account the requirements of the wearable devices and sensor nodes (low computation power) and energy efficiency, as explained in the corresponding section.

3.2.1. Formalization of an ESB-based platform for wearable services and middleware interoperability

In order to obtain a generic integrative and scalable solution, the use of an Enterprise Service Bus (ESB) was decided.

As introduced in the previous section, an ESB is a software architecture model based on the Service Oriented Architecture (SOA) paradigm. It was designed to provide asynchronous communications between applications and its prominent feature is the integration of heterogeneous solutions inside a unique communication bus, providing a single end-point for applications.

Following the specification of the generic client-server architecture, the client requests are routed (and adapted if necessary) to the corresponding service, which will answer the petition. In fact, the client sends the request to the ESB (instead of sending it to the application server or the middleware integrating the network services), and the ESB is responsible for routing the request to the provider, monitoring and logging the messages. This approach exposes a unique external interface to access different applications lying behind the bus (or, in this approach, different middleware architectures). Thus, the updating and migration issues are easier to manage.

A general overview of the integration provided by the ESB to external applications is shown in Figure 22.

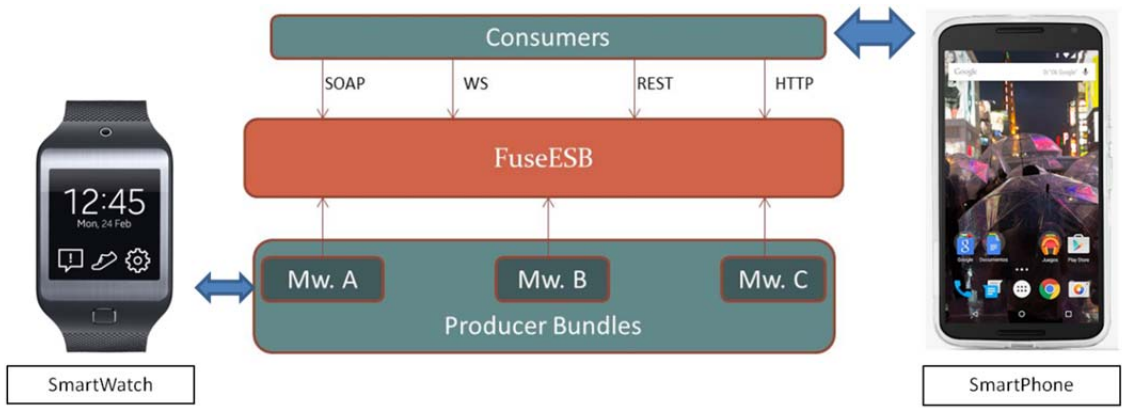


Figure 22 An ESB for different consumers and middleware implementations (centre), with a smartwatch attached (left). The services can be consumed by a wide range of applications, e.g.: a mobile application (right)

Components use the bus to communicate between them, reducing the underlying number of connections. By doing this, debugging errors or changing components is easier. The ESB can be distributed, improving the scalability and resilience features of the system. The user application would be able to communicate with the ESB by means of the external interfaces.

The ESB solution contributes to integrate all the system components, but it does not solve by itself the semantic issues. Thus, it is proposed to integrate an existing semantic middleware (or more) integrated inside the ESB as a producer bundle.

As it can observed in Figure 23, the integration proposal is divided in 4 sub-systems: the user application acting as the client connected to the ESB and accessing to the services provided by the middleware, which is connected to one or more WSNs. Optionally, the WSN can be connected to a Wireless Wearable Body Area network, measuring some physiological parameters of the user.

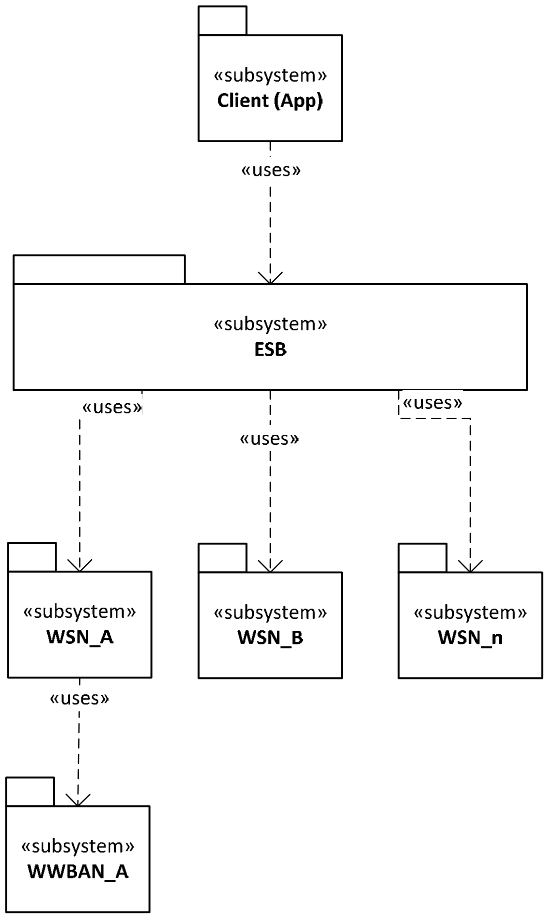


Figure 23 UML subsystem diagram

The UML diagram depicting the packages composing the proposal is presented in Figure 24.The ESB is able to manage one or more middleware implementations running concurrently. The ESB will route the application service queries to the middleware providing the required service. Then, the middleware will communicate with the WSN and will retrieve the data with regards to the selected service.

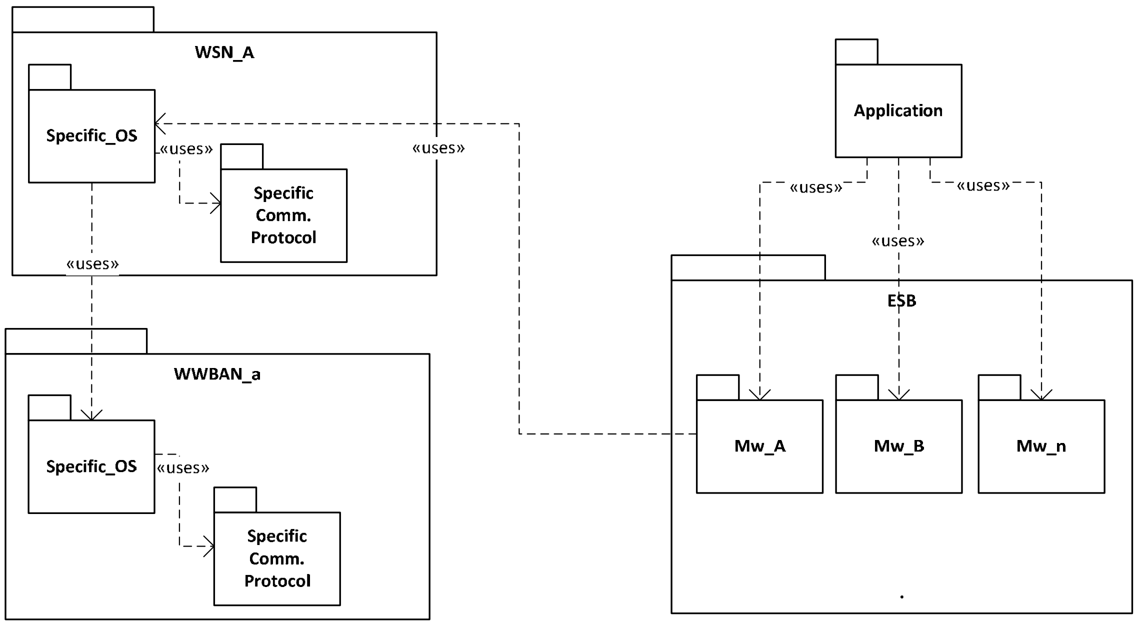


Figure 24 UML package diagram

The complete specification of the proposal (using a UML component diagram) is presented in Figure 25. The application (through an HTTPS client) will perform a service request using the REST interfaces provided by the ESB. The ESB will receive the request and will route to the corresponding middleware publishing the service. Then, the middleware will query to the WSN connected in order to obtain the data for the requested service. Finally, the response will be served to the application using JSON or XML messages. For intra-ESB middleware components communication the OSGI interfaces are used.

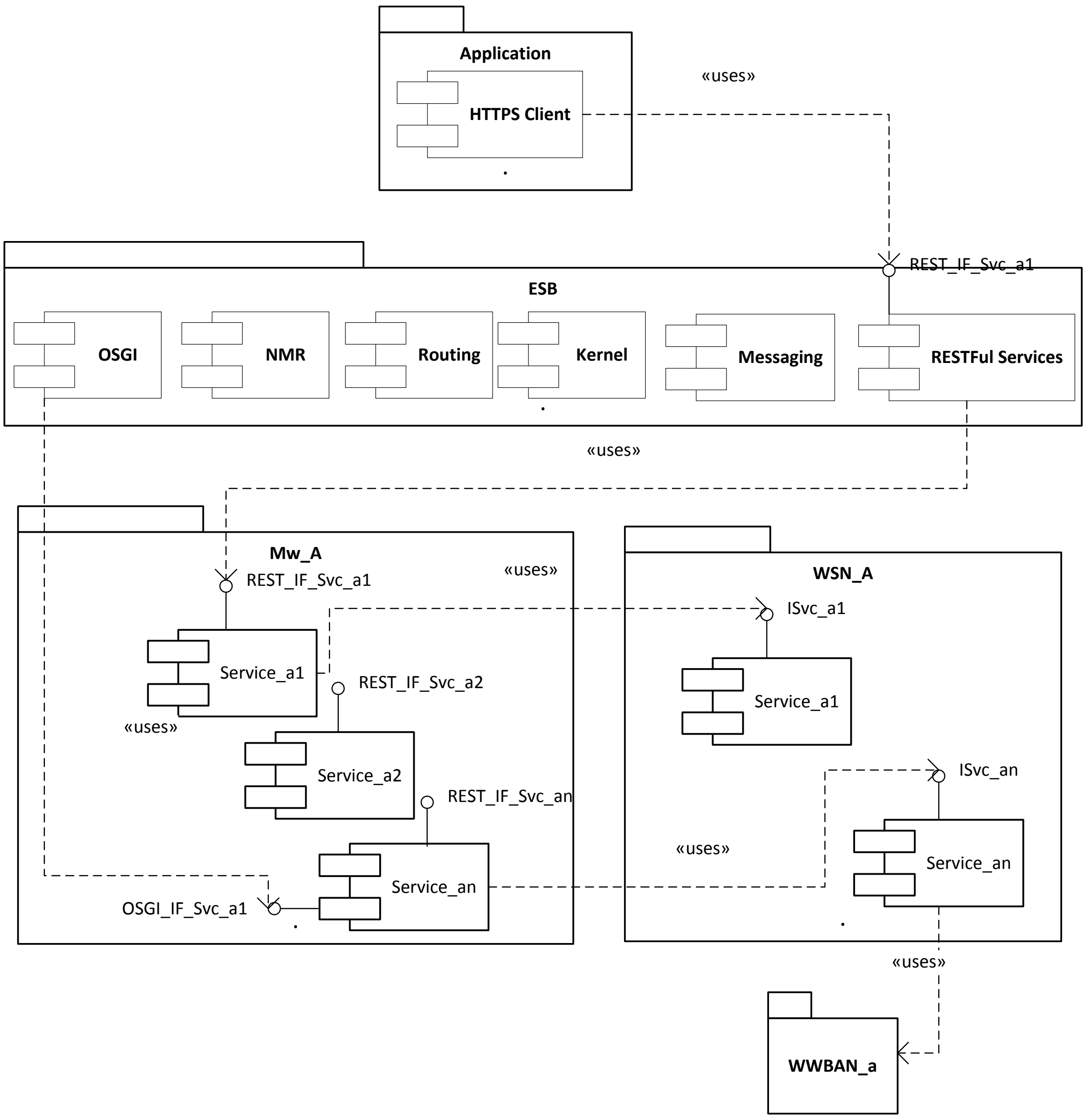


Figure 25 UML component diagram

The middleware solutions installed in the ESB are able to provide to the applications not only the single services offered by the WSN, but also to compose services using these single services.

From the applications point of view, these composed services have no difference with the single services. Indeed, applications are not able to distinguish between a single or a composed service, since the procedure to invoke the methods associated to the services is exactly the same.

In order to model the proposed elements of the architecture and the services (single and composed) provided by the middleware solutions, the following notations and definitions are introduced:

- MW_i = Middleware i installed in the ESB, for $i = 1 \dots n$
- WSN_j = Wireless Sensor Network j, for $j = 1 \dots n$
- k_j = Set of single services provided by a specific WSN, for $k_j \in \mathbb{N}^+$
- $SvcsMW_i$ = Set of services (single and composed) provided by middleware i
- $CSvcsMW_i$ = Set of composed services provided by middleware i
- $Svcs_{APP}$= Services available for applications
- n = number of middleware solutions installed = number of WSN connected (being one middleware connected to only one WSN)

As introduced in previous sections, the number of services available for applications (through REST interfaces) depends on: the number of single services provided by the WSNs connected to the middleware solutions installed in the ESB and, in addition, the composed services provided by those middleware solutions (i.e., the number of possible combinations using the single service composition mechanism).

Thus, for a middleware *i* and a WSN *j*, the number of services available for applications will be:

$$Svcs_{APP} = CSvcsMW_i + k_j$$

To obtain the number of services provided by each middleware installed, it is needed to calculate the number of combinations of single service, in addition to the single services provided by the WSN. For a specific WSN *j*, with k_j single services, the middleware *i* will be able to compose the following number of services:

$$CSvcsMW_i = \sum_{h=2}^{k_j} C_{k_j,h}$$

Where $C_{k_j,h}$ is the number of h-combinations (composed services) from a given set of k_j (single services).

Then, the total number of services available to applications will be:

$$Svcs_{APP} = CSvcsMW_i + SvcsMW_i = CSvcsMW_i + k_j = \sum_{h=2}^{k_j} C_{k_j,h} + k_j = \sum_{h=1}^{k_j} C_{k_j,h}$$

Let *n* be the number of installed middleware solutions, connected to *n* WSNs providing a number *k* of services.

Then, the total number of services available for applications will be:

$$Svcs_{APP} = \sum_{i=1}^{n} \left(\sum_{h=1}^{k_i} C_{k_i,h} \right) = \sum_{i=1}^{n} \left(\sum_{h=1}^{k_i} \binom{k_i}{h} \right)$$

Using Newton's generalised binomial theorem:

$$(a \pm b)^n = \sum_{k=0}^{\mathrm{n}} \binom{n}{k} a^{n-k} b^k$$

$$(a \pm b)^n = \binom{n}{0} a^n \pm \binom{n}{1} a^{n-1} b \pm \binom{n}{2} a^{n-2} b^2 \pm \cdots \pm \binom{n}{n} b^n$$

When *a=1*, *b=1*:

$$(1+1)^n = 1 \pm \binom{n}{1} 1^{n-1} 1 \pm \binom{n}{2} 1^{n-2} 1 \pm \cdots \pm \binom{n}{n} 1^n$$

$$2^n = 1 + \binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{n}$$

$$2^n - 1 = \binom{n}{1} + \binom{n}{2} + \cdots + \binom{n}{n}$$

Then,

$$Svcs_{APP} = \sum_{i=1}^{n}\left(\sum_{h=1}^{k_i}\binom{k_i}{h}\right) = \sum_{i=1}^{n}\left(2^{k_i}-1\right) = \left(\sum_{i=1}^{n}\left(2^{k_i}\right)\right) - n$$

Finally,

$$Svcs_{APP} = \left(\sum_{i=1}^{n}\left(2^{k_i}\right)\right) - n$$

Thus, the total number of services available to applications is given not only by the single services provided by WSNs, but also by the composed services provided by the middleware solutions installed and running in the ESB.

The number of composed services depends also on the number of single services and the possible combinations allowed by the composed services annotated in the service ontology. Consequently, the service ontology is a key element for the service composition, since the possibilities allowed by the ontology is one of the limitations to the number of total services provided to the applications. In following sections, a description of the extended ontology (in order to model services provided by wearable devices) is provided.

The services provided to the applications by the middleware solutions installed in the ESB are accessed through REST interfaces. The definition of these interfaces, as well as the protocol and the language for data interchange are provided in the next sub-section.

3.2.2. External ESB interfaces

Several interfaces for clients to access the ESB published services have been defined. In order to simplify the system and have lightweight clients, Representational State Transfer (REST) protocol and JavaScript Object Notation (JSON) messages are used. Both are well-known and widely used technologies in mobile and smart devices, giving the opportunity to external developers to create new applications using the provided application interfaces.

In order to provide the services needed for a specific scenario, it is interesting if the deployed middleware is able to include semantic annotations to the services published by the WSN. The definition of the scenario must include several interfaces to access the services published by the network. In Figure 26 there are some examples of these interfaces implemented for an e-Health scenario: pulse, distance, breathing rate and ECG.

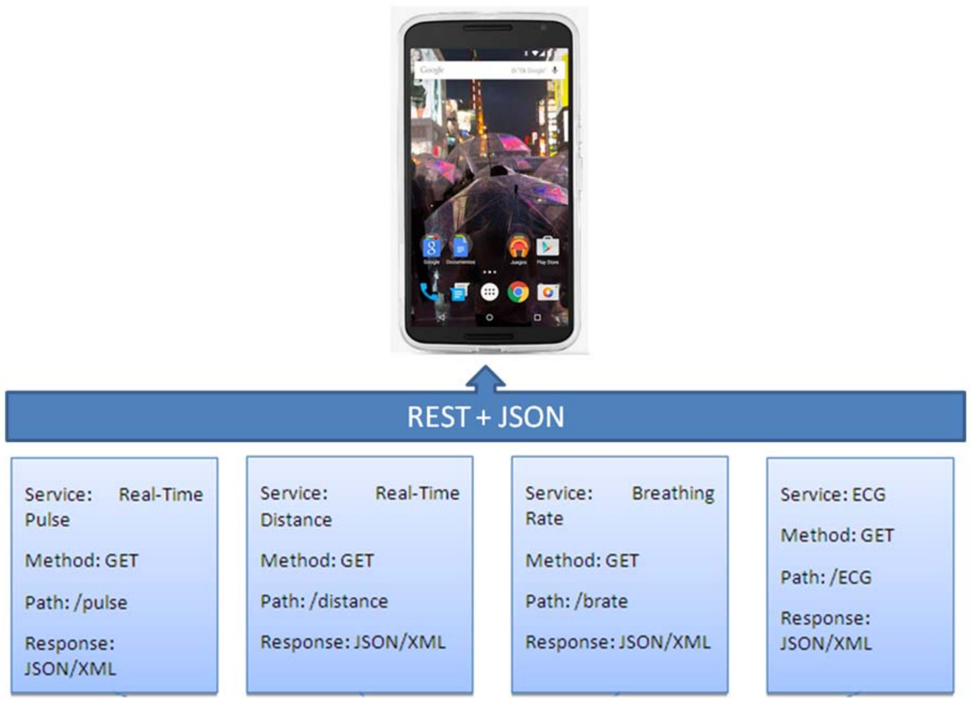


Figure 26 External interfaces for third party user applications (e.g.: a mobile application)

If a client wants to access any of those services, it needs to send a REST request (in a URL formatted HTTP packet) to the endpoint of the application, and the path of the service. The ESB will receive the request, route to the corresponding application and return the response to the client in the specified format. In this case, all the service responses have the same format (JSON/XML inside an HTTP body response).

3.3. Wearable InterDomain Protocols

So as to provide an easy way to interconnect both applications with information providers and sensors with ESB, some inter-domain protocols must be selected. The selection of the protocols and messages has been conducted taken into account the requirements of the wearable devices and sensor nodes (low computation power) and energy efficiency. Furthermore, the idea of providing a generic API for a large number of devices, manufactures and OS influenced in the undertaken decision. This solution is possible thanks to the ESB features that allow to update the protocols and upgrade the installed and running components in real-time, without system down periods. In addition, these changes remain totally transparent for the application layer.

The number of protocols and technologies currently available to be used in the field of WSN and IoT is large and it is growing. This is useful when integrating new optimized algorithms and protocols, but also it is an issue when designing application based in these protocols. The proposal is to integrate selected protocols from the already existing set, as well as to open the solution in order to include the forthcoming protocols and it is named as the Wearable InterDomain Protocols (WIDP): a set of protocols to easy the development of application for ubiquitous WSN and IoT platforms. A generic picture of the WIDP is depicted in Figure 27. The components will be presented in following sections.

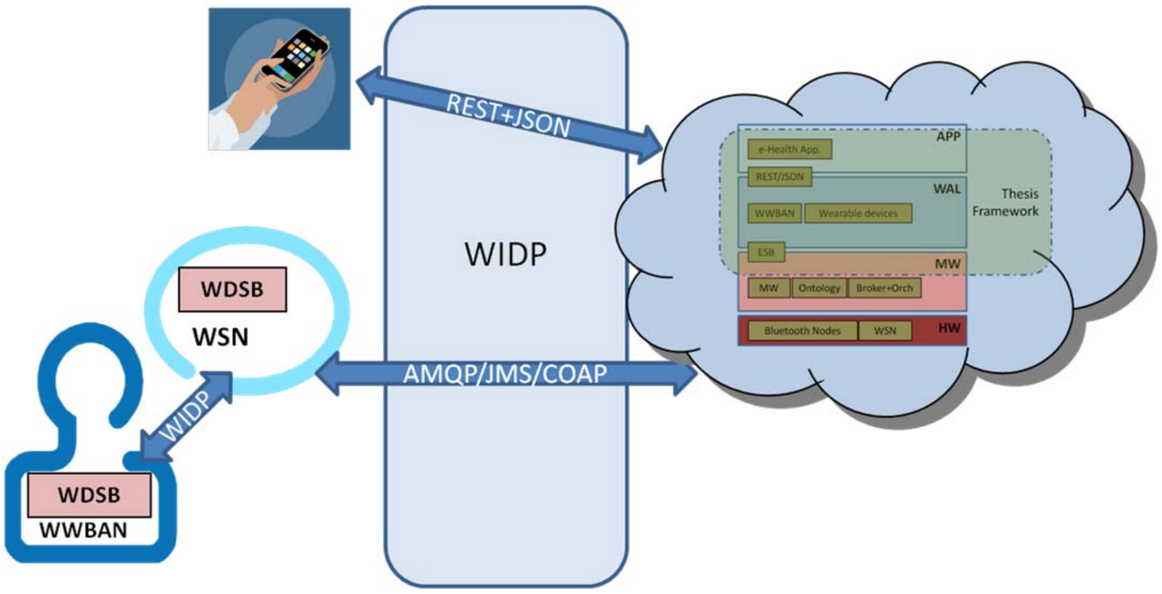


Figure 27 A generic picture of the Wearable InterDomain Protocols (WIDP)

3.3.1. Overview of technologies selected for the WIDP protocols

As commented in previous paragraphs, the number of protocols and technologies in the field of WSN and wearable computing is growing every day with new solutions. This proposal aims to integrate not only the today existing solutions, but also to provide a fully upgradable platform able to include all the forthcoming technologies. Following, a list of the most relevant protocols that were included for the definition of this proposal and that were tested in the validation stage is presented.

3.3.1.1. REST

RESTful[114] architectures (conforming to the REST constraints, a software architecture for distributed hypermedia systems) are based on HTTP protocol, but can be extended to other application layer protocols as long as they provide a representational state transfer. The basics are a client/server architecture where the clients send requests to servers. Servers receive and process the request and return the response. The basic information unit is the resource: a unique object that can be addressed through a Unique Resource Locator (URL). If a client wants to access a service it needs to send a REST command (in a URL formatted HTTP packet) to the endpoint of the application, and the path of the service. The server will receive the comand, route it to the corresponding application and return the response to the client in the specified format. Response should be codified in standardized formats, such as JSON or XML.

3.3.1.2. JSON

JavaScript Object Notation (JSON) is a text format for serialization of structured data. It is derived from the object literals of JavaScript, as defined in the ECMAScript Programming Language Standard, Third Edition. JSON can represent four primitive types (strings, numbers, booleans, and null) and two structured types (objects and arrays). A string is a sequence of zero or more Unicode characters [UNICODE]. An object is an unordered collection of zero or more name/value pairs, where a name is a string and a parameter value is a string, number, boolean, null, object, or array. An array is an ordered sequence of zero or more values. JSON's design goals were for it to be minimal, portable, textual, and a subset of JavaScript[115].

3.3.1.3. CoAP

Constrained Application Protocol (CoAP)[116] is a light protocol at the application layer (similar to HTTP but running over UDP) designed to run in low capability devices. CoAP messages can be as reduced as 4 bytes length. CoAP messages follow the same request/response as HTTP protocol. There are 4 types of CoAP messages, as presented in Table 12.

Table 12 Types of CoAP messages

Message	Functionality
CON (Confirmable request)	Response is mandatory
NON (Non-Confirmable request)	Response is optional
ACK (Acknowledgement)	The response for a CON request.
RST (Reset)	Response used when some error is issued

CoAP resources are identified by a URI (Universal Resource Identifier), including the method to be invoked. The default port for CoAP is 5683 (UDP). As an example, in order to get a measure from sensor through a standardized CoAP petition, the URI should be:

http://HOST_IP:PORT/services/sensor_ID/getMeasure{parameters}.

Table 13 Fields of a standard CoAP query URI

http	HOST_IP	PORT	services	sensor_ID	getMeasure	parameters
Protocol	Host IP address	Port open in the host	Root of the service server	ID of the sensor providing the measure	Command to be executed	Optional parameters (size, accuracy, period, date,...)

Following the publish/subscribe methodology, every client is able to subscribe to an event (data update) sending and “observable” request to the server. This procedure avoids clients to poll the server for new data. When new data is available, the server will send it to the “observers”. Block transfer for large data packet is also considered as an option in CoAP.

Every CoAP message is divided in 7 segments: version, message type, message code, message ID, token, options and payload, as shown in Figure 28.

- Version: CoAP version used
- Message type: CON, NON, ACK or RST, as explained in Table 12.
- Token length
- Message code: Request method or response code.
- Message ID: Unique ID for a CoAP message within a communication.
- Token: Used for re-assembling and re-ordering purposes.
- Padding: 1 byte
- Payload: This is the actual message that is being exchanged

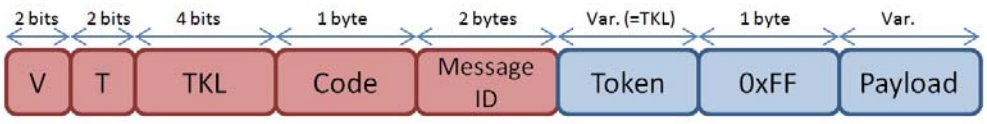


Figure 28 Fields inside a CoAP message. Mandatory fields are in red and optional in blue

3.3.1.4. JMS

Java Message Service (JMS) is a Java message oriented middleware that allows applications to send and receive messages. Designed by Sun and several partner companies, the JMS API defines a common set of interfaces and associated semantics that allow programs written in the Java programming language to communicate with other messaging implementations. JMS is part of the JAVA EE platform since version 1.3, providing the following key features [117]:

- Application clients, Enterprise JavaBeans (EJB) components, and web components can send or synchronously receive a JMS message. Application clients can in addition receive JMS messages asynchronously (applets, however, are not required to support the JMS API).

- Message-driven beans, which are a kind of enterprise bean, enable the asynchronous consumption of messages. A JMS provider can optionally implement concurrent processing of messages by message-driven beans.

- Message send and receive operations can participate in distributed transactions, which allow JMS operations and database accesses to take place within a single transaction.

3.3.1.5. AMQP

AMQP (Advanced Message Queuing Protocol) is an open standard protocol working in the application layer using TCP/IP stack. It is a message-oriented protocol providing queuing, routing and security mechanisms [118].

AMPQ defines the behaviour of both the server (called broker) and the client providing interoperability mechanisms for different protocol implementations. AMPQ also defines the data format to be interchanged as a byte flow. As depicted in Figure 29, the broker is able of manage several queues with different QoS for every connected client requirements. The message received by the broker is dispatched to the corresponding queue by the Exchanges, based on the routing rules defined by bindings. These bindings are included on message properties.

There is a growing number of AMPQ implementations, easing the application developing and providing multi-platform communication, such as:

- RabbitMQ
- ZeroMQ
- OpenAMQP
- Apache Qpid

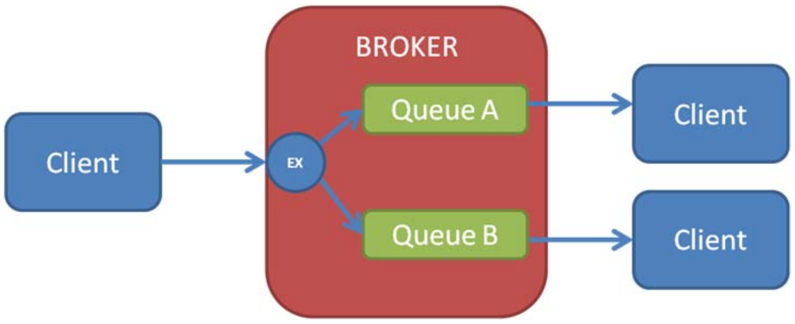


Figure 29 AMQP entities and broker components

3.3.2. UML diagrams for service requests

The complete UML sequence diagram describing a service request is presented in Figure 30. First, the WSN is initiated (at power-on stage). Meanwhile, the middleware will register it services in the ESB to be offered to the applications. If after the WSN boot-up process there are new services available, the service list will be updated.

The application is able to perform a service request using REST interfaces. Since the connection is performed using HTTPS protocol, the first step is to perform a SSL Handshake procedure (thoroughly presented in section 3.5.1.1). When the procedure is finished, the connection is secured and the service is requested by the ESB to the middleware which published that service. Then, the middleware will connect to the WSN providing the service in order to obtain the requested data. Finally, the data is returned to the requesting application using JSON or XML format.

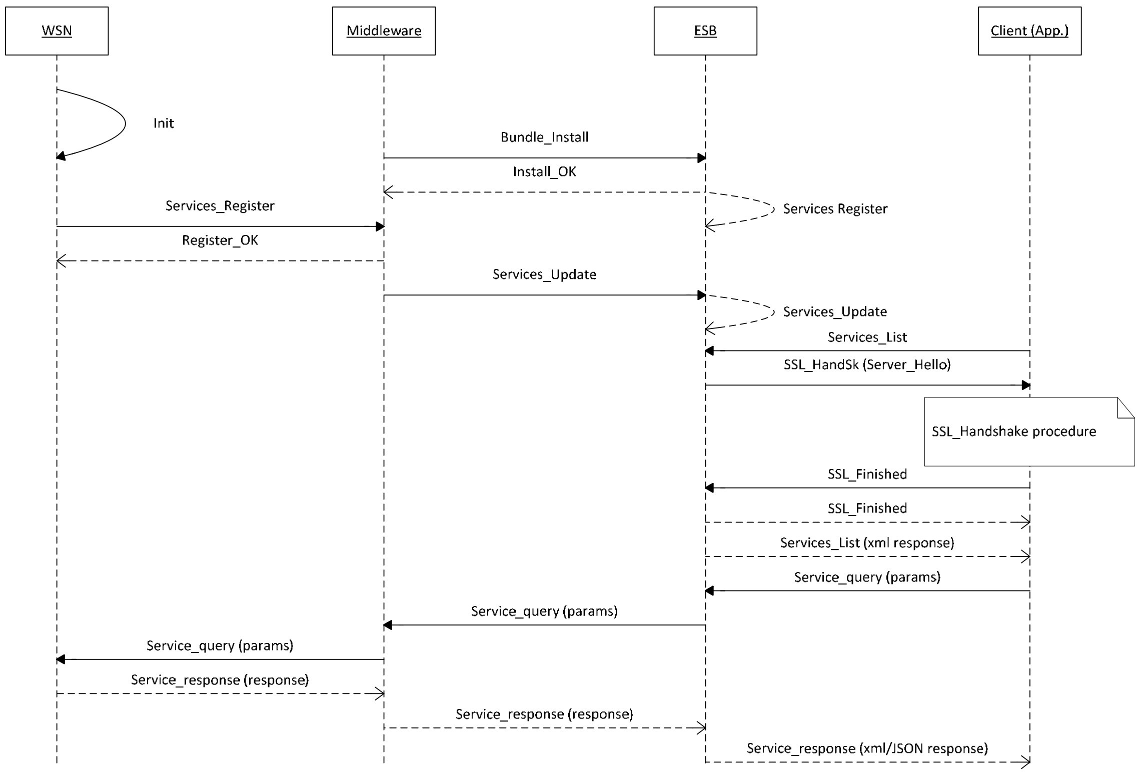


Figure 30 UML sequence diagram describing the complete service request process

A specific service request UML sequence diagram is depicted in Figure 30. In this diagram, a mobile application retrieves the heart rate from a Bluetooth sensor connected to a WSN. As commented in previous paragraphs, the starting process is the SSL handshake protocol in order to first, grant access only to authorized applications and second, to enable a secure connection between client (the mobile application) and the server (the ESB). The request of the heart rate will be then performed by the client using the REST interface provided by the ESB. The request will be routed by the ESB to the middleware who published that heart rate service. Then the middleware will connect to the WSN which will also connect with the Bluetooth heart rate sensor worn by the user specified in the field user_ID (see Table 15 for further details). Finally, the heart rate of the user will be provided in an XML message to the client which would be able to provide this information using a Graphical User Interface.

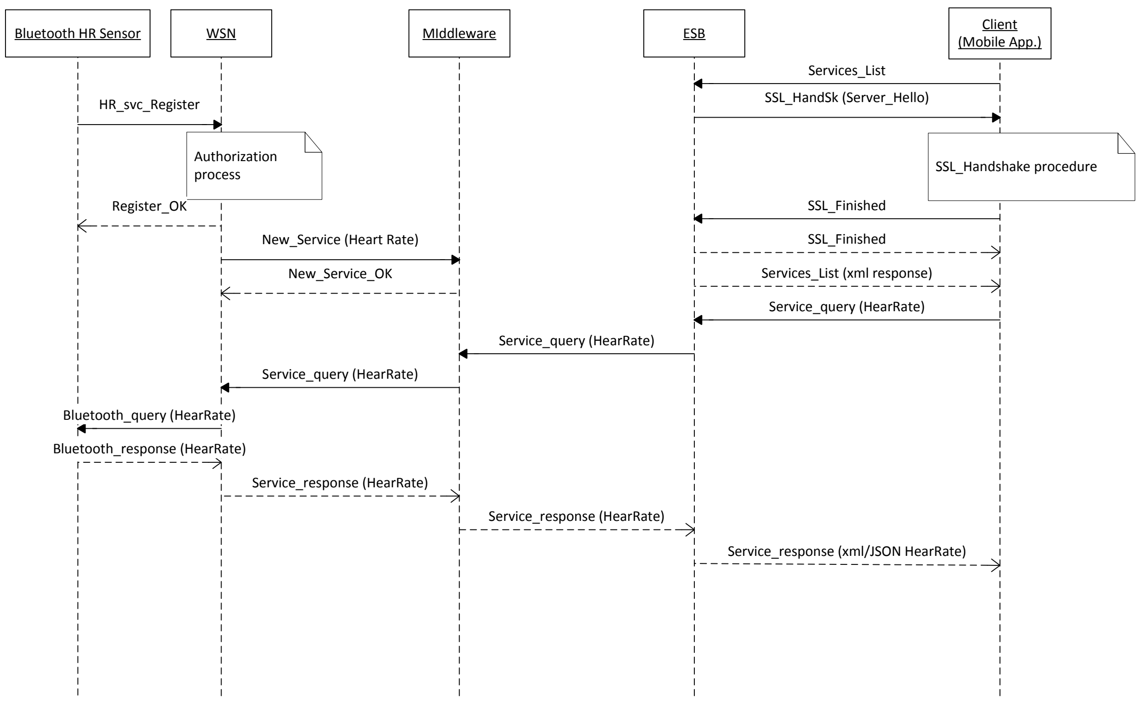


Figure 31 UML sequence diagram: a mobile application retrieves the heart rate from a Bluetooth sensor connected to a WSN

3.4. Service-oriented semantic middleware integration and services deployment definition

The integration solution proposed in this thesis eases the integration of different WSN protocols and technologies, but it does not solve by itself the data format translation issue. Thus, it is proposed to integrate an existing semantic middleware based on an extended ontology to model wearable devices capabilities.

An ontology is a formal and semantic representation of a set of concepts and the relationships between those concepts within a domain. An example of a semantic solution is SOUPA[119]. SOUPA is a proposal for a Standard Ontology in Ubiquitous and Pervasive Applications that defines core concepts by adopting several consensus ontologies. Some concepts defined in SOUPA ontology were used to model the context information presented in the proposal. For example, to describe the environment where the user is, the OpenCyc[120] Spatial and RCC ontologies were used. They include SpatialThing that is related to LocationCoordinates class. These ontologies have been extended with the Location class to describe the different areas that compose an environment by using a symbolic representation more intuitive for users (e.g. kitchen, corridor, etc.). In addition, the term EnvironmentProperty is included to describe the properties (e.g., lighting intensity, presence detection, noise level, etc.) of a certain location.

To describe the system, the terms Service, ServiceCategory, Operation, Argument, and Process were defined. The central term is Service, which represents the services (e.g., Lighting, Multimedia Player, Alarm, etc.) that the system provides. Services can be classified into categories and are described by means of the following information: Profile (the public description of the service), Process (the logic of the service) and Context (the context conditions in which the service is provided).

To describe the users of the system, the FOAF[121] specification and the SOUPA ontology were re-used, which proposes the term Person. This term is described by a set of properties that include profile information (e.g., name, gender, birth date, etc.), contact information (e.g., email, mailing address, phone numbers, etc.), and social and professional relationships (e.g., people that a person knows, relatives, etc.). In order to properly describe the users, the UserProperty class has been added, to represent the properties of users, such as user preferences (e.g., exercise routines, personal marks, thresholds, etc.). With regard to the location where a person is, the currentLocation relationship is defined, which relates each person to the location where they are at the current moment. A person is also associated to policies. A policy represents a set of operations and/or services (which group a set of operations) that are permitted for a person. The policy also describes the context information that a person can see and/or modify.

The described ontology was implemented within the GRyS research group using the Web Ontology Language (OWL)[122], a W3C standard ontology mark-up language that greatly facilitates knowledge automated reasoning.

A service can be defined as the abstract fashion to model a capability provided by a WSN node or a wearable device. In this thesis, the proposal is to deploy within the proposed integration platform a set of middleware solutions with semantic capabilities. These middleware will provide services to be used by the high-level applications.

Simple services are the most expectable from a WSN or a WWBAN, for they are the parameters that are collected from the nodes, either when they are retrieved from the context environment (as the temperature) or from an wearable device that is connected (as a heart rate sensor). Commonly, they are requested by the human operator –either locally using the Enterprise Serial Bus, or any other remote appliance with the capability of using REST- in a manner that will be dependent on whether the requested serviced is offered by a sensor from the WSN or from the WWBAN.

As introduced in the proposal model, a key feature of these semantic middleware solutions is that they make possible offering services that are retrieved not by actual nodes with actual sensors, but as a result of a process of data aggregation, in a way so similar for any human user of this system that it is defined as sensor virtualization. Under this principle, the information measured from context (temperature from different nodes) and WWBAN data (breathing rate, body temperature, heart rate) will be merged into a new single service that is obtained as any other and has been registered similarly, with the only particularity of compulsory using the node with the Orchestrator agent to have this kind of services. These new services obtained from using sensor virtualization are named as composed services. It can be considered that the orchestrator agent, regardless of not using any measure by itself, always offers composed services present in a Wireless Sensor Network and it will be the orchestrator agent the one to send them to the broker to have them registered. Wearable Extended Ontology

As defined in the first section, a middleware is an abstraction layer between the WSN and the application layer. The middleware architecture used in this proposal is a service oriented semantic middleware developed in GRyS research group, named nSOM[123], a service-oriented framework designed for WSNs. Based on a dynamic service composition model and a semantic knowledge management scheme, it provides an agent-based virtual sensor service approach that is able to create new composed services combining the existing ones. nSOM is being extended to include more semantic and context-awareness services.

The services provided by the WSN to the applications rely on measures obtained by the sensors (environmental or physiological). These measures are delivered to a broker that compounds and offers the service. Moreover, this semantic middleware architecture proposes the semantic annotation of the service provided by WSN to get a couple of advantages. The annotation of the services is based on an ontology developed specifically for these services. The ontology used here is not sensor-related but it is related with the services provided by low capabilities devices integrated in a WSN. Thus, this ontology provides several advantages provided by semantically annotated services:

- Applications that use these services can retrieve from the ontology repository the profile and the features of the services, being able to choose the most appropriate according to its function.

- The service composition can be done within the WSN, so generation of new services is a new feature provided by the WSN.

- Accurate knowledge of the environment in which services are provided.

- Controlling the function of the service.

- Adequate the service to the current state and capabilities of the Wireless Sensor Networks

```
/*global annotation*/
   {/*service*/
        "profile":{/*start profile*/
              "identification":{
                   "serviceName":{
                       "identif":"Real-Time Pulse",
                       "idService":"idInto SmartSpaceLifewear"
                                 },
                    "serviceDescription":"RT Heart rate",
                    "owner":"lifewear"
                              }
              "funcionality":{
                              "preconditionDescription":"empty",
                              "inputDescription":"empty",
                              "outputDEscription":"pulse rate"
                            },
              "security":{
                          "policy":"security policy lifewear",
                          "dataProtection":"integrity"
                   }
              "grounding":{
                           "inputMessage":"empty",
                           "outputMessage":"control, integer",
                           "endPoint":"/lifeware/pulse/"
                           }
           },/*end profile*/
```

Figure 32 Example of the extended ontology

The ontology description is divided in three parts: profile, process and context. Profile is the description of the features of the service, and it must be published in an ontology repository in order to be queried by applications or other users before the use of the service. The profile class is composed of service_identification, service_funcionality, security_profile and grounding. Process description of the service is the most important part of the ontology, as it contains the classes related with the process that is behind of the service delivery. The process class is composed of operation, atomic_process, aggregated_process, input and output. A key feature of this ontology is the specification of the context condition under which the service is provided. There are several differences when providing a temperature service indoors or outdoors or a heart-rate service at sea level or on the top of a mountain. The context information can be written in the ontology repository and can be used by processes to provide the service. An overview of the most significant classes included in the extended ontology is depicted in Figure 33.

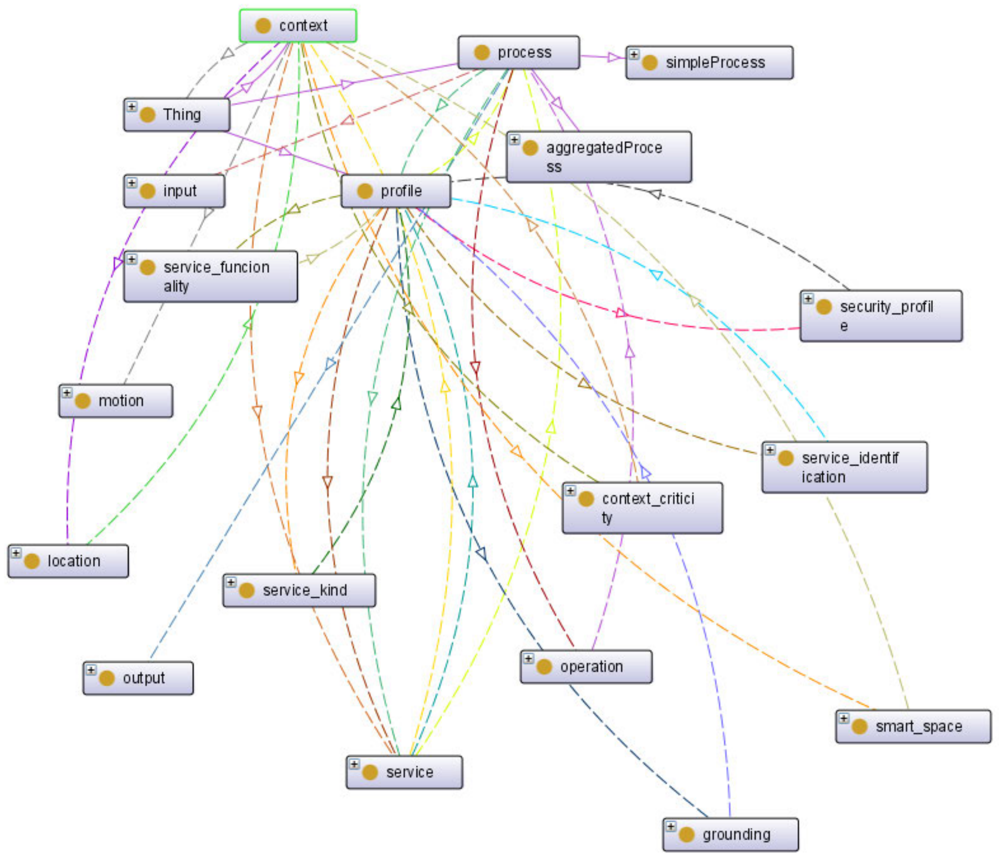


Figure 33 Overview of the ontology extended with wearable classes

Service composition can be performed using the base of the already existing services. The new service will be published as a “virtual” service, by a special node called orchestrator. The virtual service is, for all the nodes in the network but the orchestrator, a similar service to the non-composed ones. When a request for a service reaches the network, the broker will deliver the request to the node that has previously published that service: a sensing node if it is a single service or the orchestrator node if it is a composed service. Once orchestrator receives the request, it will be split into single service requests. All of these requests will be sent to the corresponding nodes and when the orchestrator has all the data needed, it will reply to the composed service request. The only expected difference when retrieving a single or a composed service will be the time needed to process the request

3.5. Security schema proposal

As commented in the state-of-the-art analysis, security is a key issue to consider when deploying applications that share private and sensitive data (such as bio-medical or personal information). In order to provide a full security solution to protect data and the access to the service, deploying a two-tier schema is appropriate. This schema is presented in Figure 34.

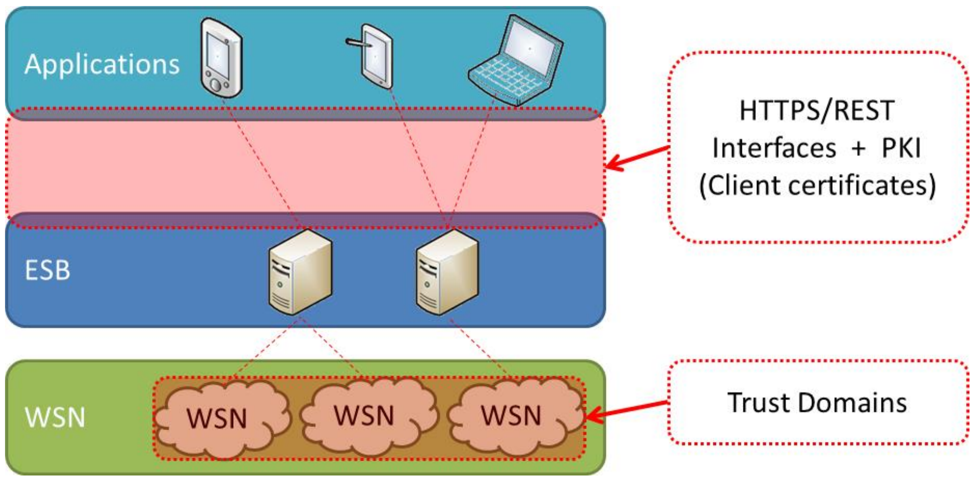


Figure 34 Two-tier security proposal (PKI and Trust mechanisms)

In this schema, the high-level interface is the upper tier, where the access from the applications must be secured (checking whether the application has permission to access to the required services as well as ciphering the connection between the end device and the server). The lower tier is the network providing the data (in this proposal, the WSN). For this lower tier, the communication channel (wireless transmissions) and the node identity must be secured.

A full description of the security mechanisms deployed for these two tiers is presented in the following sections.

3.5.1. Securing high-level interface access

In order to enable third-party application to access to the data and services provided by the WSN, the high-level interface must be public and available for any device with internet connection. This is a double-side feature: on one hand, it permits every application with internet access to retrieve the WSN data; on the other hand, it also exposes the services to possible attacks and unauthorized accesses.

In this proposal, the high-level interfaces have been designed following the RESTFul specification. Thus, every access must be performed using an HTTP connection to the endpoint provided. A good solution to protect this interface from outside attacks is to deploy a Public Key Infrastructure where every application wishing to connect to the high-level services must probe first its identity (sending its own certificate). In this way, the connection changes from HTTP to a secured protocol (HTTPS). In Figure 35, a general overview of PKI components is shown (application certificates, ESB certificate and Certificate Authority -CA-). This solution is thoroughly presented in next paragraphs.

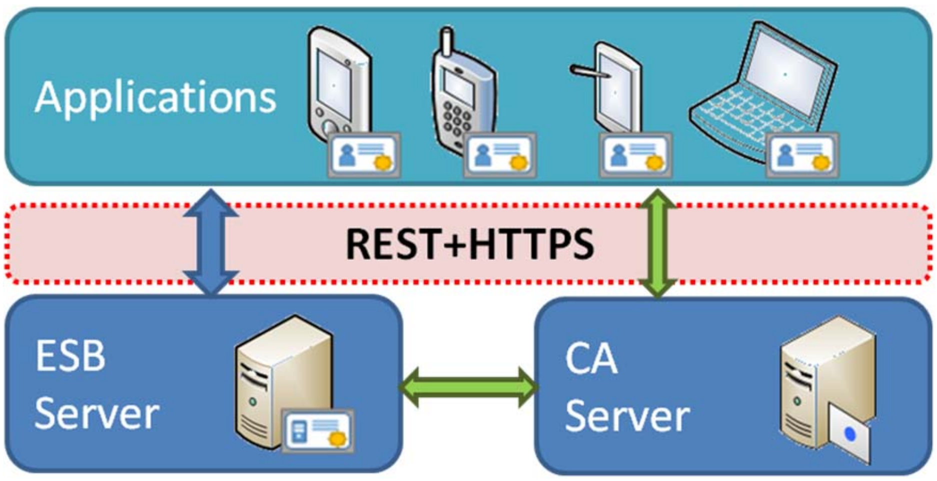


Figure 35 PKI components to secure high-level interfaces

As presented in previous paragraphs, it is possible to identify in the deployment three distinguished blocks:

- **Certificate Authority (CA).** The main task of this trusted entity is to issue, maintain, validate and revoke client and server certificates. The CA receives the Certificate Signing Request (CSR) from the client, and after some verifications it signs the certificate and return it to the sender. Furthermore, the CA issues a Certificate Revocation List (CRL) including the certificates that have been revoked and are no longer valid.

- **ESB Server with SSL enabled.** An ESB server with public REST interfaces and SSL support enabled in order to accept HTTPS requests. The server is configured to require client authentication as well as server authentication. After server authentication is successfully completed, the client must also present its certificate to the server so as to authenticate the client identity before the encrypted SSL session can be established. Then, the connection changes to HTTPS enabling secured connections.

- **ESB Server Certificate**. This is a certificate issued by the CA to the ESB server in order to permit applications to perform the authentication of the ESB server.

- **Application Certificates**. In a similar process, the CA issues certificates to every authorized application, allowing them to securely connect to the ESB server. These certificates should then be installed in the client application.

3.5.1.1. SSL Handshake Protocol

In order to query either for the list of available services or for a specific service, clients (applications) must provide a valid certificate prior to establish a secure connection with the server. The first step to accomplish is a SSL handshake protocol, as specified in the SSL definition[124].

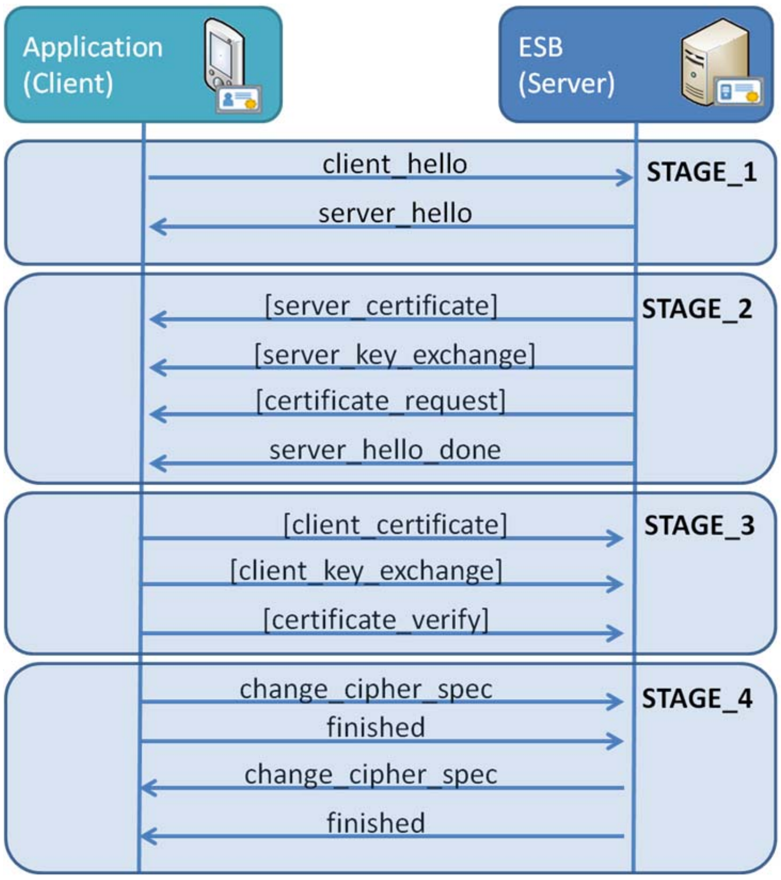


Figure 36 SSL Handshake procedure messages

As shown in Figure 36, several messages (optional messages in brackets) need to be inter-changed between the client (the application wishing to connect to the WSN services) and the server (the ESB server connected to the WSN). The process can be split in 4 stages:

- **Stage 1.** The first message that the client needs to send to the server is a client_hello. Then the server sends the server_hello message as the respond. This stage performs the exchange of security enhancement capabilities between client and server including the next attributes: Protocol Version, Session ID, Cipher Suite, Compression Method, and initial random numbers.

- **Stage 2.** This is an optional stage in order for the client to perform the server authentication. If client authentication is also required, the server may request a certificate from the client. Finally, the server_hello_done message is sent by the server to indicate the end of this stage.

- **Stage 3.** After receiving a server_hello_done message, the client sends the client_certificate message. The following step is the generation of the premaster-secret by the client. Then, this secret is ciphered with the server public key. The client sends this information along with a digitally-signed Certificate Verify message in order to provide explicit verification of a client certificate. When the server receives the message, it obtains the premaster-secret deciphering it with its private key.

- **Stage 4.** Based on the premaster-secret, both client and server generate the master-secret that will be used to generate the session keys. After that, the client sends a change_cipher_spec and all the following messages will be sent ciphered with the session keys. Finally, the client sends the finished message. The server will respond with another change_cipher_spec message. The finish message send by the server ends with the handshake process.

3.5.2. WSN security

A project to deploy a Wireless Sensor Network must be able to deal with the challenges related to the intrinsic characteristics of these types of networks. One of the most important of these challenges is energy consumption. Wearable nodes operate autonomously (and are usually battery-powered), so it is necessary for the processes and algorithms used to be efficient and have an energy-saving focus. Related to this, the aggregation of information through the network can be an interesting mechanism: some repeated data is not taken into account, not sent and, in this way, energy and resources are saved. Since these networks are being used in a wide variety of applications, security challenges must be addressed too. For example, in e-Health environments where biometric data and medical records are being shared, different national and international privacy laws are applied. As far as WSN security specific problems are concerned, it has been concluded that security in WSNs must support generic applications and the basic operation of the network, taking into account the following facts:

- these nodes are tiny devices
- they are energy- and resource-constrained regarding computational power, the amount of memory, radio bandwidth and coverage
- there are no tamper-proof zones in the commercial hardware platforms to store sensitive information (passwords, keys, identifications, etc.)
- due to exposure to adverse environmental conditions they tend to fail or lose their power, and may be removed by unauthorized personnel
- radio communications can be intercepted and crypto-analyzed to extract data, keys, etc.

As a wireless system, security challenges are greater than in wired networks because of the open access to data transmissions. By using broadcast transmissions, other devices listening in the same frequency may intercept every communication between two nodes. Another WSN challenge is the possibility of a node being captured and crypto-analyzed by a third party. If security keys, policies and other cryptographic elements are accessed, a new spurious node could be introduced in the network, and several attacks can be deployed: Denial-of-Service, Man-In-the-Middle, data sniffing and/or data modification, routes spoofing, etc.

The security proposal, as described in next sections, is a trustworthy domains model to deploy security services (authentication, integrity and privacy) in a WSN. It includes a complete trusting scheme to accept, control and exclude nodes participating in a trusted domain and defining security policies using symmetric and asymmetric (PKI) cryptography. Considering the WSN nodes constrictions, security schemas have been designed to reduce the cryptographic material stored in each node. It also minimizes the data-loss suffered when one of the nodes is compromised. In order to achieve the resilience goal, mechanisms for re-configuring a compromised trusted domain (with re-keying or server re-assignment) have been defined. This is important for e-Health applications, were data security and quality of service are severe requirements.

3.5.2.1. Deploying trust domains to increase WSN security

As WSNs are being widely used in many applications (some of them with severe requisites such as critical infrastructures monitoring, defence, surveillance, e-Health, etc.), including security mechanisms is of major importance. Due to node limited capabilities, traditional network security schemas are not suitable, so new schemas must be designed.

Privacy is an important feature to care about. If WSNs are utilized to share private data (medical records in hospital monitoring, timetables in access control, location in tracking systems, etc.), some mechanisms must be implemented to protect private information from unauthorized access.

Access control is also an important mechanism. In order to protect data in the network, some control steps must be applied to give access only to authorized nodes. To gain access, nodes must be authenticated inside the network.

Finally, in order to ensure that the data received by a node has not been modified in the route taken from the sender, an integrity control mechanism is necessary.

All of these mechanisms can be addressed in a trustworthy domains environment, where any new node joining the network must authenticate itself to obtain the necessary keys to start sending and receiving data. When nodes are interconnected, several surveillance policies must be applied in order to exclude a suspicious node from the network, preventing the node from participating in it anymore. The aim of deploying a trust domain is achieving a platform-agnostic security mechanism that can be used regardless of network topology or application requirements. Resilience is a very important concern too. In fact, if a stand-alone and self-configuring WSN is needed, an auto-setup and a re-configuring mechanism are required as well, so as to maintain an acceptable level of service in the face of security issues.

3.5.2.1.1. Trust definition and trust domains

Trust implies reliance on another person or entity. In the field of knowledge related to common networks, trust can be determined by the reliance that each one of the nodes has on the others. Thus, it is very important to measure this feature in some kind of way. In WSN, trust increases its importance. Since it is very easy to introduce a new node in a deployed WSN, capture and analyze the data traffic, some mechanisms must be developed in order to measure and assure that all the nodes participating in a WSN can rely on each other. In this proposal, several mechanisms are included: trust domains, polling systems and measures to undertake when trust has been lost in a domain. Overall, steps are needed in order to have a functional trustworthy mechanism.

The first step is trust establishment: a new node participating in the network must publish its authorization to start communications with other nodes. Once the new node has been granted access, trust is propagated throughout the network.

3.5.2.1.2. Validating trust domains and polling system

When a trust domain has been defined, and all the elements needed have been deployed (keys, roles, policies, etc.), nodes are able to start registering services and sending data. Validation of the trust domain must ensure that all the nodes participating in each domain can reach other nodes and have obtained both the keys and the security policies. Trust must be measured in some way to determine if one trusted node has been compromised or has an unexpected behaviour (in case a re-trusting mechanism has to be started). In this proposal, trust is measured by a polling system.

Once the domain has been validated, and the trust in that domain can be assured, the network will start its usual working. As an expandable network, new nodes can be inserted within the ecosystem to extend the range or the services offered by the network itself.

The way to assure that new nodes behaviour does not compromise the prior trust level is by watching the new nodes actions and comparing them with the policies defined. This surveillance system is distributed across the network, each node can report to the head-node (broker, security manager or cluster head node) the actions taken by the other nodes. In this way, compromising the network due to new nodes, or a trusted node that changes its behaviour to a risky one, can be avoided.

The polling system is simple and effective: each node assigns votes for every neighbour in its range. In a WSN, each node can promiscuously read the data sent by its neighbours, because communications are broadcasted. Data sent means messages generated by each node. If these messages do not match the rules defined in the security policy, a negative vote is emitted. The security manager should count votes for each node, and if the negative result exceeds a pre-defined threshold for a node, some actions might be taken: waiting for pre-defined time (quarantine), excluding the node from the network, re-defining a new trust domain, etc.

3.5.2.1.3. Actions to take when losing trust in a domain

When one of the nodes participating in a trust domain becomes compromised, the whole domain should be treated as compromised. Several actions must be taken in order to re-establish the trust in that domain. If the compromise affects a regular node, it is enough to renew the key that protects communications within that domain and then redistribute the new key to all the nodes in the domain but the compromised one. If the compromised node is the domain server (or cluster head or broker, as previously named) it will be also necessary to designate a new domain server and, right afterwards, renew the key.

In the following sections the deployment model to evaluate this proposal will be described, including all the mechanisms to define, evaluate and redefine a trust domain.

3.5.3. Security schema definition

In previous paragraphs the mechanisms to define security domains and establish trust in WSNs have been proposed. In order to address all the characteristics from the proposal, it has also been designed an implementation model to deal with all the security issues regarding security domains. The proposed model based on previous section directions is composed by several elements, which are presented below in Table 14.

Table 14 Components of the proposed model

SM	Security Manager
TD	Trust Domain
TP	Trust Policy
DK	Domain Key
DKn	Domain N Key
DKS	Domain Key Server
N	Node
NID	Node ID
NK	Node Key
CS	Ciphering Suite
Kp	Public Key
Ks	Private Key

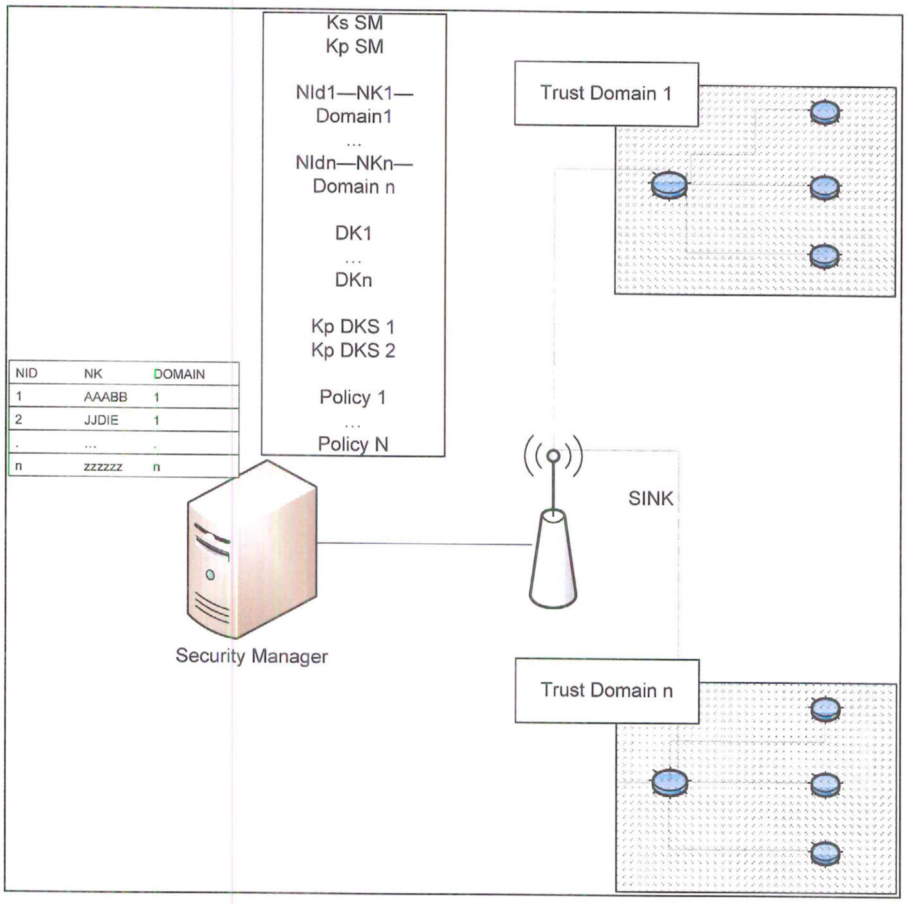
The Security Manager (SM) is the principal actor of the model, and should be an external host able to generate symmetric and asymmetric keys in a secure manner. Several nodes (N) sharing the same Domain Key (DK) and some common aspects (mission, localization, capabilities, etc.) define a Trust Domain (TD). Each TD has a Trust Policy (TP), including the behaviours, measures, actions and other features regarding the life-cycle of a Trust Domain. Also, a Domain Key Server (DKS) exists in each TD. It is a special node in charge of distributing the Domain Key when needed. A single node can act as a DKS if the Security Manager decides it. Any node in the network has its own Node ID (NID), unique identification and an individual Node Key (NK). The ciphering methods and key sizes used when ciphering a message are defined inside the Ciphering Suite (CS). Two types of keys are used in Public Key

Infrastructure (PKI) cryptography: Public Key (Kp), the part of the key assumed to be freely distributed, and Private Key (Ks), the part of the key to be kept in secret and not distributed.

Since WSN nodes have several limitations, a full and robust Public Key Infrastructure is unlikely to fit in these nodes. Due to this reason, a hybrid PKI and symmetric keys schema has been used in this proposal. PKI is used for the communication between special nodes (Security Manager and Domain Key Server), and a simple-but-secure AES symmetric key for the plain nodes. The size of this AES key is constricted by the nodes computing power and, in order to have a scalable architecture, it becomes defined by a Ciphering Suite field in the ciphered data packets, so as to determine the length of the key or select a new algorithm (such as RC5, etc.).

3.5.3.1. Architecture

The system architecture of the proposal is based on three main components: a Security Manager, Trust Domains and Domain Key Servers, as illustrated in Figure 37.



NID	NK	DOMAIN
1	AAABB	1
2	JJDIE	1
.	...	.
n	zzzzzz	n

Figure 37 Proposed model overview

The Security Manager is an external entity (but connected by the sink to the WSN), where high level security mechanisms have been deployed. Security Manager tasks include: Node Key generation and distribution, Security Policies definition and distribution, Domain Key Server role assignment, re-keying processes, and record data regarding to nodes participating in each Trust Domain.

For each Trust Domain defined, a Domain Key Server is assigned (either in deployment or run time) and a Domain Key is shared. Any node participating in a node can assume the DKS role if the former DKS has been compromised. If the compromise affects only a simple node, a re-keying method is started: DKS asks SM for a new DK, and then this new key is sent to all nodes but the compromised, using each Node Key.

Trust Domains definition can be done following several policies: number of nodes per domain, node localization, domains for specific functionality (e.g.: temperature measurement, intrusion detection, environmental monitoring, etc.) or any other distribution demanded by a specific application.

Focusing in a generic Trust Domain (illustrated in Figure 38), the main components and elements will be described.

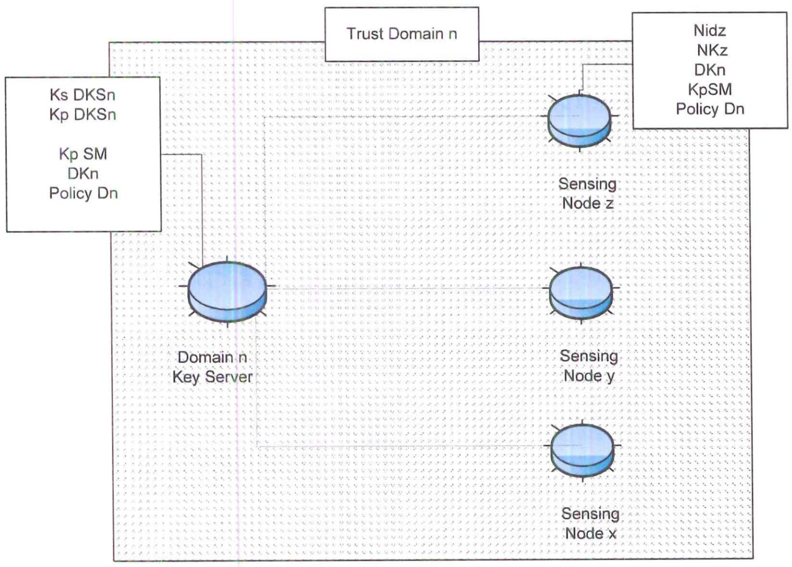


Figure 38 Generic Trust Domain components

Domain Key Server stores its own pair of public and secret keys (and also the public key of the Security Manager, so as to communicate with it), the Domain Key (to communicate with nodes participating in the domain) and the security policy for the domain.

The other nodes also need to keep some important information: the node id and key, the Domain Key, the public key of the Security Manager (used if Security Manager assigns the role of the former Domain Key Server to a new node), and also the domain security policy, where the surveillance parameters and polling system are described.

Cryptographic information shared by nodes has been reduced to a minimum, in order to minimize the risk when a node is compromised. In this schema, if a single node or a DKS is compromised, just the Domain Key is evidenced (as the public key of the SM can be revealed without security issues). Re-distributing a new DK or designating a new DKS will be enough to recover the trust. Both processes will be described in next paragraphs.

3.5.3.2. Communications

Several messages have been defined to model the communication between the elements of the system. There are messages to join a domain (HELLO), renew the Domain Key, query and send a Node Key, assign a new Domain Key Server when the active one has been compromised and renew the compromised keys. In next paragraphs, most representative of these message functionalities are described using sequence diagrams.

The first step that a new node has to complete when entering a WSN is joining a defined Trust Domain. A HELLO message is sent by the new node (Node1 in Figure 3). Data in this message includes: a random number (x) ciphered with the node key (only DKSs having that key will be able to decipher it), the Ciphering Suite used, and its unique node ID. This message is broadcasted to one (or several) active Domain Key Servers, and the DKSs will prompt the Security Manager for the Node1 key (using Node ID received as an index) to be able to cipher and decipher data to/from Node1.

The SM will decide which one of the querying DKS will be answered (the SD where the new node should join). The DKS designed will receive the Node1 key and will use it to cipher the data included in the HELLO REPLY message. These ciphered data includes: the random number sent by Node1 (in order to confirm to Node1 that the responding DKS is a trusted one) and the key for Domain participation. Additionally, the Node ID is added.

When Node1 receives the HELLO REPLY message, it will use its own Node Key to decipher it obtaining: the random number (x) previously sent in the HELLO message (checking that the replying DKS is a trusted one, since it has been able to decipher the HELLO message) and the Domain Key for the Trust Domain assigned.

At this point, Node1 has access granted to Trust Domain 1, since it has obtained Domain 1 Key and is able, for example, to publish its services to the network (ciphered with the Domain 1 Key) and then they get published by a Service orchestrator or broker existing in the WSN.

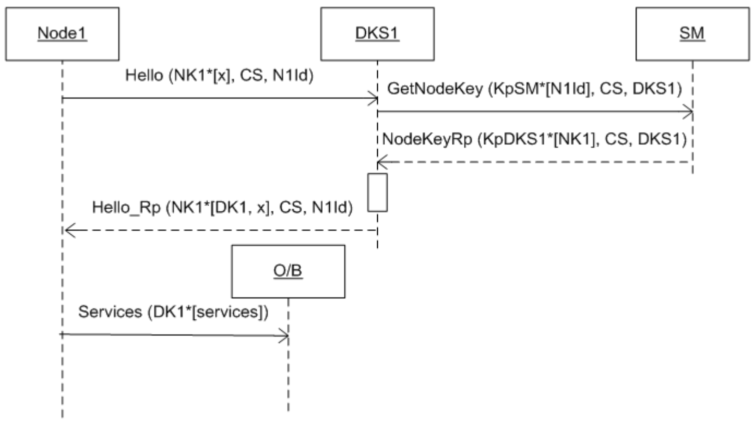

Node1
DKS1
SM
Hello (NK1*[x], CS, N1Id)
GetNodeKey (KpSM*[N1Id], CS, DKS1)
NodeKeyRp (KpDKS1*[NK1], CS, DKS1)
Hello_Rp (NK1*[DK1, x], CS, N1Id)
O/B
Services (DK1*[services])

Figure 39 Sequence diagram describing a new node registration into a Trust Domain

Another interesting process is the ability of the system to renew a Domain Key that has been compromised (for example, if a node is captured and crypto-analyzed). This process, described in Figure 4, starts with a DKRenew message sent by the DKS to the SM containing: the compromised key (ciphered with the SM public key -Kp- guaranteeing than just the SM will be able to open it), Ciphering Suite used and DKS1 identification. The SM will respond with a message containing the new Domain Key (ciphered with DKS1 public key), Ciphering Suite used and DKS1 identification. In order to re-obtain Domain1 trust, the new Domain Key must be sent to all the nodes but the compromised one. A DKRenew message is used, which contains the new DK ciphered with the individual node keys (one message per node).

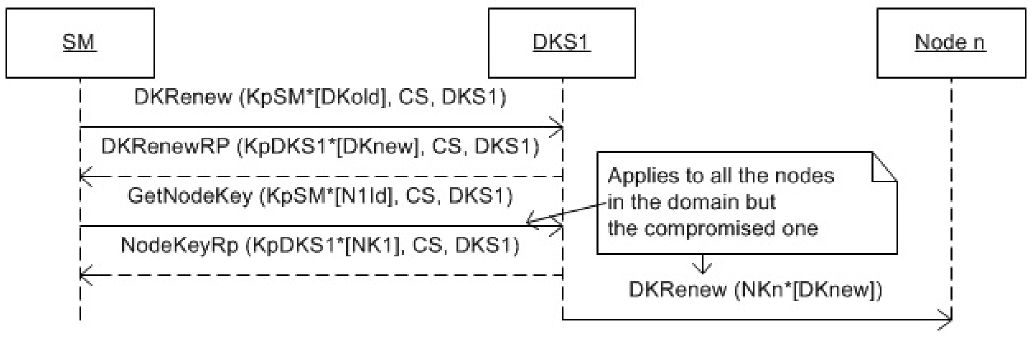

SM
DKS1
Node n
DKRenew (KpSM*[DKold], CS, DKS1)
DKRenewRP (KpDKS1*[DKnew], CS, DKS1)
GetNodeKey (KpSM*[N1Id], CS, DKS1)
NodeKeyRp (KpDKS1*[NK1], CS, DKS1)
Applies to all the nodes in the domain but the compromised one
DKRenew (NKn*[DKnew])

Figure 40 Sequence diagram describing how the Security Manager starts the renovation of a Domain Key

The worst case is when the compromised node is the Domain Key Server (which can also be named as the cluster head or the broker as presented in section 4.2). The Security Manager will choose a new node to assign the role of DKS. This election can be done using several variables, defined in the Security Policy: node reputation (using the polling system), node battery status, node capabilities, etc.

This process starts, as described in Figure 5, with a DKS_ASSIGN message using double ciphering. First, all the cryptographic information needed by new DKS to start working is added: the new Domain Key, a public/private key pair for this new DKS, and a random number (x). This data is ciphered using the SM secret key. Then, another ciphering is done with the new DKS symmetric key. This double ciphering guarantees two aspects: the only node able to decipher the message will be the new DKS (since it is ciphered with its symmetric key) and this new DKS will be able to confirm that the message was generated by the trusted SM, using the SM public key to decipher the message previously signed by the SM.

SM needs confirmation from the new DKS, so a reply message is sent. Again, a double-ciphering is used. First, the random number (x) sent by the SM is ciphered with the DKS secret key, and then the message is ciphered with the SM public key, guaranteeing that just the SM will be able to decipher the message, and will confirm that it was generated by the assigned DKS and not by a spurious one.

Once the new DKS is assigned and confirmed, the new Domain Key will be sent to all the nodes but the compromised DKS, as described in previous diagram (Domain Key renewal).

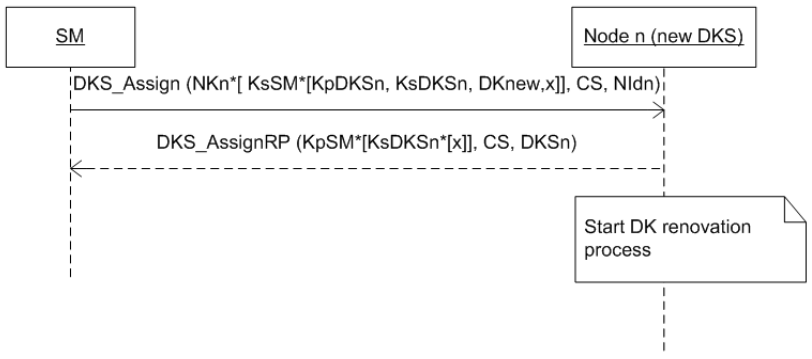

SM
Node n (new DKS)
DKS_Assign (NKn*[KsSM*[KpDKSn, KsDKSn, DKnew,x]], CS, NIdn)
DKS_AssignRP (KpSM*[KsDKSn*[x]], CS, DKSn)
Start DK renovation process

Figure 41 Sequence diagram describing how the Security Manager designates a new Domain Key Server

3.5.3.3. Critical nodes federation schema

There are some special nodes in WSNs that are of critical importance for the satisfactory performance of the network. Depending on the network topology, these special nodes can be named as cluster heads, relays, brokers, etc. An efficient solution to improve the resilience of the network is implementing a mechanism to replace one of these nodes if they get compromised or stop working properly (due to low battery, falling out of coverage, etc.).

A good example of this solution is broker federation. In a network with some nodes acting as brokers to deploy and publish the services, should the broker be compromised, the applications are not able to access sensor data anymore. The solution to this issue is re-assigning the role of “broker” to another node. This re-assignment is based on node reputation (using the polling system), node battery status or node capabilities. From this moment on, the new broker will publish all the services to the applications and the network will work properly again. This proposal includes this possibility, since the Domain Key Server can be considered as the broker or the cluster head, and DKS re-assignment mechanism has been developed in the proposal. When a new node starts acting as a DKS/cluster head/broker, the former one can be suspended if it is considered as a spurious node. The complete process is presented in next section.

3.6. Section summary

In this section, the contributions made by the author to the field of service management in wearable and ubiquitous devices are included.

The first contribution included in this section was the Wearable Device Service Bus. This proposal is based on an ESB acting as the integration point for different WSN platforms and middleware solutions. In this way, applications only need to access to a single end-point in order to obtain data provided by services available in different locations (ubiquitous) or worn by a specific user (wearable). The results of this contribution have been published in [126] and [127].

The second contribution is the Wearable InterDomain Protocols, a set of protocols to easy the development of application for ubiquitous WSN and IoT platforms selected to provide an easy way to interconnect applications with information providers and sensors with ESB. The proposal is to integrate selected protocols from the already existing collection, as well as to provide an open the solution in order to include the forthcoming protocols.

The third main contribution is the security proposal to be implemented in a WSN deployment. This security proposal is generic and scenario or application independent, and it is presented as a two-tier schema. In this schema, the high-level interface is the upper tier, where the access from the applications must be secured (checking whether the application has permission to access to the required services as well as ciphering the connection between the end device and the server). The proposal is based on HTTPS connections and PKI solution. The lower tier is the network providing the data (in this proposal, the WSN). For this lower tier, both channel (the wireless transmissions) and node identity must be secured. A trust domains solution is presented as the best option for this tier. The results of this contribution have been published in [128].

4 Validation

4.1. Validation

There are many potential available scenarios to be selected so as to test and validate this proposal. Some of them focus on the idea of improving the quality of life of people with limited mobility, like the elderly and the physically challenged, as a mean to provide particular answers to their requirements. Some other applications stress the concept of taking care of inanimate entities, like smart grid management or warehouse monitoring services, by integrating a plethora of technologies.

In order to test the functionality of the proposal, a scenario defined inside the research project "LifeWear-Mobilized Lifestyle with Wearables"[125] was used. The system was deployed in a real sportsman/woman scenario: the Technical University of Madrid (U.P.M.) student´s gymnasium.

Regarding the security proposal, a specific analysis was carried out to validate that the proposal provides a high enough security level to be used in real applications.

The results are presented in the following sections.

4.1.1. Sportsman scenario: LifeWear project validation

As introduced before, the validation of the proposal was accomplished in a real environment. The WSN nodes were spread throughout the student´s gymnasium inside U.P.M. facilities. Some of the users were selected to carry a set of wearable devices wirelessly connected to the WSN, as collected in the requirements of the research project "LifeWear-Mobilized Lifestyle with Wearables"[125].

The user´s body parameters (heart and breath rates, body temperature, etc.) were measured by a wearable sensor and sent to a special WSN node via Bluetooth. A general overview of the LifeWear testing scenario is presented in Figure 42.

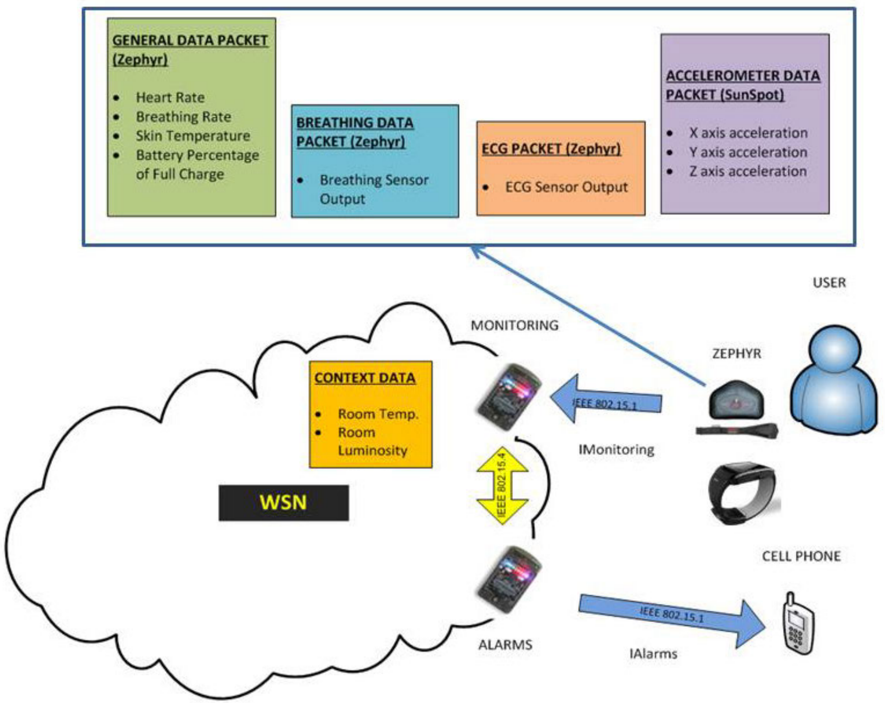


Figure 42 LifeWear requirements and testing scenario components (down) and service provided by the Zephyr body sensor (up)

Commonly, the roles that will be played by the different subsystems will be equivalent in the two conceived environments. The non-wearable, not encased within the Wireless Sensor Network components (i.e., the computer with the ESB installed and any RESTful-enabled device where requests are made from) will be used to send the service inquiries or to retrieve on-line or off-line data from the perspective of a person that, rather than taking an active part in the actions carried out, is monitoring the overall conditions of the system and the subject wearing the Wearable Wireless Body Area Network and using the services at the application level. The Wireless Sensor Network is acting as a distributed gateway between the operator and the sportsman/woman that will manage the inquiries and their answers from the human-centric subsystems. Finally, the person wearing the WWBAN, unlike the person managing the RESTful device, will perform other actions not related with the application and will carry with them the wearable devices as a support system that will monitor their physical conditions, so that if there is any unsettling indicator that can be measured by the WWBAN it will be displayed at both ends of the system (the Operator subsystem and the WWBAN subsystem).

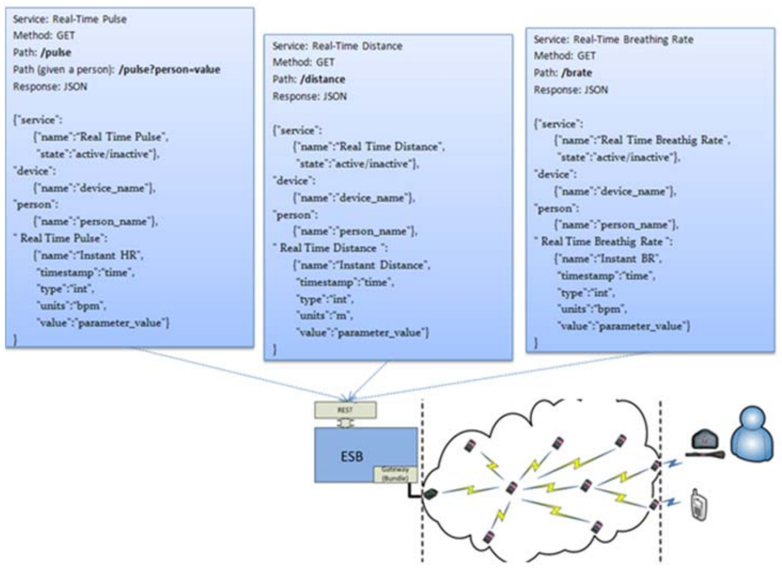
The architecture was deployed over a commercial WSN node solution: SunSPOT platform, manufactured by Oracle. Main characteristics of SunSPOT hardware platform are:

- ARM 920T CPU (180MHz - 32-bit)
- 512Kb RAM
- 4Mb FLASH
- 3.6V rechargeable 750 mAh Li-Ion battery.

In terms of security and privacy, the platform used to implement the proposed system includes mechanisms to cipher all the information inside the WSN. The Software Development Kit (SDK) includes libraries providing booth symmetric and asymmetric cryptography mechanisms. For symmetric cryptography, the ciphering algorithm used is AES (Advanced Encryption Standard), and RSA is used for asymmetric cryptography. Moreover, it is also possible to use Elliptic Curve Cryptography (ECC) algorithms in the last update of the platform SDK. A secure communication can be established using secure radio streams, which reuse Secure Sockets Layer (SSL) protocol underneath with ECC.

The ESB implementation used was FuseESB, an open-source implementation based on Apache ServiceMix that supports JBI and OSGi.

The smartwatch used was WIMM, a commercial watch developed by a manufacturer called WIMM Labs. It is an Android based watch, so it was necessary to develop and Android applications for the LifeWear project. This application is able to show alarms to the user in real time if any hazardous value is reached. As an open platform, Android gives the opportunity to any developer to create a new application for the WSN, using the interfaces provided to access the services. A smartphone application was also developed within the project and it provides the user full access to all the system services: exercise routines, alarms, profiling, historical data record, etc.



```
Service: Real-Time Pulse
Method: GET
Path: /pulse
Path (given a person): /pulse?person=value
Response: JSON

{"service":
    {"name":"Real Time Pulse",
     "state":"active/inactive"},
"device":
    {"name":"device_name"},
"person":
    {"name":"person_name"},
" Real Time Pulse":
    {"name":"Instant HR",
     "timestamp":"time",
     "type":"int",
     "units":"bpm",
     "value":"parameter_value"}
}
```

```
Service: Real-Time Distance
Method: GET
Path: /distance
Response: JSON

{"service":
    {"name":"Real Time Distance",
     "state":"active/inactive"},
"device":
    {"name":"device_name"},
"person":
    {"name":"person_name"},
" Real Time Distance ":
    {"name":"Instant Distance",
     "timestamp":"time",
     "type":"int",
     "units":"m",
     "value":"parameter_value"}
}
```

```
Service: Real-Time Breathing Rate
Method: GET
Path: /brate
Response: JSON

{"service":
    {"name":"Real Time Breathig Rate",
     "state":"active/inactive"},
"device":
    {"name":"device_name"},
"person":
    {"name":"person_name"},
" Real Time Breathig Rate ":
    {"name":"Instant BR",
     "timestamp":"time",
     "type":"int",
     "units":"bpm",
     "value":"parameter_value"}
}
```

Figure 43 Detailed view of Sports scenario services semantically annotated

In order to obtain user data (pulse rate, breathing rate, temperature, position, etc.) a wearable device is the most suitable solution. For evaluation purposes, a commercial solution was integrated in the platform: the Zephyr BioHarness Bluetooth device. BioHarness™ BT enables the capture and transmission of comprehensive physiological data via Bluetooth, providing remote monitoring of human performance and condition in the real world. This sensor is capable of monitoring heart rate, breathing rate, body temperature, body posture, and activity levels. The sensor sends data every second by default, but it can be configured if the application needs another data sampling rate.

Table 15 REST API URI examples

Service	Method	path	Parameters	Response	URI
Real-Time Pulse	GET	/pulse	userID(string)	JSON/XML	https://test_server_lw /cxf/ lw/pulse/userID
Real-Time Distance	GET	/distance	userID(string)	JSON/XML	https://test_server_lw /cxf/ lw/distance/userID
Breathing Rate	GET	/brate	userID(string)	JSON/XML	https://test_server_lw /cxf/ lw/brate/userID
ECG	GET	/ECG	userID(string)	JSON/XML	https://test_server_lw /cxf/ lw/ECG/userID
Body Temperature	GET	/btemp	userID(string)	JSON/XML	https://test_server_lw /cxf/ lw/btemp/userID
GPS tracking	GET	/GPS	userID(string)	JSON/XML	https://test_server_lw/cxf/ lw/GPS/userID
Injury Prevention	GET	/injury	userID(string)	JSON/XML	https://test_server_lw/cxf/ lw/injury/userID
Accelerometer	GET	/accel	userID(string), axis	JSON/XML	https://test_server_lw/cxf/ lw/accel/userID?axis=X
User position	GET	/position	userID(string)	JSON/XML	https://test_server_lw/cxf/ lw/position/userID

4.1.2. Third-party application development

As it can be expected, the deployed WSN can be used by developers to create new applications that will exploit the full potential of a complete deployment of the system wherever it runs. One remarkable advantage of using REST interfaces as the way to access high level middleware services is that they effectively insulate the developed application of the services and components located in lower layers, such as the middleware itself, the communication services and the devices used to collect the data that will be used for the developed applications.

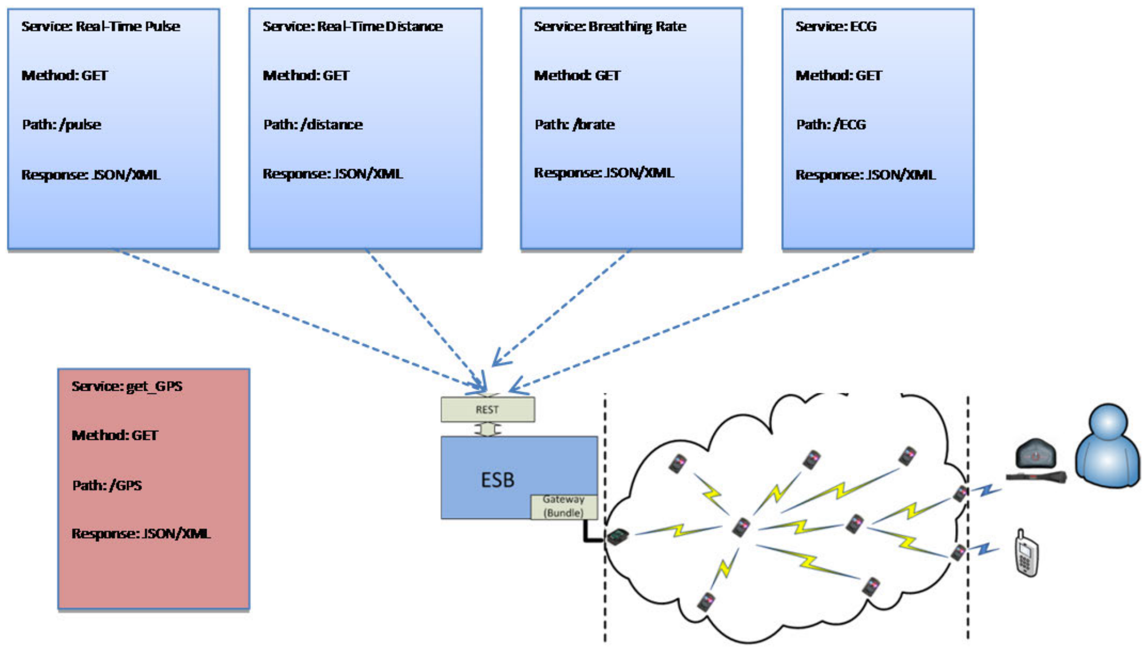


Figure 44 External REST interfaces available for third-party application development

Here, it can be acknowledged that application developers are free to implement frontends as complex as they wish to do (as the complexity and appearance of the application is decoupled from the network architecture), provided that they are using a URI that matches the ones provided by the high level services so as to connect to the system. Also, REST interfaces are operating system agnostic, so devices running can have applications developed for devices running Windows, Linux, Android or iOS.

Typically, a developer will follow the next steps to codify applications for the system:

1. Applications will be based on the collection of functionalities that services are able to provide: the developer just needs to choose what group of functionalities the application will be based on.

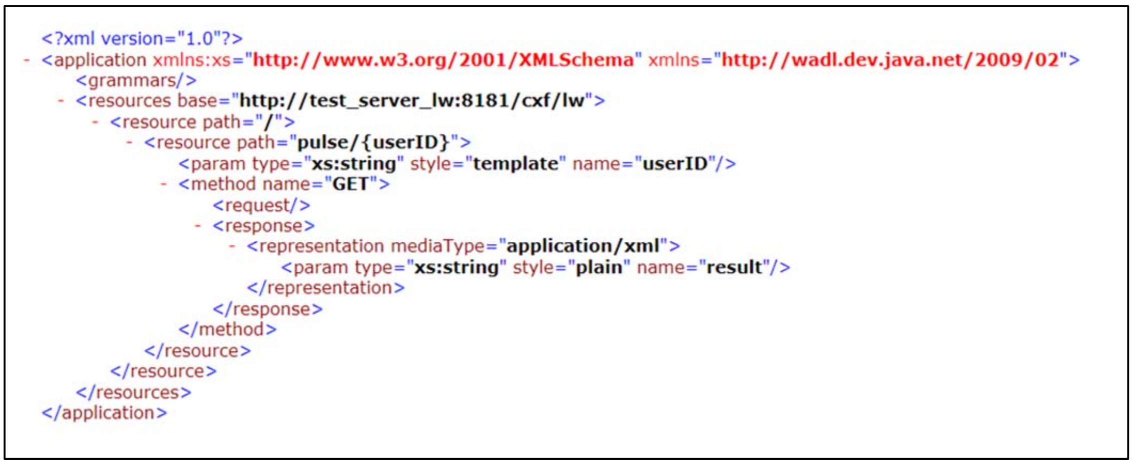
2. Once the functionality is chosen, URIs used at the specific REST interfaces must be known. This and step 3 are key points to consider, as the REST interfaces that are going to be used as access point to the architecture become known through these steps. An example of the description for a service using WADL language is provided in Figure 45.

3. The developer obtains information about the available services of a deployment. This information can be provided by executing the functionalities of the components of the architecture that are accessed via REST, which are capable of offering the needed data. It is important to note that service refers to the software capabilities of the network and the different devices that are running as part of the system.

4. When the functionality/functionalities that are going to be used, the service(s) that are able to offer it are known and the URI(s) provided by the REST interfaces have been accurately recognized, then the access point of the application has become clear. The application can be fully developed at this point; as far as the architecture is concerned, codification must only take into account the URI(s) that is/are used to access the system. The appearance of the Graphical User Interface (GUI) can have any desired features (button dimensions, text field layouts, etc.).

5. The application is tested with the actual devices that are utilized. Any minor changes are made to accommodate the expected performance of the application.

Parallel work on the application that is developed is also possible: since the Graphical User Interface characteristics of the device frontend are not dependent on the URIs offered by the WSN architecture, they can be worked on separately.

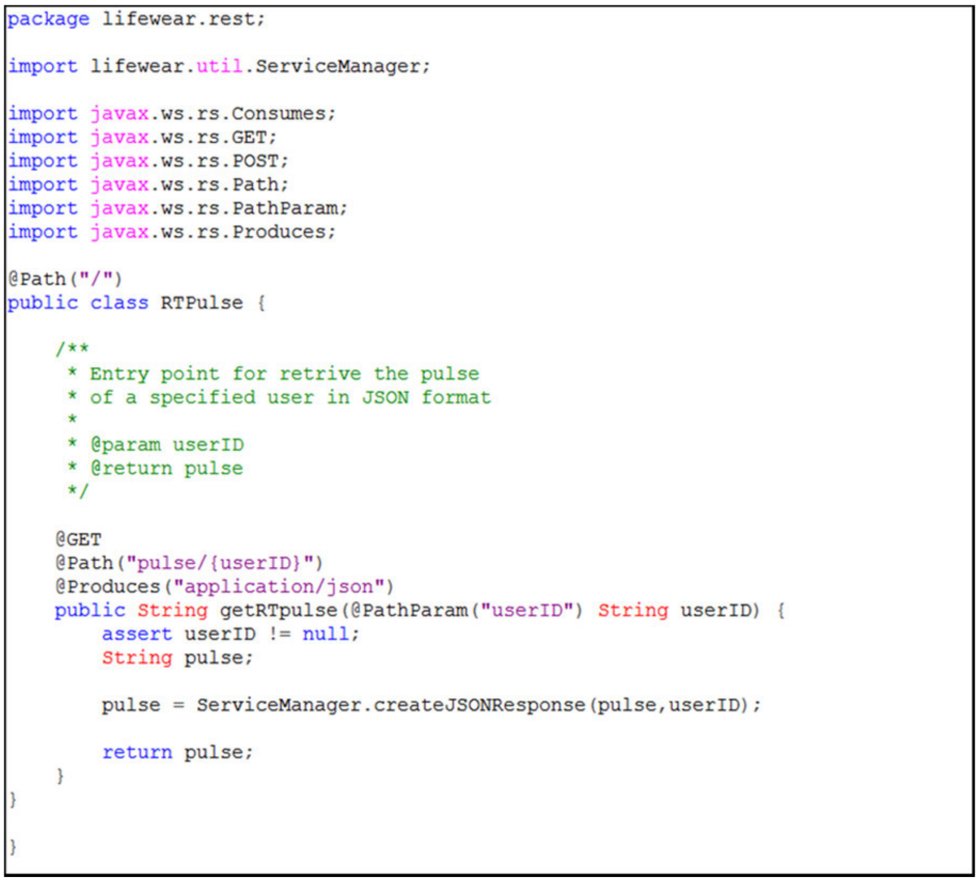


```
  <?xml version="1.0"?>
- <application xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns="http://wadl.dev.java.net/2009/02">
     <grammars/>
   - <resources base="http://test_server_lw:8181/cxf/lw">
       - <resource path="/">
           - <resource path="pulse/{userID}">
                 <param type="xs:string" style="template" name="userID"/>
               - <method name="GET">
                     <request/>
                   - <response>
                       - <representation mediaType="application/xml">
                             <param type="xs:string" style="plain" name="result"/>
                         </representation>
                     </response>
                 </method>
             </resource>
         </resource>
     </resources>
  </application>
```

Figure 45 WADL describing the "get Real Time pulse" service

As a solution to define REST interfaces in Java language, Java API for RESTful Web Services (JAX-RS) was selected.

An example depicting a service description (get heart rate from a specified user) is shown in Figure 46. As it can observed, this API provides annotations (e.g., *@GET*, *@Path*) easing the allocation of classes as resources.



```
package lifewear.rest;

import lifewear.util.ServiceManager;

import javax.ws.rs.Consumes;
import javax.ws.rs.GET;
import javax.ws.rs.POST;
import javax.ws.rs.Path;
import javax.ws.rs.PathParam;
import javax.ws.rs.Produces;

@Path("/")
public class RTPulse {

    /**
     * Entry point for retrive the pulse
     * of a specified user in JSON format
     *
     * @param userID
     * @return pulse
     */

    @GET
    @Path("pulse/{userID}")
    @Produces("application/json")
    public String getRTpulse(@PathParam("userID") String userID) {
        assert userID != null;
        String pulse;

        pulse = ServiceManager.createJSONResponse(pulse,userID);

        return pulse;
    }
}

}
```

Figure 46 REST interface definition using JAVA-RS

A mobile application following these directions was developed by SAI Wireless, a Spanish SME with interests in the mobile market, e-Health devices and Web applications. To use this application, the sportsman/woman will create their own profile, and once they have done so a set of sports (running, cycling or free practice) will be offered to establish training routines.

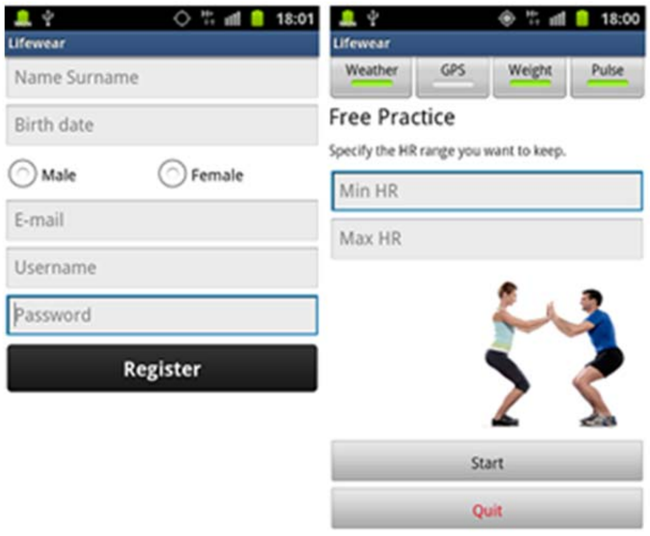


Figure 47 Third-party mobile application. Graphic User Interface screenshots.

Figure 47 shows a couple of screenshots depicting the features provided by the third-party application. As it was mentioned, this GUI is using the REST interface provided by the ESB and is retrieving the data provided by the WSN and wearable devices through the middleware.

The proposed architecture, as well as the use of REST interfaces, speeded up the development process by decoupling the development of the GUI from the middleware services development. In the first stage of the development task, a prototype set of services were deployed in the ESB in order to test the functionalities of the GUI before the installation of the middleware final version.

4.2. Test results

With all the elements connected and the network deployed, several time measurements were obtained: start-up time of the network elements, service response time and nodes lifetime. Also, the amount of memory used in the nodes to run the applications was measured.

The full system start-up time was tested in order to determine the time that the user needs to wait before starting the usage of the platform. The results for the most relevant elements are shown in Figure 48. The ESB start-up time is long (13000 milliseconds), since it is the time period for an ESB Linux-based machine to boot-up. The Zephyr device boot-up time is longer than a regular node because it needs to set-up the Bluetooth link with the WSN node.

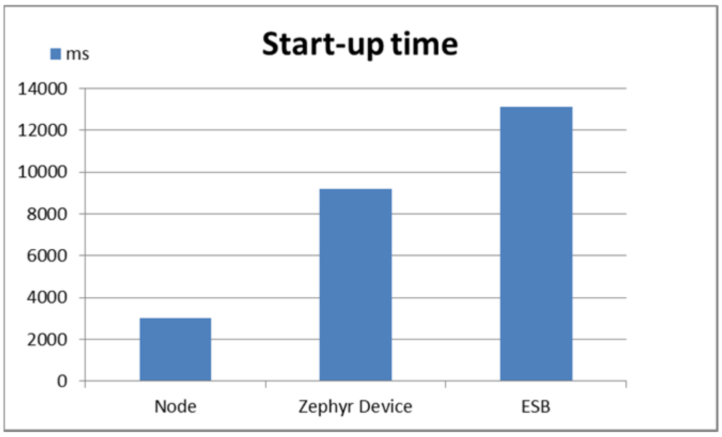


Figure 48 Network elements start-up time

The reading data delay introduced by the ESB is also high (4 seconds), and it must be improved in order to avoid bottlenecks in the network. The physiological data is sampled by the Bluetooth device every second and remains stored until the ESB performs a data query or an alarm is issued. Although the use of an ESB in the network introduces a delay, the integration and scalability facilities offered by the ESB justify the use of this element. Moreover, in the testing scenario the alarms can reach the smartwatch in a direct Bluetooth connection, so real-time alarms delivery is guaranteed.

The response time for simple and composed service queries is shown in Figure 49. It was expectable that the injury prevention composed service request takes longer, since it has to be received by the broker and then wait for the composing request to be processed.

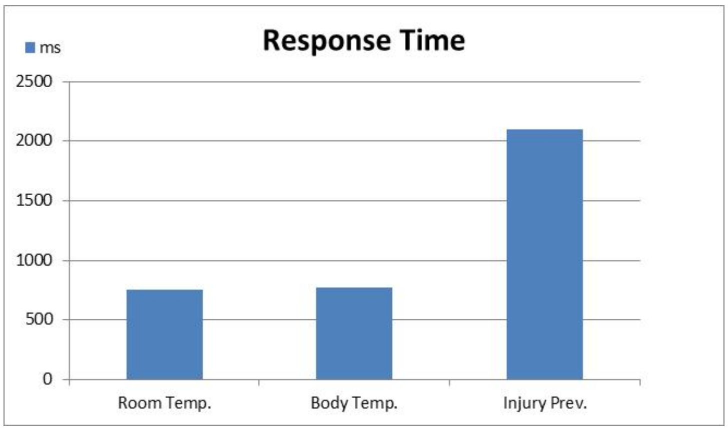


Figure 49 Response time for different service queries: simple services (temperature1, 2 and 3 and body temperature) and composed service (injury prevention)

Node lifetime is an important factor in the application. Depending on the role assigned, the same node battery would last for a different time. For example, the most active nodes are the one maintaining the Bluetooth connection. Thus, these nodes need to be charged every 6 hours of continuous activity. Other nodes lifetime lengths are shown in Figure 50.

In a sportsman scenario, it is admissible that the user devices can be recharged when the activity has ended, and the WSN nodes can be recharged every day when the activity centre closes (for example, every night in a gymnasium environment). If the application requires a longer node lifetime, the duty cycle and the data sampling frequency can be modified in order to reduce energy consumption.

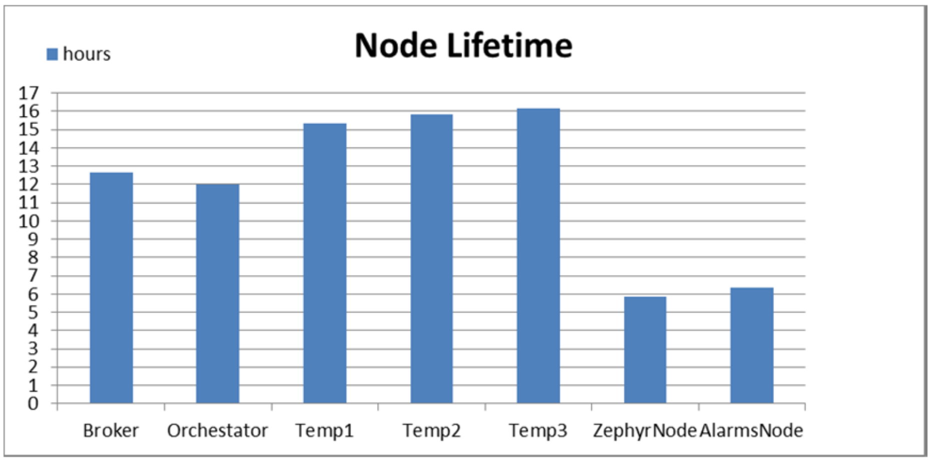


Figure 50 Consumption of the different nodes, measured in lifetime hours

Also the amount of memory used by the execution of the source code was measured inside the nodes. The most important data are shown in Table 16. The amount of memory used is low, and there is enough free space to store application data or to scale up the system.

Table 16 Memory usage in the SunSPOT hardware platform

	Storage (KB)	RAM (KB)
Free	3200	512
Used	83	4
% used	2,59%	0,78%

4.3. Security analysis

In the dissertation section, security proposal elements were presented. As a way to describe in depth the role performed by each of them, the proposal schema has been illustrated following the e-Health application example used in the validation of the proposal. This scenario consists of a medical environment with sensitive data interchange (medical records, biometric parameters of the patients, medical stuff accessing privileges, etc.). This application example deployment is shown in Figure 51.

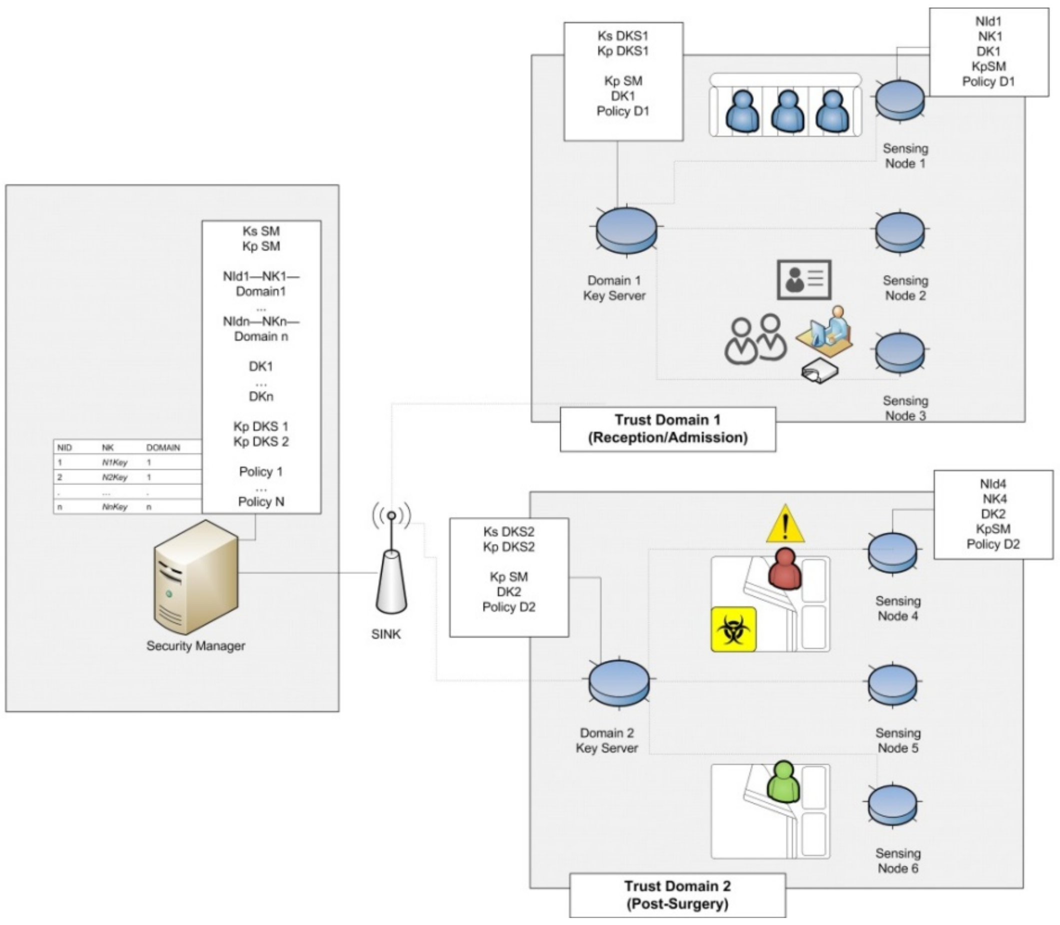


Figure 51 Proposed model applied in medical environments for an e-Health application.

In this basic example, only two trust domains have been defined (as there are two levels of security): a trust domain for the admission area, and another trust domain for a post-surgery room. At the admission area, the information circulating is related to each patient medical records and extra information (contact information, personal id, etc.). In this trust domain, the security policy must guarantee, at least, information privacy.

At the post-surgery area the information in the network is composed of medical records and real-time biometric parameters (body temperature, blood pressure, ECG, breathing rate, etc.). In this Trust Domain, not only privacy but also data integrity must be assured. Furthermore, routing and data delivery are important when an alarm is issued (for example, with a low breathing rate). In the next paragraphs, each proposal element is identified and described.

The Security Manager is an external entity (but connected through the sink to the WSN), where high level security mechanisms have been deployed. The tasks assigned to the Security Manager include Node Key generation and distribution, Security Policies definition and distribution, Domain Key Server role assignment, re-keying processes and data recording (with regards to the nodes participating in each Trust Domain).

The presented proposal is able to provide solutions to cover a wide range of security requirements. They fulfil the majority of the security requirements for WSNs-based applications, from the less critical ones (livestock control, traffic density measurement or environmental monitoring) to the most security-sensitive requirements needed by e-Health applications, which demand data integrity, privacy, authentication, etc. For each security requirement identified, a solution has been included the proposal, as depicted in Table 17.

Table 17 Security requirements and solutions

Requirement	Solution
Authentication	Node ID, PKI
Availability	Node revocation
Privacy	Keys, PKI
Data Integrity	Hash
Prot. Outsider attacks	Node ID, PKI
Prot. Insider attacks	Security policies, trust domains

Furthermore, the security proposal has been evaluated against a well-known group of security attacks selected from the literature. For each attack, the countermeasure implemented in the proposal is described as follows:

- **Denial of Service (DoS)**. This is the least sophisticated attack. It appears when either the physical layer is degraded to a level where the communication between nodes is impossible (jamming) or when a spurious node starts sending malicious data packets to the network. In both situations, an alarm is triggered in the Security Manager to alert the IT-security staff.

- **Sybil attack** occurs when a node is asking for multiple IDs. If the attack succeeds, the node is able to subvert the trust mechanism. In this proposal, every node ID is pre-configured for each node and only the Security Manager (out of the WSN) has the complete list of the IDs. Furthermore, it is possible to perform a node revocation (as explained in the previous section).

- **Message corruption**. In this attack, a message reaches the recipient with a different content than the one sent by the source. This situation is either because the message has been degraded in the transmission, or because the message has been intercepted and intentionally changed. To avoid both issues, the proposal includes the Ciphering Suite functionality, which allows performing a message hash (using MD5, SHA1, etc.).

- **Eavesdropping**. WSNs use broadcast transmissions, so other devices listening in the same frequency may intercept every communication between two nodes. To avoid data disclosure, the presented proposal provides both symmetric and PKI ciphering capabilities.

- **Node subversion**. If one of the nodes is captured and crypto-analysed the secret keys, node ID, security policies, etc. become disclosed. This proposal aim is to minimize the cryptographic and security information stored in each node. Nevertheless, if a node is captured, all the keys in the network can be renewed as explained in previous section.

- **Node replication** occurs when a node ID is copied, replicated in a new node and then introduced in the network. From that moment on, the network accepts the node with the cloned ID as an authorized node. The proposal provides two mechanisms to avoid this attack. The first one is the Node ID, which is stored in an external entity (the Security Manager) that controls all the IDs working in the network. The second mechanism is security policy. If the Security Manager detects that 2 nodes are operating with the same ID, a node revocation protocol is issued, and the node is dropped from the network.

- **False node**. This attack introduces data traffic in the network to avoid legitimate nodes to communicate (injecting false data messages, claiming for authorization continuously, etc.). Using the node ID, this proposal is able to identify the false node and, using the Domain Key renewal functionality, all the messages sent by this node will be discarded.

Attacks and countermeasures have been summarized in Table 18.

Table 18 Attacks and countermeasures implemented in the proposal

Attack	Countermeasure implemented
DoS	SM alarm
Sybil attack	Node ID, node revocation
Message corruption	Hash
Eavesdropping	Keys (symmetric and PKI)
Node subversion	Few crypto. data stored
Node replication	Node ID, security policies
False node	Node ID, Domain Keys

4.4. Section summary

In this section, the validation results of the proposal have been presented. In order to test the functionality of the proposal, a scenario defined as part of the research project “LifeWear”[125] was used. The system was deployed in a real sportsman scenario: the Technical University of Madrid (U.P.M.) student´s gymnasium. In this validation scenario, the user body parameters (heart and breath rate, body temperature, etc.) were measured by a wearable sensor and sent to a special WSN node via Bluetooth, and the parameters were displayed both in a smartwatch and in a smartphone application. The smartwatch used was WIMM, a commercial watch developed by WIMM Labs and the heart rate monitor was the Zephyr BioHarness Bluetooth device. The ESB implementation used was FuseESB, an open-source implementation based on Apache ServiceMix and the WSN platform deployed throughout the scenario was based in the Oracle SunSPOT solution.

With all the elements connected and the network deployed, several time measurements were obtained: start-up time of the network elements, service response time and nodes lifetime. Also, the amount of memory used in the nodes to run the applications was measured. The ESB start-up time is long (13000 milliseconds), since it is the time period for an ESB Linux-based machine to boot-up, but this process is only performed at the first use of the system. The Zephyr device boot-up time is longer than a regular node because it needs to set-up the Bluetooth link with the WSN node.

Regarding the security proposal, a specific analysis was carried out in order to validate that the proposal provides the enough security level to be used in real applications. Furthermore, the countermeasures implemented were evaluated.

The complete scenario was validated within the LifeWear project final review with successful results. Furthermore, the results of the validation have been published in [126-129].

5 Conclusions

5.1. Conclusions

The IoT paradigm can be built using Wireless Sensor Networks (WSN) as the leading technology to acquire and manage data. Connecting other smart elements to a WSN (phones, watches, tablets, etc.) may also improve the user experience in the IoT, and it could act as a starting point for the use of the technology. If the smart devices are wearable, the first technology-access barrier is broken: the user just has to "wear" the technology as a daily-life garment. The utilization of wearable sensors to obtain useful human body parameters is nowadays a reality, partially due to the deployment of wireless sensor networks as the back- bone of the new emerging applications. But one of the key issues arising in this field will be the heterogeneity. In fact, there will be heterogeneity at the hardware layer (WSN nodes, wearable devices, protocols, etc.) as well as in the application layer (devices, operating systems, protocols, etc.).

With an increasing number of wearable devices integrated in WSNs using wireless communications, security challenges are greater than in wired networks because of the open access to data transmissions. Communications between two wireless nodes can be intercepted by an spurious element obtaining access to the information broadcasted. Additionally, if a node is captured and crypto-analyzed (thus, security keys, policies, and other cryptographic elements are accessed), a new spurious node could be introduced in the network, and several attacks can be deployed: denial-of-service, man-in-the-middle, data sniffing and/or data modification, routes spoofing, and so on.

As defined in section 1, the main objective defined for this thesis was to define and implement a solution for the intelligent service management in wearable and ubiquitous devices in order to solve the heterogeneity issues. This main objective can be considered as fully achieved taking into account the results collected after conducting the validation phase.

The main issue faced in this thesis was to solve the heterogeneity issues arising when developing applications for WSN and wearable devices. As it has been stated, the solution proposed has been successfully validated in a real scenario and the results obtained were satisfactory, as reflected in the number of publications generated within the work in this thesis. Indeed, several papers including the results have been published in indexed journals (JCR).

Regarding the primary and secondary objectives defined in the requirements, a summary of the consecution is presented below:

- A comprehensive **State-of-the-Art** analysis
    - Section 2 includes a complete State-of-the-Art related with the technologies involved and used as part of the implementation works in this thesis, which addresses the most prominent proposals in this area of knowledge.

- Design a Wearable Device Service Bus (**WDSB**)
    - As stated in section 3, the WDSB has been fully specified and deployed at the validation stage. This bus includes the technologies collected in the requirements (ESB, middleware, WWBAN, WSN and IoT). Applications are able to access the WSN services regardless of the platform and operating system where they are running.

- Design a Wearable Inter-domain communication protocols set (**WIDP**)
    - Section 3 also includes the selection of the WIDP which integrates lightweight protocols suitable to be used in low-capacities devices (REST, JSON, AMQP, CoAP, etc...).

    - The extension of an existing service ontology based on existing services ontologies, in order to provide definitions for wearable devices and service composition

- Propose a **security solution** for the service management
    - A trustworthy domains model to deploy security services in WSNs has been designed. Furthermore, a complete trusting scheme to accept, control and exclude nodes participating in a trusted domain has also been proposed, defining security policies by means of it, and using symmetric and PKI cryptography and special elements to achieve that goal (security manager and domain key servers). Also, resilience issues have been addressed, proposing mechanisms to reconfigure a compromised trusted domain with rekeying or servers reassignment. This proposal also includes a schema to provide role federation for critical nodes that have been assigned functionalities that are mandatory for the correct performance of the system.

- **Validation** tests
    - Scenario selection: real scenario defined inside the research project “*LifeWear-Mobilized Lifestyle with Wearables*” was used.
    - Validation of the critical contributions of this thesis: WDSB, WIDP, security schema and application with acceptable results.
    - Time and code size measurements
    - Specific security analysis performed so as to validate that the proposal provides the enough security level to be used in real applications as well as the evaluation of the countermeasures implemented.

5.2. Future works

Both Internet of Things and wearable devices fields are experimenting a fast and remarkable growing. In this sense, new platforms, software and devices are expected to appear in a near future. As stated in previous section, our proposal eases the development of new applications for these devices, but there is still a lot of work to do regarding the interaction between different platforms at hardware and communication layers.

Although REST is a lightweight protocol with low header overload, some novel and lighter protocols are appearing. An interesting research line is the evaluation of these protocols as well as the integration within this proposal.

Furthermore, there is a wide range of application scenarios to explore in order to evaluate our proposal in different fields. As remarked in the evaluation section of this thesis, although our proposal was initially evaluated in an e-Health scenario, it was assessed in a completely different scenario afterwards (smart grid and renewable energies ecosystem), with promising results. Thus, it would be interesting to evaluate our proposal in several fields, such as: defence, security, hazardous environments, etc.

Regarding the security proposal, it can be improved with the integration of new security protocols and mechanisms. In our proposal, RSA and AES mechanisms have been integrated. But there are new algorithms (like ECC) that can be easily integrated in our platform in order to evaluate and compare them with the integrated mechanisms.

Finally, although ESB is a community and commercial supported platform, it would be interesting to evaluate the possibility of migrating the developed software modules (or any others that were to be developed in the future) to other distributed integration environment related to IoT or WSN platforms in order to test and compare the results with the ones presented in this work. Moreover, there is an emerging paradigm named iPaaS (integration Platform as a Service) providing cloud integration platforms that would be interesting to evaluate.

5.3. Publications and projects

5.3.1. SCI-indexed journals

- **Castillejo, P**.; Martinez, J.-F.; Rodriguez-Molina, J.; Cuerva, A., "Integration of wearable devices in a wireless sensor network for an E-health application," Wireless Communications, IEEE, vol.20, no.4, pp.38,49, August 2013. Impact factor: 6.524 (Q1) (T1)

- **Pedro Castillejo**, José-Fernán Martínez, Lourdes López, and Gregorio Rubio, “An Internet of Things Approach for Managing Smart Services Provided by Wearable Devices,” International Journal of Distributed Sensor Networks, vol. 2013, Article ID 190813, 9 pages, 2013. doi:10.1155/2013/190813. Impact factor: 0.923 (Q3) (T2)

- **Pedro Castillejo**, José-Fernán Martínez-Ortega, Lourdes López, and José Antonio Sánchez Alcón, “SensoTrust: Trustworthy Domains in Wireless Sensor Networks,” International Journal of Distributed Sensor Networks, Article ID 484820, in press. Impact factor): 0.923 (Q3) (T2)

- Rodríguez-Molina, J.; Martínez, J.-F.; **Castillejo, P**.; López, L. Combining Wireless Sensor Networks and Semantic Middleware for an Internet of Things-Based Sportsman/Woman Monitoring Application. Sensors 2013, 13, 1787-1835. Impact factor: 2.048 (Q1) (T1)

- Martínez, J.-F.; Rodríguez-Molina, J.; **Castillejo, P**.; de Diego, R. Middleware Architectures for the Smart Grid: Survey and Challenges in the Foreseeable Future. Energies. 2013, 6, 3593-3621. Impact factor: 1.602 (Q3) (T2)

- Jesús Rodríguez-Molina, José-Fernán Martínez, **Pedro Castillejo**, and Rubén de Diego, “SMArc: A Proposal for a Smart, Semantic Middleware Architecture Focused on Smart City Energy Management,” International Journal of Distributed Sensor Networks, vol. 2013, Article ID 560418, 17 pages, 2013. doi:10.1155/2013/560418. Impact factor: 0.923 (Q3) (T2)

5.3.2. International conferences

- Sanchez Alcon, J. A., Lopez, L., Martinez, J. F., Castillejo, P. (2013, August). Automated determination of security services to ensure personal data protection in the Internet of Things applications. In Innovative Computing Technology (INTECH), 2013 Third International Conference on (pp. 71-76). IEEE.

- Gregorio Rubio, Estefanía Serral, Pedro Castillejo, José Fernán Martínez. A Novel Context Ontology to Facilitate Interoperation of Semantic Services in Environments with Wearable Devices. On the Move to Meaningful Internet Systems: OTM 2012 Workshops Lecture Notes in Computer Science Volume 7567, 2012, pp 495-504

- Publicación y ponencia del artículo "Wireless sensor networks in knowledge management" en el congreso ICCS 2010 en Amsterdam (Holanda), calificada con "A" en el ranking ERA.

5.3.3. Research projects

- **European Projects**
  - **I3RES** - ICT based Intel<ligent management of Integrated RES for the Smart Grid optimal operation (FP7).
  - **e-GOTHAM** - Sustainable - Smart Grid Open System for The Aggregated Control, Monitoring and Management of Energy. (FP7).
  - **DEMANES** - Design, Monitoring and Operation of Adaptive Networked Embedded Systems (FP7).
  - **LifeWear** - Mobilized Lifestyle With Wearables (ITEA2).

- **National Funding**
  - **AWARE** - Accessible Wearable Device Platform for Smart Environments. Funded by the Spanish Ministry of Economy and Competitiveness. (TEC2011-28397).

6 Bibliography

6.1. References

[1]. Emerging Technologies that will change the World. http://www2.technologyreview.com/featured-story/401775/10-emerging-technologies-that-will-change-the/.

[2]. IEEE Bluetooth 802.15.1 standard http://standards.ieee.org/getieee802/download/802.15.1-2005.pdf.

[3]. RFC, 6LoWPAN. http://datatracker.ietf.org/doc/rfc4919/.

[4]. IEEE 802.11 standard. http://standards.ieee.org/getieee802/802.11.html.

[5]. IEEE 802.16 standard. https://standards.ieee.org/about/get/802/802.16.html.

[6]. Wavenis product page. http://www.elstermetering.com/en/key-features.

[7]. IEEE 802.15.4 standard. http://www.ieee802.org/15/pub/TG4.html.

[8]. Thread group main page. http://threadgroup.org/.

[9]. W. Heinzelman, A. Chandrakasan, H. Balakrishnan, "Energy-efficient communication protocol for wireless sensor networks", in: Proceeding of the Hawaii International Conference System Sciences, Hawaii, January 2000

[10]. A. Manjeshwar and D. P. Agarwal, "TEEN: a routing protocol for enhanced efficiency in wireless sensor networks, " in 1st International Workshop on Parallel and Distributed Computing Issues in Wireless Networks and Mobile Computing, April 2001.

[11]. M. Younis et al., "Energy-aware routing in cluster-based sensor networks", in: Proceedings of the 10th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems (MASCOTS2002), Fort Worth, TX, October 2002.

[12]. K. Akkaya et al., "An energy-aware QoS routing protocol for wireless sensor networks", in: Proceedings of the IEEE Workshop on Mobile and Wireless Networks (MWN 2003), Providence, RI, May 2003

[13]. Q. Ren and Q. Liang, A contention-based energy-efficient MAC protocol for wireless sensor networks, in Proceedings of 2006 IEEE Wireless Communications and Networking Conference (WCNC 06), Las Vegas, NV, Apr. 2006, pp. 1154-1159

[14]. D. Aguayo, J. Bicket, R. Morris, SrcRR: A High-Throughput Routing Protocol for 802.11 Mesh Networks, 2003.

[15]. R. Leung, J. Liu, E. Poon, A.L.C. Chan, B. Li, MP-DSR: a QoSaware multi-path dynamic source routing protocol for wireless Ad-Hoc networks, in: Proceedings of the Local Computer Networks (LCN'01), 2001.

[16]. S. Liu,T. Fevens, A.E. Abdallah, Hybrid position-based routing algorithms for 3D mobile ad hoc networks, in: Proceedings of the Fourth International Conference on Mobile Ad-hoc and Sensor Networks, December 2008, pp. 177–186.

[17]. Imote2 datasheet. http://wsn.cse.wustl.edu/images/e/e3/Imote2_Datasheet.pdf

[18]. Tmote Sky datasheet. http://www.snm.ethz.ch/snmwiki/pub/uploads/Projects/tmote_sky_datasheet.pdf

[19]. MicaZ datasheet. http://edge.rit.edu/edge/P08208/public/Controls_Files/MICaZ-DataSheet.pdf

[20]. Mica2 datasheet. http://www.eol.ucar.edu/isf/facilities/isa/internal/CrossBow/DataSheets/mica2.pdf

[21]. Mica2DOT datasheet. http://www.eol.ucar.edu/isf/facilities/isa/internal/CrossBow/DataSheets/mica2dot.pdf

[22]. TinyNode 584 datasheet. http://www.tinynode.com/?q=system/files/TN584EK_Datasheet_v_1_1.pdf

[23]. Shimmer3 datasheet. http://www.shimmersensing.com/images/uploads/docs/Shimmer3_Spec_1.4.pdf

[24]. Zolertia Z1 datasheet. http://zolertia.com/sites/default/files/Z1SP_Datasheet.pdf

[25]. Waspmote product datasheet. http://www.libelium.com/downloads/documentation/waspmote_datasheet.pdf

[26]. Oracle SunSPOT technical specification (rev. 8) http://sunspotdev.org/docs/Yellow/eSPOT8ds.pdf

[27]. Intel Galileo datasheet. http://download.intel.com/support/galileo/sb/galileo_datasheet_329681_003.pdf

[28]. Intel Edison datasheet. http://download.intel.com/support/edison/sb/edisonmodule_hg_331189002.pdf

[29]. Raspberry Pi (model B) features. https://www.raspberrypi.org/products/model-b-plus/

[30]. Arduino UNO specification. http://www.arduino.cc/en/Main/ArduinoBoardUno

[31]. TinyOS: An operating system for sensor networks. LEVIS, Philip, et al. Berlin Springer Berlin Heidelberg, 2005, Ambient intelligence, Vols. 115-148.

[32]. Contiki-a lightweight and flexible operating system for tiny networked sensors. DUNKELS, Adam, GRONVALL, Bjorn and VOIGT, Thiemo. 2004, 29th Annual IEEE International Conference on Local Computer Networks, pp. 455-462.

[33]. The liteos operating system: Towards unix-like abstractions for wireless sensor networks. CAO, Qing, et al. 2008, International Conference on Information Processing in Sensor Networks, pp. 233-244.

[34]. A dynamic operating system for sensor nodes. HAN, Chih-Chieh et al. s.l. : ACM, 2005, Proceedings of the 3rd international conference on Mobile systems, applications, and services, pp. 163-176.

[35]. MANTIS OS: An embedded multithreaded operating system for wireless micro sensor platforms. BHATTI, Shah et al. 4, 2005, Mobile Networks and Applications, Vol. 10, pp. 563-579.

[36]. Ethernut project main page. [Online] http://www.ethernut.de/.

[37]. Nano-rk: an energy-aware resource-centric rtos for sensor networks. ESWARAN, Anand, ROWE, Anthony and RAJKUMAR, Raj. 2005, 26th IEEE International Real-Time Systems Symposium. RTSS 2005. .

[38]. RIOT OS: Towards an OS for the Internet of Things. Emmanuel Baccelli, Oliver Hahm, Mesut Gunes, Matthias W   ahlisch, Thomas Schmidt. Turin, Italy : s.n., 2013, The 32nd IEEE International Conference on Computer Communications (INFOCOM 2013).

[39]. Oracle SunSPOT platform main page. http://www.sunspotworld.com.

[40]. Azzara, A.; Bocchino, S.; Pagano, P.; Pellerano, G.; Petracca, M., "Middleware solutions in WSN: The IoT oriented approach in the ICSI project," Software, Telecommunications and Computer Networks (SoftCOM), 2013 21st International Conference on , vol., no., pp.1,6, 18-20 Sept. 2013 doi: 10.1109/SoftCOM.2013.6671886

[41]. Haghighi, M.; Cliff, D., "Sensomax: An agent-based middleware for decentralized dynamic data-gathering in wireless sensor networks," Collaboration Technologies and Systems (CTS), 2013 International Conference on , vol., no., pp.107,114, 20-24 May 2013. doi: 10.1109/CTS.2013.6567214

[42]. Brennan, R.; Wei Tai; O'sullivan, D.; Aslam, M.S.; Rea, S.; Pesch, D., "Open Framework Middleware for intelligent WSN topology adaption in smart buildings," Ultra Modern Telecommunications & Workshops, 2009. ICUMT '09. International Conference on , vol., no., pp.1,7, 12-14 Oct. 2009 doi: 10.1109/ICUMT.2009.5345504

[43]. Pensas, H.; Vanhala, J., "WSN Middleware for Existing Smart Homes," Sensor Technologies and Applications (SENSORCOMM), 2010 Fourth International Conference on , vol., no., pp.74,79, 18-25 July 2010 doi: 10.1109/SENSORCOMM.2010.18

[44]. Boulmalf, M.; Belgana, A.; Sadiki, T.; Hussein, S.; Aouam, T.; Harroud, H., "A lightweight middleware for an e-health WSN based system using Android technology," Multimedia Computing and Systems (ICMCS), 2012 International Conference on , vol., no., pp.551,556, 10-12 May 2012 doi: 10.1109/ICMCS.2012.6320312

[45]. Tharakan, L.A.; Dhanasekaran, R., "SEEMd -security enabled energy efficient middleware for WSN," Advanced Communication Control and Computing Technologies (ICACCCT), 2014 International Conference on , vol., no., pp.800,804, 8-10 May 2014. doi: 10.1109/ICACCCT.2014.7019201

[46]. Chapin, P.; Skalka, C., "SpartanRPC: Secure WSN middleware for cooperating domains," Mobile Adhoc and Sensor Systems (MASS), 2010 IEEE 7th International Conference on , vol., no., pp.61,70, 8-12 Nov. 2010 doi: 10.1109/MASS.2010.5663965

[47]. Pomante, L.; Pugliese, M.; Marchesani, S.; Santucci, F., "WINSOME: A middleware platform for the provision of secure monitoring services over Wireless Sensor Networks," Wireless Communications and Mobile Computing Conference (IWCMC), 2013 9th International , vol., no., pp.706,711, 1-5 July 2013. doi: 10.1109/IWCMC.2013.6583643

[48]. Adrian Perrig; Robert Szewczyk; J. D. Tygar; Victor Wen; David E. Culler. SPINS: security protocols for sensor networks. Wirel. Netw. 8, 5 (September 2002), 521-534.

[49]. Chris Karlof; Naveen Sastry; David Wagner. TinySec: a link layer security architecture for wireless sensor networks. In Proceedings of the 2nd international conference on Embedded networked sensor systems (SenSys '04). ACM, New York, NY, USA, 162-175.

[50]. Doriguzzi Corin, R.; Russello, G.; Salvadori, E. TinyKey: A light-weight architecture for Wireless Sensor Networks securing real-world applications. Wireless On-Demand Network Systems and Services (WONS), 2011 Eighth International Conference on Publication, Page(s): 68 – 75.

[51]. Subash, T.D.; Divya, C. Novel key pre-distribution scheme in wireless sensor network. Emerging Trends in Electrical and Computer Technology (ICETECT), 2011 International Conference on, vol., no., pp.959-963, 23-24 March 2011.

[52]. Wen Hu; Peter Corke; Wen Chan Shih; Leslie Overs. secFleck: A Public Key Technology Platform for Wireless Sensor Networks. In Proceedings of the 6th European Conference on Wireless Sensor Networks (EWSN '09), 2009. Springer-Verlag, Berlin, Heidelberg, 296-311.

[53]. Karthik, S.; Vanitha, K.; Radhamani, G. Trust management techniques in Wireless Sensor Networks: An evaluation. Communications and Signal Processing (ICCSP), 2011 International Conference on, vol., no., pp.328-330, 10-12 Feb. 2011.

[54]. Felix Gomez Marmol; Gregorio Martınez Perez. TRMSim-WSN, trust and reputation models simulator for wireless sensor networks. In Proceedings of the 2009 IEEE international conference on Communications (ICC'09). IEEE Press, Piscataway, NJ, USA, 915-919.

[55]. Karthik, N.; Dhulipala, V.R.S. Trust calculation in wireless sensor networks. Electronics Computer Technology (ICECT), 2011 3rd International Conference on, vol.4, no., pp.376-380, 8-10 April 2011.

[56]. Blaze, M.; Feigenbaum, J.; Lacy, J. Decentralized trust management. Security and Privacy, 1996. Proceedings., 1996 IEEE Symposium on, vol., no., pp.164,173, 6-8 May 1996.

[57]. Boukerch, A.; Xu, L.; EL-Khatib, K.. Trust-based security for wireless ad hoc and sensor networks. Comput. Commun. 30, 11-12 (September 2007), 2413-2427.

[58]. Vladimir Oleshchuk. Trust-based Framework for Security Enhancement of Wireless Sensor Networks. 4th IEEE Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS 2007) Sep, 2007.

[59]. Sarma Dhulipala, V.R.; Vishnu Prabha, B; Chandrasekaran, R.M. Trust Worthy Architecture for Mobile Adhoc Networks. Springer-Verlag Berlin Heidelberg 2010, BAIP 2010, CCIS 70, pp. 557–560, 2010.

[60]. Abusalah, L.; Khokhar, A.; BenBrahim, G.; ElHajj. TARP: trust-aware routing protocol. Proceeding of the 2006 international Conference on Communications and Mobile Computing, Vancouver, British Columbia, Canada. July 03 - 06, 2006.

[61]. Shaikh, R.A.; Jameel, H.; d'Auriol, B.J.; Heejo Lee; Sungyoung Lee; Young-Jae Song. Group-Based Trust Management Scheme for Clustered Wireless Sensor Networks. IEEE transactions on parallel and distributed systems, VOL. 20, NO. 11. November 2009.

[62]. Raha, A.; Babu, S.S.; Naskar, Mrinal Kanti; Alfandi, O.; Hogrefe, D. Trust integrated link state routing protocol for Wireless Sensor Networks (TILSRP). Advanced Networks and Telecommunication Systems (ANTS), 2011 IEEE 5th International Conference on, vol., no., pp.1,6, 18-21. December 2011.

[63]. Poolsappasit, N.; Madria, S. A Secure Data Aggregation Based Trust Management Approach for Dealing with Untrustworthy Motes in Sensor Network. Parallel Processing (ICPP), 2011 International Conference on, vol., no., pp.138-147, 13-16 September 2011.

[64]. Guoxing Zhan; Weisong Shi; Deng, J. Design and Implementation of TARF: A Trust-Aware Routing Framework for WSNs. Dependable and Secure Computing, IEEE Transactions on, vol.9, no.2, pp.184-197. March-April 2012.

[65]. Bao, F.; Chen, I.; Chang, M.; Cho, J. Hierarchical Trust Management for Wireless Sensor Networks and its Applications to Trust-Based Routing and Intrusion Detection. Network and Service Management, IEEE Transactions on, vol.PP, no.99, pp.1-15.

[66]. Dave Chappell, "Enterprise Service Bus" .O’Reilly: June 2004, ISBN 0-596-00675-6.

[67]. IBM WebSphere product description. http://www-03.ibm.com/software/products/en/websphere-mq

[68]. Microsoft BizTalk server product description. http://www.microsoft.com/BizTalkServer

[69]. Oracle Enterprise Service Bus product description. http://www.oracle.com/technetwork/middleware/service-bus/overview/index.html

[70]. Apache ServiceMix project page. http://servicemix.apache.org/

[71]. Fuse/Jboss ESB project page. http://jbossesb.jboss.org/

[72]. PetalsESB project page. http://petals.ow2.org/

[73]. WS02 project page. http://wso2.com/products/enterprise-service-bus

[74]. OpenESB project page. http://www.open-esb.net/

[75]. MuleESB project page. http://www.mulesoft.org/

[76]. UltraESB project page. http://adroitlogic.org/products/ultraesb.html

[77]. Butala, P.M.; Yuting Zhang; Little, T.D.C.; Wagenaar, R.C.; , "Wireless system for monitoring and real-time classification of functional activity," Communication Systems and Networks (COMSNETS), 2012 Fourth International Conference on , vol., no., pp.1-5, 3-7 Jan. 2012

[78]. Yamasue, K.; Takizawa, K.; Sodeyama, K.; Sugimoto, C.; Kohno, R.; , "Vital sign monitoring by using UWB Body Area Networks in hospital and home environments," Medical Information and Communication Technology (ISMICT), 2012 6th International Symposium on , vol., no., pp.1-4, 25-29 March 2012

[79]. Sungmo Jung, Jae Young Ahn, Dae-Joon Hwang, and Seoksoo Kim, “An Optimization Scheme for M2M-Based Patient Monitoring in Ubiquitous Healthcare Domain,” International Journal of Distributed Sensor Networks, vol. 2012, Article ID 708762, 9 pages, 2012.

[80]. Gonzalo Blázquez Gil, Antonio Berlanga, and José M. Molina, “InContexto: Multisensor Architecture to Obtain People Context from Smartphones,” International Journal of Distributed Sensor Networks, vol. 2012, Article ID 758789, 15 pages, 2012

[81]. Quang-Dung Ho; Thanh-Ngon Tran; Rajalingham, G.; Tho Le-Ngoc; , "A distributed and adaptive routing protocol designed for wireless sensor networks deployed in clinical environments," Wireless Communications and Networking Conference (WCNC), 2012 IEEE , vol., no., pp.2746-2750, 1-4 April 2012

[82]. Prabh, K.S.; Royo, F.; Tennina, S.; Olivares, T.; , "BANMAC: An Opportunistic MAC Protocol for Reliable Communications in Body Area Networks," Distributed Computing in Sensor Systems (DCOSS), 2012 IEEE 8th International Conference on , vol., no., pp.166-175, 16-18 May 2012

[83]. Nabi, M.; Geilen, M.; Basten, T.; , "Demonstrating on-demand listening and data forwarding in wireless body area networks," Sensor, Mesh and Ad Hoc Communications and Networks (SECON), 2012 9th Annual IEEE Communications Society Conference on , vol., no., pp.82-84, 18-21 June 2012

[84]. Yang, X. et al., "Wireless telemedicine and m-health: technologies, applications and research issues", Int. J. of Sensor Networks, Vol.10, No.4, pp.202 – 236, Year 2011

[85]. Navarro, K.F.; Lawrence, E.; , "WSN Applications in Personal Healthcare Monitoring Systems: A Heterogeneous Framework", e-Health, Telemedicine, and Social Medicine, 2010. ETELEMED '10. Second International Conference on , vol., no., pp.77-83, 10-16 Feb. 2010

[86]. Ho Cheng; Weihua Zhuang; , "Bluetooth-enabled in-home patient monitoring system: Early detection of Alzheimer's disease", Wireless Communications, IEEE , vol.17, no.1, pp.74-79, February 2010

[87]. Lawrence, E.; Sax, C.; Navarro, K.F.; Mu Qiao; , "Interactive Games to Improve Quality of Life for the Elderly: Towards Integration into a WSN Monitoring System", eHealth, Telemedicine, and Social Medicine, 2010. ETELEMED '10. Second International Conference on , vol., no., pp.106-112, 10-16 Feb. 2010

[88]. Min Chen; Gonzalez, S.; Leung, V.; Qian Zhang; Ming Li; , "A 2G-RFID-based e-healthcare system", Wireless Communications, IEEE , vol.17, no.1, pp.37-43, February 2010

[89]. Yan Zhang; Ansari, N.; Tsunoda, H.; , "Wireless telemedicine services over integrated IEEE 802.11/WLAN and IEEE 802.16/WiMAX networks", Wireless Communications, IEEE , vol.17, no.1, pp.30-36, February 2010

[90]. Phunchongharn, P.; Hossain, E.; Niyato, D.; Camorlinga, S.; , "A cognitive radio system for e-health applications in a hospital environment", Wireless Communications, IEEE , vol.17, no.1, pp.20-28, February 2010

[91]. Alessandro Redondi; Marco Chirico; Luca Borsani; Matteo Cesana; Marco Tagliasacchi, “An integrated system based on wireless sensor networks for patient monitoring, localization and tracking”, Ad Hoc Networks, Volume 11, Issue 1, Pages 39-53, January 2013.

[92]. Wan-Young Chung,; Young-Dong Lee,; Sang-Joong Jung,; , "A wireless sensor network compatible wearable u-healthcare monitoring system using integrated ECG, accelerometer and SpO2", Engineering in Medicine and Biology Society, 2008. EMBS 2008. 30th Annual International Conference of the IEEE , vol., no., pp.1529-1532, 20-25 Aug. 2008

[93]. Llosa, J.; Vilajosana, I.; Vilajosana, X.; Marques, J.M.; , "Design of a Motion Detector to Monitor Rowing Performance Based on Wireless Sensor Networks", Intelligent Networking and Collaborative Systems, 2009. INCOS '09. International Conference on , vol., no., pp.397-400, 4-6 Nov. 2009

[94]. López-Matencio, P.; Alonso, J.V.; González-Castaño, F.J.; Sieiro, J.L.; Alcaraz, J.J.; , "Ambient intelligence assistant for running sports based on k-NN classifiers", Human System Interactions (HSI), 2010 3rd Conference on , vol., no., pp.605-611, 13-15 May 2010

[95]. Vijay Sivaraman; Ashay Dhamdhere; Hao Chen; Alex Kurusingal; Sarthak Grover; , “An experimental study of wireless connectivity and routing in ad hoc sensor networks for real-time soccer player monitoring”, Ad Hoc Networks, vol.11(3), pages 798-817, Year 2013

[96]. Mariotti, C.; Lakafosis, V.; Tentzeris, M.M.; Roselli, L., "An IPv6-enabled wireless shoe-mounted platform for health-monitoring", Wireless Sensors and Sensor Networks (WiSNet), 2013 IEEE Topical Conference on , vol., no., pp.46,48, 20-23 Jan. 2013

[97]. Daniele Toscani; Ilaria Giordani; Mauro Cislaghi; Luigi QuarenghiToscani; , “Querying sensor data for environmental monitoring”, Int. J. of Sensor Networks, Vol.10, No.3, pp.132 – 142, Year 2011

[98]. LG G watch product page. http://www.lg.com/global/gwatch/one/index.html

[99]. Samsung Gear 2 neo product page. http://www.samsung.com/uk/consumer/mobile-devices/wearables/gear/SM-R3810ZKABTU

[100]. Motorola Moto 360 product page. https://moto360.motorola.com/

[101]. Apple watch product page. https://www.apple.com/uk/watch/

[102]. Asus Zen watch product page. http://www.asus.com/Phones/ASUS_ZenWatch_WI500Q/

[103]. WIMM one product specification (product discontinued). http://en.wikipedia.org/wiki/WIMM_One

[104]. LG LifeBand product page. http://www.lg.com/uk/fitness-activity-trackers/lg-FB84

[105]. Fitbit Flex product page. http://www.fitbit.com/uk/flex

[106]. Nike Fuelband SE product page. http:// store.nike.com/gb/en_gb/pd/fuelband-se/pid-886061/pgid-886058

[107]. Microsoft MSBand product page. http://www.microsoftstore.com/UK/MicrosoftBand

[108]. Sony SmartBand product page. http://www.sonymobile.com/global-en/products/smartwear/smartband-swr10

[109]. Zephyr HxM product page. http://zephyranywhere.com/products/hxm-bluetooth-heart-rate-monitor/

[110]. Zephyr Bioharness3 product page. http://zephyranywhere.com/products/bioharness-3/

[111]. Garmin Forerunner 920XT product page. https://buy.garmin.com/en-US/US/into-sports/running/forerunner-920xt/prod137024.html

[112]. Polar H7 product page. http://www.polar.com/uk-en/products/accessories/H7_heart_rate_sensor

[113]. CooSpo HRM-H8 product page. http://coospo.com/h8.html

[114]. RESTFul architectures reference. Architecture of the World Wide Web, Volume One. W3C Recommendation. http://www.w3.org/TR/webarch/

[115]. JSON Reference. IETF RFC 4627. https://www.ietf.org/rfc/rfc4627.txt

[116]. CoAP Reference. IETF RFC 7252. http://tools.ietf.org/html/rfc7252

[117]. Java Message Service reference documentation: http://docs.oracle.com/javaee/6/tutorial/doc/bncdr.html

[118]. AMQP specification. http://www.amqp.org/sites/amqp.org/files/amqp.pdf

[119]. Chen, Harry, et al. "Soupa: Standard ontology for ubiquitous and pervasive applications." Mobile and Ubiquitous Systems: Networking and Services, 2004. MOBIQUITOUS 2004. The First Annual International Conference on. IEEE, 2004.

[120]. OpenCyc project main page. http://www.opencyc.org/doc

[121]. FOAF Vocabulary Specification 0.99. http://xmlns.com/foaf/spec/

[122]. OWL 2 Web Ontology Language . Document Overview (Second Edition). http://www.w3.org/TR/owl2-overview/

[123]. Familiar, Miguel S., José F. Martínez, and Lourdes López. "Pervasive smart spaces and environments: a service-oriented middleware architecture for wireless Ad Hoc and sensor networks." International Journal of Distributed Sensor Networks 2012 (2012).

[124]. The Transport Layer Security (TLS) Protocol. IETF. https://tools.ietf.org/html/rfc5246

[125]. Project LifeWear page at ITEA2 call site. https://itea3.org/project/lifewear.html

[126]. Castillejo, P.; Martinez, J.-F.; Rodriguez-Molina, J.; Cuerva, A., "Integration of wearable devices in a wireless sensor network for an E-health application," Wireless Communications, IEEE, vol.20, no.4, pp.38,49, August 2013.

[127]. Pedro Castillejo, José-Fernán Martínez, Lourdes López, and Gregorio Rubio, “An Internet of Things Approach for Managing Smart Services Provided by Wearable Devices,” International Journal of Distributed Sensor Networks, vol. 2013, Article ID 190813, 9 pages, 2013. doi:10.1155/2013/190813.

[128]. Pedro Castillejo, José-Fernán Martínez-Ortega, Lourdes López, and José Antonio Sánchez Alcón, “SensoTrust: Trustworthy Domains in Wireless Sensor Networks,” International Journal of Distributed Sensor Networks, Article ID 484820, in press.

[129]. Rodríguez-Molina, J.; Martínez, J.-F.; Castillejo, P.; López, L. Combining Wireless Sensor Networks and Semantic Middleware for an Internet of Things-Based Sportsman/Woman Monitoring Application. Sensors 2013, 13, 1787-1835.

[130]. Martínez, J.-F.; Rodríguez-Molina, J.; Castillejo, P.; de Diego, R. Middleware Architectures for the Smart Grid: Survey and Challenges in the Foreseeable Future. Energies. 2013, 6, 3593-3621.

[131]. Jesús Rodríguez-Molina, José-Fernán Martínez, Pedro Castillejo, and Rubén de Diego, “SMArc: A Proposal for a Smart, Semantic Middleware Architecture Focused on Smart City Energy Management,” International Journal of Distributed Sensor Networks, vol. 2013, Article ID 560418, 17 pages, 2013. doi:10.1155/2013/560418.

[132]. Sanchez Alcon, J. A., Lopez, L., Martinez, J. F., Castillejo, P. (2013, August). Automated determination of security services to ensure personal data protection in the Internet of Things applications. In Innovative Computing Technology (INTECH), 2013 Third International Conference on (pp. 71-76). IEEE.

[133]. Gregorio Rubio, Estefanía Serral, Pedro Castillejo, José Fernán Martínez. A Novel Context Ontology to Facilitate Interoperation of Semantic Services in Environments with Wearable Devices. On the Move to Meaningful Internet Systems: OTM 2012 Workshops Lecture Notes in Computer Science Volume 7567, 2012, pp 495-504

[134]. Martínez, José-Fernán, et al. "Wireless sensor networks in knowledge management". International Conference on Computational Science ICCS 2010. Published in Procedia Computer Science 1.1 (2010): 2291-2300.

6.2. Acronyms

AES	Advanced Encryption Standard
AMQP	Advanced Message Queuing Protocol
API	Application Programming Interface
BAN	Body Area Network
BLE	Bluetooth Low Energy
CA	Certificate Authority
CoAP	Constrained Application Protocol
CRL	Certificate Revocation List
CS	Ciphering Suite
CSR	Certificate Signing Request
DK	Domain Key
DKn	Domain N Key
DKS	Domain Key Server
DoS	Denial of Service
DSSS	Direct-Sequence Spread Spectrum
ECC	Elliptic Curve Cryptography
ECG	Electrocardiogram
EDR	Enhanced Data Rate
EEPROM	Electrically Erasable Programmable Read-Only Memory
EJB	Enterprise JavaBeans
EMI	Electro-Magnetic Interference
ESB	Enterprise Service Bus
GUI	Graphical User Interface
HS	High Speed
HTTP	Hypertext Transfer Protocol
IaaS	Infrastructure as a Service
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
IP	Internet Protocol
iPaaS	integration Platform as a Service
ISM	Industrial, Scientific and Medical
JAX-RS	Java API for RESTful Web Services
JCR	Journal Citation Report

JMS	Java Message Service
JSON	JavaScript Object Notation
Kp	Public Key
Ks	Private Key
LE	Low Energy
LED	Light-Emitting Diode
LoWPAN	Low-power wireless personal area network
MIDP	Mobile Information Device profile
MW	Middleware
N	Node
NFC	Near Field Communication
NID	Node ID
NK	Node Key
OFDM	Orthogonal Frequency-Division Multiplexing
OS	Operating System
OWL	Web Ontology Language
PaaS	Platform as a Service
PDU	Protocol Data Unit
PKI	Public Key Cryptography
QoS	Quality of Service
RAM	Random Access Memory
REST	REpresentational State Transfer
RFID	Radio Frequency Identification
ROM	Read-Only Memory
RSA	Rivest, Shamir and Adleman
RX	Reception
SDK	Software Development Kit
SM	Security Manager
SOA	Service Oriented Architecture
SotA	State-of-the-Art
SSL	Secure Sockets Layer
TCP	Transfer Control Protocol
TD	Trust Domain
TP	Trust Policy
TX	Transmission

UDP	User Datagram Protocol
UPM	Technical University of Madrid
URI	Universal Resource Identifier
URL	Uniform Resource Locator
WADL	Web Application Description Language
WDSB	Wearable Device Service Bus
WIDP	Wearable Inter-domain communication protocol
WLAN	Wireless Local Area Network
WSN	Wireless Sensor Network
WWBAN	Wearable Wireless Body Area Network
XaaS	Everything as a Service
XML	eXtensible Markup Language