Computing and Informatics, Vol. 34, 2015, 911–940

CONSTRUCTION OF HARDWARE COMPONENTS FOR THE INTERNET OF SERVICES

Paweł Bachara, Robert Brzoza-Woch, Jacek Długopolski
Piotr Nawrocki, Wojciech Zaborowski, Krzysztof Zieliński

AGH University of Science and Technology
Faculty of Computer Science, Electronics and Telecommunications
Department of Computer Science
al. A. Mickiewicza 30
30-059 Kraków, Poland
e-mail: zab@agh.edu.pl

Andrzej Ruta

Samsung R & D Institute Poland
Al. Armii Ludowej 26
00-609 Warszawa, Poland

Abstract. In this paper we focus on a hardware realization of web services (WS) and their integration within the service-oriented architecture (SOA). Previous approaches to the implementation of network-enabled services in hardware covered only very specific types of applications and were platform-dependent. Our contribution is a generic framework where heterogeneous everyday objects are enhanced with appropriate hardware extensions. This turns them into intelligent electronic devices that can sense the environment as well as interact with it, exposing their functionality via public WS interface. An integration scheme is introduced to allow the augmented objects to be used within highly distributed enterprise applications. Each web service is mapped to a functionally equivalent Open Services Gateway initiative (OSGi) service so that it can be dynamically added to the pool of elementary services accessible within the enterprise service bus (ESB). Our approach is exemplified by several web services for environment monitoring, mechanical control and visual inspection, all implemented in a reconfigurable hardware. A case study of integrating and using such services is also presented.

Keywords: Hardware web services, service oriented architecture, intelligent sensors, actuator control, FPGA, ESB

Mathematics Subject Classification 2010: 68T40, 68U10, 68W10, 70B15, 70Q05, 93C40, 93C62, 93C83, 93C85, 94A08, 94A13, 97P30, 97P60

1 INTRODUCTION

Modern web services make up the core of machine-to-machine interaction over the network in both Business-to-Business (B2B) and Business-to-Client (B2C) segments of the market. Although their role is increasingly important and they are used in more and more sophisticated distributed applications, relatively little has been done so far to integrate software and hardware under a unified web services framework. While many application-specific hardware devices, such as certain street surveillance cameras, are already accessible at any time and from any network-enabled machine in the world, their functionality usually remained fixed. There is no common support for easy exposition of heterogeneous, reconfigurable and largely autonomous hardware resources over the web.

Looking from a broader perspective, the problem can be formulated as concerning a missing layer at the contact point of two worlds: the domain of commodity objects and tools used by humans in their everyday lives and the existing computing infrastructure comprised of various large-scale enterprise systems. Those objects are called *things* further in this paper. The former, in order to be useful for internet-era consumers, require an appropriate adaptation, such as digitization or network enablement. The goal of such extensions is to turn ordinary objects into smart electronic appliances that can be localized in the network and provide data, which is the foundation of the Internet of Things (IoT) paradigm [1, 2, 3, 4, 5]. Although the idea is simple, its application in general is difficult for instance due to the heterogeneity of existing objects and the lack of well-established implementation standards.

On the other hand, contemporary enterprise systems support business processes, information flow and data analysis within and between organizations, and we are all becoming more and more heavily dependent on such infrastructures. However, they could serve humans much more efficiently if the computing power, flexibility and portability of smart hardware were better exploited. Even if appropriate networked devices are already available, they must be made capable of providing services through a unified interface and in such a way that they can be discovered, used flexibly and replaced when needed in environments with thousands of users and a variety of technologies. What lacks is another layer of integration in the form of an appropriate middleware, that could let potential consumers to perceive hardware appliances as regular software web services.

Figure 1 illustrates the general idea of integrating the World of Things (WoT) with the software world of enterprise-class computer systems. The left ellipse sym-

bolizes a heterogeneous set of utility objects (things), such as temperature sensors, door locks or surveillance cameras which, as off-the-shelf market products, cannot be used remotely. They usually have no network interface. Some are inherently mechanical and cannot even generate any digital output. The first level of integration facilitates transition from the World of Things to the Internet of Things (center ellipse) which allows us to connect the objects to one another and to the outside world. This step requires an appropriate hardware adaptation. To enable smooth integration of the resulting networked devices with enterprise systems, Service-Oriented Architecture [6] design principles must be applied. It requires implementation of the web service functionality on the device's side, as well as providing an appropriate middleware to enable provisioning of the hardware services to the corporate or public sector environments. Therefore, the second level of integration maps the Internet of Things into what we call the Internet of Services (right ellipse).

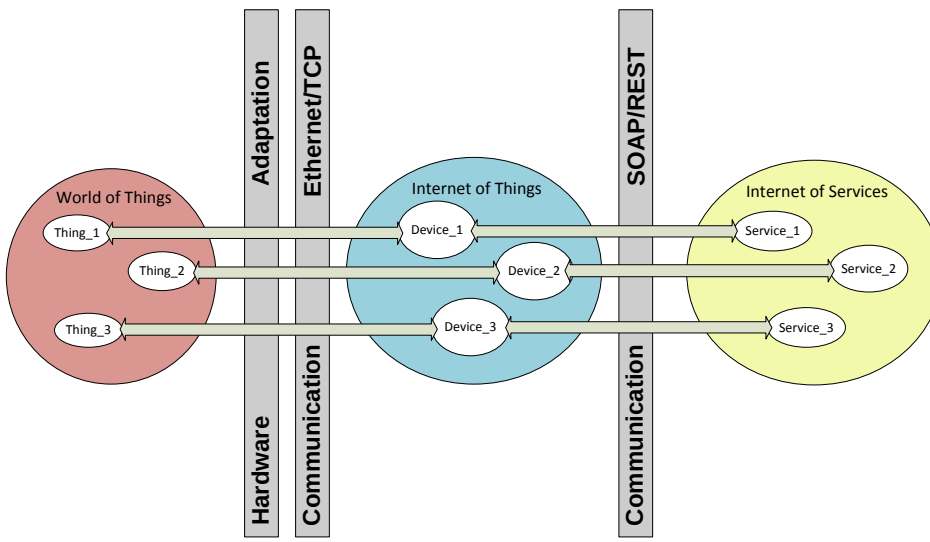

Figure 1. The idea of integrating the domain of commodity objects (things) with the enterprise software in accordance with the service-oriented architecture paradigm

The aim of this paper is to present a novel methodology for building SOA-compliant applications that make extensive use of heterogeneous hardware services for general-purpose environment monitoring and actuator control. These services are based on devices which, upon the appropriate hardware extension, become discoverable and accessible over the network through web service interfaces. A range of both custom-made and off-the-shelf appliances are investigated where the necessary communication layers and WS stacks have been implemented partly or fully in hardware. The main focus is on Field Programmable Gate Array (FPGA) based devices which are computationally powerful and can be flexibly tuned to various ap-

plication scenarios through a dynamic reconfiguration. Furthermore, we show how such services can be transparently integrated with the enterprise systems within the message-oriented architectures, such as the Enterprise Service Bus [7].

The main contributions of this paper are: 1) a set of prototype services embedded in reconfigurable, technologically diverse hardware devices, 2) a generic architecture and a methodology of implementing such services, and 3) middleware that enables flexible exploitation of these services within the ESB so that the hardware implementation and details of the internal communication protocols remain transparent from the caller's perspective. An example scenario of employing the selected hardware services in an intelligent house is also outlined. It involves smart sensors for environmental parameter monitoring and visual inspection, as well as devices for mechanical door, ventilation, heating and light switch control – all remotely customizable.

The rest of this paper is composed as follows. In Section 2 we review the state of the art in the hardware implementation of web services. In Section 3 we analyze the challenges involved in integrating such embedded services with enterprise applications. A generic set of design principles is proposed to address these challenges. Subsequently, we delve into details on how different kinds of devices can be extended to turn them into fully-functional web services, taking into account their technology-specific features and limitations. Section 4 describes how our services can be transparently used as building blocks of larger SOA systems. In Section 5 we provide an intelligent building case study where the proposed methodology and the selected prototype devices assembled can improve the resource utilization and enhance the residents security. Finally, Section 7 concludes this work.

2 PREVIOUS WORK

Service Oriented Architectures [6] are particularly well-suited for deploying multiple applications that need to be run on varied technologies and platforms while communicating with one another. An increasing number of solutions uses the SOA paradigm [8]. The applications that process data delivered by specialized hardware devices, such as environmental parameter sensors, fit perfectly in the SOA paradigm as long as these devices can be made fully interoperable over the network, regardless of the complexity of their internal implementations and interfaces. This, however, seems to be a major problem taking into account a huge technological and functional diversity of smart devices available on the market together with a multitude of related implementation standards and communication protocols. Below we briefly review the previous work on simultaneous usage of software and hardware components in contemporary distributed applications.

One of the most popular approaches to integrating hardware elements in SOA systems is the service-oriented device architecture (SODA) [9]. Most elements of this architecture reside in a layer which separates the physical and digital realms. External devices are connected to other SOA system elements via the web service

binding component and their functionality is exposed as web services. The details of communication with the physical devices are hidden behind higher-level abstractions. SODA implementations are not limited to specific technologies, what is mainly why this architecture has gained so much popularity in the industry. However, as opposed to the solutions presented in this paper, SODA limits the device's functionality to that offered by its associated connection adapter. Even if the interface is flexible enough to realize the required functionality, all low-level operations have to be performed by calling appropriate web services, which generates unnecessary communication overhead.

Due to its widespread usage and flexibility web services technology is becoming a popular means of enabling remote control of electronic devices. There are many solutions where a single-chip microcontroller does the job. It usually suffices for simple control systems, but in cases where time and synchronization are key factors, the FPGA technology proves more adequate. In such cases the time-dependent parts of the control process are typically implemented in the FPGA matrix, but the web server parts still run as sequential code executed by microprocessors, possibly under control of operating systems [10]. In a substantial number of previous studies [11] a C-to-hardware high-level description language is also used to represent hardware semantics using high-level software abstractions. This facilitates automatic hardware logic inference for a specific target behaviour. In this paper we wish to show (among others) that while this strategy is reasonable, alternative approaches are also possible. For instance, both parts: control processes and web server logic can be implemented in one FPGA chip using the very-high-speed integrated circuits Hardware Description Language (VHDL), without recourse to microprocessors and operating systems. Moreover, the same hardware platform can be reused for many different tasks upon reconfiguration.

The use of embedded devices, such as smartphones or wireless sensor networks for web services development has already been described in [12] and [13]. The implementation is normally done on a microcontroller or microprocessor with appropriately large computing power [14, 15]. Moreover, a number of web service development toolkits exist for both Java Platform Micro Edition and .NET Compact Framework, in addition to the platform-independent gSOAP toolkit [16]. However, solutions in which an FPGA chip is used to implement core server-side functions, such as Simple Object Access Protocol (SOAP) request handling, remain rather uncommon [17, 18].

One example of an embedded system utilizing FPGA as a remote co-processor for performance optimization was given in [19]. The authors of that paper used an FPGA module on a Peripheral Component Interconnect (PCI) board attached to a conventional personal computer (PC). The processing power of the FPGA was exposed as a set of web services, one per library function implemented in hardware. In this scenario the clients simply provide input data and receive processing results as though they made local library function calls. However, the system could not operate autonomously and was neither designed for sensing the environment, nor reacting to any changes in it.

There are also many solutions that incorporate FPGAs to enhance the computational power of the main system without exposing a web service interface. Some examples in the field of image processing are given in [20, 21, 22, 23, 24] and more recently in [25]. The system described in the last of these studies uses a soft-core implementation of the NIOS-II microcontroller with custom instructions to speed up certain image processing tasks.

A separate group of embedded devices that can be used to provide web services are wireless sensors. They are usually small, low-complexity devices with wireless communication capability. When used in large numbers, they create so-called Wireless Sensor Networks (WSN) [26, 27]. This technology has been applied in many areas where the acquisition and processing of various kinds on environmental data on spatially distributed phenomena or processes is required. In [28] the following applications of WSNs are listed: structural health monitoring, traffic control, healthcare, pipeline monitoring, precision agriculture, active volcano monitoring or underground mining. Due to the severe hardware and software limitations of WSN nodes, sensor networks are not able to share their functionality in a manner consistent with the web services or other SOA-compatible standard. Therefore, some kind of gateway is always needed. An example solution that allows sensor networks to be exposed as web services was presented in [29].

3 ELEMENTS OF THE ENVIRONMENT AS HARDWARE SERVICES

The role of inexpensive yet robust processing devices, such as various smart sensors or embedded computers incorporated in consumer electronics, has grown significantly over the past decade. The current development trend for such devices is towards making them active participants in business processes according to the Internet of Things paradigm [1, 2, 3, 4, 5]. This is achieved by assigning identities to hardware resources and creating services that can interact with them over the internet. Below we outline a generic framework for building and integrating hardware services, assuming heterogeneity of the underlying devices (things), i.e. their different technological realizations, varying computational capacity, existence or lack of networking capability, and the like.

Let us start with an analysis of the challenges faced when building hardware services. First of all, the technological diversity of potentially useful devices is immense. Some of them, such as door locks, are purely mechanical. Therefore, for remote control they require special actuators based on servomotors. Others may only produce analogue output that must first be converted to a digital form. The devices may also differ significantly in terms of complexity. For instance, an air pressure sensor will always be much less complex than an embedded processor based device or a piece of advanced consumer electronics. The latter categories may be based on application-specific integrated circuits (ASICs), programmable microcontrollers, FPGAs or even combinations of the above. Moreover, highly autonomous devices will usually require some degree of embedded intelligence which calls for further

hardware resources, e.g. extra memory for data storage or special-purpose actuator controllers.

Another important issue is device programming. Most commonly, it is done with the help of the so-called Electronic Design Automation (EDA) software using hardware description languages, such as VHDL [30] or Verilog [31]. This gives developers full control over the hardware design. However, the use of microcontrollers facilitates typically software-style device programming, e.g. in C/C++ languages, which increases productivity. There are also successful examples of a mixed hardware/software co-design approach to this problem.

To communicate with the outside world, a device must gain networking capability. To enable it, hardware extensions are usually required. At this point a number of new problems emerge. First, the network interface (typically a standalone off-the-shelf module) must work properly with the existing circuits. Second, the format of data sent and received has to be carefully considered and the whole Transmission Control Protocol/Internet Protocol (TCP/IP) stack must be implemented. While it is embedded in many popular hardware solutions available on the market, high-level web service functionality, such as Extensible Markup Language (XML) serialization and de-serialization, is not the case. Finally, other constraints, such as design layout (including the necessary extension boards), may prevent the device from being assembled efficiently. For instance, in many visual inspection tasks only miniature image sensors prove practically useful.

To address the above issues in a uniform way, we propose a generic design that can be consistently followed when building hardware web services of arbitrary complexity and purpose. Our concept is illustrated in Figure 2. Details related to specific technologies that are beyond the scope of this generic design are described later on in this section where we focus on different kinds of special-purpose devices.

In the most general situation, an object from the World of Things represents a simple physical appliance with no electronics, where direct manual interaction is the only possible way of using it. The goal is to enable automatic and remote control of the device from enterprise software systems through a web service interface. To achieve this goal, the following key steps are required:

Hardware adaptation – this calls for addition of the necessary mechanical parts and actuators to the device, as well as low-level protocol extensions by adding various electronic circuits so as to allow digital control.

Service logic implementation – the goal in this step is to build a dedicated server embedded in the augmented device created in the first step. Typically, it requires some programming effort.

Network enablement – this step involves augmenting the device with a communication module, typically implementing the TCP/IP protocol stack.

Web service interface development – it is aimed at implementation of the SOAP or Representational State Transfer (REST) based web services proto-

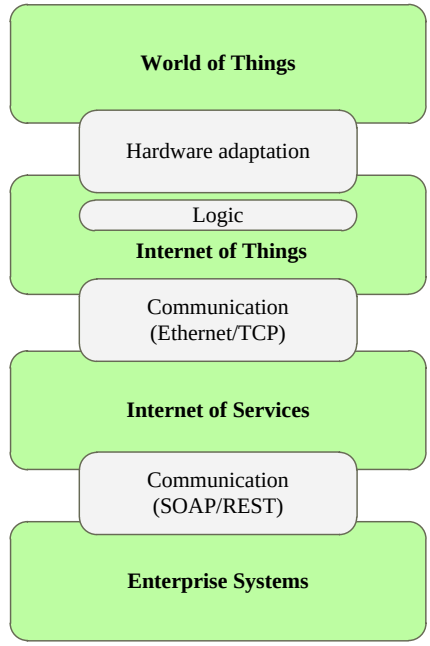

Figure 2. Generic design of the proposed hardware services

col stack and control logic to expose the functionality of the server to individual clients or enterprise applications.

Depending on the nature of the device, some of the above steps might be simplified or skipped. For example, various sensor devices might be pre-equipped with low-level communication protocol implementations by their manufacturers, providing easy digital access to sensor measurements. As a result, such devices need little or no hardware adaptation and the provided internal hardware interfaces can be used "out of the box". Other devices may already have full networking capabilities at the time of assembly, which greatly simplifies the final step of the above hardware service creation process. Similarly, service logic implementation might also be much less of a challenge in specific situations, e.g. when the device has a built-in central processing unit (CPU) with a Java virtual machine running on it.

Once augmented in compliance with the above scheme, the device should act as a server that can be queried directly through the WS interface by the client applications or used within message-oriented architecture models, such as the Enterprise Service Bus. No restrictions are assumed as to the semantics and complexity of the data exchanged. For instance, they can represent simple temperature readouts, robot movement description vectors or the results of semantic video analysis. Finally, the proposed methodology promotes construction of highly autonomous devices that can perform much more than simple environmental data acquisition.

In the following sections we elaborate the above mentioned concepts focusing on three example categories of devices: control devices, smart cameras, and intelligent sensor networks. For each category we describe how the four steps of the proposed hardware service creation process are followed taking the technology-specific constraints into account.

3.1 Control Devices

Our environment is full of commodity objects and tools which are regularly used by people in their everyday lives. More sophisticated devices, such as mobile robots, are also gaining popularity due to their usefulness in the industry, e.g. for sewage systems control, as well as in civil defence to examine contaminated ground or for bomb disposal. All these devices are used to help improve business processes or routine human activities and offer great potential for delivering different kinds of services. Usually such tools are controlled manually or by specific custom interfaces. Below we will show how to achieve the same level of control through a unified SOA-compliant web service interface for devices such as door locks, light switches, fuse switches, and a hexapod robot.

All the devices in question require a hardware adaptation based upon the addition of servomotor actuators which enable the mechanical elements to be moved automatically, according to the remote user's needs. These servos take Pulse Width Modulation (PWM) positional commands, using a digital waveform, and move the output shaft accordingly. The augmented devices can be controlled directly from digital circuits using microcontrollers or other types of digital chips.

The nature of the control devices considered here is inherently massively parallel. By default all devices work independently, but often should be controlled simultaneously in real time. This is why the controllers for groups of such devices (e.g. six legs of a hexapod mobile robot or a group of robots in an industry assembly line) need to allow parallel and independent access to them, even at the level of individual device features. Regular microprocessors and microcontrollers do not provide such capabilities as they are usually based on a single physical core that has to divide time and performance between all control threads. One possible solution to this problem could be to use many processors in a single controller, one for each control feature. However, taking into account power consumption and electronic circuit complexity, as well as expected software development problems and economic considerations, then a better solution is to employ a modern FPGA chip. An example of the approach, where the FPGA-based adaptive controller of servo drive is used, we present in [32]. A single FPGA chip, in conjunction with a network adapter, can perform all the necessary tasks for a network-controlled device to become a full-fledged hardware- implemented SOA service (HSOA). Furthermore, connecting sensors and effectors to the FPGA makes the service capable of sensing and influencing the surrounding environment.

Currently available network adapters are already fitted with hardware-implemented TCP/IP protocol stacks. For instance, Digi International Inc. manufac-

tures wired and wireless interchangeable embedded network modules: Digi Connect ME and Digi Connect Wi-ME. In fact, when using such off-the-shelf modules, the developer needs to implement in the FPGA matrix (apart from the HSOA service functionality) the relevant Hypertext Transfer Protocol (HTTP) and SOAP protocols. This implementation is done directly in a hardware description language (e.g. VHDL).

The proposed generic design scheme for integration of control devices and its mapping to the physical realization of the HSOA controller is shown in Figure 3. A device equipped with servomotors is controlled by the server through the PWM standard interface. Enterprise applications perceive the device as a web service and communicate with it over ethernet or wifi.

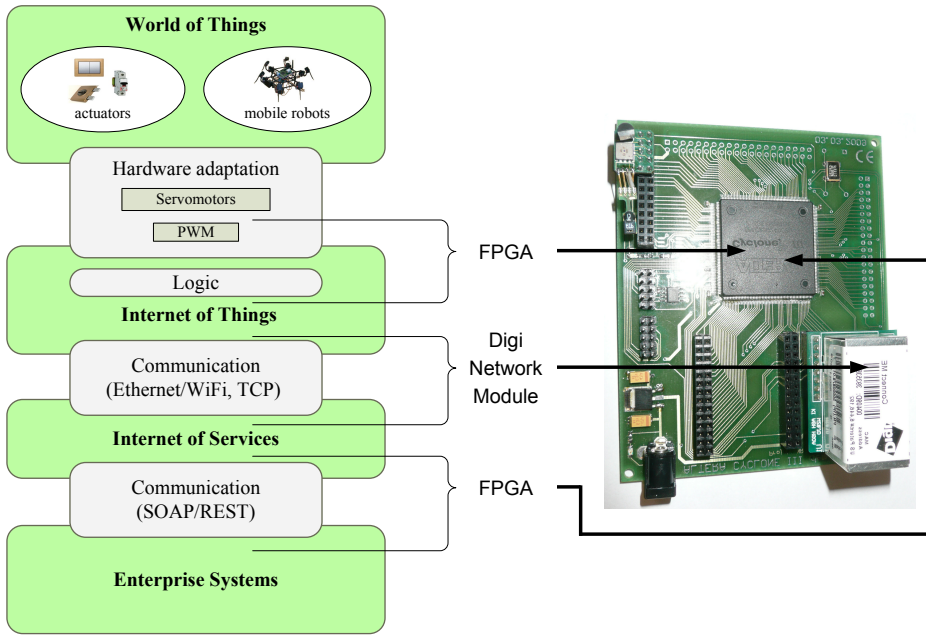

Figure 3. The design schema for a control device (left) and its hardware realization (right)

Figure 4 shows a block diagram of the HSOA board. There are two main components: the Digi Connect module which provides network communication, and the FPGA chip (part number EP3C25 from Altera) which encapsulates service's logic. Lower layers of the network module, such as the TCP/IP protocol stack, are implemented by the manufacturer. The remaining application-layer protocols required for web service functionality (HTTP and SOAP), low-level protocol interfaces (PWM or other), and the functional logic are implemented in the FPGA matrix.

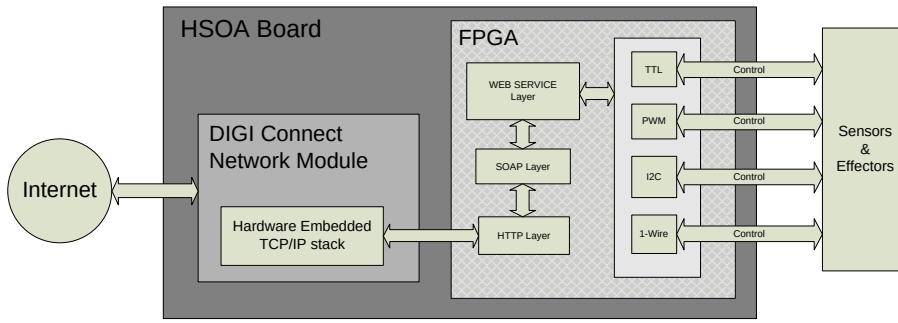

Figure 4. Architecture of the HSOA board

The HSOA board is a versatile controller ready for use with a wide range of devices. It can manipulate, in a uniform way, light switches, door locks or even mobile robots. An example usage scenario is depicted in Figure 5.

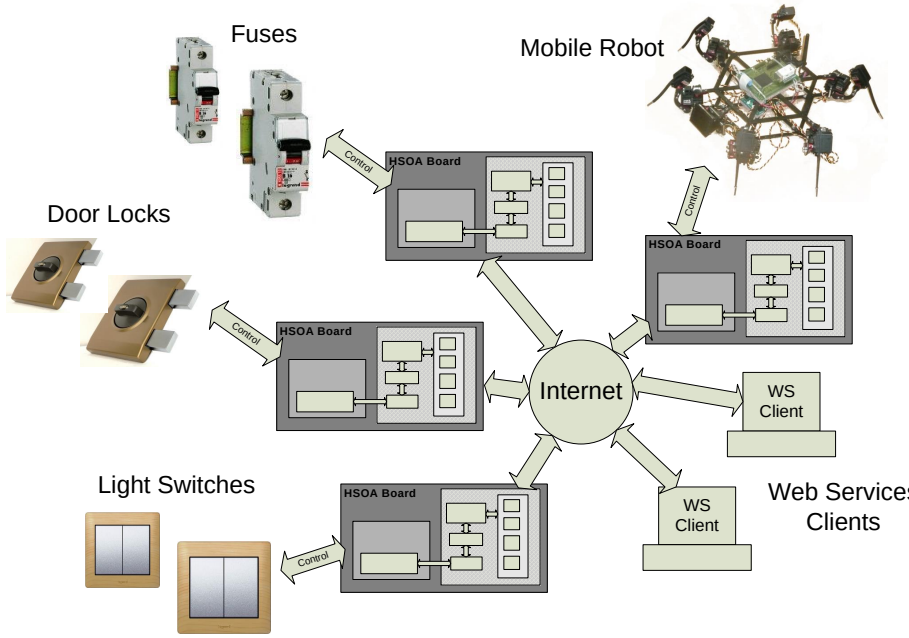

Figure 5. A generic usage scenario for the HSOA board acting as a versatile controller

It should be noted that although certain hardware extensions are necessary in the course of the proposed HSOA service development process, this extra effort is still negligible when compared to alternative approaches, e.g. those involving multiple microcontrollers.

3.2 Smart Cameras

A smart camera service is a custom hardware-based server intended to be used for various purposes, e.g. for personal security, access control, visual surveillance or object localization [33]. Unlike a regular camera which simply records and transmits a video stream, the smart camera should be able to perform analysis of the image content and autonomously react to certain events detected in the observed scene. Moreover, large amounts of rapidly incoming data require the above tasks to be performed in real (or near-real) time.

In accordance with the proposed generic service design, hardware adaptation is first carried out to create a server capable of performing a variety of machine vision tasks. It is a complex digital embedded system that consists of an integrated image sensor, a high-end FPGA chip (Stratix-II, part number EP2S60 from Altera), a modern microcontroller (ARM Cortex-M3 core, part number STM32F103RCT6), an advanced, highly integrated network interface (part number EM1206 from Tibbo), and several other logic elements. The camera can perform relatively complex computational tasks, such as motion detection or face recognition. This enables image processing to be done in place, eliminating the need for costly and delay-prone image data transfer over the network.

No internal protocol extensions are necessary as the image sensor used in the design is equipped with a configurable digital parallel interface. It outputs data in the YCbCr format. The data are received by the FPGA chip and then directed to a custom-made luminance (greyscale) extraction module described in Verilog hardware description language (HDL). The extracted greyscale pixel information is written to Synchronous Dynamic Random Access Memory (SDRAM) by a specialized Direct Memory Access (DMA) controller. The hardware platform provides access to two independent channels of SDRAM, each capable of 332 Mbit/s data transfer. Additional Verilog modules provide frame-level synchronization and start-of-frame/end-of-frame signalling required by the server program for image processing.

A prototype camera (Figure 6) was designed in the form of a modular cube for ease of physical hardware upgrade and space efficiency. Most time-critical tasks, e.g. image content analysis, are carried out in the internal logic of the FPGA chip (hardware accelerator block in Figure 7). This enables truly parallel implementation of many algorithms as well as easy integration with one or more embedded microprocessor cores. The majority of modules in the FPGA's internal design are implemented in Verilog HDL. Modules which are properly described in HDL can be efficiently implemented in the FPGA structure and typically offer a good performance. However, the implementation of complex features in HDLs is often very difficult, even for developers with the necessary hardware design experience. For this reason, to implement the selected high-complexity image processing modules, we decided to use a C-to-HDL compiler, Impulse CoDeveloper [34] from Impulse Accelerated Technologies.

Simpler tasks that are not time-critical and can be performed in a sequential manner run as executable C++ code on an embedded microcontroller, internal to the

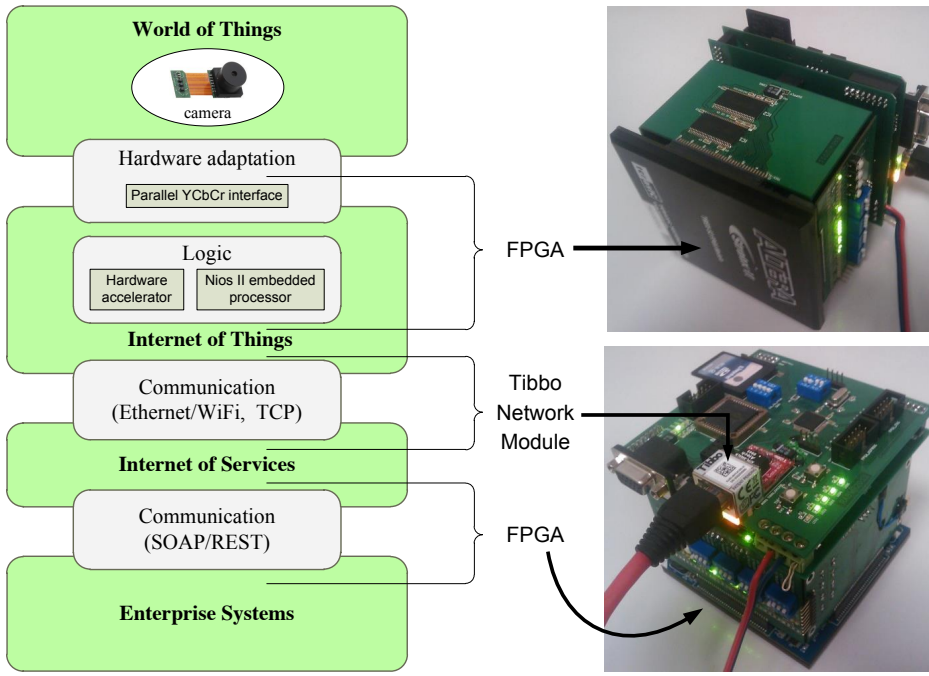

Figure 6. The design schema for a smart camera (left) and its hardware realization (right)

FPGA chip. The Altera's NIOS-II microcontroller core was chosen for this purpose. These lower-complexity tasks include, for instance, the operation of the SOAP protocol (including data transfer to/from the low-level communication module, XML serialization/de-serialization, and server method invocation), actuator control, and the synchronization of parallel modules.

The network interface used in our smart camera prototype is based on the advanced, highly integrated ethernet and wifi modules by Tibbo Technology (EM-1206 model with GA1000 as an extension for the wifi interface). In the basic server version these modules are used to handle TCP connections and to forward HTTP/SOAP-enveloped data to the FPGA and back, what provides a basic network connectivity (Internet of Things). A simplified SOAP protocol is implemented as part of the server-side program running on the NIOS-II microcontroller. It enables remote invocation of server methods and delivery of image processing results to the service caller via a web service interface (Internet of Services).

FPGA functionality is fully dependent on the configuration stored in its memory. This feature allows developers to modify the server's functionality without any changes in its underlying hardware. Moreover, when using an additional microcontroller (STM32F103 from ST Microelectronics), the above mentioned functionality replacement can be done remotely, i.e. by means of a network interface.

Figure 7 shows sample client applications cooperating with the smart camera service. Upon obtaining the interface description from the Web Services Description Language (WSDL) file, the camera can be accessed directly from an end-user application or through the Proxy Service mechanism described in Section 4, in both configurations acting as a web service. There is also an additional, more direct interface which allows a low-level monitoring, debugging, maintenance and remote configuration updates by the service's administrator. The device is equipped with an integrated image sensor and memory, and it can drive external actuators. Therefore, its functionality is not restricted to image acquisition and processing. It can also be used as an autonomous controller that reacts to specific environmental changes, logging relevant events and sending (on demand) the aggregated information to the calling application for further analysis.

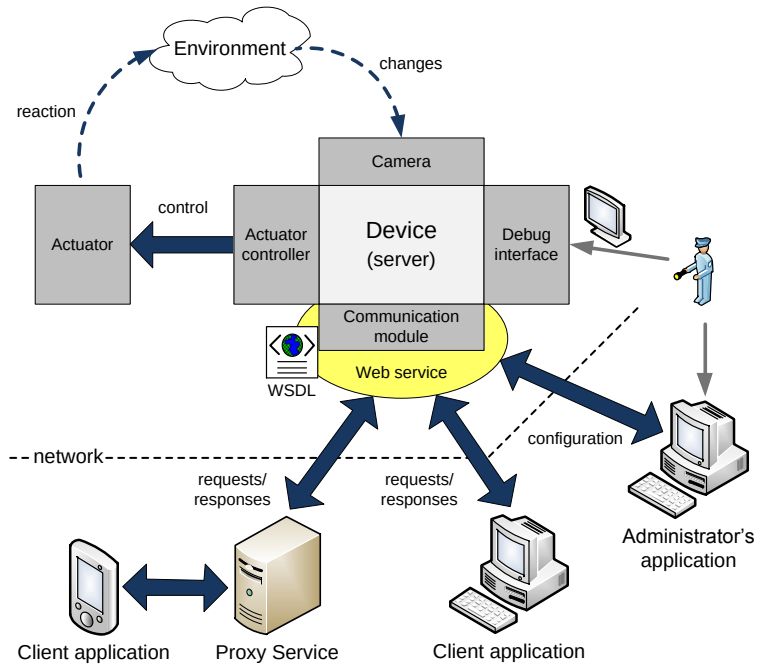

Figure 7. Architecture of the smart camera service

A wide variety of third-party programming environments required to create a full set of firmware for the smart camera service induced the need to create an additional tool to facilitate seamless FPGA configuration and server-side software design. It is an application that runs on a PC and manages such tasks as generating server code stubs, generating hardware accelerator module's HDL code from a C-like specification, compiling FPGA configuration, generating code for the communication module, and uploading the compilers' output to each programmable component.

The software templates generated during this process are intended to be the basic reference point for the software developers willing to build new web services based on our smart camera device.

Two example applications involving the above solution are proposed. The first application is a customizable intruder detection service that can capture and track all non-accidental motion in the recorded scene. The clients can interact with the server in several ways via the WS interface. First, a preview of the scene being monitored can be obtained so as to give the user an idea of what is in the camera's field of view. Based on this preview image the client can specify the relative coordinates of the region of interest (RoI) within which motion is to be sought. Moving objects are detected and tracked accordingly and a history of motion (storing the identity, time, and relative position of an object at each time step) is maintained. The server can be queried for this history within a user-specified time window. Additionally, the client can instruct the server to activate an alarm in response to intruder detection.

The second application uses a face recognition service for controlling access to a restricted area. In this scenario the smart camera's server can recognize a face by comparing it to the prototype images stored in its internal memory, log the presence of a recently-identified individual, and allow or deny access to the restricted area by activating an appropriate actuator, e.g. door lock. The algorithmical core is based on parallelized Kernel Regression Trees [35] which are efficiently implemented as a hardware accelerator in the FPGA logic. A separate PC-based application has been created to train the face classifier from a number of user-provided static images and pack its parameters into a batch for upload into the device, together with prototype face images.

The FPGA-based implementation of the Kernel Regression Tree algorithm has similar performance to a traditional PC-based implementation in terms of algorithm response time. For example, with 15 parallel recognition processes, the classification time is $\approx 0.33\,\mathrm{s}$ for both cases. However the FPGA-based version of the service is clocked with 100 MHz signal and requires less than 4 W during normal operation. In contrast, the PC-based version allows obtaining similar results using a laptop computer with dual-core 2.9 GHz processor. For detailed test results we refer to [36].

It should be noted that under a conventional design scheme, such as SODA [9] where the desired functionality is defined based on the external service's interface, achieving such a high level of autonomy and complexity of the aforementioned applications would be difficult. It is due to the fact that for such type of applications the service's logic has to be composed from custom-made, task-specific functional modules and (for maximum performance and minimum response time) as close to hardware as possible.

3.3 Intelligent Sensor Networks

Wireless Sensor Networks represent a modern technology that can be used to collect a wide range of data about the surrounding environment [38]. For instance, they can measure and visualize the distribution of temperature, humidity, or the level

of pollution over a given area. In this context they are often used in the industry. Recently, sensor networks have also found their way into business process monitoring and optimization. Creating services that use WSNs requires a slightly different approach from the solutions presented above, but it is still based upon the generic design schema proposed in Figure 2. This approach is illustrated in Figure 8.

In the World of Things realm we consider essentially similar types of devices and their hardware adaptations as that presented earlier when discussing control devices (see Section 3.1). However, this adaptation must take specific properties and limitations of WSN nodes into account. Moreover, since environmental parameters, such as temperature, atmospheric pressure or oxygen concentration, are measured as analogue and non-electrical values, the hardware adaptation process must account for conversion of such signals directly or indirectly into digital values.

When performing hardware adaptation of a WSN node, it should be kept in mind that off-the-shelf sensor network nodes are usually very small devices with severe hardware limitations. Therefore, a situation in which their hardware and software capabilities appear insufficient to meet the assumed requirements is very likely.

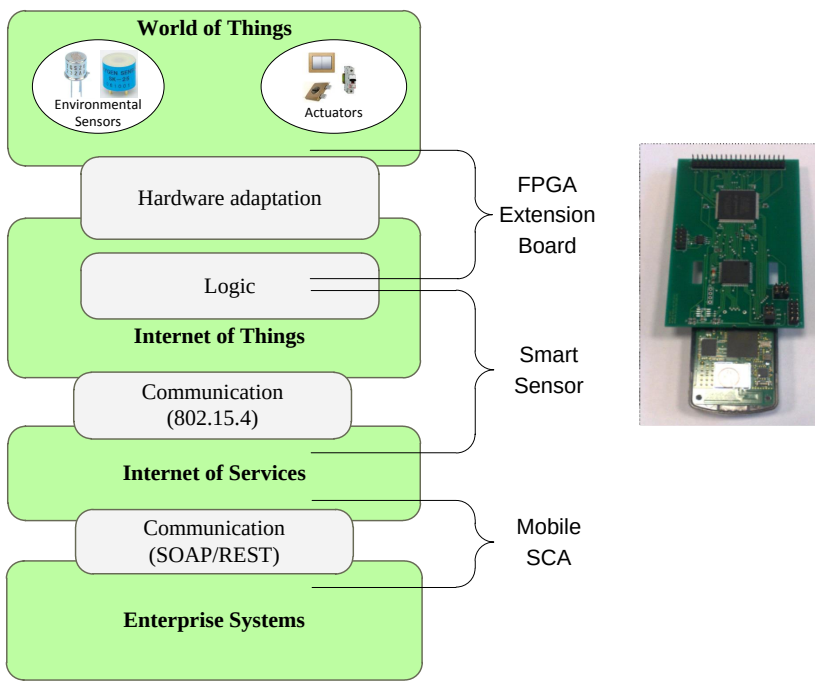

Figure 8. The design schema for a sensor network device (left) and its hardware realization (right)

Regardless of whether the role of a WSN node is to influence the environment or acquire data about it through appropriate detectors, both of these features can be made available to other network nodes upon implementation of the appropriate logic. The server created in this way has an area of operation limited only to its own sensor network. Sensor networks' operation principles and the hardware limitations of WSN nodes do not allow direct access to the sensor network services from the level of enterprise systems. This access is realized through the Mobile Service Component Architecture (SCA) mechanism, described later on in this section.

As mentioned earlier, off-the-shelf sensor network nodes are usually very small devices with severe hardware limitations (e.g. limited computational power, limited range of supported sensors, small number of input/output lines, etc.). Due to these limitations it is not possible to implement complex tasks within a single node and share the abilities of that node or the entire sensor network as a set of services. Obviously, it is possible to build each sensor node of this kind from scratch, according to the requirements of the target application. However, this would be a very difficult and time-consuming task. Instead, we propose an alternative strategy: building flexible and reconfigurable extensions to existing sensor nodes. We use Sun Small Programmable Object Technology (Sun SPOT) devices in most of our applications. The SPOT devices contain a microcontroller with ARM920T core running at 180 MHz. They have an on-board 2.4 GHz IEEE 802.15.4 radio interface. We have decided to design an extension board for that particular type of sensor nodes. For maximum flexibility the board is equipped with an FPGA chip (part number EP3C25 from Altera). The block diagram of the extension board and its prototype (connected to a SPOT device) are shown in Figure 9.

Communication between the SPOT device and the FPGA chip can be carried out using one of the standard interfaces: two-wire interface (TWI), serial peripheral interface (SPI), universal synchronous-asynchronous receiver-transmitter (USART), or by an 8-bit bi-directional parallel bus. The method of communication depends on the current needs of the sensor and the extension board. In addition, the FPGA chip can provide 36 configurable user input-output (IO) lines and numerous reconfigurable resources within the chip itself. In a simple scenario the FPGA can be configured to grant the SPOT device direct access to the IO lines. In a more advanced scenario, the logic inside the FPGA chip can be used to build a sophisticated interface for different kinds of sensors, or to implement an algorithm for preliminary processing of the data collected by the devices connected through these IO lines.

The extension board also allows the SPOT device to reconfigure the FPGA. The new configuration can be stored either temporarily or permanently. In the former case it persists until the FPGA is powered down or the upload of a new configuration has started. In the latter case, it is stored in the EPCS16 flash memory and loaded into the FPGA automatically upon power-up. The configuration file for the FPGA chip is prepared using the Quartus II software available from ALTERA. This step should be done by a person with the necessary FPGA design experience. Proper configuration files for the FPGA chip have been prepared for

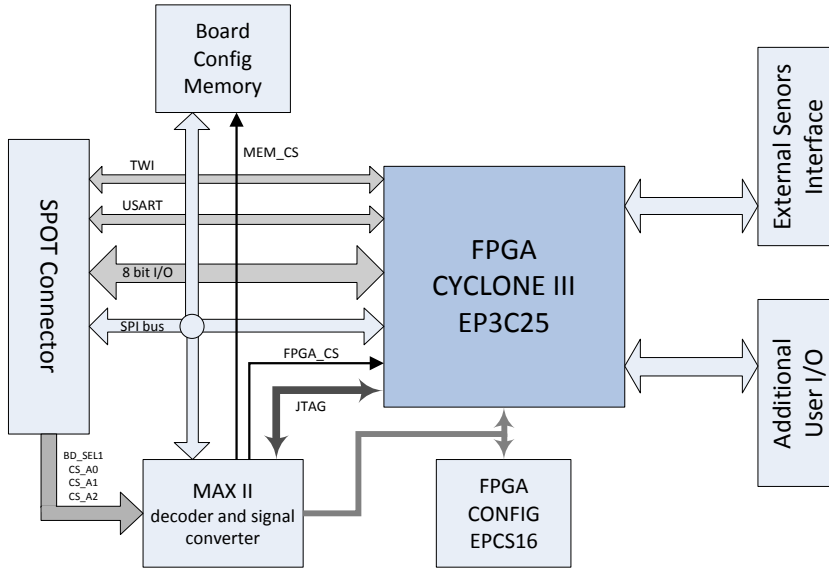

Figure 9. Diagram of a SPOT expansion board

several simple applications, e.g. when the board is used to expose 36 digital I/O lines. It is possible to share FPGA chip reconfiguration as one of the available services.

To build services which utilize sensor networks, a specific approach is needed for three main reasons: the aforementioned limited computational power of each single sensor node, severe energy consumption restrictions, and a potentially large number of devices providing a single service, as well as the dynamic properties of their connections. To address this issue, the Mobile SCA [37] solution is proposed. It is a middleware layer between SOA services and the low-level communication protocols used by the mobile devices. From the enterprise systems integration perspective the proposed system can provide direct access to each component from the web service layer or add some business logic and expose a higher-level interface. Another benefit of using mobile SCA is availability to add fault tolerance to the system. It is possible by using redundant hardware nodes to implement the service functionality. For further details, the reader should refer to the above article.

A prototype implementation of the proposed architecture has been done using Sun SPOT devices. To show the flexibility of this architecture, we also provide a simple test implementation for Sentilla Perk devices. Both device types are able to run Java code which allows rapid development and deployment of parts of the higher-level business logic that are to be executed by the devices themselves. Although the created tools are Java-based, our architecture is generic and independent of the underlying technology.

4 SOFTWARE INTEGRATION ASPECTS

A critical aspect of the research presented in this paper is the integration of different kinds of hardware services (using a variety of devices, such as smart actuators, mobile robots, smart cameras and smart sensors), as well as providing easy access to them for the potential users. One of the well-established methods of integrating reusable services is via an Enterprise Service Bus. The main advantages of this approach include standardization, scalability, reliability, mediation and manageability. The ESB may also act as a proxy for applications that do not expose a standardized service interface. Most existing Java implementations of the ESB are based on the OSGi service platform that allows a reduced complexity, reuse of components, easy deployment and dynamic component updates.

For the above reason we have decided to handle software integration through the exposition of hardware services using the Proxy Service – a service engine implemented in Java (Figure 10). The Proxy Service can be deployed within an ESB or OSGi container, or exposed as a web service reachable via the SOAP/REST communication standards. In both cases its interface can be specified in the WSDL.

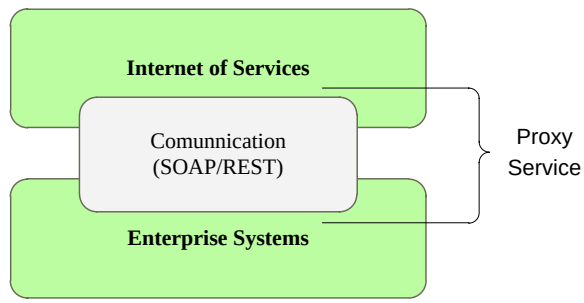

Figure 10. Software integration through Proxy Service

The WSDL files containing a functional description of hardware services should be registered in a common repository to enable discovery. For this purpose the Universal Description, Discovery and Integration (UDDI) registry can be used. It is also possible to develop a custom WSDL repository containing information about the shared services. In this case, a description of each service is published using WSDL in a Common Service Description Repository (CSDR). The user intending to interact with a given service browses the repository in search for the service's details, such as its invocation point and a signature of the operations supported. Upon finding them, the client application can connect to the hardware service directly [39].

Following the above schema, each of our hardware services is described by an appropriate WSDL file. For each such service it is possible to generate one corresponding Proxy Service with methods identical to those of the original service. All operations on a Proxy Service are delegated through standard WS requests (SOAP) to the server appropriate for the target device.

The operations of our hardware services are accessible through the Proxy Service mechanism thanks to mediation between the SOAP binding component and the Remote Service for OSGi (R-OSGi) [40] binding component within the Enterprise Service Bus. The use of such ESB implementations as Fuse ESB 4.3 or Apache Felix (installed on mobile devices supporting the Android system) enables direct access to these services through the OSGi environment. In addition, the R-OSGi technology allows remote connections to OSGi services. In the proposed scenario this solution was adopted for Android-enabled devices and the OSGi Felix environment. The proposed concept is depicted in Figure 11.

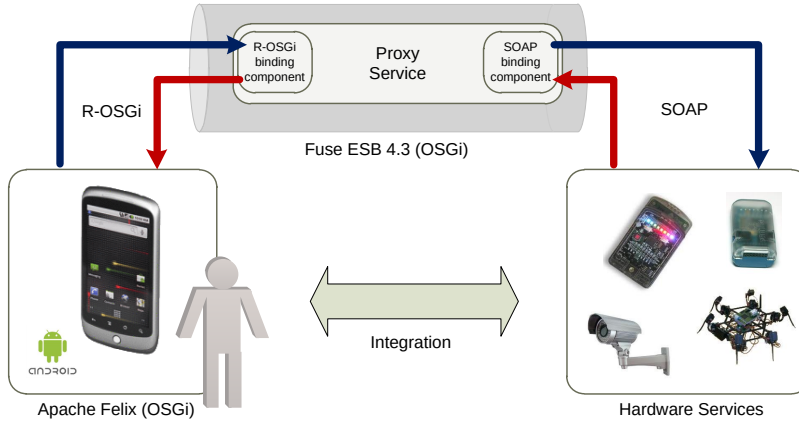

Figure 11. Integration of hardware services with the ESB/OSGi environment

In conclusion, the developed mechanism facilitates automatic generation of the service's proxy in the OSGi environment based on a standard WSDL. This proxy enables execution of the remote hardware service's operations and reception of their results within Java-based implementations of the enterprise service bus. This, in turn, allows the functionality of our smart hardware to be treated as commodity services that can be reused in heterogeneous environments, e.g. on Android-based mobile devices.

5 CASE STUDY

As a usage scenario for the developed hardware and software solutions presented in the previous sections, the smart house example is considered. This example should be treated as a stepping stone towards the broader concept of a smart city where intelligent buildings, co-managed by software and hardware services, exist alongside other public resources, such as smart road traffic infrastructure, parking lots or power stations. The presented approach differs from the body of previous work in that it introduces generic, fully customizable components that can be easily tuned for use in various application scenarios. Moreover, unlike in most of the existing solutions,

these components use open communication protocols and can be managed remotely through the web service interface.

One of the common features of the so-called smart buildings is an intelligent system for control of lighting and other electrical equipment. Ideally, it should be adaptable in a way that enables memorizing the preferences or behavioural patterns of the residents and performing appropriate self-modifications. Another possible feature is advanced control of heating and ventilation systems which requires some environmental sensors installed throughout the building. For the convenience of the residents, especially at an early stage of use, the possibility of switching between manual and remote control of the installed equipment should be preserved. Finally, the users should be able to access information about the current state of the building, as well as to control different elements of the smart infrastructure from a range of popular electronic devices, e.g. multifunction remote controllers, laptops or smartphones.

To illustrate the usefulness of the proposed intelligent hardware solutions, we present a hypothetical middle-size house. Its electrical systems are designed in a way enabling easy back-conversion into a "conventional" house installation. Almost all "smart features" of the house are realized using the SPOT-based sensor network. The architecture of the smart house system is illustrated in Figure 12.

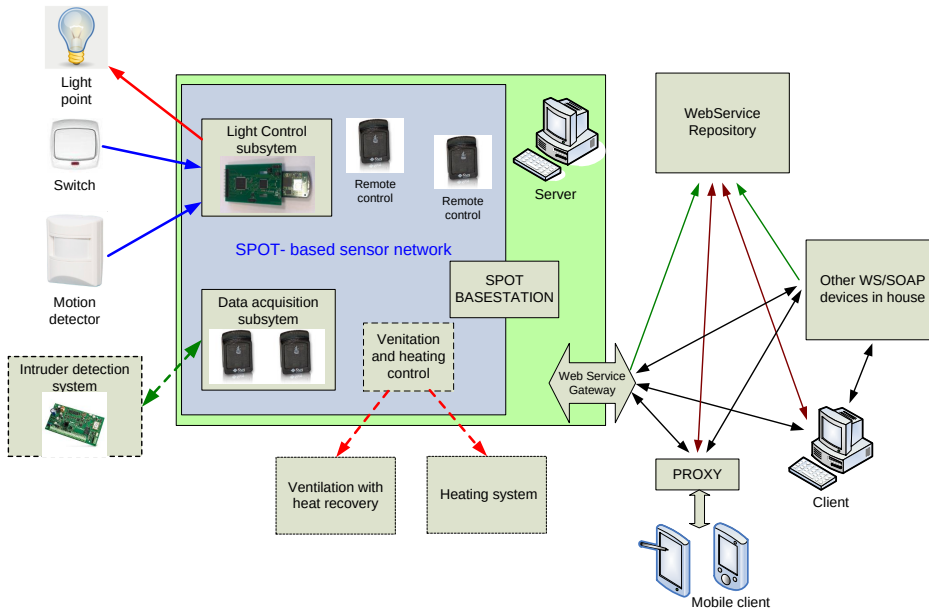

Figure 12. Logic model of the smart house

The sensor network used in the above design has been divided into several subsystems. The light control subsystem is installed in the main electrical switch board.

It is developed as a special sensor network node based on a SPOT device equipped with the aforementioned FPGA extension board. Based on the data collected from switches and other devices directly connected to them, as well as from other subsystems, it exercises control over all light fixtures in the house.

The data acquisition subsystem is a set of SPOT devices distributed throughout the house. Their task is to collect environmental data from each room. To enable this, the SPOT device can be additionally equipped with the appropriate external sensors. The acquired data can be used to control devices belonging to other subsystems (e.g. lighting, heating or ventilation). To improve diversity of the collected data, one of the SPOT devices in the data acquisition subsystem can be connected to a professional intruder detection system. Data collected by the acquisition subsystem can also be used to control heating and ventilation in the house. Another possible application for the nodes in that subsystem is to serve as gateways between the remote controllers and the rest of the system. The above mentioned remote controllers are simple wireless control devices, built on top of the SPOTs for the convenience of the household. Depending on their location they can provide access to different basic functions of the house (e.g. light control or room temperature monitoring).

The SPOT-based sensor network itself cannot expose web services. To enable this feature, the Mobile SCA middleware, described earlier in this paper, is employed. The web service gateway (see Figure 12) creates a connection point between the smart house and the outside world, i.e. it simply exposes the functionality of the smart house as a set of web services. These services are registered in the web service repository and can be accessed by other devices in the house directly or through the proxy service, the latter being dedicated for mobile clients. In a future development a trust model of accessing services can be introduced. A way of implementing those features is described in [41].

In specific situations the HSOA board can augment the infrastructure of a smart building. It could be used as a local, independent hardware controller node, accessible over the network trough a web service interface. In addition, the smart camera service could help improve the alarm system or contribute to energy savings by detecting when particular rooms are not occupied and switching off the lights accordingly.

In Figure 13 we present a simple example of interaction between the selected element of the intelligent house's infrastructure and the end user equipped with a smartphone. Let us assume that the tenant would like to remotely control the anti-burglar lock in the main door or in the garage gate. The important aspect here is to avoid changing the mechanical structure of the lock (which usually holds special safety certificates) and to maintain the possibility of a regular, manual use. Therefore, a special hardware extension based on a servomotor is implemented. As a result, the user or another service can remotely instruct the hardware control server to open, close or check the state of the lock. This server, together with the aforementioned servomotor extension, constitutes the hardware web service.

Any client who possesses a reference to the service can download the appropriate WSDL interface description so as to create custom control methods for the anti-burglar lock. Alternatively, based on this WSDL file, an OSGi service can rapidly be generated using the Proxy Service mechanism. This service can then provide the equivalent anti-burglar lock control functionality for a mobile Android-based device.

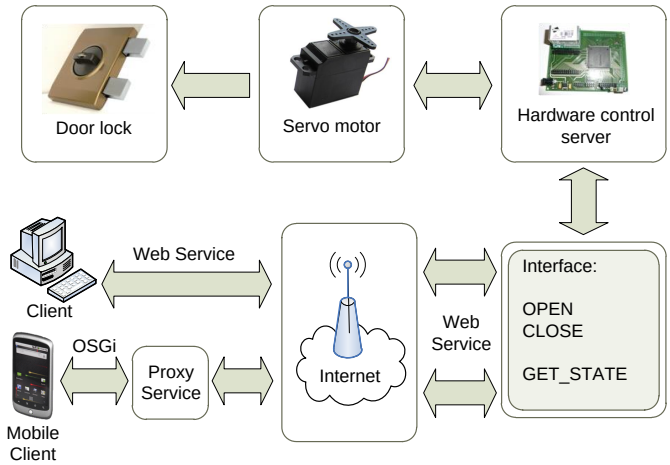

Figure 13. Remote control of an anti-burglar lock

6 FURTHER RESEARCH

The presented web services are complex embedded devices which consist of multiple hardware and software modules. Like many large systems, they can be further developed and upgraded. Security is the first issue that arises when considering practical implementations of the presented system. Typical solutions can be used in this field, e.g. Virtual Private Networks (VPN) with secure tunneling protocols. Further improvements would require an implementation of Hypertext Transfer Protocol Secure (HTTPS) which is possible yet very difficult for small embedded systems. Another, much simpler solution that can be used to provide a good degree of data security is a utilization of a well-known block encryption algorithm such as Advanced Encryption Standard (AES) for encrypting the most sensitive data payloads.

The experimental smart camera service (refer to Section 3.2) is particularly interesting in the scope of further development. Its large hardware capabilities allow authors to improve the performance of the currently implemented image processing algorithms and implementing other ones.

7 CONCLUSIONS

Practical issues involved in integrating the domain of commodity objects and the technology underlying the Internet of Things are still challenging. Addressing known problems such as the heterogeneity of physical communication, security [41], diversity of data link protocols or identification schemas does not suffice to effectively explore the advantages of a global communication between physical devices and enterprise software systems. This is due to inherent differences in the levels of logical abstraction. While enterprise systems are designed according to the SOA paradigm, the control of low-level device operations is still governed by the fairly dated imperative programming model.

In this paper we show that this gap can be successfully bridged by exposing physical appliances as services accessible via high-level protocols, such as SOAP or REST. Our approach leads to a coherent architecture model where things are visible as software services in the outside world. This, in turn, facilitates integration with the existing applications built in accordance with the SOA paradigm. Unlike in the SODA model [9], the service's logic can be defined directly at a device level, which minimizes the risk of suboptimal resource utilization.

The proposed development methodology is generic and could be applied to a wide spectrum of devices and sensors. It effectively hides the heterogeneity of the World of Things, allowing flexible access to objects with constrained computational or communication resources. As a consequence, we can build systems with distributed intelligence where a significant part of the required processing can be performed locally. The net result is an increase in IoT systems' scalability which is another important advantage of the proposed solution.

Our original contribution, apart from the aforementioned generic architecture and a corresponding methodology for implementing and provisioning hardware SOA services, also includes a set of specific prototype devices for environment monitoring and actuator control. Such devices can be successfully exploited in a smart building/smart city management scenario. The solutions presented in this paper had been practically tested in an experimental set-up deployed at the Department of Computer Science of the AGH University of Science and Technology, Kraków, Poland. The experimental set-up utilized the described various multiple sensors and actuators to provide access to a "guarded" zone.

Acknowledgements

The research presented in this paper was partially supported by the European Union in the scope of the European Regional Development Fund program number POIG.01.03.01-00-008/08 and by the Polish Ministry of Science and Higher Education under AGH University of Science and Technology Grant 11.11.230.015 (statutory project).

REFERENCES

[1] Ashton, K.: That 'Internet of Things' Thing. RFID Journal, 2009.

[2] Atzori, L.—Iera, A.—Morabito, G.: The Internet of Things: A Survey. Computer Networks, Vol. 54, 2010, No. 15, pp. 2787–2805.

[3] Zhang, K.—Han, D.—Feng, H.: Research on the Complexity in Internet of Things. Proceedings of the 2010 International Conference on Advanced Intelligence and Awareness Internet (AIA 2010), Beijing, China, October 23–25, 2010, pp. 395–398.

[4] Park, J. H.—Woungang, I.—Ma, J.—Kawsar, F.: Ubiquitous Computing for Communications and Broadcasting. International Journal of Communication Systems, Vol. 25, 2012, No. 6, pp. 689–690, doi: 10.1002/dac.1386.

[5] Ning, H.—Hu, S.: Technology Classification, Industry, and Education for Future Internet of Things. International Journal of Communication Systems, Vol. 25, 2012, No. 9, pp. 1230–1241, doi: 10.1002/dac.2373.

[6] Erl, T.: Service-Oriented Architecture: Concepts, Technology, and Design. Prentice Hall, 2005.

[7] Chappell, D.: Enterprise Service Bus: Theory in Practice. O'Reilly Media, California, 2004.

[8] Rimal, B. P.—Choi, E.: A Service-Oriented Taxonomical Spectrum, Cloudy Challenges and Opportunities of Cloud Computing. International Journal of Communication Systems, Vol. 25, 2012, No. 6, pp. 796–819, doi: 10.1002/dac.1279.

[9] de Deugd, S.—Carroll, R.—Kelly, K.—Millett, B.—Ricker, J.: SODA: Service Oriented Device Architecture. IEEE Pervasive Computing, Vol. 5, 2006, No. 3, pp. 94–96.

[10] Patel, R.—Rajawat, A.—Yadav, R. N.: Remote Access of Peripherals Using Web Server on FPGA Platform. Proceedings of the 2010 International Conference on Recent Trends in Information, Telecommunication and Computing, Kochi, India, March 12–13, 2010, pp. 274–276.

[11] Maciá-Pérez, F.—Gil-Martínez-Abarca, J. A.—Ramos-Morillo, H.—Mora-Gimeno, F. J.—Marcos-Jorquera, D.—Gilart-Iglesias, V.: Wake on LAN over Internet as Web Service System on Chip. IEEE Transactions on Industrial Electronics, Vol. 58, 2011, No. 3, pp. 839–849.

[12] Schall, D.—Aiello, M.—Dustdar, S.: Web Services on Embedded Devices. International Journal of Web Information Systems, Vol. 2, 2006, No. 1, pp. 45–50.

[13] Groba, C.—Clarke, S.: Webservices on Embedded Systems – A Performance Study. Proceedings of the 1st International Workshop on the Web of Things, Mannheim, Germany, 29 March–2 April 2010, pp. 726–731.

[14] Bucci, G.—Ciancetta, F.—Fiorucci, E.—Gallo, D.—Landi, C.: A Low Cost Embedded Web Service for Measurements on Power System. Proceedings of the IEEE International Conference on Virtual Environment, Human-Computer Interface and Measurement Systems, Giardini Naxos, Italy, July 18–20, 2005, pp. 7–12.

[15] Machado, G. B.—Siqueira, F.—Mittmann, R.—Vieira e Vieira, C. A.: Embedded System Integration Using Web Services. Proceedings of the International

Conference on Networking, International Conference on Systems and International Conference on Mobile Communications and Learning Technologies, Morne, Mauritius, April 23–29, 2006, pp. 18–24.

[16] van Engelen, R. A.—Gallivan, K. A.: The gSOAP Toolkit for Web Services and Peer-to-Peer Computing Networks. Proceedings of the 2nd IEEE International Symposium on Cluster Computing and the Grid, Berlin, Germany, May 22–24, 2002, pp. 128–135.

[17] Cuenca-Asensi, S.—Ramos-Morillo, H.—Llorens-Martínez, H.—Maciá-Pérez, F.: Reconfigurable Architecture for Embedding Web Services. Proceedings of the 4th Southern Conference on Programmable Logic, San Carlos de Bariloche, Argentina, March 26–28, 2008, pp. 119–124.

[18] Chang, C. E.—Mohd-Yasin, F.—Mustapha, A. K.: An Implementation of Embedded RESTful Web Services. Proceedings of the 2009 Conference on Innovative Technologies in Intelligent Systems and Industrial Applications, Monash, Malaysia, July 25–26, 2009, pp. 45–50.

[19] Gonzalez, I.—Sanchez-Pastor, J.—Hernandez-Ardieta, J. L.—Gomez-Arribas, F. J.—Martinez, J.: Using Reconfigurable Hardware Through Web Services in Distributed Applications. Proceedings of the 14th International Conference on Field Programmable Logic and Applications, Leuven, Belgium, 30 August–1 September 2004, Lecture Notes in Computer Science, Vol. 3203, 2004, pp. 1110–1112.

[20] Gent, G. J.—Smith, S. R.—Haviland, R. L.: An FPGA-Based Custom Coprocessor for Automatic Image Segmentation Applications. Proceedings of the IEEE Workshop on FPGAs for Custom Computing Machines, 1994, pp. 172–179.

[21] Baumann, D.—Tinembart, J.: Designing Mathematical Morphology Algorithms on FPGAs: An Application to Image Processing. Proceedings of the 11th International Conference on Computer Analysis of Images and Patterns, Rocquencourt, France, September 5–8, 2005, pp. 562–569.

[22] Neoh, H. S.—Hazanchuk, A.: Adaptive Edge Detection for Realtime Video Processing using FPGAs. Proceedings of the 2004 Global Signal Processing Expo and Conference, Santa Clara, USA, September 27–30, 2004, pp. 27–30.

[23] Djemal, R.—Demigny, D.—Tourki, R.: A Real-Time Image Processing with a Compact FPGA-Based Architecture. Journal of Computer Science, Vol. 1, 2005, No. 2, pp. 207–214.

[24] Kryjak,T.—Gorgoń, M.: Pipeline implementation of Peer Group Filtering in FPGA. Computing and Informatics, Vol. 31, 2012, pp. 727–741.

[25] Ben Atitallah, A.—Kadionik, P.—Masmoudi, N.—Levi, H.: FPGA Implementation of a HW/SW Platform for Multimedia Embedded Systems. Design Automation for Embedded Systems, Vol. 12, 2008, No. 4, pp. 293–311.

[26] Verdone, R.—Dardari, D.—Mazzini, G.—Conti, A.: Wireless Sensor and Actuator Networks. Technologies, Analysis and Design. Elsevier LTD, 2008.

[27] Liu, C. X.—Liu, Y.—Zhang, Z. J.—Cheng, Z. Y.: High Energy-Efficient and Privacy-Preserving Secure Data Aggregation for Wireless Sensor Networks. International Journal of Communication Systems, Vol. 26, 2013, pp. 380–394.

[28] Dargie, W. W.—Poellabauer, Ch.: Fundamentals of Wireless Sensor Networks. Theory and Practice. John Wiley & Sons LTD, United Kingdom, 2010.

[29] Delicato, F. C.—Pires, P. F.—Pirmez, L.—Batista, T.: Wireless Sensor Networks as a Service. Proceedings of 17th IEEE International Conference and Workshops on Engineering of Computer Based Systems (ECBS), Oxford, England, March 22–26, 2010, pp. 410–417.

[30] IEEE Std 1076-2008: IEEE Standard VHDL Language Reference Manual. The Institute of Electrical and Electronics Engineers, Inc., New York, USA, January 26, 2009.

[31] IEEE Std 1364-2001: IEEE Standard Verilog Hardware Description Language. The Institute of Electrical and Electronics Engineers, Inc., New York, USA, September 28, 2001.

[32] Horla, D.: Minimum Variance Adaptive Control of a Servo Drive with Unknown Structure and Parameters. Asian Journal of Control, Vol. 15, 2011, No. 1, pp. 120–131, doi: 10.1002/asjc.479.

[33] Gao, D.—Zhu, W.—Xu, X.—Chao, H. C.: A Hybrid Localization and Tracking System in Camera Sensor Networks. International Journal of Communication Systems, 2012, doi: 10.1002/dac.2492.

[34] Pellerin, D.—Thibault, S.: Practical FPGA Programming in C. Prentice Hall, 2005.

[35] Ruta, A.—Li, Y.: Learning Pairwise Image Similarities for Multi-Classification Using Kernel Regression Trees. Pattern Recognition, Vol. 45, 2012, No. 4, pp. 1396–1408.

[36] Ruta, A.—Brzoza-Woch, R.—Zieliński, K.: On Fast Development of FPGA-based SOA Services - Machine Vision Case Study. Design Automation for Embedded Systems, Vol. 16, 2012, No. 1, pp. 45–69.

[37] Bachara, P.—Zieliński, K.: Service Component Architecture Extension for Sensor Networks. Scalable Computing: Practice and Experience, Vol. 12, 2011, No. 1, pp. 121–135.

[38] Kułakowski, P.—Calle, E.—Marzo, J. L.: Performance Study of Wireless Sensor and Actuator Networks in Forest Fire Scenarios. International Journal of Communication Systems, 2012, doi: 10.1002/dac.2311.

[39] Rodriguez, J. M.—Crasso, M.—Mateos, C.—Zunino, A.: Best Practices for Describing, Consuming, and Discovering Web Services: A Comprehensive Toolset. Journal of Software: Practice and Experience, 2012, doi: 10.1002/spe.2123.

[40] Rellermeyer, J. S.—Alonso, G.—Roscoe, T.: R-OSGi: Distributed Applications Through Software Modularization. In Middleware 2007, LNCS 4834.

[41] Liu, Y.—Chen, Z.—Xia, F.—Lv, X.—Bu, F.: An Integrated Scheme Based on Service Classification in Pervasive Mobile Services. International Journal of Communication Systems, Vol. 25, 2012, No. 9, pp. 1178–1188, doi: 10.1002/dac.2330.

Paweł Bachara is a Ph.D. student in the Department of Computer Science at AGH University of Science and Technology, Krakow, Poland. His current research is focused on integrating sensor network devices into software systems built according to the SOA paradigm. His experience as a Sun Campus Ambassador has given him a deep knowledge of Java and Java Enterprise technologies.

Robert Brzoza-Woch received his M.Sc. degree in electronics and telecommunication from the AGH University of Science and Technology, Krakow, Poland, in 2009. Currently he is a Ph.D. student at the Department of Computer Science, AGH University of Science and Technology. He has a broad experience in hardware and software design for intelligent wireless sensor networks, remote condition monitoring systems and pervasive computing. During the last several years his professional activity revolved around single-chip microcontrollers with particular emphasis on FPGA hardware design using VHDL and Verilog hardware description languages. He is also an author of numerous commercial articles and books popularizing general electronics knowledge, mostly in the field of digital design and embedded systems.

Jacek Długopolski is Assistant Professor in the Department of Computer Science at AGH University of Science and Technology, Krakow, Poland. In the past, he cooperated with the Krakow University of Technology, Poland and with the Olivetti & Oracle Research Lab, Cambridge, UK. He has broad expertise in assembly languages, digital hardware systems, FPGA technology, VHDL hardware description language, embedded systems and microprocessors. His current research is focused on a hardware implementation of data acquisition, data processing and control systems algorithms using FPGA microchips and VHDL language.

Piotr Nawrocki is Assistant Professor in the Department of Computer Science at AGH University of Science and Technology, Krakow, Poland. His scientific interests focus on computer networks and mobile systems. He has participated in several EU research projects including MECCANO, 6WINIT and UniversAAL. He is the author or co-author of 17 papers in the areas of computer networks, telemedicine and mobile systems. His current research interests focus on Mobile SOA and his current responsibilities include the supervision of many master's degree candidates. He is a member of the Polish Information Processing Society (PTI).

Andrzej Ruta received his M.Sc. in computer science from the AGH University of Science and Technology, Krakow, Poland, in 2006. In 2009 he received his Ph.D. degree in machine vision from the School of Information Systems, Computing & Mathematics, Brunel University, Uxbridge, United Kingdom. Between 2009 and 2012 he held a position of Teaching Assistant and then Assistant Professor at AGH University of Science and Technology where he did research on embedded implementation of image processing services. He currently works as engineer and team leader at Samsung Poland R & D Center where he coordinates research activities in the field of machine perception, especially image, speech and biosignal processing. His research interests include intelligent data analysis, computer vision, machine learning and pattern recognition. He is an author of numerous publications in these areas.

Wojciech Zaborowski received his Ph.D. degree in computer science at the AGH University of Science and Technology, Department of Computer Science, Krakow, Poland in 2009. He also holds M.Sc. degree in electronic devices design. Currently he is Assistant Professor at the Department of Computer Science at AGH University of Science and Technology. He has a broad expertise in digital hardware systems, FPGA technology, embedded systems and microprocessor architectures. His current research focuses on sensor networks and FPGA-based systems.

Krzysztof Zieliński is the Head of the Department of Computer Science at AGH University of Science and Technology, Krakow, Poland. His research focuses on networking, mobile and wireless systems, distributed computing, and service-oriented distributed systems engineering. He is the author of over 200 papers in these topic areas. He was the Project/Task Leader of numerous EU-funded projects, e.g. PRO-ACCESS, 6WINIT, Ambient Networks. He served as an expert for the Ministry of Science and Education. He is an active member of IEEE, ACM and Polish Academy of Sciences. He served as a program committee member, chairman and organizer of several international conferences including MobiSys, ICCS, ICWS, IEEE SCC and many others.