Tomasz SZYDŁO[1], Szymon GUT[1], Bartłomiej PUTO[1]

AGH University of Science and Technology (1)

Smart Applications: Discovering and interacting with constrained resources IPv6 enabled devices

Abstract. *In the future, home appliances and ubiquitous devices will be equipped with IP communication interfaces and will expose their functionality as services. Such real world devices will be equipped with tiny microprocessors of constrained resources and use lightweight software protocols. They will be used by the smart applications installed by the user on his personal devices such as phones, tablets or TV sets. The paper presents the lightweight protocol stack for enabling service orientation and service discoverability for the constrained resources real world devices and the concept of smart application platform for personal devices. The paper presents also the prototype realization of example devices and applications that use these devices.*

Streszczenie. *W przyszłości, urządzenia domowe będą wyposażone w interfejsy umożliwiające im komunikację z wykorzystaniem protokołu IP, a ich funkcjonalność będzie eksponowana w postaci serwisów. Urządzenia będą wyposażone w mikrokontrolery o ograniczonych zasobach obliczeniowych oraz będą wykorzystywać lekkie protokoły komunikacyjne. Takie urządzenia będą mogły być wykorzystywane przez inteligentne aplikacje instalowane przez użytkowników na ich urządzeniach mobilnych. Artykuł przedstawia lekkie protokoły udostępniania oraz wykrywania serwisów dla bezprzewodowych urządzeń o bardzo ograniczonych zasobach obliczeniowych, a następnie przedstawia koncepcję platformy programistycznej dla urządzeń mobilnych wspierającą tworzenie inteligentnych aplikacji. Artykuł przedstawia również przykładowe urządzenia oraz aplikacje z nich korzystające.* ***Inteligentne aplikacje: Wykrywanie i interakcja z urządzeniami o ograniczonych zasobach wykorzystującymi protokół IPv6)***

Keywords: 6LoWPAN, IPv6, smart devices, embedded systems
Słowa kluczowe: 6LoWPAN, IPv6, urządzenia przenośne, systemy wbudowane

Introduction

Over twenty years ago, Mark Weiser presented in his article[13] vision of the users surrounded by smart electronic devices equipped with radio communication interfaces and serving to support a man in his everyday life. Since then, advances in miniaturization of electronic systems have contributed to realizing the Weiser vision. The well known microprocessor vendor, ARM Ltd., envisages that in the near future, market of processors for embedded solutions will be ten times greater than the market of processors for mobile phones, tablets and computers. It is expected that majority of real world devices will be embedded with the smallest and low cost microprocessors that run at a few MHz with integrated radio transceivers. These devices will be able to offer their functionality as services via particular APIs, enabling other devices and smart applications to interact with them dynamically using IP protocol stack.

The applications for personal devices, such as TV sets, tablets, mobile phones and others should follow the trend in the consumer electronic market and enable interaction with the ubiquitous real world devices equipped with constrained resources processors. Simple applications for controlling household devices with the usage of mobile phones is nothing new and there were plenty of such solutions, but the possibility to interfere cloud services, user data, its preferences and his real world devices opens unlimited opportunities for application designers. Combining this with the open standards for communication and well established application distribution markets, where users might search, download and install them on their personal devices is even more attractive. These smart applications should handle the dynamism of the environment, appearance and disappearance of the service enabled real world devices and their movement in the environment. Smart applications and a real world devices should compose an ecosystem, where interacting with real world devices should not differ from interacting with other application's building blocks that are specific for the particular operating system deployed on a personal device. Invoking an action on the real world device should be similar to, e.g., opening a webpage in the browser installed by the user on the mobile device or playing the particular audio file in the installed multimedia player application.

Enabling IP protocol stack and service orientation for real world devices equipped with constrained resources microcontrollers requires to handle the computational limitations on all of the seven layers of OSI ISO[16] protocols model. Interacting with service oriented real world devices requires to fulfill several requirements such as service discoverability, interpretability and observability. Services should be supplemented with communicative meta-data by which they can be effectively discovered and interpreted. Observability means that services should provide mechanisms for pushing changes of their state without the necessity of constantly pulling and thus consuming limited resources. From the user perspective, smart applications that use real world devices should not be tidily coupled to the devices installed in the homes but should provide mechanisms for self configuration. Such requirements should be fulfilled by the protocol stack chosen for enabling service orientation for real world devices.

Most of the pervasive electronic devices use wireless communication in the physical layer of protocol stack. Majority of existing technologies use 2.4GHz ISM band due to its unlicensed access. There are strong constraints on data link layer for service enabled real world devices because of power efficiency requirement of these devices. Broadly used wireless standards such as IEEE 802.11 b/g/n do not meet these requirements, so IEEE 802.15.4[2] standard has been designed that is supported by semiconductor vendors. They decided to put focus on the development of the compliant transceivers and integrating them in one device with the tiny processors. In the network layer, decision should be made what addressing scheme should be used. IPv4 does not have sufficient addressing space, while IPv6 can be used for addressing as its size is deemed enough for the foreseeable future. The header of IPv6 frame would fill the major part of energy efficient data link protocol as IEEE 802.15.4, so the enhanced frame format should be used as in 6LoWPAN[8] specification. On the transport layer, usage of TCP protocol might introduce the unnecessary overhead for the communications. On the other hand, usage of UDP protocol might result in the lack of guaranties necessary for proper working of service orientation protocols such as WS* and others based on reliable communication. Because of these limitations, the especially designed protocols such as the CoAP[11] or DPWS[1][3] might be used because it implements packet acknowledgments and retransmission in application layer, so it

might be used over UDP.

In this paper we propose protocol stack that enables IP connectivity and service orientation using open standards for real world devices that are equipped with low cost microcontrollers and radio transceivers. Then, we propose the concept of software platform for mobile devices that simplifies writing smart applications interacting with real world devices. The paper shows also the working prototype of such a system, which proves the concepts and ideas. It also shows that these technologies are mature enough to be used in the production. The prototype system encompasses real world devices equipped with 8-bit microprocessors running under Contiki OS, integrated router for IPv6/6LoWPAN bridging and a smart application for Android OS.

Motivating Scenario

Bob buys a decorative desk lamp, that can change its color of light according to the set RGB values. The lamp is equipped with a wireless interface and exposes its functionality as a service. Lamp manufacturers have decided to unify the control interface of their devices. It is in their interest, because it enables possibility to use their devices in different applications provided by third parties.

Bob searches the applications available on the Internet and installs the application on his smart TV that based on the weather forecast for the next day, changes the light color of the chosen lamp. The application, after downloading and installing switches to configuration mode.

1. *Bob selects the city, for which he wants to get the weather forecast.*
2. *Application finds the available lamps in the house and shows a list of devices along with meta-data containing information such as the manufacturer, model and serial number.*
3. *Bob chooses the particular lamp that he would like to be used by the smart application. The application then encourages Bob to install the appropriate driver for the lamp.*

Weather forecast indicated by the application does not envisage the weather conditions properly - Bob figured out that the results are biased. Bob searched for different weather forcast applications and found one that can use the weather station installed in home to improve the weather forecast by analysis the history of differences.

Bob then buys a universal weather station and installs it in the house, and then configures the new application to use the already bought device. The new application observes the actual weather conditions gathered by the weather station and adjusts the weather forecast taken from the Internet and then changes the color of the selected RGB lamp.

Related Work

There were several works aimed at introducing service orientation and IP connectivity to the real world devices. The most straightforward solutions use well known standards such as WS* or REST to implement device API[4]. Such an approach is only possible for devices that use full TCP/IP stack over wired or wireless medium. In most cases these devices use specialized chips that provide TCP/IP protocol stack, making these solutions not suitable for usage in commodity real world devices. Because of significant overhead of mentioned solutions, The Devices Profile for Web Services (DPWS) has been introduced[1] in the literature. It defines a minimal set of implementation constraints to enable secure Web Service messaging, discovery, description, and eventing on resource-constrained devices. DPWS was mainly developed by Microsoft and some printer device manufacturers and was supported by various research projects. There was an attempt to specify how DPWS might be used over UDP protocol[3]. Web Services for Devices (WS4D) is an initiative bringing Service Oriented Architecture (SOA) and Web services technology to the application domains of industrial automation, home entertainment, automotive systems and telecommunication systems. The WS4D toolkits advance results of the ITEA project SIRENA[15] that comply to the DPWS.

Although the fact that DPWS has been designed to be used in embedded devices, its memory footprint and requirements for connectivity is far beyond the capabilities of tiny microcontrollers equipped with radio transceivers. A commonly used and well established standard which specifies the physical layer and media access control for low-rate wireless personal area networks is IEEE 802.15.4[2]. It is maintained by the IEEE 802.15 working group and is the basis for the technologies such as ZigBee, ISA100.11a and WirelessHART, each of which further extends the standard by developing the upper layers which are not defined in the specification. The major semiconductor manufacturers like Atmel, STM and Texas Instruments have in their portfolio radio transceiver chips that are compliant to this standard, including chips like ATmega128RFA1 or STM32W108 that combine radio transceivers and a low power microcontrollers in the single enclosure. Recently, significant studies have been conducted to enable the convergence of sensor networks with the IP world and the connectivity of smart objects to the Internet. The IETF Working Group IPv6 over Low power Wireless Personal Area Networks (6LoWPAN) proposed an RFC 4944[8] to enable IPv6 packets to be carried over IEEE 802.15.4. Eventually, the IETF Working Group Routing over Low power and Lossy networks (ROLL) designed a routing protocol named IPv6 Routing Protocol for Low power and Lossy Networks (RPL)[14]. RPL was proposed because none of the existing known protocols such as AODV, OLSR or OSPF met the specific requirements of Low power and Lossy Networks (LLN)[12]. The RPL protocol targets large scale wireless sensor networks (WSN) and supports a variety of applications e.g., industrial, urban, home and buildings automation or smart grid.

Constrained Application Protocol (CoAP)[11] is a specialized web transfer protocol for use with constrained networks and nodes for machine-to-machine applications such as smart energy and building automation. It was assumed that constrained nodes often have 8-bit microcontrollers with small amounts of ROM and RAM, while networks such as 6LoWPAN often have high packet error rates and a typical throughput of 10s of kbit/s. CoAP provides a method/response interaction model between application endpoints, supports built-in resource discovery, and includes key web concepts such as URIs and content-types. CoAP easily translates to HTTP for integration with the web while meeting specialized requirements such as multicast support, very low overhead and simplicity for constrained environments. CoAP protocol provides also mechanisms for observing state changes without the necessity of constant pulling the device by the client.

Discovering and interacting with real world devices

In the motivating scenario, Bob does not deal with low level configuration issues because all the devices installed at home where discovered and then used by the smart application installed on his personal device. From the smart applica-

tion point of view, interacting with real world device should not differ from interacting with other application's building block, specific for the device operating system. Invoking an action on the real world device should be similar as e.g. opening a webpage in the webbrowser installed by the user on the mobile device.

Device should advertise its URL of provided service, type of the device and a meta-data that might be used by the user to distinguish particular one among available ones. These data might be represented as a number of key-value pairs as presented in Fig.1.

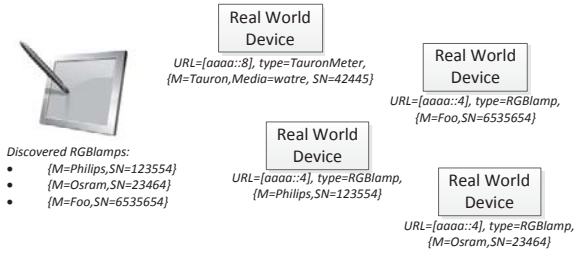

Fig. 1. Example real world devices and their description

In our concept of smart applications platform as presented in Fig.2, devices of the same type are represented as the single service in the OS for mobile devices. The application uses these devices by invoking methods on service representing the type of the device by providing a meta-data that uniquely represents the device as a context.

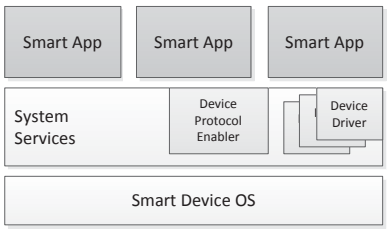

Fig. 2. Smart applications

In our concept, devices are discovered by the *Device Protocol Enabler* that uses the selected discovery protocol and provides the list of devices and accompanying meta data. It can also download, based on the device type, the proper *Device Driver* from the application market as a typical applications and be automatically installed. Presented concept might be adapted to the existing OS for mobile devices such as Android OS, Windows Mobile, iOS and others.

The next sections discuss the protocol stack and technologies used for implementation of the prototype system.

Protocol Stack for Real World Devices

Analysis of the work done in the area of smart electronic devices led to the conclusion that the most promising approach to enabling service orientation for real world devices seems to be the usage of IP protocol stack on small embedded microprocessors equipped with radio transceivers compliant to IEEE 802.15.4 standard. The proposed protocol stack for embedded devices is presented in Fig.3. There are a few operating systems for small embedded devices that provide IP connectivity. TinyOS[7] is a free and open source component-based operating system written in the nesC programming language as a set of cooperating tasks and processes. It contains the BLIP library, that is an implementation of a number of IP-based protocols. The second, most promising operating system is Contiki that is an open source implementation for networked, memory-constrained systems with a particular focus on low-power wireless devices. Examples of where Contiki is used include street lighting systems, sound monitoring for smart cities, radiation monitoring systems, and alarm systems.

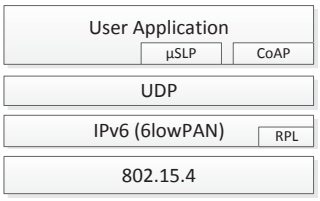

Fig. 3. IP protocol stack for embedded devices

Contiki provides three network mechanisms: the uIP TCP/IP stack [5] which provides IPv4 and IPv6 networking and the Rime stack, which is a set of custom lightweight networking protocols designed specifically for low-power wireless networks. The IPv6 stack was contributed by Cisco and was at the time of release, the smallest IPv6 stack to receive the IPv6 Ready certification. The IPv6 stack also contains the RPL routing protocol for low-power lossy IPv6 networks and the 6LoWPAN header compression and adaptation layer for IEEE 802.15.4 links. Contiki contains also the implementation of CoAP protocol called Erbium that implements `coap-8` specification.

Service discoverability

Service discovery protocols are network protocols which allow automatic detection of services offered by the devices on a computer network. We have analysed several discoverability mechanisms designed for IP enabled devices including Service Location Protocol (SLP)[6], Jini, Apple Bonjour and WS-Discovery in DPWS[1]. Finally we have decided to implement a simplified version of SLP, which has been called μSLP.

The SLP is a service discovery protocol that allows computers and other devices to find services in a local area network without prior configuration. Service Location Protocol may work in distributed manner, where Service Agents (SA) that provides services and Service Users (SU) that are interested in services are discovered using multicast communication. Another possibility requires usage of the additional node acting as Directory Agent (DA) being the service registry and changing communication to client-server model. For devices with constrained resources it is preferred option for discovering services. Therefore, even in this case, DA is discovered using multicast communication. We think that resource constrained devices should not implement multicast communication because it consumes too much resources.

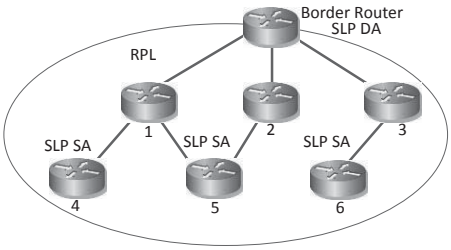

Fig. 4. Coexistence of μSLP and RPL

We have analysed the network layer protocol used in the IP communication for sensor networks and this led us to the conclusion that SLP DA could be installed on the Border Router for RPL routing protocol. This concept is presented in Fig. 4. μSLP uses the same frame format as typical SLP but it does not support multicast discovery. Therefore it can not discover SLP Directory Agent, but assuming that it is in-

stalled on the RPL border router, wireless device can send registration packets without prior discovering of SLP DA, because in RPL protocol border router is known to all of the nodes. The sequence diagram representing interaction between SLP Directory Agent, SLP User Agent on the smart device and μSLP Service Agent on real world device is presented in Fig.5.

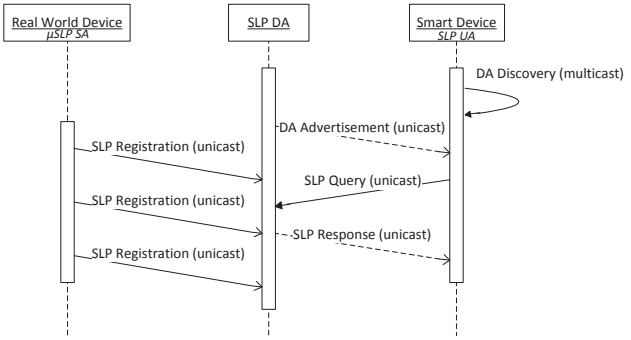

Fig. 5. Service registration and discovery using SLP and μSLP protocols

Because of the fact that Contiki OS does not provide implementation of any discoverability mechanisms, we have decided to implement μSLP as the Contiki process contained in a library that might be used in any application.

Smart Applications

The modern operating systems are designed in the way that might be executed on different hardware platforms. Kernels of these systems are designed in the modular and generic way, that drivers for hardware peripherals might be dynamically loaded during runtime or linked during compilation time. Applications and drivers for operating systems might be provided by the communities and published on the Internet for future usage by interested users. There are several operating systems for personal devices such as Windows Mobile, iOS, BlackBerry but Android OS is expected to be the most widely used. While it is designed primarily for smart phones and tablets, the open and customizable nature of the operating system allows it to be used on other electronics, including laptops and netbooks, smartbooks and smart TVs (Google TV).

Realization of the prototype smart application platform for Android OS has been performed in the two stages. First was regarded to verify IPv6 and multicast support by Android OS, while the second stage was aimed to design architecture of smart application platform according to the architecture of the operating system.

Android is a Linux-based operating system and allows IPv6 communication what has been tested on Android 2.3 and 4.0+. Android allows also to send and receive UDP multicasts, so it was possible to use SLP protocol library in the applications. `LiveTribe` library has been used for SLP discovery and `Californium` for accessing services using CoAP protocol.

Device Protocol Enabler has been implemented as separate application that registers service in Android OS and was accessible via IPC communication after binding to the service by sending Android intent:

`org.smartapps.enabler.coap`

It has been decided to use Play Store for uploading device drivers and assigning to them package names as follows:

`org.smartapps.devdrv.[type]`

where `type` is the type used as one of the property in SLP to describe real world device as presented in Fig.1. After installation, device driver registers itself as a service in the Android OS and was accessible via IPC communication in the similar way as *Device Protocol Enabler*.

Smart application written for Android OS can discover devices of appropriate type by binding to the *Device Protocol Enabler* service and invoking discovery method. As a result, list of matching devices will be returned. Application then might use this service to download appropriate *Device Driver* for the selected device type. Since that time, application can use directly the *Device Driver* service binding to it and invoke methods from the well known interface published by the manufacturers or alliances providing in each method as a parameter proper context containing meta data that represents selected device instance.

Network Connectivity

Usage of IP protocol stack for both personal and real world devices enables their interconnectivity as presented in Fig.6. Although the fact, that these devices use the IP protocol, they use different medium access control protocols. IPv6 requires the maximum transmission unit (MTU) to be at least 1280 Bytes, which is met by the IEEE 802.3 and IEEE 802.11 standards used by the personal devices but not by the real world devices using IEEE 802.15.4 where the packet size is 127 Bytes. For these networks, 6lowPAN specification has been defined by RFC4944[8] where adaptation layer responsible for IPv6 packet header compression and reassembling has been defined.

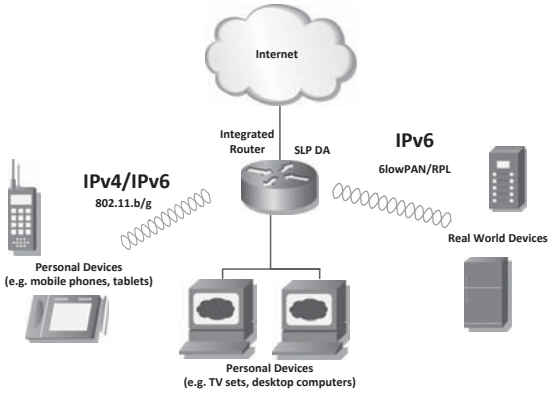

Fig. 6. Example network topology used in the use case scenarios

Due to unavailability of network equipment equipped with the 6LoWPAN interfaces, the device enabling that communication protocol and integrating other compliant with IEEE 802 standards has been developed. It has been decided to use one of the popular domestic routers based on System on Chip (SoC) design integrating a processor compatible with the MIPS architecture. Such devices use Linux operating system and provide several communication interfaces including network interfaces, USB and GPIO. The router has been extended by additional circuit board with radio transceiver and microprocessor running Contiki OS and acting as an RPL border router as depicted in Fig.7. This additional board has been connected via USB port acting as a virtual serial interface, and represented in the Linux as `tun0` interface. For transmitting IP packets over serial interface, SLIP[10] protocol has been used.

In the prototype solution, TP-Link 1043ND router has been used due to its available memory and unproblematic installation of OpenWRT[9] Linux distribution. Further, it has been decided to use additional circuit board based on ATmega128RFA1 processor running Contiki 2.6.

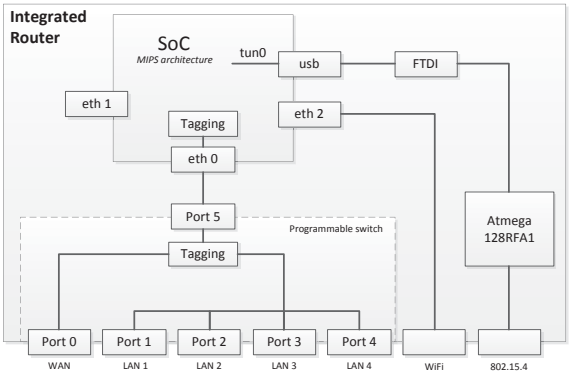

Fig. 7. Internal architecture of the Integrated Router

Because of the fact that integrated router acts as a RPL Border Router, it was necessary to deploy an SLP DA daemon. For this purpose it has been necessary to compile `OpenSLP 2.0` for the MIPS architecture, because the previous stable version does not support IPv6 addresses for advertised services.

Fig. 8. The prototype of integrated router based on TP-Link 1043ND router and Zigduino board

Example Device

For the concept verification purposes a simple device has been developed based on Zigduino circuit board presented in Fig.9, which incorporates very low cost ATmega128RFA1 microcontroller and IEEE 802.15.4 radio. Additionally, the device was equipped with the RGB LED light source and its functionality was exposed as a service. The following CoAP interface has been implemented:

- `POST /light/<color>` – turns on or off light in the corresponding color
- `GET /light/<color>` – returns whether the corresponding color of light is enabled, for `<color>` in $\{\texttt{red}, \texttt{green}, \texttt{blue}\}$

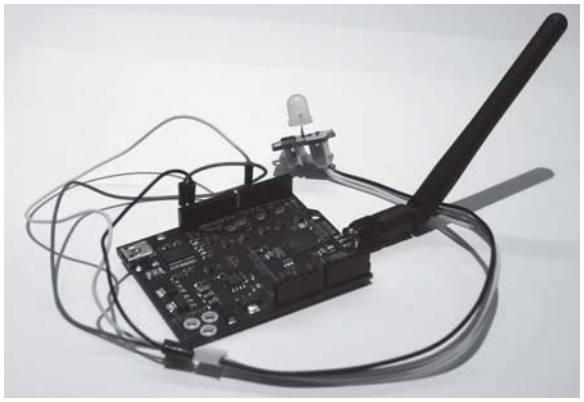

Fig. 9. Zigduino based RGB lamp

Example device uses μSLP library to enable node discovery. It registers a service with the following SLP properties:

- `type`, which equals `rgblamp`
- `SN` – serial number, used to distinguish different devices of the same type in single WSN network

According to the CoAP specification, list of available resources that were exposed by the device could be acquired by invoking GET query to relative URI `/.well-known/core/`. Nevertheless, this is not necessary, because CoAP interface is defined by device `type` parameter in SLP properties, thus device driver used by smart application is aware of the available resources.

Our implementation meets significant resource constraints as it is characterized by small memory footprint. Microcontroller flash memory usage of each code module was presented in the table.

Module	Size
Contiki OS	50kB
μSLP	1kB
CoAP	8kB
Lamp Application	8kB
Total	67kB

Example Aplication

In order to test the capabilities of the previously mentioned example device and to verify smart apps platform concept, mobile application for Android OS has been developed. To provide communication between the application and the physical device it was required to prepare an Android driver and assign it with the package name `org.smartapps.devdrv.rgblamp` as assumed in the platform concept.

The example application was called *WeatherColours* and it allowed to choose RGB lamps which could reflect weather conditions by changing their colors. It used the Yahoo! Weather API to fetch weather forecast for localization specified by the user at first application launch. It also allowed to change localization and some other parameters such as weather check frequency or displayed units at some point later by using appropriate settings menu.

As mentioned before, the user could choose which RGB lamps should reflect weather conditions, as the application provides list of automatically discovered lamps. It was assumed that the required driver has already been installed, otherwise appropriate message appeared showing some information about driver installation process. The list of avaliable devices not only contained some meta-data describing RGB lamps but it also displayed color mark that reflect current lamp status and help to simply distinguish unassigned lamps.

After selecting one of available devices the user could configure if weather status should regard current weather conditions, or today or tomorrow forecast. In such a way the user could have e.g. three lamps each of which reflects weather for different time. The application also provided menu which was dedicated to configure colors for particular weather status. Fig.10 depicts the screenshot of the example application.

Conclusion

The paper discusses the problems of interaction between service enabled real world devices and a smart applications for personal devices in the future pervasive systems. The paper proposes the protocol stack for pervasive devices that has been designed with respect to the constrained computational resources of the real world devices. Additionally,

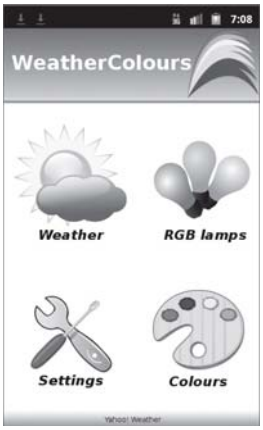

Fig. 10. Screenshot of example application

μSLP service discovery protocol and the integrated router that might be used as a residential multi purpose gateway have been designed. Finally, the concept of smart application platform that simplifies development and enables reusability of smart applications for personal devices has been presented.

The practical example of smart application presented in the paper proves that enabling software orientation in pervasive smart systems is possible with the low additional cost necessary to establish IPv6 network connectivity using available electronic modules and devices, so we might expect that they will appear in the mass market shortly revolutionizing the way of using home appliances.

Acknowledgments

The work has been supported by the POKL.04.01.01-00-367/08-00 grant and partly supported by the POIG.01.03.01-00-008/08 and MNiSW 11.11.230.015 grants.

REFERENCES

[1] OASIS Web Services Discovery and Web Services Devices Profile, 2005.
[2] IEEE Standard for Information Technology- Telecommunications and Information Exchange Between Systems- Local and Metropolitan Area Networks- Specific Requirements Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs). Technical report, 2006.
[3] OASIS SOAP-over-UDP—Version 1.1, 2009.
[4] P. Bachara, J. Dlugopolski, P. Nawrocki, A. Ruta, W. Zaborowski, and K. Zielinski. Embedded soa. In *SOA Infrastructure Tools - Concepts and Methods*. Poznan University of Economics Press, 2010.
[5] Adam Dunkels. The uIP Embedded TCP/IP Stack The uIP 1.0 Reference Manual, 2006.
[6] E. Guttman, C. Perkins, J. Veizades, and M. Day. Service location protocol, version 2, 1999.
[7] P. Levis, S. Madden, J. Polastre, R. Szewczyk, K. Whitehouse, A. Woo, D. Gay, J. Hill, M. Welsh, E. Brewer, and D. Culler. TinyOS: An Operating System for Sensor Networks Ambient Intelligence. In Werner Weber, Jan M. Rabaey, and Emile Aarts, editors, *Ambient Intelligence*, chapter 7, pages 115–148. Springer Berlin Heidelberg, Berlin/Heidelberg, 2005.
[8] G. Montenegro, N. Kushalnagar, J. Hui, and D. Culler. Transmission of IPv6 Packets over IEEE 802.15.4 Networks. RFC 4944 (Proposed Standard), September 2007.
[9] Claudio E. Palazzi, Matteo Brunati, and Marco Roccetti. An openwrt solution for future wireless homes. In *Proceedings of the 2010 IEEE International Conference on Multimedia and Expo*, pages 1701–1706, Singapore, July 2010.
[10] J.L. Romkey. Nonstandard for transmission of IP datagrams over serial lines: SLIP. RFC 1055 (Standard), June 1988.
[11] Z Shelby, C Bormann, and B Frank. Constrained Application Protocol (CoAP). *An online version is available at http://www.ietf.org/*, 13(draft-ietf-core-coap-13.txt):1–109, June 2013.
[12] Heecheol Song, Sang Hyuk Lee, Soobin Lee, and Hwang Soo Lee. 6LoWPAN-based tactical wireless sensor network architecture for remote large-scale random deployment scenarios. In *Proceedings of the 28th IEEE conference on Military communications*, MILCOM'09, pages 1044–1050, Piscataway, NJ, USA, 2009. IEEE Press.
[13] Mark Weiser. The computer for the 21st century. *Mobile Computing and Communications Review*, 3(3):3–11, 1999.
[14] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. Vasseur, and R. Alexander. RPL: IPv6 Routing Protocol for Low-Power and Lossy Networks. RFC 6550 (Proposed Standard), March 2012.
[15] Elmar Zeeb, Andreas Bobek, Hendrik Bohn, Steffen Pruter, Andre Pohl, Heiko Krumm, Ingo Luck, Frank Golatowski, and Dirk Timmermann. WS4D: SOA-Toolkits making embedded systems ready for Web Services. In *3rd International Conference on Open Source Systems, Embedded Workshop on Open Source Software and Product Lines, Limerick, Ireland*, June 2007.
[16] H. Zimmermann. OSI Reference Model-the ISO model of architecture for open systems interconnection. *IEEE Trans. Communication (USA)*, COM-28(4):425–432, April 1980. IRIA/Lab., Rocquencourt, France.

***Authors**: Ph.D. eng. Tomasz Szydło, eng. Szymon Gut, eng. Bartłomiej Puto, AGH University of Science and Technology, Faculty of Computer Science, Electronics and Telecommunications, Department of Computer Science, al. A. Mickiewicza 30 30-059 Krakow email: tomasz.szydlo@agh.edu.pl*