SMART GATEWAY FOR WIRELESS SENSOR NETWORKS

Karel Pavlata[*], **Lubomir Mraz**[**]
[*]Master Degree Programme (2), [**]Doctoral Degree Programme (1), FEEC BUT
E-mail: {xpavla05, xmrazl00}@stud.feec.vutbr.cz

Supervised by: Petr Fiedler
E-mail: fiedlerp@feec.vutbr.cz

Abstract: Today Wireless Sensor Networks provide a valuable tool for monitoring and interacting with the real world. To be able to utilize such a tool in an efficient manner we have to be able to integrate heterogeneous technologies within one complex network possibly using Internet as a backbone. For such an integration to be done effectively, we need more than simple protocol translation on Data-Link and Network layers. Abstraction on Presentation and Application layers has to be done as well, preferably using some of the well established technologies. In this paper we propose Hardware design of such a system and architectural concept of network using it.

Keywords: Wireless Sensor Network, WSN Web Integration, Web of Things, Gateway, ReST, XMPP

1 INTRODUCTION

Wireless Sensor Networks (WSN) use networked, resource constrained embedded devices to interact with its environment. Although there is possibility of self-contained deployment, usually there is a need for input (commands) and/or output (data) interactions with WSN depending on particular application scenario. Examples are Data Collection and Actuator Control services and also Service and Network Discovery. These interactions can be divided into three large classes: random access interactions (request-response model), continuous monitoring (periodic stream of data) and event-based interactions (sporadic events) which in turn can be further divided to human-machine and machine-to-machine interactions (client can be either human or computer). Interactions should be managed in a consistent way regardless of the type of a client. Although the lack of open and simple standards in this area makes it difficult, there is a proliferation of the use of technologies based on ReST[1] architectural design, which is a style of software architecture for distributed hypermedia systems (such as WWW)[3].

Important aspect of WSN is incorporation of techniques contributing to a decrease of power consumption and thus an increase in the life span of the network (devices are usually battery powered). Using techniques, such as low duty cycle operation, impose significant delays to the communication which must be dealt with in order to ensure smooth operation of the network from client's point of view. Clients have to be abstracted from peculiarities of particular WSN architecture in use. This, and the facts that many WSN platforms uses devices too constrained in resources, implies that gateways are vital part of WSN deployments. IP-based and HTTP-enabled devices are being developed and emerging, in particular 6LowPAN[2] is gaining increasing popularity. However, it is not always feasible to use this technology due to restrictions in resources. Nevertheless, these kind of devices can be supported transparently.

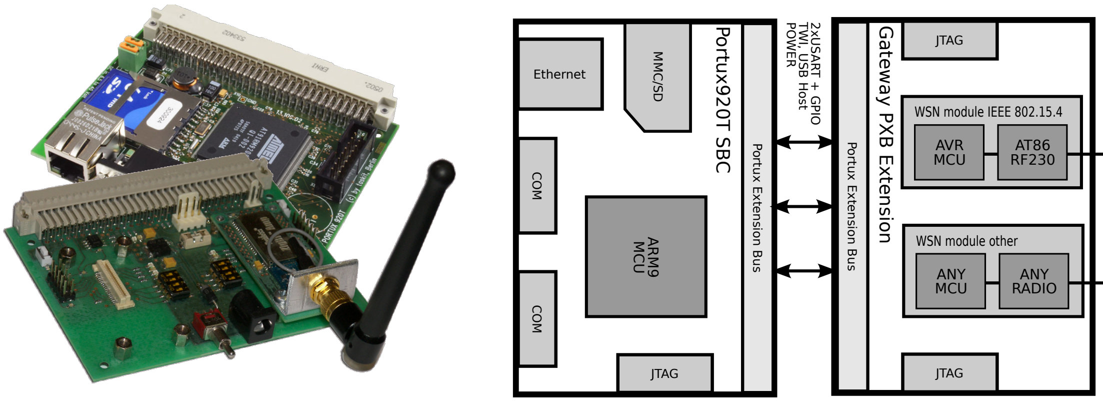
[1] ReST - Representational State Transfer
[2] 6LowPAN - IPv6 over Low power Wireless Personal Area Networks

2 HARDWARE DESIGN

Hardware of the gateway is designed with flexibility and modularity in mind. It consists of two mutually interconnected parts: Portux920 SBC[3] by taskit GmbH and Extension Board connected through PXB[4] containing two slots for various WSN modules.

Portux is an ARM9 based embedded computer capable of running Linux operating system. Its Atmel® ARM920T core is running at 180 MHz and the board is equipped with 64 MB SDRAM and 16 MB Flash memory. Further enhancement of storage space is possible via SD/MMC card slot. The board is half size of Eurocard format (possibility of 19-inch rack mount) and has a wide variety of peripherals, such as USB 2.0, Ethernet, SPI, TWI, USARTs and I/O ports, most of which are accessible through a PXB port which makes it highly modular and extensible. It features minimal power consumption, yet it is powerful enough to run intended application.

Extension board was developed as universal platform capable of carrying two WSN radio modules which are attached through board-to-board connectors. Although the interface is universal, it was specifically designed with ZigBit modules in mind. ZigBit carries IEEE 802.15.4 radio and 8-bit AVR micro-controller and is used by rest of nodes developed to be part of our network. Nevertheless, its high modular design allows it to attach any kind of WSN radio modules with just a small amount of additional work. Both parts of gateway are depicted on Fig. 1(a). In Fig. 1(b) is shown a schematic diagram of the gateway. Modules are connected through independent USART interfaces with additional GPIO pins attached which can serve whatever purpose driver decides it to be, depending on the particular module in use.



(a) Actual picture of the gateway

(b) Schematic diagram of the gateway

Figure 1: Picture and schematic diagram of gateway components

3 SOFTWARE BUILDING BLOCKS AND NETWORK ARCHITECTURE

Building blocks of the gateway's software equipment are depicted on Fig. 2(a). The application itself will run on top of GNU/Linux operating system with implemented device drivers for particular WSN modules and JamVM - lightweight Java Virtual Machine suitable for embedded devices. Java environment allows us to select from a vast amount of frameworks suitable for our purpose and achieve platform independence. The gateway maps resources of the WSN in ReSTful way, independently of the underlying WSN, using URIs[5] and stateless HTTP requests. The example of resource identification is: `http://gw1/netA/node23/resources/temperature`. All four CRUD (Create/Read/Update/Delete) operations are carried out using methods of the HTTP protocol: GET,

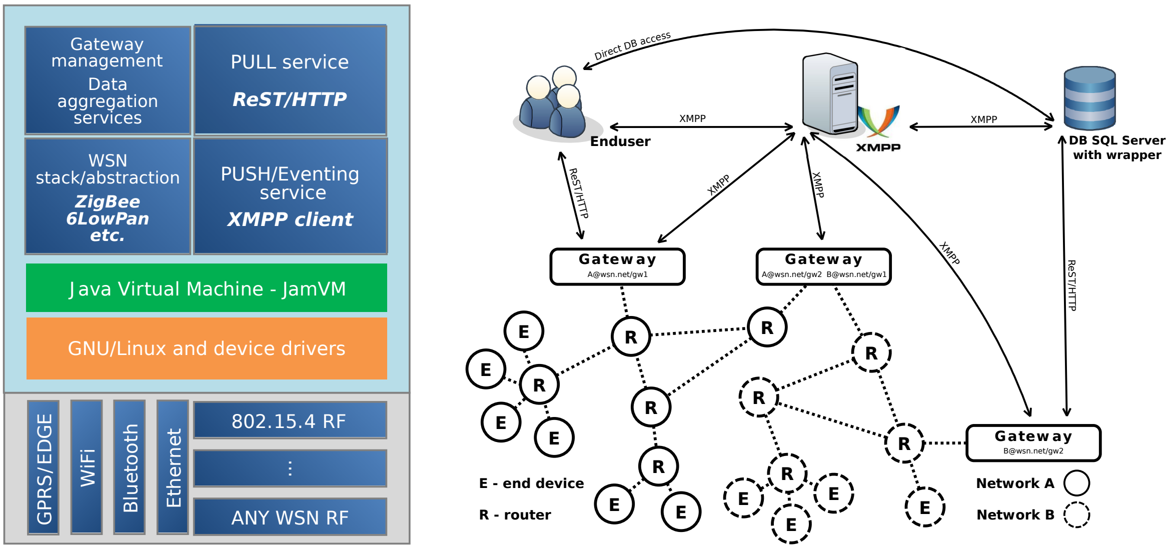
[3] SBC - Single Board Computer

[4] PXB - Portux Extension Bus

[5] URI - Uniform Resource Identifier

POST, PUT and DELETE. As the interaction is stateless, where state is actually part of the resource's URI itself, there is no need for per-client state information and the server is much less complex. Interaction with different types of clients can be handled transparently using the content negotiation mechanism of HTTP where human clients might request a different type of answer (HTML) than machines (JSON[6], XML) based on MIME types. Caching of HTTP requests also preserves bandwidth.

Since WSN is inherently asynchronous from the client's requests, XMPP (Extensible Messaging and Presence Protocol) is proposed as an eventing system. XMPP is an open, standardized, well established technology for real-time communication. Especially it's Extension Protocols *XEP-0060: Publish-Subscribe* and *XEP-0163: Personal Eventing Protocol* are useful for building highly distributed event-driven applications. Fig. 2(b) shows the proposed network architecture with individual elements forming it, their roles and interactions. The whole WSN can act as a user in terms of XMPP, whereas single gateway can act as a resource of particular user (i.e. WSN). In this way it is easy to subscribe to the particular event from WSN independently of which gateway is generating the event.



(a) Stack of the gateway

(b) Schematic diagram of the network

Figure 2: Software building blocks and schematic diagram of the network

4 CONCLUSION AND FUTURE WORK

This paper presents the design of a gateway that is capable of abstracting the nuances of a particular WSN and is capable of providing a consistent interface for various types of clients (hence called smart). Yet, there is still much work to be done in implementation of overall concept and verifying it in a large scale deployment.

REFERENCES

[1] Richardson, L., Ruby, S.: RESTful Web Services. O'Reilly Media, Sebastopol, CA, USA, 2007, ISBN-13 978-0-596-52926-0

[2] Saint-Andre, P., Smith, K., Tronçon R.: XMPP: The Definitive Guide. O'Reilly Media, Sebastopol, CA, USA, 2009, ISBN 978-0-596-52126-4

[3] Kamilaris, A.: A lightweight resource-oriented application framework for wireless sensor networks. Master's thesis, Institute of Pervasive Computing, ETH Zurich, 2009.

[6] JSON - Java Script Object Notation