SICS Technical Report T2007:04
ISSN 1100-3154
ISRN:SICS-T–2007/04-SE

Integrating Building Automation Systems and Wireless Sensor Networks

Fredrik Österlind, fros@sics.se

Erik Pramsten, epr@sics.se

Daniel Roberthson, dro@sics.se

Joakim Eriksson, joakime@sics.se

Niclas Finne, nfi@sics.se

Thiemo Voigt, thiemo@sics.se

May 2007

Abstract

Building automation systems (BAS) are used to control and improve indoor building climate at reduced costs. By integrating BAS with wireless sensor networks, the need for cabling can be removed, and both installation and operational costs significantly reduced. Furthermore, temporary BAS installations are made possible. By implementing and evaluating the open BAS standard BACnet on the ESB nodes we show that using existing standard BAS protocols is possible on resource-constrained sensor nodes.

Keywords: Building Automation Systems, Wireless Sensor Networks, BACnet, Integration

Chapter 1

Introduction

Building Automation Systems (BAS) are used to both improve the indoor climate in buildings and to reduce the operational costs. Originally, BAS mostly consisted of heating, ventilation and air-conditioning (HVAC) systems. To further increase management and reduce costs, both lighting, safety, security, and transportation supervision have been integrated into BAS. Traditionally, BAS have been used in large buildings such as schools, hospitals, and offices. For such buildings, the construction costs constitute only one seventh of the overall operational costs [12]. Hence, BAS provide a great savings potential by reducing the operational costs.

Wireless sensor networks (WSN) consist of small sensor nodes that sense the environment, perform computations, and communicate with other nodes using the on-board radio module. Typically, sensor nodes transport the measured data to a base station using multihop communication. The size of typical sensor nodes is close to a matchbox. Sensor nodes are typically powered by batteries. Since it is in general not possible, or too labor-intensive, to replace the node batteries, reducing power consumption is important in wireless senor networks. Since wireless communication is the major energy consumer [18], a particular focus has been on power-efficient communication protocols for wireless sensor networks.

Integrating WSN and BAS has a number of advantages. The main advantage is that the installation costs of WSN are lower than that of traditional BAS, since wiring is avoided. New buildings can therefore be equipped with a BAS based on wireless sensor network technology. It is also possible to extend an existing BAS in order to increase the sensor coverage. Furthermore, wireless sensors can also be installed more easily in unapproachable places such as at high heights.

In this paper, we show that it is possible to implement a standard protocol for building automation on typical resource-constrained sensor nodes found in wireless sensor networks. We report on our implementation of BACnet on ESB nodes with 2 Kb RAM and 60 Kb flash ROM. Even with such severe constraints we show it is possible to implement an important subset of the BACnet services such as the read and write property services as well as the change of value (COV) service that fits very well into the resource-constrained nature of wireless sensor networks. We also discuss the limitations and future improvements of our current implementation.

The rest of this paper is structured as follows. We give a background to wireless sensor networks and our prototype sensor node hardware in Section 2. Section 3 motivates our choice of building automation system, and Section 4 introduces the BACnet protocol in more detail. Section 5 describes our implementation of BACnet on the ESB node, and Section 6 evaluates the current system. We review related work in Section 7 and conclude the paper in Section 8.

Chapter 2

Background

System software for WSN is a challenging and active research area. The sensor node hardware constrains both the operating system and the applications. In this section, we briefly introduce both building automation systems and wireless sensor networks. We also present Contiki [8], a sensor node operating system developed at SICS, and the ESB node, our choice of sensor node hardware.

2.1 Building Automation Systems

Traditional BAS use wired technologies where sensors and actuators are connected to controllers. The first modern BAS deployed point-to-point centralized control that was later replaced by a centralized bus. Modern BAS use distributed bus control, also called field buses, and all system nodes have a built in control unit which enables them to act without centralized control.

2.2 Wireless Sensor Networks

A wireless sensor network (WSN) consists of tiny resource-limited devices called sensor nodes. Each sensor node has some sensing ability, measuring for example temperature, light, humidity, or carbon dioxide levels. Every node also has means of communication, typically by a low-power radio such as the 868 MHz TR1001. Finally, each node has processing ability by an on-board micro-controller. Typically, due to the lack of infrastructure in deployment environments, sensor nodes are also battery driven.

The sensor nodes are connected in a wireless network, where each node has a communication link to at least one other node. By collaborating, forwarding each other's messages in a multi-hop fashion, the network can cover vast areas of interest and still maintain connectivity between all nodes. A typical WSN could be connected to a BAS that monitors the temperature levels throughout a building.

Research challenges in WSNs include system software, communication protocols, power management and self-configuring networks. System software is challenging due to the often very limited memory on the nodes; the primary memory of a sensor node is in the order of a few kilobytes. Communication protocols must enable efficient and reliable multi-hop communication while also adapting to changes in the network for example due to node battery exhaustion or hardware failures. If the nodes are battery driven it is important to minimize the energy consumption in order to prolong the network lifetime. Typically this is performed by advanced sleep schedules that cause nodes to turn off their radios in a synchronized fashion.

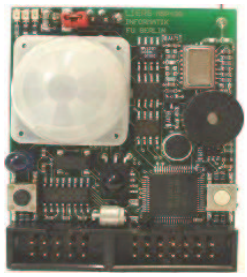

Figure 2.1. The Embedded Sensor Board

2.2.1 The ESB node

The Embedded Sensor Board (ESB), shown in Figure 2.1, is a prototype sensor board [21] developed by the Freie Universität Berlin. The ESB board consists of a Texas Instruments MSP430 [11] low power micro-controller with 60 kilobytes of Flash ROM and 2 kilobytes of RAM, an RF Monolithics TR1001 single-chip RF transceiver [20], and a set of sensors. The on-board sensors include a temperature sensor, a light sensor for the detection of visible light, a vibration sensor for motion detection and a microphone for determination of the ambient noise level. The MSP430 micro-controller is designed specifically for low-power applications and provides a set of low-power sleep modes. The micro-controller is awakened from a sleep mode by an interrupt, generated either by one of the internal timers or by an external device such as a button or the light sensor. The on-chip Flash ROM can be selectively reprogrammed by software running on the micro-controller.

The 2 Kb RAM offered by the ESB nodes is comparatively smaller than the RAM size on other sensor nodes. For example, the widely used Telos node [17] has 10 Kb of RAM memory.

2.2.2 The Contiki Operating Systems

The Contiki [8] operating system is designed specifically for resource-limited devices such as sensor nodes. Contiki uses an event-based kernel which reduces the overall system memory requirements by sharing a common stack between all processes. The operating system is implemented in C and is easily ported to new platforms. It currently supports a number of different platforms and micro-controllers such as the TI MSP430 and the Atmel AVR. Furthermore, Contiki supports dynamic loading and unloading of programs at runtime. This enables upgrading software in an already deployed WSN. Contiki also supports regular Internet networking through the uIP TCP/IP stack [7].

Chapter 3

Integrating Wireless Sensor Networks with Building Automation Systems

When applying wireless sensor networks on building automation systems it is preferable to use existing BAS standards. By using existing standards, compatibility with already deployed solutions is ensured, and future BAS protocol updates can be incorporated more easily. This section motivates integrating WSN and BAS as well as our choice of the building automation system BACnet.

3.1 Motivation

Integrating WSN and BAS has a number of advantages. The main advantage is that the installation costs of WSN are lower and the installation faster than wired systems since wiring is avoided. Furthermore, a large fraction of failures in automation networks is due to malfunctioning or disrupt cables. For example, ABB is able to give longer guarantees for the wireless variant of a product compared to the wired one due to problems with aging cables. Moreover, all products cannot be monitored using wired equipment, for example rotating arms. It is also more straightforward to equip mobile devices with wireless technology compared to wired. The decreased installation costs make it possible to increase the number of sensors and hence the spatial resolution. The increased spatial resolution allows for more fine-grained measurements and control. A further advantage is that wireless technology enables temporary measurements: a network can be set up to perform measurements during a limited time in order to measure, optimize and evaluate the effect of the optimization. We call such an effort ad hoc benchmarking.

Integrating an open standard BAS protocol with WSN, compared to designing a new BAS protocol specifically for the wireless domain, decreases company costs and product time-to-market since existing control systems can be reused.

3.2 Choice of building automation system

Before selecting a building automation system suitable for integration with wireless sensor networks, we identify the following protocol requirements:

Openness The protocol should be open, preferably standardized, free of unit licenses and widely used. This guarantees interoperability between systems from different manufacturers as well as a large existing user base.

Common communication protocols Selecting a BAS protocol that shares a common underlying communication protocol with WSN enables an easier transition to the wireless medium. Contiki supports TCP/IP via uIP, and TCP/IP is found in BAS protocols. Other protocols are usually based on a lower abstraction level and require wired connections between devices. The BAS protocol has to either directly support TCP/IP communication or tunneling in IP packets.

Small protocol memory footprint The memory footprint of the protocol should be small enough to fit on resource constrained sensor devices. If an implementation of the full protocol stack is larger than the available space, it should be possible to implement only a relevant subset of the protocol.

Node management The protocol should support node management, i.e., sensor devices joining or leaving a network should not require a reconfiguration of the entire network.

Energy-efficiency In order to decrease energy consumption and hence prolong network lifetime, a sensor node must be able to temporarily turn off its radio since the radio is the major energy consumer on the node [18]. The protocol must be able to handle temporarily offline and unavailable nodes.

We analyze a set of existing protocols including LonWorks [5], BACnet [3], OPC [22], KNX [1] and Modbus [14] to see how well they fit with the requirements we have identified. An overview of the results is shown in Table 3.2.

	LonWorks	BACnet	OPC	KNX	Modbus
Openness	YES	YES	YES	YES	YES
Native IP support	-	YES	YES	-	-
IP tunneling	YES	YES	-	YES	-
Small code size	YES	YES	YES	YES	YES
Simple node management	-	-	-	-	-
Energy efficiency	YES	YES	YES	YES	-

Table 3.1. Protocol analysis overview

Modbus' master-slave communication is not suitable for wireless sensor nodes since it forces nodes to have their radio turned on all the time which inhibits the usage of power-saving protocols.

We chose the BACnet protocol. The main reasons are that BACnet is an open, widely used standard with support for many underlying network technologies including TCP/IP. The server implementation is not very complex making it possible to implement BACnet on resource-constrained devices. Moreover, an open reference implementation is available through the BACnet homepage [4].

Chapter 4

BACnet

BACnet is a communication protocol described as “A Data Communication Protocol for Building Automation and Control Networks” [3]. The work on BACnet started 1987 when the American Society of Heating, Refrigerating and Air-conditioning Engineers (ASHRAE) was not able to find a protocol that fulfilled their criteria for a building automation and control protocol [12]. ASHRAE released the first BACnet version 1995. Later that year, BACnet became American standard. ASHRAE has maintained and developed BACnet to become a complete building automation protocol that is used in all areas of building automation, e.g. HVAC, security, fire alarm and lighting control. In 2003, BACnet became a world (ISO 16484-5) and European (EN/ISO 16484-5) standard.

Objects and devices BACnet uses an object oriented approach to describe the functionality of nodes, called devices, in a network. The objects are constructed in a uniform way to enable access to object information over the network. A BACnet object describes one certain device functionality, e.g. information about a physically attached input (analog or binary input). Every object encapsulates a collection of data elements called properties used to describe the object. The properties may be accessed and altered by other objects.

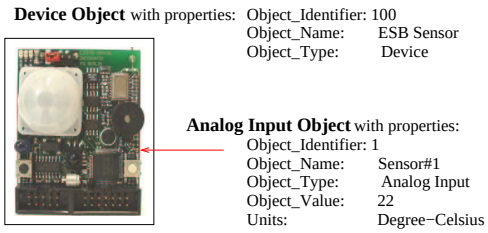

Figure 4.1. The device object is mandatory

The BACnet standard specifies a number of objects. These include objects for managing analog and binary in- and outputs, collecting trends, scheduling of operational hours and describing control loops. However, a BACnet device does not have to implement all standard objects to conform to the standard specification. Every device has to implement one special object in order to make the device available in the network. The object is called the *device object* and describes the device. An example device object is shown in Figure 4.1.

Services and Communication Model In BACnet, devices use *services* to pass information between each other. The communication model used is a traditional client-server model where each BACnet node represents a server, see Figure 4.2. Using services, BACnet devices can fetch information about other devices and objects, command devices to

perform certain actions and inform other devices that an event has occurred. The BACnet specification describes a number of standard services. Depending on in which area they are used, services have been divided into five different groups: File Access Services, Object Access Services, Alarm and Event Services, Remote Device Management Services and Virtual Terminal Services.

Similar to how all objects and properties do not need to be implemented, a device does not need to implement all services. The only mandatory service is the read property service that is used to access devices and their properties.

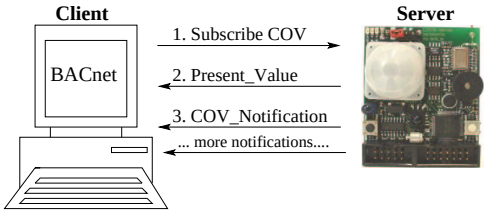

Figure 4.2. BACnet uses a client-server communication model

The Object Access Services are a collection of services to access and modify object properties, and to create and delete objects. Properties in any BACnet object, both standardized and vendor specific, can be accessed through these services. This group contains the most commonly used services: the read and write property services. There also exist more powerful variants of these services, for example to read or write multiple properties at the same time or to read properties given certain conditions. Some object types can also be created and deleted with object access services, often used for calendars and schedules.

The Alarm and Event Services are used to handle the communication regarding events. These events can be value changes for certain properties and changes in object state that meet predefined criteria, those considered serious are reported as alarms. One very interesting service for wireless sensor networks is the change of value (COV) service. Using COV, a device subscribes to receive updates of a property when it changes with a predefined amount since the last update, see Figure 4.2. This is for example useful for receiving temperature updates only when the temperature actually has changed.

Chapter 5

Implementation of BACnet on Sensor Nodes

We have implemented BACnet on the ESB nodes. The high-level architecture of the implementation is shown in Figure 5.1.

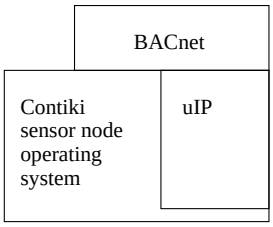

Figure 5.1. Simplified architecture.

As shown in the figure, BACnet is implemented as an application running on top of the Contiki operating system. TCP/IP communication is provided by the uIP stack [7]. The implementation follows closely the reference implementation available on the BACnet homepage. The reference implementation did not include the COV service. Since we deem this service as extremely valuable for wireless sensor networks, we have implemented this service as well.

Chapter 6

Evaluation

The key contribution of this work is to show the feasibility of applying a well-known standard protocol for building automation on wireless sensor networks. We show that the memory footprint of our BACnet implementation is small enough to fit in memory-constrained sensor nodes. We implement the important BACnet Change of Value service. Finally, we discuss possible improvements of the current implementation.

6.1 Change of Value for Wireless Sensor Networks

We claim that the Change of Value (COV) service as very interesting for wireless sensor networks, as discussed in Section 4. To demonstrate this claim we use the change of value service in a simple temperature collecting network.

We use two experimental setups. The first one was active in a home living room for 96 hours, and the second one was active for 20 hours in an office environment. We configured the change of value service to notify the client whenever the temperature had changed more than 1 degree Celsius since the last notification. Both of these environments are relatively stable in terms of temperature fluctuations, but we believe they reflect typical BAS environments well.

During the experiments we received from one sensor node 9 COV notifications in the living room and 3 notifications in the office environment. If we had not implemented the COV service directly on the sensor nodes but instead on a network gateway or other entity the temperature values must have been polled or periodically transmitted from the sensor nodes. In comparison, assuming a temperature resolution of 10 minutes would be sufficient, the sensor node would have sent 576 radio messages instead of 9 in the living room. In the office the sensor node would have sent 120 radio messages instead of 3.

Since radio communication is the most energy consuming application on sensor nodes, using the COV service can significantly improve the life-time of a battery-driven wireless building automation system.

6.2 Memory consumption

In wireless sensor networks the available memory often limits the functionality of the system. We measure the static memory consumption of a number of different BACnet configurations on our sensor nodes. See Table 6.2 for a comparison of the memory consumption of the different configurations.

The results confirm that our base system, consisting of the Contiki operating system with the uIP TCP/IP stack, is fairly small. The base system requires only around 1 KB RAM. The memory consumption increases significantly when adding the smallest possible BACnet configuration; the read property service with only one object. The table also shows that memory consumption increases with the complexity of the services and, in the case of

System configuration			Memory requirements	
Read prop. (#AI)	Write prop. (#BO)	COV (#Subscr.)	ROM (Bytes)	RAM (Bytes)
Only base Contiki & uIP system			25772	1054
Yes (1)	No	No	39934	1472
Yes (1)	Yes (1)	No	44516	1524
Yes (1)	No	Yes (1)	43924	1554
Yes (1)	No	Yes (2)	43940	1636
Yes (1)	Yes (1)	Yes (1)	48506	1606
Yes (2)	Yes (1)	Yes (1)	48510	1606

Table 6.1. Memory consumption for different BACnet configurations

COV, the number of subscribers. For example, each additional COV subscriber requires 16 bytes ROM and 82 bytes RAM.

The results in Table 6.2 do not include any multi-hop functionality since routing protocols are not part of the uIP stack. See Section 6.3 for an evaluation of a simple multi-hop protocol on the ESB nodes.

The memory consumption of our current implementation can be reduced. Since our implementation is derived from the reference implementation, which is not optimized for small memory usage, there is still room for further reduction. For example, binary values are implemented as 32-bit integers as defined by the network protocol but could be reduced to 8-bit in the implementation. Such an improvement would not only reduce the static memory usage but also the stack usage since system calls would require less stack. We further measure and discuss stack usage in Section 6.3.1.

6.3 BACnet and Multi-hop communication

Since multi-hop communication is essential for scalable sensor network solutions we measure the memory consumption of BACnet and a simple multi-hop protocol. The multi-hop protocol we use is an Ad hoc On-demand Distance Vector (AODV) protocol [16] which is part of the Contiki system. To decrease memory consumption, the protocol is limited to 8 routing entries. The rest of the system is left unchanged; the TCP/IP packets with BACnet data are just tunneled and forwarded to its final destination just above the MAC layer.

The extra memory required by the AODV routing protocol limits possible BACnet configurations. For example, we are not able to combine the COV service and the multi-hop functionality when using this, non-optimized, BACnet implementation,

Table 6.3 shows the memory requirements of the configuration with and without the routing protocol. In these measurements we use a configuration with both the read property and write property services enabled. The write property service enables a sensor node to control external devices.

Setup	ROM	RAM	Stack available
Without multi-hop	44516	1524	122
With multi-hop (AODV)	45630	1596	50

Table 6.2. Memory consumption with the AODV multi-hop protocol

6.3.1 Stack usage analysis

To analyze the actual memory requirements during run-time we use the MSPsim MSP430 emulator [10] connected to the Contiki OS sensor network simulator COOJA [15]. We measure the stack usage during different events in the system to compute the total memory usage. The results from the simulation confirm our earlier discussion about the need to

optimize the BACnet implementation for the intended low-memory platforms. The object-oriented design of BACnet and its reference implementation requires a lot of memory to handle incoming network packets. The reason for this is that each packet is structured into a set of headers and each header is handled in its own function. And each function call has several 32-bit arguments which are placed on the stack. Since these calls get deeper for each handled header and more stack is required for every such function call, the stack usage peaks during the packet handling.

The results from the stack analysis show that handling a new incoming packet requires around 400 bytes of RAM. Table 6.3 shows the **minimum** amount of available memory measured during a simulation. With the AODV protocol the lowest amount of available memory measured is only 50 bytes. This is dangerously close to stack overflow which may cause the system to malfunction. Figure 6.1 shows an overview of the stack memory usage during an incoming radio packet. The duration of the graph is 8 milliseconds.

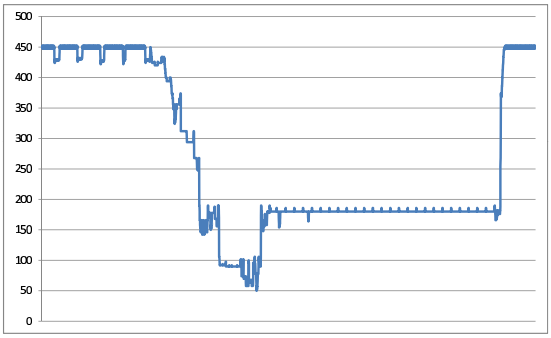

Figure 6.1. The available memory decreases significantly during packet handling (bytes)

6.3.2 Example demo network

To demonstrate and further test our BACnet implementation we setup a simple test network using the above configuration. One ESB node acts as a gateway between the WSN and a PC, and all sensor nodes use the AODV multi-hop protocol. The gateway and the ESB communicates using SLIP over a serial line. The gateway is needed only for intercepting radio communication; no other conversion is needed since the WSN already communicates via TCP/IP.

To test the write property service, we connect one of the ESB nodes to a hairdryer, hence emulating a tiny building automation system. On the PC we use the Visual Test Shell (VTS) [2], a well-known Win32 BACnet testing application. To automatically interact with the network we also developed a small network visualizer. A screenshot of the visualizer is shown in Figure 6.2.

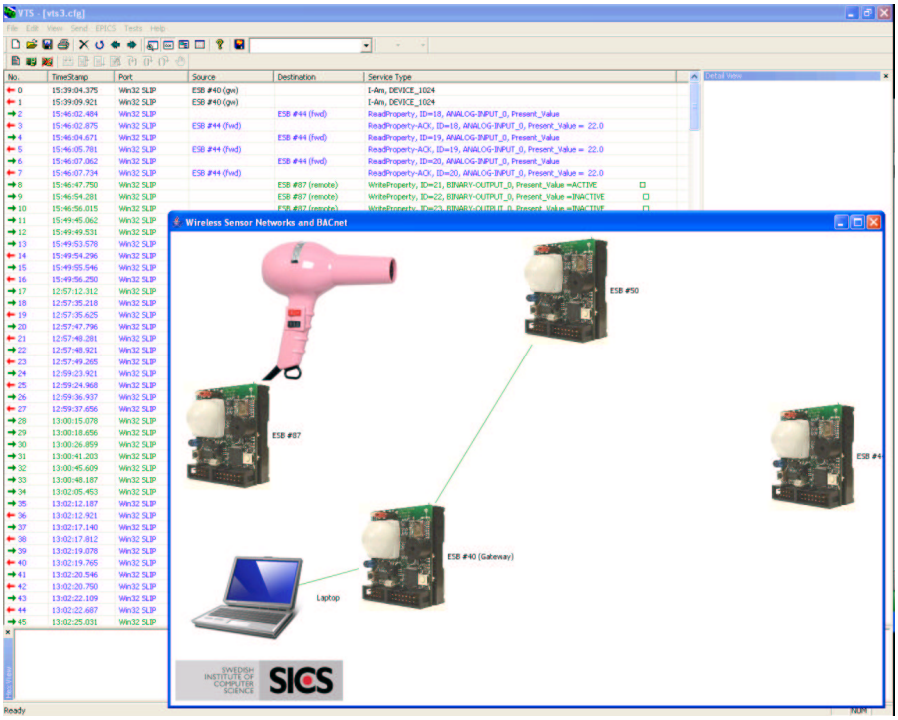

Figure 6.2. VTS and demo network application screenshot

Chapter 7

Related Work

Kintner-Meyer and Conant [13] report on their experiences from deployments of wireless sensor networks for building automation. Their focus is mainly on the monetary cost compared to wireless solutions. They use simple sensors that transmit temperature readings at predefined intervals to mains-powered receivers whereas we actually implement a building automation protocol on the sensor nodes which is a more generic solutions enabling all kinds of services.

EnOcean [9] is a spin-off company of Siemens providing control and sensing devices using wireless technology. The primary product focuses are smart energy sensors, battery less wireless devices using energy scavenging such as piezoelectric, solar, and electrodynamics. A typical building automation product from EnOcean is a wireless light switch using a piezoelectric element to generate enough power to send an on/off signal to the coordinator. Each battery less device connects to a line or battery powered coordinator in a star network topology. Each coordinator can then connect to other coordinators using ZigBee in a star/mesh network. A proprietary transmission protocol is used for the wireless devices whereas we build on standard protocols such as TCP/IP and BACnet and focus on the integration of WSN and BACnet.

KNX [1] is, similar to BACnet, a standard for building control. KNX is open, royalty-free and sustains platform independency by issuing certificates to vendors. In comparison with BACnet, KNX also focuses on smaller buildings such as home control.

Recent research on KNX in the wireless domain [19] argues that the benefits of wireless BAS include reduced installation costs, enabling sensor coverage where cabling is impossible, and enabling more flexible ad-hoc communication abilities. A prototype implementation of a wireless system supporting KNX uses, similar to our solution, a MSP430 micro-controller. However, in their system the MSP430 only acts as a gateway to the wireless sensor network. The micro-controller is connected to a 802.15.4 Chipcon 2420 radio transceiver [6], and to another dedicated KNX-controller device. All KNX data is handled by the KNX-controller and is sent to the MSP430 which in turn sends it on the network. In comparison, our solution has an entire BACnet-compatible system, including operating system and TCP/IP communication stack, on only one MSP430 micro-controller. By using only one micro-controller the final system is both cheaper and less energy consuming.

Chapter 8

Conclusions and Future Work

Large overhead costs including installation and maintenance in building automation system can be reduced by using wireless sensor networks. We have implemented and evaluated the open BACnet standard on top of a wireless sensor network. The results show that standard solutions, not originally designed for wireless sensor networks, such as BACnet and TCP/IP networking can successfully be applied in this domain. We have also identified the Change of Value service as particularly important when integrating building automation systems and wireless sensor networks. Future work includes further adapting the BACnet implementation to fit the resource-limited sensor nodes. Future work also includes evaluating different WSN energy saving techniques such as sleep-scheduling suitable for BACnet applications.

Acknowledgement and Source Code Availability

This work has been funded by the Swedish Energy Authority. Please contact the authors for more information. Note also that the source code is available on request. The source code for Contiki and uIP can be found at `http://www.sics.se/nes`.

Bibliography

[1] Konnex association. Web page. 2006-10-27. http://www.konnex.org

[2] Visual test shell for bacnet. Web page. http://sourceforge.net/projects/vts/

[3] ASHRAE. *BACnet - A Data Communication Protocol for Building Automation and Control Networks*, ansi/ashrae standard 135-2004 edition, 2004.

[4] ASHRAE. Bacnets home page. Online, 2007. http://www.bacnet.org

[5] Frank Capuano. Open systems: Lonworks technology and bacnet standard. White paper, 2001.

[6] Chipcon AS. CC2420 Datasheet (rev. 1.3), 2005. http://www.chipcon.com/

[7] A. Dunkels. Full TCP/IP for 8-bit architectures. In *Proceedings of The First International Conference on Mobile Systems, Applications, and Services (MOBISYS '03)*, May 2003.

[8] A. Dunkels, B. Grönvall, and T. Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Proceedings of the First IEEE Workshop on Embedded Networked Sensors*, Tampa, Florida, USA, November 2004.

[9] ENOCEAN. Enocean home page. Online, 2007. http://www.enocean.de

[10] J. Eriksson, A. Dunkels, N. Finne, F. sterlind, and T. Voigt. Mspsim – an extensible simulator for msp430-equipped sensor boards. In *Proceedings of the European Conference on Wireless Sensor Networks (EWSN), Poster/Demo session*, Delft, The Netherlands, January 2007.

[11] Texas Instruments. Msp430 ultra-low power microcontroller web page. Web page. http://www.ti.com/msp430

[12] W. Kastner, G. Neugschwandtner, S. Soucek, and H.M. Newmann. Communication systems for building automation and control. *Proceedings of the IEEE*, 93(6):1178–1203, June 2005.

[13] M. Kintner-Meyer and R. Conant. Opportunities of wireless sensors and controls for building operation. In *In Proceedings of 2004 ACEEE Summer Study on Energy Efficiency in Buildings*, pages 3–139–3–152. American Council for an Energy-Efficient Economy, Washington, DC., 2004.

[14] Modbus-IDA. *MODBUS APPLICATION PROTOCOL SPECIFICATION*, version 1.1a edition, June 2004.

[15] F. Österlind, A. Dunkels, J. Eriksson, N. Finne, and T. Voigt. Cross-level sensor network simulation with cooja. In *Proceedings of the First IEEE International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp 2006)*, Tampa, Florida, USA, November 2006.

[16] C. Perkins, E. Belding-Royer, and S. Das. Ad hoc on-demand distance vector (aodv) routing. RFC 3561, Internet Engineering Task Force, 2003.

[17] J. Polastre, R. Szewczyk, and D. Culler. Telos: Enabling ultra-low power wireless research. In *Proc. IPSN/SPOTS'05*, Los Angeles, CA, USA, April 2005.

[18] V. Raghunathan, C. Schurgers, S. Park, and M. Srivastava. Energy aware wireless microsensor networks. *IEEE Signal Processing Magazine*, 19(2):40–50, March 2002.

[19] Reinisch, Granzer, Neugschwandtner, Praus, and Kastner. Wireless communication in knx/eib. In *Proceedings of the KNX Scientific Conference 2006*, Institute of Automation, Vienna University of Technology, 2006.

[20] RF Monolithics. 868.35 MHz Hybrid Transceiver TR1001, 1999. `http://www.rfm.com`

[21] Jochen H. Schiller, Achim Liers, Hartmut Ritter, Rolf Winter, and Thiemo Voigt. Scatterweb - low power sensor nodes and energy aware routing. In *HICSS*, 2005.

[22] L. Zheng and H. Nakagawa. Opc (ole for process control) specification and its developments. In *Proceedings of the 41st SICE Annual Conference*, pages 917–920 vol.2, Osaka, Japan, August 2002. SICE 2002.