Scriptable Sensor Network based Home-Automation

Thomas Haenselmann, Thomas King, Marcel Busse, Wolfgang Effelsberg and Markus Fuchs

University of Mannheim,
A5, 6 · 68159 Mannheim · Germany
{haenselmann,king,busse,effelsberg,fuchs}@informatik.uni-mannheim.de

Abstract Today, proprietary home automation targets very specific applications which operate mostly on a cable based infrastructure. In contrast to that, our implementation builds a wireless ad-hoc multi-hop network based on the ESB sensor node platform from the FU-Berlin.
The nodes gather sensor readings in a home and transmit them to a central automation server. There, the readings are matched against a list of script statements. In case of a match, a specific action is performed. In this work we will show how the user can implement complex home automation applications optimized for his specific needs by defining very simple script statements. An important property of the system is also that the control of all home appliances is done by means of IR communication and Ethernet enabled multiple plugs. This way, the cooperation between manufacturers is no necessity in order to connect devices to the home automation network.

Key words: home automation, building automation, sensor networks

1 Introduction

While the field of RFID technology constantly produces new applications and solutions for real world problems, research on sensor networks tends to be a mostly academic topic in which strong commercial applications are still rare [16,10]. For this reason we want to describe a home automation project with sensor networks we have done in conjunction with Siemens Corporate Technology CT/SE2.

Home automation offers a not yet exploited degree of convenience, both for the private home and the office. Although the idea has been around for many years, the market can still be considered to be in its infancy. Today's home automation solutions are mostly proprietary. They usually target a small number of problems, such as satisfying security needs or the control of a limited number of devices, typically all from the same manufacturer. They operate based on a particular infrastructure, which requires extra cabling. So they are best suited for new buildings. They are limited to the applications a manufacturer offers.

The future proliferation of home automation will depend on its ease of installation. That is why we argue for wireless home automation. In addition, this might be the only solution for ex post installations and historic buildings which must not be remodeled. At the same time, we believe that an even more important aspect to making the smart home a success will be to offer more freedom for a user to customize home automation application to his specific needs.

In short, the idea of our prototype is to gather all kinds of sensor readings in a home and forward them hop-by-hop to an embedded system to which we refer as the home automation server. Each time a new event is detected, the server runs over a list of script statements which can be defined by the user. In case of a match between the received event and the matching part of a statement, one or more actions are performed which can either be executed by the sensor nodes themselves or by multiple plugs which can be controlled via an Ethernet connection by the embedded home automation server itself.

The strength and contribution of our application lies in the combination of a larger number of sensor readings which allows to derive higher level semantics as compared to reacting on single sensor readings only.

In the following Section 2, we analyze todays existing standards in the field of home automation. In Section 3, we describe all technical aspects of our system and how to exploit multiple readings for concluding deeper semantics as compared to using single sensor readings, only. Section 4 concludes with an analysis of strengths and weaknesses of the system.

2 Related Work

Since the beginning of electrification, switching electrical devices has been done by means of connecting or disconnecting them to the power grid. In recent years, physically disconnecting a device from its energy source has become less popular. Instead, switching is done electronically. This means, that the inner device is separated from the switching circuit. As a consequence, the device can be powered on or off by a remote control. Some computer main boards even allow to react on network events. However, the downside is that the switching unit keeps consuming energy as long as it stays alert.

The changing paradigm in home automation is also that a device is no longer disconnected from the power grid. The function of the switch on the wall or even in the device in taken over by a network which is solely signaling events. The network which controls devices by transmitting datagrams is powered with a much lower current. The earliest instance of a pure datagram based network standard for building automation is the EIB standard implemented in 1992.

Even earlier home automation systems like the X10 system, combined the signaling network with the power grid. This technology denoted as *power-line* based has regained popularity recently as an alternative to DSL technology which requires dedicated signaling cables like telephone lines. On the other hand, power-line based systems have inherent problems like radio interference, security flaws and reliability issues which have never been solved completely.

2.1 Powerline-based home automation protocol X10

X10 is a power-line based building automation protocol. It is used to transmit the control signals via existing power lines without the need of dedicated signaling cables. X10 is used to trigger simple control events. However, it never gained a strong foothold for mission critical applications because no feedback channel is provided and the effective data rates are only about 20 bit/s. The bits of a message are modulated on a 120 kHz signal. In oder to be more error resilient, only the zero-crossings of the alternating current are used. In addition to the power-line based approach, X10 provides remote controls and switches based on radio communication, as well [15].

The X10 protocol was developed by the Irish company Mico Electronics in the 70ies. Due to its adoption and promotion by General Electric is became very successful in the United States. In Europe, a modified standard was sold which did not have to same success as compared to the one overseas. Due to different regulations, the signal strength had been reduced significantly thus rendering the solution less useful for many applications. As a consequence, the technically more advanced EIB protocol became dominant in Europe.

2.2 European Installation Bus (EIB)

As early as in the mid-80ies, different companies though about using bus-topologies for home- and building-automation. Even at that time it was obvious that proprietary home automation solutions would hinder the proliferation of home automation. Leading manufacturers of electrical installation technology among them Siemens, Jung, Merten et al. founded the *European Installation Bus Association (EIBA)* in 1990 which became the *Konnex* association later. Their aim was to establish a joint standard for home automation [4,11,13]. This standard guarantees the interoperability of various devices and of systems like home appliances, air conditioning etc. from different manufacturers. In 1991, the first products were manufactured according to the standard. Today, there are as much as 4000 groups consisting of products manufactured by more than 100 companies. These products are compliant to the EIB/KNX specification which is the first globally agreed standard for home- and building automation. The standardization by the ISO committee is currently on its way.

2.3 KNX

KNX can be considered the international successor of the EIB standard. KNX is downward compatible to EIB and it has been acknowledged by more than 100 companies.

3 Scriptable sensor network based home automation

Our system consists of the ESB [8] sensor nodes describes in the next section. They transmit messages hop-by-hop over a tree topology which has to be initialized by the user semi-manually in advance. The root node is connected via its serial interface to the embedded home automation server described in Section 3.1. The embedded board runs a striped down version of Linux and a minimal web server to allow the user to configure the system remotely.

The nodes not only act as sensors but also as actuators which can control basically all devices which come with an IR remote control. Therefor, we have extended the ESB's firmware to be compatible with the three de-facto standards for remote controls by Philips, Panasonic and Sony. In addition, the buzzer and the LED lights can be used as actuators in some cases. More important is the multiple power plug with an Ethernet interface. It allows to switch all devices switch can be turned on and off only and which can not be controlled elsewise.

In the beginning, the nodes have to be distributed in the house according to the requirements of the considered applications. In the distribution process, the user is given hints by the system on where to place intermediate nodes for the purpose of communication.

In the operational phase, events are forwarded by the network to the root node and eventually to the home automation server in a tree-like fashion like it has been proposed e.g., by [9] in the context of the *TAG*-approach and may others [1,12]. There, they are matched against so-called script statements which have been configured by the user via a web-interface as described in Section 3.4. In case of a matching statement, a defined action is executed which implements one particular home automation application respectively which serves one particular purpose like e.g., baby surveillance. In this case, the executed action could be as simple as signaling the user with the buzzer or by sending him some information over the web by the embedded server, e.g., to submit an SMS via an external service.

3.1 Hardware used

The electronic sensor board Due to its rich instrumentation with various sensors we chose the ESB sensor node shown in Figure 1 developed by the FU-Berlin.

The ESB is equipped with the MSP430 [3], an embedded system on a chip from Texas Instruments. It runs at 8MHz and contains 64kB of memory in the version of the chip used here. The MSP430 is designed as a general purpose embedded system with a 12-Bit AD/DA (analog <-> digital) converter. The energy consumption is in the order of magnitude of 1mA at a current of 3V if the MSP430 if fully operational. In sleep mode which can be adjourned by external events, the power consumption is again about 1000 times lower which is roughly equal to the self-discharge of the batteries.

Most of the 64kB are implemented as flash memory which will contain the software and all constant data. The RAM occupies only 2048 bytes within the whole memory map which is a fairly limited amount of space for dynamic data. The situation is mitigated to a degree by the fact that the flash memory can also be written in chunks of 128 bytes during operation if the state of the battery allows for this energy consuming operation.

Under the ESB's white hemisphere are is a temperature and PIR (passive infrared) sensor hidden. The PIR sensor can be used for monitoring the space around the ESB up to a distance of 8 meters to detect moving objects like it is used for alarm systems.

There are two other IR (infra red) diodes on the circuit board for sending and receiving e. g., RC-5 codes that are used by remote controls for consumer electronics. The IR-communication is particularly important because it allows the nodes to serve as actuators which can influence their environment by interacting with many home appliances like air conditions, home entertainment devices etc. At the same time the IR communication provides another way to influence nodes with consumer remote controls. They are treated like any other sensor event, sent to the embedded board and matched against a user-defines script statements.

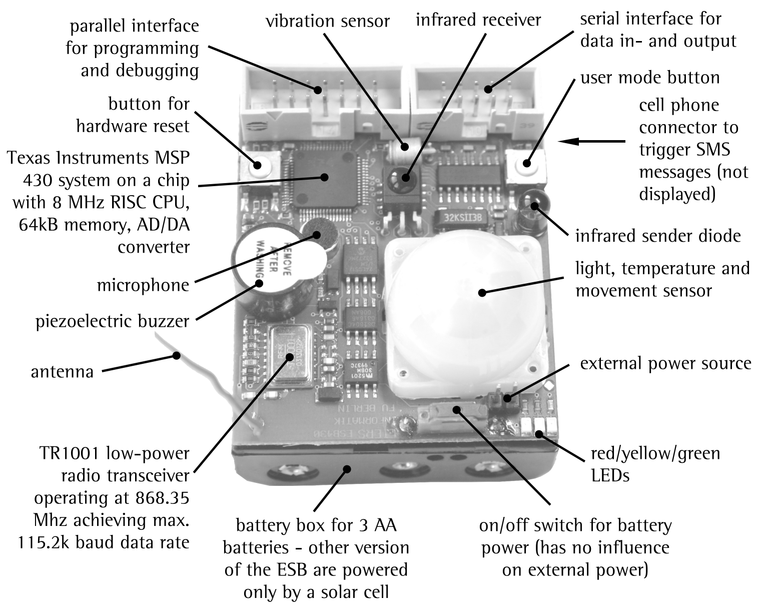

parallel interface for programming and debugging
vibration sensor
infrared receiver
serial interface for data in- and output
user mode button
button for hardware reset
cell phone connector to trigger SMS messages (not displayed)
Texas Instruments MSP 430 system on a chip with 8 MHz RISC CPU, 64kB memory, AD/DA converter
infrared sender diode
light, temperature and movement sensor
microphone
piezoelectric buzzer
antenna
external power source
TR1001 low-power radio transceiver operating at 868.35 Mhz achieving max. 115.2k baud data rate
red/yellow/green LEDs
battery box for 3 AA batteries - other version of the ESB are powered only by a solar cell
on/off switch for battery power (has no influence on external power)

Figure 1: Overview of the components of the Embedded Sensor Board (ESB) from the FU-Berlin.

Furthermore, the ESB is equipped with a vibration sensor that can sense slight vibrations of the device. Last not least, the ESB is equipped with a microphone and a piezo-electric buzzer. The buzzer is another simple actuator used to signal the user acoustically. By its design, it can only produce a single frequency, however, by switching it on and off rapidly, the firmware can also simulated other frequencies. There is also a microphone attached to the ESB. It can be used to measure the loudness of noises in the node's proximity. A very simple application would be to implement a baby-phone by signaling the owner in case of noise in the nursery.

The red, yellow and green LEDs are useful for signaling simple events. We use it in the deployment phase described in Section 3.5.

PowerPC-603 based embedded board As what we refer to as a *home automation center*, we used the embedded board EP5200 from Embedded Planet[1]. It is a complete system on a single motherboard. Though there is an IDE connector for a hard drive, we used the on-board 16MB flash memory for installing a minimal Linux installation based on the Linux distribution *Gentoo*[2]. The flash memory can be accessed like a hard-drive using the *jffs2*[3] file system.

[1] http://www.embeddedplanet.com/
[2] http://www.gentoo.org
[3] http://sources.redhat.com/jffs2/

Besides the need to cross-compile the kernel and to replace the hard drive by the flash memory, there is no different to using an IBM PC-compatible system. However, it is important to delete all files not needed in the boot process to achieve a memory footprint which fits into the bounds of the 16MB flash memory. In the development process we attached a simple USB stick to the USB port to host the GNU tool chain and other essentials.

We believe that an embedded system is suitable for its small form factor, price and stability. It has no moving parts, needs no ventilation and is certified for continuous operating in an environment of between -40 to 80 degrees Celsius.

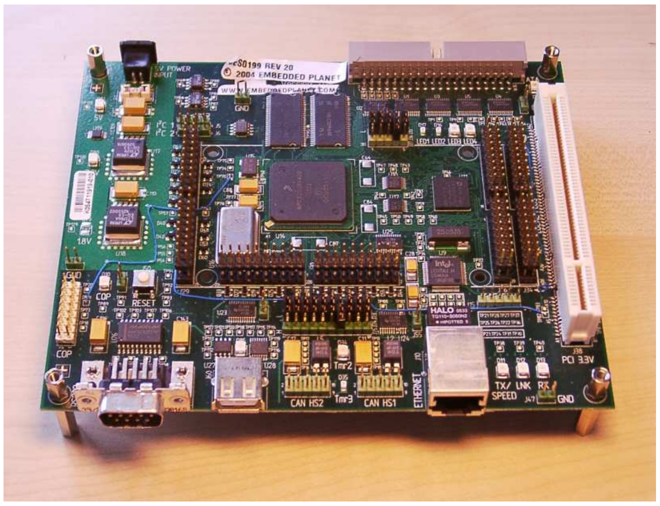


Figure 2: The EP5200 board is based on an PowerPC-603 processor. It has no moving parts. The flash memory can be used to host the root file-system.

Ethernet-enabled multiple plug An important actuator is the ethernet-enabled multiple plug shown in Figure 3. It can be used to switch all electric equipment which has an on/off switch only like e.g., lamps. From a user's perspective, the multiple plug can be controlled via a web-interface. In our implementation, the home automation center sends simple http-requests to the socket in order to switch one of the two relays on or off.

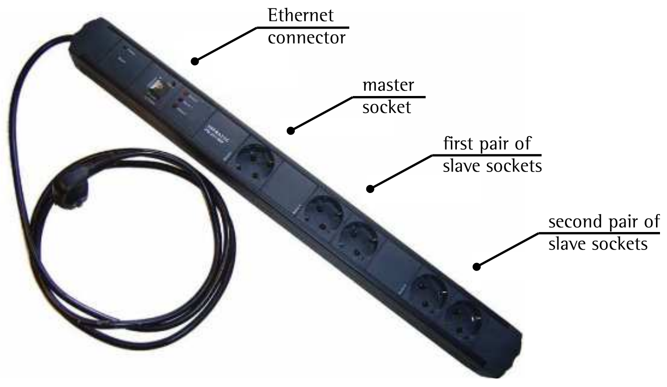


Figure 3: The Ethernet-capable multiple plug is an important actuator for controlling all kinds of devices which can be switched on and off only.

The downside of the solution is so far, that the multiple plug needs to be connected to a cable-based LAN. However, we expect similar wireless devices to be available soon as well. As an intermediate solution we connected an Ethernet bridge to the plug in order to become independent of the LAN-cable.

The master socket can be used to switch one or the two slave sockets on if the attached master device consumes energy. At the moment, we control the slave sockets via Ethernet only.

3.2 New home automation paradigm

Traditional home automation solutions target isolated problem. They may e.g., close the window's roller blinds at night, control the central heating and air conditioning or they may serve security needs.

Some solutions are helpful for handicapped people. If the doorbell or the telephone can not be heard, sensors capture the acoustic signals and trigger actuator like spot lights or vibrating haptic devices which wake or signal the hearing impaired owner.

In recent years, even solutions for pet owners have emerged. A cat's collar is equipped with a passive RFID transponder. At the cat door, a reader reads the passive tag within a range of about 30cm. Once the cat approaches the reader and the tag is authenticated, the door is unlocked by a simple mechanism. This way, alien cat, rats or other small animals can be prevented entry from the house.

All those solutions have in common, that they target a very specific application only. Especially those for disabled people can be very costly and may not always solve all individual needs. So we propose a new paradigm which enables the user himself to devise customized solutions by means of simple script statements.

3.3 User-define home automation scripts

In the home automation center, all sensor readings are gathered. The user defines script statements like the following one which are matched against the incoming sensor readings.

```
IF movement_detected(sensor-5) == true THEN
   switch_power(multi_plug-5, on),
   switch_power(multi_plug-6, on)
```

Every script has a *matching part* and an *actuator part*. The server software on the home automation center iterates over all script statements each time an event is received. In case of a match, the actuator part is executed.

The above example switches on two lamps connected to a multiple plug outlet each time a room is entered by a person. Besides these trivial statements, the strength of the approach lies in the combination of more than one sensor readings. The more readings are combined, the higher the semantics that can be derived.

The example above could be extended by the time of day to differentiate between various situations. E.g., switching on the light is not necessary at any time but only at dusk or at night. So we add a light reading on the matching side:

```
IF movement_detected(sensor-5) == true AND
   lightness(sensor-5) < 800 THEN
   switch_power(multi_plug-5, on),
   switch_power(multi_plug-6, on)
```

Not all events have to be triggered by sensors. Another independent event can be the daytime.

```
IF time_within(05:00,23:00) == true AND
   movement_detected(sensor-5) == true AND
   lightness(sensor-5) < 800 THEN
   switch_power(multi_plug-5, on),
   switch_power(multi_plug-6, on)

IF time_within(23:00,05:00) == true AND
   movement_detected(sensor-5) == true THEN
   switch_buzzer(senor-7, on)
```

In the first line, a movement at daytime causes a light configuration to be switched on. In the second line, the same movement will trigger an alarm if it occurs while the owner sleeps at night.

A word on an implementation issue: Most sensors of the ESB motes sample measurements all the time which may even be noise in case of silence. In order to trigger e.g., an audio event or a moment event, the platform's firmware allows to specify threshold values for various sensors. Once these thresholds are exceeded, a message is sent out. Alternatively, the raw values can be sent via the wireless interface, constantly. This option is less popular for its high energy consumption. Setting thresholds, however, requires some prior configuration of the nodes within their particular environment.

Another example for deriving higher level semantics from sensor readings is shown in our demo-video[4]. One of the ESB sensor nodes is attached to a door facing the inside.

[4] See http://www.informatik.uni-mannheim.de/~haensel/sn_homeautomation.avi

Two different readings are transmitted by the sensor, one originating from a passive infrared movement sensor that detects movement in a proximity of about 8 meters. The other reading comes from a vibration sensor which detects if the sensor itself is shaking. The two sensor readings can be used to differentiate between a door being opened from the inside or outside. To be more precise, the order of occurrence determines the case.

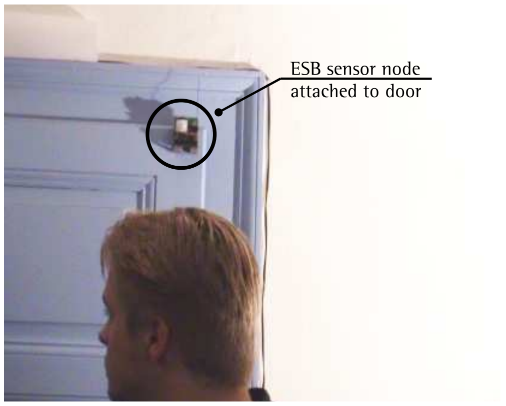


Figure 4: Extract from our demo video: Two sensor readings are used to distinguish whether the door is approached from the inside or the outside.

Door being opened from the inside: Here, a person approaches the door respectively the attached sensor node. The passive infrared sensor will trigger an event which is sent to the embedded home automation center. Then, the person will open the door which triggers the vibration sensor in addition because the entire node is moving together with the door.

Door being opened from the outside: A person approaches the door from the outside. No sensor readings are triggered so far because the sensor is attached to the opposite side. Once the door is opened, both sensors react at the same time. The vibration sensor because it is moving together with the door and the passive infrared sensor because it is rotated by the opening door. Move precisely, from its perspective, the environment rotates around the node. So both events occur more or less simultaneously.

But not only taking multiple sensor readings into account at a time can help to derive higher level semantics. Historic events can help as well.

Example: We assume that a house is empty in an initial state and no door has been opened so far. If movement occurs in a room, someone may have entered through a window. If there was prior movement in the hall or the door has been opened it may be an inhabitant. Historic events are modeled by conditions which are described in the next section.

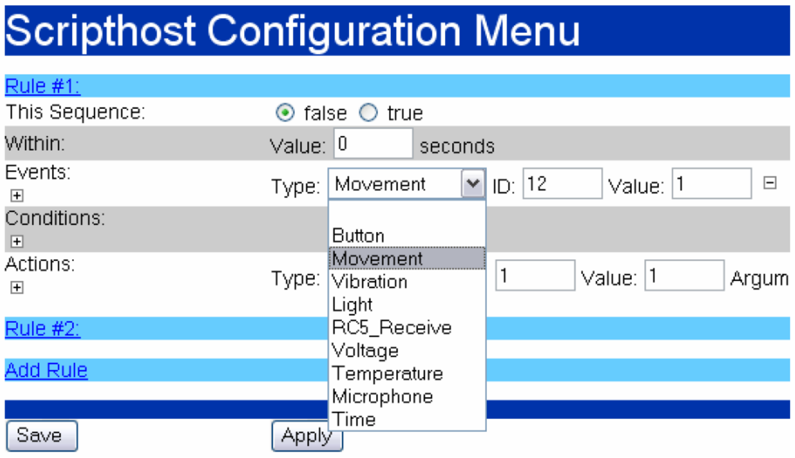
The examples above should only provide a first impression of the possibilities which emerge if many sensor reading are gathered and matched in order to draw conclusions. These conclusions can be far more valuable than those which are derived from single sensor readings as done in many isolated home automation applications. In this work we only want to sketch the idea of customized home automation and prove its feasibility by means of the implementation. As is true e.g., for the World Wide Web as well, the most interesting applications will likely come from creative practitioners and not from academia.

3.4 Web configuration of rules

Figure 5 shows the browser based configuration. The user can define an arbitrary number of rules each of which appears as a single line that can be unfolded to a dialog for later editing.

A rule consists of three elements whereas the triggering event and the action to be performed are the two compulsory elements. Whenever an event like a sensor reading occurs, it is compared with all rules. If the event matches a rule, the according action is performed.

Whether an action is performed must not always depend on an event only but also on a condition which, unlike the event, persists over some time and does not occur at a single moment only. So a rule can have an arbitrary number of conditions which have to be met in addition to an occurring event. A condition can e.g., be a specific time of the day or a prior sensor reading like light or temperature. Multiple events, conditions and even actions can be defined within a single rule by the user. This way, the above mentioned deeper semantics can be accomplished.


Scripthost Configuration Menu

Rule #1:

This Sequence: [✓ false] true

Within: Value: 0 seconds

Events: Type: Movement ID: 12 Value: 1

Conditions:

Actions: Type: 1 Value: 1 Argum

Rule #2:

Add Rule

Figure 5: Browser based configuration of home-automation rules. Each rule consists of at least one triggering event, an arbitrary number of conditions (including none at all) and at least one action.

Technically, the page is generated by a php-script on the server side and the rules are stored in a single XML file.

3.5 Deployment phase

Prior to the operational phase of the home automation network, the sensor nodes have to be deployed. In the beginning, the root node and the leaf nodes have to be installed.

The root node has to be connected to the embedded board which should have access to the home's LAN. The LAN connection is mainly used to control the Ethernet enabled multiple plug that is used to switch simple electronic devices on and off.

The location of the leaf nodes is determined by the purpose of the application. If the audio-sensor should e.g., be used as a baby-phone, a sensor node has to be positioned in the nursery. The root node and the leaf nodes are considered as *active nodes* in our implementation because they server a specific purpose. Especially in an indoor environment, the range of the sensor nodes can be limited, particularly if neighboring nodes are separated by walls. So direct communication between leaf nodes and the root can not be assumed in general. This is why the user has to bridge the gaps between the root and the leaves by positioning intermediate *passive nodes* which server as packet forwarders only.

All nodes connected to the root directly or indirectly are considered to belong to the *active partition*. The task of the user is now to connect the active partition to all leaf nodes which are unconnected. In the deployment process he picks up initialized nodes and carries them away from the active partition into the direction of a leaf nodes. While being close enough to a connected node of the active partition, the green LED is blinking to indicate a good connection. It is considered to be good as long as three consecutive ping packets return from the root node. Less than three packets result in a yellow indicator and no arriving ping packets are signaled red. The relatively small number of test packets has been used because each of them takes about 300ms to be transmitted. A latency of about 1s is still small enough as feedback for the user.

So he will start at the active partition and walk towards a yet unconnected leaf node. As soon as the quality of the connection decreases, he has to step back and position the nodes permanently. In the end, the node is put into its operational mode by pressing the user definable button. The leaf node will also try to reach the active partition by sending broadcast ping packets. As soon as there is a good connection it will start to blink green as well and can be set into the operational mode in the same way as the nodes carried around.

The user has to continue this process until all leaf nodes are connected. Note that in some cases, active nodes can and will serve as forwarders also if they are connected to the active partition themselves.

The routing of the packets among the nodes is done according to the tree which is generated by the user implicitly by deploying the nodes. As a consequence, each node forwards information only to its direct father via static routes. Gathering the sensor readings was also easy to implement on the side of the nodes as the firmware readily supports sending events in regular intervals and based on thresholds.

4 Evaluation

In the evaluation, we mainly focused on a qualitative analysis of our implementation since there were not prior wireless home automation systems available to us.

4.1 Energy supply

Even though the use of sensor nodes eliminates the need for cables, it creates the new problem of supplying the nodes with energy. Even under optimal conditions, our ESB motes will not operator more than a couple of months on a single set of batteries. Having 20 or more nodes installed, this would mean for the user to change some batteries once in a week on average.

The ESB nodes come in different versions. One of them replaces the battery box by a photo diode which charges a capacitor. Depending on the light conditions, the energy stored this way is enough to operate for a few seconds. Though this does not seem to be much, for many applications it suffices for making some measurements and sending them over the network. Often, this does not require more than several hundred milliseconds. A useful property for the ESB platform is in this context that sensor readings can be used to wake the node from its power saving mode without using the processor. Waking up a node means to raise the clock rate of the processor which can e.g., be triggered by defining a threshold for a sensor. Though the threshold is a simple means of measurement it is useful for saving energy in times of inactivity.

At a first glance, intermediate nodes must not sleep since they may have to forward packets from their neighbors at any time. However, a number of MAC schemes like *WiseMAC* have been developed in recent years which keep a network alert while almost completely reducing idle times [7,14].

4.2 Security problems

Wireless transmission is know to be error-prone in general and in particular in case of many sensor nodes. We exemplary evaluated the transmission characteristics of our sensor platform [6]. One outcome of the studies was that there is a very strong variance both in transmission and reception characteristics among different nodes of the same type [5]. Furthermore, the quality of a directed link between two nodes can deviate significantly from the inverse direction.

Another problem which is inherent to all wireless networks is that it can be sabotaged easily by jamming the frequency band used. So the wireless channel is more useful for less mission-critical applications. In case of an alarm system, a beacon based approach might be adopted. The alarm could be triggered by missing beacons so that distorting the channel would not prevent the alarm. Though less critical, an attacker could still trigger false alarms.

On the other hand, the use of wireless communication is more and more debated, e.g., for industry automation [17] and even for communication within airplanes. The later one is referred to as *fly-by-wire*. First prototypes have been build based on unmanned planes [2].

5 Conclusion and outlook

In this work we described a wireless home automation system based on sensor networks. The ESB platform we chose allows for easy installation and extension of the system.

Other than commercial home automation solutions available today, we propose to let the user come up with customized solutions for his individual requirements by formulating script statements. These statements which are entered via a web interface react on a combination of events and can trigger a list of actions in case of an occurring match. Furthermore, the execution of the actions can be made dependent on an arbitrary number of conditions which have to be met.

The nodes do not only act as sensors but as actuators as well. By sending infrared RC5 codes, almost all electronic equipment using a remote control can be switched. Simple devices are switch by means of a multiple plug with an Ethernet connection.

Though an increasing number of home appliances are controllable by IR remote controls today, we plan to connect to the European Installation Bus in addition. In this context it will make sense to adopt the EIB protocol for the wireless communication as well, at least on the application level. The extension of EIB to a wireless implementation could work in the style know from the Bluetooth standard. There, the layer two serial line connection is emulated by an underlying wireless connection. The actual application level communication does not have to be changed. In our case we aim at running the EIB protocol on top of the wireless connection in the next version of our prototype.

References

1. N. Abramson. Power-aware routing in mobile ad hoc networks. In *Proceedings of the Fall 1970 AFIPS Computer Conference*, 1970.
2. J. A. Afonso, E. T. Coelho, R. Macedo, P. Carvalhal, L. F. Silva, H. Almeida, C. Santos, and M. J. Ferreira. A fly-by-wireless platform based on a flexible and distributed system architecture. In *IEEE International conference on industrial technology*, Mumbay, India, December 2006.
3. L. Bierl. *Das große MSP430 Praxisbuch*. Franzis Verlag GmbH, 2004.
4. B. Bruegge, R. Pfleghar, and T. Reicher. Owl: An object-oriented framework for intelligent home and office applications. In *Proceedings of the Second International Workshop on Cooperative Buildings (CoBuild99)*, 1999.
5. M. Busse, T. Haenselmann, and W. Effelsberg. The Impact of Resync on Wireless Sensor Network Performance. In *Proc. of Performance Control in Wireless Sensor Networks (PWSN'06)*, Coimbra, Portugal, 05 2006.
6. M. Busse, T. Haenselmann, T. King, and W. Effelsberg. The Impact of Forward Error Correction on Wireless Sensor Network Performance. In *Proc. of ACM Workshop on Real-World Wireless Sensor Networks (RealWSN'06)*, Uppsala, Sweden, 06 2006.
7. A. El-Hoiydi, J.-D. Decotignie, C. Enz, and E. Le Roux. Wisemac, an ultra low power mac protocol for the wisenet wireless sensor network. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 302–303, Los Angeles (CA), USA, November 2003.
8. E. Kappe, A. Liers, H. Ritter, and J. Schiller. Low-power image transmission in wireless sensor networks using scatterweb technologies. In *Workshop on Broadband Advanced Sensor Networks*, San Jose (CA), USA, October 2004.
9. S. Madden, M. Franklin, J. Hellerstein, and W. Hong. Tag: a tiny aggregation service for ad-hoc sensor networks. In *Proceedings of the OSDI'02 Symposium*, Boston (MA), USA, December 2002.

10. R. Orr and G. Abowd. The smart floor: A mechanism for natural user identification and tracking, 2000.
11. S. Pitzek and W. Elmenreich. Configuration and management of a real-time smart transducer network, 2003.
12. S. Singh, M. Woo, and C. S. Raghavendra. Power-aware routing in mobile ad hoc networks. In *ACM SIGCOMM Computer Communication Review archive*, volume 28 (3), pages 5–26, 1998.
13. D. Spinellis. The information furnace: User-friendly home control. In *In Proceedings of the 3rd International System Administration and Networking Conference SANE 2002*, pages 145–174, May 2002.
14. T. van Dam and K. Langendoen. An adaptive energy-efficient mac protocol for wireless sensor networks. In *Proceedings of the 1st international conference on Embedded networked sensor systems*, pages 171–180, Los Angeles (CA), USA, November 2003.
15. Y. Wang, W. Russell, A. Arora, R. K. Jagannathan, and J. Xu. Towards dependable home networking: An experience report. In *International Conference on Dependable Systems and Networks (DSN 2000)*, page 44 ff, 2000.
16. R. Want, K. P. Fishkin, A. Gujar, and B. L. Harrison. Bridging physical and virtual worlds with electronic tags. In *CHI*, pages 370–377, 1999.
17. P.-A. Wiberg and W. Bilstrup. Wireless technology in industry-applications and user scenarios. In *8th IEEE International Conference on Emerging Technologies and Factory Automation*, volume 1, pages 123–131, Antibes-Juan les Pins, France, October 2001.