Home Automation Design Using 6LoWPAN Wireless Sensor Networks

Dan Ştefan Tudose[†], Andrei Voinescu[*], Madi-Tatiana Petrăreanu[‡], Andrei Bucur[§], Dumitrel Loghin[¶], Adrian Bostan[‖], Nicolae Ţăpuş[**]
Faculty of Computer Science and Computer Engineering
Polytechnic University of Bucharest
Bucharest, Romania
Email: {[†]dan.tudose, [*]andrei.voinescu, [‡]madi.patrareanu, [§]andrei.bucur, [¶]dumitrel.loghin, [‖]adrian.bostan}@cti.pub.ro, [**]ntapus@cs.pub.ro

***Abstract*—Wireless sensor and actuator networks (WS&ANs) are a new technology based on networks of small radio-enabled embedded devices that are being deployed in areas such as environmental monitoring, vehicle tracking, building management, body monitoring and other applications. Power sources for network nodes are often limited, which imposes restrictions on hardware resources and their use by the underlying embedded software. We propose a new wireless sensor network architecture that is especially designed for the task of home automation. Our system relies on a low power WS&AN that employs energy harvesting techniques to maximize node lifetime and an embedded residential gateway that offers user interaction and secure connectivity to the outside world. The advantages of our system are its scalability, low power, self sufficiency and versatility.**

***Keywords*-Wireless Sensor Networks, Home Automation, Pervasive Computing, Energy Harvesting, Embedded Systems, 6LoWPAN**

I. Introduction

Wireless Sensor Networks (WSNs) or more generally Wireless Sensor and Actuator Networks (WSANs) are employed in a wide range of data acquisition, data processing, and control applications. Their advantages over traditional wired sensor and actuator networks include node mobility, increased reliability (due to availability of adaptive multi-hop routing), easier installation and lower deployment cost.

Home automation is the process in which the household environment is given additional functionalities through the integration of sensors and actuators into otherwise non-automated systems like lighting, heating, air conditioning and even regular appliances with the purpose of providing improved convenience, security and energy efficiency.

Almost all of the home automation systems that are currently available on the market employ wired networks and a multitude of communication protocols like X-10, Universal Powerline Bus (UPB), MODBUS, or even via a standard Ethernet connection. They all have been available for at least a couple of decades and, while technologically and functionally proven, they offer some disadvantages that hindered their widespread adoption. For example, MODBUS and Ethernet require cabling for both power and data lines that can be expensive and aesthetically displeasing. X-10 and UPB have the major advantage of utilizing the already existing power line and outlet infrastructure but suffer from low bandwidth and are susceptible to high error rates on low quality or noisy power lines.

Wireless sensor networks seemed the logical step to address the issue, because of their ability to function using relatively small, inexpensive, low-power nodes that can form short range networks using protocols like Bluetooth, Zigbee, WirelessHART[3], or 6LoWPAN [5].

However, not all of the above standards are equally well suited to a home automation scenario. For example, Bluetooth networks are usually limited to a small number of nodes and have higher energy consumption than its Zigbee counterparts. WirelessHART, although reliable and highly flexible, is more suited to Process field device networks that are used in industrial environments.

The 6LoWPAN standard promises the fulfillment of the emerging trend of embedding Internet technology into all aspects of everyday life [4], mainly because of its low costs, low power, scalability, possibility to easily adapt existing technologies.

In this paper we present a home automation infrastructure that is built around a 6LoWPAN wireless sensor network, an embedded gateway and an application suite for deploying, monitoring and controlling the system.

II. System Requirements

Home automation systems consist of interlinked components that are in effect a type of centralized distributed system that has a set of characteristic properties and attributes. According to [4], these are the following:

- **Future-proof.** A HA system cannot be easily upgraded or uninstalled during the lifetime of a building, so it needs to use a stable, proven and future-proof technology.
- **Moderate cost.** A HA system usually consists of a large quantity of sensing and actuating entities that need to be in constant communication both with each other

and with the central entity. Because of these specifications, most of the solutions for home automation tend to be either too costly, either inefficient. For the system to be effective, a compromise between cost and functionality must be achieved, while at the same time maximizing the benefits.

- **Low installation overhead.** Because current HA solutions rely entirely on wired communication, the installation of such a system proves complex and often needes modifications in the building itself. Any modern HA system has to have a low installation overhead, requiring little or no modification to the existing home environment.
- **Configuration effort.** System configuration should be easy and time-efficient. Adding new functions or modules to the system should be facilitated by a paradigm that is similar to plug-and-play.
- **Connectivity.** All entities of the system need to be connected, either through a unified interface or through a specialized one that allows bridging different technologies and hardware. Connectivity with the outside world is also a desired functionality.
- **User interaction.** Special care must be taken with interface ergonomics. The user should not be asked for ambiguous or repetitive commands and the interface must have familiar controls that need little or no training even for an inexperienced user.
- **Security.** The system must be aware and protect its users from threats like unauthorized access, privacy invasion or destruction.

Most of today's residences and apartments already have Internet connectivity, so, utilizing the existing Ethernet infrastructure as a backbone for our application is not only logical, but also satisfies all of the above requirements.

III. Overall system architecture

Our goal is to develop a house monitoring system that is robust, flexible, easy to use and has a wide range of capabilities. The main components of the monitoring system are a gateway and a network of low power sensor and actuator nodes.

Our Wireless Sensor Network (WSN) architecture has three main hardware components:

- Wireless Sensor Motes
- Network Gateway
- Android-enabled Smartphone

The way these components are interlinked to create a reliable WSN system is shown in the diagram below:

The embedded gateway is the core of the system. It provides the user with a touchscreen interface for configuring and monitoring the sensor network and the gathered data. As an enhancement to data monitoring, the user has the possibility of setting up alarms (e.g. the gateway sends a text message to the user if the temperature crosses a threshold). The nodes may be equipped with various sensors. The collected data is wirelessly sent to the gateway, where it is stored and eventually displayed. The system also provides means to monitor data remotely. The user can connect to the gateway via the Internet and view real-time graphs and statistics of the network data using an Android smartphone.

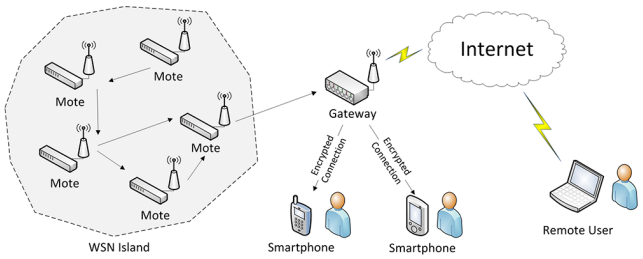

Figure 1. System architecture

A. Sensor hardware

In this section we describe the hardware implementation of our sensor motes. A sensor mote is a node in a wireless sensor network that is capable of performing some processing, gathering sensory information and communicating with other connected nodes in the network.

1) Typical wireless sensor node architecture: WSN nodes have specific hardware characteristics and limitations. Most WSN nodes have limited available energy: some rely on batteries and some employ environmental energy harvesting techniques such as solar panels, wind- or vibration-powered generators or thermoelectric generators. Therefore WSN nodes tend to be small embedded systems with few processing resources and low bit rate, low range radio links. Cost and size restrictions impose similar constraints. A typical architecture of a sensor node is shown below.

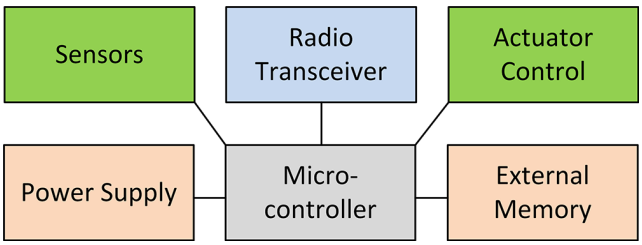

Figure 2. Typical wireless mote hardware architecture

2) The Sparrow v2 wireless sensor mote: For this project we developed the Sparrow v2 node [7] which is built around the Zigbit A2 module from Atmel. It has a low-power 8-bit RISC microcontroller connected to a 2.4GHz 802.15.4 radio transceiver. In order to increase versatility, the microcontroller is linked to an extended sensor bus that can accommodate up to three different types of analog and digital sensors. Although the mote has very low power

consumption and can function for long periods of time on a single battery charge, we designed the node for total energy-independence. Additional components for power management and energy harvesting were needed and we opted for the architecture presented in the diagram below. The voltage

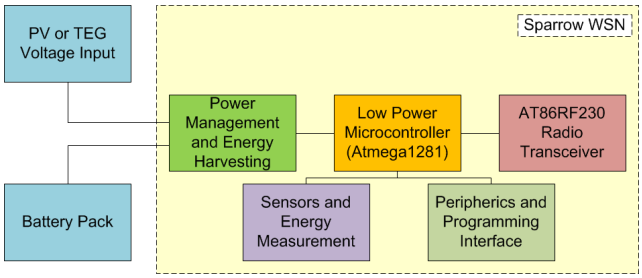

Figure 3. Sparrow v2 architecture

from the energy harvester is used to charge the battery pack by the first stage DC-DC converter. Then, battery voltage is supplied at a stable level to the node's main circuitry. For power management purposes, the node also needs to continuously monitor the voltage and the current drawn from the battery pack, which is achieved by the energy measurement module.

3) Energy Harvesting: is the process by which energy from the surrounding environment is captured and stored. In recent years the term has been applied mainly to sensor networks, where autonomous sensor nodes employ this process to replenish their energy resources. When applied to our architecture, energy harvesting increases the robustness and availability of the system, making it energy-independent.

Harvesting Technology	**Power Density**
Photovoltaic Cells(maximum illumination)	$15mW/cm^3$
Piezoelectric (cantilever structure)	$330uW/cm^3$
Vibration(kitchen appliance)	$116uW/cm^3$
Thermoelectric($\Delta t = 10 \deg C$)	$40uW/cm^3$
Acoustic noise ($100dB$)	$960nW/cm^3$

Table I
MOST COMMON ENERGY HARVESTING SOURCES

We can approximate the sensor network with a closed energy system where each node has a total energy production rate $P_p(t)$ and a total energy consumption rate $P_c(t)$. Therefore, the excess of harvested energy by the node at any moment can be estimated by the following formula:

$$E(t) = \int_0^t (P_p(t) - P_c(t))dt \quad (1)$$

A node is deemed energy-independent if its excess energy satisfies the following formula:

$$E(t) > 0, \forall t > 0 \quad (2)$$

A variety of sources for energy harvesting have been researched, such as solar power, thermal energy and kinetic energy. All of the energy sources stated above have small energy density values compared to more classic energy sources, such as batteries. In the past, the use of radio transceivers often implied large amounts of power consumption. This is no longer the case today, as recent advances in the design of low-power electronics and energy storage have made wireless sensor networks a prime candidate for the successful integration of energy harvesting techniques. By analyzing the data from the Table I we can see that solar cells offer the best efficiency while at the same time being a environmentally-friendly power source. This can be a suitable energy source for locations in which the availability of light to network nodes can be guaranteed to a sufficient degree, and for which mains and primary battery supply is impractical. A WSN node that has energy harvesting capabilities can virtually run for an infinite amount of time without the need of periodically replacing its batteries. For our project we used $2V, 200mA$ polycrystalline silicon solar cells, and we established that they can harvest sufficient energy for the mote's needs, even in low illumination conditions. To establish if they can successfully power our nodes we did the following experiment: we measured the average output power on a fixed $1k\Omega$ load when the cell was in full sunlight. We came up with an average $261mW$

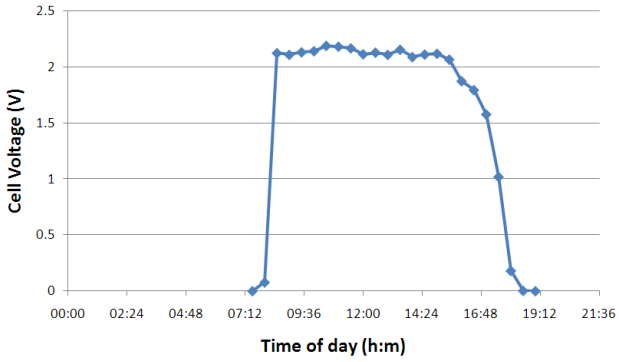

Figure 4. Photovoltaic cell voltage drop measured in the course of one day on a fixed resistive load

over the course of one day, taking into account that the solar cell is placed into an area of moderate to high illumination, such as a window frame. We can calculate the total energy harvested in one day by the solar cell:

$$E_{harvest} = P \times t = 261mW \times 24h = 22600Joule \quad (3)$$

The Sparrow node drains a maximum of $30mA$ at $3V$ from the power supply. By taking also into account the sensors and additional circuitry with an additional $10mA$ at the very most, and implying that no software sleep algorithms are implementes, the total energy required for the node to run

without pause for the duration of a single day will be:

$$E_{spent} = U \times I \times t = 3V \times 40mA \times 24h = 10368Joule \quad (4)$$

This proves that, given enough storage capacity and enough incident radiation, solar energy harvesting can power a node for an indefinite amount of time. Taking into account the fact that nodes employ power management in the software stack, alternating between long periods of sleep and only short intervals when they're active, there is actually excess energy produced. This additional energy is stored in the battery pack to be consumed during the night or on clouded days.

B. Sensor firmware

The nodes in our testbed run a light-weight operating system designed specifically for use in WSNs, named Contiki [1]. It offers a cooperative protothread model and an RFC-compliant wireless IPv6 stack, built on top of IEEE802.15.4.

Our system features two firmware versions of Contiki, one that runs on the regular sensor nodes and is suited for data gathering, and another version for a coordinator node. The coordinator has the added burden of a serial link to the embedded gateway, fulfilling the role of sensor data sink.

Sonda is the client firmware that runs on the sensor motes. It enables a node to wirelessly transmit data from its sensors to the gateway. Each sensor node can accomodate a wide range of sensors: temperature, pressure, humidity, light intensity, proximity detection, air quality, etc.

Transmission of the sensor values is done periodically over a UDP/IPv6 connection to the gateway. IPv6 addresses are fixed in EEPROM memory, as dynamic addresses are not implemented in the operating system. This makes the addresses easy to reconfigure, as opposed to rewriting the firmware on each node. The dissemination of data using UDP over uIPv6 allows a certain flexibility of the system since the coordinator does not have to know about what nodes are available, because it maintains the list of all the nodes that are contributing with sensor data. Data is sent over the network in a simple text format, therefore, is self-describing. Neither the coordinator nor the gateway have to know which are the capabilities of each individual nodes since they can be discerned easily from the data it is sending.

Sonda Gateway runs on the coordinator mote. It accepts wireless UDP connections that carry sensor data. Each mote sends datagrams containing pairs of values denoting the sensor type and measured data. The role of the coordinator mote is to forward the received pairs via UDP/IPv6 to the embedded gateway on a serial connection.

Sonda Power runs on a custom-built mote that measures power consumption at a mains outlet. The mote itself is a board based on the Sparrow module but features additional circuitry for AC voltage and current measurement.

IV. Embedded Gateway

1) Platform setup: When designing the gateway, we had to respect some specific constraints. The gateway must be a dedicated hardware device, low-cost and power efficient, easy to use by the end-user, reliable and secure. It has to provide the following functions:

- Collect data from wireless nodes (act like a gateway for the WSN) .
- Provide a direct interface for basic user settings and control via a touch screen.
- Provide a web interface for extended settings, visualization and control of the WSN.
- Process data and send it to MonALISA, a grid-based large-scale monitoring platform discussed by [2].

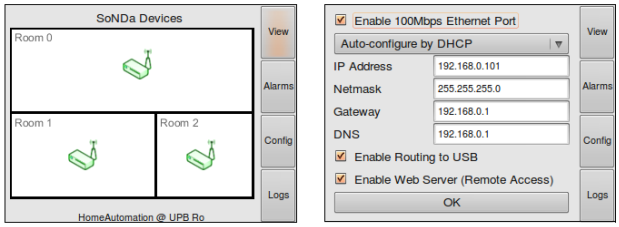

Figure 5. Screenshot of the embedded gateway system

We needed a powerful yet low-cost platform that can interface a wide range of peripherals. For the prototype, we chose the Atmel's ATNGW100 board that has all the features above and can run a Linux operating system. The board is equipped with an AP7000 (Atmel's proprietary AVR32 32 bit RISC architecture) processor that can run at 140MHz, which gives it enough computing power to run Linux, a small HTTP server with server-side scripting and a GUI application.

We interfaced to this board a 320x240 pixel color TFT display with touchscreen, to provide a basic user interface. In order to give the gateway access to the wireless network, we also added a Sparrow node.

For this system we used an up-to-date kernel version, 2.6.30.6, with several additional drivers enabled, mainly framebuffer support for the LCD display. The systems was built with buildroot, which is a set of makefiles for both kernel and userspace libraries. It includes all necessary support for a web server and a menu interface that uses Qtopia for the LCD.

The gateway receives and stores sensor data from the motes. This can be then viewed in a graphical form from the menus on the LCD screen and, at the same time, it is made available on the Ethernet connection by a HTTP server via REST-like queries. This enables a variety of possible Web applications to integrate sensor data, such as the Android platform we developed.

2) Application Design: The application running on the gateway is meant to be a terminal for the entire home automation system, providing both configuration screens and up-to-date information on the system. It has several graph

screens to plot the reported sensor data and configuration panels to setup gateway network parameters and to add alarms for certain sensor data.

As Figure 5 shows, the main screen has a representative map of the residence (designed by the user) along with the sensors mapped to each room. Users can also quickly view node availability and logged events in the system.

3) Gateway Interface: The system is designed to be easily customized after being deployed in the user's home. The graphical interface of the gateway can be configured to resemble the actual design of the house, providing the user with a simple, intuitive tool for interacting with the monitoring system. This facility is implemented through a web interface hosted on the gateway. We used Scalable Vector Graphics and Javascript in creating this interface, therefore it must be accessed form a browser that can interpret SVG files.

The purpose of the interface is to generate a PNG image that mimics the design of the home, along with some other information about each room: a room name and a list of sensors that are active in that room. The user inputs first a layout of the rooms in the house and then enters data regarding active sensors in each room. This is all that is required in order to generate a fully customized interface for the gateway.

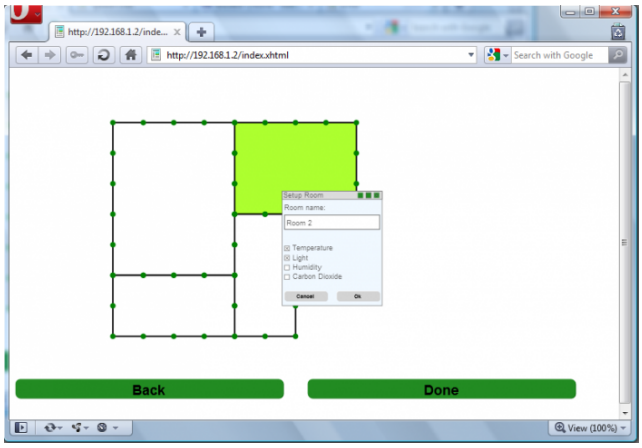

Figure 6. Customize interface screen

Generating the final image to be displayed on the gateway occurs as follows: first, the user input is used to generate a SVG image. Next, this image is scaled to the exact dimensions of the gateway's screen. This way we can obtain a clear image and use the screen's limited resolution as much as possible. Finally, the SVG image is converted to PNG format. The list of sensors for each room is stored into a configuration file. The interface configuration is completed after the PNG image and configuration file are uploaded to the gateway.

V. Android Monitoring Application

The Android application is meant to give additional ways of presenting and manipulating data obtained from the gateway. The gateway hosts an Apache 2.X HTTP server with the PHP and SSL modules installed, making the server-client connection secure. The application makes certain REST-like queries to the gateway, it makes parametrized requests and receives the data in JSON format. This format is preferred over other representation protocols because it integrates very well with JavaScript environments and many frameworks offer support for it. For example, when an user wants to observe the live variations in sensors data, a request is sent to the server by the Android application. The server responds with an HTML page containing a graphic configured with the parameters we sent. The page is displayed in an Android WebView control.

After the plot is configured, a script makes periodic XHR (XmlHttpRequests) to the server requesting the current reading for a named sensor with in certain measurement unit. When the readings for all the sensors have been updated, the graphs are redrawn. The sensors data is encoded in JSON format:

```
name: <sensor_name>,
data: [<server_time>, <sensor_value>]
```

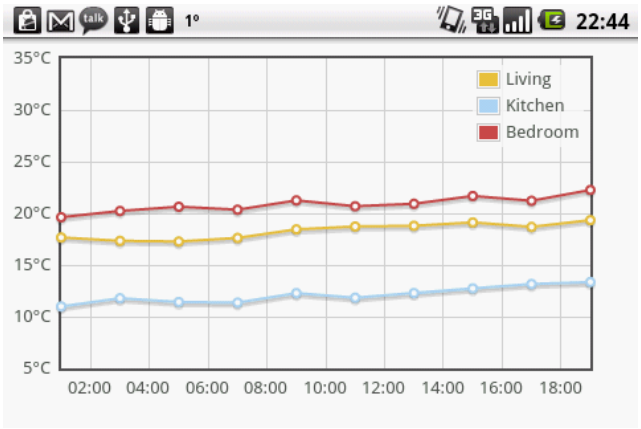

Figure 7. Android monitoring application

VI. Online Monitoring

We developed a method for monitoring and controlling a WSN remotely over the Internet, based on the successful MonALISA framework [2]. Our method uses an abstraction layer to provide remote monitoring and control to essentially any kind of WSN. MonALISA is a joint development of CERN, Caltech and UPB, typically used in monitoring large-scale systems such as computer clusters. It can be used to monitor and control any kind of system, including WSNs, as long as the appropriate interfacing software is available. In short, MonALISA employs repositories to which data

can be sent remotely using a portable software module named ApMon (Application Monitor) and to which users can connect with graphical client programs to view the data remotely. The connections can be established over the Internet, allowing user access to the WSN from any location, or over a local area network. Data in MonALISA is organized as parameter-value pairs pertaining to a ”host” or ”node”. Hosts are grouped into clusters, which are grouped into farms.

Each mote is presented as a host-type entity with a list of parameters. The name of the host is the IPv6 address and parameters include any sensor data that the mote provides (temperature, voltage, current). The user can filter parameters by name in order to concentrate on data of immediate concern.

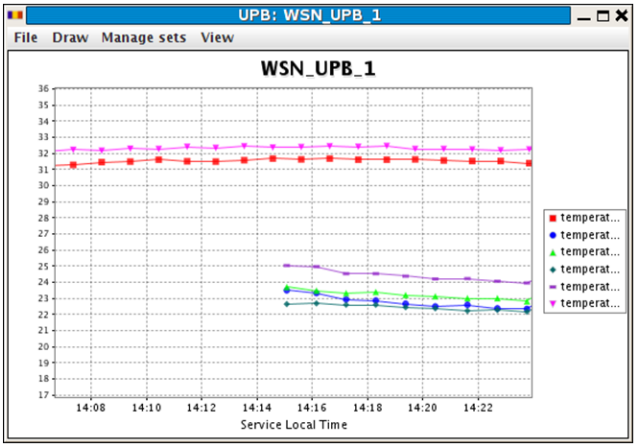

Figure 8. Data visualization with MonALISA

The parameters are sampled at defined intervals by the ApMon script and sent to MonALISA. The sensor data is made available on the graphic client in near real-time (delayed by storing the data on the various repositories), viewable from any point in the Internet.

VII. Related Work

Prior work exists in the field of Web-enabled Home Automation systems presenting both IP networks and WSNs[6]. The authors in [4] argue that Home Automation systems have a lot to benefit from using IP technology and integrating with the Web. As the sensor networks become part of the ”Web of Things”, they become much easier to use in other applications due to their Internet connectivity and standard interfaces. They present two project to argue this case, the first being an island of sensor nodes with RESTful interfaces, together with a mash-up editor in which a user can use their sensed data and actuator interfaces to create their own application. The second project involves obtaining real-time consumption data from an electricity outlet and displaying it on an IPhone.

While our system has characteristics that encompasses both of these projects, there are some key divergences: The authors interface the sensor nodes directly with RESTful interfaces, which we see as energy inefficient.

Another project [6] also uses IPv6 on sensor motes, but communication is mediated by a proxy server, which is more akin to our solution. This brings the advantages of connecting a WSN to the Internet without the drawback of increased traffic in the WSN by having the enhanced base station/proxy server mediate all the traffic.

VIII. Conclusions

The future work we envision for this system involves setting up the actuation infrastructure. Since the REST-like interface is already in place, this would only require tweaking the communication between motes and gateway to include this feature. Our customization interface can then be extended to include automation rules.

References

[1] Adam Dunkels, Bjorn Gronvall, and Thiemo Voigt. Contiki - a lightweight and flexible operating system for tiny networked sensors. *Local Computer Networks, Annual IEEE Conference on*, 2004.

[2] H B Newman et. al. Monalisa: A distributed monitoring service architecture. Technical report, CERN, Geneva, Jun 2003.

[3] Jianping Song et al. Wirelesshart: Applying wireless technology in real-time industrial process control. *Real-Time and Embedded Technology and Applications Symposium, IEEE*, 2008.

[4] Matthias Kovatsch, Markus Weiss, and Dominique Guinard. Embedding internet technology for home automation, 2010.

[5] Geoff Mulligan. The 6lowpan architecture. In *EmNets ’07: Proceedings of the 4th workshop on Embedded networked sensors*. ACM, 2007.

[6] L. Schor, P. Sommer, and R. Wattenhofer. Towards a zero-configuration wireless sensor network architecture for smart buildings. In *Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings*. ACM, 2009.

[7] Dan Tudose. Sparrow v2 specifications, 2011. URL `http://elf.cs.pub.ro/pm/wiki/media/sparrowv2.pdf`.