FONDUL SOCIAL EUROPEAN
Investeşte în oameni!
Programul Operațional Sectorial pentru Dezvoltarea Resurselor Umane 2007 – 2013
Proiect POSDRU/6/1.5/S/19 – *Pregatirea competitiva a doctoranzilor in domenii prioritare ale societatii bazate pe cunoastere*

UNIVERSITATEA **POLITEHNICA** DIN BUCUREŞTI
Facultatea: Automatică și Calculatoare
Catedra Calculatoare

Nr. Decizie Senat 111 din 15.09.2011

TEZĂ DE DOCTORAT

Rețele wireless de senzori în medii cu constrângeri de energie
Wireless Sensor Networks in Energy Constrained Environments

Autor: Ing. Dan Ștefan Tudose

COMISIA DE DOCTORAT

Preşedinte	Prof.Dr. Ing. Dumitru Popescu	de la	Universitatea Politehnica din București
Conducător de doctorat	Prof.Dr. Ing. Nicolae Țăpuș	de la	Universitatea Politehnica din București
Referent	Prof. Dr. Ing. Sergiu Nedevschi	de la	Universitatea Tehnică din Cluj-Napoca
Referent	Prof. Dr. Ing. Lucian Vințan	de la	Universitatea Lucian Blaga din Sibiu
Referent	Prof. Dr. Ing Valentin Cristea	de la	Universitatea Politehnica din București

Bucureşti 2011

In recent decades, technology opened the path to device miniaturization and specialization, which enabled the presence of an ever increasing number of embedded computing systems which interact, enhance and blend into a person's everyday life. This thesis presents research in the field of Proactive Computing and Wireless Sensor Networks with contributions in energy harvesting, energy efficiency and autonomy, task scheduling algorithms and monitoring of sensor networks, culminating with the development of a framework for sensor network management.

The theoretical research presented in this work has been validated by the contributions made to three projects: Sensei, integrating the physical world with the digital world of the network of the future; EUWB - Coexisting Short Range Radio by Advanced Ultra-Wideband Radio Technology; and TWISNET, Trustworthy Wireless Industrial Sensor NETworks.

Acknowledgements

I would like to thank my advisor, Professor Nicolae Țăpuș for coordinating my research during these three years. His advices and guidance on how to produce scientific results and accurate research have proven invaluable and I believe that without him this thesis would not have been possible. I am also thankful for the excellent example he has provided as a successful researcher and professor.

I would also like to thank Emil Sluşanschi for giving me the opportunity to be a part of the embedded systems research team. During these years we've had many interesting talks and I have come to cherish his advice and his friendship.

A big thank you to my friends and colleagues Andrei Voinescu and Alexandru Olteanu with whom I collaborated closely and debated most of the ideas in my PhD thesis as well as many, many other issues and problems. You are the best friends anyone could ask for.

This thesis would not have been possible without all the research assistants and students I have collaborated with during the past years. Thank you, Dan Dragomir, Adriana Drăghici, Răzvan Tătăroiu, Laura Gheorghe, Silvia Stegaru, Simona Poilinca, Madi Petrăreanu, Dumitrel Loghin and Adrian Bostan. I would also like to thank Traian Pătrașcu for his help in developing the mobile air pollution monitoring system. It was an original and interesting project which would have probably looked totally different without his contributions.

I would also like to thank Professor Sven Zeisberg for welcoming me at HTW Dresden and coordinating my research during my short but prolific internship. Those four months were a pleasant experience and I got to work with a very dedicated and ambitious team. I collaborated closely with Markus Wehner, Erik Mademann, Christoph Götze, Christoph Posenau and Frank Stephan and I would like to thank all of them for their support, suggestions and also for their attempts at teaching me a bit of German. Danke für alles.

To my friends, Mihai and Andreea, with whom I've spent some of the best moments during these years. I wish you the best of luck in your life together.

However, nothing would have been possible without the help of my family. Your unconditional support during all these years is what kept me going. No thanks will ever be enough.

Contents

List of Figures

List of Tables

Introduction

In 1991, Mark Weiser[1] predicted a 21st century where everyday personal computers would be replaced by a considerable number of embedded networked devices which would be completely integrated into our environment up to the point where they would become unnoticed, or even invisible to the user.

Wireless Sensor Networks are a technology that can offer a significant contribution in completing Weiser's "ubiquitous computing" paradigm and should represent a new revolution in computing, as were the mainframe and the personal computer before them.

Growing importance of context-awareness as an enabler for more intelligent, invisible and autonomous applications and services has highlighted the need for a greater integration of the physical with the digital world. Energy in particular is becoming an increasingly important topic in our lives. As we become more aware of the limitations and the costs of the energy we consume in our daily life, in our personal environment, we look on technology to give us aid in optimizing our efficiency.

Wireless Sensor Networks are subjected to severe constraints which are typically application-dependent. Constraints usually fall in, but are not restricted to, categories such as size, number of nodes, energy availability and processing capabilities. However, the prevailing constraint in almost all sensor network applications is network autonomy, that is, the network should be able to organize, manage and repair itself with minimum or no need for human intervention.

The goal of this thesis is to design highly energy and spectrum efficient mechanisms and protocols to capture and actuate the context information. The research focuses on enabling Wireless Sensor Nodes to achieve energy independence by harvesting and efficiently using energy from the environment.

Wireless Sensor Network applications are diverse and incorporate a large range of requirements for services, deployment areas, topology, lifetime, reliability and so on. Therefore, the first goal of this thesis was to extract and define the components and requirements that are generic and pertain to all sensor network classes. In order to better formulate our research goals, it is also of value to categorize sensor islands into classes which share the aforementioned characteristics.

Optimisation techniques found in this thesis are not confined only to software frameworks and algorithms. A study of hardware techniques in lowering power consumption has also been made with emphasis on implementing smart power management algorithms and circuits at sensor node level.

Also, research has been made in the direction of energy harvesting systems by studying four of the most promicing harvesting sources and adapting such circuitry to a wireless sensor node in order to prolong its lifetime, in some cases indefinitely.

The greatest contribution of this thesis is the development of a sensor network management framework which offers monitoring and actuation facilities for heterogeneous Wireless Sensor Network islands across different organizations. The suite solves issues that plague all modern sensor networks such as monitoring of different parameters, task alocation and energy profiling by implementing task scheduling algorithms based on multi-hop routing schemes and by supplying a flexible monitoring infrastructure that can be easily configured and adapted for any existing sensor network infrastructure.

Thesis Plan

Chapter 1 represents a state of the art in the field of wireless sensor networks. We describe the main architecture of a wireless sensor node and the characteristics shared by all sensor networks. Further, we go in depth with a description of sensor node components with an emphasis on current node architectures and a state of the art on power consumption for each separate subsystem. A survey on sensor network protocols and middleware is also present, with an emphasis on the two main protocols deployed on sensor nodes. The chapter finishes by setting the design goals for a low-power oriented wireless sensor network architecture.

A classification of wireless sensor & actuator networks is presented in **Chapter 2**. In this chapter we identify the generic components that pertain to all sensor networks, such as place of deployment, attached infrastructures and lifetime constraints. Based on these properties, we categorize sensor networks into three major groups: Environmental Sensor Networks (ESN), Community Sensor Networks (CSN) and Body Sensor Networks (BSN).

In this chapter we also focus on modeling sensors and actuator networks by classifying them into two main categories: the ones defined by standardization entities using XML or text-table values; and others using ontologies.

Chapter 3 delves into the topic of energy harvesting and how to apply it to a wireless sensor node. The study aims at applying a mathematical model of energy harvesting and

determining the efficiency of the most promising energy harvesting sources known to date, such as piezoelectric, thermal, photovoltaic and radiofrequency harvesters. Careful consideration is put into modeling each harvesting system and conducting experiments in order to determine the feasibility of deploying harvesting capabilities on a wireless sensor node. Also, in this chapter we model current energy storage systems that are available for sensor nodes and study their efficiency. A special emphasis is put on new energy storage systems, such as super-capacitors, and the possibility of being a better alternative to conventional batteries is being assessed.

The Sparrow sensor node platform is presented in **Chapter 4**. Sparrow is a wireless sensor network architecture that has been built as a research platform for the energy harvesting techniques described in the previous chapter. It was also used to deploy and test a series of wireless applications including IEEE 802.15.4 [2], 6LoWPAN and ZigBee [3] networks. In this chapter we focus on implementing energy harvesting circuits into a wireless sensor node and we evaluate their contribution in increasing the total functioning time for the node.

Chapter 5 details the Check Management Framework, which offers a monitoring and actuation framework for heterogeneous WSAN islands crossing the borders of different organizations, and having different network setups. One of the main components of the frameworks is an unconventional scheduling algorithm in which the main constraint is not time, but energy. As sensor networks are rarely subjected to hard deadlines, a more elegant approach is to design a scheduling algorithm that prioritizes energy consumption and task affinity. Check also offers a centralized monitoring, control and reconfiguration framework, which works toward the realization of the scalable internetworking, horizontalization and heterogeneity design goals. The last component provided by the Check Management Protocol Suite is the availability of a high-level, service-oriented self-healing strategy.

In **Chapter 6** we present a series of wireless sensor network applications that have been deployed in order to validate the concepts of the research stated in the previous chapters. The first application is a deployment in a residential environment and studies the components and services a sensor network oriented on home automation needs to offer.

The second application studies the deployment of mobile sensor nodes in an urban environment with the specific task of gathering data and relaying it to a central coordinator. The purpose of the project is to measure environmental sensor data, such as air pollution and contaminants linked to automotive exhaust and make it available to the general public via an intuitive web interface.

Chapter 7 presents in a concise form the conclusions of the research done in the thesis and elaborates on future work. The main contributions of the study to the field of wireless sensor networks and embedded systems in general are presented, with an emphasis on the algorithms and developed hardware platforms for energy harvesting.

Chapter 1

Wireless Sensor Networks

Wireless Sensor Networks (WSNs) or more generally Wireless Sensor and Actuator Networks (WSANs) are employed in a multitude of data acquisition, data processing, and control applications. Their advantages over traditional wired sensor and actuator networks include node mobility, increased reliability (due to the possibility of adaptive multi-hop data routing), easier installation and lower deployment cost. There are situations where wired data acquisition networks are impractical, such as environmental monitoring over large areas, or „intelligent" wearable devices forming so-called Body Sensor Networks.

Wireless Sensor Networks consist of autonomous devices that are deployed in order to monitor and measure specific parameters of a geographical area. Initial development of Wireless Sensor Networks has been done by the military and consisted mostly of surveillance and battlefield management applications. Civilian applications soon followed with applications such as environmental monitoring, home automation, healthcare, and traffic management.

Measured parameters are typically environmental, such as light intensity, humidity, pressure or temperature. The advantage of WSNs over traditional measuring systems is that multiple measurements of the same parameters from different locations can be taken simultaneously by the different nodes. This contributes to a better understanding of the monitored phenomenon by offering a spatial distribution of measured data.

The network is usually composed of multiple nodes that can be identical or have different purposes and capabilities. As a general rule, all nodes must have processing capabilities in order to sample data from sensors and a means of forwarding gathered data through the network. This is usually achieved by integrating a low-cost microprocessor or microcontroller into the design along with a radio or infrared transceiver. Also of note for the sensor architecture is the energy supply and storage system, which power the node's electronics. Batteries of different sizes and chemistries have traditionally been used for this purpose, although there is a recent trend in

replacing them with alternate energy delivery or storage systems, such as photovoltaic panels or super-capacitors.

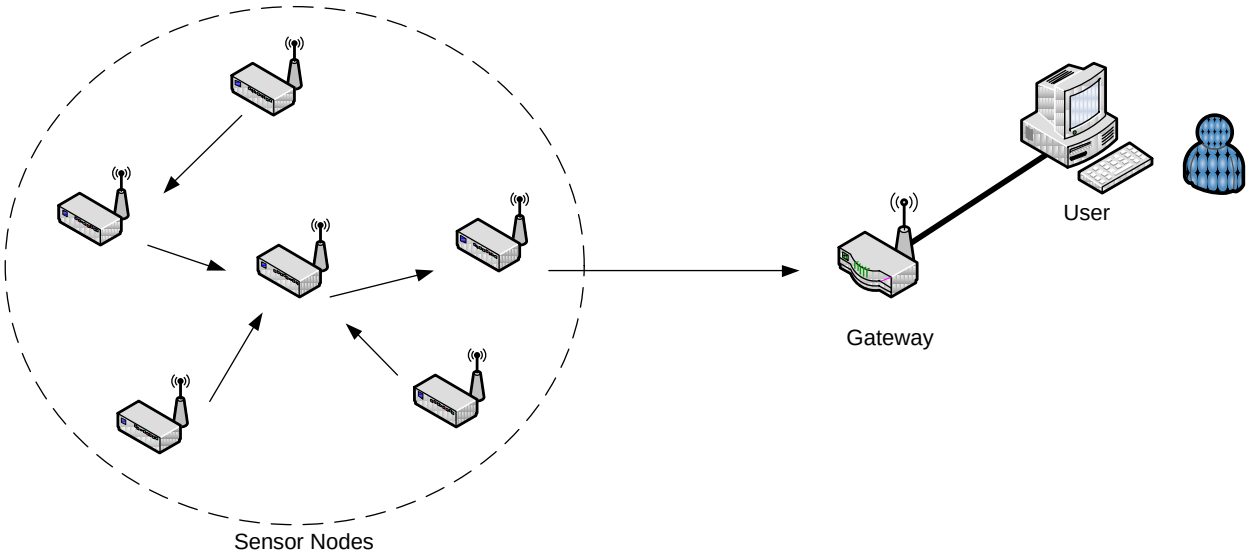


Figure 1. Typical WSN architecture

Size of nodes can vary, from large shoebox-sized ones to nodes the size of a grain of salt, although no practical functioning of a node of such small dimensions has ever been achieved.

Another important parameter for sensor nodes is the cost, which varies from a few euros to hundreds of euro, depending on the hardware and firmware capabilities of the node and the scale of deployment. However, there are some characteristics that are shared by all wireless sensor networks, such as:

- *Wireless communication*. Nodes form single or multi-hop networks.
- *Ability to withstand harsh environmental conditions.*
- *Fault tolerance*. The network employs hardware and software fault-tolerance algorithms in particular for the wireless communication channel.
- *Low-power architecture*. Most wireless sensor nodes have finite or limited energy supply.
- *Node mobility*. There is a wide range of applications, like body sensor networks, where nodes are not stationary.
- *Heterogeneity of nodes*. Not all nodes have the same functionality, measure the same parameters or even have the same hardware architecture.
- *Communication constraints.* Due to the shared radio communication environment, packets can be lost.
- *Large scale of deployment*. Typical WSN monitoring scenarios can employ hundreds to thousands of nodes spread over a large area.
- *Unattended operation*. The network must be self-sufficient and self-reconfigurable.

The limitations in cost and size of wireless sensor nodes can lead to similar limitations on resources such as memory size, energy storage capacity, processor speed and bandwidth. 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.

Sensor network nodes form a wireless network with a topology that can vary from a single-hop star to a multi-hop tree or a mesh. Outgoing network traffic is directed towards one or multiple network sink nodes, also known as gateways, which provide the network link to a standard computing network, such as a LAN or the Internet which leads ultimately to the end user. In some network architectures, the gateway nodes are different from the ordinary nodes because they posses more energy and communication resources.

1.1 Power Consumption in Sensor Node Structures

Also known as motes, sensor nodes are the main components of a WSN. They have limited communication, sensorial and processing capabilities which arise from limitations in hardware cost and energy supply. As the name implies, the main role of a sensor node is to measure parameters from its surrounding environment, do a minimum of on-board processing and forward the resulting data to its network peers. The communication can function the other way around, with motes receiving commands from supervisor entities in the network.

This thesis focuses on power consumption optimization techniques for wireless sensor networks. Thus, the following discussion will focus on the power consumption aspects for all sensor node components. The typical architecture of a sensor node is shown in Figure 2.

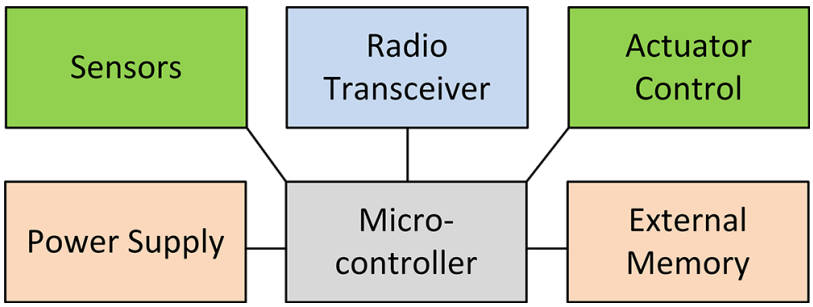


Figure 2. Structure of a typical WSN mote

The main components of a sensor node as seen from the figure are the microcontroller, transceiver, external memory, power source and one or more sensors. WSN nodes are built by multiple vendors and may vary in size, power consumption, microprocessor architecture or sensor interfaces (Figure 3).

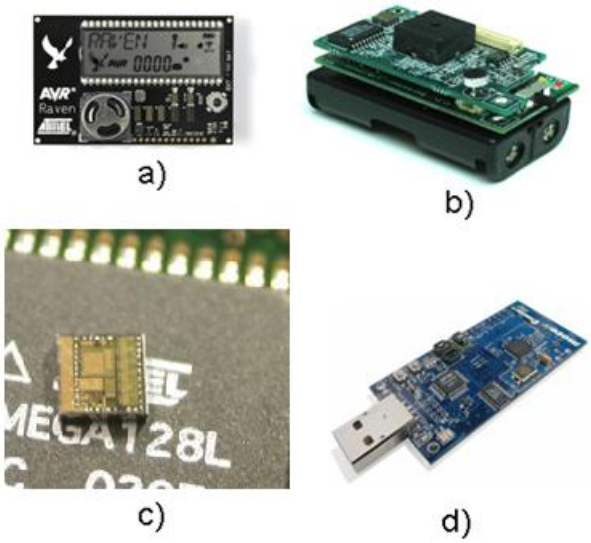


Figure 3. Different WSN nodes: a) Atmel AVRRAVEN, b) Berkeley Mica mote, c) Berkeley Spec mote, d) Tmote Sky

1.1.1 The Microcontroller

Microcontrollers are the preferred choice for integration in a sensor mote on account of their low cost and good computational capabilities. Inside a node architecture, their role is to coordinate and control the functionality of each component and to process data.

Apart from microcontrollers, there are other available alternatives, such as microprocessors, Field Programmable Gate Arrays (FPGA), Digital Signal Processors (DSP) or Application-Specific Integrated Circuits (ASIC).

Each of the four technologies has its advantages but it's mainly their disadvantages that made them unappealing for use in wireless sensor nodes.

Microprocessors generally have higher power consumption than microcontrollers which they balance by having higher processing speeds. However, most sensor network applications handle only tiny amounts of data even when compared to a modern tablet or netbook. Large amounts of computing power are therefore unnecessary and most motes employ cheap 8-bit microcontrollers as their central processing unit. Also, a significant disadvantage is the lack of integrated peripherals and communication interfaces, which are present in almost any modern microcontroller.

Field Programmable Gate Arrays have the advantage of reconfigurability. Node hardware can be reprogrammed or updated as easily as with a software update, which could prove useful in a situation where nodes need such facilities. However, FPGAs have high energy consumption requirements and reprogramming them can take a significant load on the battery pack. Also, most applications do not require such high degrees of flexibility in reprogramming as most of their functionality can be added or modified with simple firmware upgrades.

Digital Signal Processors can handle ample amounts of data and are suited to high bandwidth applications. However, wireless sensor nodes rarely need such processing power as communication bandwidth is severely limited in order to conserve energy. Therefore, DSPs are not an advisable choice for wireless sensor nodes.

Application Specific Integrated Circuits are designed for a specific application and cannot be modified or reprogrammed as their counterparts. Their advantage is relative low cost when produced in large volumes, but their lack of flexibility comes as a pitfall.

CPU	Supply Voltage [V]	Power Active [mW]	Power Sleep [μW]	Sensor Node
8-bit CPU				
ATtiny 13V	1.8-5.5	0.43	1.8	
Atmega1284p	1.8-3.6	0.72	1.26	AVR Raven
Atmega128RFA1	1.8-3.6	33.48	0.45	Sparrow v3
Atmega1281	1.8-5.5	0.9	0.18	Sparrow v2
ATmega128L	2.7-5.5	28	83	Mica, Mica2Dot, Mica2, BTnode
ATMega103L	2.7-3.6	15.5	60	Mica, IBadge
PIC18LF8722	2.0-5.5	5.6	2.4	GWNode
PIC18F452	2.0-5.5	40.2	24	EnOcean, TCM
CC1110	2.0-3.6	44	1.2	Monnit WIT
16-bit CPU				
MSP430F1612	1.8-3.6	3	15	PowWow
MSP430F1611	1.8-3.6	3	15	T-Mote Sky
32-bit CPU				
AT91SAM7x256	1.8/3.6	79.2	46.8	deRFNode
ATSAM3S4B	1.8/3.6	117	89.2	deRFusb Stick
IntelPXA255	1.8/3.6	2598	45100	Stargate

Table 1. Power consumption for some common CPUs

Typical mote CPUs are based on CMOS logic processors which can operate from a low frequency range (e.g. 1-100kHz) to 20MHz or more for input voltage levels as low as 1.8V. However, high operating frequencies can only be achieved at the cost of higher current consumption which typically increases at a rate of 1mA/MHz. With this current consumption limitation, the prospect of having a mote run at full capacity for a long period of time is not achievable [4]. Therefore, the universally adopted approach to solving this problem is to keep the microprocessor and the entire mote in a low-energy mode (e.g. idle or sleep) for most of its life, in order to conserve battery power.

Most MCUs have an internal timer or an external interrupt line that can be used to periodically enter or exit a low-power state. This allows the microcontroller to enter a program of alternating sleep-wake cycles that can be periodically executed. All sensor data acquisition, on-board processing and communication is limited to the wake interval, while in sleep mode, only the timer or event system necessary to wake the processor is active. Given the fact that the parameters that are typically monitored by a sensor mote are slow-evolving (ambient temperature, light, humidity etc.) and the amount of data sent in the network can be limited by protocol to tens or hundreds of bytes per minute, the ratio between sleep and wake times in a period can be increased to around 99% of the entire period for most applications.

Typical MCUs used in sensor motes are 8, 16 or 32-bit microcontrollers that run on CMOS logic and have reasonable amounts of flash, EEPROM and RAM memories attached along with analog circuitry such as ADCs and radio transceivers. Therefore, modeling the power profile for such a complex integrated circuit is not a trivial task and is closely linked to the current drawn by said circuitry from the power supply or battery.

The supply current in active mode can be split in two separate components: dynamic current, given by the sequential logic and static current. While the dynamic current is a linear function of the clock speed and can be summed up by the two terms in the above equation, the static current is given by the analog circuitry, memories and leakage currents and is totally independent of the operating frequency.

The complete formula for power consumption of a CMOS logic circuit is:

$$P = \frac{1}{2} ACV^2 f + \tau AVI_{short} f + VI_{leak} \qquad (1)$$

where P is power, V is the supply voltage, f clock frequency, C capacity of output lines, A activity (number of logic transitions per clock cycle), I_{leak} leakage current, I_{short} short circuit current and τ short circuit period.

Drain current, while not a big contributor to power loss in CMOS logic, is a few orders of magnitude larger for analog blocks and Flash memory, which, at low clock speeds, can drain more power than the CPU. Depending on the technology, either one of the two currents can have more significance. Dynamic current decreases as integration scale increases, while, at the same time, leakage current becomes prevalent, due to having ever larger numbers of transistors per chip. Table 1 shows power consumption of the most popular CPU installed in some standard sensor nodes [5], [6], [7].

From the data in Table 1 we can conclude that sensor motes employ all types of microprocessors, from 8 to 32-bit architectures. While 4-bit microcontrollers were originally used for sensor networking, due to their low power requirements as opposed to contemporary 8 or 16-bit architectures, modern WSN motes employ mostly 8-bit or 16-bit microprocessors that have been specially optimized for low-energy consumption. Most of them feature at least two user programmable power states (e.g. *sleep* and *active*) and various power management profiles.

On average the power consumption for the active mode can range from 3 mW up to 30 mW, and in sleep mode can decrease to about 10 μW. Modern SNs use 16/32-bit CPU with larger number of power down modes, and are intended for multimedia data acquisition (voice, image). The power consumption of 32-bit CPUs in active mode is >100 mW.

1.1.2 Radio Transceiver

Sensor motes employ different types of wireless data transceivers, from ultrasounds and infrared to lasers and radio. While lasers have the advantage of range and infrared that of low power, radio transceivers have established themselves as standard for wireless sensor networks. Almost all radio devices use communication frequencies in free or ISM bands situated from 433MHz to 2.4GHz.

An important aspect of integrating a radio transceiver in a WSN mote is taking into consideration its power efficiency. Because of their design, transceivers consume almost the same amount of power when receiving or transmitting, which is usually three orders of magnitude larger than the average energy budget of a microcontroller. In order to conserve battery capacity, transceivers usually have one or two low-power states (idle or sleep) which allow them to remain dormant and be turned on quickly in case of data reception or transmission.

Type	Frequency	Rx power	Tx power	Status -Power down
	[MHz]	[mA]	[mA]	[μA]
nRF8001	2400	13	14.5	2
nRF24LU1+	2400	12.9	11.1	480
nRF905	433/868/915	12	9	2.5
CC1000	315/915	10	17	1
ADF7020-1	433-434	17.6	21	1
CC1010	315/433/915	11.9	10.4	29.4
CYWUSB6934	2400	69.1	57.7	0.24
MC13192	2400	30	37	500
nRF2401	2400	18	10.5	0.4
AT86RF230	2400	16.5	15.5	0.02
AT86RF231	2400	12.3	14	0.02
AT86RF232	2400	11.8	13.8	0.4
AT86RF212	868	9.2	17	0.2
STD302N-R	869	28	46/0	
MC13191/92	2400	37	34/0	1
ZV4002	2400	65	65/0	140

Table 2. Power Consumption for Radio Transceivers Commonly Used in WSN Motes

The mote radio transceiver circuit enables communication with other motes and to data sink nodes, such as gateways. Depending on several factors, such as modulation, transmit power, transfer rate and duty cycle, a wide range of radio communication circuits is available on the market. Table 2 shows some of the most commonly employed radio transceiver chips in sensor motes [5], [8], [9].

From the data transmission and power consumption point of view, transceivers can be configured to operate in one of the following states:

- *Shutdown* – the transceiver is off and only consumes power due to leakage current. Returning from this state to operational may take a long time (~ms).
- *Sleep/ Standby* – the mote is in a low-power state where its consumption is low and can return to full functionality very fast.
- *Listen* – the radio is in Rx state, where it listens for any incoming data packets. Due to this fact, power consumption is higher than previous modes.
- *Transmit* – radio circuits are in the Tx state. Power consumption is dominated by the RF amplifier stage.

Data in Table 2 has been compiled with radio transceivers that use the three most common ISM bands for sensor networking: 433.05 – 434.79 MHz, 902 -928 MHz and 2400 – 2483.5MHz. Radio modules have been classified by their power rating in Rx mode, due to the fact that in a typical application receive is prevalent over transmit by a ratio of almost 1000:1. The first class has a current consumption of less than 10mA, the medium-power group has a current rating between 10mA and 50mA and high-power modules consume more than 50mA.

1.1.3 External Memory

In most sensor motes, memory is present only as the microcontroller's on-chip Flash or RAM or, in some cases as external Flash. Memory resources for a node tend to be scarce due to costs and most applications even tend to minimize memory allocation due to the high energy cost of writing and reading data. Allocation is typically made with respect to data purpose, with separate segments for program memory and for user data or device identification.

Chip Code	Type	Capacity	Power Active [mA]	Power Sleep [μA]
AT25F512B-SSH-B	Serial Flash	512Kb	6	5
SST25VF020B-80-4I-QAE	Serial Flash	2Mb	20	20
SST25VF020B-80-4I-QAE	Parallel Flash	512Kb	5	1
FM21L16-60-TG	FRAM	2Mb	8	90
24AA01-I/SN	EEPROM	1Kb	1	1
24FC512-I/SM	EEPROM	512Kb	5	400
23K256-I/P	Serial SRAM	256Kb	3	4
IS62C1024AL-35TLI	Parallel SRAM	1Mb	25	5

Table 3. Current consumption for typical on-board memories for WSN sensor motes

Due to their high cost and especially their high power consumption, external memory chips are less pervasive in wireless sensor nodes. Most of the memory space required by the on-board processing is allocated on the microcontroller's internal memories, which does not impose an additional cost to the whole system.

1.1.4 Power Supply

Power consumption in a sensor node is shared between the microcontroller, the sensors and the radio transceiver, with the latter accounting for almost the entire node energy budget. It is estimated that the cost of transmitting 1Kb of data over a distance of 100m is approximately the same as that of executing 3 million instructions on a low-power processor.

There are two energy storage technologies available for wireless sensor nodes: conventional batteries and super-capacitors.

Due to their low-power architecture and low cost, most sensor motes employ batteries as a primary energy source. Energy harvesting has also become a major trend in wireless sensor systems. Even if the amount of energy claimed from the environment is not significant, it can help extend the mote's battery life or, in some cases, even ensure perpetual functioning. Super-capacitors have also been employed as alternative storage elements, but their low energy density and high costs compared to that of batteries have not made them enter widespread use.

Batteries are electrochemical cells that store electrical charge via a chemical reaction. There are many types of batteries that use different types of chemistries, have different energy storage capacities and a wide range of sizes and costs. Therefore, not all batteries are suited to being used for powering a wireless sensor mote and certain features must be sought when choosing a good battery [10]:

- high energy density;
- large active volume out of the total packaging volume;
- low voltage per cell (0.5 – 1.0 V) so digital circuits can take advantage of the quadratic reduction in power consumption with supply voltage;
- ease of configuration into series batteries to provide a variety of cell potentials for various components of the system without requiring the overhead of voltage converters
- rechargeable, in case the system has an energy harvester.

Most battery chemistries used by wireless sensor motes fall into three categories: Nickel-Metal Hydride (NiMH), Lithium Ion (Li-Ion), and Lithium Polymer (Li-polymer).

All batteries have parameters such as voltage, charge cycles, energy density, charge time, maximum load current and discharge rate and they differ from one chemistry to another. Although all three battery types are well suited for use in wireless sensor networks, their unique characteristics can make one group more suited to a specific application. Finding the right choice is a process that involves knowing all application parameters and requirements, a process that very often boils down to finding the correct balance between price and performance.

The most relevant battery parameters are given in Table 4[12].

Type	Voltage	Energy density	Specific energy	Self discharge
	[V]	[Wh/dm^3]	[Wh/kg]	[percent/month]
Lead-acid	2.0 V	60-75	30-40	3-20%
Nickel Cadmium	1.2 V	50-150	40-60	10%
Nickel Metal Hydride	1.2V	140-300	30-80	30%
Lithium-Ion	3.6 V	270	160	5%
Lithium-polymer	3.7V	300	130-200	1-2%

Table 4. Battery Types

The following gives a short presentation of each cell chemistry characteristics:

Nickel-Metal Hydride (NiMH): nominal voltage per cell is 1.25 V and nominal charge capacities (C) range from 1100 mAh to 3100 mAh for standard AA cells. Useful discharge capacity is a decreasing function of the discharge rate, but up to around 1×C (full discharge in one hour), it does not differ significantly from the nominal capacity. Rechargeable batteries usually have 500 duty cycles per lifetime and less than 0.5 C optimal load current. Typical specific energy for NiMH AA cells is about 100 Wh/kg, and for other NiMH dry cells about 75 Wh/kg (270 kJ/kg), compared to 40–60 Wh/kg for Ni-Cd. Charge time can be up to four hours and they also exhibit a discharge rate of approximately 30 percent per month when in storage. NiMH Battery systems excel when lower voltage requirements or price sensitivity are primary considerations in cell selection. NiMH Systems can be configured with up to ten cells in a series to increase voltage, resulting in a maximum aggregate voltage of 12.5 V [11].

Lithium Ion (Li-Ion): Over the past years, Li-ion batteries have become widespread, largely due to lowering production costs and higher performance compared to NiMH. A Li-ion cell has a nominal voltage of 3.6 V and can offer around 1000 duty cycles per lifetime. Their average energy density is around 160 Wh/kg, which is almost two times greater than that of NiMH batteries of similar capacity. Their charge time is also reduced to less than four hours and typical discharge rate is approximately ten percent per month when in storage. These characteristics make Li-Ion battery systems a good option when requirements specify lower weight, higher energy density or aggregate voltage, a greater number of duty cycles, or when price sensitivity is not a consideration. Li-Ion battery systems can be configured up to seven cells in series to increase voltage, resulting in a maximum aggregate voltage of 25.2 V [11].

Lithium Polymer (Li-po): Li-polymer cells have similar performance characteristics when compared with Li-Ion cells, but have the advantage of being packaged in a slightly flexible form. However, this flexibility is often misleading, as Li-polymer cells should remain flat when installed in a device, not even bending for installation in the battery system. Characteristics of Li-polymer cells include a nominal voltage of 3.6 V, 500 duty cycles per lifetime, less than 1 C optimal load current, an average energy density of 160 Wh / kg, less than four-hour charge time, typical discharge rate of less than ten percent per month when in storage, and a semi-rigid form

factor. Li-Ion cells can be configured up to seven cells in series to increase voltage, resulting in a maximum aggregate voltage of 25.2 V [11].

1.1.5 Sensors

Sensors or transducers are the main sensing element of a wireless mote. They are responsible for converting a physical parameter like temperature or humidity into a measurable output that can be digitized and further processed by the node's circuitry.

Due to design constraints, the sensors that are suitable for a mote need to be small in size, low on consumed power and adapt to the environmental conditions. Also, in order to be deployed in large numbers, sensors need to have a low cost and must operate in high volumetric densities without interference.

Sensors can be classified into three categories:

- *Passive, Omni Directional*: Passive sensors gather input by monitoring their surrounding environment without modifying it. Usually, passive sensors require only little power and exhibit no preferred sensing direction (e.g. temperature or humidity).
- *Passive, Highly Directional*: These sensors measure the requested parameter from a specific area or volume and thus, have a sense of orientation or direction. A good example for such a sensor is a video camera or a directional microphone.
- *Active*: These sensors actively measure their environment by sampling or modifying their environment, for example radar induces radiation into an environment in order to sense its shape.

The vast majority of sensor nodes use only Passive Omni-directional sensors due to their low cost and low energy demands. Each of these sensors has a given coverage area for which it can accurately report the measured parameter values.

Sensors give motes the ability to measure physical parameters from the surrounding environment and translate them into data. Although there is a wide array of sensors that can measure parameters like temperature, humidity, light intensity, various gases, air flow, acoustic noise, proximity etc., not all of them are suited for deployment inside a wireless mote.

From the power consumption point of view, there are three major factors [13] that affect battery life: analog-digital conversion, signal conditioning circuitry and sampling and conversion

As sensor motes are usually small in size and have hard power consumption constraints, certain issues must be taken into account when selecting a suitable sensor and designing the interface circuitry:

- Power consumption
- Size
- Power cycling capabilities
- Compatibility with other circuits in the system
- Environment interface

Table 5 lists the power consumption of some common of-the-shelf sensors [5], [14].

Sensor	Type	Power consumption [mW]
micro-power		
MCP9700	Temperature	0.018
MCP9501PT	Temperature Switch	0.075
MPL115A2T1	Barometric Pressure	0.016
MMA8452QT	Accelerometer	0.1
MPR084Q	Proximity	0.082
SHT21	Temperature/Humidity	0.48
LIS302SG	Accelerometer	0.625
MCP9501PT	Capacitive Touch	0.01
BH1621FVC	Light sensor	0.231
RE46C190S16F	Smoke Detector	0.005
low-power		
FSS1500NGT	Force Sensor	5
IRA-E700ST0	IR Motion Detector	2
SL353HT	Hall Effect	1
HMC5843	Digital Compass	1.62
medium-power		
ADXRS450BEYZ	Gyroscope	30
OV7649	CCD	44
high-power		
34L 8D 103 W03263	Linear Position	125
FS5.0.1L.195	Gas Flow	900
EM-005	Proximity	180
S51-PA-2-C10-PK	Distance	350
OPB350W250Z	Level	200
IT321	GPS	110
TDA0161	Proximity	420
ultra high-power		
FS22	Water Flow Meter	15000
FR20-RLO-PSK4	Distance Meter	2000
BSP B010-EV002	Pressure	10000

Table 5. Power Consumption for Sensors Commonly Used in WSN Motes

Depending on sensor type and accuracy, power consumption can vary significantly, from low-power passive sensors, like accelerometers, temperature light and humidity sensors, to high-power transducers like ultrasonic level meters, digital imagers or industrial-grade pressure sensors, where power consumption can be one or two orders of magnitude higher than that spent by the radio transceiver.

Surveyed sensors presented in Table 5 were classified into five categories:

- Micro-power: power consumption is less than 1mW and their duty cycle can easily be changed through software.
- Low-power: less than 10mW of power and usually offer some sort of on-chip signal processing and conditioning.
- Medium-power: from 10mW to 50mW and can incorporate more extensive analog circuitry.
- High-power: have incorporated signal processing and standardized analog and digital interfaces. Power consumption for this group ranges from 50mW to 1W.
- Ultra high-power with a power consumption of more than 1W. They are usually unsuited for deployment on battery-powered motes, due to their high power consumption, but can easily be incorporated on a mote that is linked to the power grid or has some means of energy harvesting, such as photovoltaic panels.

1.2 Sensor Network Firmware

Hardware limitations give rise to specific software aspects. Software running on WSN nodes must be power-aware. Ideally the microcontroller spends most of the time in very low-power sleep states, and the radio transmits data in small bursts. Some radio chips used on WSN nodes draw more current when listening for data than when transmitting, therefore special low-duty-cycle communication protocols and algorithms are ideally employed, such that the radio is completely off most of the time. Although all the nodes in a WSN can share a radio channel, due to the low transmitter power only nodes in close proximity can communicate. From a software standpoint this is both beneficial (no interference between nodes spaced far apart) and problematic (the need for multi-hop data routing arises).

Traditional wired sensor networks usually employ one or more master nodes with generous hardware resources and energy available, such as a PC-type computer, therefore data processing usually happens on the master nodes. Wireless sensor nodes on the other hand are more autonomous because of the limited communication capacity between sensor nodes and master nodes. In some cases WSNs are purely peer-to-peer networks, lacking master nodes altogether. WSNs can perform data processing and aggregation inside the network, reducing the need to centralize large amounts of data. Given the fact that processing data on the nodes and forward the results between them can be energetically cheaper and more reliable than sending all

the raw sensor data to a central node, advanced WSNs function as distributed processing systems.

Wireless Sensor Networks are subjected to severe constraints which are typically application-dependent. Constraints usually fall in, but are not restricted to, categories such as size, number of nodes, energy availability and processing capabilities. However, the prevailing constraint in almost all sensor network applications is network autonomy, that is, the network should be able to organize, manage and repair itself with minimum or no need for human intervention.

In order to increase network autonomy, each component should be autonomic by itself, which is to say that sensor nodes need to have independent management capabilities for supervision and control over their resources. Such a manager should enable services for monitoring, analyzing, profiling and executing different application modules or tasks.

Network management systems should take into consideration the following requirements:

- *Self-healing*: detect and recover from failures or network attacks. This could be implemented at node level or as a service for the whole network. Modules that employ self-healing algorithms need to detect malfunctions or failures and apply a pre-established policy in handling them. This also applies to other exceptional states, such as network attacks or intrusions. The system should be able to cope with the failure and devise a strategy to reschedule tasks or reroute traffic in order to keep functioning.
- *Self-configuration and maintenance*: the network should adapt its parameters and respond to changing conditions in its surrounding environment. Typically, configuration is one of the first steps during network installation and can become very cumbersome if done manually for a large scale network of thousands of nodes. Therefore, the network should have at least a minimum of built-in self-configuration strategies that would enable it to achieve such actions as sensor calibration, network discovery, authentication and joining, network role assignment etc. Another important aspect of self-configuration is the node's ability to apply the same policies during its entire lifetime and respond to other factors, such as dynamic workload changing, energy-aware quality of service or ensuring that specific actions are done in a timely fashion.
- *Self-awareness*: enables the sensor node to discover and learn about its surrounding environment in order to fine-tune its behavior. Better understanding of these parameters can greatly improve node and overall network lifetime by altering its behavior in accordance to certain external or repetitive stimuli (e.g. knowledge of day-night cycles for a solar harvesting node or on how external temperature affects battery storage capacity). Such predictable and cyclic phenomena can lead to establishing rules on how to manage nodes inside a network or how to govern the interactions between them.

To sum up the above features, for a sensor node to be autonomous, it must implement three distinct tasks that need to run on a periodic basis:

-Monitor: features means of gathering data and process it from a managed resource. Such data can be related to energy status, bandwidth limitations, throughput, configuration parameters, neighbor advertisements and topology etc.

-Analyze: this function provides an algorithm on how to modify functionality depending on input gathered from the previous module, while respecting the predetermined role each node has been assigned. It can also be used as a predictive tool, to change node behavior in anticipation of certain exceptional events or to plan functionality depending on a given schedule.

-Execute: is the next step in the program while loop. It must run a given plan with consideration to the rules set by the previous module and also take into account any dynamic updates that might arrive from the network coordinator.

From a user's standpoint, a WSN must provide a number of services, such as reporting events of environmental pollution, reporting the formation of traffic jams in a city, identifying a person's urgent health problem, managing the air conditioning and lighting in a building, etcetera. There is no need for the end-user to receive real-time data from all sensors in the network, but only information that is relevant. Service-oriented WSNs use this approach - the nodes run software services that read and process large amounts of sensor data, as well as user commands, and send small amounts of relevant data back to the user, thus utilizing the radio channel efficiently. Service frameworks such as the Tiny Task Network (Titan) define tasks (services) that have a number of input and output pipes. The tasks are assigned by a scheduler to the available network nodes according to the node location, available sensors and actuators, node processor load and radio link quality, and the tasks' pipes are linked according to a service graph. More tasks can run on the same node, exchanging data through local pipes, or pipes can be connected between nodes. Tasks could also be moved between nodes in order to increase efficiency or in the event of a node or link failure. This functionality is transparent to the tasks.

From a software developer's standpoint, obtaining the lowest possible power consumption and the highest data link reliability is of significant importance. The engineer needs to be able to monitor the performance of the system when developing and testing software for WSN nodes. Indicators such as processor and memory load, wakeup frequency, duration of high-power states, amount of data sent and received over the radio, are important to the software engineer. These parameters need to be known for all network nodes. Higher-level, service-specific parameters, as well as lower-level parameters such as radio link quality, are also important for the scheduling algorithms in service-oriented networks and for self-healing or adaptive routing algorithms. Raw sensor data also needs to be monitored when debugging data processing software or when configuring the network after installation. Also, some simple applications only require obtaining periodic sensor readings on a central computer. Network administrators and users may also be interested in monitoring the health of the WSN nodes (such as the remaining battery charge), sensor data or performance metrics.

WSNs whose nodes are in inaccessible locations or spread over a large area clearly require software provisions to allow remote monitoring, without the need for physical access to the nodes. Even when WSNs occupy a small area and their nodes are easily accessible,

connecting a dedicated debugging interface to the nodes can be cumbersome and expensive when the nodes are in large numbers.

When using a WSN or when developing software, the user must also be able to control the network. Setting application-specific software parameters, controlling actuators directly, enabling and disabling services, upgrading the software running on the nodes are examples where control of the WSN is necessary.

Many real-time operating systems and network protocol stacks can run on WSN nodes, such as TinyOS, Contiki [15], Sensinode NanoStack, etc.

1.2.1 Wireless Sensor Network Protocols

A large number of industries benefit from reliable wireless sensor and actuator networks and a lot of effort has been made in the past years to develop WSN standards and protocols that address many of the constraints and problems mentioned in the previous section.

Out of these protocols, there are two that have garnered widespread adoption and use. Because of their low power specifications, we also consider them suitable in our research.

1.2.2 The ZigBee Protocol Stack

The Zigbee protocol stack (Figure 4) builds upon the IEEE 802.15.4 standard [16] that specifies the characteristics of the physical (PHY) and medium-access control (MAC) layer for low-rate personal area networks (PANs). The radio transceiver of a ZigBee device thus applies a direct-sequence spread spectrum (DSSS) scheme to transmit data at a rate of 250 kb/s if the radio is operated in the 2.4 GHz frequency band, and the MAC controls the access of the network nodes to the radio channel with an unslotted CSMA/CA protocol or, optionally, with a slotted CSMA/CA protocol if the network nodes are synchronized.

The ZigBee Alliance defined the higher layers of the protocol stack, which comprise the network (NWK) layer, an application support (APS) layer, the security service provider (SSP), the ZigBee device object (ZDO) and the application objects. The *NWK layer* is in charge of organizing a multi-hop network and routing data packets over it. The network can be organized in a star, tree, or mesh topology and is always centrally controlled by the ZigBee coordinator.

When a tree or star topology is used, packet forwarding is performed with a simple tree-based hierarchical routing algorithm while, in case of using a mesh topology, a simplified version of the ad-hoc on-demand distance vector routing algorithm is applied. The *SSP unit* provides a security service to the ZigBee network, which includes methods for ensuring freshness of data, message integrity, network and node level authentication, and encryption.

The *ZDO entity* provides the service to discover other devices and application objects in the network. The *APS layer* is responsible for binding together devices based on their service needs in order to exchange application messages between them. Finally, the user can define several *application objects* to implement its sensor application.

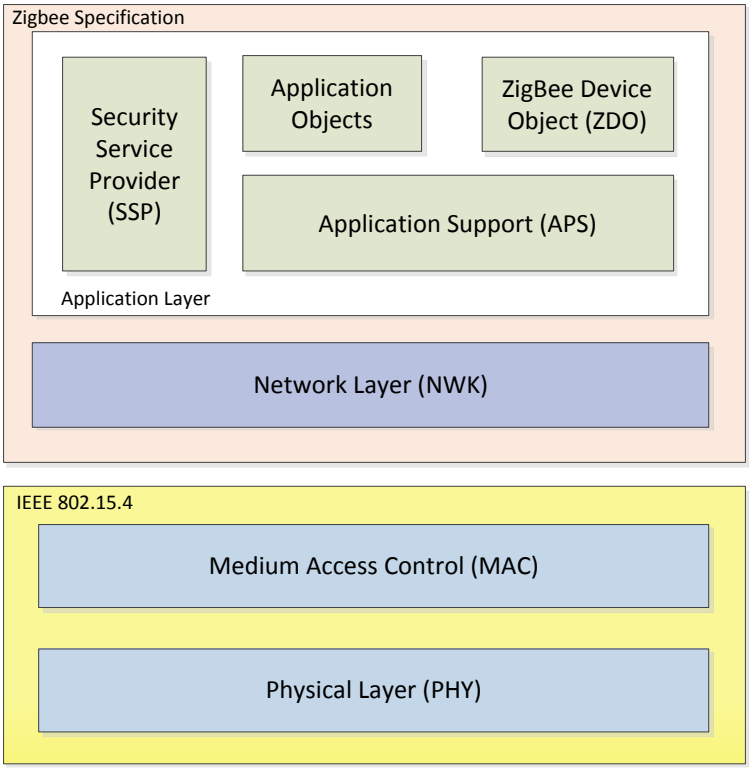


Figure 4. ZigBee protocol stack

1.2.3 The 6LoWPan Protocol Stack

IPv6 networking over low-power wireless personal area networks (6LoWPAN) is an IETF effort to enable IP based networking and compatibility for low-power wireless sensor networks. Because of the potential of direct compatibility with the existing Internet infrastructure, 6LoWPAN can be viewed as a significant factor in future sensor networks. It is also the most profound RFC clearly breaking the OSI layered model and it exploits cross-layer information to minimize protocol overhead. The 6LoWPAN effort will undoubtedly have an impact because it enables viable network and transport layer solutions for IEEE 802.15.4 type wireless personal area networks (WPANs) and it will be used for commercial WS&AN solutions.

Low-power wireless personal area networks (LoWPANs) comprise devices that conform to the IEEE 802.15.4 standard [2]. The 6LoWPAN RFC 4919 [17] gives an overview of LoWPANs and describes how they benefit from IP and, in particular, IPv6 networking. It describes LoWPAN requirements with regards to the IP above layers, and spells out the underlying assumptions of IP for LoWPANs. The main characteristics of LoWPANs are:

- small packet size (127 byte physical layer protocol data unit (PPDU)),
- support for both 16-bit short and IEEE 64-bit MAC addresses,
- low data rates (250 kbps, 40 kbps, and 20 kbps),
- star and mesh topology support,

- low cost, low power, and inherently unreliable,
- long sleep periods in many environments.

As the 6LoWPAN uses IEEE 802.15.4 technology, both full function and reduced function devices need to be supported. The application of IP technology is assumed to provide the benefits of

- allowing use of existing infrastructure,
- proven and open technology,
- readily available tools for management, diagnostics, and commissioning of networks,
- direct communications capability with other IP-based networks without a requirement for translation gateways or proxies.

However in order to benefit from the above aspects, problems relating to auto-configuration, large address space, IEEE 802.15.4 – IPv6 packet constraints, interconnectivity, low routing overhead, computational requirements, sleeping, limited management capabilities, service discovery, and security need to be addressed. The goals for 6LoWPANs have therefore been set according to priority:

- Provide a fragmentation and reassembly layer below the IP to support IPv6 packets (1280 octets) in an environment where the protocol data unit may be max. 81 bytes.
- Maximise the data portion of protocol data units by efficient header compression.
- Create IPv6 stateless address auto-configuration to reduce configuration overhead.
- Provide a routing protocol to support a multi-hop mesh network taking packet size constraints into account. Routing packets should fit within a single 802.15.4 frame.
- Reuse of existing protocols as much as possible especially for network management.
- Take into account implementation, application, and higher layer considerations.
- Take into account small code size, low power operation, low complexity, and small bandwidth requirements based security aspects (esp. confidentiality and integrity).

The RFC 4944 [18] provides a more detailed description on how to transmit IPv6 packets over the IEEE 802.15.4 network. The key functionality for this is called the LoWPAN adaptation layer, which resides between the IEEE 802.15.4 MAC layer and the network layer supporting IPv6.

An encapsulation header stack prefixes all LoWPAN encapsulated datagrams transported over IEEE 802.15.4. Each header in the header stack contains a header type followed by zero or more header fields. Figure 5 illustrates some of the header fields and how they relate to the 802.15.4 frame format. The destination and source addresses can be used for the mesh routing within the 802.15.4 network and they include personal area network (PAN) identifiers. If there are multiple LoWPAN headers in the same packet the must appear in the following order: mesh addressing header (mhop), broadcast header, and fragmentation header (frag) of Figure 5.

The dispatch (dsp) header begins with bit combination (01) followed by a 6 bit selector, which identifies the type of header immediately following it. In the case of Figure 5, the field would indicate (000010) corresponding to LOWPAN_HC1 (HC1) compressed IPv6 header.

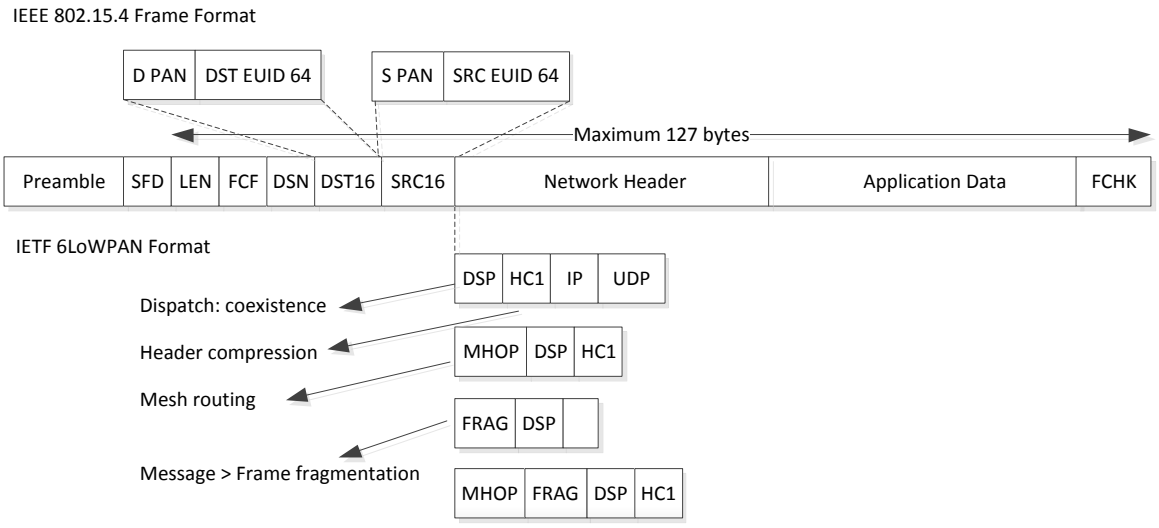


Figure 5. 6LoWPAN header format (Arch Rock Corporation [19]).

The mhop header begins with (10) followed by 1-bit "V" and "F" fields. The fields indicate whether the originator and destination addresses are 64-bit (0) or short 16-bit (1) addresses, respectively. A 4-bit field indicating the number of hops left follows them.

The frag header begins with (11) and is only present if the datagram does not fit into a single IEEE 802.15.4 frame. The first fragment starts with (000) bit field and contains the datagram size and tag. The subsequent fragments begin with (100) and contain the datagram offset in addition. Note that the datagram fragments do not need to arrive in order.

The common compressed header encoding HC1 is the key enabler of IPv6 over LoWPAN. With the help of HC1, the only IPv6 header that always needs to be carried in full is the Hop Limit (8 bits) field; often all the other necessary fields can be inferred from the rest of the frame. Use of HC1 is made possible by utilization of stateless address auto-configuration, defined either by RFC 2464 [20] or by link local addressing defined by 6LoWPAN.

Similarly, parts of the next header fields (UDP, TCP, ICMP) can be deduced from information available elsewhere in the frame. For UDP, only the length field can be deduced. Some of the non-deductable information can be compressed however reducing the UDP header up to 4 octets at best.

The method of derivation of Interface Identifiers from IEEE 64-bit MAC addresses is intended to preserve global uniqueness when possible. However, there is no protection from duplication through accident or forgery. A sizeable portion of IEEE 802.15.4 devices is expected to always communicate within their PAN (i.e., within their link, in IPv6 terms). In response to cost and power consumption considerations, and in keeping with the IEEE 802.15.4 model of reduced function devices (RFDs), these devices will typically implement the minimum set of necessary features. Accordingly, security for such devices may rely quite strongly on the mechanisms defined at the link layer by IEEE 802.15.4.

Chapter 2

Wireless Sensor Network Classification

Applications involving WS&ANs are very diverse which proves the fact that behind the term lie hidden a large range of requirements for supported services, topologies, deployment nature, lifetime, etc. Therefore, in the aim of defining generic components and requirements for WS&AN islands, it is worth categorizing them into a few classes spanning a set of common high level features and requirements. The set of generic classes is thus tentatively defined so as any application involving WS&ANs will either require a single class or will be decomposable onto a set of these classes.

We identified three generic WS&AN classes:

- Environmental Sensor Networks (ESN): mainly large scale, with severe lifetime requirements
- Community Sensor Networks (CSN): mainly medium scale, attached to infrastructures like buildings to serve a given user community
- Body Sensor Networks (BSN): mainly small scale, attached to persons and moving along with them

2.1 Environmental Sensor Networks

2.1.1 ESN Description

Environmental Sensor Network's purpose is strictly connected to the site they are deployed in: their main functionality is that of monitoring scenario properties by collecting sensor readings and reporting them back to a central server. Additionally, ESN can include actuators in order to realize control loops: for instance a typical application is that of monitoring ground humidity to activate the irrigation system when needed.

ESNs can be deployed in a large variety of different deployment sites and can run a multitude of application, hence their sensor nodes shows very different configuration. Also, it is not possible to define the typical installation environment for ESNs: in fact, they can be exploited for building integrity monitoring, as well as for agricultural management applications.

Environmental monitoring is among the earliest applications exploiting wireless sensor networks. In fact, the challenges to be tackled in this scenario are multiple: the network size may be very large, nodes may be placed in locations which are difficult to reach, power supply may be difficult or impossible to provide. Also, these networks must deal with the typical challenges of wireless channels (e.g., packet loss, random delay, scalability, etc.). Sensor networks offer a solution for this scenario as their main features are: unassisted functioning, energy efficiency, and high scalability. The main purpose of a monitoring sensor island is to collect environmental data in an energy efficient manner while reporting it to one or more central control entities.

Environmental Sensor Networks are not only designed to collect data from sensors deployed in the area, but they also include complementary functionalities like: providing connectivity through the network and enabling distributed services (i.e.: localization, event detection, etc.).

Usually, ESNs are organized hierarchically and consist of three types of devices: Resource Limited Nodes (RLN), router nodes and data gathering and processing point(s). Resource limited nodes are the most numerous entities of the network; their main goal is to monitor one or more environmental parameters, they are equipped with low power radio interfaces (such as IEEE 802.15.4), a limited power supply (which may be scavenged) and minimal computational resources.

Routers are equipped with both a more sophisticate radio interface (more reliable, larger bandwidth, more energy consumption) and with the RLNs low power radio interface. In addition, routers have a longer lifetime (either due to more capable batteries or to power connections). Routers are also in charge of connecting RLNs to the external world (often to an IP based network, where the data gathering point(s) reside) other than locally managing the network. In the following, with the term sensor cloud we mean the set of RLNs referring to the same router, and to sensor island as the set of sensor clouds managed by the same data gathering point.

The gathering point is usually the entity governing the sensor island in its entirety; it collects sensed data through routers, it issues commands to single (or groups of) RLNs and

performs global operations on the network such as configuration procedures, reprogramming, integrity check and so forth.

Figure 6 depicts the network architecture. The three-tiered structure consists of the data gathering point, a set of routers (center of the figure) and many RLNs (third tier on the bottom part of the figure). Each router is in charge of managing a single sensor cloud, however, a RLN may belong to multiple sensor clouds. This allows for some redundancy, which increases robustness in case of device/link failures.

The number of RLNs belonging to the same sensor cloud may vary with the density of RNs: as an example networks with many routers will likely have a low delay figure (and better performance in general) and high installation cost, while a limited number of routers will lead to a cheaper installation at the same price, however, exhibiting worse performance.

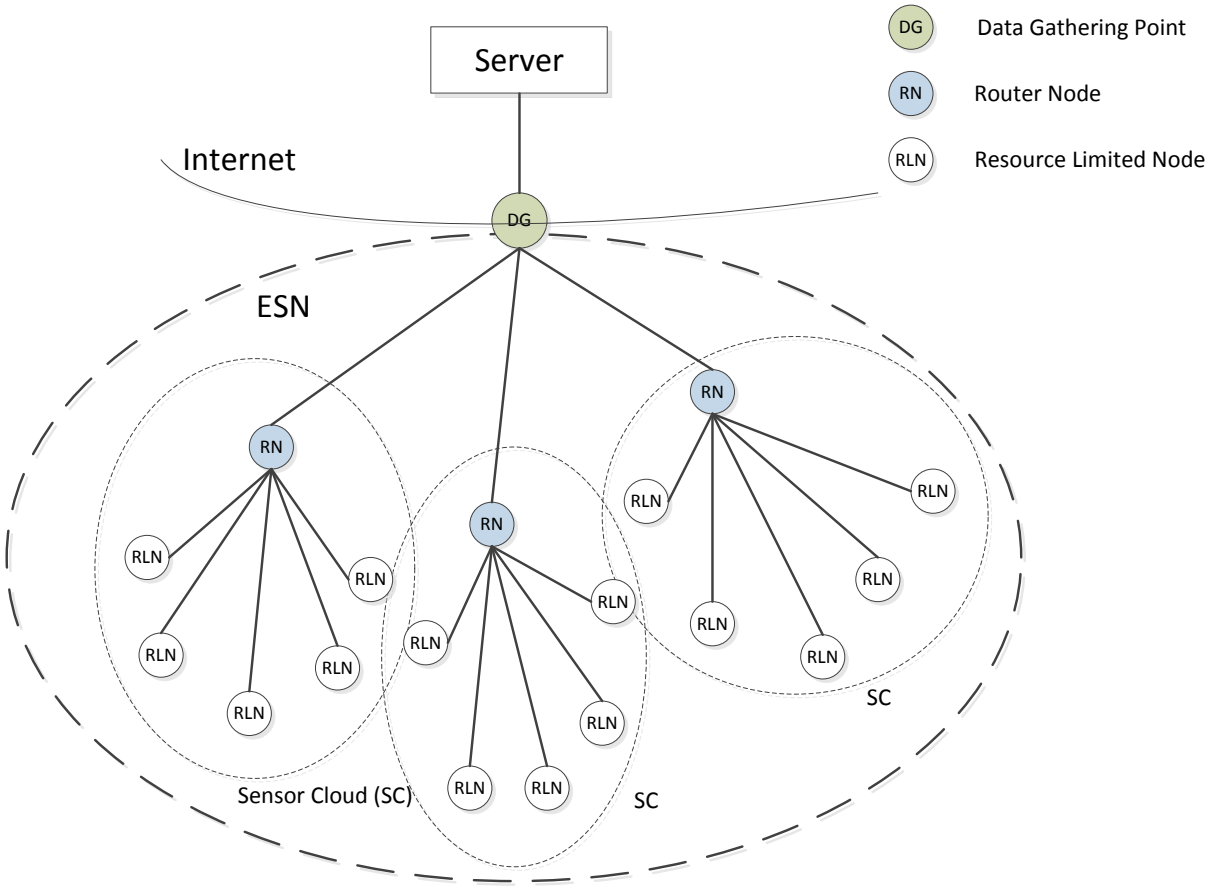


Figure 6. A typical deployment diagram of ESN. SC are sensor clouds, each containing a number of RLNs e.g., sensors and actuators. Router nodes (RN) are in charge of connecting sensor clouds to the data gathering point (DG).

In terms of security, for ESNs typically cheap sensors without any protection mechanisms against physical attacks are more likely to be found. An ESN contains a large number of sensor nodes so that a higher price of a single node will add up a significant additional cost to the whole WS&AN. It may however be assumed that some vulnerable nodes such as RN

or DG are tamper protected. Moreover, an ESN is likely to be deployed in a public, untrusted environment and physical tampering with nodes might therefore occur unnoticed. Those facts lead to a high vulnerability of ESNs to physical node compromise.

2.1.2 Resource limited nodes

As mentioned above, resource limited nodes are characterized by low-power radio interfaces and scarce energy supply. This, in turn, leads to low bit-rates, unreliable communication and the need for energy-efficiency in every performed operation.

The main operations performed by RLNs are: sensing and reporting data, operating an actuator, performing simple on-board processing (e.g., data aggregation and distributed algorithms), and forwarding packets. The RLNs belonging to the same network cloud may be heterogeneous in terms of computational power, energy supply and allowed operations. Here, we envision four types of RLNs: simple sensors, actuators, forwarders and hybrid. The last case indicates nodes having multiple capabilities (e.g., forwarder and actuator). The forwarding capability is needed when the network spans over multiple hops, and thus multi-hop routing is mandatory for connectivity support. Note that a single node can act both as an actuator and a sensor as long as a feasible schedule exists for the two tasks and the energy reserve of the node is sufficient.

2.1.3 Router Nodes

Routers are more powerful devices than regular nodes in terms of communication bandwidth, processing power and energy supply; their main objective is that of connecting RLNs to the external world. Communication between RLNs and routers is performed with the low-power radio, while communication between RNs and the data gathering point uses the most reliable connection available.

After the network is deployed, an automatic procedure associates each RLN with one or multiple routers (according to which a node will be typically associated with the closest router). In addition, during the discovery phase, a RLN can also maintain a backup router (i.e., to quickly recover from a loss of connectivity). When the discovery procedure ends, each router can store topology/connectivity information about its sensor cloud; this allows for the implementation of simple routing techniques such as source routing. However, should mobility be an issue of the environment, dynamic organization and routing protocols can be exploited to maintain the connectivity.

The main duties of sensor node routers are:

- collecting data from sensors,
- forwarding commands to actuators,
- configuring sensors,
- re-tasking network protocols,

- reprogramming RLNs within its own sensor cloud,
- updating its knowledge about the network topology/connectivity
- reporting malfunctions.

Routers can communicate with each other in order to check the condition of boundary nodes or cooperatively process the collected data (this can also be done at the data gathering point).

2.1.4 Data gathering points

One or more central gathering points are in charge of controlling the sensor island. If more gathering points are present, they should communicate with each other to maintain consistency among issued commands. This is useful in case of broken links.

In addition, data gathering points provide high level services to the system. In fact, they can act on behalf of any network node and they can combine together raw resources (e.g., sensor readings coming from different network clouds) to provide advanced resources (e.g., context dependent services).

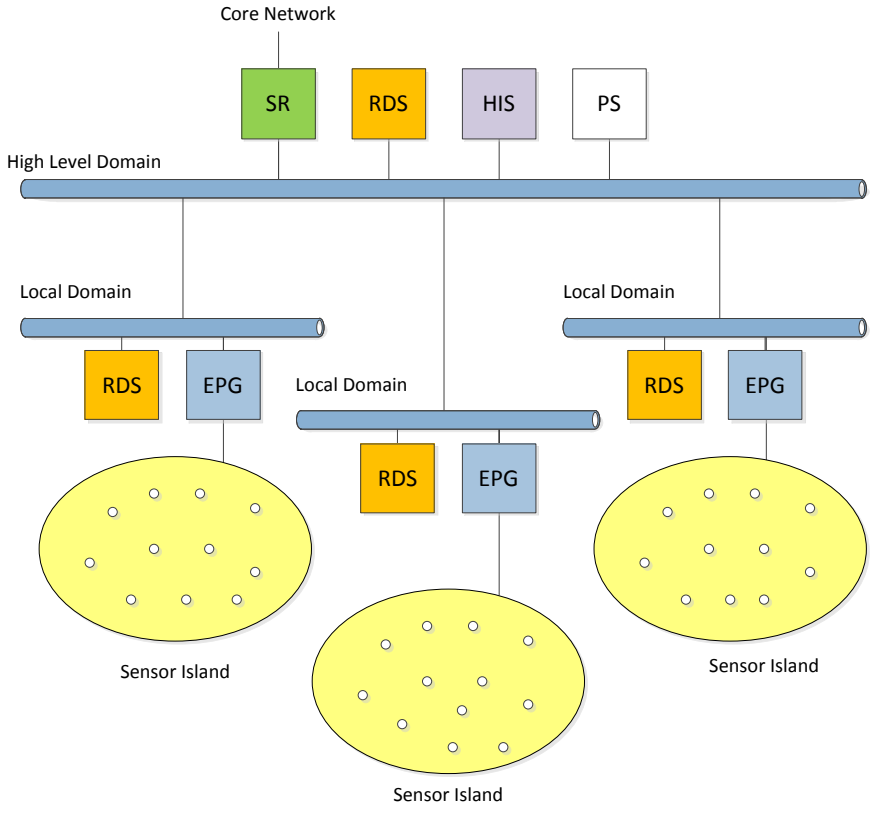


Figure 7. Example of a ESN, showing the main devices involved and the connection between them

If gathering points implement a standard resource interface, they are named End-Point Servers (EPSs); they can also include functionalities of a Processing Server (PS, to combine and process raw resources), a Resource Directory Server (RDS, keeps a list of available resources,

including pointer to their description, location and status) and a High-level Interface Server (HIS, to interface user's applications to the framework).

Figure 7 shows a possible deployment of an ESN: three local domains are present, each with both an End-Point Gateway (EPG) and an RDS. A high level domain connects the other three providing high level functionalities like HIS, PS, and RDS by means of a central station.

2.1.5 Environmental Sensor Network Features

In the following we identified the distinctive features characterizing the ESN class.

- **Network Size**. An ESN is composed of a hundred up to few thousands nodes. The resulting topology can include many hops and in many cases are organized hierarchically. The typical size of an ESN imposes highly optimized design solutions on the whole protocol stack. In particular, routing algorithms should be able to establish very long communication paths (10-20 hops).

- **Difficult access**. Due to their typical dimension, environment and configuration, ESNs require that their nodes can function properly without any human intervention. Due to nodes inaccessibility (or very high access costs), remote management functionalities are to be provided.

- **Connectivity**. ESNs can be exploited to provide limited connectivity to other neighboring networks, typically BSNs. Typical ESN connectivity consists in a data gathering protocol coupled with an interest dissemination tool in order to manage traffic to and from nodes and the central station. However, ESN can also be exploited to provide connectivity to neighboring BSNs. In such a case, communication protocols should be able to differentiate traffic types and adapt to topology changes.

- **Low processing capabilities and memory size**. ESNs are not required to support high-level operation and/or data stream. ESN nodes are generally equipped with very simple processing units. This impacts the possibility of realizing complex operation within the network. Also, memory limitation reflects on the quantity of data that can be cached in a node, thus affecting algorithm design.

- **Delay tolerance**. ESNs are subject to different level of delay tolerance, which depends mainly on the network objective. Since delays are subject to application needs they can be computed at design time and, as a consequence, protocol can be tuned accordingly. However, ESN objectives can vary with time (e.g., a change of the priorities or the needed accuracy of the monitored parameters), hence objective-oriented solution can save the day by adapting the network behavior to the different needs of the applications.

- **Energy Management**. Nodes in an ESN could not be powered via a power grid; hence, they should rely on batteries or scavenged energy. Environment Sensor Network deployment can

be realized on very different scenarios: varying from industrial locations, where a power grid may be available, to agricultural context, in which energy can be harvested, to extreme conditions, where nodes can only rely on batteries. This reflects on the overall design of node architecture: power saving modes must be available and exploited depending on the particular scenario.

- **Privacy and Security**. Data passing through an ESN can range from very sensitive to public data. However, in the stricter situations, security has to be addressed with very efficient methods, due to the scarce computational capabilities of the nodes. Data managed in ESNs range from confidential to public. Hence, a scalable security approach will be necessary in order to grant the desired confidentiality level, while keeping the processing and memory needs at a minimum.

- **Self-Healing**. This is a vital feature for long lasting, reliable networks; in fact, as an example, ESN should continue functioning even after that some of their nodes depleted their batteries. The impossibility to complete a task should be automatically reported to an operator. ESN must operate unassisted, hence a self-healing protocol must be able to correct erroneous behavior and adapt to topology/environmental changes. Also, critical situations should be promptly reported to operators.

- **Distributed data processing.** ESNs are intended for long term monitoring. Therefore, it is expected that they will gather large volume of data. To reduce the communication overhead and prolong the network lifetime, in-network and distributed data processing mechanism are considered as an important functionality of ESNs. Such online and local data processing also has advantages with regard to real-time detection of events and errors. To cope with energy constraints and reduce communication overhead intelligent aggregation and adaptive sampling mechanisms need to be in place.

- **Heterogeneity.** ESNs often include various types of sensors and sensor nodes with different capabilities. The devised techniques need to account for and cope with this heterogeneity. The devised techniques for ESNs need to account for, and cope with this heterogeneity.

2.2 Community Sensor Network

2.2.1 Community Sensor Network Description

A Community Sensor Network (CSN) consists of wireless/wired sensors, actuators and forwarders. These nodes are usually attached to buildings, static fixtures, or carried by human beings. Compared to ESNs, CSNs may be applied in more flexible environments such as in shopping malls, supermarkets, gyms, etc. The application purposes of a CSN in these environments may differ dramatically, e.g. its application in a restaurant may mainly manage the scheduling/booking system for customers whereas in a supermarket may mainly provide the guidance of shopping such as price and location of the goods. A CSN may also usually be involved into several environments simultaneously with frequent cooperation between them.

Nodes (sensors, actuators, and forwarders) in CSNs can vary from resources to capabilities. Though such nodes can occasionally be mobile, most of them are usually static and attached to buildings though not necessarily indoor.

In terms of security, CSNs can be similar to ESNs, namely large networks and built up from sensors without any protection mechanisms against physical attacks. The probability of attack also depends on the environment where the CSNs are deployed. Usually, CSNs will often be deployed in partially protected environments, letting the physical danger to nodes be lower than in ESNs.

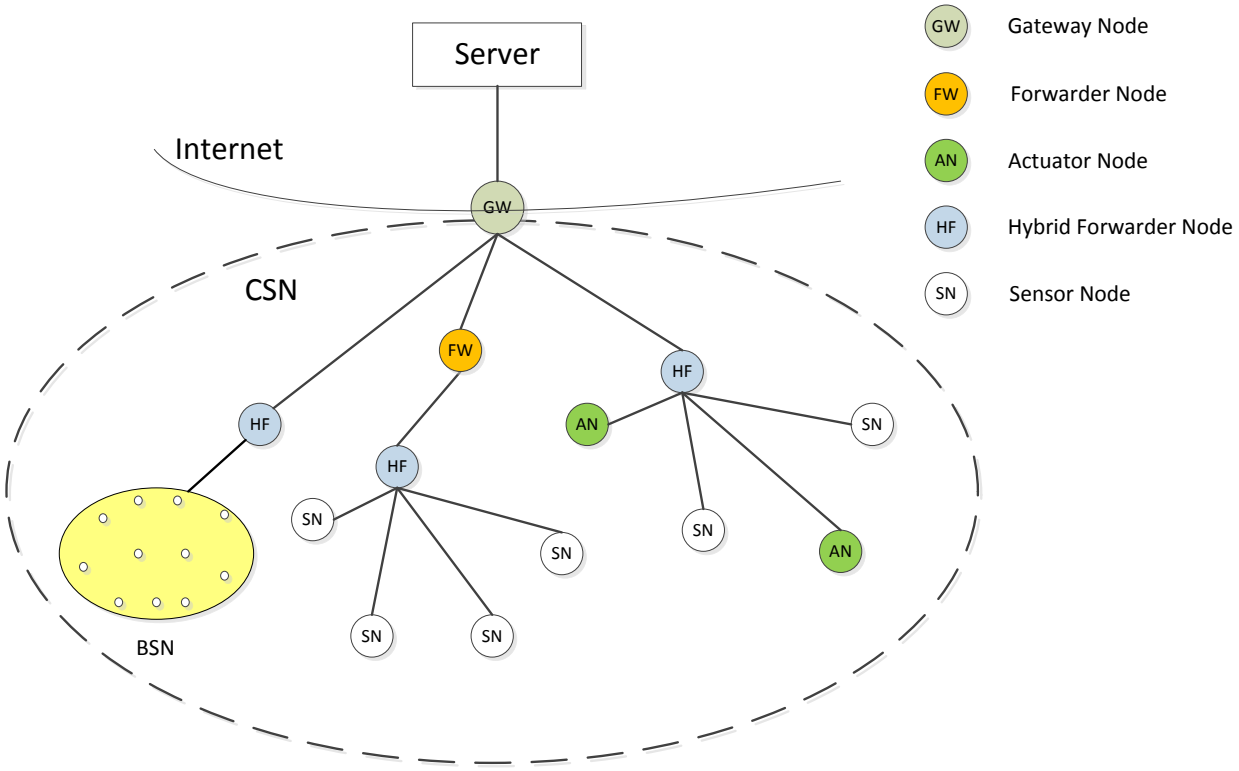


Figure 8 A typical deployment diagram of CSN

2.2.2 Community Sensor Network Features

The nodes in a CSN can work together to provide a single service, but there may also be several different services, which combined provide the required service to a community. The main difference to for example an ESN is that the same network can be used by multiple different applications by non-connected users in order to aid the operations of a community, e.g. a hospital. Figure 8 presents a typical deployment diagram of a CSN.

The following features of CSNs are identified. These features do not deal with a specific CSN scenario. However, they accurately summarize the common characteristics of CSNs.

- **Network Sizes.** A CSN is composed of tens to a few thousand nodes. The resulting topology may include single or many hops, and in many cases the nodes are organized hierarchically. The typical size of a CSN may vary depending on the application scenario. For those with a large size, hierarchical topology and longer multi-hop paths need to be supported. For small size networks, the connection paths can mostly be single-hop, and flat topologies may be applicable.

- **Node Deployment.** Nodes (sensors, actuators or forwarders) in CSN are usually fixed after being deployed. However the nodes can also be mobile in some selected scenarios. In fixed deployment, topology maintenance is not as vital as in mobile sensor networks, though topology information still needs to be updated to find out any possible link changes or failures. Therefore topology maintenance can be performed occasionally or periodically. The relatively static feature of the network nodes makes the topology more stable. Topology maintenance however still needs to be performed whenever necessary, as the potential mobility or failure of the nodes is still possible. Node failure may occur from time to time. When the deployment of the nodes is performed, the scalability of the network needs to be considered.

- **Heterogeneity.** CSN can consist of four types of nodes, namely sensors, actuators, forwarders and hybrid nodes. Sensors and actuators have limited capabilities and do not usually provide packet forwarding services for other nodes. Local connectivity is an exception here, since the locally connecting node may depend on a sensor node for enabling the connection. Meanwhile hybrid nodes are more powerful nodes and are capable of providing forwarding services for other nodes while being sensor/actuator devices themselves. Forwarders are limited capable nodes and provide only forwarding services for other nodes, but do not have sensing or actuation capabilities. Moreover, since services in a CSN may be provided by different WS&AN supplier, capabilities of sensor nodes involved in such services may be different, e.g. some sensor nodes may be quite powerful and have several types of sensing devices themselves, whereas others may have extremely limited resources and incorporate only one type of sensor. Node capabilities need to be considered, especially during the routing design. Those with higher capabilities may take more responsibility such as to perform data relay, data aggregation, and topology maintenance in case mobility or node failure occurs. Those with extremely limited

resources, except its own responsibility, need to take as little load as possible. Energy conservation is an important issue. Some nodes in the network may access the energy supply from power mains whereas most others are battery driven. Data flows need to be forwarded over those nodes with more available energy.

- **Multiple Services Support**. CSNs may support multiple WS&AN suppliers. Importance of data from these suppliers may be different from one to another. Therefore a CSN should be able to support different types of data with various priorities. Different services can co-exist in a CSN. Such co-existence in fact separates the network into sub-domains. For example the billing system for water, gas, and electricity usually becomes three independent sub-domains though co-operations between them may occur. Therefore this feature brings the requirement to be able to deal with the different service separately and also together whenever necessary. Furthermore, different types of data flows may be generated in CSNs. Some services may require data flows with higher importance. The priority of such data needs to be addressed. High reliability and security need to be guaranteed for these data flows. Some others may be not so important and can be aggregated to save network resources. For instance, the clients are only interested in one simple packet from an area, yet the nodes in this area may generate many forms of data. In such cases, data fusion or aggregation can be performed before the data are sent to the gateway.

- **Energy Management**. Nodes in CSNs can either be supported by batteries or mains power depending on the deployment. Energy supply for the battery driven nodes may be extremely limited. Note that while the mains powered nodes are not energy limited, they may still be constrained with respect to processing cost. A CSN consists of nodes with different resources. The lifetime of nodes with extremely limited energy resources needs to be efficiently prolonged. The selection of the protocol stack being used needs to consider allocating load evenly according to node resources to prevent any premature node energy exhaustion.

- **Gateways to Future Internet.** A CSN can be connected to the future Internet via single or multiple gateways. Access to the Future Internet via gateway requires the gateway nodes to have the necessary functions, like AAA, in order to have the necessary access control. Such features also require considering the security issues if necessary.

- **Connectivity to other Islands**. CSNs can be exploited to provide limited connectivity to other neighboring networks, typically BSNs. This is called local connectivity and it is an important characteristic in various applications. Cooperation between different WS&AN islands needs to be addressed. Cooperation means to share the information if necessary, to route data for other services or node, and to share the resources if necessary.

- **Privacy and security**. Since a CSN can support multiple service types, data passing through the network may vary from very sensitive to public data. Therefore the level of security will also vary depending on the service or data type. Data passing through CSNs range from very

sensitive to public. To grant the desired confidentiality level of these data flows, a scalable security approach will be necessary.

2.3 Body Sensor Networks

2.3.1 Body Sensor Network Description

A Body Sensor Network (BSN) consists of wired and/or wireless sensor and actuator nodes used in close proximity to or within the human (or animal) body. Network nodes may be carried in the hands or pockets, integrated in garments worn, or attached or implanted into the body. BSNs can form individual sensor and actuator islands around their body.

The devices that participate in a BSN have a wide range of capabilities. While mobile phones or PDAs constitute the high-end side and additionally provide connectivity to other networks, most nodes will be tightly integrated into garments or implanted into the human body, necessitating minimal physical size and thus providing only very limited processing and energy capabilities.

A BSN consists of relatively few sensors, such that tamper protection of all sensors might be available at reasonable costs. Also the fact that the BSN is worn on the body reduces the possibility of attackers to physically tamper with the nodes. Nevertheless, the possibility of a malicious sensor should not be easily excluded and active or passive attacks on the network remain a possibility.

2.3.2 Body Sensor Network Features

BSNs have some features that distinguish them from the other networks.

- **Network Size.** A BSN is composed of single to tens of nodes. The resulting topologies include 1-2 hops and in many cases are organized as trees. However, also mesh networks are of interest in order to lower total bandwidth requirements. The network stack needs to operate efficiently in networks with relatively high density.

- **Wired and Wireless clusters.** BSNs might be composed of multiple clusters of nodes that are wired, while interconnected using a wireless sensor network. An example would be a shirt enhanced by a multitude of wired sensors, which connects to the rest of the BSN via a wireless link. Multiple clusters where certain nodes can only be reachable via specific links impose routing and addressing problems which need to be solved on the routing layer.

- **Mobility.** As people move around, they carry their network with them. Other networks in the environment connect and disconnect to/from the BSN, offering their services for the time the user passes by. Mobility introduces clusters of nodes which move together and stay connected for an extended amount of time. Other clusters may only be available for a short time, needing

protocols which can quickly establish links and exchange data between such meeting clusters. This introduces a need for caching information, which may be accessed in shorter time than figuring out nodes at every request.

- **Gateways to Future Internet.** Single nodes, typically a mobile phone carried with the user may provide a link to the Future Internet. In the absence of such a link, a BSN may use surrounding CSNs to connect. Retaining a link to the Future Internet via gateways with mobile networks require mechanism to keep links stable despite changing intermediate nodes. Security is especially needed, if the gateway is not part of the local BSN.

- **Heterogeneity and Self-Configuration.** BSNs are formed of a wide range of devices that may never have seen each other and seamlessly work together. If a user buys a new product or picks up something to carry along with him, it should integrate itself into the network and provide its special services. Device descriptions must be handled by the local devices with possibly very low processing and memory resources – efficient representation of service access format is needed.

- **Privacy and Security.** BSNs collect much sensitive data, such as long-time biomedical measurements. Such data must be protected from unauthorized access and protected when transmitted to other entities. Data that is shared with the community may require distinction between groups that are allowed access, and others from which it should be protected. Different data stored on the BSN needs different levels of security for access authentication and transmission. Mechanisms should include automatic mediation for certain data. A BSN collects typically personal sensitive information. This makes privacy protection of the data an important issue. Protection against falsified information is also relevant, as wrong information might have impact on the safety of the network.

- **Energy Management.** As network nodes are moving with the user, there is no possibility of connecting them to a power grid. Consequently the sensor and actuator nodes need to provide their own power sources. Power might be provided by batteries, and nodes could make use of some energy harvesting techniques collecting solar or motion energy to replenish their energy resources. BSN devices are carried with the user, and may provide charging stations. Devices need to be able to run for several days.

- **High data rates.** Recognizing human activities requires sampling sensors at high rates, e.g. to follow body motions or heart rates. This data may have to be logged for medical supervision or can be processed to produce higher level context information, such as the heart beat, or abstract motions such as “walking”. The network must provide the possibility to transmit data at high rates from multiple sources to multiple destinations. This may also require the transmission of multimedia content.

2.4 Sensor and Actuator Modeling

Several works have been performed in the field of modeling sensors and sensor networks. In our survey we have divided them into two main categories, the ones defined by standardization entities using XML or text-table values; and others using ontologies.

There are several industrial standards that aim at unifying sensor interfaces and data formats.

IEEE 1451 [21], a suite of Smart Transducer Interface Standards, describes a set of open, network-independent communication interfaces for connecting transducers (sensors or actuators) to microprocessors, instrumentation systems, and networks. The key feature of these standards is the definition of Transducer Electronic Data Sheets (TEDS) that stores transducer identification, calibration, correction data, measurement range, and manufacture-related information, etc. TEDS are paper sensor data sheets in electronic form. A TEDS is a text table completely focused on describing the capability and characteristics of individual sensors. It does not capture any derived semantics from sensor data, nor does it capture how the sensor is used, or parameters such as the location, age, or condition of the sensor.

The ANSI provides a standard data format for Radiation Detectors used for homeland security specified in the **ANSI N42.42** [22]. The purpose of this standard is to facilitate manufacturer-independent format to transfer information from radiation measurement instruments to a standard file format for use in U.S. Homeland Security applications without reference to manufacturers' documentation. The purpose of this data is for analysis and storage of the radiation measurements. The data may consist of raw or unprocessed data, analysis results, device parameters or settings, or other measurements needed or applicable to the further analysis or to verify the quality of the results produced by the instrument. This standard does not address instrument control, data transmission protocols, or the physical media used for communications. The data format is specified using XML schemas.

Common Chemical, Biological, Radiological, Nuclear (**CBRN**) Sensor Interface (**CCSI**) [23] is a standard for sensor physical and electronic interfaces including components interconnects, power, external connectors, XML communications, a standard basic command set and modularity. The goal of the CCSI is to develop a set of common standards that enable CBRN sensor interoperability, net centric operations, and ease of integration into command and control systems.

The standard characterizes sensors as packages of capabilities rather than what they detect. Following this approach, the sensors are describe in terms of installation (fixed, mobile, dismountable, dismounted, personal), power source (facility, platform, internal, external), communications characteristics (wired-network, wireless-network, wired-local, wireless-local) and usage (constant, periodic, non-periodic, mixed), operation (local, remote, mixed, environment (combat, severe, outdoor, indoor) and security (physical, electronic, communications).

The main drawback of the CCSI is that is intended to be used in the scenarios and existing processes from The US Ministry of in the field of detection, classification, identification and notification of CBRN threads, mainly in battlefields. Some of the documentation is not intended to be publically available and most of the descriptions values have a very restrictive number of alternatives (for example, the sensor environmental characteristics are restricted to *combat, sever outdoor and indoor).*

The **OGC**® Sensor Web Enablement (SWE) working group has created three different standards for defining encodings for describing sensors and sensor observations: Sensor Model Language [24], Transducer Markup Language[25] and Observations & Measurements [26], [27].

Transducer Markup Language (TransducerML), is a XML based standard for specifying a standardized way to exchange raw or pre-processed sensor data. TransducerML is a language which will enable the seamless communication of digital transducer data between transducers (either a receiver like a sensor, or a transmitter) and processor. In particular TransducerML was designed to facilitate interoperability of sensors and sensor processors, fusion of heterogeneous sensor data, and the accurate and precise capture of sensor data. TransducerML is a sensor data exchange language that allows for the fusion of data at levels not presently achievable (i.e. upstream data). It defines a means to capture and describe streaming transducer data for the purpose of exchanging and processing raw & processed data from disparate transducers.

The main features of TransducerML are:

- Promote interoperability and fusion of all types of sensor data.
- Automatic association between data in space and time (absolute or relative) from sensors on different platforms.
- One data exchange protocols for all transducers.
- Facilitate Plug&Play transducers.

SensorML is defined in XML schemas and can, but generally does not provide a detailed description of the hardware design of a sensor. Rather it is a *general schema for describing functional models* of the sensor. It provides a common framework for any process and process chain, but is particularly well-suited for the description of sensor and systems and the processing of sensor observations. The main features of SensorML are:

- Sensor and systems descriptions
- Support for resource and observations discovery
- Support for processing and analyzing sensor observations
- Support for geo-location of the sensors and measured data
- Performance characteristics (accuracy, threshold, etc).
- Support for specifying how to process sensors measures to derive new information

In SensorML, all components are modelled as processes. This includes components normally viewed as hardware, including transducers, actuators, and processors (which are viewed as process components) and sensors and platforms (which are modeled as systems). The modeled processes can take input, and through the application of an algorithm defined by a

method and parameter values, generate output. All such components can therefore participate in process chains. Process chains are themselves processes with inputs, outputs, and parameters.

Complementary to the SensorML is the Observations & Measurements (O&M) standard, since it provides a homogeneous way of representing the measures or observations taken by the sensors.

ZigBee is a communication standard for WS&AN island with a multitude of solutions in consumer electronics, home and building automation, industrial controls, PC peripherals, medical sensor applications, toys, and games [3]. ZigBee defines application profiles which are collections of device descriptions, which together form a cooperative application. For instance, a light switch on one physical node communicates with a lamp on another physical node. Together, these devices cooperatively form a lighting application profile. Before these related (according to the profile) devices send messages they must be bound to each other by a setup phase called binding. Each device with potentially multiple onboard sensors and actuators may have multiple applications running. To distinguish between applications ZigBee uses the notion of application endpoints or just endpoints which have their own identifiers for message de-multiplexing purposes. Application endpoints are similar to ports in the TCP/IP stack. A device communicates with another device by sending and receiving messages over the ZigBee networking layer by specifying the device address and the application endpoint address. Each endpoint in turn may implement multiple interfaces which are referred to as clusters according to the ZigBee terminology. Clusters have their own identifiers (Cluster IDs) and are collections of attributes and commands. In turn attributes are categorized in terms of the access rights: as a) "read only" if they can only be read (e.g. a measured sensor value) or b) "read/write" if they be read or written (e.g. configuration or actuation settings). Attributes are also categorized as: i) mandatory if the standard mandates the existence of the attribute on the device that implements a specific cluster which includes the attribute in question or b) optional if the standard allows devices not to include the specific attribute in their cluster implementation. Clusters have also commands which act on the attributes. Commands are described later in the document,

ZigBee provides several predefined clusters bundled in the generic ZigBee Cluster Library (ZCL) specification and as well as individual application profile specifications such as the Smart Energy or the Home Automation profile. The ZCL and the application profile specifications include possible device and application configurations apart from the cluster specification i.e. specification of attributes and commands for each cluster. An example of such configuration is a thermostat device connecting to a device with a temperature sensor in order to implement a heating application profile. These possible example configurations will be presented later in the document under the sensor and actuation task modeling section.

ZigBee clusters are collections of attributes and commands regarding sensors and actuators. Zigbee uses attribute-value pairs to represent the attributes on each device including the sensed values for sensors and the desired outcomes of set points for actuators. Each attribute has a specific data type which imposes a specific bit length limit on the stored value of the attribute. For example there are 8-bit, 16-bit etc signed integer or character strings with a

maximum value of 254 bytes with a run-length encoding (the first byte denotes the length, rest of the bytes is the actual string). Attributes cannot have names as arbitrarily long strings because of the memory limitation of the sensors/actuator devices. As a result ZigBee employees an encoding of each attribute to a 16-bit attribute IDs.

Apart from the usual attributes (such as the measured light value from a light sensor, sensor tolerance, min, max capabilities), sensors and actuators have other more interesting attributes such as: a) event thresholds for generating alarms, b) accumulator/aggregator attributes that maintain for example the total operational time of a device or the total energy consumption of a sensor or actuator , c) control loop set-points (e.g. target temperature for a thermostat-temperature sensor application) or control loop support attributes such as the *PI(Heating/Cooling)Demand* attribute that can be used for a control loop employing a Proportional-Integral (PI) controller and d) attributes for setting up control loops e.g. if and how a thermostat connects to an occupancy sensor.

One important note is that the possible values of a sensed/measured quantity may not fit to the pre-assigned data type so ZigBee defines its own mapping functions. For example the mapping function between the 16-bit representation (*MeasuredValue* attribute of the Luminance Measurement Cluster) of luminance in lx and the actual luminance is $MeasuredValue = 10{,}000 \times \log_{10} luminance + 1$. With this 16-bit representation the possible range of measured luminance is $1\ lx <= Luminance <= 3.576\ Mlx$.

As a summary of the study performed about the standards to model sensors, we can conclude that SensorML provides a good framework to represent not just physical sensors, but also virtual, processors, sensor networks, mechanisms to derive high level information from raw sensor data, etc. Furthermore, SensorML has plans, in future releases, to work on the mapping between IEEE 1541 and ANSI N42.42 standards and on the harmonization of SensorML and TransducerML.

SensorML, as an XML based language, does not provide semantics, but introduces two important elements that can be combined with other kind of information: Phenomena and Feature of Interest. The *Phenomena* represents a physical property that can be measured of a Feature of Interest; while *Feature of interest* has the same meaning of the *Entity of Interest*. The definition or representation of the Features of Interest is out of the scope of SensorML, but, using some other information sources (like ontologies) semantics about the entities can be expressed.

It is also important to remark that SensorML is being used by a big number of organizations like the NASA or the European Space Agency as the basis for the development sensor related applications.

Apart from the OGC standards the ZibBee sensor and actuator models are useful as examples of real-world models. However ZigBee provides fixed and design-time sensor and actuator models with implicit semantics. For example a temperature sensor provides a temperature measurement with resolution and units that are specified in the ZigBee Cluster Library. No ontology is maintained on the sensor nodes for the obvious reasons of memory, power and computation limitations. The interfaces to the sensors and actuators are also

predetermined by the standard specification and the sensor/actuator task models are design-time specified by the applications. Nevertheless ZigBee provides useful notions such as that of Scenes which are sets of stored attributes for configuring the application on demand and restoring the application when something occurs e.g. a node battery fails. ZigBee provides a framework for mainly WS&AN island applications apart from the recently proposed Smart Energy profile. The Smart Energy profile defines roles for Automatic Metering applications such as building energy measurements. This profile defines the AMI (Advanced Metering Infrastructure) server which is positioned outside the WS&AN island and interacts with sensors, actuators, and displays within the WS&AN island. The objective of such an AMI application is to provide the user with real-time measurements about the user's house and the possibility to control the energy usage.

Chapter 3

Energy Harvesting

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 small autonomous sensorial nodes employ this process to replenish their energy resources [28]. When applied to sensor network architecture, energy harvesting increases the robustness and the availability of the system making it energy-independent.

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.

3.1 Mathematical Modeling of Energy Harvesting

Battery-powered systems can employ multiple strategies in order to conserve energy and increase operational lifetime. In most cases, systems minimize their energy consumption [29], [30] or dynamically scale performance so that all computing goals are achieved within given constraints [31], [32], [33], [34]. In order to increase their operational periods, sensor nodes use energy harvesting in order to complement their energy budget. Surplus energy is stored in a buffer, such as a rechargeable battery or super-capacitor for later use. Further optimization of such a system can be done in the direction of minimizing energy consumption to such a degree that it becomes smaller than the energy production rate from the harvester. Achieving such a condition of energy neutral operation will grant, at least in theory, perpetual operational lifetime for a wireless sensor node. Hardware ageing and wear and tear are well documented problems in the field of reliability that affect all systems, including sensor nodes. However, a vast majority of sensor nodes use components that are very reliable with mean times to failure measured in decades, such as microcontrollers or silicon sensors. This, coupled with the use of reliable energy

storage devices, such as the super-capacitors described in the previous section could grant the node an extended lifetime of neutral energy operation.

3.1.1 Design Considerations

One of the simplest harvesting systems that can be designed is one where the energy which is produced is greater at any given moment than the one which is consumed. However, in a real-life scenario this is rarely the case due to the fact that energy production rates are almost never constant and can exhibit large variations during a given period of time. For example, the energy output of a solar panel can vary widely during the day, due to meteorological conditions and will drop to almost zero during the night. The same is the case with other energy harvesting systems, such as the ones that employ temperature gradients or vibrations. If energy from such a source is fed directly to the system, it will power it only during the periods of high output from the harvester. To achieve full energy neutrality it is obvious that an energy buffering and power management system is needed in order to store surplus energy and release it when production is low or non-existent.

Energy neutrality is defined as a condition where the energy consumption rate of a system is balanced to always fall below the energy production rate. This might seem like a trivial matter for a simple sensor node with few integrated components. However, when scaled up to a network of hundreds of such nodes, a power manager needs to take into consideration multiple other constraints such as quality of service or task scheduling and redistribution in case of node failure. This can be achieved by performance balancing algorithms, which use the given energy budget for the entire network in order to maintain it operational at the maximum allowed level of performance.

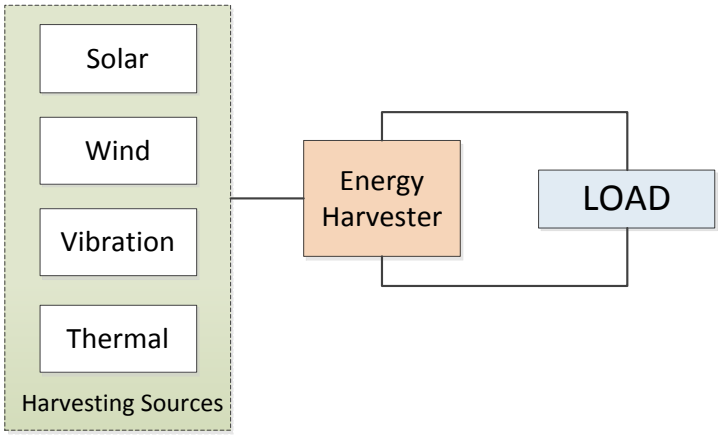


Figure 9. Energy Harvesting System Diagram

A typical energy harvesting system is comprised of three subsystems:

- **Energy Transducer:** is responsible for transforming a physical phenomenon such as wind, thermal or solar energy into a variable voltage. It usually employs solar panels, alternators, or piezoelectric transducers, according to which type of energy is available to the system. Usually, the voltage output is not stable and depends on environmental conditions which cannot be easily modeled or predicted by the system. In systems such as wireless sensor networks, each sensor node can employ the use of such energy sources and transducers.
- **Harvesting Circuit**: provides impedance matching and supplies an optimal voltage to the system load from the variable energy supply of the harvesting source. The harvesting system is also in charge for storing excess energy and managing the charge and discharge of the energy buffer. Power management algorithms are usually implemented at this level. In the case of sensor networks, each node has separate harvesting circuitry but the power management algorithm can be distributed, in order to ensure a unified power consumption policy.
- **Consumer**: also known as the load. It is often variable, according to the activity of the system it models. For example, a sensor node consists of many subsystems which can serve different purposes and have different functions. These modules can be switched on or off, or their consumption adjusted according to the power management policy which is implemented in the sensor's firmware. Most nodes execute a simple while loop in which sensor sampling, data processing and radio transmission are executed sequentially. This allows a simple power management scheme to be implemented, in which hardware modules can be periodically switched in a low power state, according to their usage.

3.1.2 Energy Source Modeling

In this section we describe a model for energy producers and consumers which can be applied to any energy harvesting system. This model takes into account the steps needed for a system to achieve energy neutrality. As lifetime is influenced by several conditions, but mostly by the availability of a steady supply from the energy source, we perceive energy neutrality as a different concept, namely the difference between the energy production and consumption rate of a given system. Lifetime is hard to define due to the fact that systems that employ harvesting can stop functioning for long periods of time when the energy production rate dwindles but resume with full functionality when energy goes over a certain threshold.

Energy harvesting systems might use only one or all of the types of energy sources mentioned above. A mathematical model that includes all four types of energy sources was derived, starting from a simplified system in which there is no energy buffering. Starting from this simple model, we iterated, adding further complexity as we progressed to a model that was very close to reality. The first step was to add an ideal energy buffer in order to store surplus

energy from the harvesting source. The second iteration added complexity to the buffer, by modeling it as a non-ideal storage medium, with energy loss from leakage and during charging. The final step was to take into consideration the fact that any storage medium can hold a finite amount of charge and that the energy consumption rate from the load can vary.

We can approximate the sensor network with a closed energy system in which each node acts both as a producer and a consumer of energy. We can measure the total energy production rate of the node $P_P(t)$ and the energy consumption rate, $P_c(t)$. The excess of harvested energy by the node at any moment will be:

$$E(t) = \int_0^t (P_P(\tau) - P_c(\tau))d\tau \quad (2)$$

From this, we can deduce that a node is deemed energy-independent if its excess energy satisfies the following formula:

$$E(t) > 0 \; \forall \; t > 0 \quad (3)$$

3.1.3 Energy Harvesting System Categories

As mentioned before, the mathematical model is split into three iterations. The main idea behind each case is to establish the relation between the consumed and produced energy rates using the laws of energy conservation. When the equation is balanced, energy neutral operation is achieved.

The first iteration covers only the energy storage part, with no concern about the storage of the produced energy. In this case, the system gathers energy from the surrounding environment and powers the load directly, without any buffering. As an example, imagine a system powered by a photovoltaic cell. This system will function only during periods of proper illumination, remaining inactive at night. Another example is a piezoelectric device such as the one in [35] which has no conventional energy storage and powers its electronics momentarily.

We define $E_P(t)$ and $E_c(t)$ as the energies produced, respectively consumed by the harvesting device:

$$E_p(t) = \int_0^t P_P(\tau)d\tau \quad (4)$$

$$E_C(t) = \int_0^t P_C(\tau)d\tau \quad (5)$$

For such harvesting systems, the device can operate at all *t* when

$$E_P(t) - E_C(t) \geq 0 \tag{6}$$

Of course, the lack of any energy storage medium will mean that all of the energy gathered when $E_P(t) < E_c(t)$ will be wasted, even though the system does not have enough energy to function. Also, when $E_P(t) \geq E_c(t)$, the surplus energy $E_P(t) - E_c(t)$ will be wasted.

As previously discussed, the consumption and the energy generation are almost never synchronized, with the latter lagging behind in many cases. Also, from the previous example we can deduce that there are some cases when gathered energy is greater than the load can absorb, so it is simply wasted. A solution for such scenarios is to first devise a way to store incoming energy in a buffer and only then supply it to the system. For simplicity, we will first choose an ideal energy storage medium. This ideal buffer can store an infinite amount of energy and can supply it instantly to the load at any given time, without any losses. In this case, applying the rules of energy conservation yields the following equation:

$$E_C(t) - E_P(t) + E_{B_0} \geq 0 \forall t \in [0, \infty) \tag{7}$$

$$\int_0^t (P_c(\tau) - P_P(\tau))d\tau + E_{B_0} \geq 0 \; \forall t \in [0, \infty) \tag{8}$$

where E_{B_0} is the amount of energy that is stored in the buffer at *t=0*.

The last step in the modeling process is to take into consideration the non-ideal nature of the energy storage medium. There are two parameters that are relevant to this issue, namely the losses that occur during the charging process and the leakage of power from the storage medium over long periods of time. These two phenomena affect all energy storage mediums, be it batteries or super-capacitors. For this, we introduce two new parameters: the charging efficiency, η, which is strictly less than unity and the energy buffer leakage power, $P_{leak}(t)$, which models how much power is being lost by the storage at time *t*. Another important parameter which needs to be taken into account is the fact that any battery or storage medium has a maximum capacity, which we define as E_B.

Applying the same energy conservation laws to the new model, we can establish the following relations:

$$E_{B_{max}} \geq E_{B_0} + \eta \cdot E_P - E_C - E_{leak} \geq 0 \;\; \forall t \in [0, \infty) \tag{9}$$

$$E_{B_{max}} \geq E_{B_0} + \int_0^t (\eta \cdot P_P(\tau) - P_C(\tau) - P_{leak}(\tau))d\tau \geq 0 \;\; \forall t \in [0, \infty) \tag{10}$$

The authors of [36] further refine this relation by bounding the produced and consumed energy to a specific interval. This is done by finding the $(\rho,\ \sigma_1,\ \sigma_2)$ functions associated to the energy production and consumption rates.

A non-negative, continuous and bounded function P(t) *is said to be a* $(\rho,\ \sigma_1,\ \sigma_2)$ *function if and only if for any value of finite positive real numbers* τ *and* T, *the following relation is satisfied:*

$$\rho T - \sigma_2 \leq \int_{\tau}^{T+\tau} P(t)dt \leq \rho T + \sigma_1 \tag{11}$$

The ρ parameter signifies the constant rate at which energy is produced or consumed and $\sigma_1,\ \sigma_2$ model the burstiness of the energy harvesting source or of the consumer. For their model, they presume that $P_P(t)$ is a $(\rho_1,\ \sigma_1,\ \sigma_2)$ and $P_C(t)$ is a $(\rho_2,\ \sigma_3,\ \sigma_4)$ are such functions and that leakage is happening at a constant rate $P_{leak}(t) = \rho_{leak}$.

By using these assumptions and the relation in (10), they derive the following inequalities:

$$E_{B_0} + \eta \cdot \sigma_1 - \sigma_4 \leq E_B \tag{12}$$

$$E_B \geq \eta \cdot (\sigma_1 + \sigma_2) + \sigma_3 - \sigma_4 \tag{13}$$

$$\eta \cdot \rho_1 - \rho_{leak} \leq \rho_2 \tag{14}$$

These equations are useful in determining two very important parameters of a harvesting system: the maximum size of the energy buffer and the required rate of energy production from the harvester circuit.

In (12) and (13), the maximum energy of the storage element is assessed by taking into consideration the burstiness of the harvested source and that of the consumer, and in (14) we can compute the rate at which the harvesting circuit needs to supply energy to achieve a sustainable rate.

3.2 Radio Transceiver Consumption Modeling

Research in the area of low-energy radio integrated circuits is ongoing and is motivated mainly by the applications in mobile and embedded market. In most countries duty cycling is imposed at a certain value for the standard ISM bands [37], [38]. In Europe, for the 434MHz band, duty cycling needs to be smaller than 10% and smaller than 1% for the 868MHz band. The duty cycle is calculated as the percentage of time the radio is on during a predetermined time interval, which, for this standard is an hour. In order to increase the availability of a sensor

network, duty cycling is one of the first parameters to be evaluated, as it has a drastic effect on the energy efficiency of the network.

In the following, we present a model for estimating radio energy consumption in a wireless sensor network. The main issue is how to estimate the energy needed to send a package of n bits of data from the transmitter to the receiver, as in Figure 10.

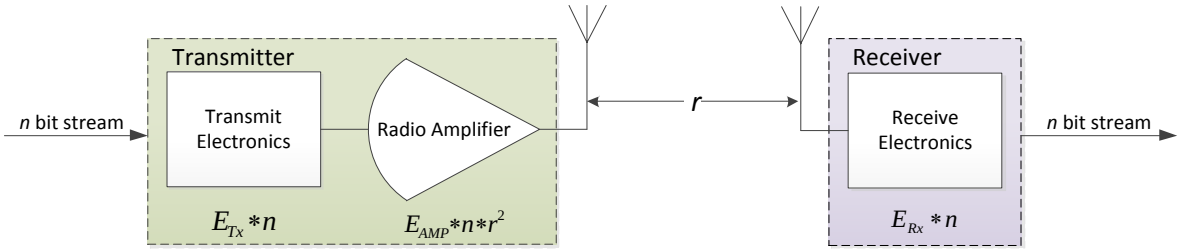


Figure 10. Radio model for the transmission of n bits of information

In order to transmit a package of *n* bits at a distance of *r*, the radio transmitter will spend the following amount of energy:

$$E_{Tx}(n,r) = E_{tc}(n) + E_{amp}(n,r) \tag{15}$$

where $E_{tc}(n)$ is the energy that the radio circutry needs to expend in order to process *n* bits, and $E_{amp}(n,r)$ is the energy needed by the radio amplifier circuit to send *n* bits at *r* meters.

We can further refine (15) by elaborating of the formula for $E_{amp}(n,r)$:

$$\begin{aligned} E_{Tx}(n,r) &= E_{tc}(n) + E_{amp}(n,r) \\ &= n \cdot E_{trans.} + n \cdot \varepsilon_{amp} \cdot r^{\gamma} \end{aligned} \tag{16}$$

where $E_{trans.}$ is the energy needed to process a single bit by the radio transmission circuits, ε_{amp} is the transceiver's energy dissipation and α represents the path loss exponent.

Path loss is a major factor in estimating the link budget for a radio transceiver. For the present research, we used the standard log-distance path loss model:

$$PL = P_{Tx[dBm]} - P_{Rx[dBm]} = PL_0 + 10\gamma log_{10}\frac{d}{d_0} + X_g \tag{17}$$

where PL is the ideal path loss measured in dB, $P_{Tx[dBm]}$ is the transmitted power in dBm, $P_{Rx[dBm]}$ is the received power in dBm, PL_0 is the path loss at a reference distance d_0 (usually 1km), γ is the path loss exponent, *d* is the path length and X_g is the attenuation due to fading.

Path loss exponents are linked to the medium of propagation [39] and usually range from 2 to 4, where 2 is the path loss of free space propagation and 4 is the path loss exponent for lossy environments such as buildings or stadiums.

An explicit relation for ε_{amp}can be found in [40]:

$$\varepsilon_{amp} = \frac{\frac{S}{N_r} \cdot NF_{Rx} \cdot N_0 \cdot BW \cdot \left(\frac{4\pi}{\lambda}\right)^{\gamma}}{G_{ant} \cdot \eta_{amp} \cdot R_{bit}} \tag{18}$$

where $\frac{S}{N_r}$ is the signal to noise ratio at the receiver, NF_{Rx} is the receiver noise figure, N_0 is the thermal noise for a 1Hz bandwidth, BW is the channel noise bandwidth, λ is the wavelength in meters, γ is the path loss, G_{ant} is the antenna gain, η_{amp} is the transmitter efficiency and R_{bit} is the channel data rate in bits per second.

Alternatively, we can express in the same way the energy required for the transceiver to successfully receive and process *n* bits of data:

$$E_{Rx}(n) = E_{rc}(n) = n \cdot E_{recv.} \tag{19}$$

This model assumes that the communication through the radio channel is symmetric and that the energy to send a package from node A to B is the same as the one needed to send the same package from B to A, for a constant SNR. As can be seen in the above relations, any type of communication is not a low cost operation so the protocol stacks that run on the nodes should always try to minimize the number of transmit and receive operations in order to keep the energy budget of the network under a certain threshold.

So far, we have been focusing on modeling the communication between only two nodes, but the same model can be scaled up to estimate the energy consumption at network level. For this, there are two cases worth taking into consideration: a network in which nodes talk to the gateway using a direct communication protocol, and the more general multi-hop network scenario, in which messages are passed from neighbor to neighbor until they reach the data sink.

Using the direct communication approach, each node has direct access to the gateway. As the distance between nodes and the gateway is not constant and can vary within radio connectivity range, some remote nodes will need greater amounts of transmit power to communicate with the data sink. In this case, r in (16) is large, which leads to more energy spent and quicker battery drainage. On the other hand, there is no need for the nodes to receive any information from their neighbors, as the communication is done over a star topology network. This could prove advantageous or even optimal if nodes are in close proximity to the gateway or the cost of reception on the battery-powered nodes is sizeable.

The second approach is to use a power-aware multi-hop routing protocol, as discussed by [41], [42], [43], [44]. In this case, data is disseminated in the network though paths that will

ultimately lead to the sink. These paths are chosen according to the routing algorithm used by the protocol stack and can vary, depending on the different metrics involved.

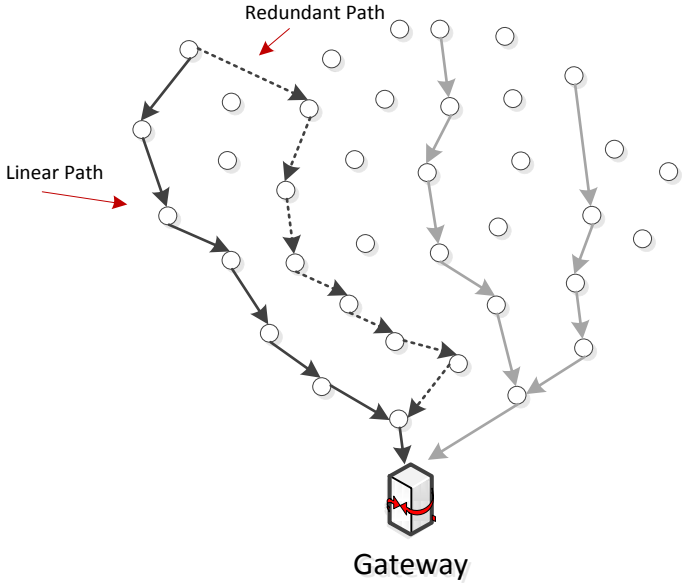


Figure 11. Linear and redundant paths in a sensor network

Consider the example in Figure 12, which represents a typical linear sensor network where nodes are spread at equal distances from each other. Based on the equations we described earlier, we can estimate the energy cost of communication in such a network.

First, for the single-hop case, the node is communicating directly to the gateway. For the N-th node, this would imply that it needs to increase its transmitter signal strength in order to cover the entire distance to the gateway, which would in turn lead to higher energy consumption.

This can be expressed as:

$$E_b(N,n,r) = E_{Tx}(n, N \cdot r) = n \cdot E_{trans.} + n \cdot \varepsilon_{amp} \cdot (N \cdot r)^{\gamma} \quad (20)$$

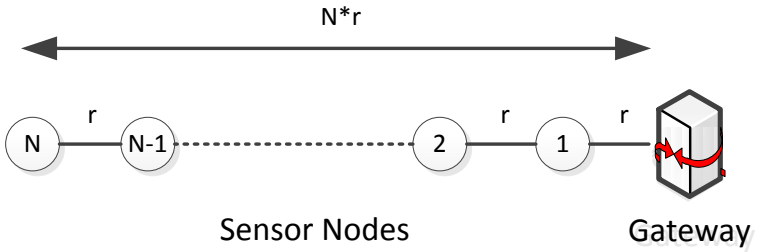


Figure 12. Simple linear sensor network

For the multi-hop case, the N-th node needs to send data to his nearest neighbor, which would expend energy in receiving the package and retransmitting it to its nearest neighbor, and so on until it reaches the data sink.

The total energy expenditure of the network can be calculated as a sum of *N* transmits and *(N-1)* receives:

$$\begin{aligned} E_{MH}(N,n,r) \\ &= N \cdot E_{Tx}(n,r) + (N-1) \cdot E_{Rx}(n) \\ &= N \cdot n \cdot (E_{trans.} + \varepsilon_{amp} \cdot r^{\gamma}) \quad + (N-1) \cdot n \cdot E_{recv.} \\ &= n \cdot \left(N \cdot \left(E_{trans.} + E_{recv.} + \varepsilon_{amp} \cdot r^{\gamma}\right) - E_{recv.}\right) \end{aligned} \tag{21}$$

where n is the number of bits in a message.

In most cases, however, all nodes in the network need to send packages to the base station. For the multi-hop case, we can generalize the relation in (21) to *N* nodes:

$$\begin{aligned} E_{MH}^{all}(n,r) = \sum_{i=1}^{N} E_{MH}(i,n,r) = N \cdot E_{Tx}(n,r) + (N-1) \cdot E_{Rx}(n) \\ = \frac{N \cdot (N+1)}{2} \cdot n \cdot (E_{trans.} + \varepsilon_{amp} \cdot r^{\gamma}) + \frac{N \cdot (N-1)}{2} \cdot n \cdot E_{recv.} \end{aligned} \tag{22}$$

The same generalization can be made with the single-hop case given by (20):

$$E_b^{all}(n,r) \; = \sum_{i=1}^{N} E_{Tx}(n, i \cdot r) = n \cdot N \cdot E_{trans.} + n \cdot \varepsilon_{amp} \cdot r^{\gamma} \cdot \sum_{i=1}^{N} i^{\gamma} \tag{23}$$

Using the equations in (22) and (23), we can derive the conditions for which direct communication to the gateway has a lower energy cost for the whole network, compared to the multi-hop scenario. This is equivalent to the following condition:

$$E_b^{all}(n,r) < \; E_{MH}^{all}(n,r) \tag{24}$$

Certain assumptions must be made in order to simplify the above relation. First, we can assume that the energy expended in processing one bit for transmission is roughly equal to the energy of processing a received bit, as most radio transceivers use the same electronics for both functions:

$$E_{trans.} = E_{recv.} = E_{circ.} \tag{25}$$

Secondly, we can assume a constant path loss exponent for the entire network. In most cases where it cannot be easily measured, the path loss exponent is estimated to be the standard value for free space propagation, $\gamma = 2$.

Using these two assumptions, we can write the relation in (24):

$$nNE_{circ.} + n\varepsilon_{amp}r^2 \sum_{i=1}^{N} i^2 < \frac{N \cdot (N+1)}{2} n(E_{circ.} + \varepsilon_{amp}r^2) + \frac{N \cdot (N-1)}{2} nE_{circ.} \tag{26}$$

$$NE_{circ.} + \varepsilon_{amp}r^2 \frac{N(N+1)(2N+1)}{6} < \frac{N(N+1)}{2}(E_{circ.} + \varepsilon_{amp}r^2) + \frac{N(N-1)}{2} E_{circ.} \tag{27}$$

$$\varepsilon_{amp}r^2 \left(\frac{(N+1)(2N+1)}{6} - \frac{N+1}{2} \right) < (N-1)E_{circ.} \tag{28}$$

$$\frac{E_{circ.}}{\varepsilon_{amp}} > \frac{N+1}{3} r^2 \tag{29}$$

The relation in (29) is applicable for an ideal medium, without any interference. A model that is nearer to reality can be obtained if we modify the path loss exponent to a value of 4, which is characteristic to a lossy indoor environment. Rewriting (24) for this new parameter value yields the following equation:

$$\frac{E_{circ.}}{\varepsilon_{amp}} > \frac{(N+1)(6N^2 + 15N + 16)}{30} r^4 \tag{30}$$

where *N* is the number of nodes in the linear path and *r* is the distance between nodes.

3.3 Energy Harvesting Circuits

A variety of sources for energy harvesting have been researched, such as solar power, thermal energy, vibration energy and radio-frequency energy. From the energy standpoint, all of the energy sources stated above have small energy density values compared to more classic energy sources, such as batteries of fuel cells.

The most prevalent energy sources that are used in energy harvesting are specified in Table 6. Due to the nature of this study, we focused only on the ones that are most suited for powering wireless sensor nodes.

Energy Source	Power Density & Performance	Reference
Vibrations (Piezoelectric)	200 μW/cm3	[45]
Thermoelectric	60 μW/cm2	[46]
Ambient Radio Frequency	1 μW/cm2	[47]
Ambient Light	100 mW/cm2 (direct sun)	Commercially available
	100 μW/cm2 (illuminated office)	
Temperature Variation	10 μW/cm3	[48]
Vibration	4 μW/cm3 (human motion—Hz)	[49]
	800 μW/cm3 (machines—kHz)	
Airflow	1 μW/cm2	[50]
Push buttons	50 μJ/N	[51]
Shoe Inserts	330 μW/cm2	[52]
Hand generators	30 W/kg	[53]
Heel strike	7 W/cm2	[54]
Acoustic Noise	0.003 μW/cm3 @ 75Db	[55]
	0.96 μW/cm3 @ 100Db	

Table 6. Energy Harvesting Sources.

While in the past, the use of radio transceivers often implied large amounts of power consumption, 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.

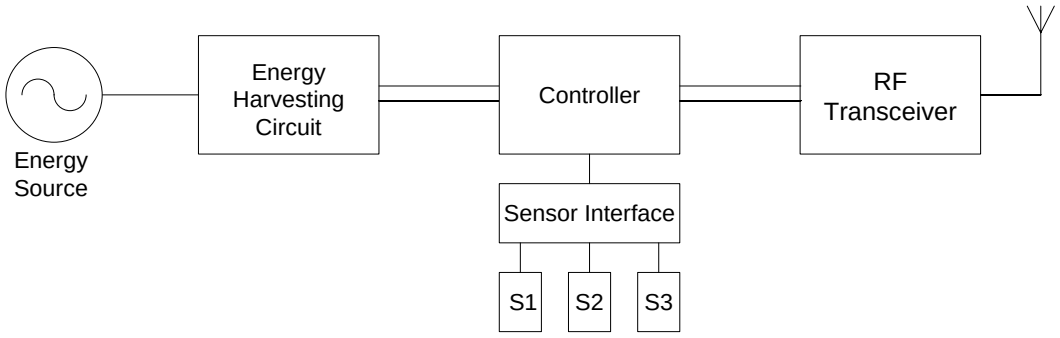


Figure 13. System block diagram of a typical energy harvesting sensor node

This chapter discusses a few power management techniques that are unique to wireless sensor networks. The field of energy harvesting is also covered, with four of the most promising technologies being researched: vibration harvesting, thermal conversion, photovoltaic conversion and radiofrequency energy harvesting.

3.3.1 Energy Profiling for a Sensor Mote

Energy consumption in a node is directly related to the amount of processing, sensing and communication the node has to achieve in a given time interval. These activities can be defined as the node's workload and, it is composed of two distinct phases [56], as shown in Figure 14:

- *low workload* – is the state in which the sensor node resides for the most part of its activity. The node wakes up periodically, samples data from its sensors and may or may not relay data to its peers, depending on its programming. In order to better implement this functionality, sensor nodes must provide short wake-up times and a low power sleep mode.
- *high workload* –the node is executing large amounts of computation and communication with other sensor nodes.

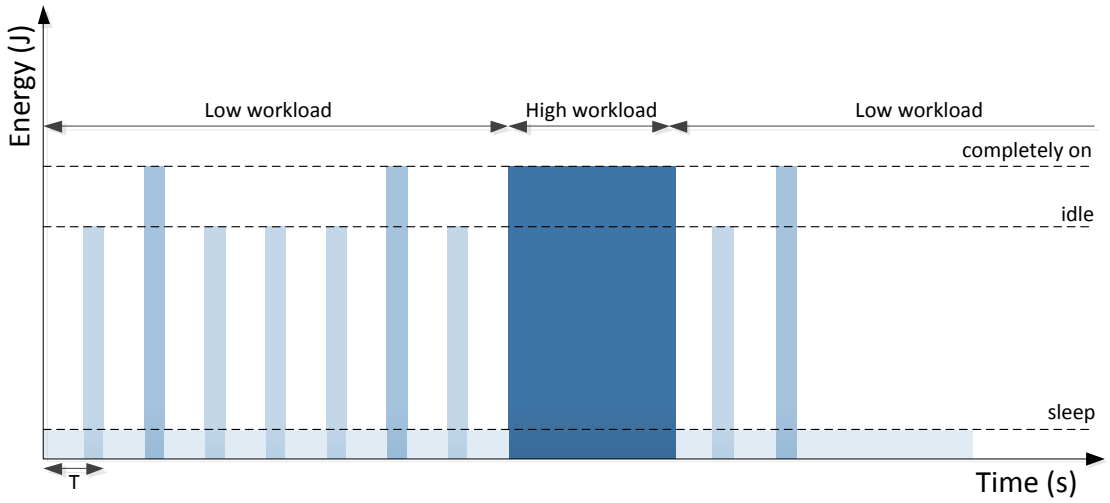


Figure 14. Typical Energy Profile of a Sensor Mote

In reducing node power consumption, the most common approaches [57] are the following:

- *duty cycling* implies switching on and off a sensor node on a periodical scheme. The difference between on and off periods needs to be adjusted according to the designated application of the sensor network and can be considerable for slow evolving processes.
- *adaptive-sensing* is a strategy that is able to modify the sensor network activity to dynamically follow and react to change in the monitored process.

There are also other parameters that must be taken into consideration when programming sensor node applications and duty-cycling. *Wake up latency* is the time interval from node power-up until it generates its first valid results and, as such, adds to the duration of sensor on-time in one period. If wake-up latency is not taken into account by the application programmer, the nodes might broadcast invalid or corrupted data, affecting overall network functionality. Another important parameter is the *break-even cycle* that determines under which conditions it is

efficient to implement power management strategies in a sensor node, as opposed to not implementing them at all. Some systems and circuits require considerable amounts of power to switch on or off. Implementing duty cycling for a node that uses such circuits may not only drain significant amounts of power, but also prove more energy-inefficient than keeping the node on at all times [58].

3.3.2 Vibration Harvesting

Vibration harvesting is based on the piezoelectric effect, namely the capacity of a material to generate electricity when subjected to mechanical stress such as stretching or compressing. Conversely, a mechanical deformation can be achieved if the crystal is subjected to a certain voltage.

There are several sources that can be used in order to produce an electrical charge by employing the piezoelectric effect. Acoustic noise, machine vibration or even human motion are some of the sources researched in [59], [60], [61].

The piezoelectric effect generates an AC current, ranging from microwatts to the order of milliwatts of power. While this is an insignificant figure for a large consumer, it can prove enough to power a low power mobile or hand-held device. Harvesters can be in the form of shoe inserts, to recover energy from walking [62], embedded in walkways in pedestrian areas [63] or even as implantable energy sources[64].

Most piezoelectric materials are anisotropic, exhibiting different behavior depending on the direction of deformation and the orientation of the polarization of the internal crystal lattice. These characteristics are defined by a standard set of notations using tensors and matrices [65], and the most general notations are shown in Figure 15. Piezoelectric activity is defined by the axes of strain application and a series of constants that describe the material.

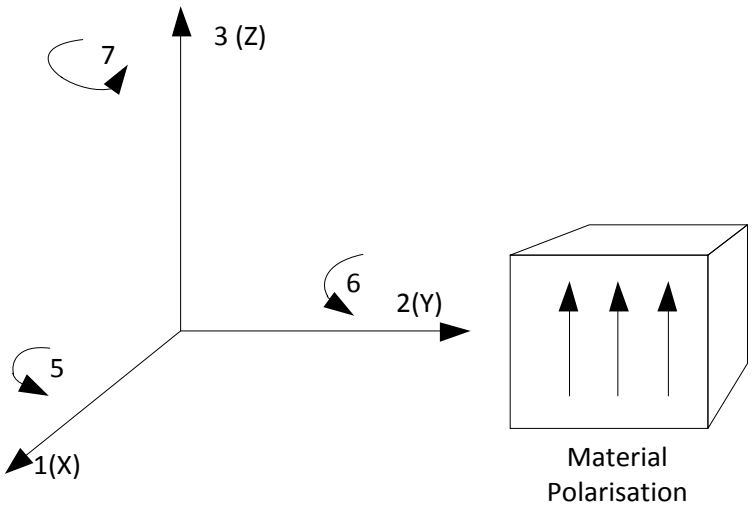


Figure 15. Axis notation for a piezoelectric crystal

An important parameter in describing a piezoelectric material is the strain constant *d*. This can be defined as the ratio between the developed strain and the applied electric field, in the case of applying a voltage excitation to the piezoelectric crystal, or as the ratio between the charge density and stress, in the reverse case of applying mechanical force.

There are two major types of applying a compressive strain to a piezoelectric element: perpendicular to the gathering electrodes, to exploit the so called d33 coefficient or transversal in order to utilize the d13 coefficient. Increased power outputs can be achieved if multiple piezoelectric elements are stacked and connected electrically together. The elements can also be arranged in a way that enhances or amplifies the initial excitation, in order to produce higher power outputs.

Another important coefficient in describing a piezoelectric material is the electro-mechanical coupling coefficient, *k*. This quantifies the efficiency with which the mechanical strain is converted into electrical energy, and can be defined by the following equation:

$$k_{ij}{}^2 = \frac{W_i^e}{W_j^m} \tag{31}$$

where W_i^e is the electrical energy stored in the *i* axis and W_j^m is the mechanical input energy for the *j* axis.

The relation between *k* and *d* is expressed by the following equation:

$$k = \frac{d^2 Y}{\varepsilon} \tag{32}$$

where *Y* is Young's modulus.

Piezoelectric materials have a behavior that combines mechanical stress and electricity. The electrical behavior of the material is given by the electrical displacement:

$$D = \varepsilon_0 \varepsilon_r E \tag{33}$$

where *D* is the electric charge density displacement, ε_0 and ε_r are the vacuum and relative electrical permittivities and *E* is the electric field strength.

On the other hand, mechanical behavior is modeled according to Hooke's law:

$$S = sT \tag{34}$$

where *S* is the normalized measure of deformation or strain, *s* is the resistance to deformation, or compliance and *T* is the value of the deformation force over a unit of area, also technically defined as stress .

A piezoelectric harvester has the electrical equivalent circuit presented in Figure 16. It behaves like a high AC voltage source in series with a parasite capacitor and resistor. The series resistance can be neglected for most usage scenarios, as it typically has a small value.

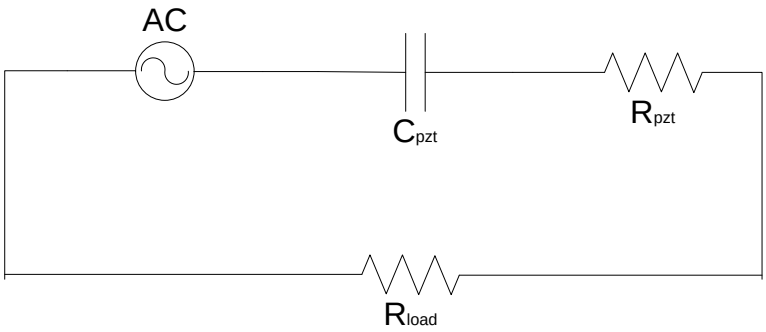


Figure 16 The piezoelectric generator model

Most of the piezoelectric materials employed in energy harvesting behave as a high voltage source with only very little amounts of generated current. A formula for the generated voltage of a piezoelectric element subjected to a certain amount of strain is given below.

$$V_{open} = -\frac{td}{\varepsilon s} S \tag{35}$$

where *d* is the strain constant, *t* is the thickness of the piezoelectric material, *S* is the strain, *s* is the compliance, and ε is the electric permittivity of the piezoelectric material. This is, however, an AC voltage which needs to be rectified in order to be able to power an electronic device.

The short-circuit current of such a piezoelement can be expresses as:

$$I_{sc} = fdASY \tag{36}$$

where *f* is the frequency of oscillation, *S* is the strain, *A* is the element's surface area, *d* is the piezoelectric strain constant and *Y* is Young's modulus.

We can assume that on average, a piezoelectric harvester connected to an optimal load will generate half the open-circuit voltage and half of the short-circuit current. Using this assumption and the equations (32), (35) and (36), we can estimate the average power output of a piezoelectric harvester :

$$P = \frac{1}{2} V_{open} \frac{1}{2} I_{sc} = \frac{1}{4} \left(\frac{td}{\varepsilon s} S \right) (fdASY) = \frac{fVk}{s} S^2 \tag{37}$$

where $V = tA$ and is the total volume of the element.

The piezoelectric harvester design which yields the best results employs a flexible cantilever structure which can be attached to the vibration source. The beam is made out of two piezoelectric crystals laminated on a sheet of flexible metal, usually steel. A tip mass is added in order to provide additional strain and to tune the cantilever's resonating frequency to that of the vibration source.

The natural frequency of oscillation of the cantilever beam in Figure 17 is [66]:

$$f_n = \sqrt{\frac{3YI}{L^3(M_b + 0.24M)}} \tag{38}$$

where I is the inertia moment, L is the length of the beam, M is the mass of the beam, and M_b is the bulk end mass and Y is the Young elasticity modulus.

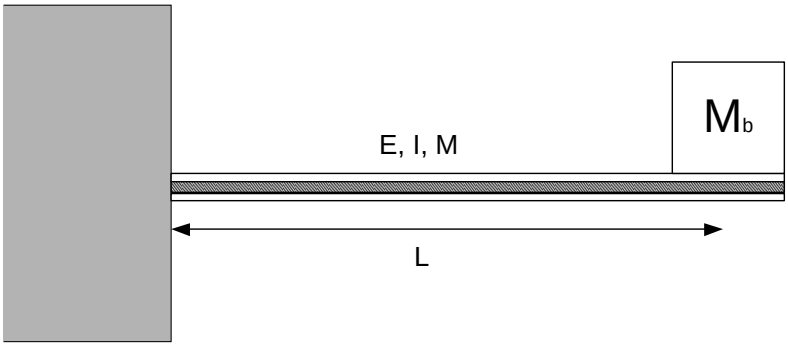


Figure 17. Cantilever beam with added tip mass

The concept of vibration harvesting can be successfully applied to a WSN node if it is attached to vibration sources such as stairs, microwave oven, door frame, external windows or even a person's shoe. These vibration sources can be harvested by attaching a piezoelectric harvester to the source.

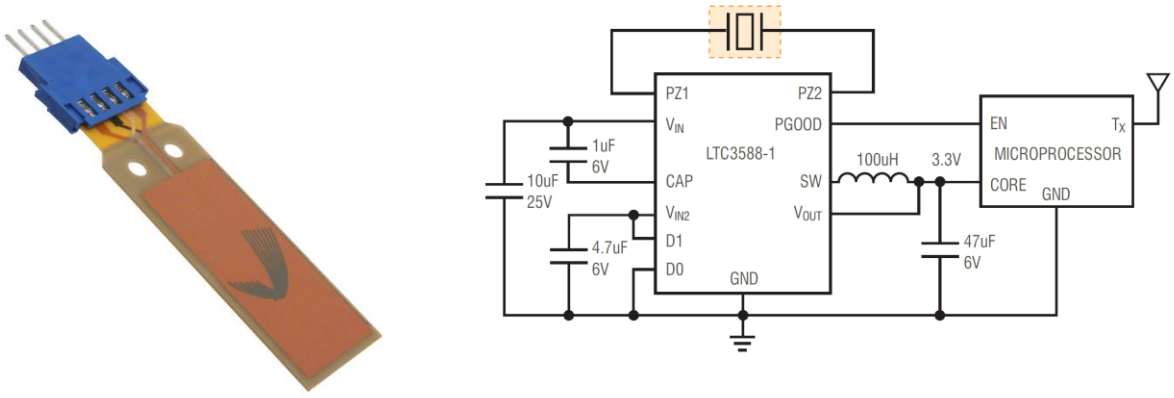
Vibration Source	Peak Frequency (Hz)	Peak Acceleration (m/s^2)
Blender Casing	121	6.4
Clothes Dryer	121	3.5
Door Frame	125	3
Microwave Oven	121	2.25
Ventilator Fan in Office Building	60	0.2-1.5
People Walking on Wooden Deck	385	1.3
External Windows (on a busy street)	100	0.7
Washing Machine	109	0.5
Refrigerator	240	0.1

Table 7. Typical vibration energy sources

The challenge for vibration harvesting is that not all vibration sources have the same frequency or amplitude, which makes it difficult to have a universal harvester device. Instead, most harvesters need to be tuned to a specific frequency, which can prove to be a costly solution. According to [67], typical everyday life vibration sources and their physical characteristics are given in the table below.

The technology has been researched in different universities around the world and commercial solutions are already available on the market.

One such device is Volture PEH20w Piezoelectric Vibration Energy Harvester which can deliver up to 9mW of electrical power when attached on conventional vibration sources.



a) b)

Figure 18. Vibration scavenging devices: a) Commercial Volture harvester, b) Vibration harvesting circuit employing a DC/DC converter [68]

3.3.3 Thermoelectric Harvesting

Thermoelectric harvesting relies on the observation that a thermal gradient between two conductors made out of different materials will generate a voltage. This phenomenon is known as the Seebeck effect and is explained by the fact that heat flow inside the conductor will result in a diffusion of charge carriers, which in turn will generate a voltage difference between the hot and the cool side of the two conductors. The reverse effect is also possible: running an electrical current through the two dissimilar conductors will generate a temperature differential, with the device acting as a heat pump. This effect is also known as the Peltier effect.

The Seebeck effect occurs when a voltage V is induced into an element that is subjected to a temperature gradient ΔT. The generated voltage is directly proportional to the temperature gradient:

$$V = \alpha \Delta T \tag{39}$$

where α is the Seebeck coefficient.

The above relation is mainly exploited in thermocouples for the exact measurement of temperature. However, if many thermocouples are thermally connected in parallel, they can generate enough energy to run a low power device, such as a wireless sensor node or charge a small battery [69].

These thermoelectric modules can be designed to generate electricity from a wide array of thermal sources, such as the exhaust pipe of an automobile, the nuclear decay of a radioactive element in RTGs or even the body heat of a human being.

The materials out of which thermoelectric generators are currently being built are P and N doped semiconductors, especially bismuth-telluride or silicon-germanium, due to its high Seebeck coefficient and low thermal conductivity. These semiconductors are set into arrays which are connected in series, in order to cumulate the voltage generated by each element. Thermally, they are all connected in parallel between two ceramic plates, which ensure good electrical insulation and thermal conductivity.

Compared to vibration harvesters, thermoelectric modules have the advantage of a long operational lifetime, because they don't contain any moving parts. The main disadvantage of thermoelectric harvesters is their low efficiency, which rarely exceeds 10% [70]. However, new advances in material science and especially in the development of materials that have low heat conduction but can conduct electricity will result in improved efficiency in the near future.

An electrical-equivalent model of a thermoelectric generator (TEG) is given in Figure 19. The TEG acts as a DC voltage source with an internal series resistance. Maximum power point matching can be achieved for the module if the internal resistance is matched to the load resistance. In the case that the TEG is charging a battery or a capacitor, an additional diode must be connected in series with the circuit in order to ensure that the stored charge does not flow back through the TEG when the thermal gradient is low.

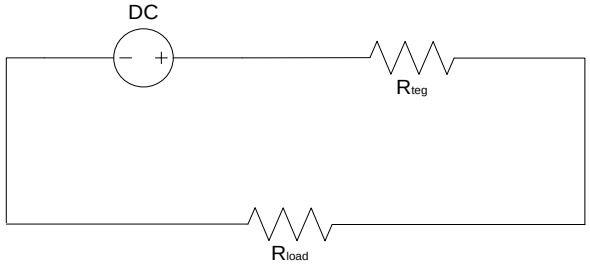


Figure 19. Electrical equivalent of a thermoelectric generator device

The calculated power output P_i based on the electrical-equivalent model presented above is:

$$P_i = \frac{(2\alpha\Delta T)^2}{4R_{TEG}} \quad (40)$$

,where α is the Seebeck coefficient, ΔT is the temperature difference across the thermoelements, and R_{TEG} is the total series electrical resistance, given by

$$R_{TEG} = 2n\rho\left(\frac{L}{A}\right) \tag{41}$$

,where ρ is the electrical resistivity, n is the number of thermoelement pairs, and L/A is the ratio of the thermoelement length to cross sectional area.

Figure 20 shows a typical thermocouple module with multiple thermoelement pairs of "p" and "n" branches which are linked in series by a metal interconnect, in order to maximize the harvested voltage. Thermally, the elements are all in parallel between the two sides of the module.

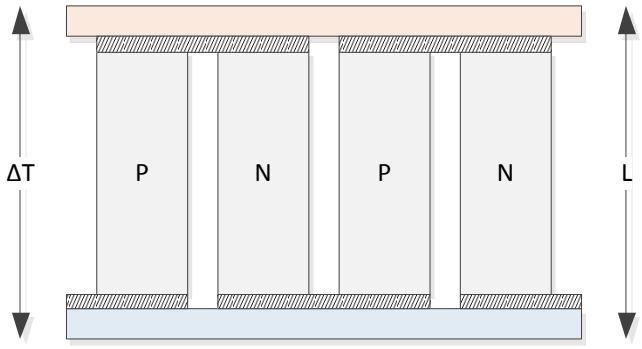


Figure 20. Diagram of an ideal Peltier device

To determine if a TEG can be efficiently used to power a WSN node, a series of experiments have been made. The thermoelectric element used was a Peltier cooler rated at 18V, 2A and capable of a maximum temperature differential of 50 degrees Celsius.

Each side of the element was fitted with one aluminum radiator and immersed in a bath of liquid. A very fine control of the temperature on one of the radiators was achieved by slowly heating the liquid into which it was immersed while the other radiator was kept at constant room temperature.

A potentiometer was used to simulate a variable load on the system, which would replicate the real life conditions of a wireless node consumer.

Two types of experiments were conducted. The first one intended to measure the power output of the system on a static load while the second experiment measured the power output on a variable load.

For the first experiment, a static load resistance of 1KΩ was chosen. The temperature differential on the two plates was varied while measuring the current generated by the element and the voltage drop across the load.

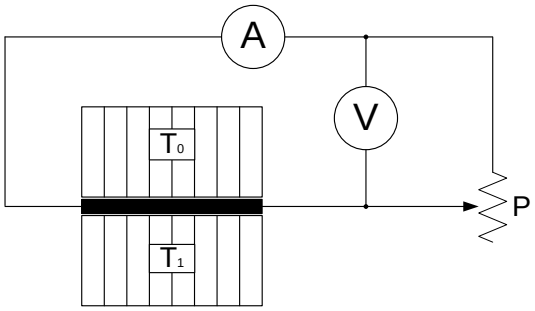


Figure 21. The experimental setup used to determine the efficiency of current generation in a TEG

As predicted by the Seebeck effect law, the voltage drop across the load was measured to be in direct proportion to the temperature variation. A maximum voltage of 1,065V was achieved for a temperature difference of 37.7 degrees Celsius.

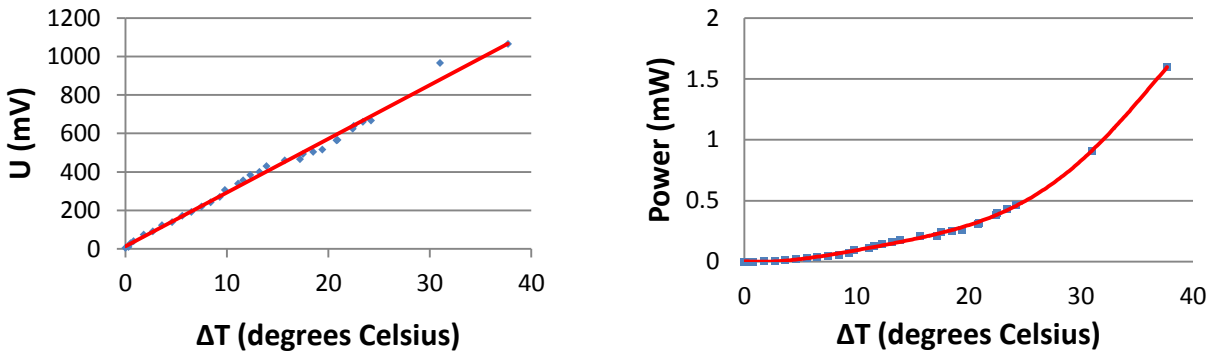


Figure 22. Generated output voltage and power as a function of temperature differential between the two sites of the Peltier element.

The power output of the system, as a function of the temperature differential is plotted in Figure 22. The maximum measured power was 1.6mW for a temperature differential of 37.7 degrees Celsius.

The second experiment focused on measuring the output power of the generator for a variable load. For this, the temperature difference between the hot and the cold plates of the thermoelectric element was kept constant at 35 degrees Celsius. The load resistance was varied while measuring the values for current and voltage.

Figure 23 shows the variation of the load voltage depending on the current in the circuit.

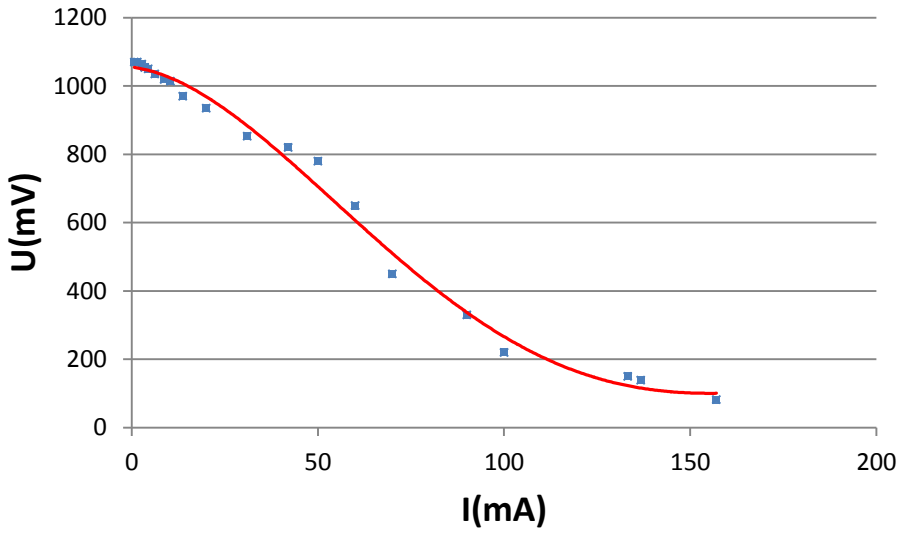


Figure 23. Voltage drop across the variable load resistor as a function of the circuit current

The purpose of this experiment was to determine the circuit load that yields maximum power from the generator. This information is useful in the design of a circuit that uses the energy from the TEG to charge a battery or a super-capacitor. Such an optimal charger circuit can be made to track this maximum power point (Figure 24) and give the best performance in any given situation.

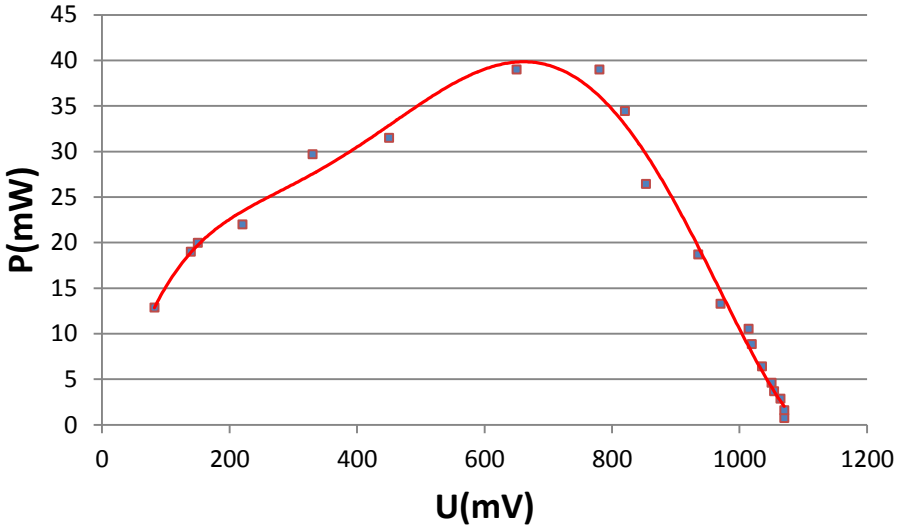


Figure 24. Peak generated power as a function of the load voltage drop.

For this experiment, the maximum power achieved at a temperature differential of 35 degrees Celsius was 39mW (0.78V at 50mA). This power is enough to continuously power a Sparrow WSN node (minimum power is 36mW), but environments that have a sustained high temperature differential as the one presented are scarce. However, with adequate energy buffering it is more than enough to ensure a partially-on power scheme in which the node

periodically wakes up from a sleep state, takes measurements and relays that information to the sink node, before going back into sleep mode.

Thermoelectric energy scavenging can be adapted to WSN nodes that lie in an environment with moderate temperature variations. In order to produce electricity, a node can be placed on a hot surface, such as a room heater or radiator or even on a window to make use of the temperature difference between a room and the outside.

A high efficiency system can be deployed if a node is attached to the exhaust pipe of an automobile. The internal combustion engine in automobiles is highly inefficient, and much of the energy is transformed into waste heat which dissipates to the environment. According to studies, from 100% of energy produced by an internal combustion engine, 40% of the energy generated is dissipated through exhaust. If even a small fraction of this energy could be reclaimed, it could be used to run electronics and reduce the electrical loading on the engine.

The sensor node attached to the vehicle can measure reliable information on the state of the vehicle and even gas emissions without ever having the need of replacing its batteries. In this scenario, the system can log the measured parameters in its internal memory and download them to a wireless access point located in the automobile's garage or even at fixed locations along the road.

3.3.4 Solar Energy Harvesting

One of the most widely-employed energy harvesting techniques uses the photovoltaic cells in order to generate electrical power from ambient light. This type of energy harvesting is usually employed in places with high availability of light and where mains or battery supply is impractical, such as remote outdoor locations.

The basic unit of a solar converter is the photovoltaic cell. It is semiconductor a device with two terminals which behaves like a diode when not illuminated and generates a voltage when subjected to luminous flux. The surface of the cell is designed to be light absorbing and is strewn with metallic contacts for establishing the required electrical connections.

A basic cell will typically generate from 0.5V to 1V when subjected to full illumination from the sun and a current of around 10 milliamps per each square centimeter of surface. In order to generate bigger voltages or currents, the cells are connected in series or parallel arrays of 20 or 30 identical modules. Standard output voltages for a solar panel constructed in this manner is 12V or a multiple of this value.

As almost all solar panels rely on solar illumination for energy production, there will always be an interval when generated power will not be sufficient for powering the given load. For this reason, most photovoltaic systems also include a power storage subsystem which acts as an energy buffer during periods of low energy production.

A solar cell behaves as a voltage source, with a current that depends on incident illumination. A good metric is the photovoltaic current density at short-circuit, or J_{sc} which has the following formula:

$$J_{sc} = q \int b_s(E)\eta_{QE}(E)dE \tag{42}$$

where $b_s(E)$ is the flux density for the incident light and $\eta_{QE}(E)$ is the quantum efficiency, or the probability that an incident photon will dislocate a corresponding electron from the silicon substrate. The quantum efficiency plays a vital role in cell performance under different conditions, due to the fact that it gives the efficiency of charge absorbtion by the device and the absorbtion coefficient of the photovoltaic material.

A solar cell also exhibits a dark current, which flows through the device when an external voltage is applied and the cell is not subjected to any incident illumination. Under these circumstances, solar cells behave like diodes, with higher forward currents than when polarized in reverse. We can express the dark current density in a similar fashion to the shortcircuit current density:

$$J_{dark}(V) = J_0\left(e^{\frac{qV}{kT}} - 1\right) \tag{43}$$

where J_0 is constant, *k* is Boltzmann's constant and *T* is the temperature.

When the solar cell is not connected to any load, the voltage difference at its terminals has a maximum value. This is known as the open circuit voltage, or V_{oc} and can be expressed as

$$V_{oc} = \frac{kT}{q} ln\left(e^{\frac{J_{sc}}{J_0}} + 1\right) \tag{44}$$

From the above equation, we can deduce that output voltage will increase logarithmically with incident light intensity.

From an electrical standpoint, the solar cell's internal circuit is equivalent to an ideal current source which is in parallel with a diode. To this ideal model we need to add parasitic resistances: a shunt resistor in parallel to the diode and a series resistor. When subjected to illumination, the cell produces a current which is proportional to the incident light intensity. The diode in the model provides the actual output voltage of the solar cell, due to the fact that a part of the photocurrent flows through it and the remaining part through the load. For small loads, this current is diverted mainly through the load and, as a result, the output voltage of the panel will decrease. The opposite is true for loads that have large values.

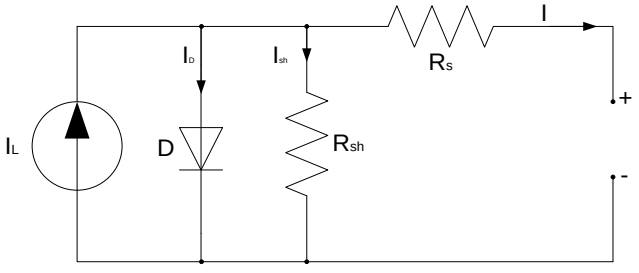


Figure 25. Equivalent schematic of a photovoltaic cell.

By analyzing the equivalent circuit we can express the current produced by the solar cell the current produced by the source from which we substract the current which flows through the diode and the current which is lost through the shunt resistor:

$$I = I_L - I_D - I_{SH} \tag{45}$$

The current flowing through these elements governed by the voltage across them:

$$V_j = V + IR_S \tag{46}$$

where V is voltage across the output terminals, I is the output current and R_S is the series resistance .

We can define the power density of the photovoltaic cell as follows:

$$P = JV_j \tag{47}$$

When the power density reaches maximum, the *maximum power point* for the photovoltaic cell is achieved. Presuming that this occurs for a certain voltage V_m and current density J_m, we can estimate the efficiency of the photovoltaic cell as the ratio between maximum power density delivered and the power density of the incident light, P_l:

$$\eta = \frac{P_m}{P_l} = \frac{J_m V_m}{P_l} \tag{48}$$

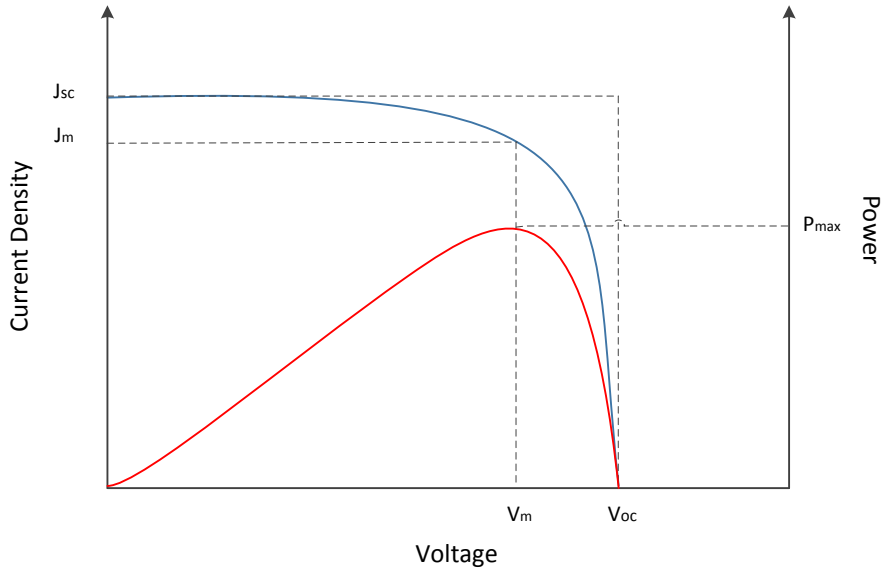


Figure 26. Power and current density variation to output voltage for a solar cell

For the *J-V* dependency plot in Figure 26, we can define the fill factor as the ratio between the power density at the maximum power point and the ideal power density obtained at V_{oc} and J_{sc}:

$$FF = \frac{P_m}{P_{max}} = \frac{J_m V_m}{J_{sc} V_{oc}} \tag{49}$$

The fill factor defines the roundness of the J-V curve and is an important parameter in modeling the behavior of a solar cell, along with J_{sc}, V_{oc} and the efficiency, η. We can express all parameters and incident luminous flux in a single equation by substituting P_m in (48) with the value from (49):

$$\eta = \frac{J_{sc} V_{oc} FF}{P_l} \tag{50}$$

We can further elaborate on (57) by expressing the current that flows through the diode as:

$$I_D = I_0 \left(e^{\frac{qV_j}{kT}} - 1 \right) \tag{51}$$

, and the current that is flowing through the parasitic shunt resistance as:

$$I_{sh} = \frac{V_j}{R_{sh}} = \frac{V + IR_s}{R_{sh}} \tag{52}$$

By replacing these into (57), we get the characteristic equation of a solar cell:

$$I = I_L - I_0 \left(e^{\frac{q(V + IR_s)}{kT}} - 1 \right) - \frac{V + IR_s}{R_{sh}} \tag{53}$$

To test the efficiency of solar energy harvesting, the performance of an amorphous silicon PV cell has been measured in a series of experiments.

The experimental testbed is described in the schematic below. The PV cell used was rated at 2V and a short-circuit current of 200mA at maximum illumination. A potentiometer was used to simulate the variable load of the system while an ammeter and voltmeter were used to measure the power output.

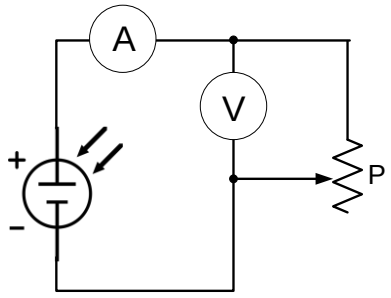


Figure 27. The experimental setup for PV cell efficiency testing

The first experiment involved exposing the cell to different illuminations and measuring its voltage-current characteristic. Light intensity was measured by a luxmeter and varied from full sunlight (approx. 28000 lux) to the conditions inside a regular office room (2000-5000 lux) and a dark room (100 lux). Current and voltage across the variable load resistance were measured and the characteristics can be seen plotted in Figure 28.

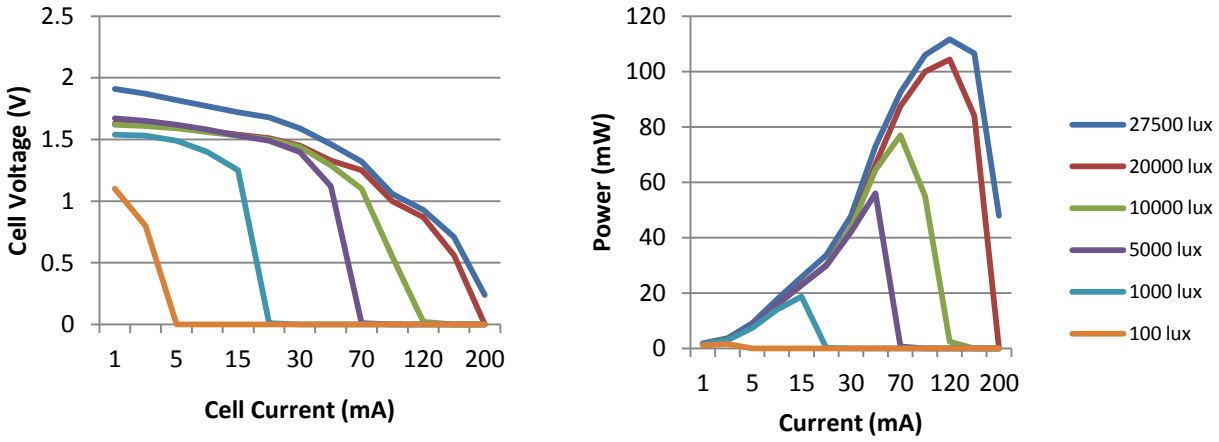


Figure 28. Cell voltage and generated power on a variable load for different illuminations of the photovoltaic panel.

From the above figure we can draw several conclusions. The first one, dictated by the model we presented earlier, is that the photovoltaic panel behaves as a current source which is limited in voltage.

The second observation is that by drawing the I-V curve, we can easily observe and measure the maximum power point of the cell for different illuminations. At this point, power extracted from the cell is at a maximum.

Another observation is that as the illumination decreases, so does the amount of current produced by the cell, which has a direct effect on its maximum power point. From Figure 28 we can see that this point is reached at lower currents as the illumination is decreasing. Due to this behavior, a solar panel cannot be used to ensure the continuous functioning of a device on its own. Most solar harvesters employ a secondary energy buffer which stores and releases power in a complementary manner, in order to ensure a steady supply for the system.

The second experiment intended to measure the total energy that can be harvested with a single PV cell over a long period of time. The goal of this experiment was to determine whether solar panels are indeed capable of powering a low power embedded device, such as a wireless sensor node over indefinite periods of time. In order to achieve this, we measured energy production in two locations for a period of ten days and averaged the measurements. The two locations were chosen as complete opposites: the first was situated outdoors with an un-obscured day-long view of the sun, and the second was indoors, on top of a desk in our laboratory.

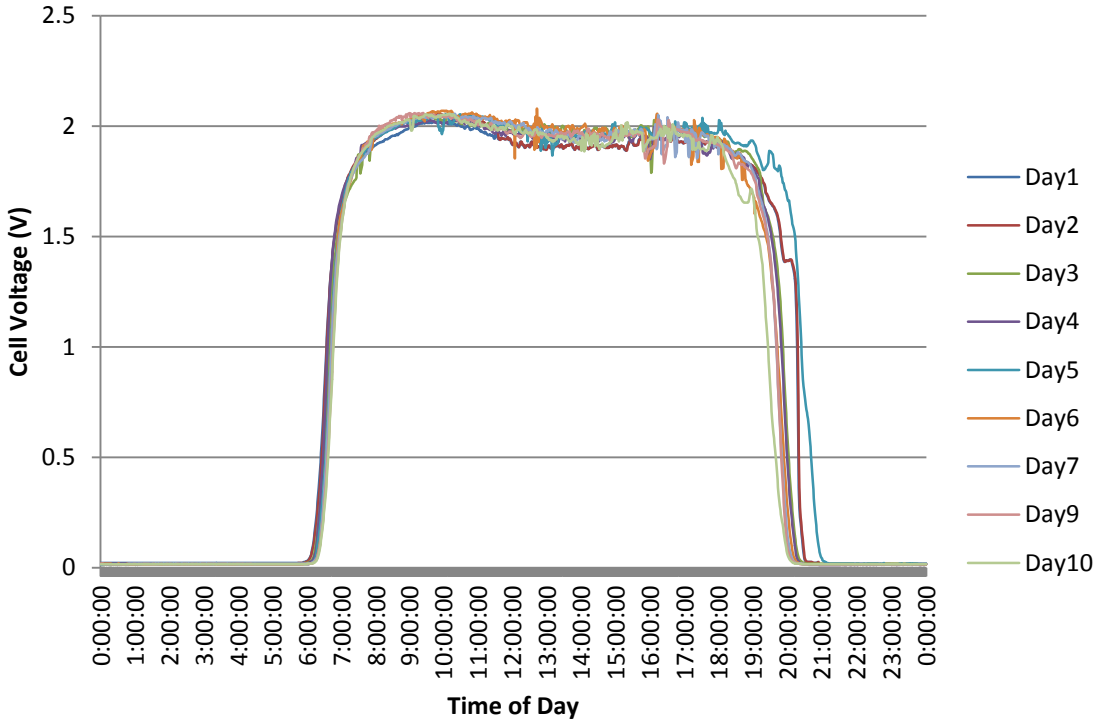


Figure 29. Photovoltaic cell voltage drop measured in the course of ten days for the outdoor location.

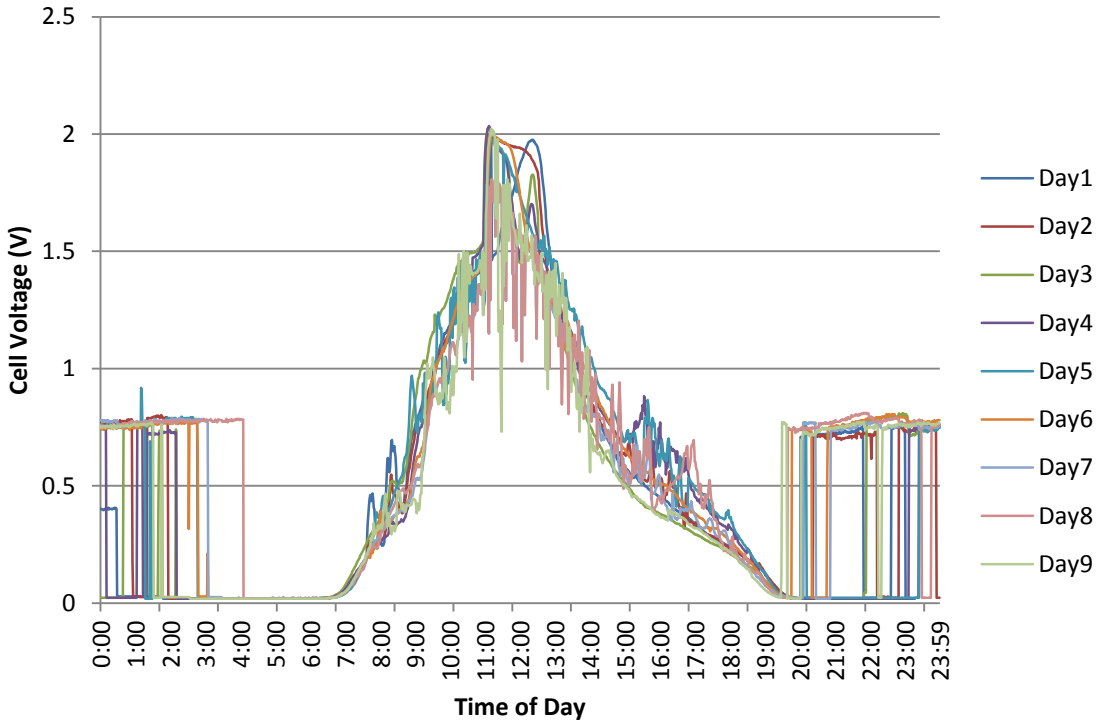


Figure 30. Photovoltaic cell voltage drop measured over ten day for the indoor location

From the perspective of received incident illumination, the first location was as close to ideal as we could get, while the panel situated indoors had only a fraction of the incident light received by the former and was thus considered to be the worst case scenario.

Each solar panel was connected in parallel with a 1KOhm resistor which would simulate the load and with a current and voltage measurement system which would log these two parameters into its memory every sixty seconds. The graphs for power production in the two locations are given in Figure 29 and Figure 30.

From the graph in Figure 29 we can see that the panel is capable of supplying the load with a stable voltage during the day, even though, due to weather conditions, the illumination of the panel was not constant. We can also see that there is a sharp increase in power production at sunrise followed by a similar sharp decrease at dusk, with close to zero generated power during the night. These conditions are very close to an ideal behavior and we can assume that through careful installation, they can be achieved by any wireless sensor system that employs solar energy harvesting in an open environment, such as a forest or an urban area.

The graph in Figure 30 shows power production for the indoor location over a series of days. Compared to the previous figure, energy production is lower and shows a wider variation during daytime. Due to low lighting conditions, the panel was unable to supply a steady amount of power to the load. We can see large variations, especially during days eight and nine of the experiment, when the weather was mostly overcast. However, we can also observe that there is a significant amount of power produced during the night which is mostly due to the photovoltaic panel scavenging the room's interior lighting.

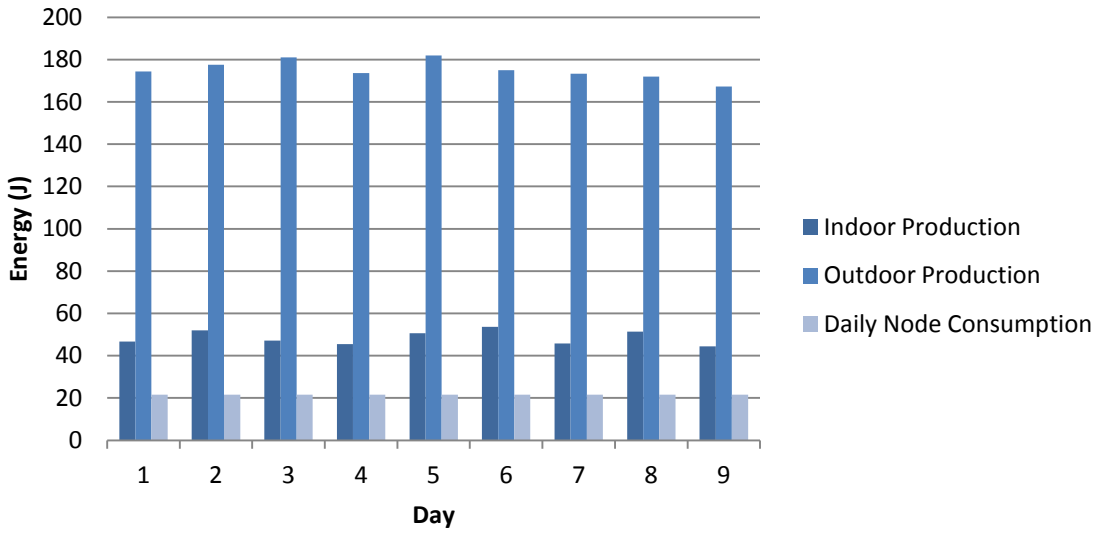


Figure 31. Comparison of indoor and outdoor harvested energy from the solar panels

Figure 31 shows a comparison of the daily energy harvested by the panel in the two locations over the whole duration of the experiment. We can see a clear difference between the two, the energy gathered by the outdoors PV cell being on average three to four times larger than

the one harvested by the indoor panel. The purpose of this experiment was to determine whether photovoltaic harvesting is capable of powering a sensor node indefinitely, even in the worst case scenario. We can estimate the total energy consumption of the Sparrow sensor node presented in Chapter 4 and compare it to the produced energy.

The Sparrow Wireless Sensor node has a current consumption of 20mA at a stable supply voltage of 1.8V. This yields a power consumption of 36mW. If the node were to function uninterrupted during 24 hours at this power level, it would require an energy of 3310.4 Joules, which is more than the solar panel can offer. However, as is the case with all wireless sensor nodes, the Sparrow node employs duty-cycling in order to reduce its power consumption. During its low power sleep state, the node draws roughly 5μA of current from the power supply. Almost all the applications in which we used the node had a duty cycle of five minutes, during which the node was fully operational only for 2 seconds and the remaining 298 seconds were spent in a sleep state. When applying the same calculations on energy spending for the whole day, we came op to a daily energy consumption of 21.5 Joules, as shown in Figure 31.

This proves that, given enough storage capacity and enough incident radiation, solar energy harvesting can power a node for an indefinite amount of time. The excess energy produced during the day can be stored in a battery or super-capacitor to be consumed during the night or on clouded days.

3.3.5 Radio Frequency Harvesting

A rectifier antenna, or rectenna is a device used to convert radio frequency energy into electrical energy through the use of a rectifier circuit attached directly to a regular antenna. For most purposes, rectennas are used for microwave energy transmission, due to their high efficiency in converting microwaves into electrical energy. In some situations, efficiencies of over 85% [71] have been measured. This figure gave us the idea of considering the use of rectennas in powering Wireless Sensor Nodes.

There are many publications that deal with rectifier antennas and their design, such as [72], [73], [74]. In the past, research was focused on the efficient transmission and reception of high amounts of power, but in recent years the paradigm shifted to the capture of microwave radiation of relatively low power densities [75], [76].

This is the case of wireless sensor nodes, which, due to their specifications, need to operate far away from the RF transmission source. We focused on the development of such an antenna, which operates at low power densities but has good conversion efficiency because it would allow one to power nodes located several feet away from the emission source.

A simple rectifier antenna consists of a dipole and a diode connected across its elements, usually a Schottky diode due to their low forward voltage drop. The alternative current that the antenna picks up from the environment is rectified by the diode in order to produce a DC voltage output. Of course, multiple rectenna cells can be combined into an array, in order to increase the amount of energy gathered from the environment.

When employed in Wireless Sensor Networks, rectennas are normally tuned to the ISM frequency bands which impose restrictions on the maximum transmission power. Thus, the challenge is to formulate a very efficient design within reasonable dimensions.

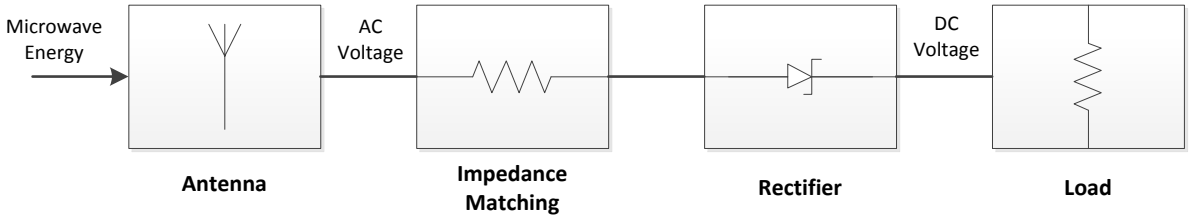


Figure 32. Typical rectenna circuit

A typical rectenna setup is shown in Figure 32. The microwave energy is converted into AC voltage by the antenna and then passed through an impedance matching circuit to the rectifier, which, in turn, converts it into DC voltage for the load. The impedance matching circuit ensures maximum power transfer by matching the input impedance of the rectifier to that of the antenna. In some designs, however, the matching circuit is rendered useless by creating an antenna that matches perfectly the rectifier input impedance.

In our research regarding wireless energy transmission, we studied its applicability on a family of antennas called microstrip patch antennas. The reasons for this choice are the small dimensions of the antenna and its low manufacturing cost, which would make it an ideal choice for wireless sensor networks.

The antenna designed in this section is tuned for the 2.4GHz ISM band, as most of the WiFi, Bluetooth and Zigbee transceivers that can be found in a typical use-case scenario utilize this frequency for communication.

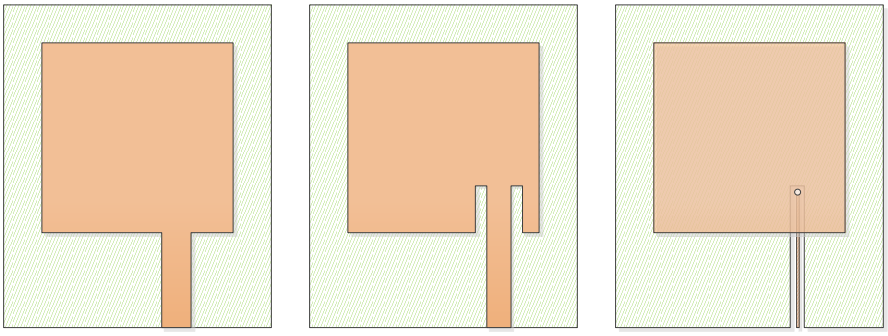


Figure 33. Different types of feeds for rectangular microstrip antennas. From left to right: edge feed, inset feed and probe feed.

The simplest model of a patch antenna consists of a conducting top layer, which forms the effective surface of the antenna, an insulating dielectric substrate and a bottom conductive ground plane. Due to this fact, most patch antennas can be easily fabricated out of a two layer

PCB, which significantly decreases their price. Most patch antennas differ in design by the way they are fed, as shown in Figure 33.

For our purposes, we settled on designing a patch antenna that is using edge feed, as an inset feed would introduce some parasitic capacitance between the line of the feed and the patch itself, which could cause performance degradation.

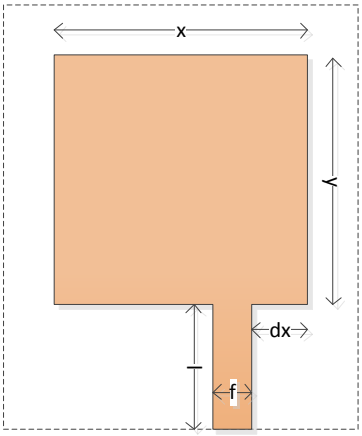


Figure 34. Dimension specifications for an edge feed patch antenna

Design and simulation of the patch antenna was done with the help of Ansys' HFSS suite, which is a 3D full-wave electromagnetic field simulation tool employed in high-frequency and high-speed component design. The antenna was made out of a FR4 double layer PCB with a thickness of 1.6mm. Its relative permittivity is $\varepsilon_r = 4.28$ and a loss tangent, tan (δ) of 0.016. The conductivity of the copper layers is $\sigma = 5.8 \cdot 10^7 \Omega^{-1} m^{-1}$.

Finding the antenna patch dimensions for the frequency of 2.45GHz was done through iterative testing and simulation in the full-wave simulator software, choosing a design that yielded the best results. The dimensions we found for the patch antenna in Figure 35 are the following: *x=27.7, y=30.8, dx=0mm, dy=0.45mm* and *l=0.4mm.*

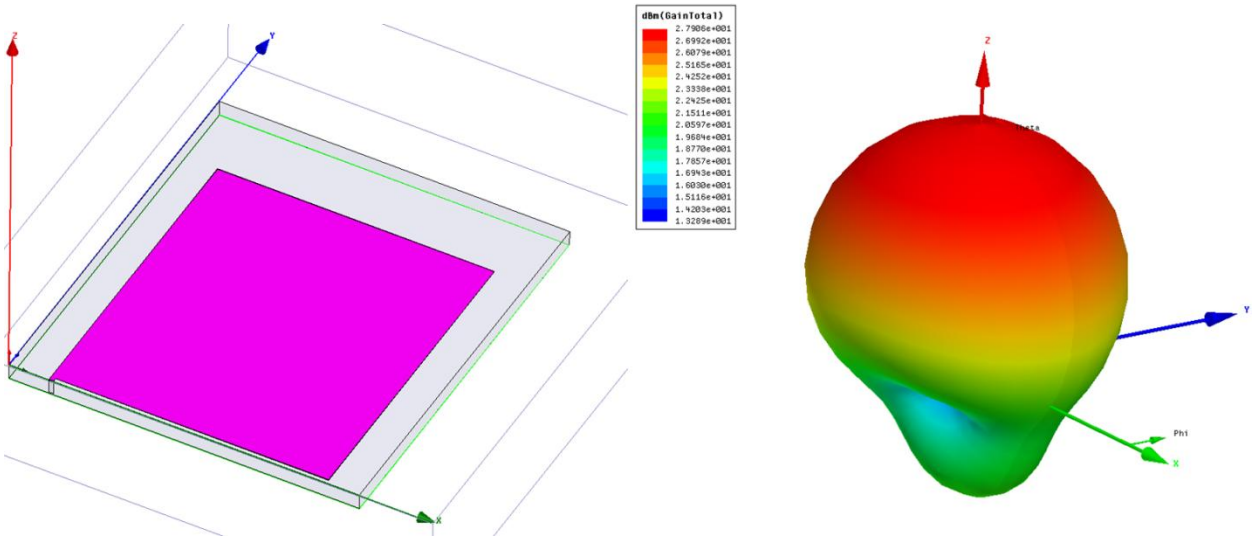


Figure 35. Patch antenna model and simulated 3D radiation pattern

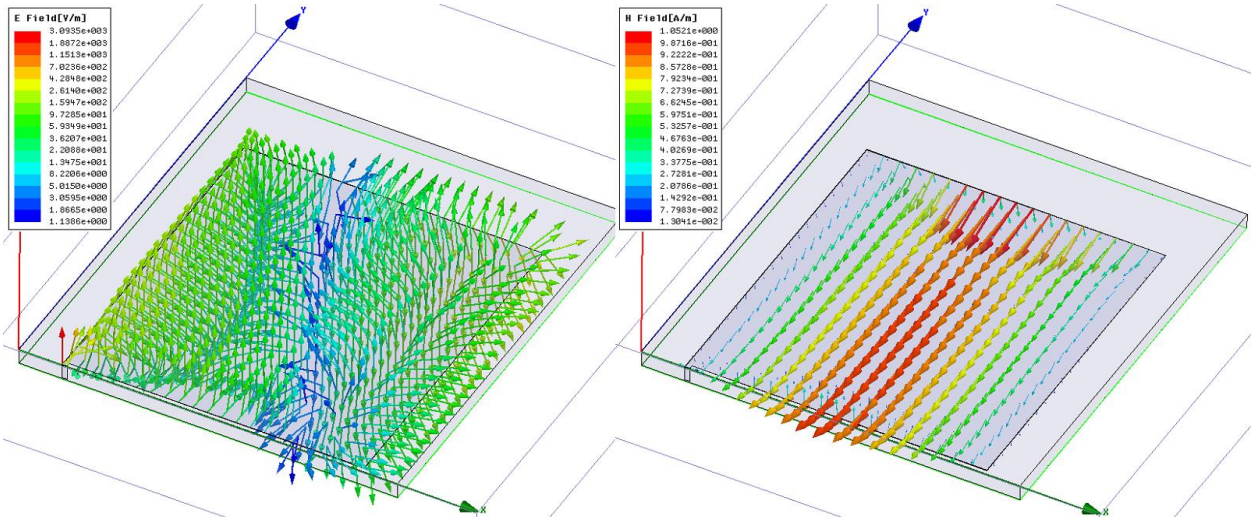


Figure 36. Electric (E) and magnetic (H) field distribution inside the patch antenna

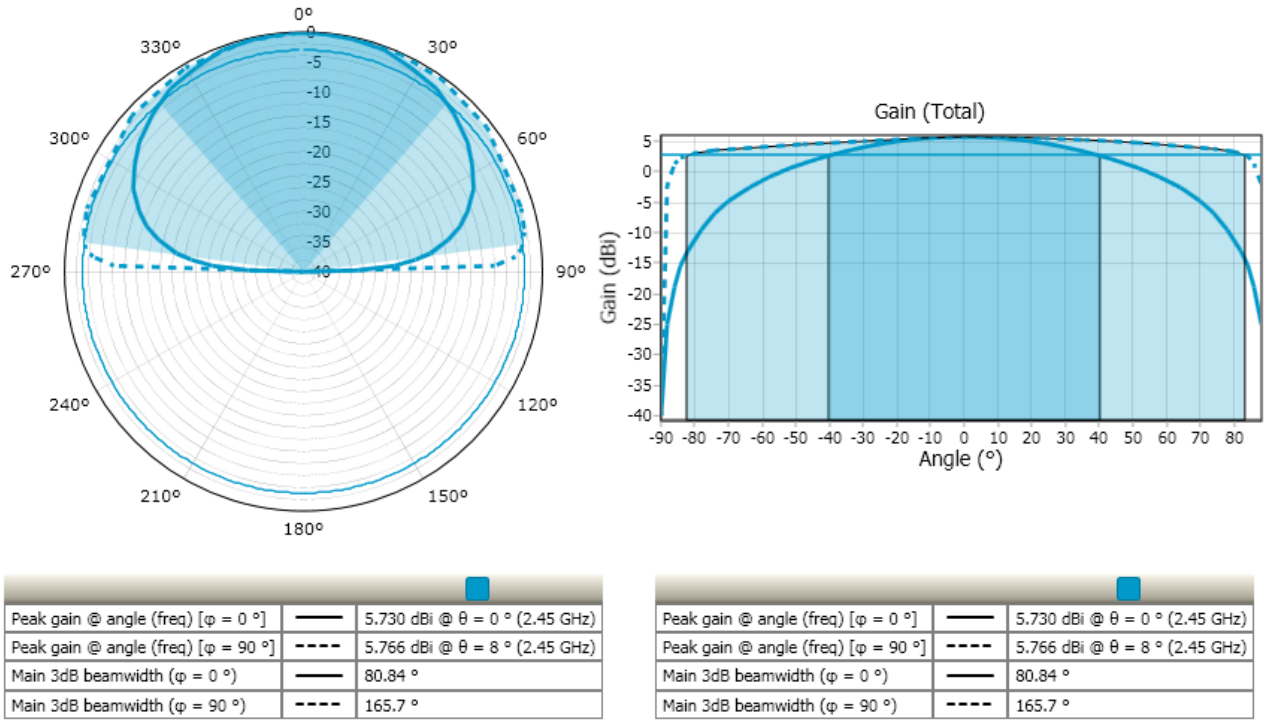


Figure 37. Total antenna gain patterns in polar and Cartesian coordinates

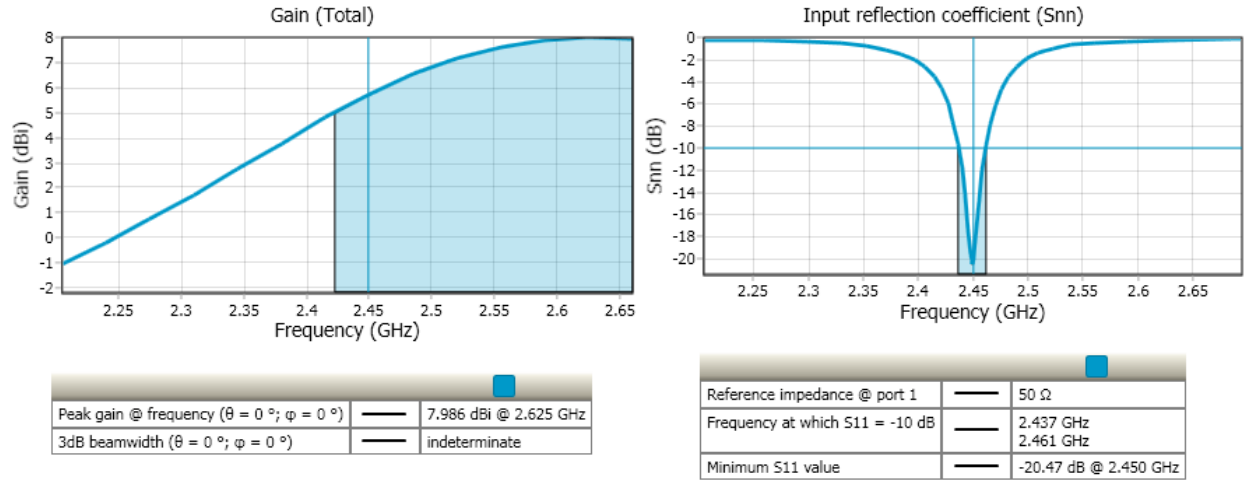


Figure 38. Gain and antenna reflection coefficient vs. frequency

Simulation data shows that the antenna resonates at the frequency of 2.45GHz as shown in Figure 38 and that it can deliver a gain of around 6dB for that frequency, which is encouraging, taking into consideration its dimensions.

The rectifier antenna system was formed when the antenna was coupled with a diode rectifier circuit, as shown in Figure 39.

The antenna voltage output is a 2.4GHz sinusoidal AC voltage of varying amplitude, depending on the distance from the transmitter and its power. I order to convert it to a DC voltage, the rectifier circuit in Figure 39 is employed. However, due to the very low voltages induced in the antenna, a very efficient diode rectifier circuit is used which also acts as a voltage doubler. The capacitor C1 value was chosen to act as a shortcircuit at 2.4GHz and the two Schottky diodes, D1 and D2, are connected in series, virtually doubling the rectified voltage.

The most important aspect of designing such a circuit is matching its input impedance to that of the antenna. The diodes we used for our design were HSMS2852, two zero bias small signal diodes in a single package. Their low forward voltage drop and fast switching time proved them ideal for our purposes. We calculated their input impedance and halved it, as they can be viewed as connected in parallel at microwave frequencies. This gave us the input impedance for the rectifier circuit which we used to determine the patch antenna dimensions that would yield matching impedance. Results were incorporated in the design of the patch antenna described in Figure 35.

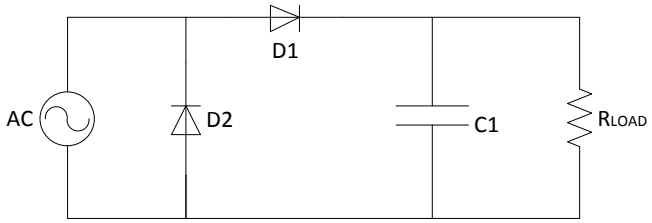


Figure 39. Voltage doubler rectifier circuit

Efficiency of the RF energy harvester circuit presented in Figure 32 can be derived as the ratio between the power available at its output and power collected by the antenna from the environment. This relation can be expressed as:

$$\eta = \frac{P_{DC}}{P_{recv.}} = \frac{V_{DC}}{{R_{load}}^2 \cdot P_{recv.}} \tag{54}$$

The Friis equation allows us to compute the received power of an antenna as a function of the distance from the transmitter.

$$P_{recv.} = P_T \cdot G_T \cdot G_R \cdot \left(\frac{\lambda}{4\pi R}\right)^2 \tag{55}$$

, where P_T is the transmitter power output, $\cdot G_T$ and G_R are antenna gains, λ is the wavelength of the microwave radiation and R is the distance between transmitter and receiver.

In order to test the effectiveness of powering a device using scavenged microwave energy, an experiment has been set up. We needed to test the harvester in a situation close to a real life scenario, so we employed a standard 2.4GHz wireless router as an energy source. The output of the router was connected to a cylindrical wave guide and the router was set to transmit at maximum power, which we measured to be around the regulated limit of -10dBm or 100μW.

The harvester was oriented parallel to the wave guide aperture, in order to ensure the highest power transfer possible. The harvester was connected in parallel with a 10mF capacitor that was intended to simulate an energy storage medium. Several measurements of the capacitor charging curve were taken while incrementally increasing the distance from the wave guide from 20 to 60 centimeters.

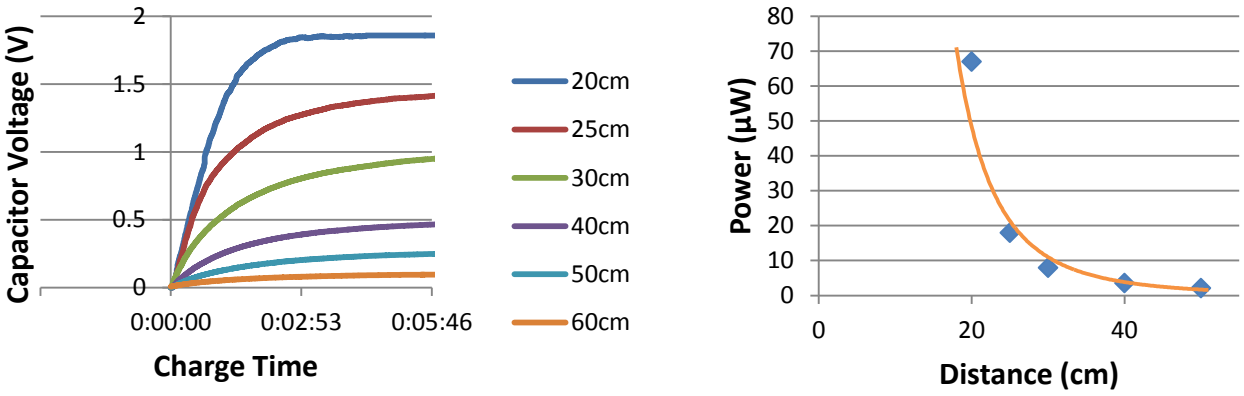


Figure 40. Capacitor charge and average received power depending on the distance from the microwave emitter

The capacitor charge curves are plotted in Figure 40 and they clearly show that energy is being received by the patch antenna and converted into usable DC voltage. After each charge, capacitor voltage reaches a steady value which decreases with distance. This is more clearly shown in the second plot from Figure 40 which shows how the average DC power generated at the output of the RF harvester varies over distance. Maximum power was achieved at 20cm and was 67µW (due to experimental setup constraints, this was the nearest we could reliably measure the power transfer), or a 67% efficiency in energy transfer from the transmitter to the harvester. However, this figure dropped dramatically when we increased the distance between the two entities to around 2µW at 0.5m. This is in accordance with the Friis equation in (55), where is it shown that received power varies by the inverse square of the distance from the transmitter.

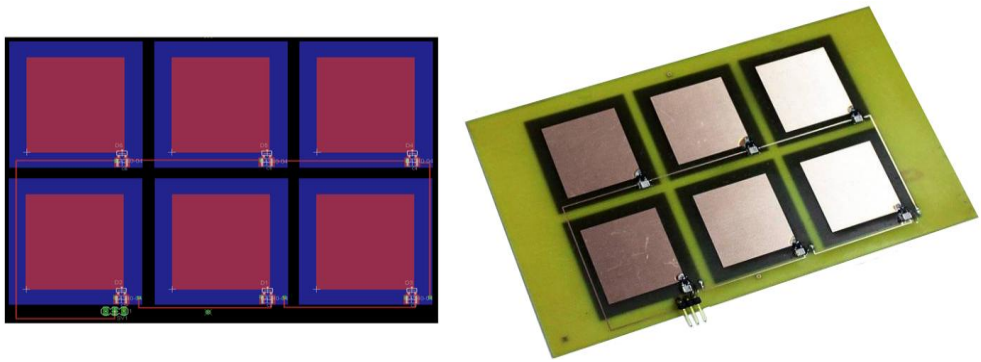


Figure 41. Rectenna array of six elements employed as a wireless battery. Printed circuit board design and actual harvester.

In order to overcome the drop in harvested RF energy with distance, a simple solution of using multiple rectifier antennas was successfully employed. The antennas were all etched on the same PCB in an array and linked together in series to form a wireless battery, as shown in Figure 41.

Chapter 4

A New Approach In the Design of Wireless Sensor Nodes

For the past decades, the speed of processors and memory has increased at an exponential rate reaching to orders of magnitude more than even ten years ago. However, during the same timeframe, battery energy density has only tripled which denotes an ever increasing gap between storage and supply [77]. Energy is a precious resource in WSNs due to the fact that almost all nodes run on batteries and their deployment nature and placement does not allow for an easy battery change or recharge. In this context, it is necessary to design every component of the node, from its hardware to the last level of its software stack, in order to minimize energy consumption. Although there has been some significant research in the area of power management, the problem of extending a node's lifetime is still an open research question, especially when deploying a sensor network in a real-life environment [78].

As described in Chapter 1, power consumption for most wireless sensor nodes is low enough that it enables us to explore the possibility of employing renewable energy sources to power nodes in an efficient manner. Such sources have been described in the previous chapter and their efficiency at gathering energy from the surrounding environment has been analyzed. There is also a large body of articles that cover this topic [79], [80], [81], [82], [83], [84].

As a conclusion to the experiments made in the previous chapter and of the previously mentioned articles, solar energy is the most promising candidate for powering a sensor node in an outdoor location. Currently, photovoltaic panels are employed in charging the batteries of sensor nodes as well as supplying power to their circuitry [85], [86], [87]. As a defining trait, these systems, either have a low efficiency and achieve only small increases in node lifetime, or rely on degradeable storage elements, such as NiMH batteries. Using a solar panel to recharge

such batteries on a daily cycle puts an extra strain which leads to a significant decrease in their lifetime.

Therefore, the main points a designer should bear in mind when building an inexhaustible power supply for a sensor node are not only related to the means of producing energy but also taking into consideration its storage.

Sparrow [88] is a wireless sensor network architecture that has been built as a research platform for the energy harvesting techniques described in the previous chapter. It was also used to deploy and test a series of wireless applications including IEEE 802.15.4, 6LoWPAN and ZigBee networks. The node architecture is given in the Figure 42.

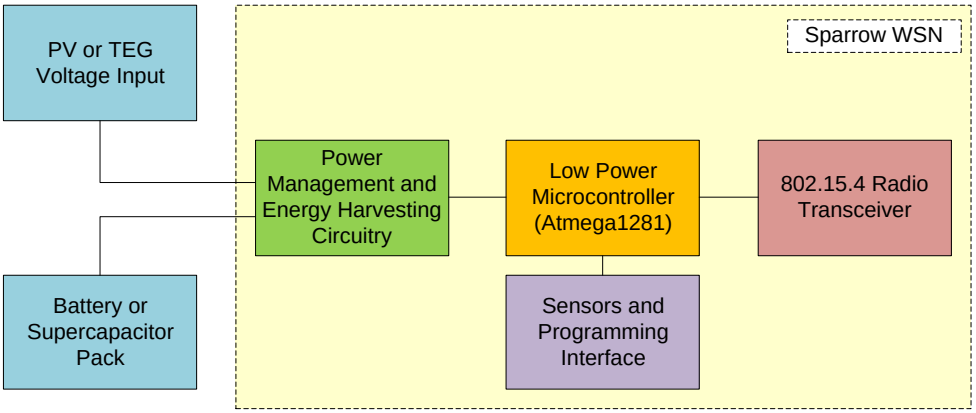


Figure 42. Sparrow mote architecture

In this chapter we present the results of our research in energy storage and delivery circuits. These circuits have been designed and incorporated into the Sparrow platform presented above. We will first study traditional energy storage systems, such as batteries and compare them to a new type of storage devices called super-capacitors. We will assess their efficiency at storing and releasing charge when coupled with the harvesting techniques presented in the previous chapter and determine whether these circuits are able to produce and store sufficient energy to power a sensor node indefinitely.

4.1 Energy Storage Systems

Sensor motes are designed to function with minute amounts of energy and have a lifetime measured in years; therefore, energy storage is a crucial aspect in order to guarantee a good WSN design.

Also of great significance is the power supply design for a wireless sensor node because a system that is not properly designed can use too many components, can be too expensive or affect node performance by drawing too much energy.

In concept, energy supply for a sensor node is a simple problem of balancing production and consumption. The node must balance its performance, which is directly linked to its power

consumption in order to adapt to variations in power supply. In practice, however, power supply is complicated by the fact that supplied energy is rarely optimal for the given load and that, due to power management states, the load itself varies over time.

Wireless sensor nodes are systems that exhibit, on average, low power consumption. However, there are instances when nodes require large amounts of power such as high computational loads or a sudden increase in network traffic. Although these conditions are exceptional and occur only in short bursts, the power supply system must be able to cope with them and ensure full functionality.

There are two major approaches to energy management systems. The first one employs electrochemical cell (batteries of super-capacitors) and aims to maximize the amount of energy drawn from the source in order to extend node life for as long as possible. Another approach is to use renewable energy supplies, such as photovoltaic or vibration harvesters in order to power the node. Due to low energy demands, sensor nodes are a prime candidate to such an approach. Even though the energy harvested from the surrounding environment is not significant in powering a modern embedded device like a smartphone, it is more than sufficient to power a sensor mote that employs good power management. The problem is still significant for nodes that have long periods of high power consumption, or the energy harvester fails to deliver enough energy (i.e. nighttime or long periods of cloudy weather for photovoltaics). For such exceptional cases, there is still need for an additional energy buffer or some sort of reduction in node activity to adapt to these low-energy periods. There is still a problem of the cost such a circuit would add to the sensor node.

4.1.1 Batteries

Due to their placement, most wireless sensor nodes do not have easy access to the mains or any other form of stable energy grid. The only suitable option for such a system would be the use of an individual energy source, and batteries, due to their low cost and ubiquitous availability, have proven a worthy candidate. However, employing batteries in a sensor node requires design considerations.

A battery is an electrochemical cell that has two electrodes separated by an electrolyte. They are characterized by parameters such as capacity, charge current, lifetime and charging speed. There are also other parameters which pertain to their imperfections, such as leakage or self-discharge current, number of charge/discharge cycles and temperature dependence.

Unfortunately, batteries do not keep up with the growth of computing power which characterizes the silicon industry. To paraphrase Moore’s Law, battery capacity for the same volume doubles only once every ten years, so there is a growing gap between the fast exponential increase in transistors per chip and the energy that can actually be delivered to them by modern-day batteries.

Another parameter which governs batteries is the charging time, which is expressed by the amount of time for the battery to reach its nominal discharge current. For example, in an ideal case, to charge a 100mAh battery, one would need to connect it to a 100mA power supply

for one hour (1C). Charging a battery twice the capacity with the same current supply would require two hours (2C), and so on.

However, due to imperfections, a battery such as the first one above would need more than 1C to fully recharge. Charging and recharging efficiency could prove a challenge for a system that has indefinite operation as one of their main specifications. Although new chemistries such as lithium polymer have delivered higher battery capacities in a small package, they usually require very specific charge conditions that can be achieved only by using dedicated integrated circuits. These increase the overall node cost and drain additional current only for operating.

Another design point that needs to be addressed is battery lifetime, which normally is limited by a certain number of charge-discharge cycles in rechargeable cells. Primary cells, which can't be recharged, have higher energy densities than their rechargeable counterparts for the same or lower price. However, in a deployment of a hundred or a thousand-node network, primary cells are less favored due to the cost of purchase and of periodically replacing them. Rechargeable cells avoid some of the aforementioned problems, but usually have a lower energy density, so recharges need to be performed more often than replacement of a primary cell.

For example, Mica2 is a sensor node uses a 16MHz microcontroller and a CC1000 900MHz radio transceiver that has a range of ~30m indoors. When at 100% duty cycle and powered by two AA batteries, the node can run for less than a day. When lowering the duty cycle to less than 1% by employing a power management algorithm, the node can survive for a few months. If a photovoltaic panel is added, the batteries can be recharged, which improves significantly the node operation time. However, due to a low number of recharge cycles and environmental conditions such as day-night temperature variations, the battery will need to be replaced after one or two years. Such maintenance costs are recurring and, when scaled to a network of a few hundred nodes, become very high if not prohibitive.

Battery use and recharging is a proven technology and has been used extensively in WSN node implementations. Because of this and of the shortcomings that were enumerated in the above paragraphs, this study focuses on finding and researching alternate technologies that can be used for energy storage at node level. One of the most promising technologies is analyzed in the next section.

4.1.2 Super-capacitors

A capacitor is a circuit element that consists of two metal plates insulated by a dielectric. Capacitance is given by the following formula:

$$C = \varepsilon_0 \varepsilon_r \frac{A}{D} \tag{56}$$

where ε_0 is the permittivity of free space, ε_r is the relative permittivity of the insulating material between the electrodes, *A* is the surface area of each electrode and *D* is the distance between the electrodes.

Electrolytic capacitors have one of the metal plates replaced by a chemical compound or electrolyte and have capacitances ranging from hundreds on nanofarads to thousands of microfarads. A special set electrolytic of capacitors are the super-capacitors or ultra-capacitors. Due to the discovery of porous substances like aerogel or activated charcoal, an unprecedented surface area can be squeezed inside a relatively small volume, thus allowing the creation of capacitors with capacitances in the area of tens to hundreds of farads.

Most super-capacitors store electrostatic charge in the form of ions. They obey the same fundamental theory as ordinary capacitors but achieve much higher capacitance due to the large surface area of the electrodes or due to thinner dielectrics. Because these devices store charge by electrochemistry, they are closer to conventional batteries, with comparable energy densities.

When modeling a super-capacitor, the equivalent series resistance or ESR must be taken into account as it can exert a heavy influence on charging time or voltage leakage over large periods of time. Also of importance is the equivalent parallel resistance, which usually has a high value and is responsible for the capacitor leakage or self-discharge. Another parasitic component of this first order model is the equivalent series inductance L of the capacitors' voltage terminals.

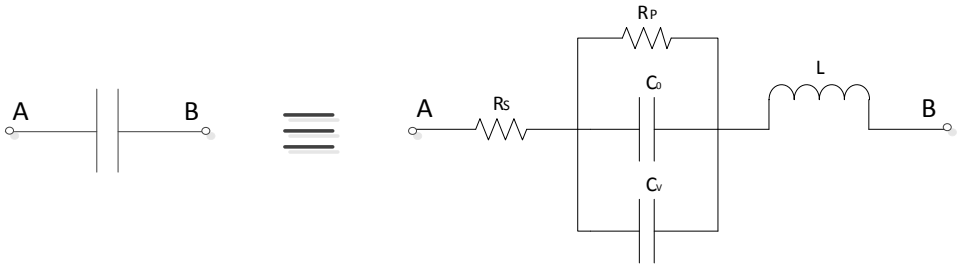


Figure 43. Schematic model of a real capacitor

The electrical equivalent model presented in Figure 43 is, however, a simplified first order model of a super-capacitor. Due to the porous nature of the electrodes used by electrical double-layer capacitors, their behavior resembles more that of long transmission lines. This model has been proposed and described in [89], but it does not take into account the variations in capacity due to charge voltage or temperature. These are addressed in [90], in which Zubieta et. al. propose an additional capacitance that has a linear dependence to the voltage. Also, similar models have been described in [91] and [92]. In these, the capacitor is modeled as comprised of two parts: a constant capacitance C_0 and a variable one, C_V which is directly proportional to the voltage: $C_V=K_VV$, where K_V is a constant given by the chemistry of the capacitor. These two capacitances are in parallel, so we can compute the total capacitance as being: $C = C_0 + C_V$.

By using this relation, we can determine the variation of the charge current with time inside a capacitor:

$$i(t) = \frac{dQ}{dt} = \frac{d(CV)}{dt} = \frac{d(C_0 V + K_V V^2)}{dt} = (C_0 + 2K_V V)\frac{dV}{dt} \quad (57)$$

We can also write the energy which a capacitor can store at a given voltage as:

$$E(V) = \int P(V)dV = \left(C_0 + \frac{4}{3}K_V V\right)\frac{V^2}{2} \quad (58)$$

As a conclusion, the relations above show that both the current and the energy that a capacitor can store for a given voltage are larger than the ones given by a standard model, which does not take into account the internal particularities of a double-layer capacitor.

P_{max}, or the maximum power of a capacitor is given by:

$$P_{max} = \frac{V^2}{4R_S} \quad (59)$$

where *ESR* is the equivalent series resistance of the capacitor.

By having a small ESR, super-capacitors can reach high power densities, although, due to material and chemistry limitations, the nominal voltage for such capacitors is low, in the order of a few volts. This aspect, and the high manufacturing costs have prevented super-capacitors to become a suitable replacement for rechargeable batteries from the mid to high-capacity range.

However, in the field of wireless sensor networks where node consumption is in the range of tens of milliamperes, such large energy densities are not necessary and the advantages of using such a capacitor far outweigh the inconveniences. Also, the problem of designing efficient charging circuits for batteries (that they themselves need power to operate) is non-existent in the case of capacitors.

In order to determine if super-capacitors are capable of replacing batteries in powering a wireless sensor node, a series of experiments were made. In all the experiments we used two electrical double layer capacitors: a 1F, 5.5V electrolytic manufactured by Panasonic which has an internal resistance of 50Ohm and a 22F, 2.2V Rubycon, with an internal resistance of 0.1Ohm. Internal resistance is one of the most important parameters that affect the rate of charge absorption for a super-capacitor and, as a rule of thumb, decreases as the nominal capacity increases. In order to have a reference for our experiments, we also used a regular 10000μF electrolytic capacitor and a low capacity NiMH rechargeable battery.

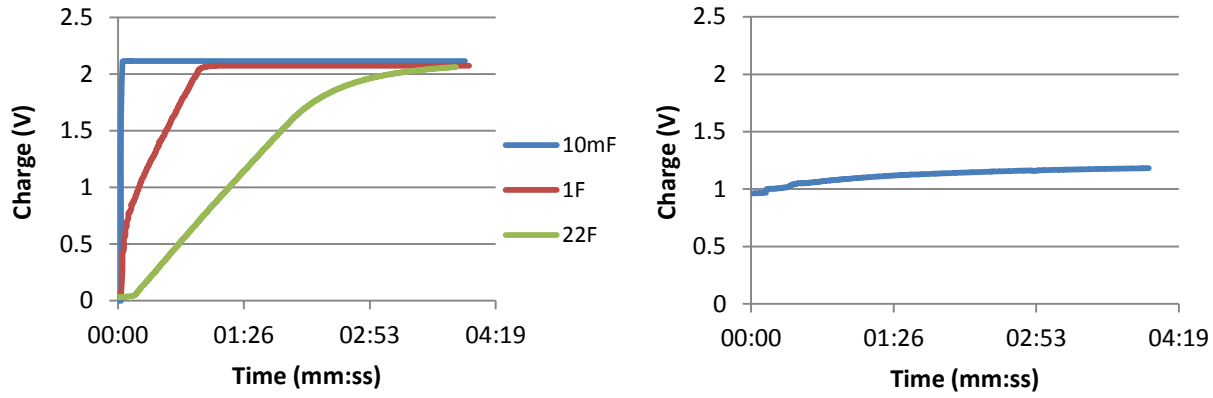


Figure 44. Capacitor charge times compared to a 300mAh battery behavior for the same charge current

The first experiment tried to determine how fast a super-capacitor can store a given amount of energy compared to a rechargeable battery and a normal electrolytic capacitor. For this, we connected each capacitor to a power supply limited in current to 50mA and measured their charge time. We applied the same procedure to a 3.2V NiMH rechargeable battery that was previously discharged to around 1V. The plot in Figure 44 shows the charge curves of each capacitor compared to that of the rechargeable battery.

By analyzing the graph, it is clear that all capacitors are capable of absorbing charge much faster than the rechargeable battery. The differences in charge times between capacitors are given mostly by their nominal capacitance. However, these differences are small, as even the 22F capacitor reaches full charge after only three minutes. This type of behavior is an advantage when employing them as an energy buffer in conjunction with energy harvesting. As established in the previous chapter, the harvesting module might provide only very short bursts of energy into the system, which cannot be efficiently absorbed by a standard battery which relies on chemical processes to store charge. Super-capacitors have the ability to store charge quickly and efficiently, making them a prime candidate for piezoelectric harvesting, for example.

The second experiment involved testing discharge of the same capacitors used for the first experiment. This was done in order to determine the rate of energy transfer the capacitors can achieve compared to that of regular batteries. For this, we connected each of the capacitors to a fixed 100Ohm resistive load and measured their discharge time. This was then compared to the discharge time of a 21mAh coin cell NiMH battery over the same load.

The results are plotted in Figure 45:

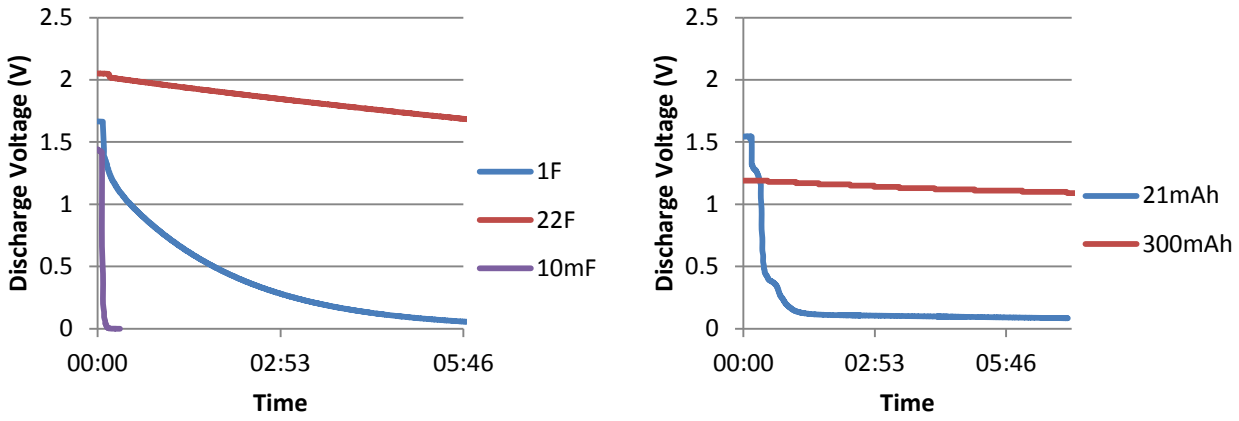


Figure 45. Capacitor and battery discharge curves for a fixed load

Results of the above plot can be justified by the difference between the energy density of capacitors and batteries. Capacitors have a high power density, meaning they can release their charge very fast, unlike batteries. Regular capacitors have very low energy density, and due to that they cannot store large amounts of charge. This sort of behavior is exhibited in the plot by the 10,000µF capacitor, which has a very steep discharge curve. However, modern super-capacitors also have a sizeable energy density which can be compared to that of batteries. This is why the discharge curve of the 1F capacitor is comparable to that of the 21mAh battery and the 22F capacitor exhibits an even slower discharge, comparable to that of the 300mAh battery.

In conclusion, super-capacitors are very good at storing and releasing large amounts of charge not only over large time intervals, as also rechargeable batteries do, but for very small time intervals as well.

Another positive aspect about the high power density of super-capacitors is their ability to release all of their charge, without any residual energy left over. This behavior is not exhibited by batteries which cannot release their charge when subjected to a high discharge rate and will normally rebound to their previous voltage if the load is disconnected from them. This sort of behavior is measured in Figure 46 where a 21mAh battery was discharged over a 100Ohm load. After the battery voltage dropped to nearly zero, the load was removed from the circuit and the battery regained its previous voltage. The explanation for this rebound is the fact that batteries store charge through an electrochemical process which is several orders of magnitude slower than the electrostatic process by which charge is stored in a capacitor.

Charge cycles, which are always a wear and tear issue in rechargeable batteries are virtually unlimited in the case of capacitors (millions of cycles as opposed only to thousands for batteries). The only age-related degradation they suffer is a decrease in capacity, which can be around 20% less after 10 years. Even so, due to their low ESR, they still have 80% of their energy available, unlike batteries which have a higher ESR and a steeper decrease in capacity with age. Ageing and extending operational lifetime can easily be achieved using capacitors with higher values than the ones needed for the node to function. Thus, even in case of diminishing capacity, the node still has more than 100% energy storage space over a longer period of time.

The same strategy can be applied with batteries only with limited results and little extension of lifetime.

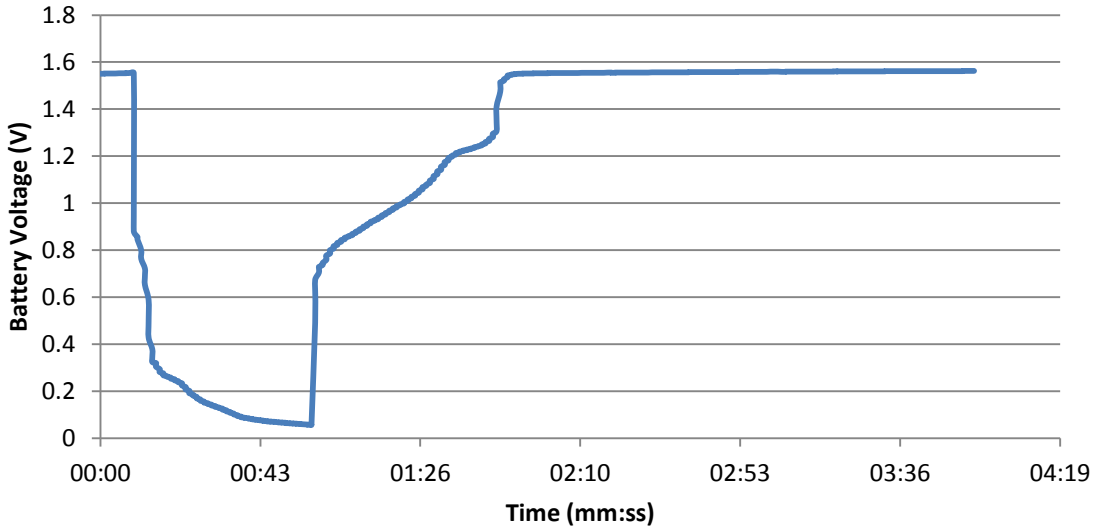


Figure 46.Battery behavior after being subjected to a sudden discharge cycle

Another important point in the case against using batteries is the environmental aspect. Most wireless sensor networks are deployed in the wild or in outdoor environments where retrieval or servicing is difficult. Batteries wear out in years and their safe disposal poses some serious issues, especially when involving a network of hundreds or thousands of nodes. On the other hand, capacitors do not employ electrolytes that can leak or that can be as damaging to the environment as the ones found in batteries.

4.2 Charging Circuit Design

4.2.1 Super-capacitor Charger

The number of cycles a rechargeable battery can endure before significantly diminishing its capacity is at most 500 for conventional NiMH. On a daily charge-recharge cycle, such as the one batteries in [79] are subjected to, this would translate in changing a node's batteries once every year and a half, or even sooner, if using cheaper alternatives.

Super-capacitors have higher energy density than batteries [93] and, unlike them, they can be subjected to a infinite number of charge-recharge cycles without any degradation whatsoever.

The proposed solution uses aerogel capacitors from Panasonic that have a very low ESR. Selecting the correct capacity for a given application is a simple matter of performing a calculation of the average current a sensor node will demand. Presuming a 20mA active current and a 5μA for a sensor node and a 1% duty cycle, the average current value is 0.01×20 mA + 0.99×5 μA = 0.205 mA.

This current needs to be supplied for the duration the energy harvesting circuit is inactive, which, in the case of a photovoltaic panel is during nighttime. Taking a maximum interval of 10 hours for the duration of the night, and a discharge in supply voltage from 3.6V to a minimum of 3.3V during this interval, we can calculate the capacity needed for storage is around 25F.

For convenience, and due to the fact that there was no suitable 25F capacitor on the market that would also have an operating voltage higher than 2.3V, we chose two 50F 2.3V capacitors that we linked in a series configuration.

The following simple charge circuit is proposed:

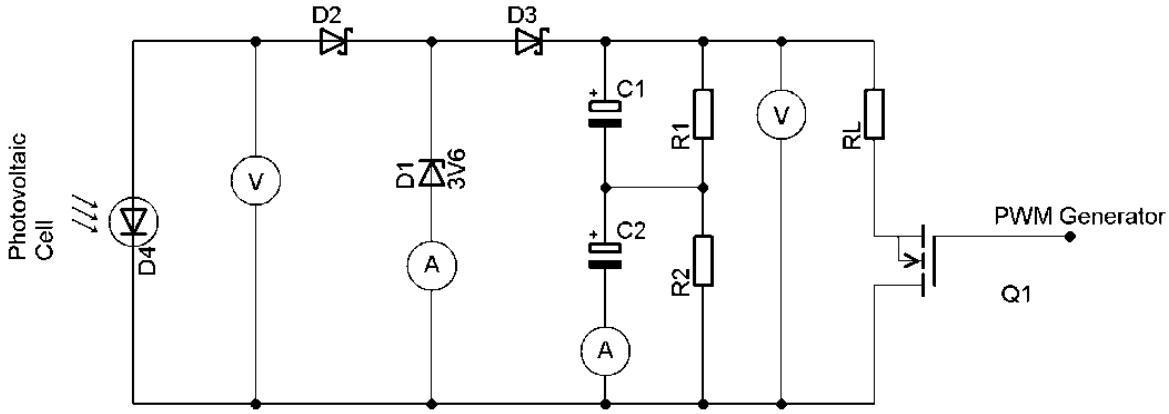


Figure 47 Charging test circuit

For the first part of the experiments, in order to simulate a sensor node we used a resistor that is connected to a FET transistor acting like an on/off switch for the load. Its gate is linked to a signal generator that is set to output a square wave of variable duty cycle and period, in order to simulate a node's own duty cycling.

On the left of the schematic, the solar panel is connected to the capacitors through a series of diodes (D2 and D3). These are Schottky diodes that act as a one way valve, directing the charge current from the panel to the capacitors and halting it not in reverse. Such a condition could happen at night, when the voltage drop on the panel is low and there is the risk of the capacitors discharging through it.

A higher than rated voltage could damage the super-capacitors irreparably, so another necessary circuit element is the Zener diode D1. It clamps voltages higher than 3.6V that may come from the panel during high light intensity intervals.

The super-capacitors have 2.2MΩ resistors connected in parallel which form a voltage divider. This is necessary in order to reduce the panel voltage to an acceptable level for each capacitor. Even though the capacitors have the same value, their rated tolerance is of ± 20%, which would lead to a charge and voltage drop imbalance.

4.2.1.1 Charging Circuit Experiments

In order to test the correct functioning of the solar panel-super-capacitor system, a series of experiments were conducted.

In the first experiment, the load was left unconnected and we tested the behavior of the charging circuit under different lighting conditions. For this we used an ordinary 100W lamp and different measurements were taken while varying the distance from the solar panel. The light source was positioned in order to induce different currents from the photovoltaic panel and charge the capacitors at different rates. In this way, we managed to measure the current to voltage characteristics in Figure 48.

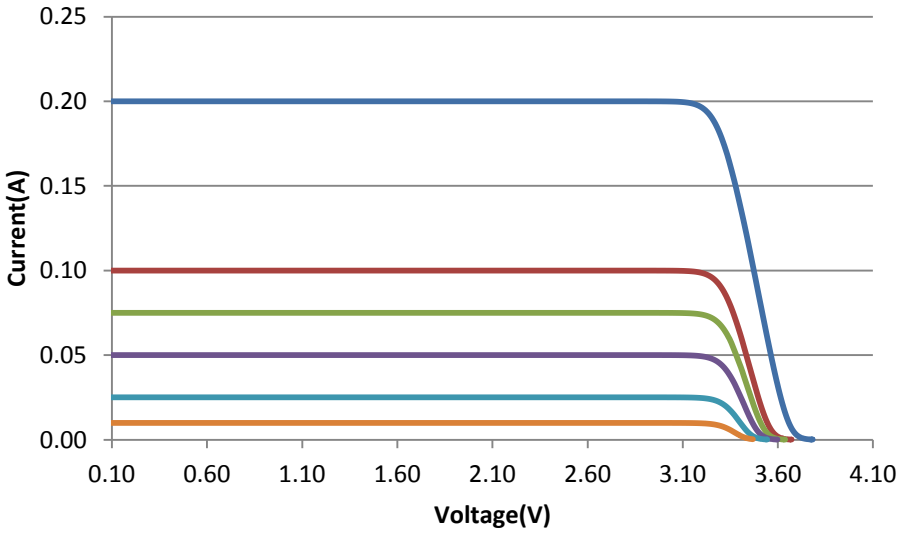


Figure 48 Panel characteristics for different illuminations

As can be observed, for each of the supplied currents, the capacitors are being charged until they reach a threshold given by the Zener diode (with a tolerance of 20%). All the possible configurations have been tested and we can conclude that the voltage on the capacitors will never exceed 4V, no matter how strong the illumination conditions the node is exposed to.

The second experiment's purpose was to determine whether a steady illumination of the panel can provide continuous functioning for the sensor node, while, at the same time, keeping the energy storage at a constant level. The light source was placed at a distance where a supply current of 30mA was given by the solar panel and the signal generator was set to varying duty cycles between 1% and 100%. The total time the load was switched on remained constant at 1 second.

Results for each pulse width setting were very similar, so we present only the most relevant one, with the node's duty cycle at 1% in Figure 49.

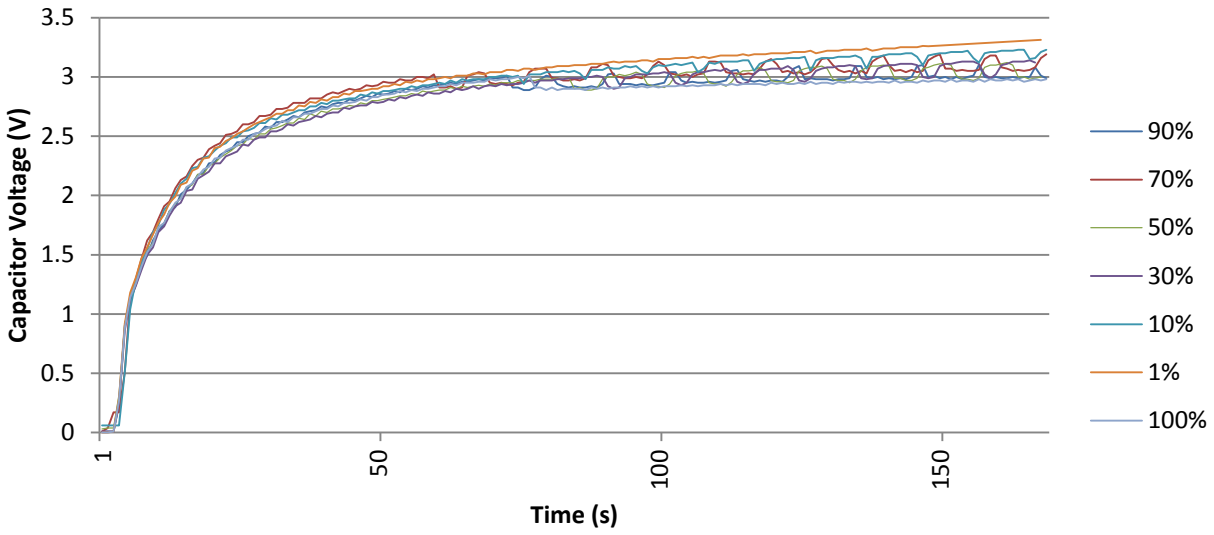


Figure 49. Super-capacitor voltage for a wireless sensor load of varying duty cycles

Starting this experiment, the load was disconnected and the capacitors left to charge until they reached 3.3V. From then on, the load was switched on and the capacitor voltage was monitored. A part of the current from the solar panel was channeled to the load, affecting the charge time of the capacitors. However, due to the fact that the load was at 1% duty cycle, this had little or no effect on the total charge time.

The final experiment was set to demonstrate the ability of the circuit to power the sensor node during periods when the illumination is low or non-existent. This would be a definite requirement for a real-life deployment of a sensor node, when it is expected for it to survive on stored energy for the duration of cloudy days or during nighttime.

For this, we designed a real-life experiment where the node would be powered by the solar panel at 1% duty cycle as before and set indoors, near our laboratory's window. We carried out the experiment for six days, during which the node had no other means of replenishing its energy supplies other than what it harvested though its solar cell.

The voltage output from the solar panel was monitored, alongside the voltage levels of the capacitors. The following plot in Figure 50 shows the results.

The plot shows the solar panel is producing energy in cycles, starting in the morning and peaking at noon. The capacitor charges following these cycles, reaching a top value of around 2.8V at maximum illumination and discharging over night. Due to the fact that the node was placed indoors, there is also the contribution of room lighting by decreasing the voltage drop rate. The plot clearly shows also periods of inactivity from the solar harvester, when incident illumination was too low to make any significant contributions.

As a general observation, the capacitor voltage never drops below the 2V threshold, which guarantees that the sensor node was always well inside its normal operating voltage domain (3.3V – 1.8V). This, and the fact that after six days of trial the node continued to

function, proves that out system can indeed power a sensor node for an unlimited amount of time, given certain conditions and that we have accomplished our goal of energy independence.

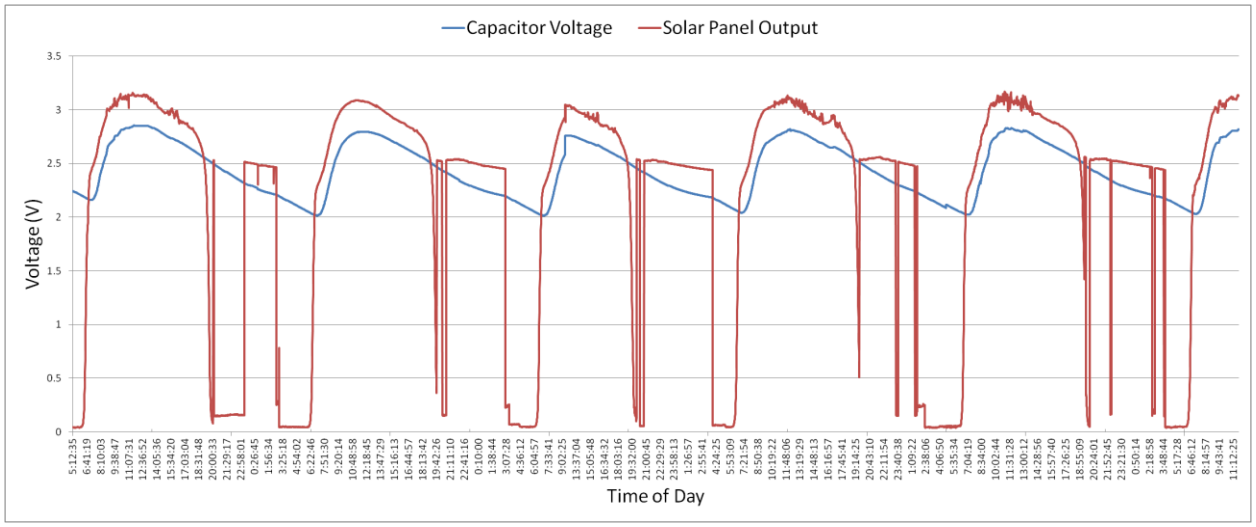


Figure 50. Solar and capacitor voltages for a energy harvesting sensor node during a period of six days

4.2.2 DC-DC Converters Charger with Power Management

The main challenge in the design of the Sparrow nodes was the power management and energy harvesting circuit. Its main function is to collect energy from an attached photovoltaic cell of thermoelectric generator and continuously charge a super-capacitor. When the charge level on the capacitor exceeds a predetermined threshold, the WSN node is powered.

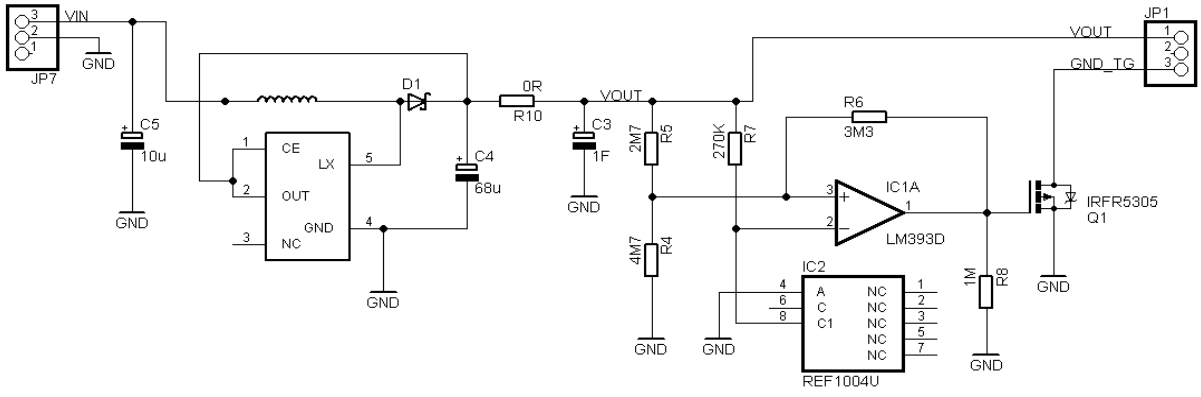


Figure 51. Schematic of the energy harvesting and power management unit

The circuit is comprised of two separate sub-systems: the step-up switching regulator and the voltage monitoring and control circuit.

Energy harvested by the photovoltaic cell or by the TEG can vary and is in direct relation to the environment condition. For example, the voltage output of the PV panel can be anything between 0V and 2V, depending on the amount of solar radiation the panel receives during a day. The same applies to the thermoelectric generator.

As the WSN node electronics operate at minimum 1.8V, the voltage generated by energy harvesting is not high enough. Therefore, a switching converter was used to step-up the gathered voltage to a fixed value. This was accomplished using the NCP1402 high-efficiency micropower regulator from On Semiconductor. This device is designed to start at a minimum input voltage of 0.8V and reliably operate down to 0.3V while keeping a fixed output voltage value of 3.3V.

The output voltage from the switcher is used to slowly charge a 1F double layer super-capacitor which acts as a power supply for the rest of the node's electronics.

The monitoring circuit's role is to power on and off the WSN node, depending on the amount of charge stored in the capacitor. It achieves this function by continuously monitoring the capacitor voltage.

Because of its continuous functioning, the circuit must be very low power and operate over a wide supply voltage range. The circuit in Figure 51 uses the MCP6542 comparator, which has a typical operating current of 600nA, for the monitor function. The MCP6542 is used with the threshold and hysteresis setting resistors R4, R5, and R6, and the LM385 voltage reference to control a FET switch, Q1, to turn on power to the MCU circuitry. The FET is on when the voltage on C3 is greater than 3V and off when the voltage on C3 is less than 2V.

Calculating the capacitor size for energy storage requires an estimate of the current flow in the circuitry, what is the voltage change on the capacitor and how much time is required complete a task. For example, when the node is active, the Sparrow circuitry requires about 23mA for continuous operation.

The time the circuitry can operate continuously for a change in C3 capacitor's voltage from 3V to 2V is calculated from:

$$t = \frac{C\Delta V}{I} = \frac{1F \cdot 1V}{0.023A} \approx 44\ seconds \qquad (60)$$

This short functioning time can be extended in two ways: by increasing the value of the capacitor and by rigorous software power management on the node.

For example, a 100F super-capacitor can easily power the node in continuous mode for 100 times the amount of time calculated above, which means that the node can function for about 73 minutes without any sunlight.

The second approach is even more effective and relies on the observation that in most WSN application scenarios a node does not need to stay on 100% of the time. This behavior is called beaconing and implies that the node stays in sleep state for a period of time and wakes up on regular intervals to measure sensor data and relay it to the sink node.

By using this behavior, large amounts of energy that would have otherwise been wasted on powering the node, can now be stored for future use in the capacitor, thus increasing the system's lifetime.

For example let us presume that a Sparrow node uses a beaconing scheme and wakes up every 30 minutes for about 1 second. Taking into consideration that the node has 6uA of current consumption in sleep mode and 23mA during wake-up, the total functioning time of the node can be increased from 44 seconds to 2.2 hours which is an improvement of almost 200 times.

By combining the two approaches, of increasing the capacitor size and beaconing, the Sparrow node can function for an indefinite amount of time on the energy gathered from its surrounding environment.

The ZigBit A2 module from Meshnetics was chosen as the main component of the design. It is a system on a chip (SoC) that includes an 8-bit low power ATmega1281 microcontroller, an Atmel AT86RF230 802.15.4 radio transceiver and all the auxiliary drive electronics, including a dual radio antenna.

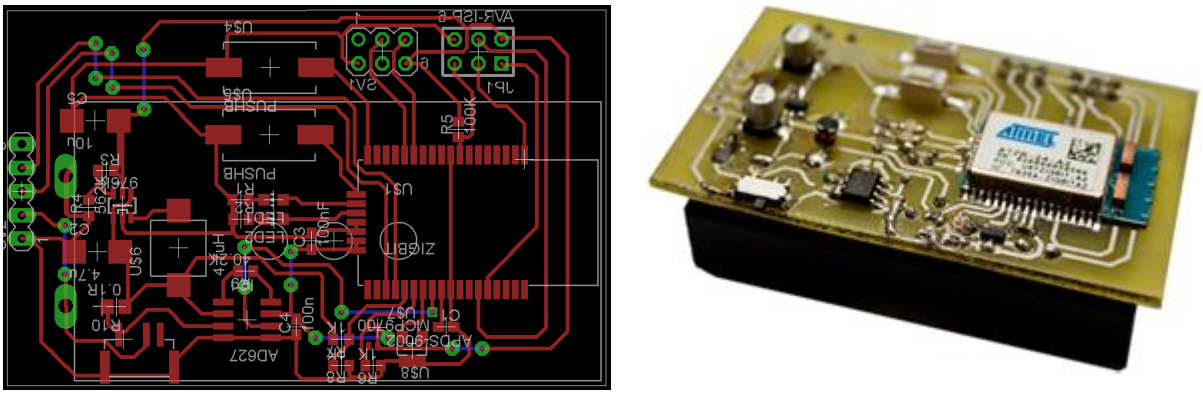


Figure 52. The Sparrow WSN Node

4.3 Deployment of Sensor Nodes in a Real-Life Environment

The first large scale deployment application for our wireless sensor nodes was in the Frauenkirche [94] in Dresden, Germany. Built in the 18th century, the church was destroyed during World War II and remained in ruins for more than fifty years before the city of Dresden finalized its reconstruction in 2005. Given its large scale, which sets Frauenkirche as an important landmark of Dresden, the church administration wanted to constantly monitor environmental conditions throughout the edifice, namely air temperature and humidity. These parameters were needed for determining whether the conditions were right for preserving the internal stone masonry and murals and serve as an early warning for potentially dangerous situations such as water seepage through the new stone walls or interior condensation.

As one can imagine, installing wired sensors in such a location would have been impossible without damaging the existing structure or having a negative impact on the location's aesthetic. Therefore, a wireless sensor network deployment was preferred because it would leave a very small footprint on the overall appearance and also have a low installation. The hardware requirements for the sensor nodes were given based on a set of constraints:

- *small form factor*: the nodes need to be installed in a historical venue and should not interfere with the general aesthetics of the location.
- *ultra-low power consumption*: due to their location in hard to reach zones of the edifice, operation should last at least one year without the need for servicing or battery replacement.
- *sensing capabilities*: humidity and temperature must be measured with an accuracy of 0.1 deg. C. Using low-power digital sensors that are already calibrated is preferable.
- *power metering and management system*: the nodes should constantly measure the voltage and current drawn from the battery pack and estimate the total energy consumption. Nodes should have the capability to turn off some of their subsystems (sensing, radio, microcontroller peripherals) and turn them on only when necessary.
- *built-in energy harvesting*: nodes should take advantage of the energy available to them from the surrounding environment. Usage of miniature photovoltaic panels and/or Seebeck effect thermal harvesters can partially or completely satisfy the energy budget of a node.

Requirements were also specified for the software protocols and application the nodes needed to run

- *multi-hop network*: Nodes should form a multi-hop wireless network that sends measured data and receives commands to/from a central sink
- *software energy management*: nodes employ a smart energy management algorithm that maximizes the sleep times and minimizes power consumption
- *over-the-air reprogramming*: the software stack has the capability to reprogram itself using the new firmware received from the coordinator
- *location tracking*: nodes should be able to run a protocol that enables them to determine their relative location to all the other nodes in the network.

This task enabled us to test our research platform in a real–life environment for the first time. We needed to determine whether the energy harvesting techniques we developed and tested in the laboratory would also work in a less than perfect environment that is prone to radio interference and imposes heavy availability constraints on the nodes.

In the tests we conducted on-site, we determined that radio transmission on the 2.4GHz band was less than adequate for the given venue. Due to meter-thick stone walls, interior variable geometry and an existing surveillance infrastructure that was broadcasting in the same frequency range, the nodes could not associate properly from the given distance.

Following the poor results we had with the 2.4GHz band, we decided to test whether the same problems applied to the 868MHz band. Test results were encouraging, as the frequency spectrum was less crowded and the lower frequency guaranteed a better wall penetration.

The only problem was to modify the Sparrow node to work in the new frequency band. This is where the modular aspect of our node came in handy, because we just replaced the 2.4GHz Zigbit module with an 868MHz one, with no other major modifications to the existing circuits.

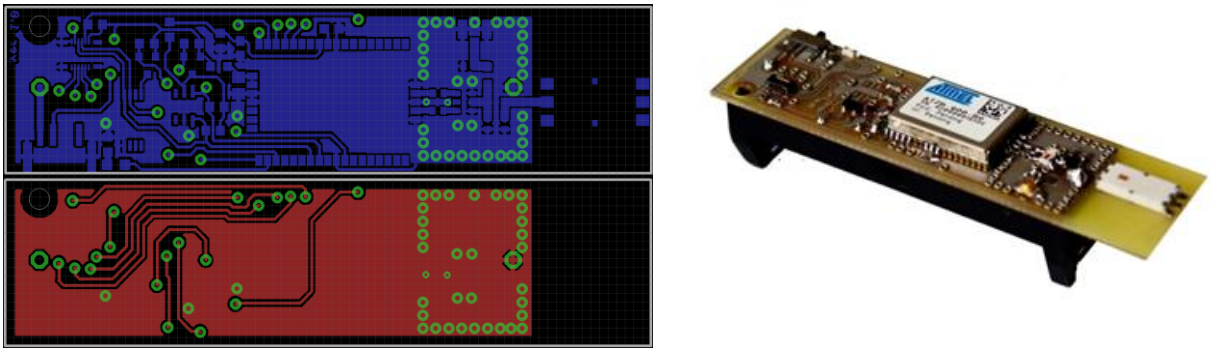


Figure 53.The sensor node developed for the Frauenkirche application

The second modification we applied to our system was to add low power digital sensors for humidity and temperature measurement. We were lucky to find both integrated in the same package, the Sensirion SHT21 [95]. Its low profile, low power specifications made it an ideal choice for our application needs.

The monitoring infrastructure was supplied by Zigpos [96], which was this project's industry partner.

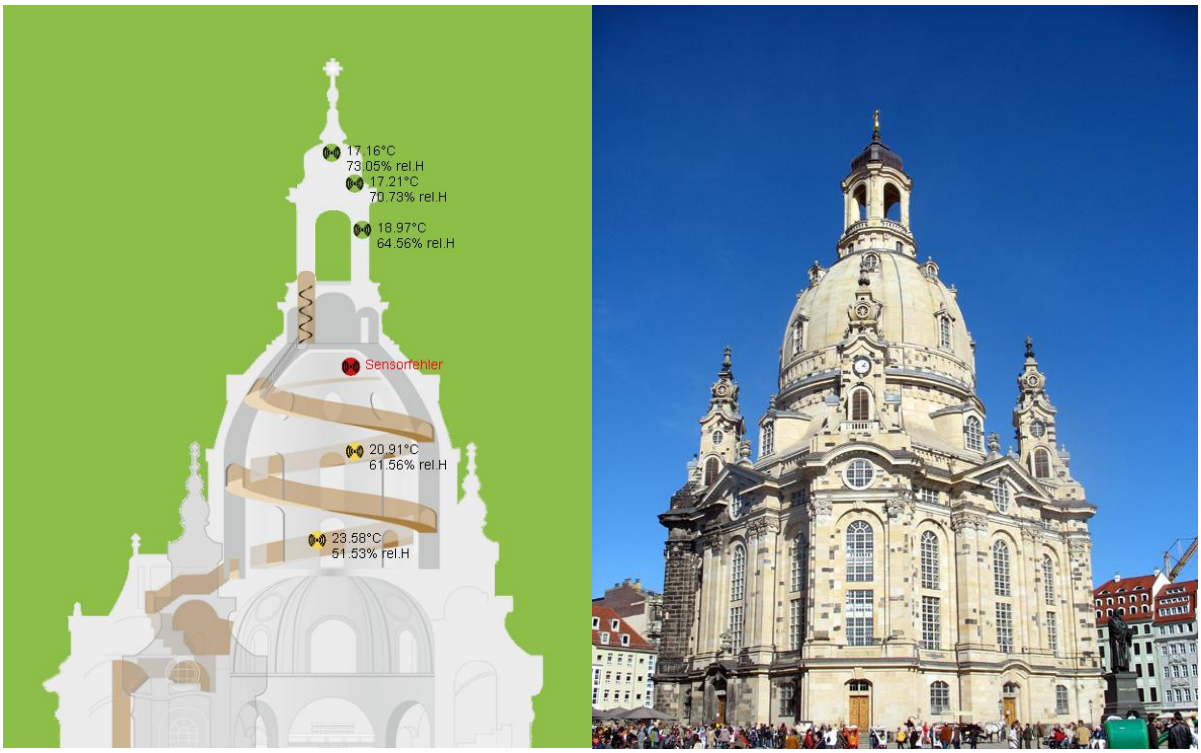


Figure 54. The monitoring interface showing node placement and measured paramater values

Chapter 5

A Novel Management Protocol Suite for WSNs

The Check Management Protocol Suite offers a new approach into monitoring and actuation through a framework for heterogeneous WSAN islands. It crosses the borders of different organizations and network setups. The monitoring of all the components of Environmental Sensor Networks (ESN), Community Sensor Networks (CSN), and Body Sensor Networks (BSN), such as load, link quality, processor and radio usage on the nodes, as well as enabling actuation in these networks, is not yet fully solved in a fully heterogeneous environment. The Check Suite addresses this point by implementing multi-hop task-scheduling algorithms, based on multi-hop routing schemes for homogeneous wireless sensor networks.

In WSANs, packets are typically forwarded in first-come first-served order. However, this scheduling does not work well in real-time networks where packets have different end-to-end deadlines and distance constraints. The multi-hop task-scheduling scheme from the Check framework needs topology information from the network, which will be gathered using network discovery algorithms.

Task-scheduling is a fundamental requirement for various subsystems of WSAN islands. It can be used, for example, in a smart places scenario where a person is automatically assisted by the system to reschedule the appointments at a post office or a shopping mall, or for a worker in a plant to enable the parallel execution of tasks and to prevent workers from performing conflicting operations. Present research on scheduling is generally focused on hard time deadlines. Instead, the Scheduling component of the Check Suite proposes a solution where energy consumption, battery awareness, availability and affinity are considered as more

important than execution time. The tasks being scheduled are the smallest indivisible parts of an application. Network Control is provided by the Check scheduling service, which also schedules tasks to given nodes. Possible commands sent to sensors are starting a task, ending a task, or subscribing to a task's output.

Check also offers a centralized monitoring, control and reconfiguration framework, which works toward the realization of the scalable internetworking, horizontalization and heterogeneity design goals. A big challenge is to enable push and pull algorithms to get monitoring data and issue commands to and from the monitoring nodes transparently, efficiently and reliably.

The last component provided by the Check Management Protocol Suite is the availability of a high-level, service-oriented self-healing strategy. Here the WSAN is regarded as a service provider. Check thus offers a high-level framework for assuring service availability in WSANs. Whenever a component of a WSAN island fails, it is of paramount importance that the functionality it provides is not lost, to ensure the availability and reliability of the services being offered. Check Self-Healing is a component providing the recovery strategies employed when these events occur. A service model is used for data input, output and processing in between various WSAN nodes. The service model includes data sources and sinks which are connected to form a service graph. Services can be allocated to nodes by the Check-Scheduling component of the Check Suite. The self-healing component identifies failing or poorly performing services and signals and orders the scheduler to reallocate them or reallocates them itself and configures the nodes directly. The self-healing algorithms can manage multiple WSAN islands through the Check – Monitoring and Reconfiguration management component, and can thus move services in heterogeneous WSAN islands if data dependencies allow it. It must also be noted that this coupling of the Self-Healing and Monitoring components, by being able to gather information from a wide variety of devices, using many operating systems and offering numerous services, will provide a hardware- and operating system-independent mechanism for service-level self-healing in a system.

Further, we present the features implemented in the Check Management Protocol Suite:

- **Horizontalisation:** The monitoring and reprogramming component of the Protocol Suite supports horizontalisation by allowing applications to access and modify WSAN data and behavior of resources from various islands through get and set interfaces.
- **Manageability:** Through the reconfiguration and task-scheduling components, the Check Protocol Suite allows the flexible management of WSAN nodes. Through reconfiguration, simple commands like start, stop, or restart, are sent to the nodes, while task-scheduling permits the optimal distribution of user tasks into the available nodes from various WSAN islands.
- **Continuity:** The self-healing component of Check insures that services are continually available in WSANs that are managed by the Protocol Suite. On node or service failure, the self-healing component reallocates the corresponding services to other healthy nodes from the same or other WSAN islands. Accordingly, tasks that were scheduled to

run on the failed resources are re-scheduled to the new nodes using the task-scheduling component of the Check Protocol Suite.

- **Evolvability:** Using the reprogramming component of the Check Protocol Suite it is possible to change the code that is running on a specific WSAN gateway, or to change various network parameters in the controlled islands.
- **Scalability & Locality:** The scalability of the monitoring and reconfiguration components of Check has been proven with data from numerous WSAN islands across Europe which is monitored in real-time in a single online application. Check also allows for on-demand reprogramming of gateways or nodes on all WSAN islands.
- **Heterogeneity:** The monitoring, reprogramming and self-healing components of the Check Protocol Suite are deployed on different WSAN islands and controlling various sensor types, more specifically on RZRaven, Sparrow and Sensinode platforms.

5.1 Framework Overview

As can be observed in Figure 55 the Check Management Protocol Suite consists of the following entities:

- **WSAN Island**: inside the island sensor nodes organize into a multi-hop wireless network with tree topology in which some nodes act as routers for the end node leaves.
- **Edge Router**: this is an embedded network coordinator which acts as a sink to the sensor node data and routes the island network traffic to IPv6 networks.
- **Gateway**: Binary Web Services (BinaryWS) that are running on a machine in the same network with the Edge Router.
- **Check Services**: the Management Protocol Suite offers Self-healing, Task Scheduling, Monitoring and Reprogramming services. Each of these services is operating at various levels in the WSAN Islands, Gateway or Edge Router.

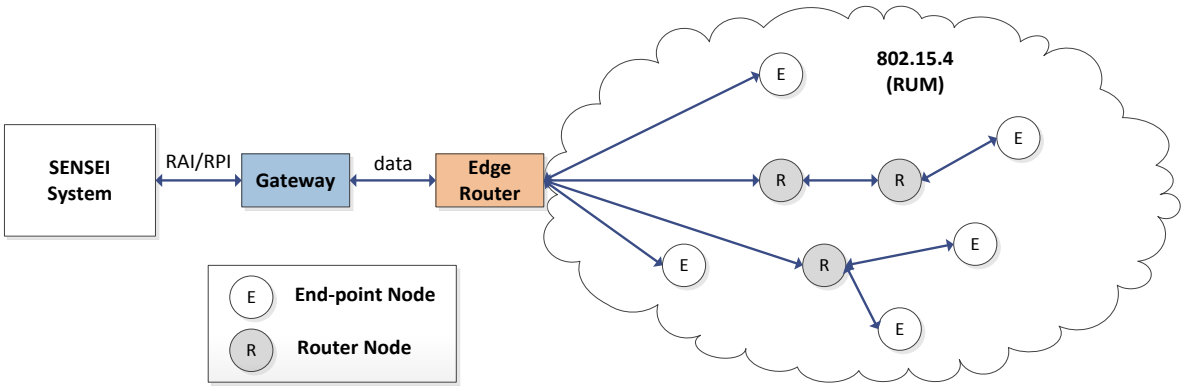


Figure 55. Check Management Protocol Suite components.

The integration of the Check Management Protocol Suite in the SENSEI [97] system is given in Figure 56. The sensor network itself is built around RZRAVEN and Sparrow nodes that form a heterogeneous multi-hop network using a modified version of Atmel's RUM protocol

over which we ported the Binary Web Services application and our own task scheduling, self-healing, reprogramming and monitoring applications. The Gateway service acts as a mediator between the network and the Check Management Protocol Suite.

The edge router is an embedded ARM7 board running a modified version of the µTasker OS. An IPv6 layer was added over the existing OS to allow the SAM7X platform to act as an IPv6 Edge Router in addition to an 802.15.4 PAN Coordinator. The PAN Coordinator performs the classical functions defined in section 5.3 of the IEEE 802.15.4-2006 specification. It will start and maintain a non-beaconing network. The edge router functionality will route IPv6 network traffic to the appropriate end and router nodes based on their specific IPv6 addresses. SAM7X provides multiple interfaces for users to interact with the 802.15.4 wireless network. Among these are RS232, USB, telnet and simple direct web interface.

As platforms for the development of the Check Management Protocol Suite [98], we are using Raven sensor boards with the Contiki Operating System, IETF 6LowPAN compliant Sensinode boards as well as simulated nodes. The graphical user interface of the Management Suite depicts the current status of the testbed. The management application is a J2EE (Java 2 Platform, Enterprise Edition) Java Web App that acts as a client. It can query all services, display the status (e.g. test or query attributes) and where possible issue commands to all units developed by each SENSEI component, such as restarting or reconfiguration of node services by setting node attributes. For example, the monitoring and management component can communicate with a Google maps enabled application that shows real-time sensors and sensor data.

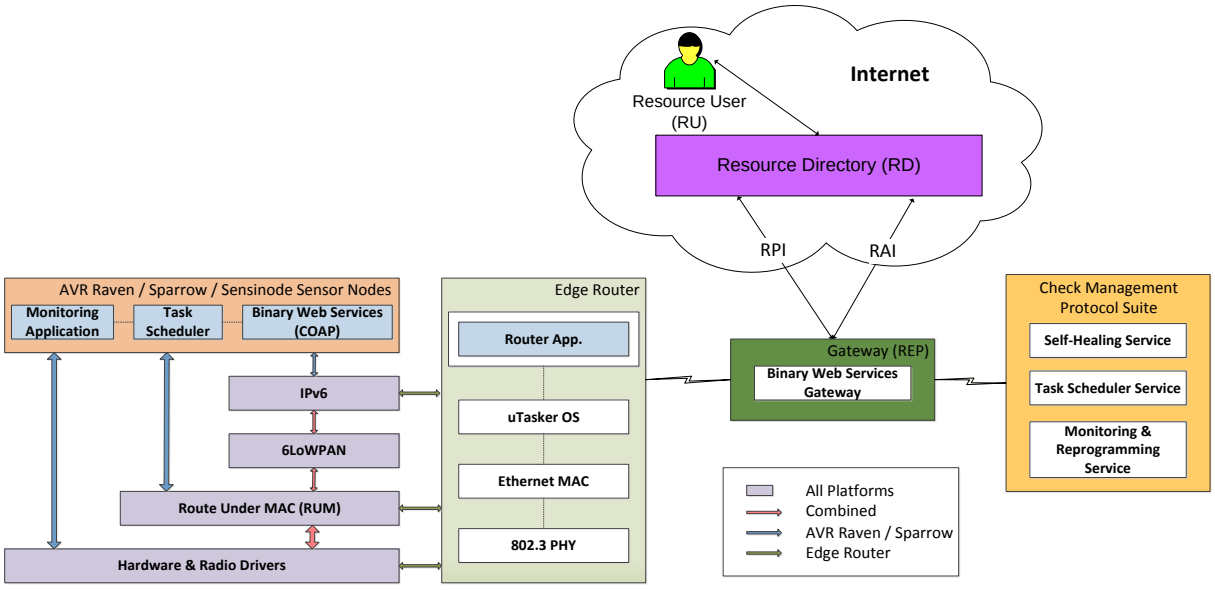


Figure 56 Integration of the Check Management Protocol Suite into the SENSEI system.

5.2 Considerations on Self Healing Networks

Wireless sensor and actuator islands play a major role in future Internet scenarios. In fact, they are exploited in a great number of different situations, ranging from the industrial application space to military and medical scenarios.

One of the thesis objectives is to detail how WS&ANs can be integrated into the future internet. Since these networks are, generally, optimized for very specific scenarios, they need an abstraction framework to enable their functionalities to be exploited in a common system. We envision four main areas to be taken into consideration by this framework:

- *Information management*: to retrieve both the information collected by the network and its own status (configuration parameters, topology, operational state). It has also to be possible to issue on demand data gathering procedure.
- *Actuation triggering*: the framework has to be able of conveying instructions and to issue commands to any node in the network or to any set of nodes.
- *Dynamic configuration*: sensible application parameters have to be directly managed (i.e., through set, get operations) by the framework. Moreover, it has to be possible to reprogram the application running on the nodes in part or in its entirety.
- *Resource management*: information about the status of any given node is to be collected and stored within the framework, in order to monitor network conditions and to control network access for different users.

An autonomic system is composed of interrelated autonomic elements. Each of these elements has managed hardware or software resources that build the IT infrastructure and autonomic managers that supervise and control these resources. Self-healing components can detect system malfunctions or failures and start corrective actions based on defined policies to recover the network or a node. The automatic recovery from damages improves the service availability.

As described in [99], [100], [101], [102] there are different healing strategies based on locality awareness of the sensor node and/or energy efficient algorithms.

- **Graph heal**: On each deletion, the neighbors of the deleted node in a binary tree are reconnected regardless of whether any cycles in the graph formed by the new edges introduced for healing. The downside of this algorithm is that the nodes use more edges than what is required for maintaining connectivity.
- **Binary tree heal**: [103] On each deletion, the neighbors of the deleted node in a binary tree being are reconnected careful not to introduce any cycles in the graph formed by the new edges introduced for healing. This is done using random IDs which can then be used to identify which tree a particular node belongs to. This is an improvement on the previous algorithm but still naive since it does not take into consideration the previous degree increase suffered by nodes during healing.
- **Automatic Fault Recognition**: [104] Rather than having a system response based on node deletion events, another self-healing approach is trying to detect fault patterns and

have so-called B-scripts react and repair the fault. This strategy defines two new node types: lymph and thymus that detect anomalies and malicious behavior and submit corrective commands to the system.

The basic assumption of all current algorithms is that the network is initially a connected graph over n nodes. An adversary repeatedly attacks the network. This adversary knows the network topology and the algorithms, and it has the ability to delete arbitrary nodes from the network. However, we assume the adversary is constrained in that in any time step it can only delete a single node from the network. Another assumption is that after the adversary deletes some node x from the network, that the neighbors of x become aware of this deletion and that the network has a small amount of time to react by adding and deleting some edges. Several algorithms implement these strategies, the most efficient being:

The Forgiving Tree [104] is based on a rooted spanning tree T layout, which without loss of generality may as well be the entire network. Each time a non-leaf node v is deleted, it is replaced by a balanced binary tree of „virtual nodes" with the leaves of the virtual tree taking v's place as the parents of v's children.

When a leaf node is deleted, it is not replaced. However, if the parent of the deleted leaf node was a virtual node, its degree has now reduced from 3 to 2, at which point it is considered redundant and „short-circuited", removing it from the graph, and connecting its surviving child directly to its parent. This helps to ensure that, except for heirs, every virtual node is of degree exactly 3. The replacement of each deleted node by its virtual tree can be done in O(1) time. The total size and number of messages which must be sent is O(1) per deleted vertex. In addition, there is a startup cost for communicating the initial "wills"; this is O(1) per edge in the original network.

DASH [103] improves on [104] upon several aspects, one of witch being that no node ever increases its degree by more than 2 log n, where n is the number of nodes initially in the network. DASH adds new edges only among neighbors of deleted nodes; and has average latency and bandwidth costs that are at most logarithmic in n. DASH has these properties irrespective of the topology of the initial network, and is thus orthogonal and complementary to traditional topology-based approaches to defending against attack.

When a node x is deleted, the neighbors of x react to this deletion by adding some set of edges amongst themselves. These edges can only be between nodes which were previously neighbors of x. This is to ensure that, as much as possible, edges are added which respect locality information in the underlying network.

DASH is a fully distributed algorithm and it also has several "flavors" like **SDASH** (Surrogate degree based binary tree heal), a heuristic algorithm that tries to keep both node degrees and path lengths small.

SASHA [105] is a self-healing hybrid sensor network architecture that is inspired by and co-opts several mechanisms from the acquired Natural Immune System to attain its autonomy, and adaptability to unknown faults. SASHA encompasses automatic fault recognition and response over a wide range of possible faults and tries to define an adaptive architecture that can

learn and evolve its monitoring and inference capabilities over time to deal with unknown faults. Two new node types are defined: the *lymph*, which tries to detect network anomalies, and the *thymus*. In a WSN the survey of a forest can be undertaken by means of mobile scripts running on all monitors, called B-script. A script is dynamically generated code and it acts as a filter for the behavior and statistical analysis of a forest. The second node type is the *thymus* that tries to store a representation of the current system status known as self and confirm the presence of faults.

This approach is different than the rest of the standard algorithms and more complex. SASHA proposes algorithms for automatic fault recognition, adaptive network monitoring and coordinated response. Although there is no working implementation of the algorithm, this self-healing strategy could base itself on other services already described in this document like code migration, sensor code update and in-network computation and could also provide better overall system response.

In WSNs the scarcest resource is energy, and one of the most energy-expensive operations is data transmission. For this reason, algorithmic research in WSN mostly focuses on the study and design of energy aware algorithms for data transmission from the sensor nodes to the base stations. Data transmission is usually multi-hop (from node to node, towards the base stations), due to the polynomial growth in the energy-cost of radio transmission with respect to the transmission distance.

The algorithmic approach to WSN differentiates itself from the protocol approach by the fact that the mathematical models used are more abstract, more general, but sometimes less realistic than the models used for protocol design.

In the Check Service Self-Healing [106] component, the Self-Healing Agent was tested on the RZRaven, Sparrow, and Sensinode NanoSensor-based WSAN islands. The Self-Healing component was designed to be hardware-agnostic, protocol-independent, and to coexist with other management components [107]. Thus, an abstraction mechanism was designed to translate WS&AN-specific data into a unified, well-defined format. Similar abstraction layers can be used to receive data from other SENSEI components such as the Titan Framework. The Self-Healing Agent was designed so that it can be run in multiple instances and can accommodate various self-healing strategies based on the service data. An abstraction mechanism was designed to translate decisions issued by the Self-Healing Agents into WS&AN-specific commands that have the effect of starting, stopping or configuring a service.

5.3 Scheduling Algorithms for Task-based WS&ANs

5.3.1 Description

Task-scheduling is a fundamental requirement for the middleware subsystem in WS&ANs. It can be used, for example, in the smart places scenario to reschedule the

appointments at the post office or the shopping mall, and in the worker in a plant to enable the parallel execution of tasks and to prevent workers from performing conflicting operations.

The Check framework implements a multi-hop task-scheduling algorithm, based on multi-hop routing scheme for homogenous wireless sensor networks. The multi-hop scheme needs topology information from the network, which is gathered using network discovery algorithms that are implemented in the framework, as well as possible solutions from other middleware components (e.g. Titan [108]).

Network control is provided by the Check scheduling service, which schedules tasks to given nodes through a dedicated interface. Possible commands sent to sensors are starting a task, ending a task, subscribing to a task's output (which means that the task will send its output to multiple data sinks).

The Check scheduling service also responds to the self-healing service and reallocate tasks as needed together with other management components. The service bases its reallocation on performance metrics and capability specifications obtained from the network.

5.3.2 Task Allocation and Management

As a platform for development of the task-scheduling algorithms we used Raven sensor boards and the Contiki Operating System [109]. Although the scheduling service is meant to be platform independent, the implementation of the middleware system local to a sensor board is deeply architecture specific. This means that the protocol can be defined and standardized for generic scenarios, while its specific implementation will depend on the hardware platform. The Contiki OS uses a cooperative model for processes. Each process that runs on the processor at any given time must at some point relinquish its position, or the system would come to a halt. Contiki's cooperative process subsystem is based on event queues, where events are associated with processes and "wake" them up. In reality, when one process yields it's control of the processor, the "scheduler" looks for a new process to run, so an event is taken out of the queue and the process associated with it takes hold of the processor.

There are two possibilities regarding the implementation of generic tasks, one is to have them all run under a single process, using timer events with a scheduler similar to real-time operating systems. A timer would be set to expire when the first task is due. The downside of this method is a difficulty in coming up with a metric for processor use – versus idling. The other possibility is to treat a task as a process. The task manager starts and stops tasks, alters settings, append subscribers, etc. A performance metric can thus be calculated using this method concerning the scheduler. As the system keeps more and more tasks running and using even more CPU time, the time slices between schedules will consequently increase.

Starting the task is similar to starting a process in Contiki, while stopping a task means marking it as a "waiting" process, as can be seen in Figure 57. While being marked as "waiting", the task is in its (presumed) main infinite loop and it waits until it is taken out of this state by the scheduler.

Obtaining information from the task can be done in one of two ways, namely: collecting data directly, with get/set <parameter>, or using the data sink method, where the entity that connects to the sensor can register itself or another entity as a data sink for the output that is generated from the task. Parameters can be given, such as the frequency, with which data is forwarded to the subscriber. A list of processes can be kept with the linked list API available in Contiki OS.

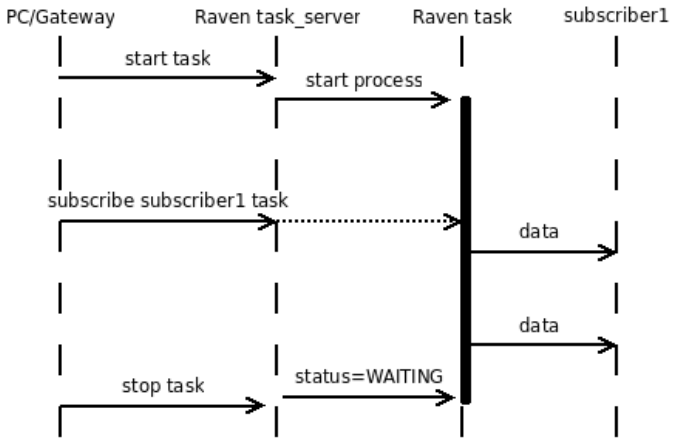
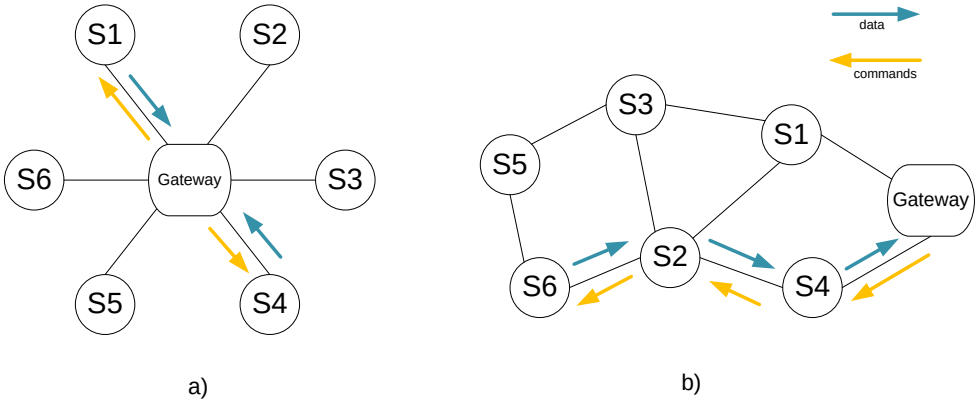


Figure 57. Task-scheduling in Contiki OS

The scheduling service thus has three ways of controlling the running of tasks in the WS&AN island: by starting/stopping a task on a sensor; choosing data sinks for output data of a task; and by adjusting the frequency with which data is outputted. The scheduler then takes decisions taking into account the state of the network, the priority and complexity of the task to be executed.



a) Task scheduling in a network with star topology.
b) Task scheduling in mesh networks.

Figure 58 Task-scheduling in star and mesh networks

Task scheduling in a star network topology involves the gateway sending commands directly to the sensor nodes, while in mesh networks the current implementation of task-scheduling uses the gateway to schedule tasks on remote nodes by sending commands via a multi-hop routing scheme, as shown in Figure 58.

The state of the network, other than the topology of the network – which is not discussed here, consists of the load on each sensor node. This can be estimated by measuring the time between two schedules of a given task, and taking that value, or a weighed sum over a number of iterations. This data is obtained by the scheduler either by subscribing to this task or by querying it directly from the task server interfaces on each sensor [110]. The energy left in the sensor must also be taken into account when scheduling tasks on a node. The energy estimation module (energest) provided by the Contiki platform, together with ADC data from the power supply are used to determine whether the given node is able to finish the task or not. Since providing continuous service for the task is paramount, it must first be determined if the task is not repeatable. If this is the case the node with the highest available energy is chosen when scheduling the task.

Currently there are two types of tasks being considered, namely: tasks that only have output (e.g. periodic tasks, that just calculate different metrics or send sensor data to data sinks), and tasks that process data. The latter can aggregate sensor data from several sensors (they are subscribed by the scheduler to the other sensors' data) and, for example, transmit a mean value or detect sudden change in the data. The scheduler then picks the nodes or sensors that can handle the extra energy loss due to the extra communication required to execute this type of tasks and schedules the tasks on these new resources.

5.3.3 A Novel Task Scheduling Algorithm for WS&AN

The problem we address is an unconventional scheduling algorithm, in the sense that the main constraint is not time, but energy [111]. Present research on scheduling is generally focused on hard time deadlines. Instead, we propose a solution where time is of least importance, preceded by energy consumption, battery awareness, availability and affinity.

The task that we wish to schedule is the smallest indivisible part of an application. Tasks can be classified into sensing tasks, actuating tasks, detection tasks, etc. For example, we have a fire detection system implemented with a WSN. We can have smoke sensing tasks on nodes that have smoke sensors, an event detection task, which detects in a stream of sensor input when smoke levels have risen, and an alarm task, which handles the behavior of the network in the case of fire (bell, speaker, opening doors, etc.).

For the previous scenario, the scheduler needs some basic information for each task: its importance, an affinity to a certain type of node (a smoke sensing task can only be assigned to wireless nodes that have a smoke sensor), a frequency with which to run (if the task is repeatable) and dependencies (both data sinks and data sources). The scheduler will have to choose which assignment is best for energy consumption, to put intermediary tasks on sensing

nodes or to put them on the alarm nodes.

In this subsection we will try to formalize the problem defined. First, we will consider the total energy remaining in a node W_{m_k}, m_k being the node. This energy can be deduced from the battery voltage and the discharge profile of the type of battery used. We will use this energy, together with the estimated consumption per second, to deduce the number of seconds that the node can run. The minimum of these times, calculated over each node in the network, will be the network lifetime. The purpose of the scheduling algorithm is to maximize this value.

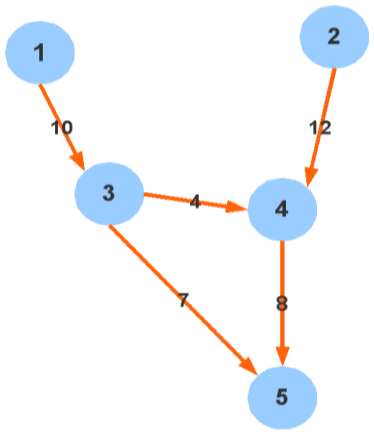


Figure 59: A Directed Acyclic Graph with edges proportional in weight to transmission energy cost

Because the platforms that we use never enter sleep mode (and their consumption is always very low compared to the transmission/reception power consumption), we will consider the energy they use incorporated into P_{idle} which can be specific to each node $P_{idle_{m_k}}$ This value represents the energy consumed by the sensor node during idle mode in one second.

We consider the energy wasted in transmitting/receiving directly proportional to the number of bits in the payload. To model the tasks and their dependencies we use a Directed Acyclic Graph (DAG), in which edges represent data dependencies, their cost being t he maximal number of bits transmitted between the tasks (We consider that a transmission occurs after each period).

Let:

- $\boldsymbol{T(m_k)}$ be the set of tasks allocated to node $\boldsymbol{m_k}$
- $\boldsymbol{P_{idle_{m_k}}}$ the idle energy consumed by a node $\boldsymbol{m_k}$ in one second
- $\boldsymbol{B(e_{ij}), e_{ij} \in E}$ (E the set of edges in the task DAG) is the average number of bits per second transmitted by task i to task j
- $\boldsymbol{P_{tr,m_k}}$, $\boldsymbol{W_{rcv\,b,m_k}}$ the energy cost of transmitting/receiving a payload bit on/from node $\boldsymbol{m_k}$
- ***M(v)*** is the node to which the task *v* was assigned.

The power used by a task while receiving data would be:

$$P_{rcv,v_i} = \sum_{j,v_j \notin M(v_j)} B(e_{ji}) W_{rcv\ b,M(v_i)} \quad (61)$$

Aside from the exit point of the application, all tasks will also transmit data:

$$P_{tr,v_i} = \sum_{j,v_j \notin M(v_j)} B(e_{ji}) W_{tr\ b,M(v_i)} \quad (62)$$

Thus the network lifetime is:

$$\max_{k,m_k \in M} \frac{W_{m_k}}{P_{idle_{m_k}} + \sum_{i,v_i \in T(m_k)} (P_{rcv,v_i} + P_{tr,v_i})} \quad (63)$$

Taking into account $P_{idle_{m_k}}$ is almost the same on all nodes, and presuming that we have the same amount of energy available from the batteries, the important part remains the energy consumed in radio transmission. Thus, to maximize network lifetime, a general goal would be to minimize this consumption. We do not consider tasks that need to be run on all capable nodes; we will address this as a restriction in the algorithm.

We have accounted for energy while transmitting and while receiving. Even if they are not exactly the same, their sum should be uniform over each bit transmitted, so we can say that the power used by the network in radio communication, P_{radio} is:

$$P_{radio} = \sum_{i,j,M(v_i) \neq M(v_j)} B(e_{ji}) K \quad (64)$$

We have reduced the scheduling problem to a known graph-problem, for which a polynomial algorithm has been found in 1988. It is called the min k-cut problem. If we imagine in our setup the assignment of tasks to nodes, and ascertain that no task is duplicated among nodes (we can enforce that easily by duplicating in advance tasks that have to be run on all nodes), then the scheduling is in fact a partition of the node sets $\{C_1, C_2, .., C_k\}$, each resulting set containing the tasks that have to run on that node. In graph theory, this is called a k-cut.

5.3.3.1 Minimal K-Cut

Given a graph *G = (V,E)* with a weight function *w* and an integer $k \in [2..|V|)$ the k-cut is a partition of V into *k* disjoint sets $F = \{C_1, C_2, .., C_n\}$ and its measure is the sum of the weight of the edges between the disjoint sets

$$\sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \sum_{v_1 \in C_i, v_2 \in C_j} w(v_1, v_2) \quad (65)$$

We can express the same formula in a simpler manner:

$$\sum_{i,j,v_i,v_j in\ different\ sets} w(v_1, v_2) \tag{66}$$

The minimal k-cut algorithm is described in the listing below:

Algorithm 1 Min K-cut algorithm

function KCut(V,k)
if k is even
  then
  k` = k - 2
else
  k` = k - 1
end if
S ← the set of subsets of k` elements from V
T ← the set of subsets of k-1 elements from V
Find $s \in S, t \in T$ such that W(cut(s,t)) = min
/* cut(s,t) splits V into s' and t'*/
/* Find the minimal cut(s,t) with maximal source set */
return s` ∪ KCut(V-s`, k-1)

5.3.3.2 The maximal source minimal s-t cut

The (s, t) cut, needed by the min K-Cut algorithm, is a partitioning of the vertices of a flow graph such that the source is in S and the sink is in T. The min K-Cut algorithm needs a version where the source and sink are a set of nodes, not just one. To do this, we can collapse the nodes in the sink set into a supernode. Edges on the interior of the supernode do not count for the search of the minimal s-t cut, only those on its exterior. We then proceed to solve the maximum-flow problem on the graph, interpreting weights as flow capacities. Using the residual graph (graph with edges that have weight the capacity-flux passing at one time), we can start from the sink and expand until we hit 0 residual capacity edges.

The nodes found will be the smallest sink set of a minimal s-t cut, the source set being the rest of the nodes.

$$r(i,j) = c(i,j) - f(i,j) \tag{67}$$

Algorithm 2. Maximal source minimal s-t cut

/* replace sink and source set by supernodes */
$V' = V \cup \{s,t\} - (S \cup T)$
modify the edges external to supernodes
$E' = \{e_{ij} | i,j \notin S,T\} \cup \{e_{si} | e_{ji} \in E \text{ and } j \in S\} \cup \{e_{ti} | e_{ji} \in E \text{ and } j \in T\}$

$\cup \{e_{st} | e_{ij}, i \in S \text{ and } j \in T\}$

$F_{ij} = 0, \forall i, j \in V$

loop

find path p from s to t in residual graph

$m \longleftarrow$ minimum residual capacity on path p

for all edges e_{ij} such that e_{ij} on path p do

$$F_{ij} = F_{ij} + m$$

$$F_{ji} = F_{ji} - m$$

end for

end loop

$A \leftarrow$ set of nodes reachable by BFS from t

$B \leftarrow V - A$

5.3.3.3 Adaptation of K-Cut

When reducing the scheduling problem to K-Cut, some constraints were ignored that now must be satisfied. Since every step of the algorithm is partitioning the node set in half, one being final and one remaining to be further partitioned, we can say that each step represents scheduling tasks on a single node. Constraints must be inserted in each step, relevant to the node whose tasks are being scheduled. In the pseudo code of the algorithm, finding the source set *S* is what must be modified to satisfy constraints.

We have several constraints that must be added to the algorithm:

- Some tasks can only run on compatible nodes
- Some tasks have to run on all capable nodes (e.g. sensing tasks)

To enforce the first constraint we have to include only compatible tasks (tasks *v* such that *NA(v,m) = 1*for the current node *m*) in the source set, as well as filter the cuts in which the first resulting set contains incompatible tasks. For the second constraint, we will include the tasks that have to be duplicated in the sink set of the cut (so that they will be available for the next step of the algorithm), then in the end we add those tasks to those obtained in the minimal (s,t) cut.

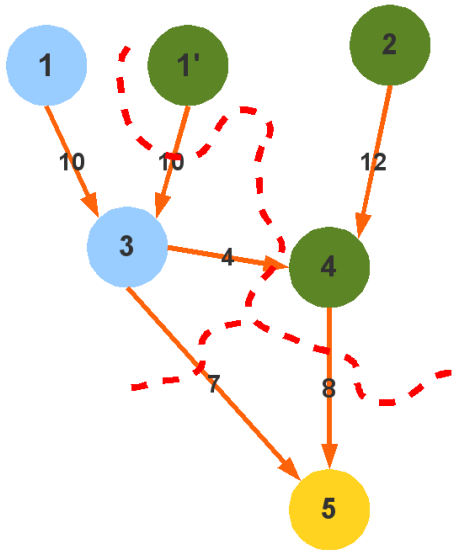


Figure 60: A directed graph with a task that has to be duplicated on all capable nodes

Algorithm 3. Adapted min K-Cut

function $AKCut(V, k, m_i)$
if k is even
then
k` = k - 2
else
k` = k - 1
end if
MT ← tasks that have multiplicity
V` ← V - MT
S ← the set of subsets of k` elements from V`
T → the set of subsets of k - 1 elements from V` ∪ MT
Find $s \in S, t \in T$ such that *W(cut(s,t)) = min*
/* cut(s,t) splits V into s` and t` */
/* Find the minimal cut(s,t) with maximal source set */
$T(m_i) = \{s'\} \cup \{v_j | v_j \in MT, NA(v_j, m_i) = 1\}$
return $T(m_i) \cup AKCut(V - s', k - 1, m_{i+1})$

Assigning the tasks for each node at each step gives great versatility in constraints management for the algorithm to better model and solve the problem. For instance, battery status has no role yet, but it is easy to add: We set a threshold for the battery/tasks ratio, if the current node crosses that threshold we will settle on another solution, not necessarily with the same min k-cut, but with less strain on the node.

5.3.3.4 Algorithm Complexity

The complexity of the min k-cut with the algorithm we described is:

$$T(n) = O\left(n^{k'+k+1}\right) \cdot O(n^3) + T(n-1) \tag{68}$$

Where $O(n^3)$ is the bound for the minimum (s,t)-cut algorithm.
For k even, we have:

$$O\left(n^{2k-3}\left(n^3 + n^{2k-4}\left(n^3 + .. + n^4(n^3 + n^3)\right)\right)\right) = O(n^{k^2}) \tag{69}$$

A more precise formulation is:

$$O\left(n^{k^2 - \frac{3k}{2}+2}\right), k\ even \tag{70}$$

$$O\left(n^{k^2 - \frac{3k}{2}+\frac{5}{2}}\right), k\ odd \tag{71}$$

The algorithm proposed solves the schedule problem with minimal energy consumption. It is designed for heterogeneous networks and it is application-independent. Although it was described as for a mesh topology network, the idea can be easily extended by introducing a "hop" factor in the communication between certain nodes - as dictated by topology. The scheduler has however algorithmic complexity, a different solution based on an approximation algorithm of the same problem is required for large-scale networks.

5.3.4 Task-Scheduling Implementation in the Check Framework

The scheduling [112], [113][114] component of Check [115] has three ways of controlling the running of tasks in WSAN islands, namely: by starting/stopping a task on a sensor; choosing data sinks for output data of a task; or by adjusting the frequency with which data is outputted. The scheduler then takes decisions by considering the state of the network, as well as the priority and complexity of the task to be executed. Currently there are two types of tasks being considered, namely: tasks that only have outputs (e.g. periodic tasks, that just calculate different metrics or send sensor data to data sinks), and tasks that process data. The latter can aggregate sensor data from several sensors (they are subscribed by the scheduler to the other sensors' data) and, for example, transmit a mean value or detect sudden change in the data. The scheduler then picks the nodes or sensors that can handle the extra energy loss due to the extra communication required to execute this type of tasks and schedules the tasks on these new resources. Obtaining information from the task can be done in one of two ways, namely: collecting data directly, with get/set operations, or by using the data sink method, where the entity that connects to the sensor can register itself or another entity as a data sink for the output that is generated from the task.

The Scheduling component of the Check Management Protocol Suite is presented as a stand-alone application that gathers information (e.g. types and characteristics of the nodes or topology) about the network from a Resource Directory. Prior to executing the scheduler, the sub-tasks of the application have to be already programmed on the nodes, as stopped tasks. The scheduling component itself can be invoked with the following command:

./check-schedule <RD> <tdag> <tskmap>

The arguments of the task-scheduling component are the following:

- RD is the URI of the Resource Directory
- tdag is a Task Dependency Acyclic Graph. It contains a graph of data dependencies between tasks and their associated bandwidth. The format is a text file containing the following components:
    - t – the number of tasks.
    - d – the number of dependencies.
    - a – a tuple (x,y,z) for each edge in the dependency graph on one line, with x,y being incident tasks and z being the bandwidth.
- tskmap is a task mapping file, associating each task in the Task Dependency Acyclic Graph to the names of the tasks that have to be run on the network.
    - tuples (x,s) are present on every line, x being the task index in the tdag file and s being the name of the task associated. Tasks are started based on their names.

An example of a tdag file is given here:

4 4
0 1 10
1 2 1
0 3 20
2 3 1

The application in question has one sensing task (0), two different event detections (tasks 1 and 3), and a sink task which is notified in the case of an event (task 2). Accordingly, the associated tskmap of the tdag file will be:

0 sense
1 ed1
3 ed2
2 present

One run of the application is enough to calculate the best scheduling of the application over the given network, the application will directly access the nodes and start the required tasks, provided the tasks are already present on the sensor nodes.

An unconventional scheduling algorithm that uses energy as main constraint instead of time is currently employed in the Task-Scheduling Component of Check. The scheduler has only basic information about the tasks, for example: importance, affinity to a certain node, running frequency and dependencies and chooses which assignment is the best for a minimal energy

consumption scheme. Energy costs are considered to be proportional to the quantity of data to be transmitted. The proposed algorithm solves the scheduling problem so that a minimal energy consumption scheme is ensured. This scheme is designed for heterogeneous WSAN islands and it is application-independent. Although it was first designed for a mesh topology network, the idea can be easily extended by introducing a "hop" factor in the communication between certain nodes – as dictated by the network topology.

The scheduler suffers from a considerable algorithmic complexity; and a different solution based on an approximation algorithm for the same problem is required for large-scale WSANs. A viable approach would be to use the theorem in [116], which states that the k-cut problem can be solved with Gomory-Hu trees within twice of the optimal solution. An implementation based on this approximation, is detailed in [117][118], and has proven to be within twice of the global optimum.

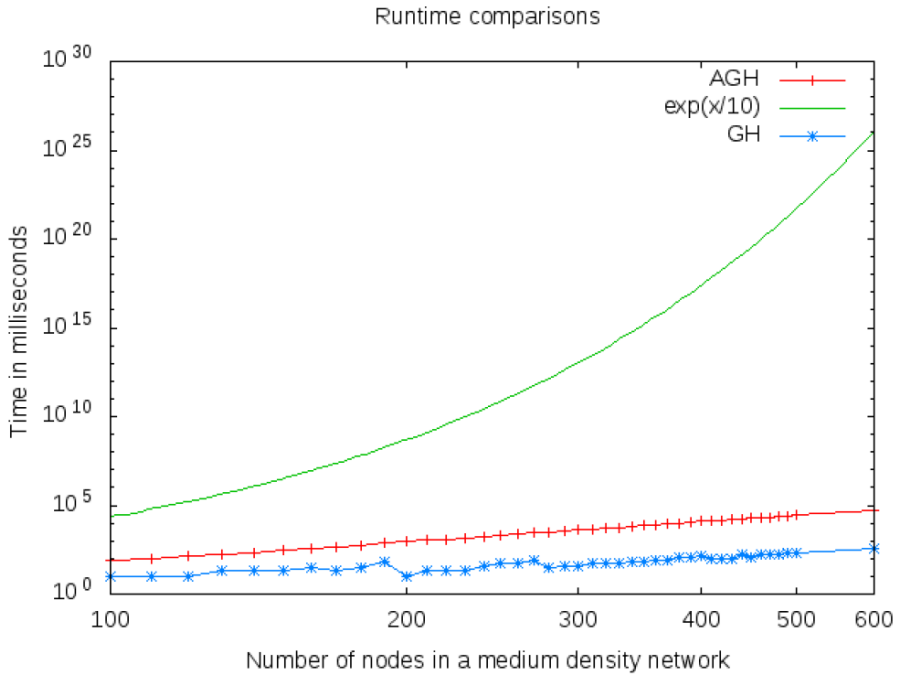


Figure 61. Runtime comparisons (on a semi-logarithmic scale) of variants of the approximation solution, GH is the standard Gomory-Hu algorithm, while AGH is our solution of the scheduling problem based on Gomory-Hu.

The approximation solution to the scheduling problem currently implemented in the Check Task-Scheduling Component, as shown in Figure 61, also proves to be a viable alternative in terms of complexity. Asymptotically it has the same complexity as the Gomory-Hu algorithm on which it is based, although with a higher constant.

5.4 Wireless Sensor Network Monitoring for the Check Framework

The architecture of the Check Management Protocol Suite is based on MonAlisa (MONitoring Agents using a Large Integrated Services Architecture) [119], [120] which is used to collect, store and display the data in a scalable manner. There are several components and libraries used in this system. The monitoring service is a Java Linux application that collects data from a machine and stores it in a local database. The repository is a web application that can store and display sensor data. It acts as a client for the monitoring services by registering itself to specific data sets and receiving near-real-time updates of the changes.

5.4.1 The Monitoring Component Implementation

To fully exploit the capabilities of a wireless sensor network, the motes must be programmed with efficient code that handles the power management tasks, data acquisition and communication between the nodes. The recent trend in WSN software is to run small implementations of operating systems on the motes themselves to better manage the multitude of tasks that run in parallel.

Operating Systems for wireless sensor network nodes are typically less complex than general-purpose operating systems both because of the special requirements of sensor network applications and because of the resource constraints in sensor network hardware platforms. For example, sensor network applications are usually not interactive in the same way as applications for PCs. Because of this, the operating system does not need to include support for user interfaces. Furthermore, the resource constraints in terms of memory and memory mapping hardware support make mechanisms such as virtual memory either unnecessary or impossible to implement.

We experimented with different OS images (TinyOS, ArchRock Stack, Meshnetics Stack) and we found that the best performances were given by Contiki OS.

Contiki is a small, open source, highly portable, multitasking computer operating system developed for use on a number of memory-constrained networked systems ranging from 8-bit computers to embedded systems on microcontrollers, including sensor network motes. The name *Contiki* comes from Thor Heyerdahl's famous Kon-Tiki raft.

Despite providing multitasking and a built-in TCP/IP stack, Contiki only needs a few kilobytes of code and a few hundred bytes of RAM. A full system, complete with a graphical user interface, needs about 30 kilobytes of RAM.

The basic kernel and most of the core functions were developed by Adam Dunkels at the Networked Embedded Systems group at the Swedish Institute of Computer Science.

Contiki is designed for embedded systems with small amounts of memory. A typical Contiki configuration is 2 kilobytes of RAM and 40 kilobytes of ROM. Contiki consists of an event-driven kernel on top of which application programs are dynamically loaded and unloaded

at runtime. Contiki processes use light-weight protothreads that provide a linear, thread-like programming style on top of the event-driven kernel. Contiki also supports per-process optional preemptive multi-threading, inter-process communication using message passing through events, as well as an optional GUI subsystem with either direct graphic support for locally connected terminals or networked virtual display with VNC or over Telnet.

Contiki runs on a variety of platform ranging from embedded microcontrollers such as the TI MSP430 and the Atmel AVR to old home computers. Code footprint is on the order of kilobytes and memory usage can be configured to be as low as tens of bytes.

Contiki was successfully ported to the Sparrow motes and has since been used in all the experiments concerning the wireless sensor network.

We developed a method for monitoring and controlling a heterogeneous WSN remotely over the Internet, based on the successful MonALISA framework (Monitoring Agents using a Large Integrated Services Architecture). Our method uses an abstraction layer to provide remote monitoring and control to essentially any kind of WSN, including the Sparrow motes [121].

MonALISA is 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 client software can also access control services that run next to the data repositories using a secure, authenticated protocol. The connections can be established over the Internet, allowing user access to the WSN from any location, or over a local area network. Because multiple ApMon instances can run independently from the data repository, a WSN composed of multiple islands, in different locations, can be transparently managed as one single entity.

5.4.1.1 Monitoring System Architecture

WSNs are typically accessed through one or more devices generically called gateways or routers. These are typically connected to large computers such as PCs, along with specific drivers. Together with the driver, the gateway allows user programs to access the WSN through standard Internet protocols such as IPv6, or through special interface programs.

A monitoring service („monalisa-wsn”) runs on the computer with the WSN gateway, calling WSN-specific programs that report incoming data. These programs call the WSN drivers and perform WSN-specific data formatting, while presenting a unified interface to the monitoring service, effectively forming an abstraction layer. The monitoring service then uses ApMon to upload the data into a MonALISA repository running on the same computer or on a remote server. Users can then connect to the repository through a graphical client program and retrieve the data and analyze it. Using asymmetric key authentication, the user can connect to a MonALISA WSN control module that runs next to the data repository. A control service („monalisa-wsn-ctl”) runs next to the WSN gateway and connects to the MonALISA control

module, enabling the user to send data and commands to the WSN nodes. Figure 62 shows a schematic diagram of the WSN monitoring and control system.

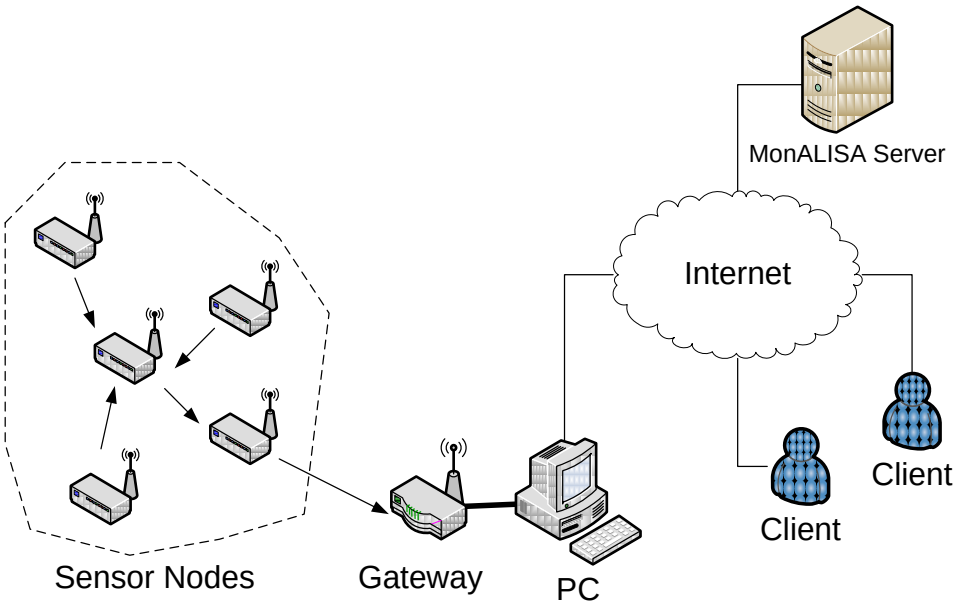


Figure 62. Sparrow WSN architecture

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. This stems from its main usage as a grid monitoring framework. Farms, clusters, hosts and parameters are identified by their name and presented to the user in a hierarchical interface. The application that uploads data is free to define any host name, parameter name or parameter value. A convenient way of using MonALISA to monitor WSNs is to present a WSN as a host-type entity with a list of parameters. The names of the parameters include a part which identifies the WSN node in case of node-specific parameters. For example, if monitoring temperature sensor readings from a WSN containing 4 nodes, the parameters may be named „temperature1" to „temperature4". The user can filter parameters by name in order to concentrate on data of immediate concern.

The parameters are sampled at defined intervals by the monalisa-wsn program by polling the WSN, or they can be reported by the WSN services automatically. In any case, users can view parameter values in near-real-time, as well as their history. Figure 63 shows an example of monitoring a sensor network composed of two islands located in two different locations. Each island is connected to a PC via a gateway device and each PC is running WSN-specific drivers and polling adapters. One island is using Sensinode NanoSensor hardware, which is readily capable of measuring temperature and light, and the other is using Atmel Raven hardware which only measures temperature readily. Numerous other types of sensors can be added to both platforms.

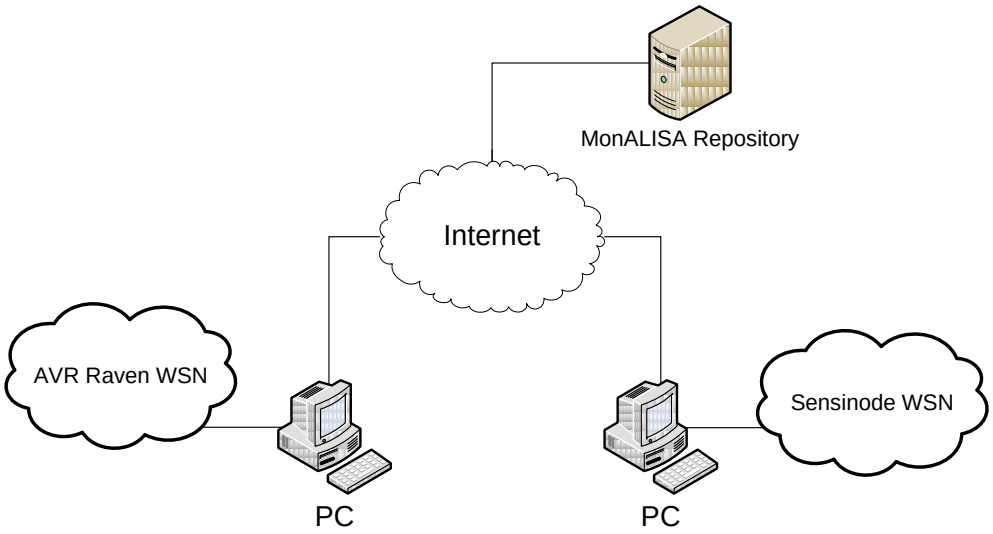


Figure 63. Monitoring of different geographically-located WSN islands

Although all WSN nodes can run an infinite variety of custom-designed real-time operating systems and embedded applications, a readily-available solution is often preferred. For instance, the Sensinode NanoSensors can run the Sensinode NanoStack, which is a 6LoWPAN implementation (IPv6 over low-power wireless personal area networks). For debugging purposes and simple applications, it can also run a simpler version that uses MAC addressing instead of IPv6. The Atmel Raven comes pre-programmed with a ZigBee-based network stack that uses 16-bit node IDs for addressing. The nodes have to explicitly associate with a coordinator (gateway), unlike Sensinode NanoSensors which can be detected by broadcast queries. The Raven can be programmed with the Contiki operating system, which implements 6LoWPAN, allowing IP addressing.

Above the network protocol, each case uses a different application protocol for polling data from the sensors. NanoSensors use two command-line programs called „nPing" and „SSI-Browser" to detect sensor nodes and obtain sensor values respectively. The Raven nodes running the default software use a graphical program that connects to a „wireless services" back-end. The protocols used are known. Contiki uses a web interface accessible directly from a web browser over IPv6, but can also be configured to use a protocol with lower overhead.

Each case needs an adaptation program that is called from the main monitoring service and returns parameter-value pairs in a consistent, WSN-independent format. This abstraction layer has been implemented and tested for the technologies listed above. A version for the Titan framework can be developed, which would allow monitoring performance parameters and custom service data.

The adaptation program can choose to be executed periodically by monalisa-wsn and return a data point each time, or to provide data points on the standard output periodically. The first option is used when polling the WSN and the second is used when the WSN itself pushes the data. When polling the WSN, each node from a list is queried for certain data such as sensor

readings, performance metrics or debugging information. The nodes are identified by their address, which can have a wide variety of formats, depending on the hardware and software used. A technology-specific detection program, „detect”, is used to build that list, which monalisa-wsn then uses transparently. The list can also be built dynamically when nodes announce their presence to the gateway, as is the case with the Raven platform. In this case the corresponding „detect” runs permanently and updates the list when needed.

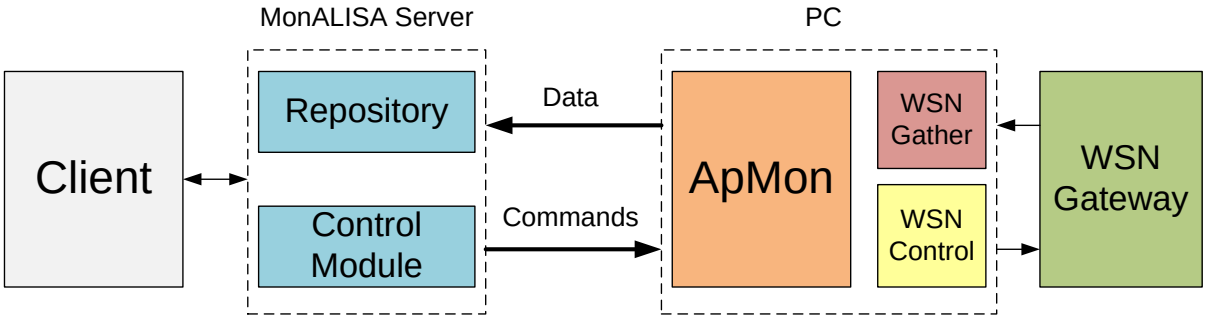


Figure 64. Monitoring and control system architecture

In order to provide increased uptime, a supervisor program can watch that the various components of the framework are running correctly. It can for instance restart programs that have crashed or locked, such as the WSN driver (Sensinode for example uses a stand-alone process as a driver, to which the other programs connect through sockets), the node detection program or the main monalisa-wsn program.

It is important that the WSN monitoring service be able to run on a large variety of computer systems. Some WSNs for example are not connected to a PC, but use an embedded system such as an ATNGW100 for remote access. ApMon and monalisa-wsn are written in Perl, which is a portable scripting language. Perl can run on a large variety of computers, including embedded systems. The WSN drivers are usually written in C and can be compiled for mostly any system if their source code is available. The abstraction layer is written in Perl, Python and Unix shell, making it also highly portable.

The MonALISA graphical client (Figure 65) is written in Java, therefore it is capable of running on any modern PC operating system.

The monitoring component of the Check Management Protocol Suite is plugin-able. Check does not internally store monitoring information, but uses other monitoring suites to provide these services. In this case, we use a MonAlisa service. This service is used to store and graph data for long periods of time. Check plugs itself as a resource into the Resource Directory and can thus be used by any service just by using a REST interface to register new nodes and parameters to monitor.

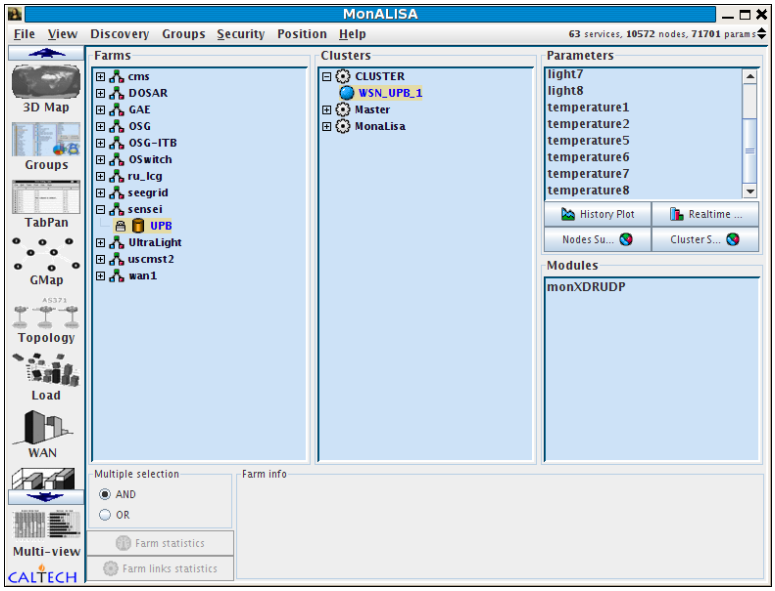


Figure 65. MonALISA graphical client user interface showing the UPB WSN cluster data

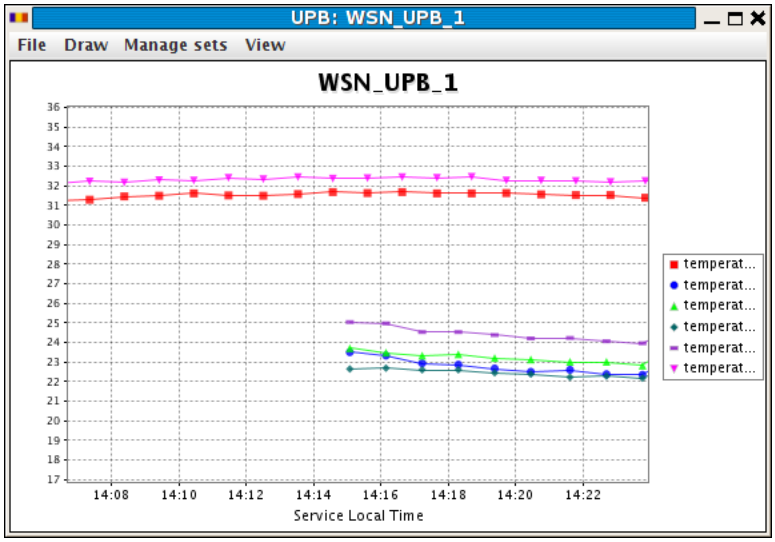


Figure 66. Data aggregation example from geographically distinct areas monitored by two WSN islands.

The installation of this component requires the availability of a Java 6 instance, a MonAlisa Service which is used to monitor sensor data, as well as a MonAlisa Repository that is employed to store the data for long periods of time and register new nodes that should be monitored by the Check Protocol Suite. This Check component belongs to the management directory of the SENSEI Source Code Repository, as follows:

- /SenseiRepos – the MonAlisa Repository where files and configuration are saved.
- /SenseiService – where the Monitoring Plugins are stored.

- /SenseiHibernate – contains the Rest Monitoring Interface for Check.

To enable the monitoring plugins one must configure the service as follows:

 # Properties to add
lia.Monitor.CLASSURLs=file:${MonaLisa_HOME}/Service/usr_code/SenseiService/build/

The plugins that are available on the monitoring Check component are:

- monSensei – monitors resources using REST. The resources to monitor are then given by the REST interface of the Check Protocol Suite.
- monPing6 – monitors the status of a node by pinging it through IPv6 ping.
- monService – checks the availability of a service.

In order to be able to use these plugins, one must enable them as follows:

 # To add in myFarm.conf
***Topology**
>gw.ncit.pub.ro
monPing%60

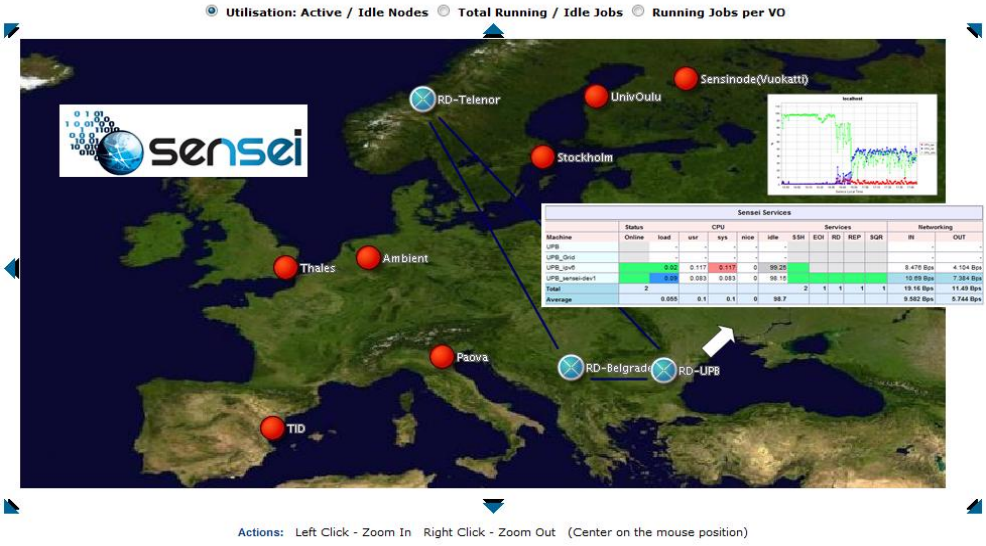


Figure 67. Graphical User Interface of the Check Monitoring Component.

As was mentioned before, the data extracted from all WSANs sensor nodes is stored in a local database for three hours. For long term storage however, a dedicated database is employed. Examples of monitored data stored in these databases are given in Figure 67 and Figure 68.

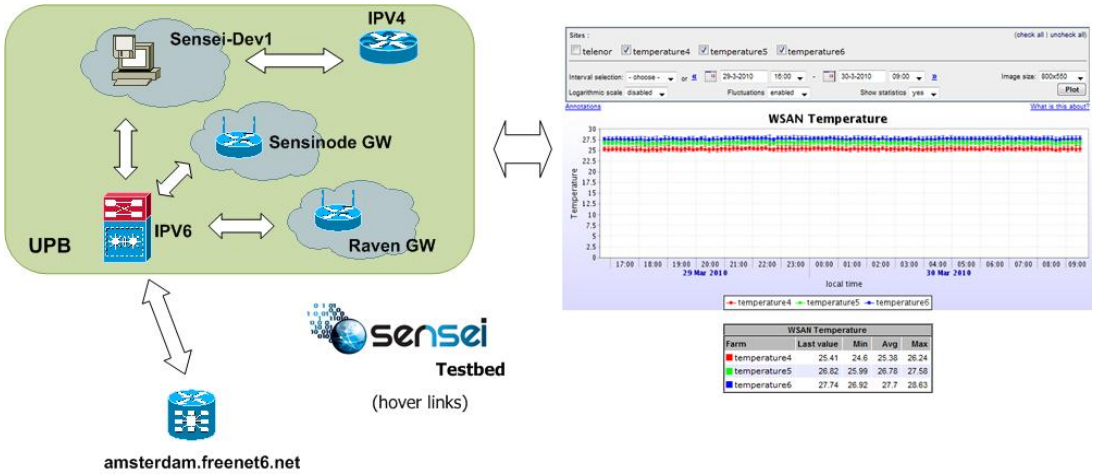


Figure 68: Data gathered and presented using the Check Monitoring Component.

Chapter 6

Wireless Sensor Networks Applications

In order to validate the theoretical and experimental research done in the previous chapters, we needed to evaluate our system's behavior and assess its efficiency when deployed in a real-life environment. For this we chose two separate types of application scenarios, each with its own distinct set of constraints and requirements. They are presented in the following sections.

All applications use the Sparrow sensor nodes presented in Chapter 4, to which additional circuits and extension boards have been added to enhance their functionality. New platforms have also been developed and built, such as an embedded Linux gateway for the home automation application and a mobile measurement unit for the air pollution monitoring application.

6.1 Home Automation

We propose a new wireless sensor network architecture that is especially designed for the task of home automation [122]. 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.

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 [123], Zigbee [16], WirelessHART [124], or 6LoWPAN [125].

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.

Prior work exists in the field of Web-enabled Home Automation systems presenting both IP networks and WSNs[126]. The authors in [127] 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 [126] 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.

The 6LoWPAN standard promises the fulfillment of the emerging trend of embedding Internet technology into all aspects of everyday life [127], 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.

6.1.1 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 [127], 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 needs 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.

6.1.2 Overview of System Architecture

Our goal is to develop a house monitoring system that is robust, flexible, user-friendly 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:

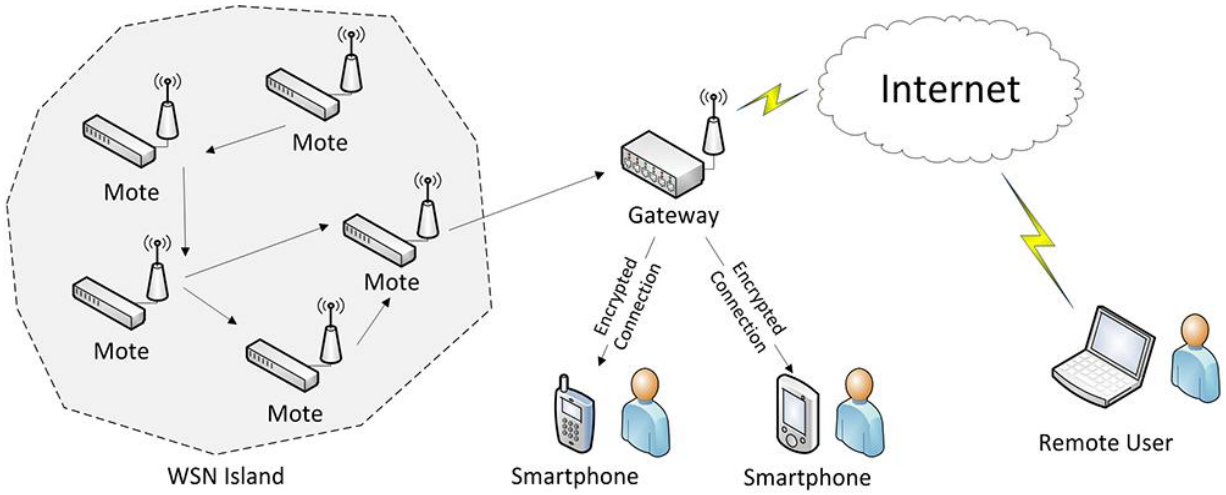


Figure 69 System Architecture

The embedded gateway is the core of the system. It pro- vides the user with a touch screen 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. Collected data is then 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.

6.1.3 Sensor hardware

For this project we used the Sparrow v2 node [88], presented in Chapter 4 which we extended for use 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 by attaching an energy harvesting module.

The voltage 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.

6.1.4 Sensor firmware

The nodes in our testbed run a light-weight operating system designed specifically for use in WSNs, named Contiki [15]. 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 interface with 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 has 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.

6.1.5 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 [128].

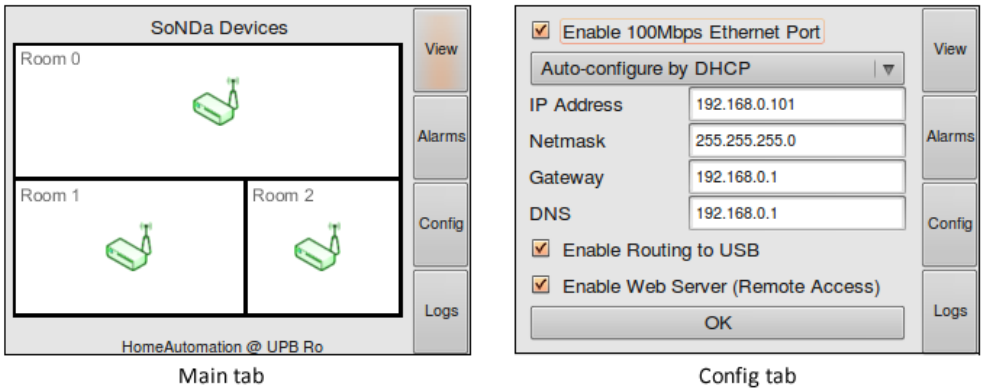


Figure 70 Screenshot of the embedded gateway system

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 touch screen, 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 70 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.

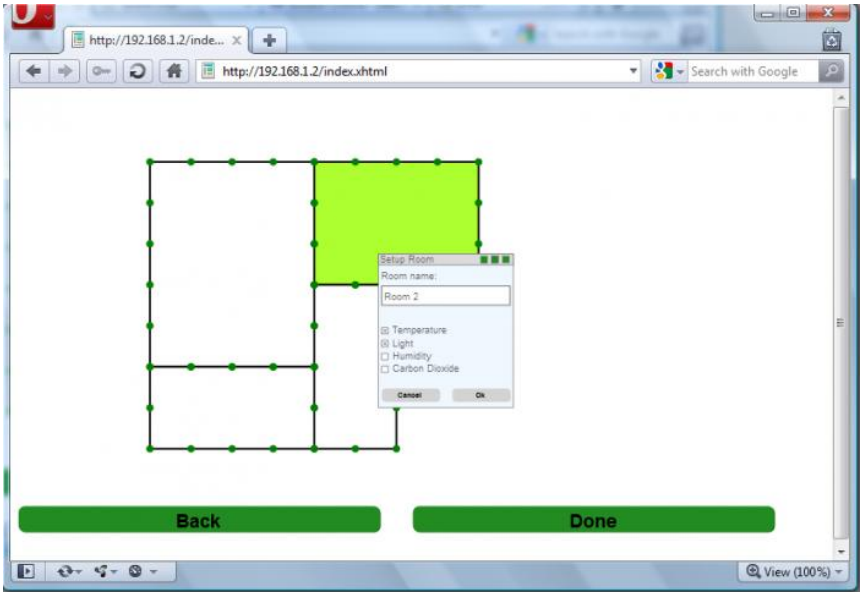


Figure 71. Customize interface screen

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.

6.1.6 Application Monitoring

6.1.6.1 Embedded Monitoring Application for the Android Platform

The Android application is meant to give additional ways of presenting and manipulating data obtained from the gate- way. 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 parameter 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 a 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 within 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>]

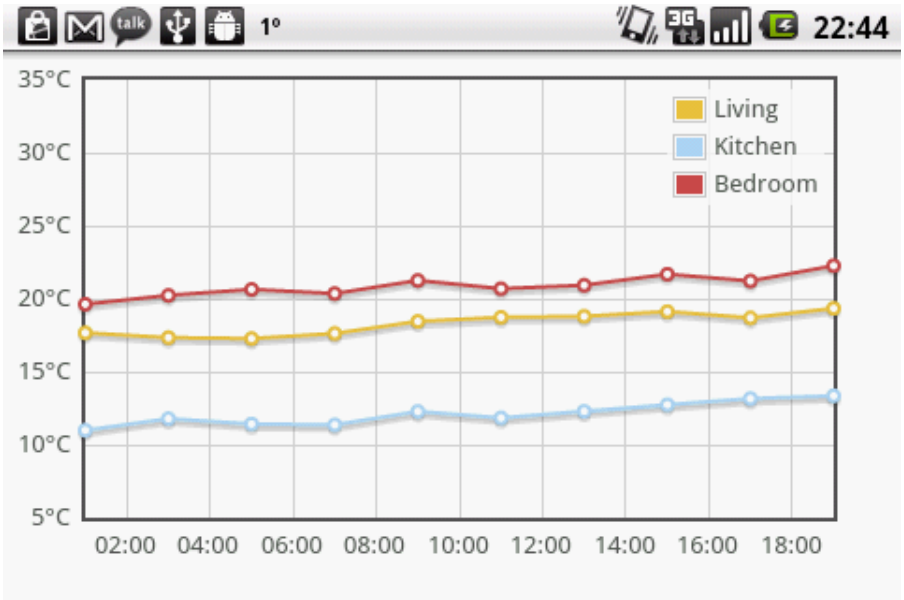


Figure 72. Android monitoring application

6.1.6.2 Online Monitoring

We developed a method for monitoring and controlling a WSN remotely over the Internet, based on the successful MonALISA framework [128]. 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.

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.

6.1.7 Conclusions

We presented a new wireless sensor network architecture that was especially designed to be employed in home automation. Our system employs the concept of energy harvesting to maximize network lifetime and availability, an idea which is novel to the field of home automation.

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.

6.2 Mobile Pollution Monitoring

In this section we present a mobile system for air quality and pollution measurement suited for the urban environment [129]. We designed, tested and built a reliable measurement device that can acquire information about the air quality of its surroundings, store it in a temporary memory buffer and periodically relay it to a central on-line repository. Real-time gathered data can be freely accessed by the public through an on-line web interface. Users can select and view different gases and concentrations overlapped on a map of the city.

One of the major environmental concerns of our time is air pollution. Apart from severely degrading the natural environment, air pollution directly affects our health. Short term and long term effects range from light allergic reactions - irritation of the nose, throat and eyes - to serious conditions like bronchitis, pneumonia, aggravated asthma, lung and heart diseases. Air pollution is also the cause of many premature deaths (50,000 to 100,000 deaths per year in the U.S. alone, 300,000 in the EU and over 3,000,000 worldwide [130], [131], [132].

We propose a system that provides live access information about air quality in crowded urban areas such as crossroads or highways. The goal is to monitor the air quality by designing a system that can actively pinpoint pollution sources and polluted areas in traffic.

The solution offered by our system relies on a portable device that can detect an abnormally high concentration of an air pollutant. The device records the measured data along with location coordinates and can periodically transfer it to a computer through a wireless GPRS connection. With the user's acceptance, the application can share the data that will be displayed on a dedicated web site. As a result, the system's user - and the entire community - can benefit from a potentially wide information-gathering network.

Probably the best approach to solving the environmental problems caused by air pollution is to make people aware of it and of their actions that possibly favor it. Our system does just that by gathering and publishing meaningful and up-to-date information. This is the result of people using our devices and sharing the measured data.

Some sources of information on air pollution already exist and are publicly available [133], [134], [135]. However, they do not cover the entire monitored area as they are based on measurements performed at fixed locations. Thus, it is not easy to discover localized pollution sources. Our system has the potential of collecting much more data than a traditional one due to its multi-agent architecture. The more users in the system, the larger the area covered will be and better granularity. In addition, our devices could be installed on public transportation vehicles in order to provide an accurate overview of the pollution generated by traffic.

6.2.1 Air Quality

Engine exhaust consists of various gases and particulate emission, which in turn consist of various inorganic and organic compounds with great molecular weight. Their composition depends on driving condition, engine type, gas emission controller, operational temperature, and other factors, all of which make the emission pattern more complicated. The type of pollutant emitted by the engine fueled with gasoline compared to the one fueled with diesel is similar; the difference is in its proportion because of differences in engine operation.

Although engine exhaust gas consists of harmless compounds such as nitrogen, carbon dioxide, and water vapor, it also contains chemical compounds that are harmful to humans and to the environment alike. These compounds are carbon monoxide (CO), various hydrocarbon compounds, various oxides of nitrogen (NO_x), oxides of sulfur (SO_x), and dust particles including lead (Pb), [136], [137]. Here are the reasons why our Mobile Unit sensors cover the above-mentioned compounds:

- Carbon Monoxide (CO). This is produced when the fuel in the engine does not burn properly. Road traffic produces 91% of all CO emissions. Problems caused: When inhaled it reduces the oxygen carrying capacity of one's blood and can cause headaches, fatigue, stress, respiratory problems and at high levels - death.
- Nitrogen Oxides (NO_x). These are produced from the burning of fuel in the engine. Road traffic is responsible for 49% of all NO_x emissions. Problems caused: NO_x emissions help to make 'acid rain'. They also combine with hydrocarbons to form low level ozone pollution and may contribute to lung disease.
- Hydrocarbons (HC). These are compounds of hydrogen and carbon and are present in petrol and diesel. Benzene is an example. Road traffic is responsible for about 35% of all HC emissions. Problems caused: Hydrocarbon emissions are carcinogenic and a major ingredient of smog.

Bearing all these aspects into mind, it comes as a consequence that our sensors measure these pollutant factors right from the emission source. The data gathered must on the one hand be compared to the admitted levels (in Europe, there is the Euro standard) and on the other hand recorded locally onto the vehicle itself. A bare 256 MB non-volatile memory card suited on each Mobile Unit can record for as much as several years' emissions log.

6.2.2 System Overview

Our system is comprised two major components:

- a mobile client, called the Mobile Unit
- an on-line web server

The **Mobile Unit**, or MU, is entrusted with harvesting the information from the sensors enclosed; the parameters are: combustible gas, carbon monoxide, temperature, air contaminants and gasoline / diesel exhaust. The various parameter levels are transmitted by a GSM link to the web server and displayed live on a liquid crystal display.

The **On-line Web Server** provides user access to pollution statistics. It queries the database and reveals the information gathered at a certain time of day in a specific location. Multiple recordings of geographical locations that are in close time and space proximity are averaged. As a consequence, there is a tighter control over mobile sources of pollution which until now, could not be managed remotely.

6.2.3 Mobile Unit Design

The Mobile Unit has been designed and built according to the project's specifications. Being a mobile device, meant to be embedded onto a car, it relies solely on the car's power supply. The entire system can be easily included into the car's onboard computer, and the live information on polluting parameters could be provided to the driver through the dashboard display. As it is built to measure exhaust gas concentration, the Mobile Unit will be powered up only when the engine of the car is running.

The device must be exposed to a clear airflow so that it can properly take readings. In order to fulfill this requirement the sensors are placed in a separate case which can be attached to the outside chassis of the vehicle using magnetic clips.

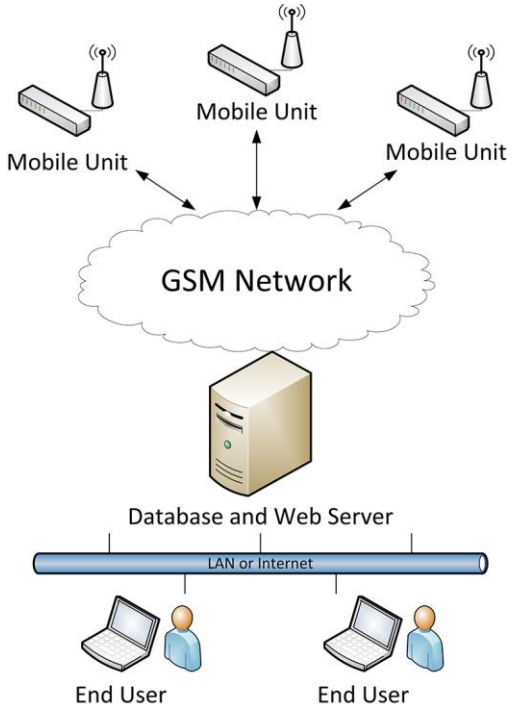


Figure 73. Overview of the information flow in the system

1) The Main Module: The Main Module is a control unit that implements the functionality of the device. It is build around a microcontroller and its main purposes are:

- control the data acquisition from the sensors
- record and store the data measured by the sensors
- display sensor data in a graphical form on the LCD display
- direct the communication between the device and the data server over the GPRS data link

2) The Data Acquisition Module: This module measures the concentrations of different gases and converts them to a format understood by the main module. Because uniform access to the sensors was required, the module has a Sensor Interface. The device is equipped with three sensor slots that can accommodate various sensors. One of the most important aspects of the device is the fact that the sensors are pluggable: the user can select from various types of sensors

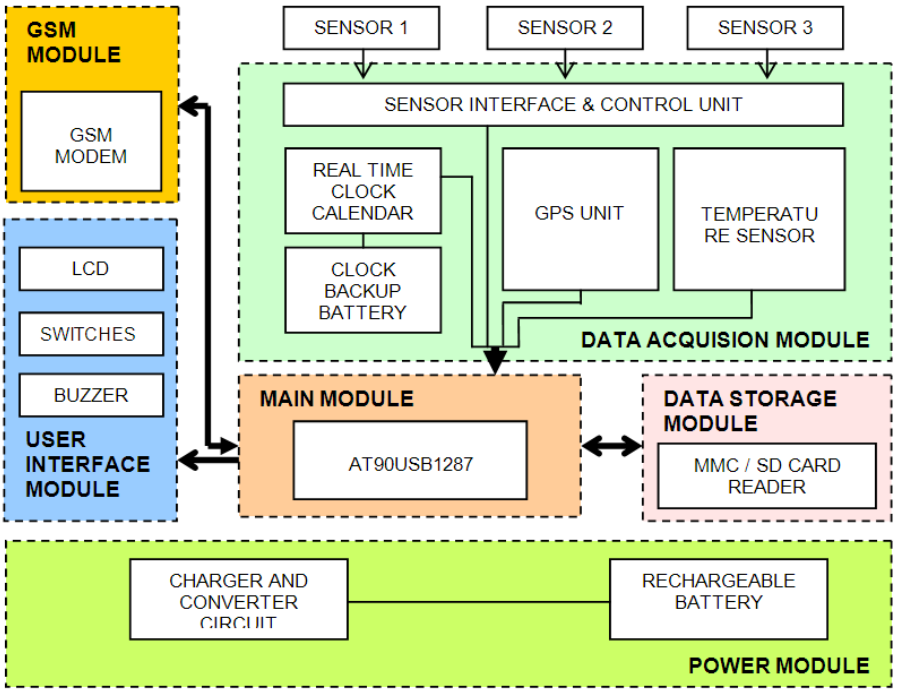


Figure 74 Mobile Unit hardware diagram

the ones that are most relevant to his situation and plug them into the device. A list of sensors that our device currently supports and the associated health threats includes sensors for the following pollutants:

- Carbon monoxide - generated by incomplete combustion of carbon - even relatively small amounts of it can lead to hypoxic injury, neurological damage, and possibly death [138];
- Ammonia - one of the most widespread gases - children with asthma may be particularly sensitive to ammonia fumes; also a significant part of respiratory

allergies are related to this gas and prolonged exposure to ammonia may cause nasopharyngeal and tracheal burns, bronchiolar and alveolar edema, and airway destruction resulting in respiratory distress or failure [139];

- Hydrogen sulfide - generated by bacteria as part of organic material decomposing - can cause asthma attacks, eye, throat and lung irritation, nausea, headache, nasal blockage, sleeping difficulties, weight loss and chest pain [140];
- Gasoline and diesel exhaust - major pollutants of populated areas - exposure to this mixture may result in asthma attacks, increase likelihood of cancer, chronic exacerbation of asthma and other health problems [141];
- Natural gas, propane, methane and other petroleum derivative gases - essentially fossil fuels - that can cause irritations to the upper respiratory tract or, in contact to a source of heat, can provoke fires and explosions [142].
- Carbon dioxide and general indoor pollutants - generated by a multitude of human activity - indirectly increase the likelihood of asthma attacks and may cause a rise in asthma cases among children [143].

Thick film metal oxide semi-conductor sensors from Figaro [144] were the preferred choice for our system, because of their good sensitivity to target gases, simple interface circuitry, low cost and long life.

Because the sensors' output values depend on the temperature of the environment, a temperature sensor is used to obtain this information and adjust measured values accordingly.

Gas Type	Measurement	Sensitivity	Response Time
CO_2	350−10000ppm	350ppm	1.5min
NO_x	0.1−10ppm	0.3ppm	30s
CO, HC	10 − 1000ppm	10ppm	30s
NH_4	30 − 300ppm	50ppm	2min

Table 8. Sensor specifications

In order to provide an accurate description of the pollutant agents' situation in a geographical area, the device is equipped with a GPS module that has an embedded antenna. When GPS connectivity is available, the records will have the current geographical coordinates marked. When the GPS signal is blocked, the geographical locations for the records are interpolated based on the coordinates immediately before and after the hiatus. For performance considerations, the system performs the interpolation in the on-line application. Along with the geographical coordinates and the recorded values of the pollutant gases, the time and date of the data acquisition must be recorded. This is done by a real-time clock chip. The module also contains a backup battery. When the device is on, the main power module powers the clock chip. If the power module is unavailable, the backup battery takes up the power supply function. The internal clock is synchronized with the real-time clock of the GPS whenever a GPS connection is available.

3) The Data Storage Module: One of the requirements for the Mobile Unit is to store the measured data together with contextual information such as geographical coordinates and time. We have installed a SD Card Interface, which allows the connection of any commercially available memory card, thus providing the device with virtually infinite storage space. The Serial Peripheral Interface (SPI) protocol is used to address the memory.

4) The User Interface Module: This module provides a way for the device to display real-time measurements from the sensors and GPS module. It also displays information about GSM connectivity and can signal if certain air pollutant values have crossed a preset threshold.

5) The GSM Module: The GSM Module serves as an interface between the device and the PC. It is built around a WAVECOM dual band GSM/GPRS modem that can stream sensor data over the existing cellular network.

6) The Power Module: The power source is a 3.6V Li- Ion cell mobile phone battery. All of the electronic devices are powered directly from the battery. The sensors, however, require a 5V power supply. A DC-DC converter is used to achieve this. The module also has a circuit that can recharge the depleted battery from an external DC adapter.

6.2.4 Software Design

The software development for our system is split into two parts - the Mobile Unit firmware and the Data Server. On the Mobile Unit side, the software was developed in C, using the WinAVR compiler suite for AVR microcontrollers. The server side was developed using Flex for the web interface and MySQL for the database that stores sensor and location data.

A. Mobile Unit

The microcontroller periodically reads sensor data and stores it in the flash memory along with the positioning data and time stamp from the GPS module. Sensor data is also displayed on the Mobile Unit's LCD and the user can select viewing instantaneous values or historical values for each gas type in the form of a data plot. From the system's memory, gathered data is sent via the GSM/GPRS modem to the server once every hour.

B. Data Server

The server is a desktop machine that is permanently connected to the Internet [145] and has a GSM modem attached to its USB port. A special daemon was written to interface the modem to the MySQL database and to handle data calls from multiple Mobile Units. For data visualization, the Flex application queries the database and overlays sensor data on a map of the city, using the Google Maps API. Users can browse through the data and display individual gas concentrations or all pollutants at the same time. They can also select specific time periods for the displayed data or zoom in to see individual measurement point values.

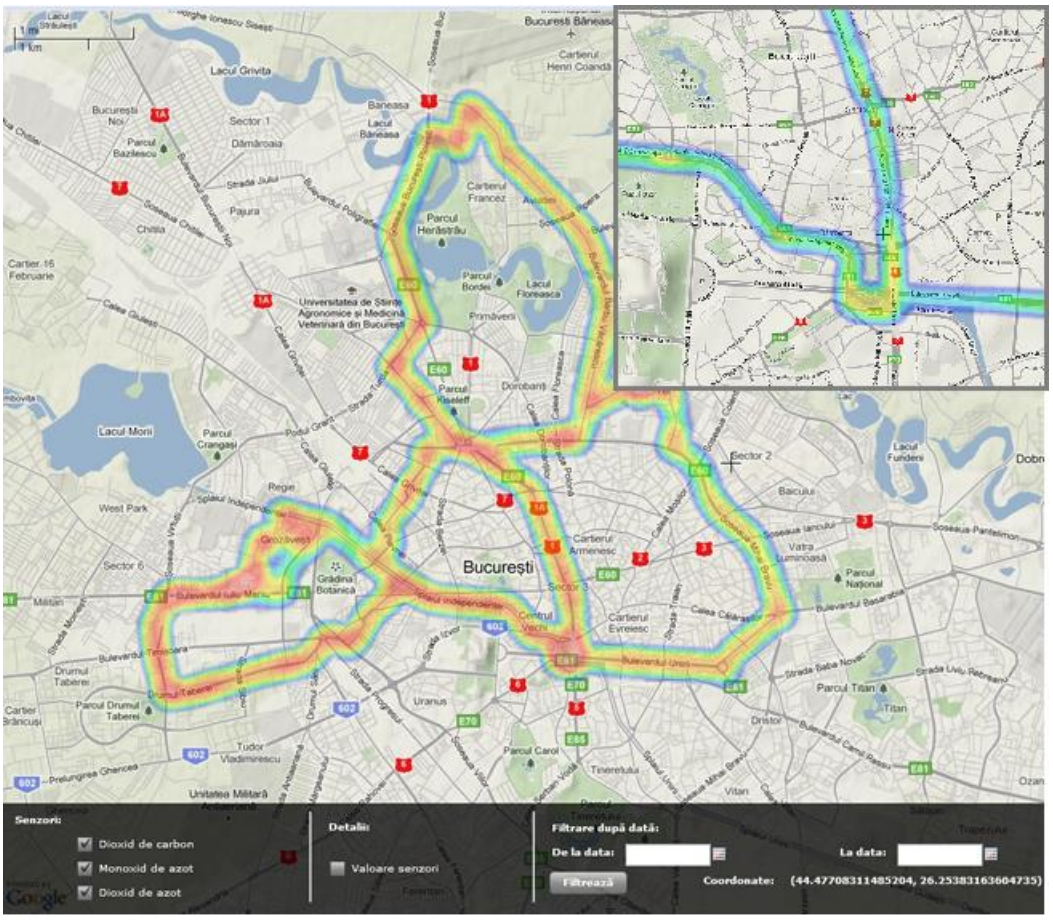


Figure 75. Web interface showing air pollution along some of Bucharest's busiest streets. Zoom-in showing pollution around Unirii Square.

6.2.5 Testing and Experimental Results

For early prototype testing, we carried static measurements over long periods of time. The Mobile Unit was placed in a fixed position on the rooftop of our faculty building and the sensors were exposed to the atmosphere. We measured variations in CO_2, NO_X and CO/HC gas concentration over the course of twelve hours. These values were compared to expected normal values that occur during daily cycles [146], [134].

Our measurements reveal that NO_X concentrations were very small and stable around 0.05ppm. Also, we found that CO_2 gas concentration varies during the day-night cycle, dropping in value from around 430ppm in the afternoon to 350ppm early in the morning.

CO and hydrocarbon gas emissions are almost at a constant value throughout the period, seeming to drop in the early hours of the morning. This appears to be correlated to the decrease in vehicle traffic during the night.

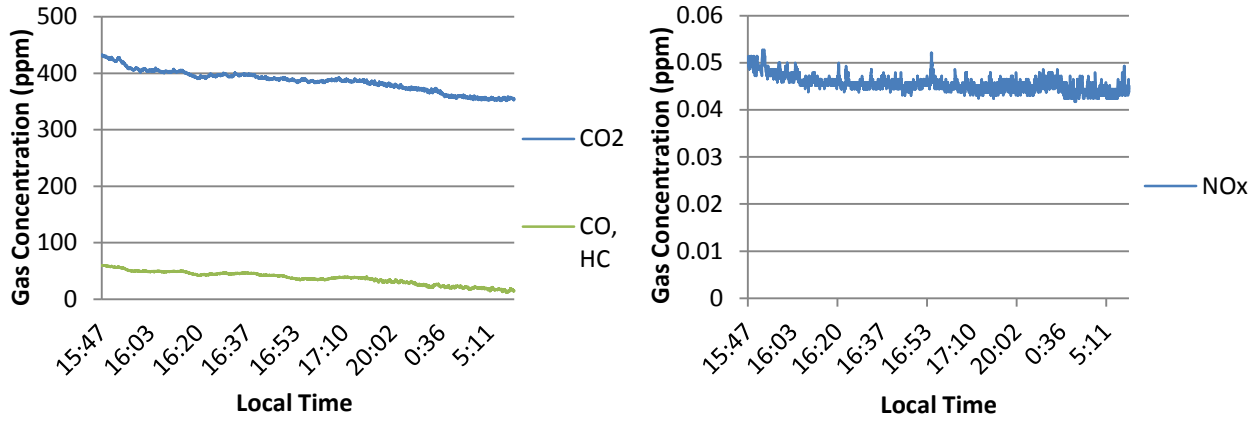


Figure 76. CO2 , NOx and CO/HC gas concentration variation for a fixed location

Mobile system testing in traffic was done by installing the Mobile Unit prototype on a privately owned car and driving around a predetermined path in the city of Bucharest. In order to view the changes in air pollution, measurement sessions were made on the same course but at different times of day, both during high traffic hours in the morning and afternoon and with little or no road traffic, late at night.

One example of higher than normal air pollutant concentration can be seen in Figure 75. The web page accurately shows an increase in pollution in ”Unirii” area, which is a square situated at the convergence of five of the city’s major road traffic arteries.

6.2.6 Conclusions

Our prototype implementation of the hardware and software proves that the concept is a viable solution for air pollution monitoring. It is not universally applicable, as it only measures pollution in the areas where people live (most of the times these are the most polluted areas). However, this is exactly its intended purpose: to provide relevant information about the areas people are actually interested in.

Many people will immediately benefit from our system: asthmatics, people concerned about the air quality, joggers, etc. On the long term, government agencies that regulate and impose pollution standards can benefit from the large amounts of data gathered by our system which can result in better statistics and understanding of the way pollutants affect the urban environment. It can also lead to better air quality management and to pinpoint major pollution sources inside of a city.

Due to its modular design, our system can be extended to offer additional functionality. For instance, multiple Mobile Units could be installed on public transportation buses and trams, offering an up-to-date and detailed picture of urban pollution. Furthermore, it could use the location tracking capabilities of the mobile telephony networks instead of the GPS system.

More complex logic could be incorporated in the server application, allowing the automated identification of problem areas and possibly the prediction of air pollution patterns and expansion, based on meteorological data.

Chapter 7

Conclusions

Although we always presented our research in a top-down manner, this thesis has been built from a large number of small steps and iterations. We needed to know how to address the issue of energy efficiency for wireless sensor networks and we did that by starting with a definition of what wireless sensor networks are, where they are employed and why they are needed.

We set about in defining what are the most important issues that pertain to power consumption in sensor networks and we saw that battery life is a crucial parameter along with how a sensor node uses the battery energy to power its components and accomplish its intended task. A survey of the most common types of circuits, controllers, transceivers and sensors that are employed in wireless sensor nodes has been made. We emphasized on the power consumption issues associated with each component and found that the greatest culprit for energy consumption is not processing, but radio communication. Designing a service that minimizes network chatter will automatically increase network lifetime.

We classified sensor network according to shared features such as deployment scale and location, attached infrastructures and lifetime constraints. Based on these properties, we categorized sensor networks into three major groups: Environmental Sensor Networks (ESN), Community Sensor Networks (CSN) and Body Sensor Networks (BSN), each with its own particularities and features.

We also focused on modeling sensors and actuator networks by classifying them into two main categories: the ones defined by standardization entities using XML or text-table values; and others using ontologies.

Energy harvesting was addressed as a solution to energy consumption and battery depletion in sensor nodes. We found that, if applied correctly and tailored to specific constraints it can serve as viable alternative power source for all classes of sensor networks. We applied a

mathematical model to energy harvesting in order to define and better understand it. We experimented with and assessed the efficiency of three types of energy harvesters, namely piezoelectric, thermal and photovoltaic. The latter proved to have a superior yield and, as a consequence, was used in our renewable energy harvesting circuitry.

We designed and built a new type of wireless sensor node which was based on the results we had with our energy harvesting experiments. We studied also energy storage systems at node level and found that batteries are not a good solution if aiming for total energy independence and extended operational lifetime. We found that supercapacitors offer a solution which is close to optimal from the standpoint of efficient charging and energy storage. Further, we employed this knowledge in the design of a sensor node which proved our theories regarding energy independence.

This thesis would have not been complete without delving into the software protocols, services and frameworks that give wireless sensor networks their purpose and application. We designed Check, a novel Management Framework which unifies and supplies different WSAN islands with services and extended functionality.

As an important part of our framework, we implemented a new type of task scheduling algorithm in which the main constraint is not task completion time, but energy consumption. We modeled and assessed the algorithm's complexity and behavior when deployed on a given set of nodes.

Another part of the Check framework is the centralized network monitoring, control and reconfiguration tool, which we implemented with the goals of scalable internetworking, horizontalization and heterogeneity.

In order to apply the knowledge accumulated during this research and to validate our work, we designed and successfully deployed a series of real-life applications, two of which are worth noting. The first one is a deployment in a residential environment and studies the components and services which a sensor network oriented on home automation needs to offer.

In the second application, we studied the deployment of mobile sensor nodes in an urban environment with the specific task of gathering data and relaying it to a central coordinator. The purpose of the project is to measure environmental sensor data, such as air pollution and contaminants linked to automotive exhaust and make it available to the general public via a intuitive web interface.

The work has been validated also by the European Commission as the three European projects where this software was developed were praised as a success by the evaluators. All the work described here was validated through publishing in international and national conferences and workshops.

Out of the many original contributions enumerated above which are brought to the field of wireless sensor networks, we like to enumerate the ones which we think are the most important:

- A classification of wireless sensor networks was proposed, according to energy consumption criteria, such as battery lifetime, node positioning and deployment scale.
- The applicability of energy harvesting to a wireless sensor mote was tested and analyzed. Four of the most promising energy sources were researched: vibration, thermal, solar and radiofrequency energy scavenging. Each of the four sources were analyzed in terms of the total amount of energy produced and their ability to ensure the continuous functionality of a sensor node. As a result of this study it was determined that solar energy offers the best results both in terms of the quantity of harvested energy and in the amount of instantaneous power the system can generate when subjected to full illumination.
- Energy storage mediums were researched from the point of view of their applicability to wireless sensor networks. Usual rechargeable batteries were found to be sub-optimal when used on a long-term service-free architecture as wireless sensor networks. Instead the emerging technology of super-capacitors was researched and their ability to store large amounts of charge over long periods, without any significant loss of performance over time.
- A wireless sensor network infrastructure named Sparrow was built to test the research concepts of energy harvesting and to serve as the backbone of the software protocols and frameworks we developed. It proved that energy independence and sustained long term operation are possible when employing renewable energy and non-standard energy storage.
- Two different types of wireless sensor nodes were designed and built to serve as a backbone for our wireless sensor network. The nodes implement the concepts of energy harvesting studied before and, due to their dedicated power supply circuits, are suited to be powered from a wide range of energy sources.
- A mathematical model for energy consumption estimation in multi-hop wireless sensor networks was proposed. The model takes into consideration the topography of the network and the radio environment disturbances in order for a node to tune its radio transceiver energy consumption to optimum levels.
- A novel and unconventional scheduling algorithm was developed as part of our Check Management Framework in which the main constraint is not time, but energy. As sensor networks are rarely subjected to hard deadlines, a more elegant approach is to design a scheduling algorithm that prioritizes energy consumption and task affinity. Results showed that tasks are scheduled on an energy-optimal basis.
- An innovative method for monitoring and controlling WSN nodes from a graphical interface over an Internet connection was proposed, using the successful MonALISA framework. Data is gathered from the network and stored in an Internet-based repository, from where it can be read remotely using a graphical

client program. The interface between the WSN and the Internet services contains an abstraction layer, allowing uniform access to nodes built using various technologies and running different software and protocols.

- Our algorithms and hardware were tested and successfully deployed in real-life applications in order to validate our assumptions. In the first phase, laboratory tests were conducted and the results were carefully analyzed and integrated into future revisions of the hardware and software platforms. After this phase, the wireless sensor network platform was tested and two different application spaces were selected, the large-scale urban environment and the smaller scale home automation environment.
- A new wireless sensor network architecture especially designed for the task of home automation was built and tested. The 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 this system are its scalability, low power, self sufficiency and versatility.
- A mobile system for air quality and pollution measurement suited for the urban environment was developed. The system is based on a reliable measurement device that can acquire information about the air quality of its surroundings, store it in a temporary memory buffer and periodically relay it to a central on-line repository. Real-time gathered data can be freely accessed by the public through an on-line web interface. Users can select and view different gases and concentrations overlapped on a map of the city.

7.1 Future Work

One of the future research goals for this project is testing the network in real life conditions over long periods of time. For this, the nodes that are equipped with energy scavenging capabilities will be deployed in a remote area (i.e. forest, urban area, building) and performance measurements will be taken. Different power management schemes and algorithms can also be deployed to aid the total system up-time.

Another important research question that has not been properly addressed in this study is that of network behavior to node failure. Because nodes are subjected to different types of failure (i.e. mechanical, electrical, lack of sufficient energy, software malfunctions), data gathered from their sensors may be lost. Proper self-healing algorithms must be deployed on the WSN in order for it to automatically detect a node failure and, if possible, take corrective actions to avoid a decrease in performance.

Also, an important research topic in which research can be further developed is network security. As wireless sensor networks are installed in human environments and due to the fact that sensors typically communicate over the air, there is always a certain danger that information can be accessed and modified by unauthorized parties. The research question is how can we provide a platform for an efficient, secure and reliable integration of sensor networks into large scale industrial environments.

As pervasive computing applications in wireless sensor networks are just beginning to be implemented on a commercial scale, we believe that there are still many research topics to be approached and many interesting questions to be discovered. The best, as always, are the ones we do not yet have an answer for.

Published Papers

Razvan Tataroiu, **Dan Tudose**, Remote Monitoring and Control of Wireless Sensor Networks, 17th International Conference of Control Systems and Computer Science – CSCS17. Vol. 1, pp. 187-192, Bucharest, Romania, May 2009.

Andrei Voinescu, **Dan Tudose**, Nicolae Tapus, Task Scheduling in Wireless Sensor Networks, Sixth International Conference on Networking and Services, Cancun, Mexico, March 7-13 2010 DOI : 10.1109/ICNS.2010.10

Dan Tudose,Traian Alexandru Patrascu, Andrei Voinescu, Razvan Tataroiu, Nicolae Tapus, Mobile Sensors in Air Pollution Measurement, 8th Workshop on Positioning, Navigation and Communication 2011 (WPNC'11), Dresden, Germany, April 07-08, 2011.

Dan Tudose, Nicolae Tapus, Energy Harvesting and Power Management in Wireless Sensor Networks, 18th International Conference of Control Systems and Computer Science – CSCS18. Vol. 1, pp. 174-180, Bucharest, Romania, May 2011.

Dan Tudose, Andrei Voinescu, Madi-Tatiana Petrareanu, Andrei Bucur, Dumitrel Loghin, Adrian Bostan, Nicolae Tapus, "Home Automation Design Using 6LoWPAN Wireless Sensor Networks", International Conference on Distributed Computing in Sensor Systems and Workshops (DCOSS), Barcelona, Spain, June 2011.

Posters

Dan Tudose,Traian Alexandru Patrascu, Andrei Voinescu, Razvan Tataroiu, Nicolae Tapus, Mobile Sensors in Air Pollution Measurement, 8th Workshop on Positioning, Navigation and Communication 2011 (WPNC'11), Dresden, Germany, April 07-08, 2011.

Datasheets

Dan Tudose, Sparrow v2 Wireless Sensor Node Speciffication Sheet, 2011, [Online] http://elf.cs.pub.ro/pm/wiki/media/sparrowv2.pdf.

Bibliography

1. **Weiser, M.** *The Computer for the 21st Century.* 1991, Scientific American, No. Communications, Computers, and Network.
2. **IEEE802.15.4.** *Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs).* s.l. : 802.15.4 IEEE Standard for Information technology, 2006.
3. **ZigBee.** ZigBee Alliance. [Online] http://www.zigbee.org.
4. **Berringer, K.** *High-performance Mixed-signal MCUs in Low- power Applications.* s.l. : EPN, 2008. p. 12. Vol. 7.
5. **M.V. Marcos Augusto, Diogenes Cecilio, M. Jose.** Survey on Wireless Sensor Network Device. [Online] 3 15, 2009. http://perso.ens-lyon.fr/isabelle.guerin-lassous/Enseignement/survey-WSN-devices.pdf.
6. **Atmel Corporation Datasheets.** www.atmel.com/products/. [Online] 09 1, 2011.
7. **Bokareva, Tatiana.** Mini Hardware Survey. [Online] 09 1, 2011. http://www.cse.unsw.edu.au/~sensar/hardware/hardware_survey.html.
8. **Texas Instruments Datasheets.** [Online] 09 1, 2011. www.ti.com.
9. **J. Polastre, R. Szewczyk, D. Culler.** *Telos: Enabling Ultra-Low Power Wireless Research.* Los Angeles : s.n., 2005. Proceedings of IPSN/SPOTS.
10. **Pistoia, F.** *Battery Operated Devices and Systems: From Portable Electronics to Industrial Products.* Amsterdam : Elsevier, 2008.
11. **Crompton, T.R.** *Battery Reference Book 3-ed.* Oxford : Newnes, 2000.
12. **Jens, Eliasson.** *Low-Power Design Methodologies for Embedded Internet Systems.* Sweden : EISLAB Lulea University of Technology, 2008.
13. **V. Raghunathan, C. Schurgers, S. Park, M. Srivastan.** *Energy-aware Wireless Microsensor Networks.* 2002, IEEE Signal Processing Magazine Vol. 40, No. 3, pp. 40-50.
14. **Engineering, Omega.** *Complete Measurement, Control and Automation Handbook and Encyclopedia.* 2008.
15. **Adam Dunkels, Björn Grönvall, and Thiemo Voigt.** *Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors.* 2004. IEEE Emnets.
16. **ZigBee, Specification.** 2005.
17. **N. Kushalnagar, G. Montenegro, C. Schumacher.** *IPv6 over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals.* s.l. : RFC 4919, 2007.
18. —. *Transmission of IPv6 Packets over IEEE 802.15.4 Networks.* s.l. : IETF, 2007.
19. **Arch Rock Corp.** IP Based Wireless Sensor Networking: Secure, Reliable, Low Power IP Connectivity for 802.15.4 Networks. [Online] 2007. http://www.cs.berkeley.edu/~jwhui/6lowpan/Arch_Rock_Whitepaper_IP_WSNs.pdf.
20. **Crawford, M.** *Transmission of IPv6 Packets over Ethernet Networks.* s.l. : IETF, 1998.
21. **NIST.** National Institute of Standards and Technology Website. [Online] http://ieee1451.nist.gov/.

22. **ANSI, N42.42.** *American National Standard Data Format Standard for Radiation Detectors Used For Homeland Security.* s.l. : ANSI, 2006.
23. **Joint Program Executive Office for Chemical and Biological Defense.** *Common Chemical, Biological, Radiological, Nuclear (CBRN) Sensor Interface (CCSI).* Falls Church : s.n., 2008.
24. **SensorML.** OGC® OpenGIS® Sensor Model Language (SensorML) Implementation Specification. Version 1.0.0. OGC Reference number 07-000. [Online] 2007. http://www.opengeospatial.org/projects/groups/sensorweb.
25. **TransducerML.** OGC®. OpenGIS® Transducer Markup Language (TML) Implementation Specification. Version 1.0.0. OGC Reference number 06-010r6. . [Online] 2007. http://www.opengeospatial.org/projects/groups/sensorweb.
26. **ObservationML1.** OGC®. OpenGIS® Observations and Measurements –Part 1 – Observation schema. Version 1.0. OGC Reference number 07-022r1. [Online] 2007. http://www.opengeospatial.org/projects/groups/sensorweb.
27. **ObservationML2.** OGC®. OpenGIS® Observations and Measurements –Part 2 – Sampling Features. Version 1.0. OGC Reference number 07-022r3. [Online] 2007. http://www.opengeospatial.org/projects/groups/sensorweb.
28. **Dan Tudose, Nicolae Tapus.** *Energy Harvesting and Power Management in Wireless Sensor Networks.* Bucharest : 18th International Conference of Control Systems and Computer Science – CSCS18, 2011.
29. **Sinha, A. and Chandrakasan, A.** *Dynamic power management in wireless sensor networks.* 2001. IEEE Design and Test of Computers 18. pp. 62–74.
30. **Min, R., Bhardwaj, M., Cho, S., Sinha, A., Shih, E., Wang, A., and Chandrakasan, A.** *An architecture for a power-aware distributed microsensor node. .* 2000. IEEE Workshop on Signal Processing Systems (SiPS ’00).
31. **Singh, S., Woo, M., AND Raghavendra, C. S.** *Power-aware routing in mobile ad hoc networks.* 1998. ACM/IEEE Mobicom.
32. **Younis, M., Youssef, M., AND Arisha, K.** *Energy-aware routing in cluster-based sensor networks.* 2002. Proc. 10th IEEE/ACM International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems.
33. **Shah, R. C. AND Rabaey, J. M.** *Energy aware routing for low energy ad hoc sensor networks.* Orlando, FL : s.n., 2002. Proc. IEEE Wireless Communications and Networking Conference (WCNC). pp. 350–355.
34. **Li, Q., Aslam, J., and Rus, D.** *Online power aware routing in wireless ad-hoc networks.* Rome, Itay : s.n., 2001. ACM SIGMOBILE.
35. **Paradiso, J. A. and Feldmeier, M.** *A compact, wireless, self-powered pushbutton controller.* Atlanta, GA : Springer-Verlag Berlin Heidelberg, 2001. ACM Ubicomp. pp. 299–304.

36. **Aman Kansal, Jason Hsu, Sadaf Zahedi, and Mani B. Srivastava.** *Power management in energy harvesting sensor networks.* s.l. : ACM Trans. Embed. Comput. Syst. 6, 4, Article 32, 2007.
37. **THK.** Regulation On Collective Frequencies For Certain Radio Transmitters And Their Use. [Online] August 2006. http://www.ictregulationtoolkit.org/en/Publication.2753.html.
38. **25, ERC REPORT.** Frequency Range 29.7 MHz To 105 GHz And Associated European Table Of Frequency Allocations And Utilisations. [Online] February 1998. http://www.ero.dk/doc98/official/pdf/Rep025.pdf.
39. **Guoqiang Mao, Brian D. O. Anderson, and B. Fidan.** *Path loss exponent estimation for wireless sensor network localization.* s.l. : Comput. Netw. 51, 10, 2007. pp. 2467-2483.
40. **Callaway, B. O. Priscilla Chen and E.** Energy Efficient System Design with Optimum Transmission Range for Wireless Ad-Hoc Networks,. *Proceedings of ICC.* 2002, Vol. volume 2, pp. 945.952.
41. **Ettus, M.** *System Capacity, Latency, and Power Consumption in Multihop-routed SS-CDMA Wireless Networks.* s.l. : Radio and Wireless Conference (RAWCON '98), 1998. pp. pp. 55-58.
42. **Volkan, T. Meng and R.** *Distributed Network Protocols for Wireless Communication.* s.l. : Proceedings of IEEEE ISCAS, 1998.
43. **Shepard, T.** *A Channel Access Scheme for Large Dense Packet Radio Networks.* s.l. : Proceedings of ACM SIGCOMM, 1996. pp. pages 219–230.
44. **S. Singh, M.Woo, and C. Raghavendra.** *Power-Aware Routing in Mobile Ad Hoc Networks.* s.l. : Proceedings of the Fourth Annual ACM/IEEE International Conference on Mobile Computing and Networking (MobiCom '98), 1998.
45. **Roundy, S., & Wright, P. K., & Pister, K. S.** *Micro-electrostatic vibration-to-electricity converters.* s.l. : Proceedings of the ASME International Mechanical Engineering Congress and Expo., 2002.
46. **Stevens, J.** *Optimized thermal design of small thermoelectric generators.* s.l. : Proceedings of 34th Intersociety Energy Conversion Eng. Conference. Society of Automotive Engineers, 1999.
47. **Yeatman, E.M.** *Advances in power sources for wireless sensor nodes.* London : Proceedings of International Workshop on Wearable and Implantable Body Sensor Networks, 2004.
48. **Roundy, S., Steingart, D., Fréchette, L., Wright, P. K., & Rabaey, J.** *Power sources for wireless networks.* Berlin, Germany : Proceedings of 1st European Workshop on Wireless Sensor Networks (EWSN '04), 2004.
49. **Mitcheson, P. D., Green, T. C., Yeatman, E. M., & Holmes, A. S.** *Analysis of optimized micro-generator architectures for self-powered ubiquitous computers. .* London : Imperial College, 2004.
50. **Holmes, A. S.** *Axial-flow microturbine with electromagnetic generator: Design, CFD simulation, and prototype demonstration.* s.l. : Proceedings of 17th IEEE International Micro Electro Mechanical Systems Conf. (MEMS 04), IEEE Press, 2004.

51. **Paradiso, J., & Feldmeier, M.** *A compact, wireless, self-powered pushbutton controller.* s.l. : ubicomp: Ubiquitous Computing., 2001.
52. **Shenck, N. S., Paradiso, J. A.** *Energy scavenging with shoe-mounted piezoelectrics, I.* s.l. : IEEE Micro, 2001.
53. **Starner, T., & Paradiso, J. A.** *Human-generated power for mobile electronics.* New York, NY : In C. Piguet (Ed). Low-power electronics design, 2004.
54. **Yaglioglu, O.** *Modeling and design considerations for a micro-hydraulic piezoelectric power generator. Master's thesis.* s.l. : Department of Electrical Eng. and Computer Science, MIT, 2002.
55. **Rabaey, J. M., Ammer, M. J., Da Silva Jr, J. L., Patel, D., & Roundy, S.** *Picoradio supports ad hoc ultra-low power wireless networking.* s.l. : IEEE Computer, 2000.
56. **V. Raghunathan, S. Ganerival, M. Srivastava.** *Emerging Techniques for Long Lived Wireless Sensor Networks.* 2006. IEEE Communication Magazine Vol. 44, No. 4. pp. 108 – 114.
57. **C. Alippi, G. Anastasi, M. Di Francesco, M. Roveri.** *Energy Management in Wireless Sensor Networks with Energy-hugry Sensors.* 2009. IEEE Instrumentation & Measurement Magazine, Vol. 12, No. 2. pp. 16-23.
58. **R. Want, K. Farkas, Marayanaswami.** *Energy Harvesting and Conservation, Guest Editor's Introduction.* 2005. IEEE Pervasive Computing; Mobile and Ubiquitous Systems, Vol. 4, No. 1. pp. 14-17.
59. **S. Roundy, P.K. Wright, and J.M. Rabaey.** *Energy Scavenging for Wireless Sensor Networks with Special Focus on Vibrations.* Norwell, MA : Kluwer Academic Publishers, 2004.
60. **Youfan Hu, Chen Xu, Yan Zhang, Long Lin, Robert L. Snyder, and Zhong Lin Wang.** A Nanogenerator for Energy Harvesting from a Rotating Tire and its Application as a Self-Powered Pressure/Speed Sensor. 2011, Vol. Adv. Matter.
61. **Wang, Zetang Li and Zhong Lin.** Air/Liquid-Pressure and Heartbeat-Driven Flexible Fiber Nanogenerators as Micro/Nano-Power Source or Diagnostic Sensors . Adv. Mater., 2010.
62. **Nathan S. Shenck and Joseph A. Paradiso.** *Energy Scavenging with Shoe-Mounted Piezoelectrics.* s.l. : IEEE, May 2001, Vols. IEEE Micro 21, 3, pp. 30-42.
63. **M Wischke, M Masur, M Kröner and P Woias.** *Vibration harvesting in traffic tunnels to power wireless sensor nodes.* 2011, Vol. Smart Mater. Struct. 20.
64. **Lewandowski, B., Kilgore, K., Gustafson, K.** *Design Considerations for an Implantable, Muscle Powered Piezoelectric System for Generating Electrical Power.* 5, s.l. : Springer Netherlands, 2007, Vol. Annals of Biomedical Engineering 34.
65. **Nye, J.F.** *Physical properties of crystals: their representation by tensors and matrices.* s.l. : Clarendon Press, 1985.
66. **Blevins, R. D.** *Formulas for Natural Frerquency and Mode Shape.* New York : Robert E. Krieger Publishing Company, 1979.
67. **Stojmenovic, Ivan.** *Handbook of Sensor Networks - Algorithms and Architectures.* s.l. : John Wiley & Sons., 2005.

68. **Piezoelectric Energy Harvesting Power Supply**. *Linear Technology Website.* [Online] http://cds.linear.com/docs/Datasheet/35881fa.pdf.
69. **Henry A. Sodano, Garnett E. Simmers, Remi Dereux, Daniel J. Inman.** *Recharging Batteries using Energy Harvested from Thermal Gradients.* 1, January 2007, Vol. Journal of Intelligent Material Systems and Structures vol. 18.
70. **Slack Glen A., Hussain Moayyed A.** *The maximum possible conversion efficiency of silicon-germanium thermoelectric generators.* 5, s.l. : Journal of Applied Physics, 1991, Vol. 70.
71. **Berland, B.** *Photovoltaic Technologies Beyond the Horizon: Optical Rectenna Solar Cell.* Golden, Colorado : National Renewable Energy Laboratory, 2002.
72. **Brown, W. C.** *The history of power transmission by radio waves.* s.l. : IEEE Transaction on Microwave Theory and Techniques, 1984, Vols. Vol. 32, No. 9.
73. **McSpadden, J. O. and K. Chang.** *A dual polaried circular patch rectifying antenna at 2.45 GHz for microwave power conversion and detection.* s.l. : IEEE MTT-S, Microwave Symp., Dig., 1994.
74. **Ren, Y. J. and K. Chang.** *5.8 GHz circular polarized dual-diode rectenna and rectenna array for microwave power transmission.* s.l. : IEEE Transaction on Microwave Theory and Techniques, 2006, Vols. Vol. 54, No. 4.
75. **Hagerty, J. A., F. B. Helmbrecht, W. H. McCalpin, R. Zane, and Z. B. Popovic.** *Recycling ambient microwave energy with broad-band rectenna arrays.* s.l. : IEEE Transaction on Microwave Theory and Techniques, 2004, Vols. Vol. 52, No. 3.
76. **Akkermans, J. A. G., M. C. van Beurden, G. J. N. Doodeman, and H. J. Visser.***Analytical models for low-power rectenna design.* s.l. : IEEE Antennas and Wireless Propagation Letters, 2005.
77. **Casas, R. Casas and O.** Battery Sensing for Energy-Aware System Design . *Computer.* 2005, Vol. vol. 38, pp. 48–54.
78. **V. Raghunathan, S. Ganeriwal, and M. Srivastava.** *Emerging Techniques for Long Lived Wireless Sensor Networks.* 4, s.l. : IEEE, 2006, Communications Magazine, Vol. 44, pp. pp. 108–114.
79. **V., Raghunathan, et al.** *Design Considerations for Solar Energy Harvesting Wireless Embedded Systems.* s.l. : IEEE International Conference on Information Processing in Sensor Networks , 2005.
80. **Roundy, S., et al.** *Power Sources for Wireless Sensor Network.* Berlin, Germany : Proc. 1st European Workshop on Wireless Sensor Networks (EWSN), 2004.
81. **Park, C, Liu, J. and Chou, P.H.** *Eco: An ultra-compact low-power wireless sensor node for real-time motion monitoring.* Los Angeles : IEEE International Conference on Information Processing in Sensor Networks (IPSN), 2005.
82. **Hande, A., et al.** *Indoor Solar Energy Harvesting for Sensor Network Router Nodes.* s.l. : Journal of Microprocessors and Microsystems Special Issue on Sensor Systems, 2006.

83. **Chou, Pai H. and Park, C.** *Energy-Efficient Platform Designs for Real-World Wireless Sensing Applications.* San Jose, CA : IEEE International Conference on Computer Aided Design, (ICCAD), 2005.
84. **Alberola, J., et al.** *Solar Inexhaustible Power Source for Wireless Sensor Node.* Victoria-Canada : IEEE International Instrumentation and Measurement technology Conference, I2MTC, 2008.
85. **Arms, S.W., et al.** *Power Management for Energy Harvesting Wireless Sensors.* San Diego, CA. : International Symposium on Smart Structures & Smart Materials (SPIE), 2005.
86. **Rahimi, M., et al.** *Studying the Feasibility of Energy Harvesting in a Mobile Sensor Network.* Taipei, Taiwan : Proc. IEEE International Conference on Robotics and Automation (ICRA), 2003.
87. **Hsu, J., et al.** *Heliomote: Enabling self-sustained wireless sensor networks through solar energy harvesting.* San Diego, California : International Symposium on Low Power Electronics and Design (ISLPED), 2005.
88. **Tudose, Dan.** Sparrow v2 specifications. [Online] 2011. http://elf.cs.pub.ro/pm/wiki/media/sparrowv2.pdf.
89. **R., de Levie.** *Advances in Electrochemistry and Electrochemical Engineering.* 1967.
90. **Zubieta L, Bonert R, Dawson F** *Considerations in the design of energy storage systems using double-layer capacitors.***.** s.l. : IPEC Tokyo, 2000.
91. **Dougal RA, Gao L, Liu S.** *Ultracapacitor model with automatic order selection and capacity for dynamic system simulation.* s.l. : Journal of Power Sources , 2004.
92. **Belhachemi F, Raël S, Davat B.** *A physical based model of power electric double-layer supercapacitors.* Roma : IEEE-IAS'00, 2000 .
93. **Bonert, L. Zubieta and R.** *Characterization of Double-Layer Capacitors (DLCs) for Power Electronics Applications.* s.l. : Industry Applications Conference, 1998. Thirty-Third IAS Annual Meeting, 1998.
94. **Dresden Frauenkirche.** *Wikipedia page on.* [Online] 2011. http://en.wikipedia.org/wiki/Dresden_Frauenkirche.
95. **Sensirion.** SHT21 Sensor Datasheet. [Online] http://www.sensirion.com/en/pdf/product_information/Datasheet-humidity-sensor-SHT21.pdf.
96. **Company, Zigpos.** Zigpos. [Online] http://www.zigpos.com/.
97. **SENSEI.** SENSEI - Integrating the Physical with the Digital World of the Network of the Future. [Online] www.sensei-project.eu.
98. **R. Tătăroiu, D. Tudose.** *Remote Monitoring and Control of Wireless Sensor Networks.* Bucharest, Romania : s.n., 2009. 17th International Conference of Control Systems and Computer Science – CSCS17. Vol. 1. pp. 187-192.
99. **R. Muller, G. Alonso, D. Kossmann.** *SwissQM: Next Generation Data Processing in Sensor Networks.* 2007. Proceedings of the 3rd Biennial Conference on Innovative Data Systems Research.
100. **Spot, Sun.** Sun Spot. [Online] http://www.sunspotworld.com.

101. **Yang, S.** Redwoods go high tech: Researchers use wireless sensors to study California's state tree. [Online] 2003. http://www.berkeley.edu/news/media/releases/2003/07/28_redwood.shtml,.
102. **Lanthaler, M.** *Self-Healing Wireless Sensor Networks.*
103. **Trehan A., J. Saia.** *Picking up the pieces: Self-healing in reconfigurable networks.* 2008. IEEE International Parallel & Distributed Processing Symposium.
104. **T. Hayes, N. Rustagi, J. Saia, and A. Trehan.** The forgiving tree: A self-healing distributed data structure. [Online] http://www.cs.unm.edu/ amitabh/pubs/ForgivingTreeNV19-525P.pdf.
105. **T. Bokareva, N. Bulusu, S. Jha.** *SASHA: Toward a Self-Healing Hybrid Sensor Network Architecture.*
106. **M. Rubenstein, W-M. Shen.** *A scalable and distributed model for self-organization and self-healing.* Estoril Portugal : s.n., 2008. AAMAS '08: Proceedings of the 7th International joint Conference on Autonomous Agents and Multiagent Systems. pp. 1179—1182.
107. **L. Gheorghe, R. Rughinis, N. Tapus.** *Fault-Tolerant Flooding Time Synchronization Protocol for Wireless Sensor Networks.* Cancun, Mexico : s.n., 2010. Sixth International Conference on Networking and Services.
108. **Lombriser C., Roggen D., Stäger M. and Tröster G (2007).** *Titan: A Tiny Task Network for Dynamically Reconfigurable Heterogeneous Sensor Networks.* Berlin : Springer, 2007. Kommunikation in Verteilten Systemen (KiVS). pp. 127-138.
109. **Dunkels A., Grönvall B, Voigt T.** *Contiki - a Lightweight and Flexible Operating System for Tiny Networked Sensors.* 2004. IEEE Local Computer Networks. pp. 455-462.
110. **Perillo, M., Heinzelman, W.** *Optimal sensor management under energy and reliability constraints.* New Orleans : s.n., 2003. Proc. of the IEEE WCNC2003.
111. **Andrei Voinescu, Dan Tudose, Nicolae Tapus.** *Task Scheduling in Wireless Sensor Networks.* Cancun, Mexico : Sixth International Conference on Networking and Services, 2010.
112. **Y. Tian, E. Ekici, and F. Ozguner.** *Energy-constrained task mapping and scheduling in wireless sensor networks.* 2007.
113. **Janecek, T. Hagras and J.** *A high performance, low complexity algorithm for compile-time task scheduling in heterogeneous systems.* 2005. Parallel Computing, vol. 31, no. 7. pp. 653–670.
114. **F. C. Delicato, F. Protti, J. F. de Rezende, L. F. R. da Costa Carmo, and L. Pirmez.** *Application-driven node management in multihop wireless sensor networks.* ICN (1), ser. Lecture Notes in Computer Science vol. 3420.
115. **A. Voinescu, D. Tudose, N. Tapus.** *Task Scheduling in Wireless Sensor Networks.* Cancun, Mexico : s.n., 2010. Sixth International Conference on Networking and Services.
116. **Jacokes, B.** Lecture notes on multiway cuts and k-cuts. [Online] 2006. math.mit.edu/_goemans/18434S06/multicuts-brian.pdf.
117. **Guha, C. Chekuri and S.** *The steiner k-cut problem.* 2006. SIAM J. Discret. Math., vol. 20, no. 1. pp. 261–271.

118. **Checkuri, C.** Approximation algorithms: Lecture on multiway cut problem. [Online] 2009. www.cs.illinois.edu/class/sp09/cs598csc/Lectures/lecture 7.pdf.
119. **C. Grigoras, A. Costan, C. Cirstoiu, I. Legrand, V. Cristea.** *Monitoring and automating actions in the Alice grid with the MonAlisa Repository.* Bucharest, Romania : s.n., 2008. CSCS16.
120. **C.Cirstoiu, C.Grigoras, M.Toarta, C. Dobre, R.Voicu, I.C.Legrand, H.B.Newman.** *An Agent based, Dynamic Service System to Monitor, Control and Optimize Grid based Applications.* Interlaken, Switzerland : s.n., 2004. CHEP 2004.
121. **Razvan Tataroiu, Dan Tudose.** *Remote Monitoring and Control of Wireless Sensor Networks.* Bucharest : 17th International Conference of Control Systems and Computer Science – CSCS17, 2009.
122. **Dan Stefan Tudose, Nicolae Tapus, Andrei Voinescu, Madi-Tatiana Petrareanu, Andrei Bucur, Dumitrel Loghin, Adrian Bostan.** *Home Automation Design Using 6LoWPAN Wireless Sensor Networks.* Barcelona : 1st HOBNET Workshop on IPv6 Sensor Networking for Smart/Green Buildings, 2011.
123. **Bluetooth.** Bluetooth Consortium Official Website. [Online] 2002. https://www.bluetooth.org.
124. **Jianping Song, et al.** *Wirelesshart: Applying wireless technology in real-time industrial process control.* 2008, IEEE Real-Time and Embedded Technology and Applications Symposium.
125. **Mulligan, Geoff.** *The 6lowpan architecture.* 2007, EmNets'07: Proceedings of the 4th workshop on Embedded networked sensors. ACM.
126. **L. Schor, P. Sommer, and R. Wattenhofer.** *Towards a zero-configuration wireless sensor network architecture for smart buildings.* s.l. : ACM, 2009. Proceedings of the First ACM Workshop on Embedded Sensing Systems for Energy-Efficiency in Buildings.
127. **M. Kovatsch, M. Weiss, D. Guinard.** *Embedding Internet Technology for Home Automation.* Bilbao, Spain : s.n., 2010. Proceedings of the 15th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA 2010).
128. **H. B. Newman, et. al.** *Monalisa: A distributed monitoring service architecture.* Geneva : Technical report, CERN, 2003.
129. **Dan Tudose, Traian Alexandru Patrascu, Andrei Voinescu, Razvan Tataroiu, Nicolae Tapus.** *Mobile Sensors in Air Pollution Measurement.* Dresden : 8th Workshop on Positioning, Navigation and Communication 2011 (WPNC'11), 2011.
130. **Egmond, ND Van.** *Historical perspective and future outlook. Studies in Environmental Science.* 1998.
131. **BBC.** [Online] 2005. http://news.bbc.co.uk/2/hi/health/4283295.stm.
132. **Holgate S, Samet JM, Maynard RL, et al.** *Air pollution and health.* New York : NY: Academic Press, Inc, 1999.
133. **Sudantha, N. Kularatna and BH.** *An environmental air pollution monitoring system based on the IEEE 1451 standard for low cost requirements.* 2008. IEEE Sensors Journal. pp. 415–422.

134. **F. Tsow, E. Forzani, A. Rai, R. Wang, R. Tsui, S. Mastroianni, C. Knobbe, A.J. Gandolfi, and NJ Tao.** *A wearable and wireless sensor system for real-time monitoring of toxic environmental volatile organic compounds.* 2009. Sensors Journal, IEEE.
135. **Y.J. Jung, Y.K. Lee, D.G. Lee, K.H. Ryu, and S. Nittel.** *Air Pollution Monitoring System based on Geosensor Network.* 2009. Geoscience and Remote Sensing Symposium, 2008. IGARSS 2008. IEEE International, volume 3.
136. **Berger, H. H. Schrenk and L. B.** *Composition of Diesel Engine Exhaust Gas.* 1941, American Journal of Public Health and the Nation's Health. , pp. 669–681.
137. **P.F. Nelson, S.M. Quigley,** *The hydrocarbon composition of exhaust emitted from gasoline fuelled vehicles,.* 1984, Atmospheric Environment, Volume 18, Issue 1, pp. pp. 79-87.
138. **Oxford, Physical and Theoretical Chemistry Laboratory.** Safety data for carbon monoxide. [Online] 2011. http://msds.chem.ox.ac.uk/CA/carbon_monoxide.html.
139. **ATSDR, Agency for Toxic Substances and Disease Registry.** Medical Management Guidelines for Ammonia. [Online] 2011. http://www.atsdr.cdc.gov/MHMI/mmg126.pdf.
140. **NYSDH, New York State Department of Health.** Hydrogen Sulfide Chemical Information Sheet. [Online] 2011. http://www.health.ny.gov/environmental/chemicals/hydrogen_sulfide/docs/sulfide.pdf.
141. **McClellan, R. O.** *Health Effects of Exposure to Diesel Exhaust Particles.* 1987. Annual Review of Pharmacology and Toxicology vol.27. pp. 279-300.
142. **Jonathan A. Bernstein, Neil Alexis, Charles Barnes, I. Leonard Bernstein, Jonathan A. Bernstein, Andre Nel, David Peden, David Diaz-Sanchez, Susan M. Tarlo, P. Brock Williams.** *Health effects of air pollution.* 2004, he Journal of allergy and clinical immunology (volume 114 issue 5), pp. pp. 1116-1123.
143. **Samet JM, Marbury MC, Spengler JD.** *Health effects and sources of indoor air pollution. Part I.* 1987. American Review of Respiratory Diseases.
144. **Figaro.** Figaro Thick Film Metal Oxide Semiconductor Gas Sensors. [Online] 2011. http://www.figarosensor.com/.
145. **Poluare.** Online Pollution Monitoring Website. [Online] 2011. http://www.poluare.com/.
146. **F. Massen, A. Kies , N. Harpes et al.** Seasonal and Diurnal CO2 Patterns at Diekirch, Luxemburg. [Online] 2011. http://meteo.lcd.lu/papers/co2_patterns/co2_patterns.html.
147. **N. Guarino, M. Carrara, and P. Giaretta.** *An Ontology of Meta-Level Categories.* San Mateo, CA : Morgan Kaufmann, 1994. Proceedings of the Fourth International Conference (KR94).
148. **McGuinness, N. F. Noy and D. L.** *Ontology Development 101: A Guide to Creating Your First Ontology. .* Stanford Knowledge Systems Laboratory Technical Report KSL-01-05 and Stanford Medical Informatics Technical Report SMI-2001-0880.
149. **S. Avancha, C. Patel, and A. Joshi.** *Ontology-Driven Adaptive Sensor Networks,.* 2004. Proceedings of the First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (MobiQuitous'04).

150. **R. Jurdak, C. V. Lopes, and P. Baldi.** *A Framework for Modeling Sensor Networks.* Vancouver, Canada : s.n., 2004. Proceedings of the Building Software for Pervasive Computing Workshop at OOPSLA'04.
151. **M. Eid, R. Liscano, A. E. Saddik.** *A Novel Ontology for Sensor Networks Data.* La Coruna - Spain : s.n., 2006. Proceedings of 2006 IEEE International Conference on Computational Intelligence for Measurement Systems and Application.
152. **Protégé.** Protégé. [Online] http://protege.stanford.edu/.
153. **M. Eid, R. Liscano, A. E. Saddik.** *A Universal Ontology for Sensor Networks Data.* Ostuni- Italy : s.n., 2007. Proceedings of 2007 IEEE International Conference on Computational Intelligence for Measurement Systems and Applications.
154. **Pease, I. Niles and A.** *Origins of the Standard Upper Merged Ontology: A Proposal for the IEEE Standard Upper Ontology.* Seattle : s.n., 2001. Working Notes of the IJCAI- 2001 Workshop on the IEEE Standard Upper Ontology.
155. **D.J. Russomanno, C. Kothari and O. Thomas.** *Building a Sensor Ontology: A Practical Approach Leveraging ISO and OGC Models.* Las Vegas : s.n., 2005. The 2005 International Conference on Artificial Intelligence. pp. 637-643.
156. **G. V. Cybenko, G. Jiang, and W. Chung.** *Semantic Agent Technologies for Tactical Sensor Networks.* Orlando, FL : s.n., 2003. Proceedings of the SPIE Conference on Unattended Ground Sensor Technologies and Applications V. pp. 311-320.
157. **Zhao, J. Liu and F.** *Towards Semantic Services for Sensor-Rich Information Systems.* Boston, MA : s.n., 2005. Proceedings of the 2nd IEEE/CreateNet International Workshop on Broadband Advanced Sensor Networks (Basenets 2005).
158. **Roundy, S., Wright, P. K., and Rabaey, J. M.** *Energy Scavenging for Wireless Sensor Networks with Special Focus on Vibrations.* s.l. : Kluwer Academic, 2004.
159. **Cruz, R. L.** *A calculus for network delay, part I: Network elements in isolation.* 1991. IEEE Transactions on Information Theory 37. pp. 114 – 131.
160. **Parekh, A. K. and Gallager, R. G.** *A generalized processor sharing approach to flow control in integrated services networks: the single-node case.* 1993. IEEE/ACM Transactions on Networking (TON). pp. 344–357.
161. **Kansal, A., Potter, D., and Srivastava, M.** *Performance aware tasking for environmentally powered sensor networks.* 2004. ACM SIGMETRICS.
162. **Min, R., Furrer, T., and Chandrakasan, A.** *Dynamic voltage scaling techniques for distributed microsensor networks.* 2000. IEEE Computer Society Workshop on VLSI. pp. 43 – 46.