

**IMPLEMENTAÇÃO DE UM ESTIMADOR DE
CONSUMO E ENERGIA DISPONÍVEL PARA
SIMULADORES DE RSSF**

HUGO VINÍCIUS BITENCOURT

IMPLEMENTAÇÃO DE UM ESTIMADOR DE
CONSUMO E ENERGIA DISPONÍVEL PARA
SIMULADORES DE RSSF

Dissertação apresentada ao Programa de Pós-Graduação em Engenharia Elétrica do Instituto de Ciências Exatas da Universidade Federal de Minas Gerais como requisito parcial para a obtenção do grau de Mestre em Engenharia Elétrica.

ORIENTADOR: DIÓGENES CECÍLIO DA SILVA JR.
COORIENTADOR: ADRIANO BORGES DA CUNHA

Belo Horizonte

Abril de 2013

HUGO VINÍCIUS BITENCOURT

**IMPLEMENTATION OF A CONSUMPTION AND
AVAILABLE ENERGY ESTIMATOR FOR WSN
SIMULATOR**

Dissertation presented to the Graduate Program in Electrical Engineering of the Federal University of Minas Gerais in partial fulfillment of the requirements for the degree of Master in Electrical Engineering.

ADVISOR: DIÓGENES CECÍLIO DA SILVA JR.
CO-ADVISOR: ADRIANO BORGES DA CUNHA

Belo Horizonte

April 2013

© 2013, Hugo Vinícius Bitencourt.
Todos os direitos reservados.

Bitencourt, Hugo Vinícius

Implementation of a consumption and available energy
estimator for WSN simulator / Hugo Vinícius Bitencourt. —
Belo Horizonte, 2013
xv, 136 f. : il. ; 29cm

Dissertação (mestrado) — Federal University of Minas Gerais
Orientador: Diógenes Cecílio da Silva Jr.

1. Wireless Sensor Networks. 2. simulators. 3. Cooja.
4. Energy. 5. Battery. I. Título.


"Implementação de um Estimador de Consumo e Energia Disponível para Simuladores de Rssf"

Hugo Vinícius Bitencourt Paula

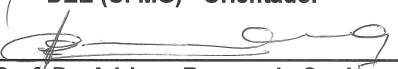
Dissertação de Mestrado submetida à Banca Examinadora designada pelo Colegiado do Programa de Pós-Graduação em Engenharia Elétrica da Escola de Engenharia da Universidade Federal de Minas Gerais, como requisito para obtenção do grau de Mestre em Engenharia Elétrica.

Aprovada em 31 de janeiro de 2013.

Por:



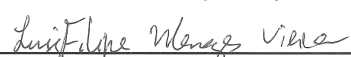
Prof. Dr. Diógenes Cecílio da Silva Júnior
DEE (UFMG) - Orientador



Prof. Dr. Adriano Borges da Cunha
COLTEC (UFMG) - Co-Orientador



Prof. Flávio Henrique Vasconcelos
DEE (UFMG)



Prof. Dr. Luiz Felipe M. Vieira
DCC (UFMG)



Filipe

Agradecimentos

Agradeço ao meu orientador Diógenes Cecílio e co-orientador Adriano Borges pelos ensinamentos, ajuda e conselhos durante todo o processo de desenvolvimento desta dissertação. Agradeço a minha amiga Emilia Reis pelos ensinamentos relacionados a língua inglesa que foram de extrema importância na escrita dessa dissertação. Agradeço aos bolsistas Rodrigo Oliveira e Marlon Wanderlich pela ajuda na implementação e experimentos dessa dissertação. Agradeço aos meus colegas de mestrado que estudaram comigo e ajudaram-me muito nas disciplinas do mestrado.

Resumo

O número de aplicações utilizando sistemas embutidos conectados em redes de comunicação vem crescendo nos últimos anos, em particular, as Redes de Sensores Sem Fio (RSSF). Contudo, RSSF apresentam alguns desafios que devem ser tratados, por exemplo, capacidade de energia limitada. É muito importante nos estágios iniciais de um projeto de RSSF estimar o consumo e a energia disponível em cada nó sensor e em toda a rede. Vários simuladores de RSSF foram desenvolvidos nos últimos anos, no entanto, vários desses simuladores não modelam o hardware do nó sensor e não simulam o código da aplicação que é utilizado no nó sensor real, visto que esses dois fatores são importantes na estimação do consumo de energia. Além disso, simuladores de RSSF implementam somente modelos de bateria simples e lineares. Portanto, é muito importante detectar as diferenças e semelhanças entre os simuladores e selecionar o simulador mais adequado para uma determinada aplicação. Algumas taxonomias foram propostas, porém elas consideram somente a modelagem da rede e do software utilizados em RSSF. Este trabalho propõe uma nova taxonomia para simuladores de RSSF, que é baseada na modelagem dos quatro domínios que um simulador deve tratar, são eles: nó sensor, rede, ambiente e energia. Baseado na taxonomia proposta, o simulador COOJA foi selecionado para a implementação de um novo estimador de consumo de energia e a inclusão de um modelo de baterias alcalinas que considera o efeito de relaxação, a retenção de capacidade e a taxa de descarga. Este trabalho estende o simulador COOJA provendo uma nova versão que estima a energia consumida e disponível em cada nó sensor e em toda a rede. A nova extensão do Cooja apresenta um erro médio menor do que 4,5% para as estimativas da capacidade remanecente final e tempo de vida do nó sensor.

Abstract

Networked Embedded System (NES) is a fast growing application for embedded systems, in particular, for Wireless Sensor Networks (WSN). WSNs pose a set of challenges that should be address, such as the limited available energy. It is very important at the early stages of a WSN project to estimate the consumption and available energy of each sensor node and the whole network. Many WSN simulators have been developed in the recent past, but some simulators were not conceived for hardware modeling and they do not simulate the same application code used in the real sensor node, which are very important in order to estimate the consumption and available energy. Besides, WSN simulators have implemented only very simple and linear battery models and a realistic battery model have not been implemented yet. Therefore, it is very important to find out differences and similarities of WSN simulators and select the right one for a given application. Some taxonomies were proposed, but they addressed mostly the network and software domains only. This work proposes a new taxonomy for WSN simulators that is based on modeling four domains that WSN simulators must address, namely, the sensor node, the network, the environment and the energy. Based on the proposed taxonomy, the COOJA simulator was selected to implement a new energy consumption estimator and a realistic and non-linear battery model for alkaline batteries that incorporates the relaxation effect, discharge rate and capacity retention. This work also extends the COOJA simulator providing a new version that estimates the consumption and available energy of sensor nodes and the whole WSN. The proposed extended version of Cooja presents an average error less than 4.5% for the final remaining capacity and sensor node lifetime estimative.

List of Figures

2.1	Sensor Node Architecture [Vieira et al., 2003]	6
2.2	MicaZ platforms	8
2.3	Tmote Sky	8
3.1	Taxonomies for WSN simulators	17
3.2	Different aspects of a WSN that simulators should address	18
3.3	Sensor Node Domain and layers.	20
3.4	The most referred simulators in the literature classified according to the proposed taxonomy.	23
4.1	CoojaED Extension plugin GUI.	34
4.2	Cooja simulated sensor node [Österlind, 2006].	35
4.3	An illustration of the CoojaED Extension plugin operation. Each part of the process of consumption and available energy estimation is illustrated.	36
4.4	CC2420 radio control state machine [CC2420-datasheet, 2012].	38
4.5	CC2420 radio control state machine simplified.	39
5.1	Experiment model (a schematic for a process with controlled inputs, outputs, controlled and uncontrolled factors).	44
5.2	The experimental setup bench.	45
5.3	Tmote Sky sensor nodes and NI-USB 6216 DAQ.	46
5.4	A MN1604 Duracell 9V battery composed of six 1.5V alkaline batteries was open and connected in groups of two cells with 3V potential.	46
5.5	Measurement of battery voltage and sensor node voltage and current	47
5.6	Case Studies	48
5.7	MN1604 battery typical constant current discharge characteristics at 21°C [MN1604, 2012].	49
5.8	Sens application voltage and current profiles	50

5.9	Sens application zoomed portion of current consumption plotted over a time interval of 10 seconds.	51
5.10	Oscilloscope application voltage and current profiles	51
5.11	Oscilloscope application zoomed portion of current consumption plotted over a time interval of 10 seconds.	52
5.12	Radio Test application voltage and current profiles.	52
5.13	Radio Test application zoomed portion of current consumption plotted over a time interval of 10 seconds.	53
5.14	RSSI-Scanner application voltage and current profiles.	53
5.15	RSSI application current zoomed portion of consumption plotted over a time interval of 1 seconds.	54
5.16	TestAMOnOffSlave application voltage and current profiles.	54
5.17	TestAMOnoffSlave application zoomed portion of current consumption plotted over a time interval of 20 seconds.	55
6.1	Remaining capacity for Sens 1 application.	62
6.2	Remaining capacity for Sens 2 application.	63
6.3	Remaining capacity for Oscilloscope 1 application.	64
6.4	Remaining capacity for Oscilloscope 2 application.	65
6.5	Remaining capacity for RSSI-Scanner 1 application.	66
6.6	Remaining capacity for RSSI-Scanner 2 application.	67
6.7	Remaining capacity for Radio Test 1 application.	68
6.8	Remaining capacity for Radio Test 2 application.	69
6.9	Remaining capacity for TestAMOnOffSlave 1 application.	70
6.10	Remaining capacity for TestAMOnOffSlave 2 application.	71
6.11	Real and estimated sensor node lifetimes.	76
6.12	Validation of the estimated sensor node lifetime(Cooja1) with a sensor node cutoff voltage equal to 1.8V.	78
6.13	Validation of the estimated sensor node lifetime (Cooja2) with a sensor node cutoff voltage equal to 1.8V	80

List of Tables

4.1	An example of a log generated during the plugin execution using a sample rate of 1 ms.	37
4.2	An example of a log generated during the plugin execution using a sample rate of 100 μ s.	37
4.3	An example of the current consumption estimation for the radio when the CoojaED Extension was programmed to sample at every millisecond , and based on the radio operation mode, an electric current consumption is associated.	39
4.4	Output power configuration for the CC2420 radio [Sky, 2012] [CC2420-datasheet, 2012].	40
5.1	Sensor node lifetime (hh:mm:ss) with different sensor node cutoff voltage, 1.8V and 2.1V.	55
5.2	Maximum current consumption values of Tmote Sky for 3V power supply.	56
5.3	Nominal current consumption values of Tmote Sky for 3V power supply.	56
5.4	Nominal current consumption values of tihe CC2420 radio and the MSP430 microcontroller for 3V power supply.	57
5.5	Simulated sensor node lifetime for Tmote Sky platform.	57
6.1	Battery initial voltage.	60
6.2	Comparison between measured and Cooja1 total effective capacity.	61
6.3	Comparison between actual and Cooja2 total effective capacity.	61
6.4	Comparison between Cooja1 and Cooja2 final remaining capacities.	72
6.5	Cooja1 remaining capacity versus the absolute difference between the measured and Cooja1 total effective capacities	73
6.6	Measured sensor node lifetime versus Cooja1 sensor node lifetime with a sensor node cutoff voltage equal to 1.8V	74

6.7	Measured sensor node lifetime versus Cooja2 sensor node lifetime with a sensor node cutoff voltage equal to 1.8V	75
A.1	Temperature ($^{\circ}C$) versus Capacity Loss Look up Table	97
B.1	Effective Capacity	126

Contents

Agradecimientos	vi
Resumo	vii
Abstract	viii
List of Figures	ix
List of Tables	xi
1 Introduction	1
1.1 Motivation	1
1.2 Objectives	3
1.3 Organization	4
2 Basic Concepts of Wireless Sensor Networks	5
2.1 Sensor Nodes	5
2.1.1 Processing Unit	5
2.1.2 Communication Unit	6
2.1.3 Sensing Unit	6
2.1.4 Power Unit	7
2.2 Hardware Platforms for WSN	7
2.3 Power Supply for WSN	9
2.3.1 Terminology	9
2.3.2 Recovery Effects and Rate Capacity	10
2.3.3 Thermal Effects	10
2.3.4 Battery Types	10
2.4 Operating Systems for WSN	11
2.4.1 Contiki	11

2.4.2	TinyOS	12
2.5	Simulators for WSN	13
2.5.1	TOSSIM	13
2.5.2	MSPsim	13
2.5.3	Avrora	13
2.5.4	Cooja	13
2.5.5	Castalia	14
2.5.6	ns-2	14
2.5.7	IDEA1	14
2.5.8	ATLeS-SN	15
2.5.9	SNOPS	15
3	Taxonomy for WSN Simulators	16
3.1	Taxonomies for WSN simulators	16
3.2	The proposed Taxonomy for WSN simulators	17
3.3	Proposed Domains	18
3.3.1	Network Domain	19
3.3.2	Sensor Node Domain	20
3.3.3	Environment Domain	21
3.3.4	Energy Domain	21
3.4	Classification Scheme	22
3.5	Classification of the WSN Simulators	23
3.5.1	TOSSIM	23
3.5.2	Cooja	24
3.5.3	Castalia	26
3.5.4	IDEA1	26
3.5.5	ATLeS-SN	27
3.5.6	ns-2	28
3.5.7	SNOPS	28
3.6	Discussion and Comparison	29
3.6.1	Sensor Node Domain	29
3.6.2	Network Domain	30
3.6.3	Environment Domain	30
3.6.4	Energy Domain	31
3.6.5	Summary	31
4	CoojaED: Extending Cooja Energy Domain	33

4.1	Overview	33
4.2	Cooja Simulation Design Overview	34
4.3	Design of CoojaED Extension	35
4.3.1	Energy Awareness	37
4.3.2	Energy Awareness Model	40
4.3.3	Energy availability	41
4.3.4	Battery Model	41
5	Experiments	43
5.1	Design of Experiments Model	43
5.2	Experimental Setup	45
5.3	Case Studies	48
5.4	Sensor Node Lifetime	49
5.5	Simulations	56
6	Results and Validation	58
6.1	Overview	58
6.2	Total Effective Capacity and Remaining Capacity	59
6.2.1	Total Effective Capacity	59
6.2.2	Remaining Capacity	61
6.3	Sensor Node Lifetime	74
7	Conclusions and Future Works	82
7.1	Future Works	84
	Bibliography	85
	Appendix A Temperature versus Capacity Loss Look up Table	93
	Appendix B Effective Capacity Table	98
	Appendix C CoojaED Extension source code	127

Chapter 1

Introduction

With the goal to instrument the physical world with pervasive networks that are able to sense, process and interact with their surrounding environment, networked embedded systems have emerged, in particular, the concept of a wireless sensor network (WSN).

WSNs are composed of many devices called sensor nodes. Sensor nodes are autonomous, compact, low-power sensors integrated with a low-power embedded CPU, memory and wireless communication, and can also have local data processing and multihop communication capabilities. Sensors nodes can be used to detect or monitor a variety of physical parameters or conditions (e.g. humidity, temperature and seismicity). Therefore, WSN can be used in wide range of applications such as habitat monitoring, military applications, health care applications and monitoring volcanic activity [Vieira et al., 2003][Warneke and Pister, 2002][Zheng, 2009].

1.1 Motivation

Wireless Sensor Networks poses a set of challenges that need to be addressed. First, the lifetime of the network. Second, the limited available energy. Third, a network formed by sensor nodes that needs to cover larger areas, but the communication is provided via short-distance radios, as a result, the communication is the most expensive operation. Fourth, computation and storage are limited. [Fonseca, 2009].

Energy management is a critical concern in a WSN since sensor nodes are commonly powered by primary batteries, with a finite capacity and need replacement when depleted, which increase the maintenance costs. In some WSNs, the replacement of batteries can be cost prohibitive or even not feasible. It is desirable that each sensor node manages and balances its energy resources efficiently and autonomously. Without energy management, the lifetime of a WSN is limited to only a few weeks or

even days, depending on the type of application it is running and the size of batteries it is using. Besides, energy management in a WSN is an important requirement for the maintenance, development of self-management techniques, planning, viability and design of protocols and applications for WSN [da Cunha, 2010]. Therefore, WSN designers need to know for what and why the energy is consumed and the amount of available energy and necessary to make the WSN operational. For example, how much energy do individual operations, such as transmitting and receiving packets or using CPU, cost?

It is very important at the early stages of a WSN project to estimate the energy consumption and the amount of available energy of each sensor node and the whole network. For this reason, the use of simulators in a WSN project is almost imperative. There are many reasons for that. First, WSNs are not cheap, hence experiments on a real-world WSN, which could consist of hundreds or thousands of sensor nodes, are very expensive. In some applications, sensor nodes are deployed in inaccessible and dangerous areas. For this reason, it is more cost-effective to use simulation. Secondly, simulation allows debugging. Thirdly, simulation allows experiments under controlled conditions. [Jevtić et al., 2009].

Many WSN simulators have been presented in the recent past, however, these WSN simulators were developed to accomplish different goals. For example, TOSSIM [Levis et al., 2003] and COOJA [Österlind et al., 2006] are designed to simulate the behavior of an operating system. TOSSIM and COOJA simulate the same application code used in the real sensor node. Castalia [Boulis, 2011] is intended to simulate the behavior of algorithms and protocols in a wireless communication channel.

Some simulators such as PAWiS [Glaser et al., 2008], ns-2 [Fall and Varadhan, 2011], j-sim [Sobeih et al., 2006] and Castalia were not conceived for hardware modeling and they do not simulate the same application code used in the real sensor node, which is very important to estimate the consumption and available energy [Haase et al., 2011]. These simulators also provide a very simple and inaccurate energy model. For example, in ns-2, the level of energy is decreased at every transmitted and received packet by some given value and the energy is linearly subtracted from a virtual battery in Castalia.

COOJA uses the duty command of MSPsim [Eriksson et al., 2007] (i.e. COOJA and MSPsim are integrated) that only prints the duty cycle of components of a sensor node in different activity states. TOSSIM has an extension called PowerTOSSIM [Shnayder et al., 2004] that estimates the energy consumption per-node and per-component. PowerTOSSIM is a post-processor that uses data annotated during a TOSSIM simulation. After simulation, PowerTOSSIM analyzes the collected data and generates power information. However, TOSSIM loses the fine-grained timing

and interrupts properties of the code that can be important when the application runs on the hardware, that are important to estimate the consumption and available energy [Titzer et al., 2005].

Besides, WSN simulators have implemented a very simple and linear battery model that provides the available energy in a WSN. A realistic battery model like BMAB (Behavioral Model of Alkaline Batteries) [da Cunha, 2010] [da Cunha and da Silva, 2012] that incorporates the discharge rate, capacity retention, and relaxation effect are not implemented yet.

Therefore, it is very important to find out differences and similarities of WSN simulators and select the right one for a given application. In the literature, some classification schemes [Jevtić et al., 2009] [Haase et al., 2011] [Du et al., 2010c] were proposed aiming at helping WSN designers to find differences and similarities. However, these classification schemes do not address all the aspects that compose a WSN with the purpose of simulation, such as the sensor node (hardware and software), the network, the environment and the energy.

These classification schemes address mostly the network and software domains only. Energy, environment and hardware domains are not considered. They do not expose the advantages, disadvantages and limitations of the WSN simulators in the implementation of energy awareness and energy availability estimators.

1.2 Objectives

This work has the following objectives:

1. Propose a new taxonomy for WSN simulators with four domains to capture and describe a real WSN. With these four domains one can characterize and compare different simulators in order to properly choose the most adequate for a given application. The proposed taxonomy aims at helping WSN designers to find the most appropriate WSN simulator for their WSN projects providing a detailed vision on different aspects of a complete WSN simulation. The proposed taxonomy enables designers to easily find out differences and similarities among WSN simulators.
2. Implement a new energy consumption estimator and a realistic and non-linear battery model for alkaline batteries that incorporates the relaxation effect, discharge rate and capacity retention. Using the proposed taxonomy the COOJA simulator was selected. This work also extends the COOJA simulator providing a

new version that estimates the consumption and available energy of every sensor nodes of a WSN.

1.3 Organization

This document is organized as follows. Chapter 2 presents a brief summary for some main concepts of WSN. Chapter 3 presents the proposed taxonomy. The implementation of an energy consumption estimator and a realistic and non-linear battery model for alkaline batteries is described in Chapter 4. Chapter 5 describes the experimental project to validate the implementation of the energy consumption estimator and battery model. Chapter 6 presents the results and model validation of the consumption and available energy estimator. Chapter 7 concludes this work and presents some future works.

Chapter 2

Basic Concepts of Wireless Sensor Networks

This Chapter presents a brief summary for the main concepts of WSNs: sensor node architecture, WSN hardware platforms, WSN power supply, WSN operating system and WSN simulators.

2.1 Sensor Nodes

A sensor node or mote (both names will be used in the rest of this work) is a tiny computer network node that can sample analog signals and communicate with other sensor nodes wirelessly, especially by radio. A sensor node is composed of four main parts: sensing unit, processing unit, communication unit and power unit, as show in Figure 2.1. The sensing unit is responsible for collecting data from the environment (e.g. temperature, heat, seismicity). The processing unit is composed of a internal memory to store data and applications programs, a CPU to process data and an Analog-to-Digital Converter (ADC) to convert analog signals from the sensing unit. The communication unit consists of a bidirectional wireless communication channel that allows sensor nodes to communicate. The power unit has the purpose to supply the energy to the sensor node provided by primary batteries [Vieira et al., 2003] [Zheng, 2009].

2.1.1 Processing Unit

The processing unit is responsible for controlling all the modules of the sensor node and to perform any required computation. The processing unit is composed

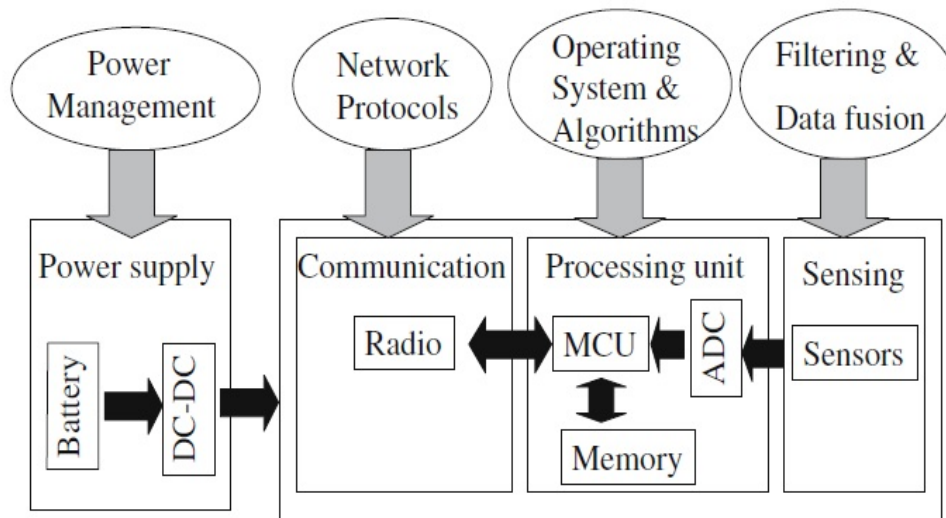


Figure 2.1: Sensor Node Architecture [Vieira et al., 2003]

of a microcontroller unit (MCU). Microcontroller is a microprocessor with memory and peripherals or support devices embedded in a single chip. Beyond the CPU, microcontrollers also include volatile and non-volatile memories and interfaces such as UART, USB, SPI and I2C, and peripherals such as Analog-to-Digital Converters (ADC), counters and timers. In order to be energy efficient, the microcontroller hardware needs to be low power, with different operating modes, and fast wake-up time. Another important feature is the start-up time since the microcontroller of a sensor node will usually switch between the idle and active modes [Vieira et al., 2003] [Zheng, 2009].

2.1.2 Communication Unit

Sensor nodes communicate using a wireless communication channel. The communication unit must be energy-efficient, and allows output power setting. Several aspects affect the power consumption of a radio including the type of modulation, data rate, and transmission power. In general, radios can operate in three distinct modes: transmit, receive, and idle.

2.1.3 Sensing Unit

The sensing unit is composed of sensors, which are devices that produce electrical signals in response to a change in a physical condition. Examples include thermal sensors, seismic sensor, tilt sensor, pressure sensors, temperature sensor, radiation

detectors, magnetic sensors, and chemical and biological sensors. In short, the sensing unit of the sensor node is the link to the outside world [Vieira et al., 2003] [Zheng, 2009].

2.1.4 Power Unit

The power unit has the purpose to supply power to the sensor node, and sensor nodes are usually powered by batteries. Batteries are classified into two groups, primary and secondary. A primary battery is a battery that is designed to be used once and discarded (i.e. non-rechargeable battery) while secondary batteries are rechargeable and can have their stored energy replenished many times.

2.2 Hardware Platforms for WSN

Several hardware platforms have been developed over the past few years to aid researches in WSNs. The capabilities of these platforms vary significantly and they can be classified into two categories based on their capabilities and usage, low-end and high-end platforms. This section presents a brief description of them, in particular MicaZ and Tmote Sky, since they are the most mentioned and used hardware platforms in this work.

Low-end platforms such as Mica2, MicaZ (Fig. 2.2), Telosb and Tmote Sky (Fig. 2.3) have been used for the validation and assessment of new applications and protocols, and they are characterized by their limited capabilities in terms of processing, memory, and communication.

The main difference between Mica2 and MicaZ platforms is the wireless transceiver, Mica2 uses the Chipcon CC1000 [CC1000-datasheet, 2012] while MicaZ is based on the Chipcon CC2420, but both use the same microcontroller, the ATmega128L. The ATmega128L is an 8-bit low-power microcontroller that provides 128 kbytes of Flash, 4 kbytes of EEPROM, 4 kbytes of SRAM, 53 general purpose I/O, four flexible Timer/Counters with compare modes and PWM, two USARTs, a byte oriented Two-wire Serial Interface, a 10-bit ADC with 8-channels, a SPI serial port and an internal calibrated RC oscillator [Lajara et al., 2010].

Telosb and Tmote Sky share the same design, but they were designed by two different companies, Crossbow and Moteiv Corporation, respectively. Moteiv and Crossbow changed their names and are now called Sentilla and Memsic, respectively. Both platforms use the MSP430F1611 MCU and the CC2420 wireless transceiver.



Figure 2.2: MicaZ platforms

The MSP430F1611 is part of the Texas Instruments MSP430 family based on an ultra-low-power microcontroller with a clock of 8 MHz. It is a 16-bit RISC processor with several power-down modes and extremely low sleep current consumption. It also features 10 kbytes of RAM, 48 kbytes of program flash memory, SPI, UART, timers with capture and compare functionality, a 2-port 12-bit Digital-to-Analog converter (DAC) module, a supply voltage supervisor and a 3-port DMA controller [Sky, 2012] [Lajara et al., 2010]. The CC2420 radio implements the IEEE 802.15.4 standard and can be configured and controlled by an SPI port and a series of digital I/O lines and interrupts. The CC2420 offers reliable wireless communication, a programmable output power and power management capabilities with a very low-power consumption [Sky, 2012] [Lajara et al., 2010] [CC2420-datasheet, 2012].

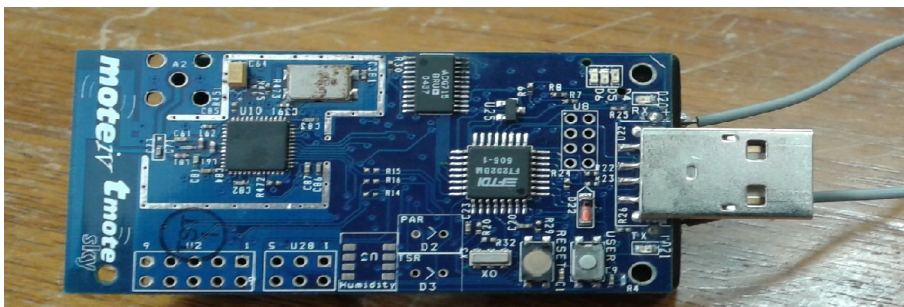


Figure 2.3: Tmote Sky

Beyond sensing, local processing and multiple hops communication, WSNs require some functionalities that cannot be efficiently executed by the low-end platforms. For example, sensor network management require higher processing power and memory,

the integration of a WSN with existing network infrastructure demand multiple communication techniques to be integrated through gateway modules. To address these requirements, high-end platforms have been developed such as Stargate [Stargate, 2012]. Stargate is a high-performance processing platform developed for sensing, signal processing, control, and sensor network management. Stargate uses the Intel PXA-255 Xscale processor with a clock of 400MHz, has 23Mbytes of flash memory, 64Mbytes of SDRAM, and an onboard connector for Mica family platforms as well as PCMCIA Bluetooth and IEEE 802.11 cards.

2.3 Power Supply for WSN

Energy is a critical concern in a WSN, since sensor nodes are usually powered by primary batteries, and these batteries have a finite capacity and need replacement when depleted, which increases the maintenance costs. The replacement of batteries can be cost prohibitive or even not feasible in some WSNs.

For this reason, it is very important to control the power consumption of sensor nodes, since with high power consumption the batteries will be depleted very quickly. Each sensor node should manage and balance its energy resources efficiently and autonomously. Without energy management, the lifetime of a WSN is limited to only a few weeks or even days, depending on the type of application it is running and the capacity of its batteries [da Cunha, 2010]. Harvesting energy from the ambient has emerged as an alternative to provide supply to the sensor node, but requires secondary batteries to store the extra energy.

2.3.1 Terminology

We presented a brief summary of the some common terms related to batteries [Pop et al., 2005]:

1. **Cell:** a cell is the basic electrochemical unit of a battery. A cell consists of two electrodes in a container filled with an electrolyte, and it is used to generate electrical energy from stored chemical energy or to store electrical energy in the form of chemical energy.
2. **Battery:** a battery is a device capable of converting chemical energy into electrical energy and vice versa. A battery is composed by one or more cells connected in an appropriate series arrangement to obtain the required operating voltage and capacity for a certain load.

3. **State of Charge (SoC)**: is the percentage of the maximum possible charge that is present inside the battery.
4. **Cycle Life**: the number of times the battery can be charged/discharged.
5. **Cutoff Voltage**: is the lowest operating voltage of a cell at which the cell is considered depleted.

2.3.2 Recovery Effects and Rate Capacity

Two important factors of batteries are the charge recovery effect and the rate capacity effect. Charge recovery effects are due to the relaxation effect, when the battery is discharging at a high rate for some time, and then its discharge current is cutoff or reduced, there is a capacity recovery. Rate capacity effects are due to the dependency between the actual capacity of a battery and the magnitude of the discharge current [Castillo et al., 2004][da Cunha, 2010].

2.3.3 Thermal Effects

Researchers should choose a battery with a low self-discharge and relative immunity to temperature variations. In fact, the effect of temperature on battery efficiency depends on the specific chemistry being considered. At lower temperatures, increased internal resistance of the battery lead to reduced capacity, and at high temperatures, decreased internal resistance leads to increased effective capacity of the battery, but continuous exposure to high temperatures leads to a shortened cycle life [Park et al., 2005].

2.3.4 Battery Types

There are several primary and secondary battery types available such as alkaline, nickel cadmium, nickel metal-hydride and lithium ion, and this section presents a brief summary of them [Pop et al., 2005][da Cunha, 2010][Castillo et al., 2004].

1. **Alkaline** is a primary battery that uses an alkaline aqueous solutions as electrolyte and its advantages over zinc-carbon include longer shelf life, higher energy density, and superior leakage resistance.
2. **Nickel Cadmium (NiCd)** is a secondary battery that uses a positive nickel electrode of a $Ni(OH)_2/NiOOH$ compound and a negative cadmium electrode of Cd and $Cd(OH)_2$, and its advantages include low cost, fast charge and discharge times. The main disadvantage of NiCd is the so-called memory effect, which

causes the battery to deliver only the capacity that was used during the repeated charge/discharge cycles before.

3. **Nickel Metal-Hydride (NiMH)**: is a secondary battery with a construction is very similar to the NiCd battery. The major difference between them is the fact that in a NiMH a metal hydride alloy is used for the negative electrode instead of cadmium, and it has roughly twice the energy density of NiCd.
4. **Lithium Ion**: is a secondary battery that uses a positive electrode built of metal oxides to store the lithium ions and a negative electrode made of graphite or petroleum coke. During the discharge, lithium ions move from the negative electrode to the positive electrode and back when charging. It has significant higher energy density and twice the life cycle of a NiMH battery.

For the interested reader, more information about batteries can be find in [da Cunha, 2010] [Pop et al., 2005] [Castillo et al., 2004].

2.4 Operating Systems for WSN

Contiki [Dunkels et al., 2004] and TinyOS [Levis et al., 2004] are the two most common operating systems used in WSN and for this reason, they will be detailed in this section.

2.4.1 Contiki

Contiki is a lightweight and highly portable open source operating system written in C for WSNs that was built around an event-driven kernel, and provides optional threading facilities for individual processes. The Contiki kernel comprises a lightweight event scheduler that dispatches events to running processes. Process execution is triggered by events dispatched by the kernel to the processes or by a polling mechanism. Any scheduled event will run to completion, however, event handlers can use internal mechanisms for preemption [Lajara et al., 2010][Farooq and Kunz, 2011].

Contiki supports many hardware platforms (e.g. Tmote Sky, MicaZ, Z1 [Z1, 2012] and Wismote [Wismote, 2012]), microcontrollers (e.g. TI MSP430 family, Atmel AVR family and ST STM32w) and radios (e.g. the Texas Instruments CC2420, CC2520 and CC1020, and RFM TR1001) [Contiki-WebSite, 2012].

One of the main contributions of Contiki is the introduction of protothreads. Protothreads are lightweight threads developed for severely memory constrained devices with a very small memory overhead for concurrent programming. They enable

programmers to block conditions that stop a thread waiting for the activation of an event from another concurrent thread and simplify and reduce the number of the state machines needed to implement the sequence of high-level operations [Lajara et al., 2010][Farooq and Kunz, 2011].

In Contiki, an application can use either IPv4 or IPv6. Contiki provides an implementation of uIP (i.e. a TCP/IP protocol stack for small microcontroller). uIP supports TCP, UDP, ICMP and IP protocols, and has a minimum set of features needed for a full TCP/IP stack. Contiki also provides another layered protocol stack, named Rime. Rime provides single hop broadcast, single hop unicast, and multiple hops communication support, and enables applications to run their own routing protocols in multiple hops communication. Finally, Contiki also provides an implementation of RPL (IETF IPv6 Routing Protocol for Low-power and Lossy Networks) [Jamieson and Moss, 2012] by the name ContikiRPL [Farooq and Kunz, 2011].

2.4.2 TinyOS

TinyOS is an open source, flexible and component-based operating system written in NesC [Gay et al., 2003] (a variant of the C programming language) for WSN. According to the requirements of the application, different components are connected together by means of interfaces with the scheduler to compose a static image that runs on the sensor node. Therefore, new applications can be programmed by combining components connected using their interfaces and only the components that are really needed in the application are compiled and included in the final executable file, with a significant reduction of the total amount of memory required. TinyOS provides a robust and reliable functionality by making use of static memory allocation and a non-preemptive FIFO scheduler. When an interruption occurs, the microcontroller jumps immediately to the corresponding event handler, stopping the execution of the current task [Farooq and Kunz, 2011][Lajara et al., 2010].

TinyOS supports several hardware platforms (e.g. Tmote Sky, MicaZ, Mica2 and IRIS [IRIS, 2012]), microcontrollers (e.g. MSP430 family, the Atmel Atmega128, the Atmega128L, the Atmega1281, and the Intel PX27ax processor) and radios (e.g. The Texas Instruments CC2420, CC1100 and CC2500, Atmel RF212 and RF230, the Infineon TDA5250 and the Semtech XE1205) [Stehlik, 2011].

At the MAC layer, TinyOS provides an implementation of the following protocols: a single hop TDMA protocol, a TDMA/CSMA hybrid protocol that implements Z-MAC's slot stealing optimization, B-MAC, and an optional implementation of an IEEE 802.15.4 compliant MAC. Regarding network protocols, TinyOS has implemented

several protocols such as TYMO (TinyOS implementation the DYMO protocol - a point-to-point routing protocol for Mobile Ad-hoc Networks), TinyRPL (TinyOS implementation of the IETF IPv6 Routing Protocol for Low-power and Lossy Networks (RPL)), and 6lowpan (TinyOS implementation of the IETF IPv6 for Low power Wireless Personal Area Networks) [Farooq and Kunz, 2011] [TinyOS, 2012].

2.5 Simulators for WSN

Many simulators for WSN have been developed in recent past and six simulators were selected based on the criteria of popularity, availability, and maintainability after 2008. The selected simulators are: TOSSIM 2.x [TOSSIM, 2012], Cooja 2.5 [Contiki-WebSite, 2012], MSPsim 2.5 [Eriksson et al., 2007], Avrora [Titzer et al., 2005], Castalia 3.2 [Boulis, 2011], and ns-2 2.35 [Fall and Varadhan, 2011]. Recently, another class of simulators using the SystemC language have been presented and IDEA1 1.0 [Du et al., 2011], ATLeS-SN 2.0 [ATLeS-SN, 2012], and SNOPS 1.0 [Fall and Varadhan, 2011] simulator were also selected.

2.5.1 TOSSIM

TOSSIM [Levis et al., 2003] is a discrete event simulator for the operating system TinyOS [TinyOS, 2012]. It is part of the TinyOS distribution. TOSSIM maps directly to TinyOS code, i.e. compiling an application for the simulation environment is equivalent to compiling it for an actual real sensor node. TOSSIM works by replacing hardware components by simulation wrappers using a Hardware Abstraction Layer [Perla et al., 2008][TOSSIM, 2012]. Unfortunately, TOSSIM only supports the MicaZ mote [MICAz, 2012].

2.5.2 MSPsim

MSPsim is a java-based instruction level simulator for the MSP430 microprocessor series, that provides the execution time of each instruction. MSPsim includes a sensor board simulator that simulates hardware peripherals such as sensors, radio, and LED's. MSPsim can simulate a complete sensor node such as the Tmote Sky [Sky, 2012]. MSPSim can simulate both TinyOS and Contiki applications [Eriksson et al., 2009] [Eriksson et al., 2007] [Stecklina et al., 2011].

2.5.3 Avrora

Avrora is a java-based cycle-accurate simulator for embedded sensing programs. It can simulate Mica2 motes [MICA2, 2012]. Avrora includes a nearly complete ATmega128L MPU, and the CC1000 radio. Avrora is also capable of running a complete WSN simulation, allowing applications to communicate via the radio using the software stack provided by TinyOS. Avrora has some extensions such as AVRORAZ [de Paz Alberola and Pesch, 2008], that allows simulation of the MicaZ motes and includes an indoor radio channel model [Haase et al., 2011][TinyOSWiki-Avrora, 2012][Titzer et al., 2005].

2.5.4 Cooja

Cooja [Österlind et al., 2006] is a java-based simulator designed for sensor nodes running the Contiki operating system [Dunkels et al., 2004]. MSPsim and Avrora are integrated into Cooja.

Cooja can simulate Contiki applications in two ways, either by running the application as a compiled native code directly on the target CPU, or by running compiled application code in MSPSim. Cooja can also simulate a WSN without a specific sensor node or Contiki application focusing only on the network behavior. Using MSPsim, Cooja can also simulate TinyOS applications for MSP430 motes (e.g. Telosb [Polastre et al., 2005]) [Eriksson et al., 2009]. Cooja also uses the Avrora simulator to simulate Contiki AVR-based motes applications (e.g. MicaZ [MICAz, 2012]).

2.5.5 Castalia

Castalia [Boulis, 2011] is a simulator for WSN and BAN (Body Area Network). It is based on the OMNeT++ platform [Varga and Hornig, 2008]. An OMNeT++ based simulator is built from elements called modules which might be simple or compound. A simple module is the basic unit of execution. A compound module is a construction of simple and/or other compound modules. Modules communicate using packets that are sent through connections that span between modules or directly to their destination modules [Jevtić et al., 2009] [Boulis, 2011] [Varga and Hornig, 2008].

2.5.6 ns-2

ns-2 [Fall and Varadhan, 2011] is the most notorious and well known simulator for networks. It is a discrete event simulator targeted to computer network research.

It began as a wired network simulator, but today it also offers support for mobile ad-hoc wireless networks. ns-2 is an object-oriented simulator, written in C++, with an OTcl [OTcl, 2012] interpreter as a front-end. The protocols, models and simulator itself are implemented in C++ while simulator configuration, specification scenarios and network topology set up are written in OTcl scripts [Jevtić et al., 2009][Fall and Varadhan, 2011]. There is a new simulator, named ns-3 [Henderson et al., 2006], that is a new improved version and eventual replacement of ns-2.

2.5.7 IDEA1

IDEA1 (hIerarchical DEsign plAtform for Sensor Networks Exploration) [Du et al., 2010a] is a WSN simulator written in SystemC language and uses the SCNSL (SystemC Network Simulation Library) alpha library [Fummi et al., 2008]. The main contributions of IDEA1 are the following: a graphical user interface (GUI), an energy model that enable energy awareness, processors (e.g. ATMEL ATmega128), radio transceivers and an IEEE 802.15.4 media access algorithms model [Du et al., 2011].

2.5.8 ATLeS-SN

ATLeS-SN (Arizona Transaction-Level Simulator for Sensor Networks) [Hiner et al., 2010] is a transaction-level modeling and simulation environment for WSN implemented in SystemC language. ATLeS-SN enables the modeling of various hardware components of the sensor nodes (e.g. sensor, communication units) and across WSN (e.g. wireless communication channels).

2.5.9 SNOOPS

SNOOPS (Sensor Networks Optimization by Power Simulation) [Damm et al., 2010] is a simulation library to design WSNs, written in SystemC and TLM 2.0 [Aynsley, 2009]. It uses transaction concepts from TLM 2.0 [Aynsley, 2009] to model end-to-end communications among sensor nodes, including multiple hops communication [Haase et al., 2011] [Moreno et al., 2011].

Chapter 3

Taxonomy for WSN Simulators

3.1 Taxonomies for WSN simulators

It is very important to find out differences and similarities of WSN simulators and select the right one for a given application. In the literature, some taxonomies were proposed in past years aiming at helping WSN designers to find differences and similarities.

Jevtic et al. [Jevtić et al., 2009] classified WSN simulators into three types: generic, code level and firmware. A generic WSN simulator simulates a WSN with the focus on networking aspects while operating system and hardware architecture are not considered. A code level WSN simulator simulates the same application code as in the real sensor node. The firmware WSN simulator uses instruction set simulators and CPU cycle-accurate models to execute deployable application and system operation code compiled for the target sensor node.

Haase et al. [Haase et al., 2011] classified WSN simulators in a similar manner as the Jevtic classification (see Figure 3.1), but using a different nomenclature: network and system simulators, operation system emulators and microcontroller emulators. These categories are equivalent to the ones presented by Jevtic, with one exception: network and system simulators also include WSN simulators based on SystemC language [Grotker, 2002].

Du et al. [Du et al., 2010c] classified WSN simulators into four categories: network simulators with node models (NSNM), network simulators with node emulators (NSNE), node system simulator with network models (NSSNM), and node emulators with network models (NENM). NSNM is equivalent to the generic type of Jevtic classification. NSNE includes WSN simulators that integrates the generic and the firmware types of Jevtic classification. NSSNM includes the WSN simulators that were designed using hardware description languages (e.g. SystemC). NENM includes the

WSN simulators classified as code level and firmware types of Jevtic classification.

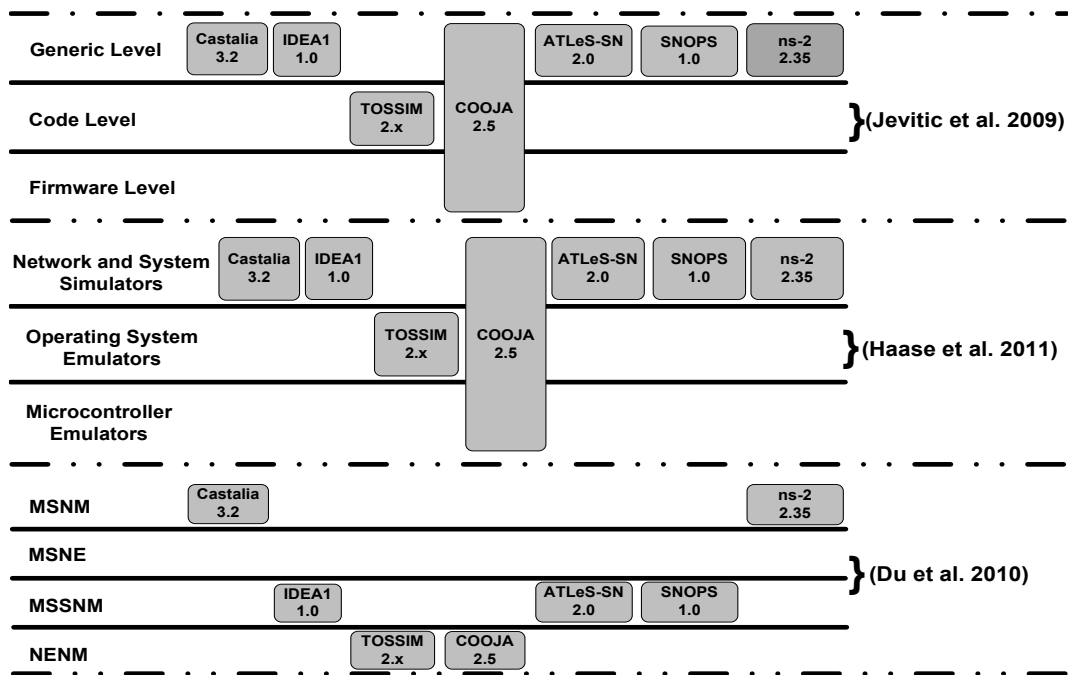


Figure 3.1: Taxonomies for WSN simulators

Figure 3.1 shows the WSN simulators presented in Chapter 2 classified according to the taxonomies found in the literature, as described above. All taxonomies addresses mostly the network and software domains only. Energy, environment and hardware domains are not considered. These taxonomies do not provide a detailed vision of different aspects of a real WSN. A new proposed taxonomy that addresses all the mentioned limitations will be described in the following.

3.2 The proposed Taxonomy for WSN simulators

This work proposes a new taxonomy for WSN simulators with four domains to capture and describe a real WSN. With these fours domains one can characterize and compare different simulators in order to properly choose the most adequate for a given application.

The proposed taxonomy aims at helping WSN designers to find the most adequate WSN simulator for their WSN projects providing a detailed vision on different aspects of a complete and closer to WSN simulation. The proposed taxonomy enables designers to easily find out differences and similarities among WSN simulators.

3.3 Proposed Domains

Figure 3.2 illustrates a real WSN. Each sensor node collects environmental data and processes them. The sensor nodes use a network protocol stack to communicate to each other and to the Base Station node through a wireless communication channel.

A WSN is formed by three main domains: sensor nodes, a network (communication among sensor nodes) and the environment (communication medium and sensing). Besides these three domains, energy should also be included as another central domain, since it is a critical constraint of a real WSN.

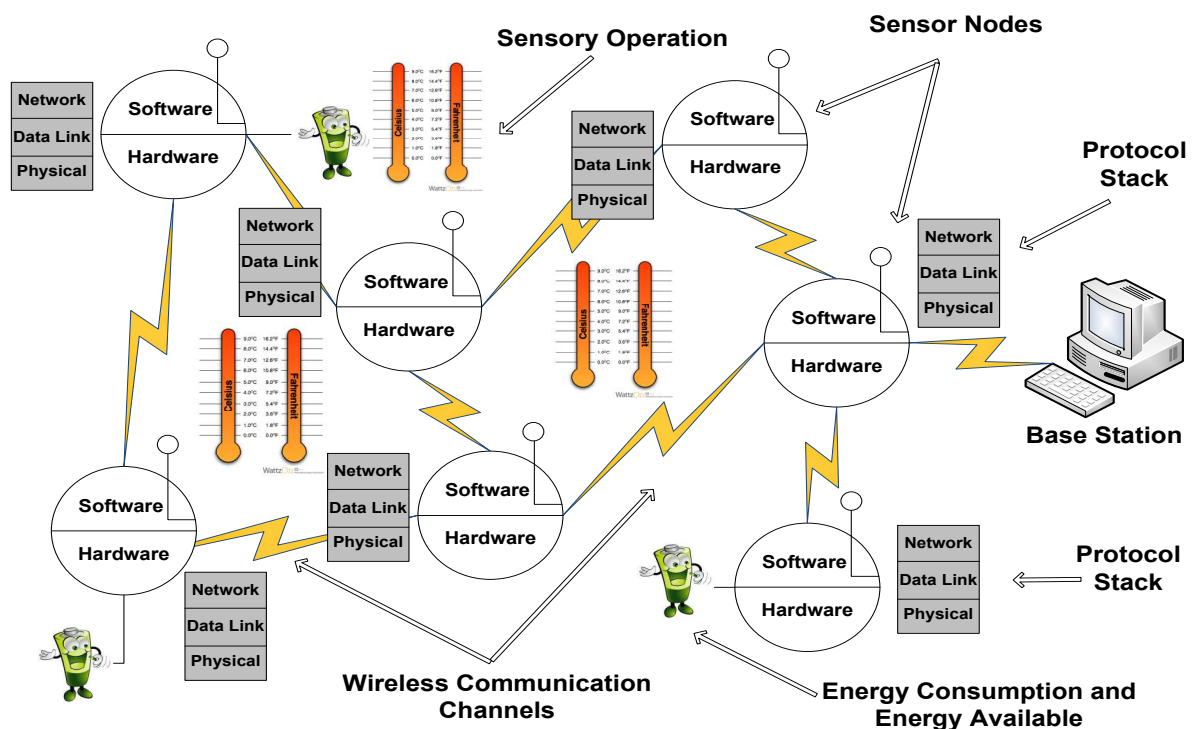


Figure 3.2: Different aspects of a WSN that simulators should address

Therefore, for a detailed simulation that provides accurate information on different aspects of a real WSN, the simulator must address different domains that compose a real WSN. This work proposes a new view for a WSN: the sensor node computing platform, the communication network, the surrounding environment, and the available and necessary energy to make and sustain the WSN operational. The proposed domains names are: sensor node, network, environment and energy.

The sensor node domain models the computing platform and is divided into two separated sub-domains: hardware and software. The network domain models

the communication. The environment domain models the ambient where the WSN is deployed and involves the communication medium and the sensory operation. The energy domain models how the sensor nodes and the WSN perceives the available energy for the operation.

Each domain is divided into layers that represent the level of details used in the simulation. The layers of each domain are explained in the following subsections.

3.3.1 Network Domain

The network domain is usually modeled after the OSI model from ITU-T and is commonly known as layered protocol stack. However, protocols for traditional wireless networks cannot be applied directly to WSN without modification, because they do not take into account energy, computation and storage constraints as primary concerns. Besides, transport and application layers are not used in many WSN applications [Yick et al., 2008][Zheng, 2009].

The physical layer is responsible for encoding bit streams into signals that are suitable for transmission over the communication medium. For this purpose, the physical layer must deal with various related issues such as transmission medium and frequency selection, carrier frequency generation, signal modulation and detection.

For a WSN, minimizing energy consumption starts at the physical layer. A proper selection of transmission/listen/reception power and modulation is needed to minimize energy consumption. Physical layers for WSN include some requirements: low power radio design, power-aware transmission schemes, and power-aware modulation schemes.

The data link layer is responsible for data frame creation and detection, data stream multiplexing, medium access control (MAC), and error control in order to provide reliable point-to-point and multipoint transmissions. The design of the data link layer for WSN must address attributes such as energy efficiency, flow control, scalability to node density, and bandwidth utilization.

The network layer handles the routing of data across the network from the source to the destination. In general, a source sensor node can transmit the data to the base station node either via a single hop or multiple hops. Single hop is usually costly in terms of energy consumption for sensor nodes. In contrast, multiple hops cannot only significantly reduce the energy consumption of sensor nodes, but also effectively reduce the signal propagation and channel fading effects inherent in single hop, and is therefore preferred.

In summary, the network domain is defined by three layers: physical, data link and network.

3.3.2 Sensor Node Domain

The sensor node domain is divided into two sub-domains: software and hardware. Figure 3.3 illustrates the layers of the sensor node domain.

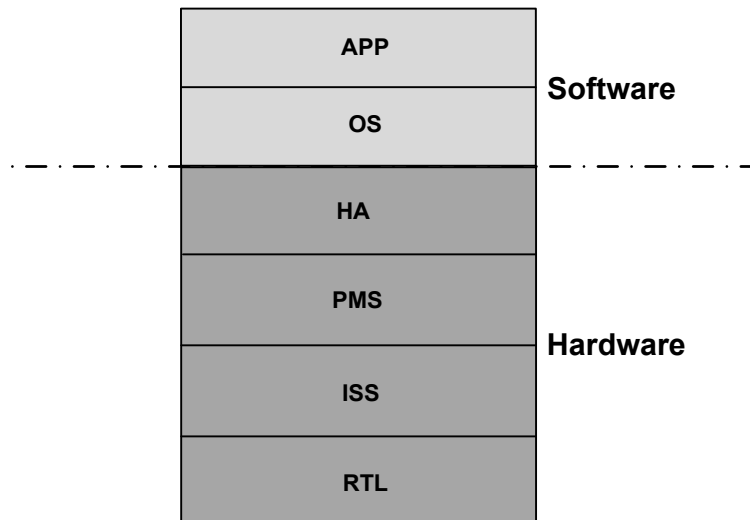


Figure 3.3: Sensor Node Domain and layers.

The Software domain has two layers: APP (application) and OS (operating system). APP is the description level of an application program to be executed on the sensor node, generally using Nesc [Gay et al., 2003], C, C++, SystemC, or Java languages. This application program can be executed either on the development environment or on the sensor node. OS is the description that includes the application program and an operation system or microkernel.

The Hardware domain has four layers: HA (Hardware Abstraction), PMS (Process/Memory/Switch), ISS (Instruction Set Simulator), and RTL (Register Transfer Level). HA is the hardware abstraction level where modules and subsystems are described as an executable code or simulatable code. At this level no specific hardware description is used, and no mapping to existing actual hardware is enforced. PMS is a notation where the embedded system architecture is described as a composition of modules like processors, memories, buses, and peripherals [Siewiorek et al., 1982]. At PMS level a real hardware can be described and more details related to real implementations can be used. ISS is the level equivalent to an instruction set simulator of a processor. At this level, the hardware behavior can be described as a sequence of machine instructions. RTL is the description level of a digital system composed by registers, functional units and their interconnections. RTL is the lowest

and more detailed level of abstraction that is useful for networked embedded systems designers.

3.3.3 Environment Domain

The environment domain is divided into three layers: wireless channel, sensing and sensing model.

In WSN, sensor nodes communicate through a wireless communication channel. However, this communication presents imperfections due to collisions, noise, fading, path loss, deflection, reflection, interferences (internal and external), obstacles (e.g. walls, floors, trees, light sources) and so forth. WSN simulators should model the wireless communication channel, since it is an important requirement for the simulation of a real WSN.

In a sensor node, sensors are responsible to produce some measurable responses to the changes in physical or chemical conditions. But, sensed-data generation (i.e. sensing) is usually neglected in WSN simulators, in other words, WSN simulators usually only feed random numbers to sensor nodes. However, WSN simulators can use sensed-data generation more realistically (e.g. feeding the sensor nodes with traces of sensed-data based on specific physical process) [Boulis, 2011] [Hossain et al., 2008]. Since sensing is an important requirement for a real WSN, this proposed taxonomy model the sensing activities.

Besides, in WSN, the sensing ability is marked by the sensing model. It has a direct influence on the coverage ability (i.e. how well an area of interest is being monitored by a deployed WSN) and service quality. Sensing model can be either simple (e.g. if the occurrence of an event is within the sensing range of a node then the event will be assumed to be detected, otherwise not) or more realistic (e.g. sensing model takes into account many elements (e.g. obstacles) which might interfere with the sensing ability) [Hossain et al., 2008] [Ningning et al., 2010]. Since network coverage is an important issue for WSN, the proposed taxonomy also takes into account sensing models.

3.3.4 Energy Domain

As sensor nodes usually operate on limited battery power and when a sensor node is depleted of energy, it disconnects from the network and that can significantly impact the performance of the application. WSN lifetime depends on the number of active sensor nodes, so energy must be used efficiently in order to maximize the WSN lifetime.

Each sensor node must manage and balance its energy resources intelligently and autonomously. Besides, energy management in WSN is an important requirement for the maintenance, development of self-management techniques, planning, quality of service (QoS), viability and design of protocols and applications for WSN [da Cunha, 2010].

Since energy usage is a very important concern for a WSN, designers need to know accurately the energy resource of their applications before the deployment in real environments. Therefore, it is necessary that WSN simulators calculate accurately the energy consumption and the amount of available energy of a WSN [da Cunha, 2010].

The proposed taxonomy defines energy awareness as the ability to provide the energy consumption of a WSN. Energy awareness consists of providing at any time the energy consumption for each sensor nodes and the whole WSN. Also, the proposed taxonomy defines energy availability as the ability to estimate the amount of available energy in a WSN. Energy availability consists of providing at any time the amount of available energy of the sensor node and the whole WSN [da Cunha, 2010].

The implementation of the energy availability in WSN simulators requires the modeling of the power unit of the sensor nodes. Then, to WSNs with no capability of energy harvesting, energy availability consists in the use of battery models [da Cunha, 2010]. Therefore, the energy domain has two layers: energy awareness and energy availability.

3.4 Classification Scheme

In the proposed taxonomy, WSN simulators are represented by rectangles covering domains layers, and each layer covered by the simulator shows the level of approximate detail used by the WSN simulator in the modeling of each domain.

Regarding implementation details, a simulator can use a sophisticated or a very simple or partial model and the corresponding rectangle will be drawn with a different gray tone. A dark gray rectangle means that the simulator properly covers that modeled layer and a light gray rectangle means that it is partially covered. For example, ATLeS-SN [Hiner et al., 2010] models the wireless communication channel considering only the signal attenuation using an equation for the free space attenuation based on the distance, then its rectangle color in the wireless channel layer of the Environment Domain is light gray. In contrast, Castalia models the average path loss calculation using lognormal shadowing and allows sensor node mobility, then its corresponding rectangle color is dark gray.

Figure 3.4 shows several referred WSN simulators in the literature classified according to the proposed taxonomy and these two examples can be found on the first and fifth columns. The current classification of WSN simulators depends on their versions, thus this classification can be modified with simulators further improvements and new versions.

3.5 Classification of the WSN Simulators

The simulators presented in Chapter 2 are classified according to the proposed approach and the level of implementation detail is compared. As a result, Figure 3.4 is generated and a clear view of each simulator and a corresponding comparison can be made.

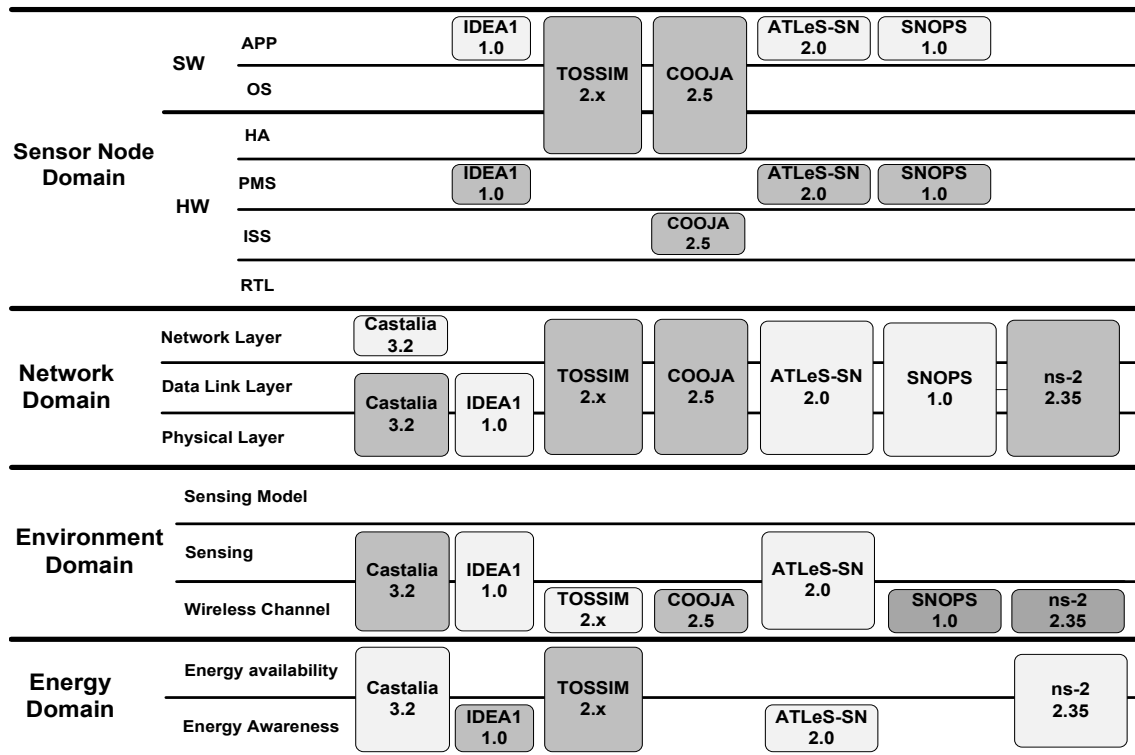


Figure 3.4: The most referred simulators in the literature classified according to the proposed taxonomy.

3.5.1 TOSSIM

In regard to sensor node domain, TOSSIM models the sensor nodes at the APP, OS and HA layers. TOSSIM maps directly to TinyOS code, i.e. compiling an application for the simulation environment is equivalent to compiling it for the real sensor node.

TOSSIM works by replacing hardware components by simulation wrappers using a Hardware Abstraction Layer [Perla et al., 2008][TOSSIM, 2012].

For the network domain, the behavior of the physical layer is simulated through a radio object that is based on experimental results using the TI CC2420 [CC2420-datasheet, 2012] radio. TOSSIM also provides some primitives that can be used to model others radios. The MAC object deals with the data link layer, and can control several properties (e.g. backoffs) important for the CSMA protocol. Besides, TOSSIM supports routing protocols available in TinyOS [TOSSIM, 2012].

For the environment domain, the default radio propagation model is based on signal-strength. Designers can provide a set of data to TOSSIM which describes the propagation strengths, and can also specify noise floor and receiver sensitivity. TOSSIM simulates the radio frequency noise and interference using the Closest Pattern Matching (CPM) algorithm [TOSSIM, 2012]. CPM takes a noise trace as input and generates a statistical model from it. This model can capture bursts of interference and other correlated phenomena. However, this model is not perfect since it does not handle correlated interference at sensor nodes that are close to one another. TOSSIM does not implement the sensing and sensing model layers.

Regarding the energy domain, TOSSIM has an extension called PowerTOSSIM [Shnayder et al., 2004] that estimates the energy consumption per-node and per-component. PowerTOSSIM is a post-processor that uses data annotated during a TOSSIM simulation. After the simulation, PowerTOSSIM analyzes the collected data and generates power information. There are three versions of PowerTOSSIM: PowerTOSSIM [Shnayder et al., 2004] for TinyOS 1.x, PowerTOSSIM2 [Prabhakar et al., 2008] for TinyOS 2.x and MICA2 motes [MICA2, 2012] and PowerTOSSIMz [Perla et al., 2008] for TinyOS 2.x and MICAz [MICAz, 2012] motes. PowerTOSSIMz can calculate the energy consumption per-node and per-component, and has a battery model that estimate the energy availability in each sensor node [Perla et al., 2008] [Haase et al., 2011]. This battery model assumes that the recovery probability varies according to the battery's remaining charge using a stochastic approach, modeling the non-linearity of the discharging rate of the battery.

3.5.2 Cooja

In regard to the sensor node domain, Cooja models the sensor nodes using the APP, OS, HA and ISS (using either MSPsim or Avrora) layers. Cooja can simulate Contiki applications in two ways, either by running the application as a compiled native code directly on the host CPU, or by running compiled application code in MSPSim. Cooja

can also simulate a WSN without a specific sensor node or Contiki application focusing only on the network behavior. Using MSPsim, Cooja can also simulate TinyOS applications for MSP430-based motes (e.g. Telosb [Polastre et al., 2005]) [Eriksson et al., 2009]. Cooja also uses the Avrora simulator to simulate Contiki application of AVR-based motes (e.g. MICAz [MICAz, 2012]).

For the network domain, the set of protocols supported by Cooja is equivalent to Contiki and has the following main features: TI CC2420 and TR1100 [TR1100-datasheet, 2012] radios; IEEE 802.15.4 standard [IEEE, 2006]; uIPv4, uIPv6 and RPL [Jamieson and Moss, 2012] routing protocols. Due to MSPsim, Cooja also enables applications to communicate via radio using the software stack provided in TinyOS.

In regard to the environment domain, Cooja provides four radio propagation models: Unit Disk Graph Medium (UDGM) Constant Loss; Unit Disk Graph Medium (UDGM) Distance Loss; Directed Graph Radio Medium (DGRM); Multi-path Ray-tracer Medium (MRM) [Stehlik, 2011].

UDGM Constant Loss models the transmission range as an ideal circle in which all the sensor nodes outside that circle do not receive packets while the sensor nodes within that circle receive all the packets. UDGM Distance Loss is an extension in which interferences are considered and the packets can be transmitted with SUCCESS RATIO TX probability and received with SUCCESS RATIO RX probability. DGRM specifies the transmission success ratio in an asymmetric per-link base and can define propagation delays for the links. MRM model uses ray tracing technique. The receiver power is calculated using Friis formula, and obstacles are considered as attenuators. It also calculates refractions, reflections and diffractions. Furthermore, others radio propagation models can be added [Österlind et al., 2006]. However, COOJA does not implement the sensing and sensing model layers.

Cooja does not implement the energy domain itself, but designers can use either the duty command of MSPsim or the Energy Profiling used in Contiki [Dunkels et al., 2007]. Duty command prints the duty cycle of components of a sensor node (e.g. microcontroller and transceiver) in different activity states (e.g. listen and transmit). Then, based on this information, the designer can calculate the energy consumption by multiplying the time spent in each activity state with the respective electric current consumption [Eriksson et al., 2009].

Contiki has implemented an energy profiler named Energest. When the component is turned on, Energest is called to produce a time stamp. When the component is turned off, the time interval is computed. These Energest values describe how much time in clock ticks was spent for radio communication or common tasks using

the CPU. Energy Profiling is implemented in the device driver for the hardware which energy is to be estimated. Energest values can be converted to seconds by dividing them by the clock rate used by the microcontroller (e.g. one second is equal to 32,768 clock ticks in the Tmote Sky running at 32,768Hz). Thus, developers can calculate the energy spent by multiplying the time interval with the correspondent current consumption that can be found in datasheets. Contiki also provides an interface named Powertrace for the Energy Profiling system.

3.5.3 Castalia

Castalia does not model the sensor node domain, however, the implementation of the physical and data link layers of the network domain in Castalia are very detailed.

The radio module captures several features of a typical low-power radio used in sensor nodes. Some of the main features are: multiple states (transmit, receive/listen), multiple configurable sleep states; transition delays from one state to another; multiple (configurable) transmission power levels; multiple modulation schemes (e.g. FSK, PSK, DiffQPSK); and continuous RSSI calculation. Three radios are already implemented: TI CC2420, TI CC1000 [CC1000-datasheet, 2012] and a so-called BANRadio which describes the narrowband radio proposed in the IEEE 802.15 Task Group 6 documents [Boulis, 2011].

Castalia has four main MAC modules implemented: Tunable MAC (a duty-cycle MAC that exposes many parameters to the user, and the application for tuning), T-MAC [van Dam and Langendoen, 2003], S-MAC [Ye et al., 2002], IEEE 802.15.4 MAC [IEEE, 2006] and IEEE 802.15.4 MAC draft proposal for BAN.

In contrast, the network layer in Castalia receives less attention. It only provides `multipathRings` and `bypassRouting` that, as the module name suggests, does not implement any routing.

The environment domain in Castalia has the following features: models the average path loss calculation using lognormal shadowing; allows sensor node mobility; models temporal variation of path loss in order to simulate a large variation in the received signal with time; allows the use of simple radio propagation model so that designers can test different hypotheses; calculates radio interferences, and SINR (signal to interference plus noise ratio), and can decide whether or not it receives a packet. An interesting feature allows to use more realistic values for sensing. For example, a mathematical formula that determines the value of a physical process at a certain location and time. Castalia also enables the modeling of sensor deviations (e.g noise and bias). However, Castalia does not implement the sensing model layers.

In regard to the energy domain, the computation of energy consumption is the main task for the resource manager module. It tracks the energy spent by the sensor nodes and this energy is linearly subtracted from a virtual battery.

3.5.4 IDEA1

In regard to the sensor node domain, IDEA1 models the sensor node at the APP and PMS levels. IDEA1 models each component of a sensor node as an individual module. Microcontroller and transceiver are modeled as finite state machines. IDEA1 simulates an application program written in SystemC language. IDEA1 has implemented some available off-the-shelf (COTS) processors and transceivers that are basic components of some actual nodes (e.g. MICAz).

For the network domain, IDEA1 has implemented three transceivers as finite state machines (FSM): TI CC2420 [CC2420-datasheet, 2012], TI CC1000 [CC1000-datasheet, 2012] and Microchip MRF24J40 [Microship-datasheet, 2012]. There are two IEEE 802.15.4 media access algorithms implemented: unslotted CSMA-CA and slotted CSMA-CA.

IDEA1 models the environment domain using the SCNSL library that models propagation delay, interferences, collisions and path loss, takes into account the spatial positions of sensor nodes and their on-going transmissions. Furthermore, sensors are simulated as a stimuli generator, but IDEA1 does not model the sensing model layer.

IDEA1 models the energy domain using FSMs. It works as follows: during simulation, the state transitions of each component, such as the microcontroller and transceiver, are recorded and each state is associated with an electric current consumption value. Based on this information, the battery module calculates the energy consumption for each component [Du et al., 2010b]. The battery module also calculates the battery remaining capacity of sensor nodes (i.e. the energy availability layer), however, no results were shown [Du et al., 2011].

3.5.5 ATLeS-SN

In regard to sensor node domain, ATLeS-SN models the sensor node at the APP and PMS layers of the proposed taxonomy. Each sensor node is composed by an *App* component to implement the sensor node functionality (user-level application), a *Sensor* component for modeling the sensing unit and a *NetStack* component for designing the communication unit. ATLeS-SN also simulates an application program written in SystemC through the *App* component. An *App* component is analogous to

the user-level application software that will execute on a physical sensor node. However, ATLeS-SN does not have implemented COTS processors or transceivers.

A *NetStack* component provides the foundation for modeling the network domain. It enables designers to implement their own protocols such as MAC and routing protocols. However, at this moment, ATLeS-SN does not include standard protocols (e.g. IEEE 802.15.4).

As for the environment domain, ATLeS-SN provides a *PhysChannel* component that enables the modeling of wireless communication channels. A designer can implement various radio propagation models within the *PhysChannel* component, from the simplest (e.g. a simple equation based on distance in free space) to the more complex (e.g. deflection or echoes from obstacles and terrain). However, ATLeS-SN does not include advanced radio propagation models. *PhysChannel* also simulates collisions.

The *Sensor* component enables designers to model the sensor unit at various abstractions levels, from sensor that only reads values from an input file for specific sensors that incorporates the device driver code. But, advance sensed-data generation is not available. ATLeS-SN does not model the sensing model layer.

Regarding the energy domain, ATLeS-SN calculates the energy consumption per-node and per-component. ATLeS-SN also calculates the energy consumption per-state (e.g. active for CPU) for each component. However, ATLeS-SN does not have implemented COTS processors and transceivers, and it does not implement the energy availability layer.

3.5.6 ns-2

ns-2 does not model the sensor node domain. In regard to network domain, the IEEE 802.15.4 standard [IEEE, 2006] is available since version 2.26. ns-2 has implemented five different routing protocols for ad-hoc wireless network: DSDV (Destination-Sequenced Distance Vector), DSR (Dynamic Source Routing), TORA (Temporally-Ordered Routing Algorithm), AODV(Ad hoc On-Demand Distance Vector) and PUMA (Protocol for Unified Multicasting Through Announcements) [Jevtić et al., 2009] [Fall and Varadhan, 2011].

As for the environment domain, there are three different radio propagation models implemented in ns-2: free space model, two-ray ground reflection model and log normal shadowing model. ns-2 does not implement the sensing and sensing model layers.

In regard to the energy domain, ns-2 has implemented a very simple energy model that enables energy availability. The level of energy is represented by an attribute that

is set with a value that represents the initial level of available energy of the sensor node. Then, the level of energy is decreased at every transmitted and received packet by some given value.

3.5.7 SNOPS

In regard to the sensor node domain, SNOPS models the sensor node at the APP and PSM layers. SNOPS enables designers to model their own sensor node components and applications [Haase et al., 2011].

As for the network domain, SNOPS enables designers to implement their own protocols and integrate them. However, no protocols has been provided [Haase et al., 2011].

In regard to the environment domain, the radio propagation model includes time-variant effects in the attenuation calculation, such as fading or mobility.

As for the energy domain, SNOPS does not provide energy models itself, but allows designers to create and integrate their own energy models and energy profiles into SNOPS, that can enable energy awareness.

3.6 Discussion and Comparison

This section presents an elaborate analysis for the simulators surveyed in the previous section. For each domain all the simulators are classified according to the proposed approach and the level of implementation detail is compared. As a result, a clear view of each simulator is generated and a corresponding comparison can be made. With such view a designer can choose the best simulator for a given WSN project. With this taxonomy, the advantages and disadvantages of the these WSN simulators in the implementation of energy awareness and energy availability estimators are presented.

3.6.1 Sensor Node Domain

IDEA1, ATLeS-SN and SNOPS are WSN simulators written using the SystemC language. SystemC is a hardware description language that enables the simulation of sensor nodes with different functionality at different abstraction levels, the communication network and the surrounding environment. The use of a single tool for the whole system development can be an advantage for the design of networked embedded systems, in particular, sensor nodes [Fummi et al., 2008].

SystemC simulators can also model sensor nodes at the RTL layer and can be integrated to an ISS. An ISS-SystemC integration enables the simulation of the same application code used in real sensor nodes. SystemC language supports the hardware and software co-simulation of sensor nodes closer to the real implementation. However, at the moment, IDEA1, ATLeS-SN and SNOOPS model the sensor node domain only at the PMS layer and they do not simulate the same application code used in real sensor node. They enable the simulation of heterogeneous WSN, composed by different sensor nodes and applications. They are also scalable simulators.

TOSSIM and Cooja simulate the same application code used in the real sensor node, as the result, the designers do not need to spend effort in rewriting the application code already developed. They also simulate the operating system used by the sensor nodes.

Cooja enables the simulation of a heterogeneous WSN, but its scalability is usually poorer than other simulators. Cooja simulates a network composed by MSP430 and AVR-based motes running different Contiki applications, and can also simulate TinyOS applications on MSP430-based motes. In contrast, TOSSIM do not allow simulation of a heterogeneous WSN since it only simulate a network composed by MICAz motes and a single TinyOS application, however, TOSSIM is a scalable simulator.

Finally, Castalia, and ns-2 do not implement the sensor node domain and no WSN simulator analysed implement the sensor node at RTL layer.

3.6.2 Network Domain

IDEA1, ATLeS-SN and SNOOPS partially cover the network domain. IDEA1 has implemented some available COTS transceivers and two IEEE 802.15.4 media access algorithms as FSM, while ATLeS-SN does not model standard protocols and COTS transceivers and no library module of protocols stack has been provided in SNOOPS.

TOSSIM provides the CC2420 radio, CSMA and routing protocols also available in TinyOS. Cooja is primarily a Contiki operating system simulator, then the set of network protocol supported by Cooja is equivalent to Contiki's. Due to MSPsim, Cooja also allows applications to communicate via the radio using the software stack provided in TinyOS.

The network domain in Castalia is very detailed. It provides a realistic physical layer, several MAC protocols and has three radios modeled, but it partially covers the network layer.

ns-2 partially cover the network domain, it does not implement MAC or network protocols specifically designed for WSN like B-MAC, X-MAC and SPIN. Besides, ns-2 provides only routing protocols for ad-hoc wireless network.

3.6.3 Environment Domain

The SystemC language is able to implement the environment domain easily, but IDEA1 and ATLeS-SN partially cover this domain, providing a very simple wireless channel and sensing layers. In contrast, SNOOPS provides a radio propagation model that include time-variant effects in the attenuation calculation, such as fading or mobility.

Cooja has implemented different radio propagation models from the simplest to the most complex, as a result, developers can test the behavior of their TinyOS and Contiki applications on different radio propagation models. In contrast, TOSSIM partially covers the wireless channel layer.

The environment domain in Castalia is the most detailed. Castalia provides realistic models of wireless channel and sensing. ns-2 provides three radio propagation models.

Finally, no simulator analysed implement any sensing model.

3.6.4 Energy Domain

Despite the urge for WSN simulators to provide a way to estimate the energy consumption and the amount of available energy of a WSN, only two simulators properly covers at least one layer of the energy domain. TOSSIM is the only simulator that properly covers the energy availability layer. It has implemented a battery model that provides the amount of available energy for each sensor node.

However, TOSSIM only simulates a network composed by MicaZ motes and a single TinyOS application. TOSSIM does not support Low Power Listening (LPL) [LPL, 2012]. As the result, is not possible to simulate low power TinyOS applications. Besides, TOSSIM loses the fine-grained timing and interrupt properties of the code that can be important when the application runs on the hardware [Titzer et al., 2005].

In contrast, IDEA1 enables energy awareness of a heterogeneous WSN composed by different applications and sensor nodes platforms with different functionality or description at different abstraction levels, that is the main advantage of IDEA1. However, it does not simulate the same application code used in the real sensor node.

Castalia and ns-2 partially covers the energy domain. They provide a very simple and inaccurate energy model. They are not conceived for hardware modeling and they

do not simulate the same application code as in the real sensor node, which is very important to estimate the energy consumption [Haase et al., 2011].

Finally, ATLeS-SN partially covers the energy awareness layer. Cooja and SNOPS does not implement the energy domain.

3.6.5 Summary

The current version of TOSSIM is only suitable for a WSN composed by a single TinyOS application and several MICAz motes. In contrast, Cooja is adequate for a WSN composed by MSP430 and AVR-based motes running different Contiki applications, and can also simulate TinyOS applications on MSP430 motes.

IDEA1, ATLeS-SN and SNOPS are suitable for a heterogeneous WSN composed by many sensor nodes. Besides, designers can test a WSN that includes their own sensor node platform. SystemC enables the modeling of embedded systems at RTL level that is closer to the real implementation. As a result, designers can test the energy consumption of a WSN in a more accurate way.

Castalia is more adequate for designers who want to test the behavior of their algorithms and protocols in a wireless communication channel and ns-2 is not suitable for WSN.

Chapter 4

CoojaED: Extending Cooja Energy Domain

Chapter 3 presented a new taxonomy for WSN simulators and defined the energy domain. Based on this taxonomy, the Cooja simulator was selected to design the energy domain, since it does not implement the energy domain, was integrated to a cycle-accurate simulator, simulates the same application code used in the real sensor node (TinyOS and Contiki applications), provides different radio propagation model from the simplest to the most complex, and the most important, can be extend easily.

4.1 Overview

This work extends Cooja simulator providing a new version (i.e. CoojaED Extension) that estimates the consumption and available energy of sensor nodes and the whole WSN. This new version of Cooja provides the energy information at simulation time, in other words, it calculates and prints the consumption and available energy at every simulation time. Besides, designers can save a log file with all this information.

A new plugin for Cooja named New Energy Domain, shown in Figure 4.1 was developed. In the Appendix C some parts of the source code of the CoojaED Extension is provided. With this new plugin, Cooja properly covers the two layers of the energy domain, since the CoojaED Extension plugin provides, at any time, the consumption and available energy of every sensor node. The CoojaED Extension tracks the transition of states and the timing information as the sensor network system is under operation and provides the energy awareness of every hardware components.

The plugin interface has two text box named ("Energy Awareness (mAh)", "Energy Availability (mAh)") and one button ("Save energy information file").

These text boxes show the total energy consumption total and the battery remaining capacity, respectively. The button enables designers save a log file with the energy information.

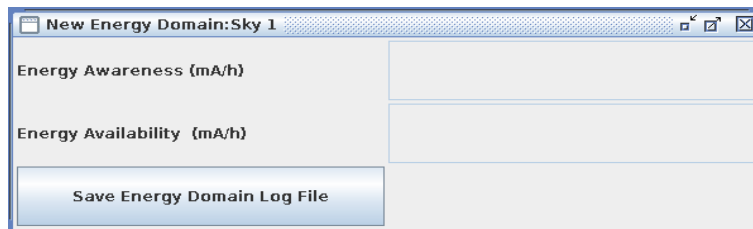


Figure 4.1: CoojaED Extension plugin GUI.

4.2 Cooja Simulation Design Overview

A Cooja simulation consists of a network composed by a number of sensor nodes being simulated. A simulated sensor node has three properties: the node type, its data memory, and its hardware peripherals, as illustrated in Figure 4.2. Each simulated sensor node is connected to a node type, and has its own memory and a number of interfaces. Sensor nodes of the same type are initialized with the same data memory, and during simulation, sensor nodes data memories can eventually differ since the sensor nodes can have different external inputs. The hardware peripherals of simulated nodes are called interfaces, and enable Cooja to detect and trigger events such as incoming radio traffic. Interfaces also represent properties of simulated nodes like positions that the actual sensor node is not aware of [Österlind et al., 2006] [Österlind, 2006].

All interactions with simulations and simulated nodes are performed via plugins. The plugins are implemented like a regular Java panel, and they are registered at runtime before they can be used. Plugins can be of different types [Österlind et al., 2006] [Österlind, 2006]:

1. The GUI plugin is the simplest plugin type. It only needs a running GUI to be constructed, and is passed as an argument when a user initializes the plugin. Via the GUI, relevant information like the current simulation and the simulated sensor nodes can be accessed. It is not removed when the current simulation is removed and can be used to transfer information between different simulations.
2. The Simulation plugin depends on a simulation, when such a plugin is created, the current simulation is passed as an argument. The plugins are removed when

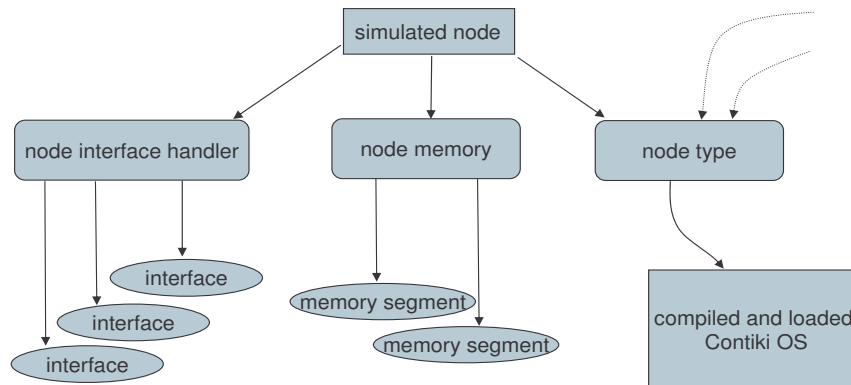


Figure 4.2: Cooja simulated sensor node [Österlind, 2006].

the simulation is also removed. An example of Simulation plugin can be either displaying the number of simulated sensor nodes and the current simulation status or a graphical representation of the positions of all simulated sensor nodes.

3. The Mote plugin depends on a simulation sensor node, if such sensor node is removed, so will the plugin be. An example of this plugin type is the CoojaED Extension.

4.3 Design of CoojaED Extension

Cooja and MSPsim were written in JAVA language and they were included in the Contiki project. To access to Cooja and MSPsim source code. Once the directories are reached, any part of the simulator can be changed. This section describes the implementation of the CoojaED Extension and all the java classes and source code file that were modified and used to create the plugin are presented. The Figure 4.3 shows the CoojaED Extension operation, and the process of consumption and available energy estimation is illustrated. Each part of the Figure 4.3 is described along the text.

Cooja was implemented using the observer pattern. It is a software design pattern in which an object, called the subject, maintains a list of its dependents, called observers, and notifies them automatically of any state changes, usually by calling one of their methods. Interfaces are observable, any entity of the simulator can register as

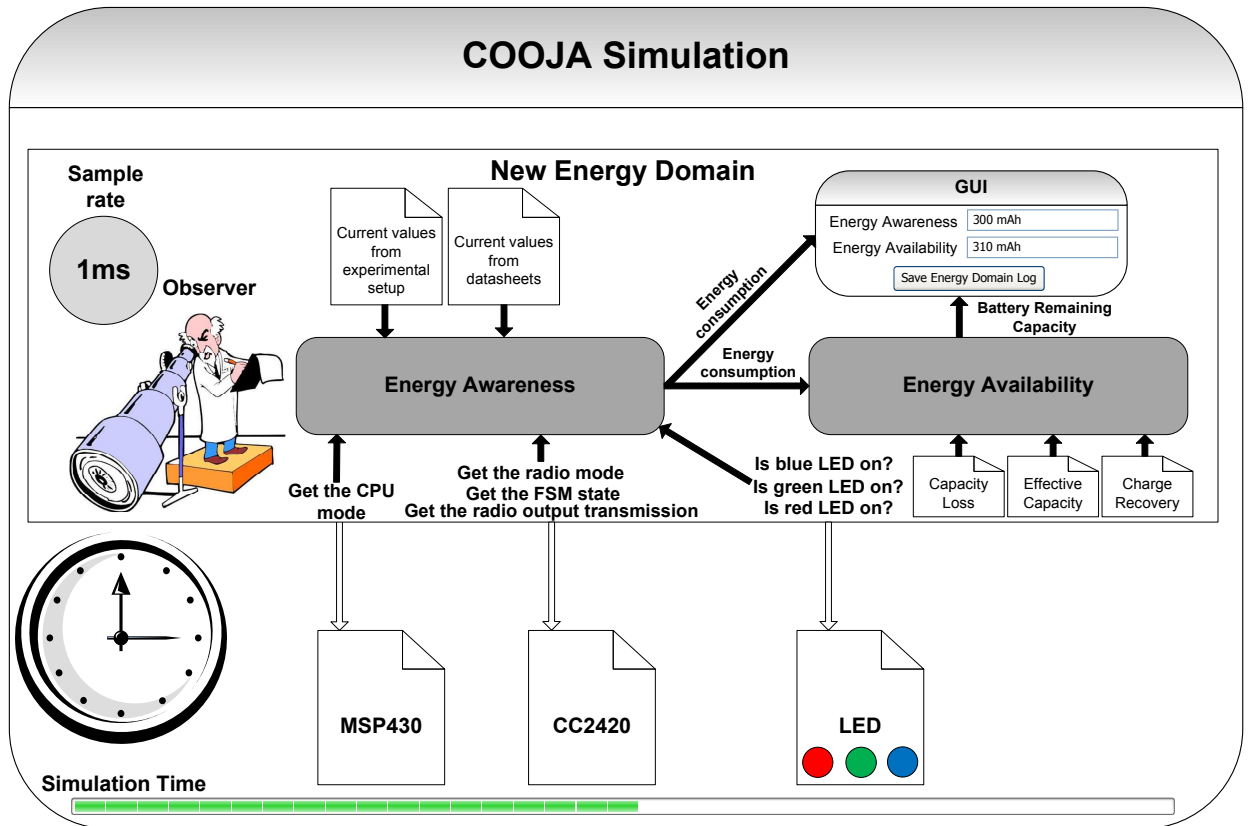


Figure 4.3: An illustration of the CoojaED Extension plugin operation. Each part of the process of consumption and available energy estimation is illustrated.

an observer (i.e. an interface that enables the implementing class to be notified of any changes in the observable objects), and is notified whenever the interface decides so. The observer-observable approach enables very dynamic interactions between different parts of the simulator, and not only simulation interfaces use this approach. For example, the simulation notifies all observers when a new sensor node have been either created or added or removed [Österlind, 2006].

A new observer was included in `Simulation.java` source code file (see Listing C.1), with this modification, the CoojaED Extension can track all state transitions of all components (e.g. cpu, radio) of the sensor nodes during the simulation with a sample rate that can be defined, as can be seen on the left side of Figure 4.3. The designer can track the current radio and CPU modes at a rate of 1ms or $100\mu\text{s}$, for example, Table 4.1 and Table 4.2 illustrate two logs generated during plugin execution where

the CoojaED Extension was programmed to sample at every millisecond and $100\mu s$ the currently radio mode, respectively.

Simulation time (ms)	Radio mode
1	RX
2	RX
3	IDLE
4	TX
5	TX

Table 4.1: An example of a log generated during the plugin execution using a sample rate of 1 ms.

Simulation time (μs)	Radio mode
100	RX
200	RX
300	IDLE
400	TX
500	TX

Table 4.2: An example of a log generated during the plugin execution using a sample rate of $100\mu s$.

4.3.1 Energy Awareness

The CC2420 radio has a built-in state machine that is used to switch between different operational states or modes, which is shown in Figure 4.4. In summary, the CC2420 radio has four modes: receive, transmit, idle, and power off, and can be simplified to the FSM presented in the Figure 4.5 that is used by the CoojaED Extension. When the sensor node is receiving, transmitting or idle, the corresponding states are RX, TX, and IDLE, respectively. When the sensor node is powered off, the state can be either VROff or Power Down.

In MSPsim, the CC2420 class models the complete radio state machine (see Figure 4.4) and provides a method named `getState()`, that allows to get the currently radio operational state. MSPsim also provides a method called `getMode()`, that returns whether the radio is either receive, transmit, idle or power off state. Then, during the simulation, the CoojaED Extension tracks radio mode through `getMode()` method, and each operation mode is associated with an electric current consumption value. If the mode is power off, the CoojaED Extension call the method `getState()` to return whether radio is in VROff or Power Down state, since according to the CC2420 datasheet the current consumption in VROff is $0.02\mu s$ and Power Down is $20\mu s$ (see Listing C.3).

Besides, the MSP430 class provides the method `getMode()` that can be used to get the current microcontroller operational state (i.e. active, LMP0, LPM1, LPM2, LPM3, LPM4). The CoojaED Extension also tracks the current microcontroller operation mode, and each mode is also associated with an electric current consumption value (see Listing C.3).

In Cooja, the LED class implements the three sensor node LEDs (red, green and blue) and provides four methods: `isAnyOn()`, `isRedOn()`, `isGreen()`, and `isYellowOn()`. The first one returns true whether any LED is on and false otherwise, the second, third and fourth returns true whether the red, green, and blue LED are on and false otherwise, respectively, and the current spent when a specific LED is on is also computed (see Listing C.3). The process described above can be seen in Figure 4.3.

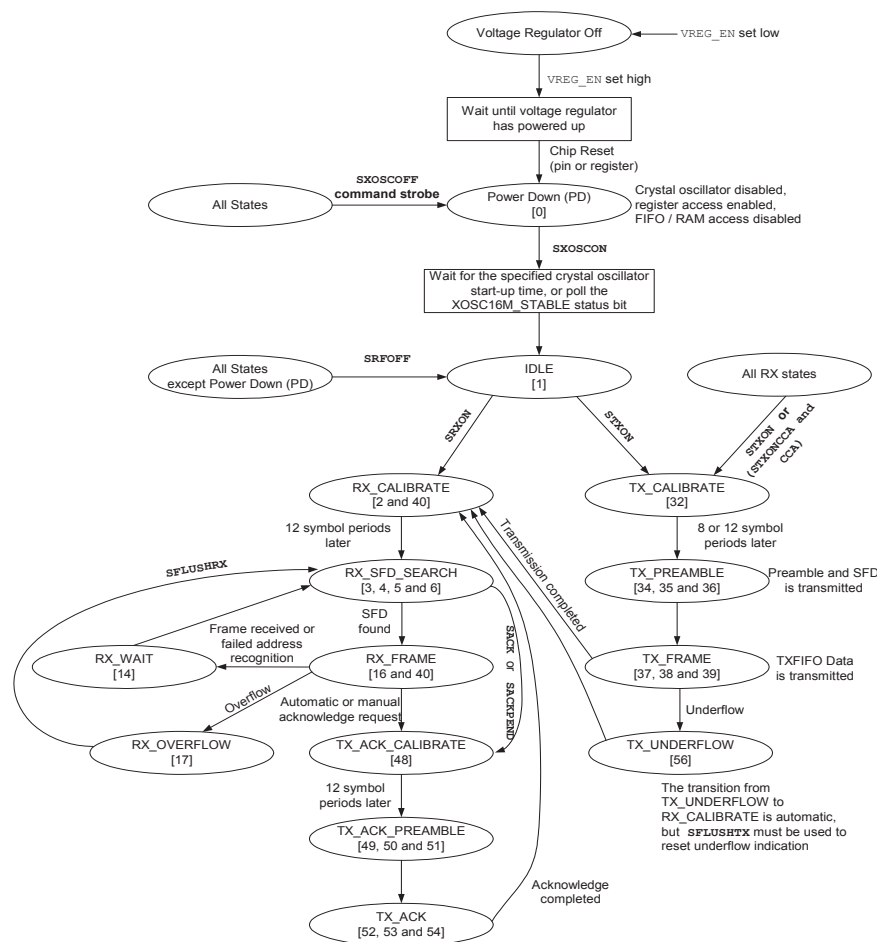


Figure 4.4: CC2420 radio control state machine [CC2420-datasheet, 2012].

The Table 4.3 illustrates an example of the current consumption estimation for the radio when the CoojaED Extension was programmed to sample at every millisecond,

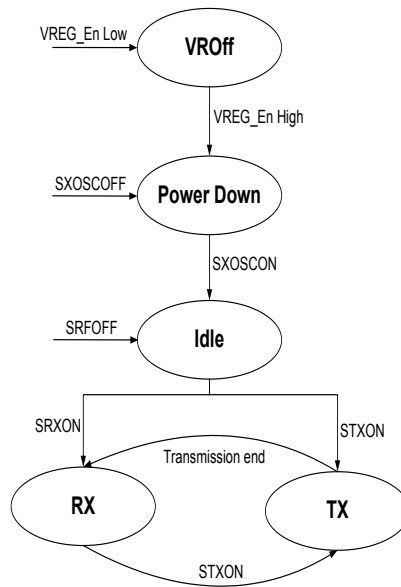


Figure 4.5: CC2420 radio control state machine simplified.

Simulation Time (ms)	Radio operation mode	Current consumption sample (mA)
1	RX	19.7
2	RX	19.7
3	IDLE	0.426
4	IDLE	0.426
5	RX	19.7
6	TX	17.4
7	TX	17.4
8	IDLE	0.426

Table 4.3: An example of the current consumption estimation for the radio when the CoojaED Extension was programmed to sample at every millisecond, and based on the radio operation mode, an electric current consumption is associated.

and the current consumption was extracted from the CC2420 datasheet. Based on the radio mode, the corresponding current consumption is associated, according to /the CC2420 datasheet, the current spent when the radio is in receive mode is 19.7mA.

The CoojaED Extension also tracks the current radio power transmission transition (see Listing C.2) since CC2420 can transmit packages with different transmission power, from -25dBm to 0dBm. It is very important to track the current radio power transmission since each power transmission has a different current consumption, which is shown in Table 4.4. The CC2420 class provides a method called

PA_LEVEL	TXCTRL register	Output Power (dBm)	Current Consumption (mA)
31	0xA0FF	0	17.4
27	0xA0FB	-1	16.5
23	0xA0F7	-3	15.2
19	0xA0F3	-5	13.9
15	0xA0EF	-7	12.5
11	0xA0EB	-10	11.2
7	0xA0E7	-15	9.9
3	0xA0E3	-25	8.5

Table 4.4: Output power configuration for the CC2420 radio [Sky, 2012] [CC2420-datasheet, 2012].

getOutputPower() that provides the current radio power transmission, which can be seen in Figure 4.3.

The CoojaED Extension not only calculates the sensor node energy awareness using current consumption values obtained from datasheet, but it can also compute the energy awareness based on experimental values. Chapter 5 describes the experimental setup to validate the CoojaED Extension where the sensor node current consumption was sampled during its battery discharge and saved in a log file. These current consumption values can be also used by the CoojaED Extension

4.3.2 Energy Awareness Model

The CoojaED Extension uses an energy model for the sensor node energy awareness developed by da Cunha [da Cunha, 2010] [da Cunha and da Silva, 2012]. The energy awareness E (mAh) is defined by equation 4.1

$$E = \frac{1}{3600} \times \left[\sum_{k=1}^n (I_k \times \Delta t) + \frac{1}{1000} \times \sum_{n_T=1}^m (I_f \times t_{off} \times n_T) \right] \quad (4.1)$$

where I_k is the current consumption sample (mA), Δt is the sampling period (s), n is the n th time interval of the n th current sample, I_f is the current consumption sample (μA) at instants where the current dropped to the lower level (I_f is a constant), n_T is the number of period (i.e. how much time the current was dropped to the lower level),

t_{off} is the time that the current of the sensor node was dropped to the lower level(s), and m is the n th period of the n th periods.

Equation 4.1 has two summations, the first calculates the energy consumption when the sensor node is on, and the second one computes the energy consumption when the load drops to a lower level. The energy model takes into account the low current consumption when the sensor node is idle. As the sensor node energy awareness is provided in mAh, the first and second terms are divided by 3600, in order to have the value expressed in mAh.

4.3.3 Energy availability

The implementation of the awareness of energy availability in simulators for WSN requires the modeling of the power unit of the sensor nodes. Then, to WSN with no capability of energy harvesting, awareness of energy availability consists in the use of battery models [da Cunha, 2010]. Battery models for WSN vary from a simple linear model to a complex one that take into account the relaxation effect.

4.3.4 Battery Model

The Behavioral Model of Alkaline Batteries (BMAB) [da Cunha, 2010] [da Cunha and da Silva, 2012] is a realistic and non-linear battery model for alkaline batteries that incorporates the relaxation effects, discharge rate and capacity retention.

BMAB takes into account the discharge rate in a more realistic approach. Instead of factoring the effective capacity for a given load to the maximum capacity or datasheet maximum capacity, the actual effective capacity is defined using the following equation:

$$C_{eff-f-a} = R_{c-a} \times C_{eff}(I_m) \quad (4.2)$$

where $C_{eff-f-a}$ is the final effective capacity of an alkaline battery, R_{c-a} is the capacity retention, I_m is the mean value of current samples during the first switch on the sensor node, and C_{eff} is the effective capacity for an I_m that is obtained from a lookup table (see Table B.1 and Listing C.4). The capacity retention is provided using the following equation:

$$R_{c-a} = 1 - P_c \quad (4.3)$$

where P_c is defined by:

$$P_c = P_{ca} \times n_a \quad (4.4)$$

where P_{ca} is the capacity loss due to temperature (in Celsius) and n_a is the alkaline battery shelf time (years). For each temperature, there is a corresponding capacity loss that is obtained from a lookup table (see Table A.1 and Listing C.5).

BMAB also takes into account the capacity recovery (see Listing C.6) due to the relaxation effect, that is defined by the following formula:

$$C_{rT} = \begin{cases} 0s & \text{if } t_{off} < 1s \\ 0,051 \times t_{off} + 0,445 & \text{if } 1s \leq t_{off} \leq 4s \\ 0,540 & \text{if } t_{off} > 4s \end{cases} \quad (4.5)$$

where C_{rT} is the capacity recovery per period ($\mu\text{Ah}/\text{T}$) and t_{off} is the time(s) the radio is either idle or in power down or VROff mode, the cpu is in LPMx mode, and all LEDs are off.

The battery remaining capacity (see Listing C.8) is provided by the following equation:

$$C_{res} = C_{eff-f-a} - \frac{1}{3600} \times \left[\sum_{k=1}^n (I_k \times \Delta t) + \frac{1}{1000} \times \sum_{n_T=1}^m (I_f \times t_{off} \times n_T) \right] + \frac{1}{1000} \times \left[\sum_{n_T=1}^m (C_{rT} \times n_T) \right] \quad (4.6)$$

where C_{res} is the battery remaining capacity (mAh), $C_{eff-f-a}$ is the final effective capacity of the alkaline battery, I_k is the current consumption sample (mA), Δt is the sampling period (s), n is the nth time interval of the nth current consumption sample, I_f is the current sample (μA) at instants where the current dropped to the lower level, n_T is the number of period, and m is the nth period of the nth periods.

The difference between equations 4.6 and 4.1 is the third sum (capacity recovery) and the battery effective capacity ($C_{eff-f-a}$). The accumulated actual energy consumption is subtracted from $C_{eff-f-a}$, and the result is added to capacity recovery, giving the battery remaining capacity C_{res} (i.e. the sensor node available energy). This means that BMAB estimates the sensor node lifetime based on the actual load minus the past energy consumption plus the capacity recovery.

Chapter 5

Experiments

We performed several experiments in order to validate the power consumption and available energy estimator. The estimator or BMAB battery model, uses as input the electrical current consumed for several WSN applications in a given period of time. To validate BMAB implementation in CoojaED extension, the input voltage is also sampled in order to compare the actual power consumption. As a result of these requirements, the experiment can be characterized according to the model presented in Figure 5.1 with the following parameters: inputs, outputs, controlled factors, and uncontrolled factors.

5.1 Design of Experiments Model

The validation of CoojaED extension, sensor nodes were equipped with fully charged batteries and the parameter measured is the time period in which the sensor nodes remain in operation running different applications (i.e. the sensor node lifetime). The sensor node lifetime is directly related to the battery voltage discharge rate. The power consumption is also measured, since it is directly related to the current drawn. The current gives an indication of the total power consumption and it can be easily measured by sampling the voltage drop across an i-v converter connected in series with the battery. Actual measurements provide an understanding of the battery discharge behavior under different loads and the voltage discharge profile.

Five applications were selected: Sens, Oscilloscope, Radio-Test, RSSI-Scanner, and TestAMOnOff. Sens, Oscilloscope and TestAMOnOff are TinyOS applications, and Radio-Test and RSSI-Scanner are Contiki applications. Sens is a typical WSN application where the sensor node reads the temperature sensor every second and transmits this value wirelessly. Oscilloscope is a simple data-collection demonstration

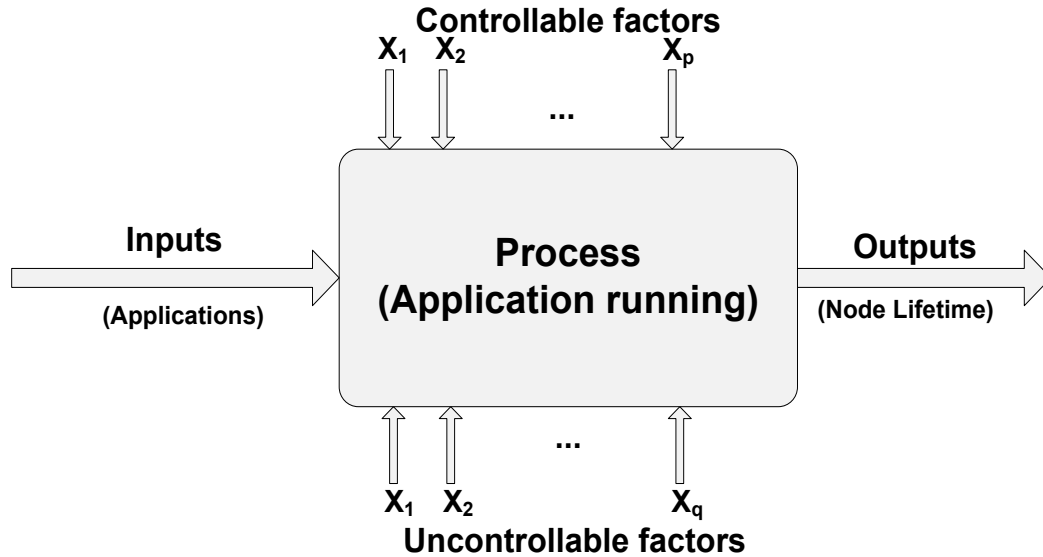


Figure 5.1: Experiment model (a schematic for a process with controlled inputs, outputs, controlled and uncontrollable factors).

application, and it periodically samples a given sensor and broadcast a message every ten readings. TestAMOnOff has two different versions, slave and master. The master sensor node is always on and broadcasts a packet every second. Four of five packets are data packets and one is a control packet. It toggles the red LED when it sends a data packet and the green LED when it sends a control packet. When a slave sensor node receives a data packet, it toggles the red LED, when and it receives a control packet, it turns off its radio for a short and random period of time. The green LED indicates whether the slave radio is on. Oscilloscope and TestAMOnOff are available in the TinyOS examples directory while Sens is available by Lajara et al.[Lajara et al., 2010]. Radio-Test is a simple application to indicate connectivity between two sensor nodes. At every second the sensor node sends a packet and toggles its red LED. The other sensor node toggle its green LED whenever it can hear other sensor nodes but not necessary vice versa (i.e unidirectional communication). The blue LED indicates that both sensor nodes can communicate to each other (i.e. bidirectional communication). RSSI-Scanner scans the 2.4 GHz radio frequency range using the

radio channel and prints the RSSI values. These applications are available in the Contiki examples directory.

Some parameters can potentially affect the application lifetime such as the transmission power (the power level at which the radio transmits a bit) and the distance (distance between the sender and receiver), however, these parameters can be controlled (i.e. they are controllable factors). The power transmission was programmed to be 0dBm for all applications and the distance between sensor nodes was fixed in approximately 25cm in order to reduce collisions, packet loss, interferences and attenuation. It is acknowledged that temperature can also affect battery lifetime, but in these tests, the temperature was stable. All tests were conducted at the LabSCI laboratory with a temperature approximately 21°C . The temperature is also a controllable factor. However, there are other factors that cannot be easily controlled (i.e. uncontrollable factors) such as sensor node and data acquisition failures.

5.2 Experimental Setup

The experimental setup for measuring the sensor node lifetime is presented in Figure 5.2. The setup includes two NI-USB 6216 DAQ boards [Instruments, 2012], LabVIEW SignalExpress 2010 instrumentation software, one power supply, one protoboard, five Tmotes Sky, several MN1604 alkaline batteries, one the LM35 temperature sensor [LM35, 2012], two i-v converters, one multimeter and one computer.

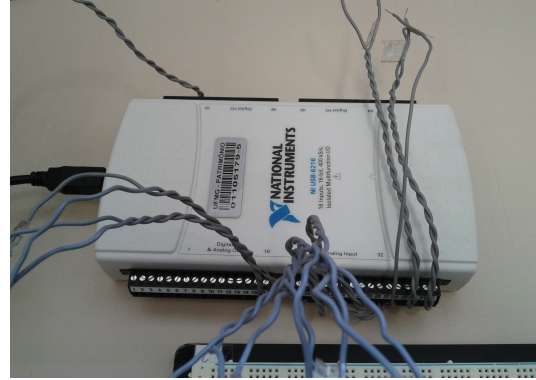


Figure 5.2: The experimental setup bench.

The sensor nodes and the NI-USB 6216 DAQ are show in Figure 5.3a and Figure 5.3b, respectively.



(a) Tmote Sky sensor nodes.



(b) NI-USB 6216 DAQ.

Figure 5.3: Tmote Sky sensor nodes and NI-USB 6216 DAQ.

A Duracell MN1604 9V battery is composed of six 1.5V alkaline batteries. One such battery was open and the cells were connected in groups of two cells with 3V potential (2×1.5 each), which can be seen in Figure 5.4.



(a) The Duracell MN1604 9V battery.



(b) The cells connected in groups of two cells with 3V potential.

Figure 5.4: A MN1604 Duracell 9V battery composed of six 1.5V alkaline batteries was open and connected in groups of two cells with 3V potential.

The i-v converters have a 1 ohm value and was placed in series between the alkaline battery and the sensor node in order to measure the current consumption. The battery voltage is also sampled, as illustrate in the Figure 5.5. The NI USB-6216 was connected to the computer via USB interface. The NI USB-6216 is a DAQ with 16-bit accuracy and capable of 400 kSamples/s and was programmed to sample at every millisecond. The DAQ was used to measure the voltage across the i-v converter with a relation of 1mV to 1mA. The DAQ was also programmed to measure the battery voltage to calculate the battery remaining capacity and the sensor node lifetime. The LabVIEW SignalExpress 2010 is an interactive and data-logging software for DAQ devices that was used to collect and save a log file with the voltage across the i-v converter, battery voltage, sensor node voltage, temperature and the time stamp.

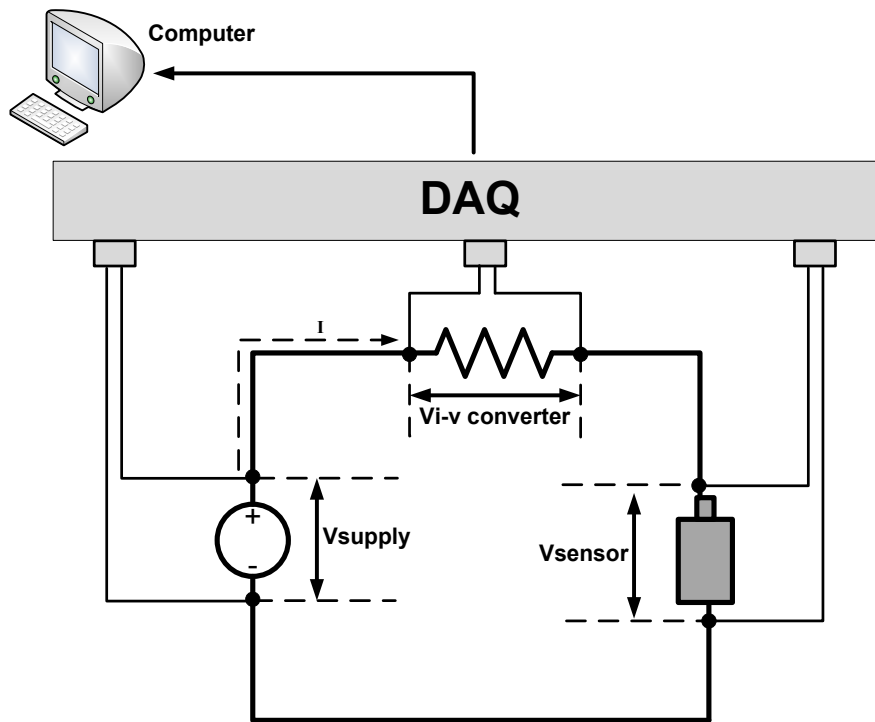
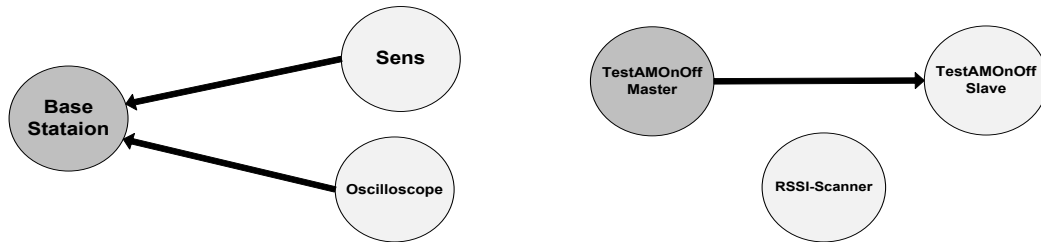


Figure 5.5: Measurement of battery voltage and sensor node voltage and current

5.3 Case Studies

To validate the CoojaED extension, three case studies were developed, and are shown in Figure 5.6.



(a) A WSN composed by three sensor nodes: a base station application node, Sens application node, and Oscilloscope application node.

(b) A WSN formed by three sensor nodes: RSSI-Scanner application node, master and slave TestAMOnOff application nodes.

(c) A WSN composed by two Radio Test application nodes.

Figure 5.6: Case Studies

The first case study consists of a WSN composed by three sensor nodes: a base station, Sens, and Oscilloscope nodes, as shown in Figure 5.6a. The Sens node reads the temperature sensor every second and transmits to the base station node, and the Oscilloscope node samples the default sensor every second and transmits a message every ten readings to the base station node.

The second case study consists of a WSN formed by three sensor nodes: RSSI-Scanner, TestAMOnOff master and slave nodes. The RSSI-Scanner node scans the 2.4 GHz radio frequency range using the radio and prints the RSSI values. The master sensor node broadcasts a packet every second and keeps its radio always on. When the slave node receives a data packet, it toggles the red LED, and when a slave node receives a control packet, it turns off its radio for a short random interval.

The third case study consists of a WSN composed by two Radio Test nodes. Every second both Radio-Test node 1 and node 2 transmit a packet and toggles their

red LED, if one of them can hear the other, this node toggle its green LED. If both nodes can communicate to each other, they toggle their blue LED.

The Sens and Oscilloscope, RSSI-Scanner, TestAMOnOff slave, and Radio Test nodes were connected in series with a 1ohm i-v converter and the alkaline battery as the diagram shown in Figure 5.5. For each applications, two experiments were performed and data were collected until the battery voltage dropped to 1.5V.

5.4 Sensor Node Lifetime

The sensor node cutoff voltage is defined as the minimum supply voltage that permits the sensor node to be fully operational. It is usually defined as the minimum operating voltage for the communication unit. Battery cutoff voltage affects the time period in which the battery delivers power. Battery cutoff voltage is the lowest operating voltage at which the cell is then considered depleted. Figure 5.7 presents four cutoff voltages (7V, 6V, 5.4V, and 4.8V) of a typical constant current discharge characteristics at 21 °C for a MN1604 battery. Each MN1604 has six cells and a nominal voltage of 9V. Dividing each cutoff voltage by 6, then each cutoff voltage for a cell is 1.17V, 1V, 0.9V and 0.8V, respectively. The cutoff voltage for two cells are 2.3V, 2V, 1.8V and 1.6V, respectively.

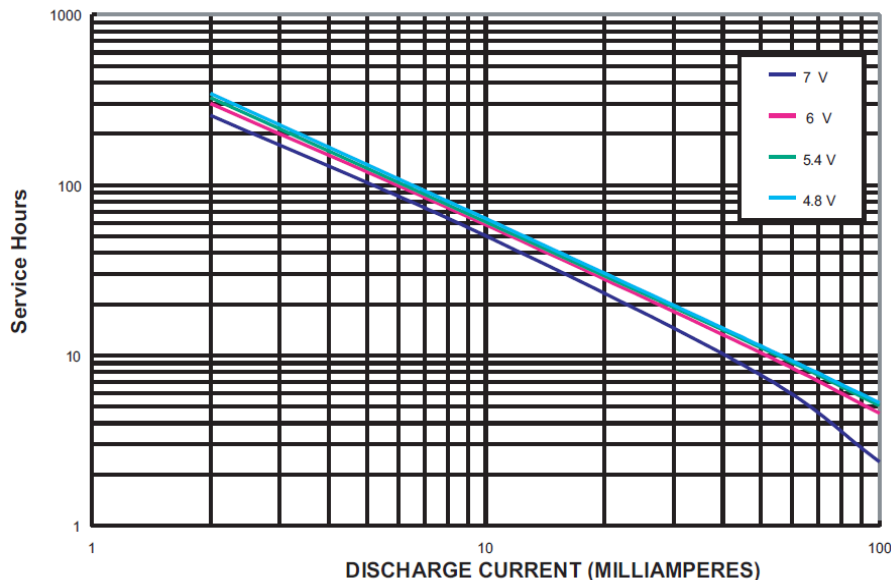


Figure 5.7: MN1604 battery typical constant current discharge characteristics at 21°C [MN1604, 2012].

In order to define the sensor node lifetime, the relationship between voltage and current for each application is plotted over the time, the voltage curve plotted in dV (decivolts volts), i.e. a value of 10 in the graph means 1V, and the current curve is plotted in mA. The Sens, Oscilloscope, Radio Test, RSSI-Scanner, and TestAMOnOffSlave applications power behaviors are plotted along with a zoomed portion of the current plot to show the detailed current profile. The current consumption is also plotted over a very short time interval in order to detailed the current profile for each application, which is shown Figure 5.9, Figure 5.11, Figure 5.13, Figure 5.15, and Figure 5.17.

Figure 5.8 shows the voltage and current profiles for the Sens application and Figure 5.10, Figure 5.12, Figure 5.14 and Figure 5.16 show the power behaviors for the remaining applications. All voltage curves show a similar behavior close to 1.8V, where the supply drops to 1.5V and the current curve loses its usual behavior. This event defines the sensor node cutoff voltage value.

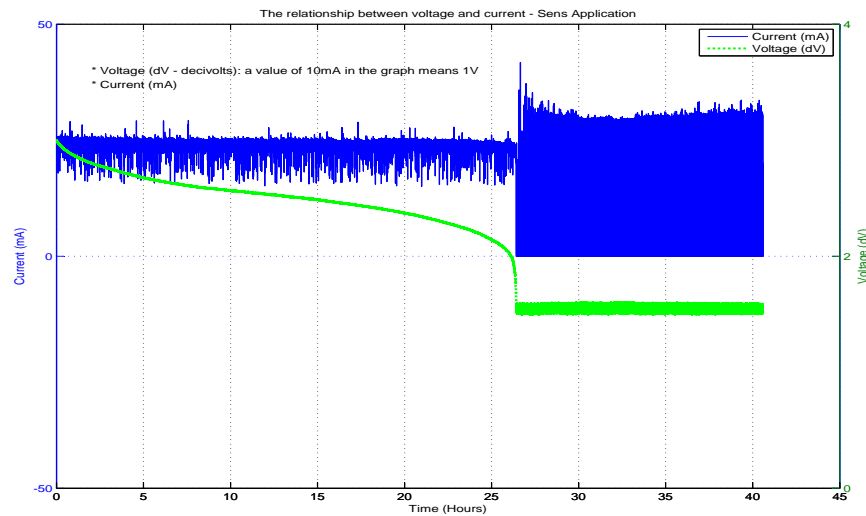


Figure 5.8: Sens application voltage and current profiles

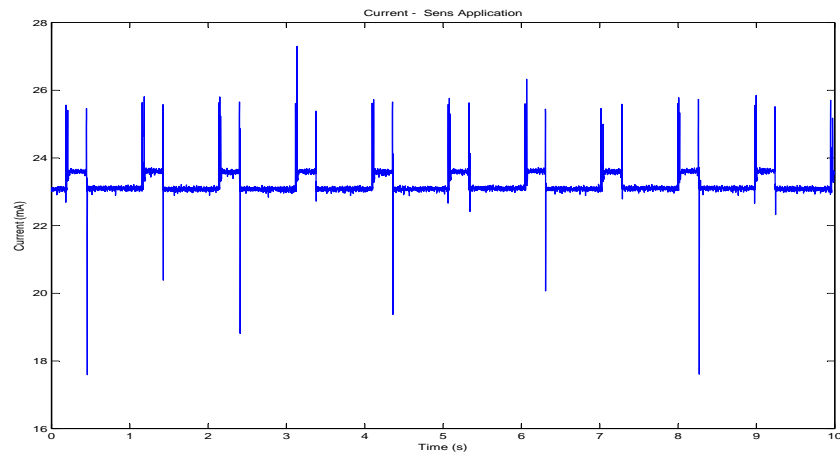


Figure 5.9: Sens application zoomed portion of current consumption plotted over a time interval of 10 seconds.

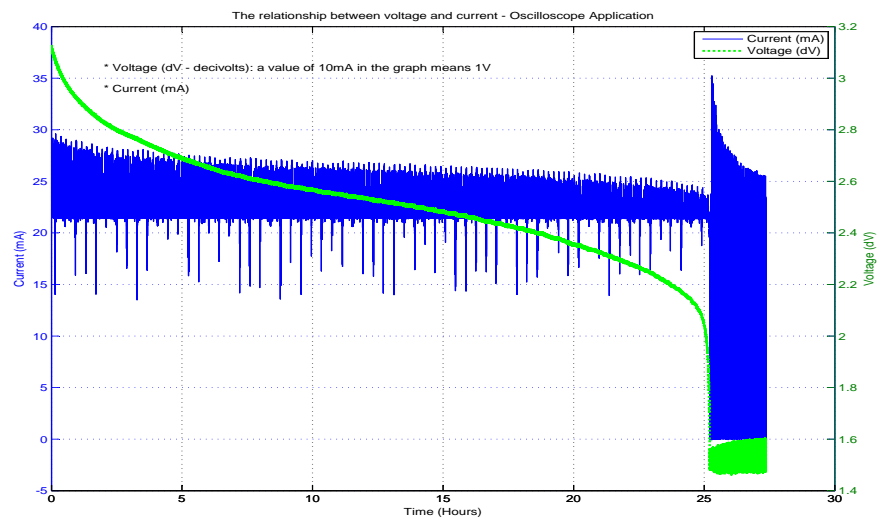


Figure 5.10: Oscilloscope application voltage and current profiles

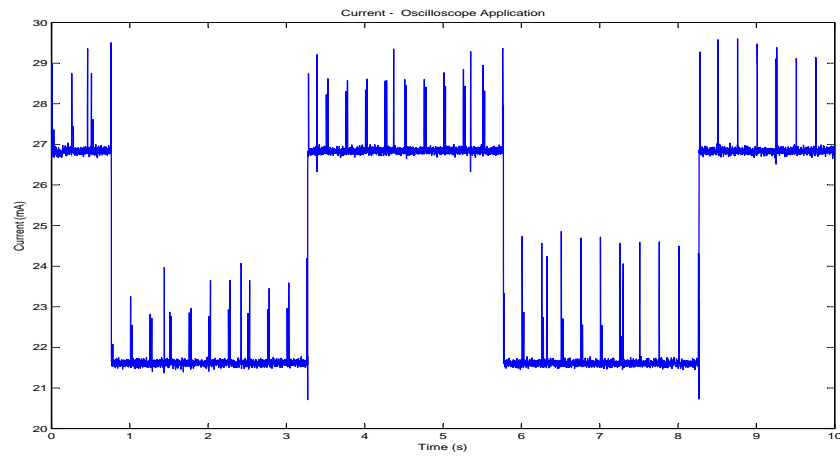


Figure 5.11: Oscilloscope application zoomed portion of current consumption plotted over a time interval of 10 seconds.

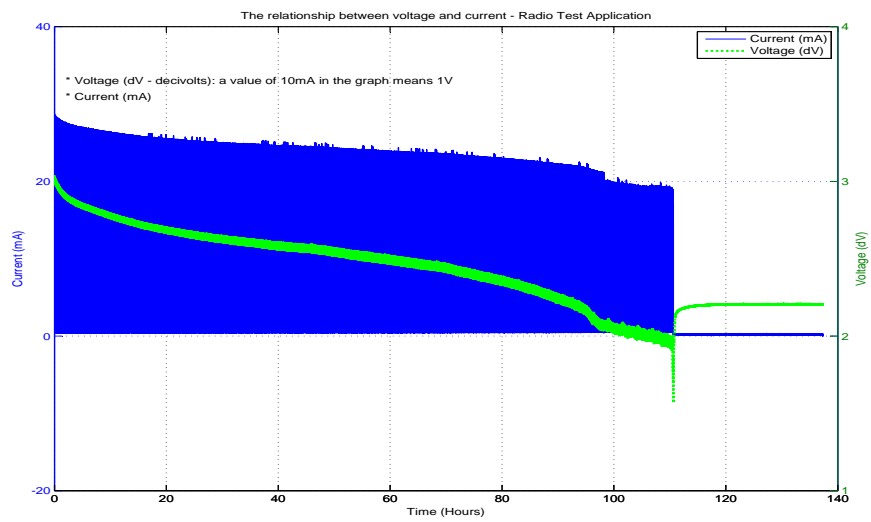


Figure 5.12: Radio Test application voltage and current profiles.

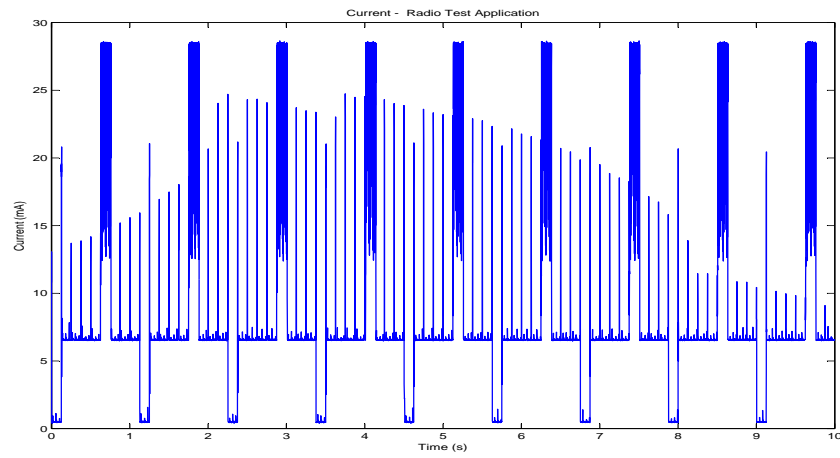


Figure 5.13: Radio Test application zoomed portion of current consumption plotted over a time interval of 10 seconds.

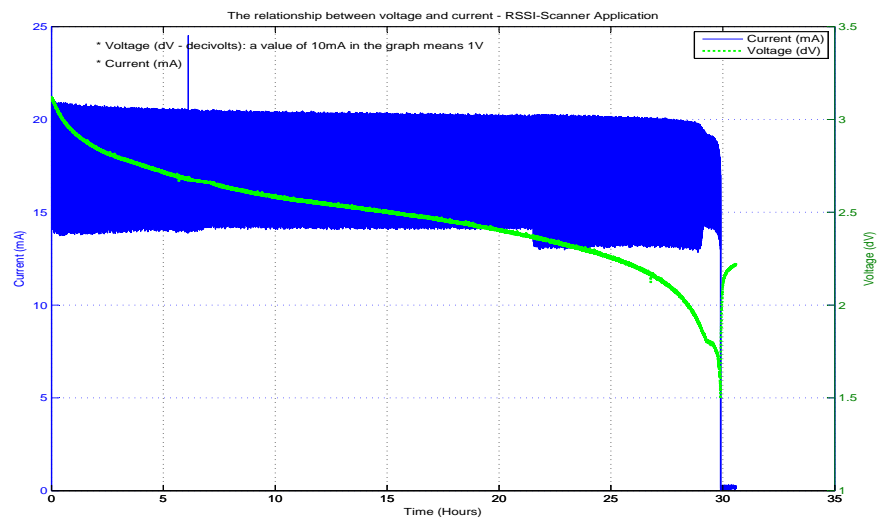


Figure 5.14: RSSI-Scanner application voltage and current profiles.

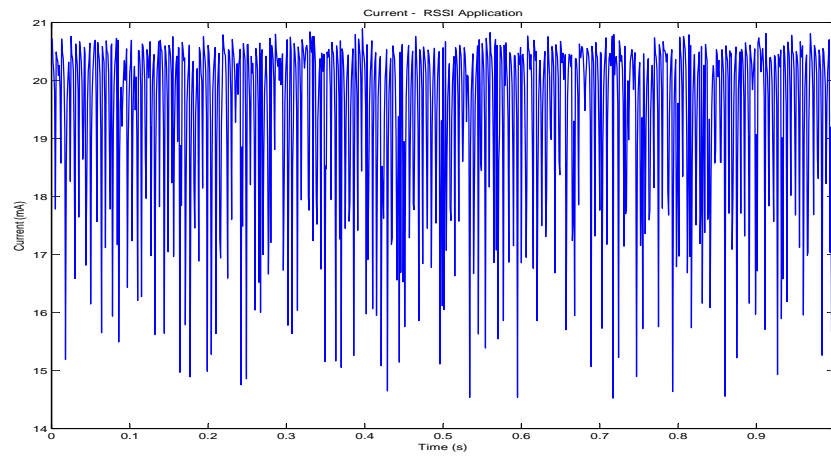


Figure 5.15: RSSI application current zoomed portion of consumption plotted over a time interval of 1 seconds.

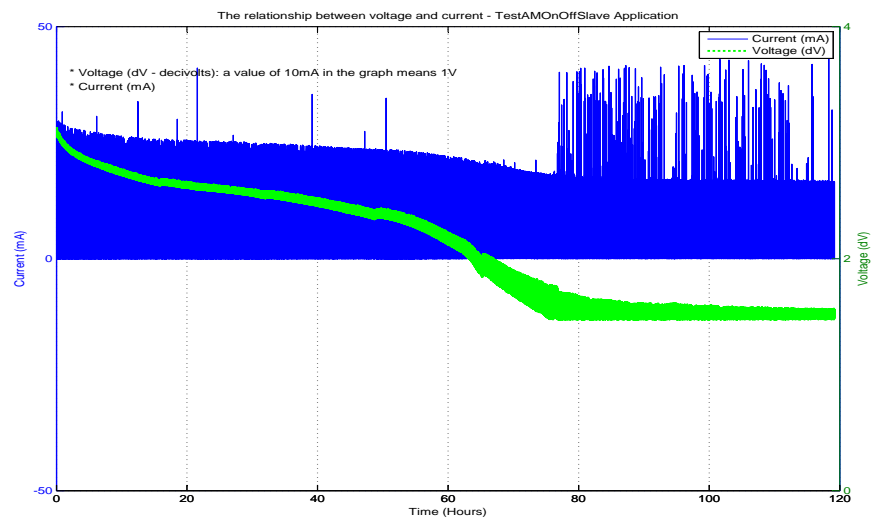


Figure 5.16: TestAMOnOffSlave application voltage and current profiles.

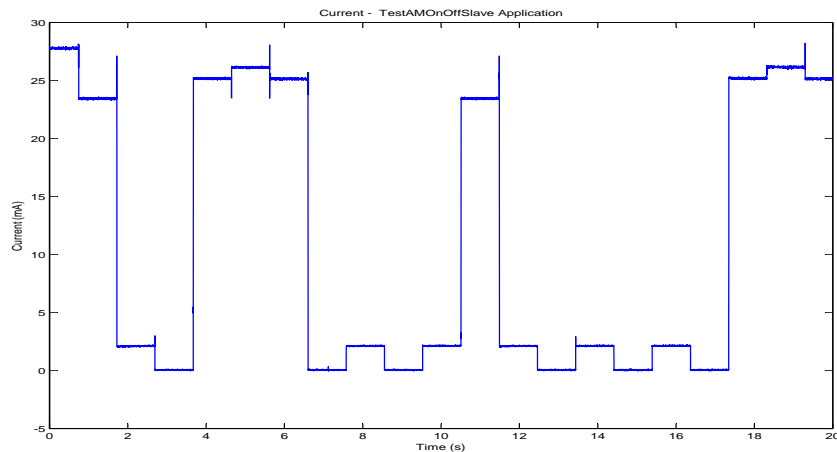


Figure 5.17: TestAMOnoffSlave application zoomed portion of current consumption plotted over a time interval of 20 seconds.

Applications	Voltage cutoff = 1.8 V	Voltage cutoff = 2.1 V
Sens 1	25:37:12	25:11:24
Sens 2	26:22:12	25:25:12
Oscilloscope 1	25:10:48	24:45:36
Oscilloscope 2	26:03:36	25:21:00
Radio Test 1	108:17:24	94:25:48
Radio Test 2	110:32:24	95:51:00
RSSI-Scanner 1	28:53:24	26:46:48
RSSI-Scanner 2	29:16:12	27:31:12
TestAMOnOffSlave 1	66:59:24	60:37:12
TestAMOnOffSlave 2	69:33:36	62:21:36

Table 5.1: Sensor node lifetime (hh:mm:ss) with different sensor node cutoff voltage, 1.8V and 2.1V.

According to the Tmote Sky datasheet, the sensor node and radio operating voltage range is from 3.6V to 2.1V. For this reason, the sensor node lifetime is also computed for a cutoff voltage equal to 2.1V, and Table 5.1 presents a comparison of sensor node lifetimes with cutoff voltages of 1.8V and 2.1V.

During the Radio Test application experiments some LEDs stopped to blink when the battery voltage dropped to 2.1V. After the battery voltage dropped below 2.1V, the radio was still working. Since toggling LEDs is not a very important task in WSN applications, the actual sensor node cutoff voltage can be considered as 1.8V.

5.5 Simulations

To validate CoojaED extension, all applications described above were simulated using the same configuration of the case studies. The power transmission was fixed in 0dBm, the distance between sensor nodes was approximately 25cm and the UDGM (Unit Disk Graph Medium) propagation model was used. UDGM models the transmission range as an ideal circle in which all the sensor nodes outside that circle do not receive packets while the sensor nodes within that circle receive all the packets. For these simulations the current consumption values for the cpu, radio and LED were extracted from datasheets, the battery voltage was fixed as 3V, and the current consumption value for each LED was defined as 2.1mA.

Each case studies application was simulated using CoojaED extension and the value of power consumption and available energy were calculated. Three simulations were made for each application using different current consumption values, resulting in three simulated sensor node lifetime results, which is shown in Table 5.5. For each application, one of these results was selected to validated the CoojaED extension.

Operation mode	Current Consumption (mA)
Radio RX and CPU on	23
Radio TX and CPU on	21
Radio idle and CPU on	2.4
Radio RX and CPU low power mode	19.5
Radio TX and CPU low power mode	21.8

Table 5.2: Maximum current consumption values of Tmote Sky for 3V power supply.

Operation mode	Current Consumption (mA)
Radio RX and CPU on	21.8
Radio TX and CPU on	19.5
Radio idle and CPU on	1.8
Radio RX and CPU low power mode	18.3
Radio TX and CPU low power mode	20.6

Table 5.3: Nominal current consumption values of Tmote Sky for 3V power supply.

The first simulation used the current consumption values shown in Table 5.2. These values represents the maximum current consumption for each operation mode of radio and CPU, and these values were extracted from Tmote Sky datasheet. The second simulation used the current consumption values shown in Table 5.3. These

CC240 Radio	MSP430 Microcontroller
Voltage regulator: 0.02 μA	Active mode: 500 μA
Power down mode: 20 μA	LPM0 mode: 75 μA
Idle mode: 426 μA	LPM2 mode: 17 μA
Receive mode: 19.7mA	LPM3 mode: 2.6 μA
Transmit mode (0dBm): 17.4mA	LPM4 mode: 0.2 μA
	LPM1 mode: 80 μA

Table 5.4: Nominal current consumption values of the CC2420 radio and the MSP430 microcontroller for 3V power supply.

values represents the nominal current consumption for each operation mode of radio and CPU, and they were also extracted from Tmote Sky datasheet. The third simulation used the current consumption shown in Table 5.4. The values were extracted from the CC2420 radio datasheet and the MSP430 datasheet, and they represent the nominal current consumption for each operation mode of radio and CPU, respectively.

Applications	Sim. 1 (hh:mm:ss)	Sim. 2 (hh:mm:ss)	Sim. 3 (hh:mm:ss)
Sens	26:34:48	29:13:48	30:28:12
Oscilloscope	26:05:24	27:42:00	29:00:36
Radio Test	90:57:00	96:27:00	94:01:12
RSSI-Scanner	26:09:00	27:37:48	29:43:12
TestAMOnOffSlave	53:06:36	63:37:12	66:30:00

Table 5.5: Simulated sensor node lifetime for Tmote Sky platform.

CoojaED extension can also use current consumption values measured through a data acquisition module, such as the experimental setup explained above, then the sensor node current consumption values that were sampled and saved in a log file are also used by the CoojaED extension. These results are presented in Chapter 6.

Chapter 6

Results and Validation

This Chapter describes the results and model validation of the proposed CoojaED extension. Validation of simulation models means determining whether the simulation model is an acceptable representation of the real system [Kleijen, 1999]. A comparison between the real system (i.e. measuring the sensor node lifetime using an experimental setup) and the simulation model (the CoojaED extension) is made. A comparison means checking the simulation model output behavior to either the system output behavior or another reference model output behavior using graphical displays or statistical tests and procedures [Sargent, 1998]. With this validation approach and the results presentation, the BMAB battery model and its implementation are validated.

6.1 Overview

There are some approaches used in comparing the simulation model output and the real system output such as the use of graphs, tables, and statistical techniques. A comparison means checking the simulation model output behavior to the real system output behavior using graphical displays, tables or statistical tests [Sargent, 1998].

Several types of graphs can be used, such as histograms, box plots, and scatter plots. Statistical techniques are also usually applied for the validation of simulation models. They depend on the availability of data of the real system, and there are three possible situations: no real data are available; there is only real data on the output; and both output and input data are known (trace-driven) [Kleijen, 1999] [Sargent, 1998].

In this work, real data input (applications) and output (sensor node lifetime) are known. Trace-driven simulation means to feed real input data into the simulation model (usually a computer program), in historical order. After running the simulation

program, we compared the time series of simulated output with the time series of the real output. [Kleijen, 1999].

In trace-driven simulation, a particular kind of regression analysis is applied. Hypothesis tests can also be used in the comparison of means and variances of the output data of the simulation model and the real system for a set of experiments to determine if the simulation models output behavior has an acceptable range of accuracy. An acceptable range of accuracy is the amount of accuracy that is required of a model to be valid for its intended purpose [Sargent, 1998].

A comparison between the real system and simulation model (i.e. CoojaED extension using electric current consumption values that can be obtained from both the datasheets and from an experimental setup) is made. For each WSN application, CoojaED extension was set to calculate the available energy (mAh) at every ten seconds interval during the sensor node lifetime time interval (hours) with a sensor node cutoff voltage equal to 1.8V.

In the rest of this work, the name **Cooja1** will be used to refer to the CoojaED extension using electric current consumption values obtained from the experimental setup described in the previous chapter, and the name **Cooja2** will be used to refer to the CoojaED extension using electric current consumption values obtained from datasheets.

6.2 Total Effective Capacity and Remaining Capacity

6.2.1 Total Effective Capacity

A major constraint in modeling the battery model is to determine the battery effective capacity (i.e. the capacity for a given effective voltage of battery) during an application execution, since the battery initial voltage is not fixed, the voltage battery curve is not linear, and is not exactly 1.5V for alkaline batteries. Besides, there is a charge recovery phenomena due to the relaxation effect. Table 6.1 presents the initial voltage of a set of two alkaline batteries in series used in each of the experiments. As can be seen, it varies from 2.99V to 3.13V.

Application	Initial Voltage (V)
Sens 1	2.998878
Sens 2	3.007412
Oscilloscope 1	3.066985
Oscilloscope 2	3.119681
Radio Test 1	3.018767
Radio Test 2	3.030478
RSSI-Scanner 1	3.085252
RSSI-Scanner 2	3.113558
TestAMOnOffSlave 1	3.108292
TestAMOnOffSlave 2	3.131660

Table 6.1: Battery initial voltage.

The total effective capacity is the total energy supplied by the battery to the sensor node. It is measured in milliampere-hour (mAh), which is the number of hours the battery can supply a certain amount of current before its voltage drops below a predetermined threshold value (1.8V for two alkaline batteries in series). In reality the total effective capacity is the initial battery capacity plus the eventual recovered capacity.

The real total effective capacity is computed through the method known as coulomb counting, where it calculates the battery final effective capacity by integrating the sensor node current over time. The estimated final effective capacity is provided by the BMAB battery model parameter $C_{eff-f-a}$. An inaccurate final effective capacity estimative $C_{eff-f-a}$ influence the remaining capacity result (C_{res}), since the accumulated energy consumption is subtracted from $C_{eff-f-a}$, and the result is added to the recovered capacity, giving the battery remaining capacity (C_{res}).

Table 6.2 presents the measured and estimated (Cooja1) total effective capacity estimative for each application and their respective differences (absolute and percentual) to the measured total effective capacity. Contiki applications show a large difference between measured and Cooja1 total effective capacity. In particular, RSSI-Scanner 1 and RSSI-Scanner 2 applications show a percentual difference of more than 9%, as can be seen on the fourth column of Table 6.2. TinyOS applications show a difference of less than 3%, and Oscilloscope applications present the smallest difference.

Table 6.3 presents the measured and estimated (Cooja2) total effective capacity estimative for each application and their respective differences (absolute and percentual) to the measured total effective capacity. Contiki applications also show a large difference between measured and Cooja2 total effective capacity. RSSI-Scanner

Application	Real (mAh)	Cooja1 (mAh)	Difference (mAh)	Difference (%)
Sens 1	592.61	601.34	8.73	1.47
Sens 2	609.91	601.52	8.39	1.38
Oscilloscope 1	577.13	583.39	6.26	1.08
Oscilloscope 2	582.29	588.84	6.19	1.06
Radio Test 1	548.64	576.18	27.54	5.02
Radio Test 2	555.36	576.81	21.45	3.86
RSSI-Scanner 1	545.41	601.98	56.57	10.37
RSSI-Scanner 2	551.32	601.52	50.2	9.11
TestAMOnOffSlave 1	583.55	597.07	13.52	2.32
TestAMOnOffSlave 2	588.06	601.31	13.25	2.25

Table 6.2: Comparison between measured and Cooja1 total effective capacity.

Application	Real (mAh)	Cooja2 (mAh)	Difference (mAh)	Difference (%)
Sens 1	592.61	601.68	9.07	1.53
Sens 2	609.91	601.68	8.23	1.35
Oscilloscope 1	577.13	596.63	19.5	3.38
Oscilloscope 2	582.29	596.63	14.34	2.46
Radio Test 1	548.64	582.96	34.32	6.26
Radio Test 2	555.36	582.96	27.6	4.97
RSSI-Scanner 1	545.41	600.55	55.14	10.11
RSSI-Scanner 2	551.32	600.55	49.23	8.93
TestAMOnOffSlave 1	583.55	593.03	9.48	1.62
TestAMOnOffSlave 2	588.06	593.03	4.97	0.85

Table 6.3: Comparison between actual and Cooja2 total effective capacity.

applications present a percentual difference of more than 8%, which is also show on the fourth column in Table 6.3. TinyOS applications show a percentual difference of less than 3.5%, and TestAmOnOffSlave 2 present the smallest difference.

6.2.2 Remaining Capacity

In this section, we present the remaining capacity values for each application. The remaining capacity is plotted against the sensor node lifetime time interval (hours), and we compare the measured, Cooja1 and Cooja2 remaining capacity curve.

Figure 6.1 presents Sens 1 application remaining capacity plots curves. The measured (reference) and Cooja1 estimative curves have the same linear profile. Cooja2 estimative curve presents a slightly smaller inclination than the measured curve. Both Cooja1 and Cooja2 indicate that there is available energy (8.66mAh and 42.87mAh, respectively, from Table 6.4) when the battery stopped to supply the energy to the sensor node, i.e., the battery reached its defined battery cutoff voltage (1.8V).

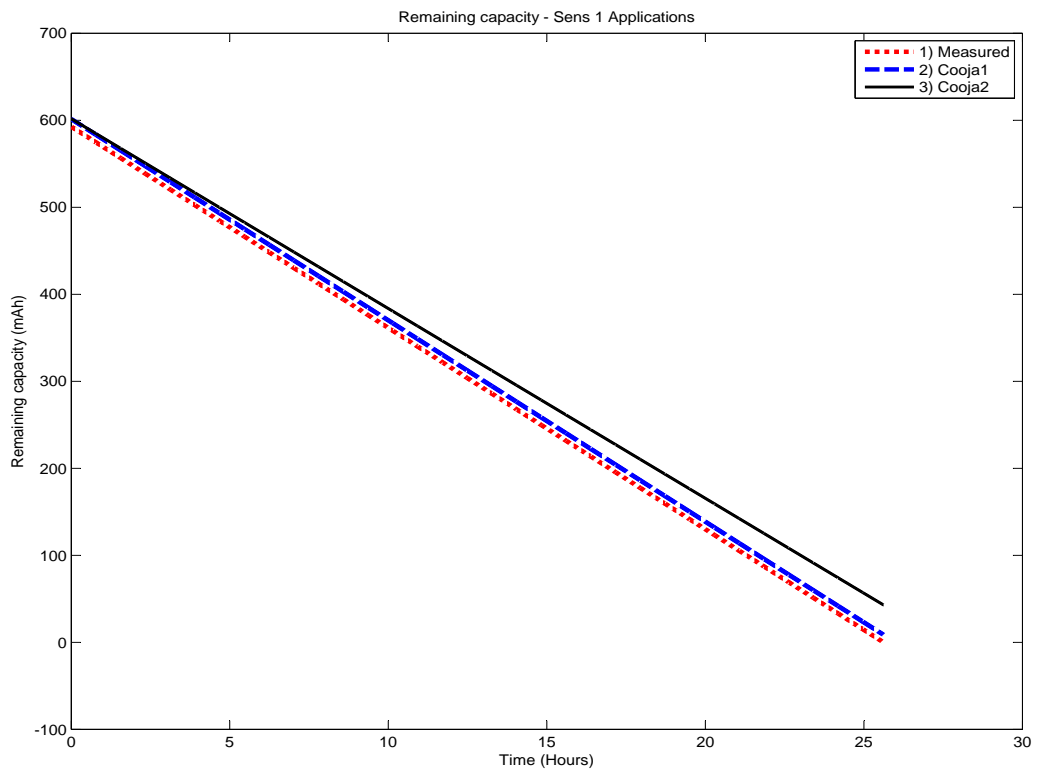


Figure 6.1: Remaining capacity for Sens 1 application.

Figure 6.2 shows Sens 2 application remaining capacity plots curves. Cooja1 estimative curve profile as the same as the measured curve while Cooja2 estimative curve present a slightly smaller inclination than the measured curve. Cooja1 provides a result that indicates that there is not available energy when the sensor node is still working (-8.46mAh). In contrast, Cooja2 indicates that there is available energy (26.51mAh) when the battery stopped to supply energy to the sensor node.

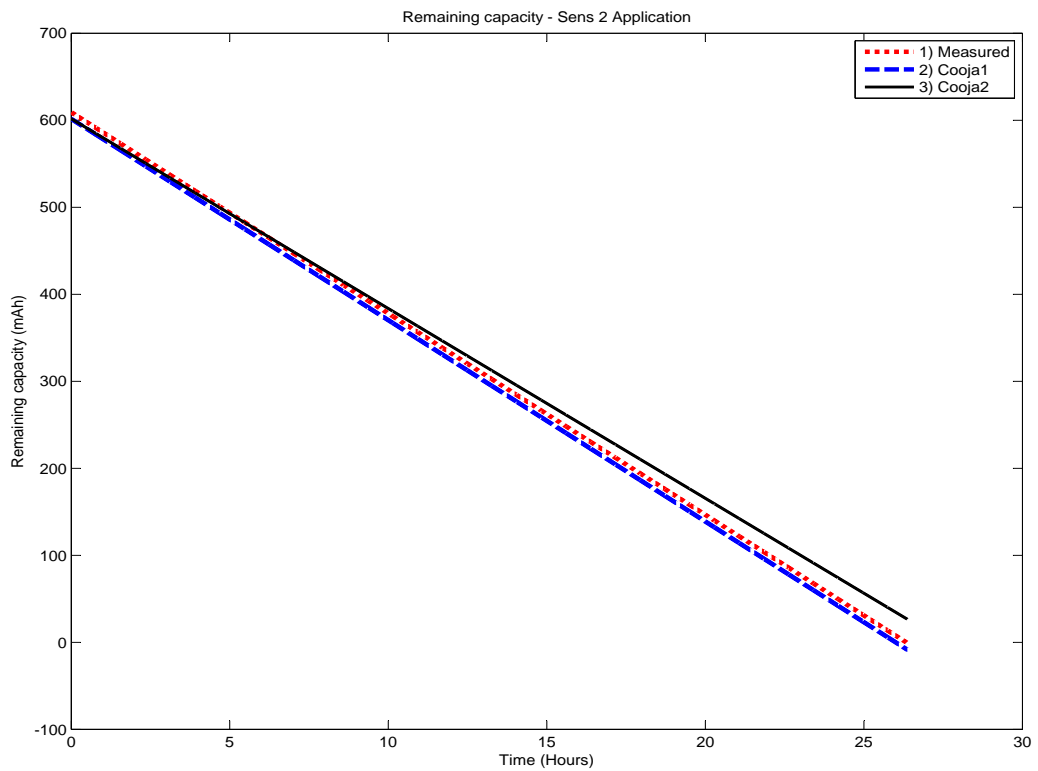


Figure 6.2: Remaining capacity for Sens 2 application.

Figure 6.3 presents Oscilloscope 1 application remaining capacity plots curves. The measured and the both estimative curves present the same inclination. Cooja1 is very close the measured curve, while Cooja2 is a bit far from the measured one. Both Cooja1 and Cooja2 indicate that there is available energy (6.19mAh and 20.9mAh, respectively) when the battery stopped to supply the energy to the sensor node.

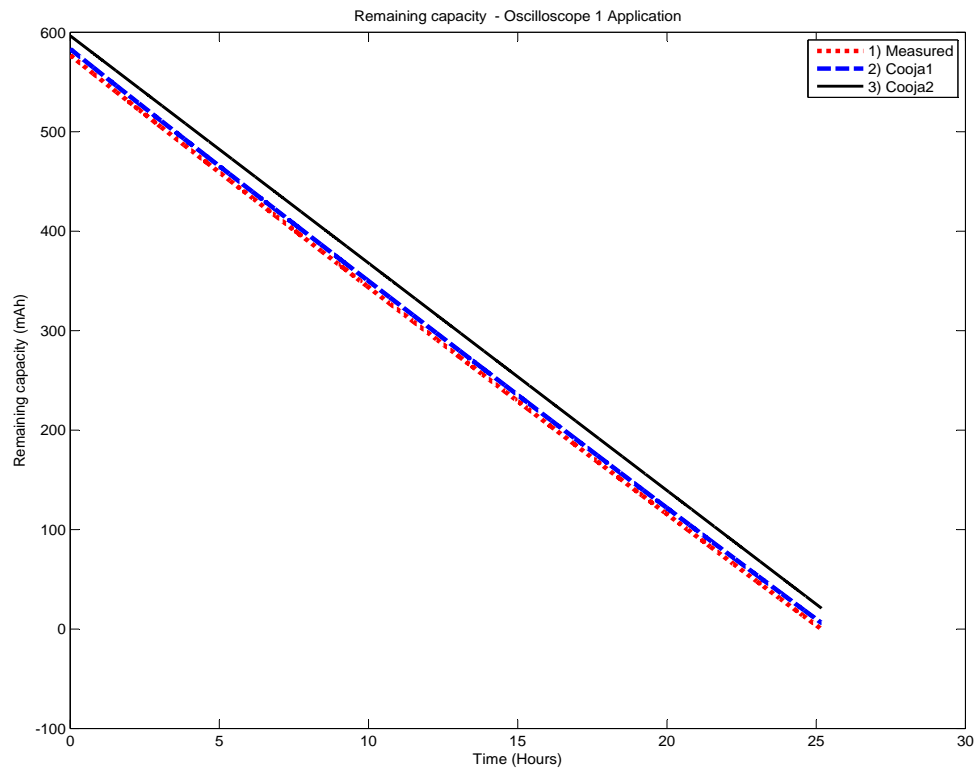


Figure 6.3: Remaining capacity for Oscilloscope 1 application.

Figure 6.4 shows Oscilloscope 2 application remaining capacity plots curves. The measured and the both estimative curves present the same inclination. Cooja1 is very close the measured curve, while Cooja2 is a bit far from the measured one. Both Cooja1 and Cooja2 indicate that there is available energy (6.74mAh and 0.76mAh, respectively) when the battery stopped to supply energy to the sensor node.

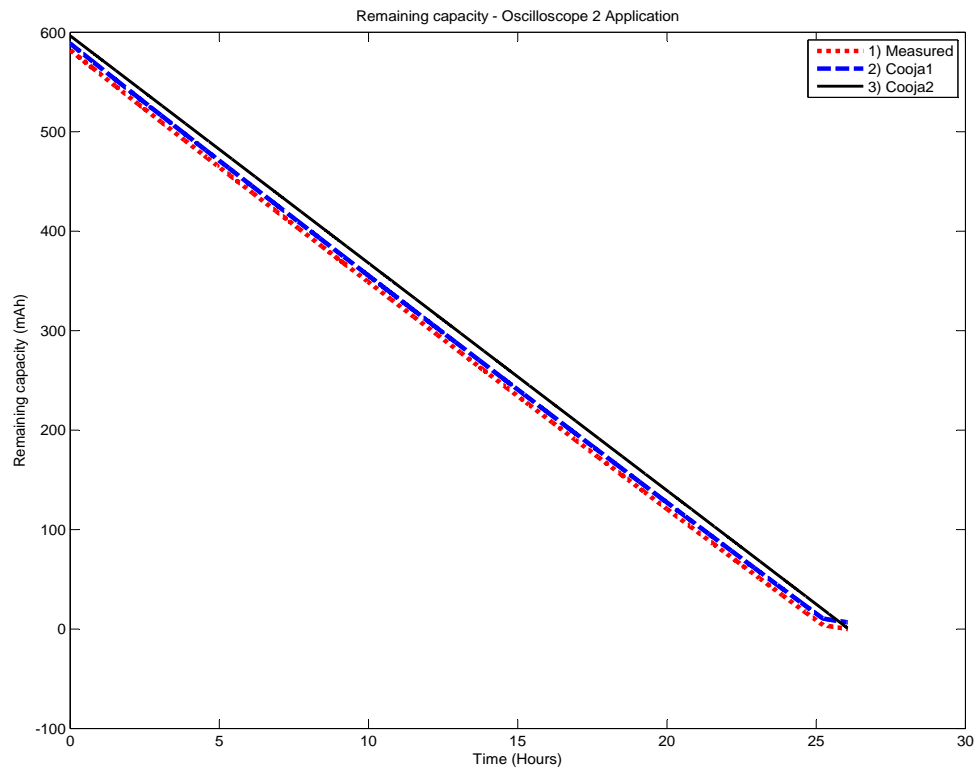


Figure 6.4: Remaining capacity for Oscilloscope 2 application.

Figure 6.5 presents RSSI-Scanner 1 application remaining capacity plots curves. The measured and Cooja1 estimative curves have the same linear profile, and Cooja1 estimative is very far from the measured one. Cooja2 curve presents a stronger inclination than the measured curve. In regard to final remaining capacity estimative, Cooja2 present errors lower than Cooja1, however, Cooja1 is a better approximation than Cooja2, since for every point in the curve the distance to the measured one is approximately fixed. A simple calibration can provide an accurate estimative. Both Cooja1 and Cooja2 estimatives indicate that there is available energy (56.48mAh and 16.86mAh) when the battery stopped to supply the energy to sensor node.

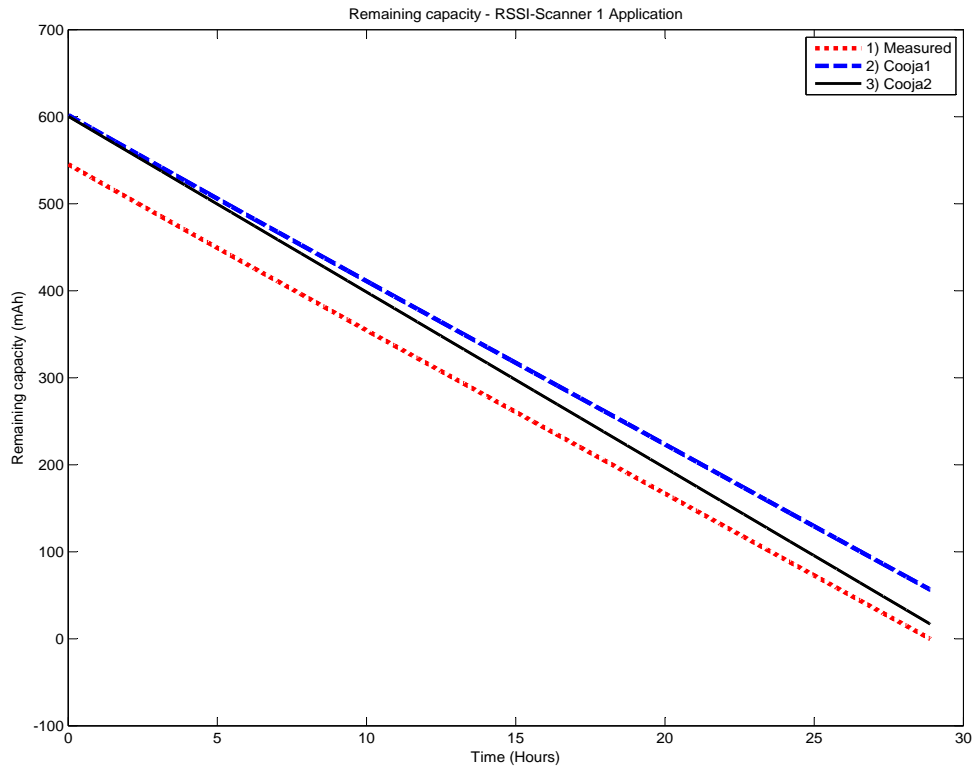


Figure 6.5: Remaining capacity for RSSI-Scanner 1 application.

Figure 6.6 shows RSSI-Scanner 2 application remaining capacity plots curves. Cooja1 curve profile is the same as the measured curve while Cooja1 estimative is very far from the measured one. Cooja2 curve presents a stronger inclination than the measured curve. In regard to final remaining capacity estimative, Cooja2 present errors lower than Cooja1, however, Cooja1 is a better approximation than Cooja2, since for every point in the curve the distance to the measured one is approximately fixed. A simple calibration can provide an accurate estimative. Cooja2 curve present a stronger inclination than measured curve. Both Cooja1 and Cooja2 estimative indicate that there is available energy (49.02mAh and 9.18mAh, respectively) when the battery stopped to supply energy to the sensor node.

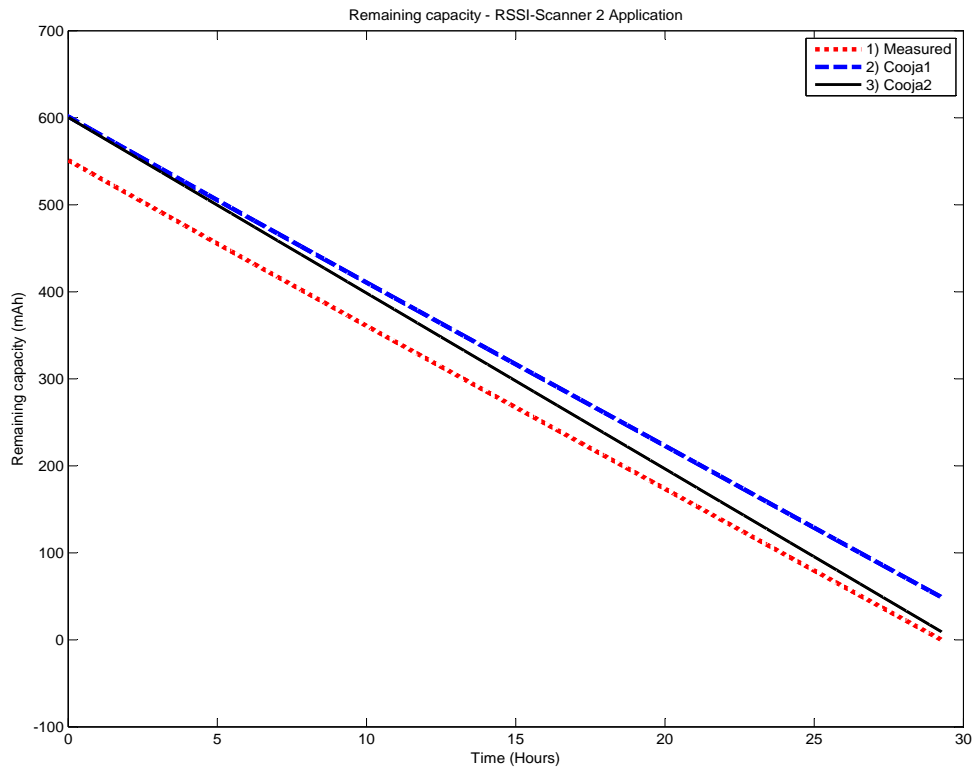


Figure 6.6: Remaining capacity for RSSI-Scanner 2 application.

Figure 6.7 shows Radio Test 1 application remaining capacity plots curves. Cooja1 curve presents a similar profile to the measured curve while Cooja2 curve shows a stronger inclination than the measured curve. Cooja2 computed more energy consumption than was actually consumed. Since measured and Cooja1 do not have a linear profile, there is evidence of the charge recovery phenomena (relaxation effect), and there is also an evidence that energy consumption decreased during application execution. Cooja2 estimated energy consumption was fixed during the application execution, and the charge recovery was not computed. Cooja2 provides a result that indicates that there is not available energy when the sensor node is still working (see Table 6.4). In contrast, Cooja1 indicates that there is available energy (32.93mAh) when the battery stopped to supply energy to the sensor node.

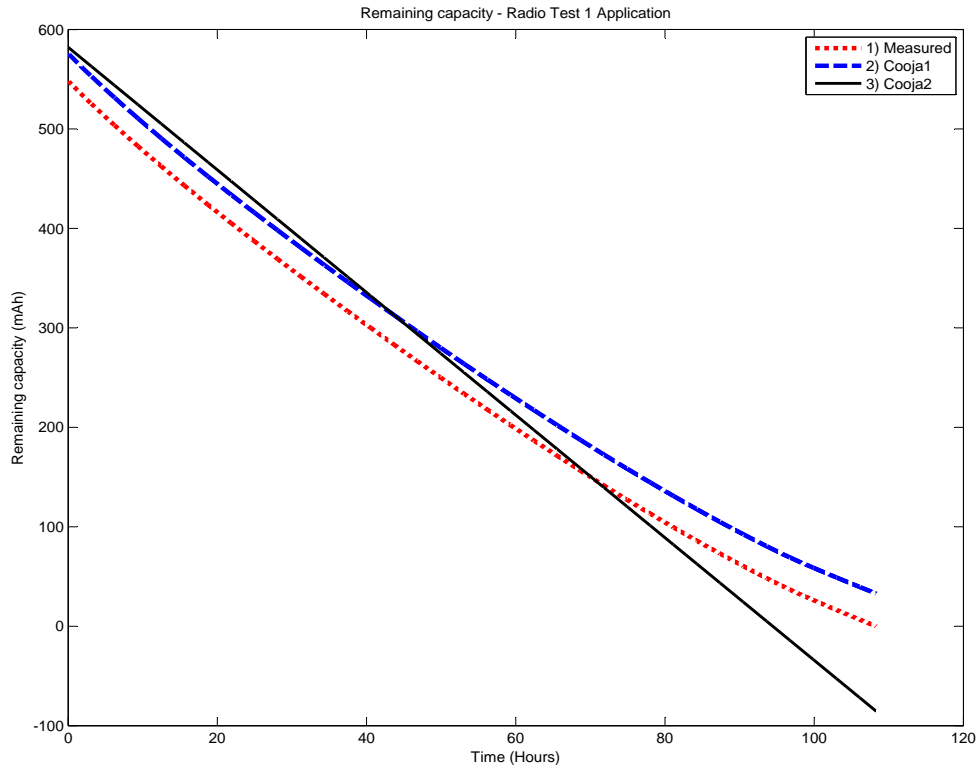


Figure 6.7: Remaining capacity for Radio Test 1 application.

Figure 6.8 shows Radio Test 2 application remaining capacity plots curves. Very similar to Radio Test 1 application, the only difference is the final remaining capacity of 27.32mAh.

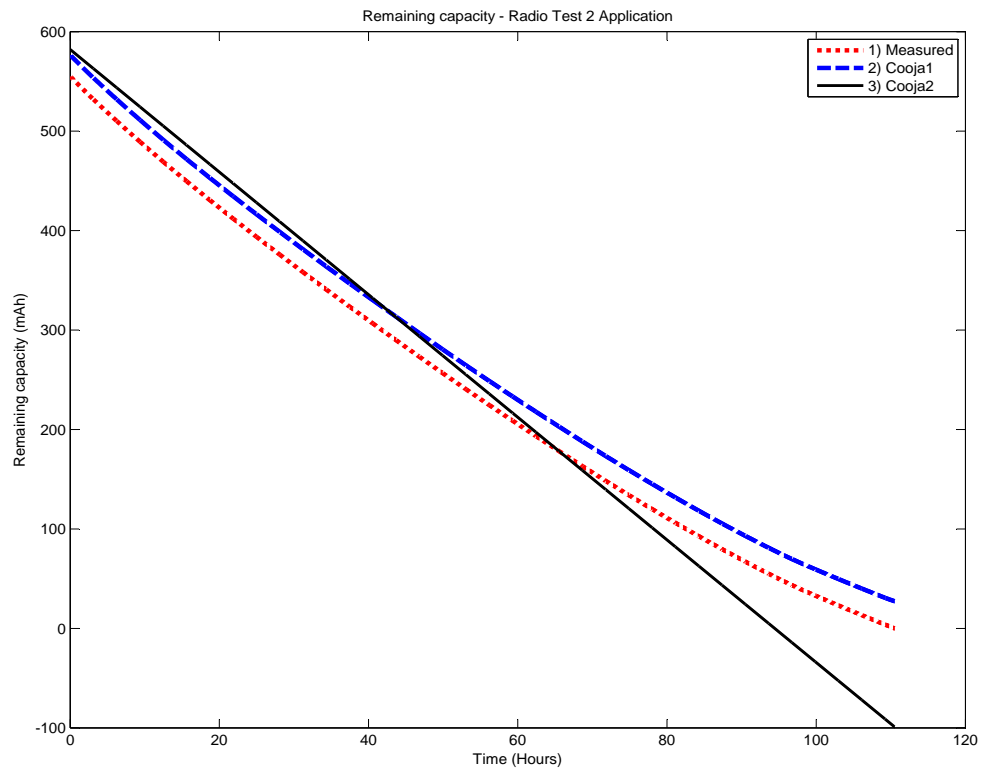


Figure 6.8: Remaining capacity for Radio Test 2 application.

Figure 6.9 shows TestAMOnOffSlave 1 application remaining capacity plots curves. The measured and Cooja1 curves have three points of inclination rate changes, which is maybe due to charge recovery phenomena (relaxation effect). Based on the measured and Cooja1 curves profiles, there is evidence of a charge recovery, and there is also an evidence that energy consumption decreased during application execution. Cooja1 curve has a slightly smaller inclination than the measured curve. Cooja2 provides a result that indicates that there is no available energy when the sensor node is still working (see Table 6.4). In contrast, Cooja1 indicates that there is available energy (18.03mAh) when the battery stopped to supply energy to the sensor node. Again, Cooja2 profile is a straight line, due to the fixed value of voltage supply and current consumption.

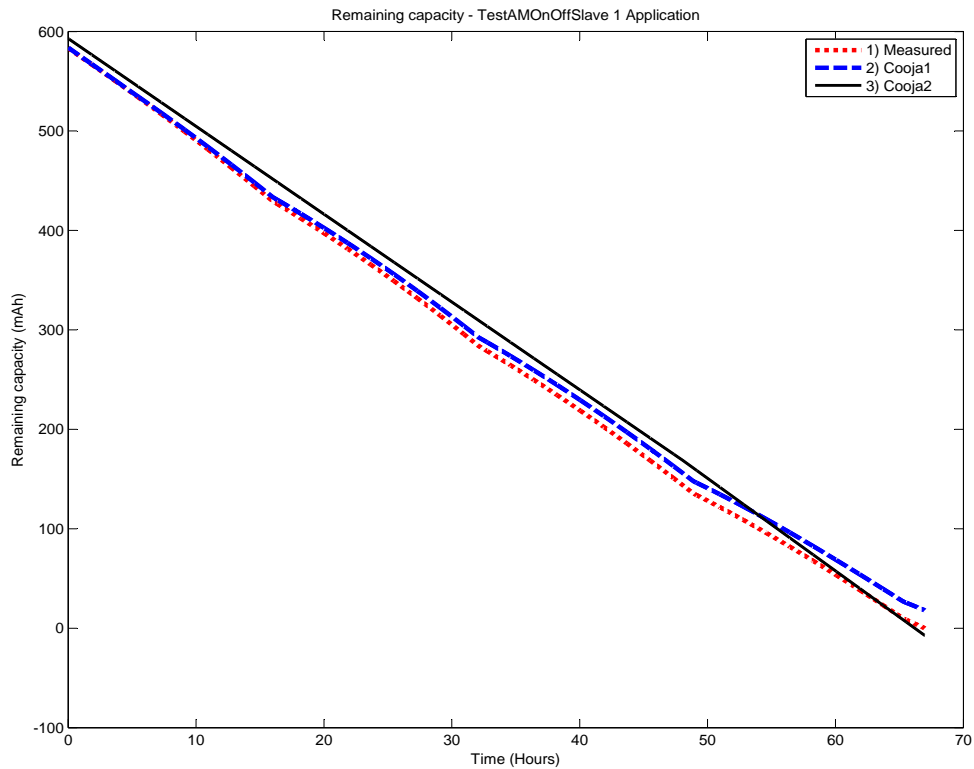


Figure 6.9: Remaining capacity for TestAMOnOffSlave 1 application.

Figure 6.10 presents TestAMOnOffSlave 2 application remaining capacity plots curves. Very similar to TestAMOnOffSlave 1 application, there are two differences: only two inclination changes for the measured and Cooja1 estimatives, and the final remaining capacity for Cooja1 is 14.69mAh.

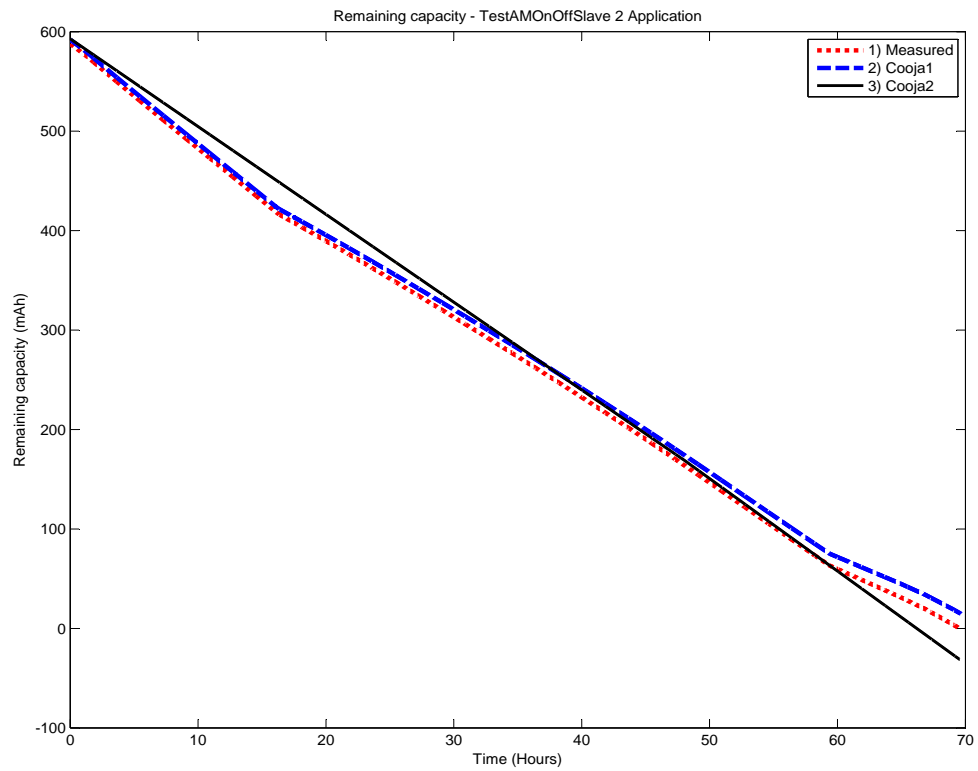


Figure 6.10: Remaining capacity for TestAMOnOffSlave 2 application.

Application	Cooja1(mAh)	Error(%)	Cooja2(mAh)	Erro(%)
Sens 1	8.66	1.46	42.87	7.23
Sens 2	-8.46	1.39	26.51	4.35
Oscilloscope 1	6.19	1.07	20.9	3.62
Oscilloscope 2	6.74	1.16	0.76	0.13
Radio Test 1	32.93	6.00	-85.27	15.54
Radio Test 2	27.32	4.92	-99.14	17.85
RSSI-Scanner 1	56.48	10.36	16.86	3.09
RSSI-Scanner 2	49.02	8.89	9.18	1.67
TestAMOnOffSlave 1	18.03	3.09	-7.29	1.25
TestAMOnOffSlave 2	14.69	2.50	-31.22	5.31

Table 6.4: Comparison between Cooja1 and Cooja2 final remaining capacities.

Table 6.4 shows Cooja1 and Cooja2 final remaining capacity values for each application when the battery stopped to supply energy to the sensor node (i.e. when the battery voltage dropped below 1.8V).

For TinyOS applications, Cooja1 shows errors of less than 3.1% when compared to the measured sensor node data. Cooja1 for Contiki applications shows error of more than 5%, in particular RSSI-Scanner applications that presents an error of more than 8%. Except for Sens 2 application, Cooja1 provides results that indicates that there is available energy when the sensor node stopped working. The average error for Cooja1 is 4.08%.

For Oscilloscope 1, TestAMOnOffSlave 1, RSSI-Scanner 1 and RSSI-Scanner 2 applications, Cooja2 presents errors lower than Cooja1. In contrast, for Sens 1, Sens 2, Radio Test 1, Radio Test 2, TestAMonOffSlave 1, and TestAMonOffSlave, Cooja2 errors is larger than Cooja1 errors, in particular Radio Test 1, Radio Test 2, Sens 1 and TestAMOnOff 1 applications, that present errors of more than 17%, 15%, 7%, and 5%, respectively. For Radio Test and TestAMOnOff applications, Cooja2 provides results that indicate that there is no available energy when the sensor node is still working. In contrast, Sens, RSSI-scanner and Oscilloscope applications indicate that there is available energy when the sensor node stopped working. The average error for Cooja2 is 6.0%.

Application	Cooja1 remaining capacity (mAh)	Absolute Difference (mAh)
Sens 1	8.66	8.73
Sens 2	-8.46	8.39
Oscilloscope 1	6.19	6.26
Oscilloscope 2	6.74	6.19
Radio Test 1	32.93	27.54
Radio Test 2	27.32	21.25
RSSI-Scanner 1	56.48	56.57
RSSI-Scanner 2	49.02	50.2
TestAMOnOffSlave 1	18.03	13.52
TestAMOnOffSlave 2	14.69	13.15

Table 6.5: Cooja1 remaining capacity versus the absolute difference between the measured and Cooja1 total effective capacities

Table 6.5 presents Cooja1 final remaining capacity and the absolute difference between the measured and Cooja1 total effective capacities. As can be seen in Table 6.5 and in the graphs shown above, there is evidence that the major source of error of Cooja1 remaining capacity estimative is the inaccurate battery initial capacity estimative. For example, according to the measured Sens 1 application remaining capacity curve (Figure 6.1), there is no evidence of a charge recovery phenomena due to relaxation effect, since the third sum of the equation 4.6 computed zero for charge recovery and the measured remaining capacity curve presents a linear behavior. Then, the actual battery initial capacity is 592.61mAh, but Cooja1 computed 601.34mAh, the energy consumption started to subtract from 601.34mAh instead of 592.61mAh, resulting in 8.66mAh as the final remaining capacity.

Regarding to Radio Test, beyond the error due to an inaccurate initial battery capacity estimative, the Radio Test applications spent more energy than was computed by Cooja1. For TestAMOnOffSlave applications, Cooja1 computed more charge recovery than actually occurred.

In regard to Cooja2 estimatives, part of their errors is due to the difference between the measured and Cooja2 total effective capacities, but the major source of error of Cooja2 estimative is the current consumption values used by Cooja2 to calculate the consumption and available energy that are based on electric current consumption values extracted from datasheets.

Figure 5.8, Figure 5.10, Figure 5.12, Figure 5.14, and Figure 5.16 presented the relationship between voltage and current for each application (Sens, Oscilloscope, Radio Test, RSSI-Scanner, TestAMOnOff, respectively). These figures showed that

ID	Application	Measured (hh:mm:ss)	Cooja1 (hh:mm:ss)	Difference (hh:mm:ss)	Difference (%)
1	Sens 1	25:37:12	25:29:24	00:22:12	1.44
2	Sens 2	26:22:12	26:00:00	-1:37:48	-1.40
3	Oscilloscope 1	25:10:48	25:27:36	00:16:48	1.11
4	Oscilloscope 2	26:03:36	27:01:12	00:57:36	3.68
5	Radio Test 1	108:17:24	118:28:48	10:11:24	9.41
6	Radio Test 2	110:32:24	119:29:36	09:07:12	8.25
7	RSSI-Scanner 1	28:53:24	31:53:24	03:00:00	10.38
8	RSSI-Scanner 2	29:16:12	31:31:12	02:15:00	7.69
9	TestAMOnOffSlave 1	66:59:24	69:48:36	02:49:12	4.21
10	TestAMOnOffSlave 2	69:33:26	71:46:48	02:13:11	3.19

Table 6.6: Measured sensor node lifetime versus Cooja1 sensor node lifetime with a sensor node cutoff voltage equal to 1.8V

during sensor node lifetime the battery voltage decrease from 3.1V to 1.8V, and current consumption values present a large variability. However, Cooja2 current consumption values are fixed during the simulation, since they are extracted from datasheets and the battery voltage values was also fixed. For each radio, CPU and LED operation modes a given constant current consumption value is used.

For some applications, such as Radio Test and TestAMOnOffSlave, there were sharp drops in the current curves close to 2.1V, that maybe indicates that some sensor node components, such as LEDs stopped working. Besides, during the experiments, in these applications was observed that some LEDs stopped to blink when the battery voltage dropped around 2.1V, but during Cooja2 simulation, the LEDs did not stop working and the energy spent by them were computed.

6.3 Sensor Node Lifetime

This section presents the results and describes the model validation of CoojaED extension that determines whether the simulated sensor node lifetime (Cooja1 and Cooja2) is an acceptable representation of the measured sensor node lifetime.

Table 6.7 shows the measured and Cooja1 sensor node lifetime for each application and their respective differences to the measured sensor node lifetimes.

Table 6.6 presents the measured and Cooja2 sensor node lifetime for each application and their respective differences to the measured sensor node lifetimes. The Cooja2 sensor node lifetime result is deterministic.

ID	Application	Measured (hh:mm:ss)	Cooja2 (hh:mm:ss)	Difference (hh:mm:ss)	Difference (%)
1	Sens 1	25:37:12	26:24:48	00:57:36	3.75
2	Sens 2	26:22:12	26:24:48	00:12:36	0.80
3	Oscilloscope 1	25:10:48	26:05:24	00:54:36	3.61
4	Oscilloscope 2	26:03:36	26:05:24	00:01:48	0.12
5	Radio Test 1	108:17:24	96:27:00	-12:09:36	-10.93
6	Radio Test 2	110:32:24	96:27:00	-15:54:36	-12.75
7	RSSI-Scanner 1	28:53:24	29:43:12	00:49:48	2.87
8	RSSI-Scanner 2	29:16:12	29:43:12	00:26:00	1.54
9	TestAMOnOffSlave 1	66:59:24	66:30:00	-1:30:36	-0.73
10	TestAMOnOffSlave 2	69:33:26	66:30:00	-4:56:24	-4.40

Table 6.7: Measured sensor node lifetime versus Cooja2 sensor node lifetime with a sensor node cutoff voltage equal to 1.8V

Although the consumption and available estimatives using values extracted from datasheets (Cooja2) present errors larger than based on experimental values, this estimation approach has some advantages. First one, designers do not need to develop an experimental setup to sample and save in a log file the current consumption and battery voltage of a measured sensor node application in order to evaluate the sensor node energy consumption and sensor node lifetime. Second one, depending on the sample rate and type of application, the log file size can be very large, for example, the log file size of each Radio Test application was approximately 20GB. Finally, the same current consumption values can be used for several applications. For this reasons, the results and a validation of the Cooja2 sensor node lifetime are presented below.

Except for Sens 2 application, Cooja1 provides results that indicates that there is no available energy when the sensor node is still working. For TinyOS applications, Cooja1 show errors of less than 4.5% when compared to measured sensor node lifetime. Cooja1 for Contiki applications shows errors of more than 7%, in particular, RSSI-Scanner 1 and Radio Test 1 applications that present an error of more than 9%. The average error for Cooja1 is 5.07%.

For Radio Test and TestAMOnOff applications, Cooja2 provides results that indicate that there is no available energy when the sensor node is still working. In contrast, for Sens, RSSI-scanner and Oscilloscope applications, indicates that there is remaining capacity when the sensor node stopped working.

Radio Test applications show the largest percentage difference between measured and estimated sensor node lifetime, and present errors of more 10%. During the

experiments, it was observed that some LEDs stopped to blink when the battery voltage dropped to around 2.1V, but Cooja2 simulation continued to compute their energy consumption. Thus, Cooja2 computed more energy consumption than it was actually consumed. For this reason, Radio Test applications present this large difference between the measured and estimated sensor node lifetime. In contrast, Sens, Oscilloscope, RSSI-Scanner and TestAMOnOffSlave applications show errors of less 5% than when compared to measured sensor node data. These applications present errors of less 3.8%, 3.7%, 2.9%, and 4.5%, respectively. The average error for Cooja2 is 4.29%.

Figure 6.11 presents a comparison between the measured and estimated sensor node lifetime (Cooja1 and Cooja2) for each application. The applications are numbered according their ID column in Table 6.7.

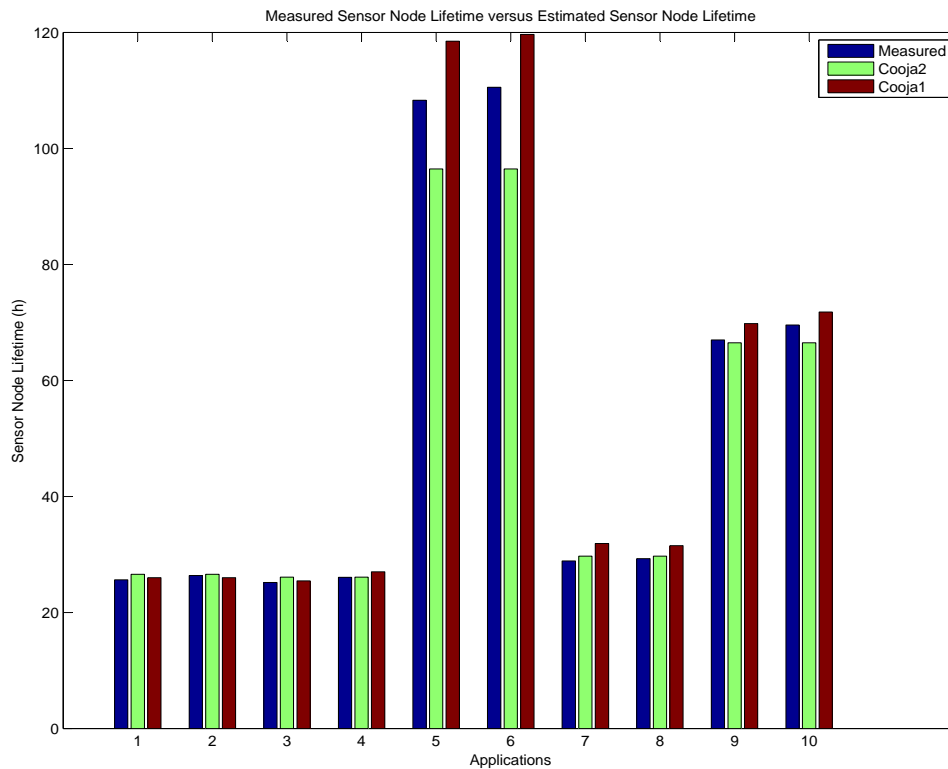


Figure 6.11: Real and estimated sensor node lifetimes.

Figure 6.13 and Figure 6.12 present scatter plots with measured sensor node lifetime (x) and estimated sensor node lifetime (y). A scatter plot is a diagram displaying observations of two variables, x and y. Each observation is represented by a point showing its x-y coordinates. The scatter plot is very effective in revealing the joint variability of x and y or the nature of the relationship between them [Montgomery and Runger, 2006]. An application of linear regression analysis describes the relationship between the measured and simulated sensor node lifetime (Cooja1 and Cooja2) is investigated. If the regression line passes exactly through every point on the scatter plot, it would be able to explain all of the variations.

A correlation analysis establishes if a linear relationship exists between the measured and estimated sensor node lifetime. The correlation value quantifies the strength of a linear relationship between two variables. When there is no correlation there is no tendency for the values of the variables to increase or decrease in tandem [Center, 2012][Montgomery and Runger, 2006].

Figure 6.12 shows a scatter plot of the measured sensor node lifetime versus Cooja1 sensor node lifetime presented in Table 6.6. We used MATLAB and its Basic Fitting GUI to display the graph, and the fitting data, respectively. There is an evidence of a linear relationship between measured and Cooja1 sensor node lifetimes since the pair of points in the graph (measured-estimated coordinates) have a slightly linear behavior.

Correlation coefficient is a dimensionless measure of the interdependence between two variables, usually lying in the interval from -1 to $+1$, where a value close to 1 indicates that there is a positive linear relationship between x and y; a value close to -1 indicates a negative linear; and a value close to or equal to 0 (zero) suggests there is no linear relationship.

The most common form of the correlation coefficient used in practice is a measure of the linear association between the two variables y and x [Montgomery and Runger, 2006]. The correlation coefficient when applied to a population is represented by the letter ρ . The correlation coefficient when applied to a sample is represented by the letter r , and is obtained by the following equation

$$r = \frac{\sigma_{xy}}{\sqrt{\sigma_x \sigma_y}}$$

where σ_{xy} is the covariance between measured (x) and Cooja1 (y) sensor node lifetime outputs.

The sample correlation coefficient (r) is 0.9992. The coefficient of determination

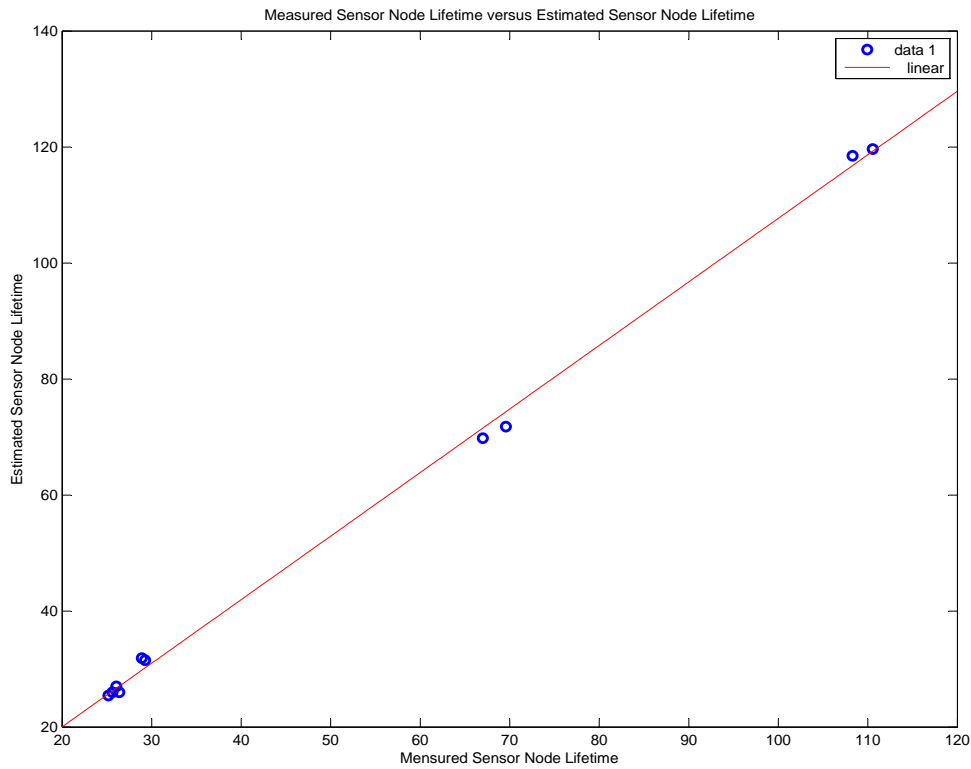


Figure 6.12: Validation of the estimated sensor node lifetime(Cooja1) with a sensor node cutoff voltage equal to 1.8V.

R^2 (the square of sample correlation coefficient) is often used to judge the adequacy of a regression model. In this case R^2 is equal to 0.9985, that indicates that approximately 99.56% of the variability in Cooja1 sensor node lifetime is explained by the linear relationship to the actual sensor node lifetime.

Sometimes, there is a correlation between the measured and simulation model sample, however, it does not guarantee a correlation between the measured and Cooja1 populations since $r \neq 0$ is no guarantee that $\rho \neq 0$. In order to solve this problem, the test hypotheses with a confidence interval α must be applied.

$$\begin{aligned}
 H_0 : \rho &= 0 \\
 H_1 : \rho &\neq 0
 \end{aligned}
 \tag{6.1}$$

where $\rho = 0$ means that there is no correlation between measured and simulation populations. The appropriate test statistic for these hypotheses is

$$t_0 = \frac{R\sqrt{n-2}}{\sqrt{n-r^2}} \quad (6.2)$$

which has the t distribution with $(n - 2)$ degrees of freedom (i.e., the number of independent comparisons that can be made among the elements of a sample) if $H_0 : \rho = 0$ is true. The null hypotheses is rejected if $|t_0| > t_{\alpha/2, n-2}$, where n is the sample size (i.e., the number of observations in a sample).

We applied a hypotheses test with $\alpha = 0.5$. The result of equation 6.2 is $t_0 = 76.7403$. Since $t_{0.025, 8} = 2.306$, H_0 is rejected, and implies that the correlation coefficient $\rho \neq 0$ which indicates that there is a strong correlation between measured and Cooja1 sensor node lifetime.

It is possible to construct an appropriate $100(1 - \alpha)\%$ confidence interval for ρ , using the transformation in Equation 6.3.

$$\tanh(\operatorname{arctanh}(r) - \frac{Z_{\alpha/2}}{\sqrt{n-3}}) \leq \rho \leq \tanh(\operatorname{arctanh}(r) + \frac{Z_{\alpha/2}}{\sqrt{n-3}}) \quad (6.3)$$

where $Z_{\alpha/2}$ is the critical value (i.e. the value of a statistic corresponding to a stated significance level as determined from the sampling distribution) of the 0.05 level of significance. Therefore, an approximate 95% confidence interval on ρ is constructed which reduces to $0.9966 \leq \rho \leq 0.9998$.

There is a strong positive relationship between the measured and estimated sensor node lifetimes. If two variables are correlated, it is possible to predict one based on the other. Then it is possible to predict the measured sensor node lifetime based on Cooja1 sensor node lifetime. However, a correlation cannot indicate anything about whether one variable caused the other.

Figure 6.13 shows the scatter plot of the measured sensor node lifetime versus Cooja2 sensor node lifetime (presented in Table 6.7). We also used MATLAB and its Basic Fitting GUI to display the graph, and the fitting data, respectively. There is also an evidence of a linear relationship between measured and Cooja2 sensor node lifetime since the points in graph (measured-estimated coordinates) have a slightly linear behavior.

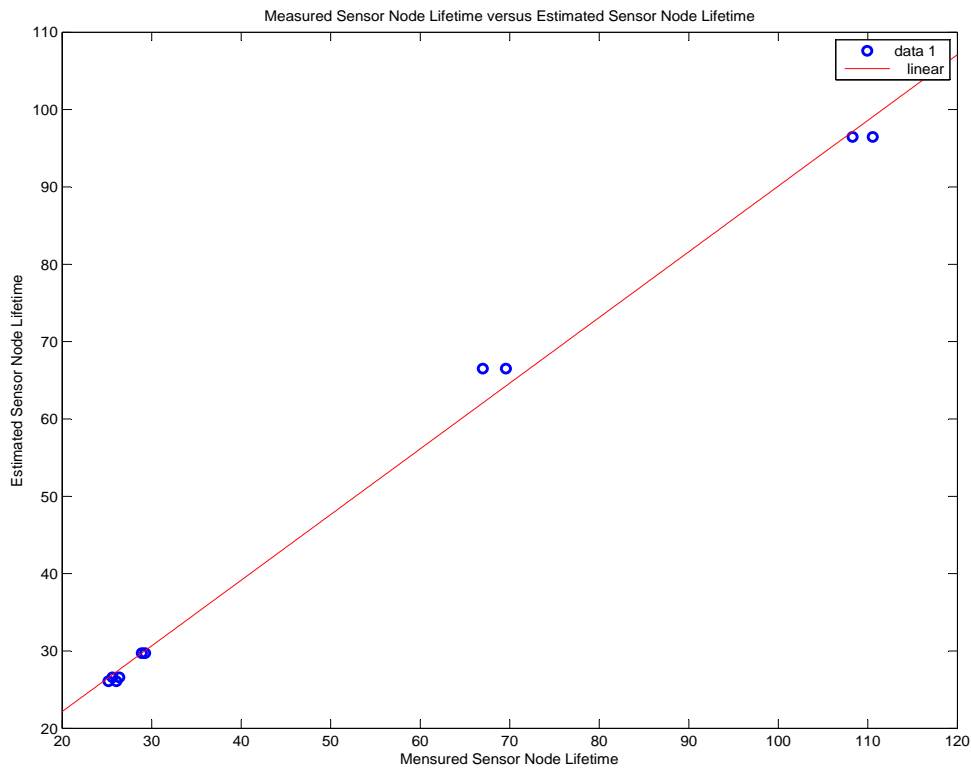


Figure 6.13: Validation of the estimated sensor node lifetime (Cooja2) with a sensor node cutoff voltage equal to 1.8V

The calculated sample correlation coefficient r is 0.9978 and the coefficient of determination R^2 is 0.9956, which indicates that approximately 99.56% of the variability in Cooja2 sensor node lifetime is explained by the linear relationship to measured sensor node lifetime.

Applying a hypotheses test with $\alpha = 0.5$ (Equation 6.2) to obtain $t_0 = 45.096$. Since $t_{0.025,8} = 2.306$, H_0 is rejected, and implies that the correlation coefficient $p \neq 0$

(Equation 6.1) which indicates that there is a strong correlation between measured and Cooja2 sensor node lifetime.

We also constructed an approximate 95% confidence interval on ρ provided by Equation 6.3, which reduces to $0.9903 \leq \rho \leq 0.9995$. There is also a strong positive relationship between the measured and Cooja2 sensor node lifetime. It is possible to predict the measured sensor node lifetime based on Cooja2 sensor node lifetime.

Chapter 7

Conclusions and Future Works

In this work, a new taxonomy for WSN simulators with four domains to capture and describe a real WSN was proposed. The proposed taxonomy aims at helping WSN designers to find the most appropriate WSN simulator for their WSN projects providing a detailed vision on different aspects of a complete WSN simulation. Designers can characterize and compare different simulators in order to properly choose the most adequate for a given application.

The proposed taxonomy has the following advantages: the first one: proposes and addresses four domains that characterize a real WSN: sensor node (hardware and software), network, energy and the environment. Secondly, provides the actual level of detail for each WSN simulator in each domain. Thirdly, provides a detailed vision about different aspects of the simulation of a WSN. Fourth one, enables WSN designers to easily find out differences and similarities among WSN simulators. Finally, includes the new generation of WSN simulators based on the SystemC language, that can describe the sensor node and the environment with a broader scope and details closer to the real implementation. In order to show that the proposed taxonomy is reasonable and comprehensive, several WSN simulators referred in the literature were classified and an elaborate analysis of these simulators was also presented.

Using the proposed taxonomy the CoojaED simulator was selected to implement a new energy consumption estimator and a realistic and non-linear battery model for alkaline batteries that incorporates the relaxation effect, discharge rate and capacity retention. This work extends COOJA simulator providing a new version that estimates the power consumption and the available energy of sensor nodes. CoojaED extension provides the energy information at simulation time, in other words, it calculates and prints the energy consumption and the available at every simulation step and designers can save a log file with all this information.

To determine whether the simulation model is an acceptable representation of the real system, an experimental setup was built to validate CoojaED extension, and a comparison between the real system and the simulation model, using graphical displays and statistical tests, was made.

Regarding to Cooja-ED extension (Cooja1) estimative, there is evidence that major source of error for the remaining capacity estimative is the inaccurate battery initial capacity estimative. The total effective capacity results showed a great difference between real and estimated estimatives, in particular for RSSI-Scanner application.

Radio Test applications spent more energy than was computed by Cooja1, and more charge recovery than actually recovered in TestAMOnOffSlave applications. For all applications, the Cooja1 remaining capacity curve showed a similar profile to the real curve, and the average error of the final remaining capacity estimative is 4.08%. For sensor node lifetime estimatives, Cooja1 showed errors of less than 4.5% for TinyOS applications. In contrast, for Contiki applications, Cooja1 presented errors of more than 7%.

In regard to Cooja2 estimatives, part of their errors are due to the difference between the real and Cooja2 total effective capacities, but the major source of error of Cooja2 estimative is the electric current consumption values used to calculate the consumption and available energy that are based on current consumption values extracted from datasheets and the battery voltage values are also fixed, and remaining constant during the simulations.

For some applications such as, Oscilloscope 1, TestAMOnOffSlave 1, RSSI-Scanner 1 and RSSI-Scanner 2, Cooja2 final remaining capacity result present errors lower than Cooja1. Cooja2 remaining capacity curves did not show a very similar profile to real curve, and the average error for Cooja2 remaining capacity estimatives is 6.0%. Regarding to sensor node lifetime estimatives, Radio Test applications showed the largest error. Sens, Oscilloscope, RSSI-Scanner and TestAMOnOffSlave applications showed errors of less 5% than when compared to real sensor node data.

The scatter plot showed an evidence of a linear relationship between real and simulated sensor node lifetimes (Cooja1 and Cooja2). There is a strong positive relationship, which indicates that is possible to predict the real sensor node lifetime based on estimated sensor node lifetime. The results indicate that the simulation model is an acceptable representation of the real system.

7.1 Future Works

This work is the starting point to the following further research. First one, the implementation of the BMAB battery model in the Contiki operating system, and also extends the Contiki energy profile system. As explained in Chapter 3, Contiki has implemented an energy profiler named Energest, and provides an interface, named Powertrace, for the energy profiling system. However, it only prints how much time in clock ticks was spent for radio communication or common tasks using the CPU, in other words, the duty cycle of sensor node components in different activity states. The proposed research has the objective to extend the Contiki energy profiler, providing a new version that estimates the consumption and available energy of a real sensor node platform during the application execution. This is a work in progress research.

Second one, at the moment, CoojaED Extension only estimates the consumption and available energy for a network composed by sensor nodes based on the MSP430 family and the CC2420 radio, and does not provide the energy information for the sensors and flash memory. This proposed research extends CoojaED, providing a new version that estimates the consumption and available energy of networks based on others cpu, radio, and sensor node platforms (e.g. AVR-based motes), and can also provide the energy information for the sensors and flash memory.

Finally, up to now, CoojaED extension has implemented only a non-rechargeable battery model (the BMAB battery model). Another further research is the inclusion of rechargeable battery model and energy harvesting.

Bibliography

- ATLeS-SN (2012). *Dynamic Profiling and Optimization (DPOP) for Sensor Networks*. Available from: <http://www2.engr.arizona.edu/~dpop/Main/ATLeSSN>.
- Aynsley, J. (2009). *OSCI TLM-2.0 LANGUAGE REFERENCE MANUAL*. TLM 2.0.1.
- Boulis, A. (2011). *Castalia, A simulator for Wireless Sensor Networks and Body Area Networks, User's manual, version 3.2*. "<http://castalia.npc.nicta.com.au/pdfs/Castalia-UserManual.pdf>".
- Castillo, S., Samala, N. K., Manwaring, K., Izadi, B. A., and Radhakrishnan, D. (2004). Experimental analysis of batteries under continuous and intermittent operations. In Arabnia, H. R., Guo, M., and Yang, L. T., editors, *Proceedings of the International Conference on Embedded Systems and Applications, ESA 04 & Proceedings of the International Conference on VLSI, VLSI 04, June 21-24, 2004, Las Vegas, Nevada, USA*, pages 18–24. CSREA Press.
- CC1000-datasheet (2012). *Single Chip Very Low Power RF Transceiver*. Available from: <http://www.ti.com/lit/ds/symlink/cc1000.pdf>.
- CC2420-datasheet (2012). *2.4 GHz IEEE 802.15.4 ZigBee-ready RF Transceiver*. Available from: <http://inst.eecs.berkeley.edu/~cs150/Documents/CC2420.pdf>.
- Center, M. D. (2012). *Linear Correlation*. Available from: http://www.mathworks.com/help/matlab/data_analysis/linear-correlation.html.
- Contiki-WebSite (2012). *Contiki-The Open Source OS for the Internet of Things*. Available from: <http://www.contiki-os.org/>.
- da Cunha, A. and da Silva, D. C. (2012). Behavioral model of alkaline batteries for wireless sensor networks. *Latin America Transactions, IEEE (Revista IEEE America Latina)*, 10(1):1295–1304.

- da Cunha, A. B. (2010). *In Portuguese: Uma Abordagem para a Modelagem da Consciência de Disponibilidade Energética em Nós Sensores de Rede de Sensores Sem Fio*. PhD thesis, Universidade Federal de Minas Gerais.
- Damm, M., Moreno, J., Haase, J., and Grimm, C. (2010). Using transaction level modeling techniques for wireless sensor network simulation. In *The Design, Automation, and Test in Europe (DATE), 2010*, pages 1047–1052.
- de Paz Alberola, R. and Pesch, D. (2008). Avroraz: extending Avroza with an IEEE 802.15.4 compliant radio chip model. In *Proceedings of the 3rd ACM workshop on Performance monitoring and measurement of heterogeneous wireless and wired networks*, Performance Monitoring, Measurement and Evaluation of Heterogeneous Wireless and Wired Networks Workshop (PM2HW2N), 2008, pages 43–50, New York, NY, USA. ACM.
- Du, W., Mieleville, F., and Navarro, D. (2010a). Idea1: A SystemC-based system-level simulator for wireless sensor networks. In *IEEE International Conference on Wireless Communications, Networking and Information Security (WCNIS), 2010*, pages 618–622.
- Du, W., Mieleville, F., and Navarro, D. (2010b). Modeling energy consumption of wireless sensor networks by SystemC. In *International Conference on Systems and Networks Communications (ICSNC), 2010*, pages 94–98.
- Du, W., Navarro, D., Mieleville, F., and Gaffiot, F. (2010c). Towards a taxonomy of simulation tools for wireless sensor networks. In *SIMUTools, 2010*, pages 52:1--52:7, ICST, Brussels, Belgium, Belgium. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).
- Du, W., Navarro, D., Mieleville, F., and O'Connor, I. (2011). Idea1: A validated system c-based simulator for wireless sensor networks. In *International Conference on Mobile Ad hoc and Sensor Systems (MASS), 2011*, pages 825–830.
- Dunkels, A., Gronvall, B., and Voigt, T. (2004). Contiki - a lightweight and flexible operating system for tiny networked sensors. In *Local Computer Networks (LCN), 2004*, pages 455–462, Washington, DC, USA. IEEE Computer Society.
- Dunkels, A., Osterlind, F., Tsiftes, N., and He, Z. (2007). Software-based on-line energy estimation for sensor nodes. In *International conference on Economics and Management of networks (EmNets), 2007*, pages 28–32, New York, NY, USA. ACM.

- Eriksson, J., Dunkels, A., Finne, F., Österlind, F., and Voigt, T. (2007). MSPSim - an extensible simulator for MSP430-equipped sensor boards. In *European Conference on Wireless Sensor Networks (EWSN), 2007*.
- Eriksson, J., Österlind, F., Finne, N., Tsiftes, N., Dunkels, A., Voigt, T., Sauter, R., and Marrón, P. J. (2009). Cooja/MSPSim: interoperability testing for wireless sensor networks. In *SIMUTools, 2009*, pages 27:1–27:7, ICST, Brussels, Belgium, Belgium. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).
- Fall, K. and Varadhan, K. (2011). *The ns-2 Manual, formerly ns-2 Notes and Documentation*. UC Berkeley, LBL, USC/ISI, and Xerox PARC.
- Farooq, M. O. and Kunz, T. (2011). Operating systems for wireless sensor networks: A survey. *Sensors*, 11(6):5900–5930.
- Fonseca, J. (2009). *Towards a Test Framework for Networked Embedded Systems*. PhD thesis, University of Copenhagen.
- Fummi, F., Quaglia, D., and Stefanni, F. (2008). A SystemC-based framework for modeling and simulation of networked embedded systems. In *FDL, 2008*, pages 49–54.
- Gay, D., Levis, P., von Behren, R., Welsh, M., Brewer, E., and Culler, D. (2003). The nesC language: A holistic approach to networked embedded systems. *SIGPLAN Not.*, 38(5):1–11.
- Glaser, J., Weber, D., Madani, S. A., and Mahlknecht, S. (2008). Power aware simulation framework for wireless sensor networks and nodes. *EURASIP Journal Embedded System*, 2008:3:1–3:16.
- Grotker, T. (2002). *System Design with SystemC*. Kluwer Academic Publishers, Norwell, MA, USA.
- Haase, J., Molina, J., and Dietrich, D. (2011). Power-aware system design of wireless sensor networks: Power estimation and power profiling strategies. *Industrial Informatics, IEEE Transactions on*, 7(4):601–613.
- Henderson, T. R., Roy, S., Floyd, S., and Riley, G. F. (2006). ns-3 project goals. In *Proceeding from the 2006 workshop on ns-2: the IP network simulator*, WNS2 '06, New York, NY, USA. ACM.

- Hiner, J., Shenoy, A., Lysecky, R., Lysecky, S., and Ross, A. G. (2010). Transaction-level modeling for sensor networks using SystemC. In *IEEE International Conference on Sensor Networks, Ubiquitous, and Trustworthy Computing (SUTC), 2010*, pages 197–204.
- Hossain, A., Biswas, P., and Chakrabarti, S. (2008). Sensing models and its impact on network coverage in wireless sensor network. In *International Conference on Information and Intelligent Systems (ICIIS), 2008*, pages 1–5.
- IEEE (2006). *Part 15.4: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (WPANs)*. IEEE Std 802.15.4.
- Instruments, N. (2012). *NI USB-621x User Manual*. Available from: <http://www.ni.com/pdf/manuals/371931f.pdf>.
- IRIS (2012). *IRIS-Datasheet*. Available from: http://bullseye.xbow.com:81/Products/Product_pdf_files/Wireless_pdf/IRIS_Datasheet.pdf.
- Jamieson, O. G. P. L. R. F. K. and Moss, D. (2012). *CTP: Collection Tree Protocol*. Available from: <http://sing.stanford.edu/gnawali/ctp/>.
- Jevtić, M., Zogović, N., and Dimić, G. (2009). Evaluation of wireless sensor network simulators. *Telecommunications Forum (TELFOR)*, pages 1303–1306.
- Kleijnen, J. (1999). Validation of models: statistical techniques and data availability. In *Simulation Conference Proceedings, 1999 Winter*, volume 1, pages 647–654 vol.1.
- Lajara, R., Pelegrí-Sebastiá, J., and Solano, J. J. P. (2010). Power consumption analysis of operating systems for wireless sensor networks. *Sensors*, 10(6):58096–5826.
- Levis, P., Lee, N., Welsh, M., and Culler, D. (2003). Tossim: accurate and scalable simulation of entire tinyos applications. In *SenSys, 2003*, pages 126–137, New York, NY, USA. ACM.
- Levis, P., Madden, S., Polastre, J., Szewczyk, R., Woo, A., Gay, D., Hill, J., Welsh, M., Brewer, E., and Culler, D. (2004). Tinyos: An operating system for sensor networks.
- LM35 (2012). *Precision Centigrade Temperature Sensors*. Available from: <http://www.ti.com/lit/ds/symlink/lm35.pdf>.
- LPL (2012). *TinyOS Documentation Wiki - Low-Power-Listening*. Available from: <http://www.tinyos.net/tinyos-2.x/doc/html/tep105.html>.

- MICA2 (2012). *MICA2 Datasheet*. Available from: http://bullseye.xbow.com:81/Products/Product_pdf_files/Wireless_pdf/MICA2_Datasheet.pdf.
- MICAz (2012). *MICAz Datasheet*. Available from: http://bullseye.xbow.com:81/Products/Product_pdf_files/Wireless_pdf/MICAz_Datasheet.pdf.
- Microship-datasheet (2012). *MRF24J40 IEEE 802.15.4 2.4 GHz RF Transceiver*. Available from: <http://ww1.microchip.com/downloads/en/devicedoc/39776c.pdf>.
- MN1604, D. (2012). *Discharge characteristics of MN1604 duracell battery*. Available from: <http://datasheet.octopart.com/MN-1604-Duracell-datasheet-90236.pdf>.
- Montgomery, D. C. and Runger, G. C. (2006). *Applied Statistics and Probability for Engineers, 4th Edition, and JustAsk! Set*. John Wiley & Sons, 4 edition.
- Moreno, J., Wenninger, J., Haase, J., and Grimm, C. (2011). Energy profiling technique for network-level energy optimization. In *AFRICON, 2011*, pages 1–6.
- Ningning, Q., Fan, X., Jing, Y., and Guisheng, L. (2010). Research on the sensing model in wireless sensor networks. In *Intelligent Computation Technology and Automation (ICICTA), 2010*, volume 3, pages 169–172.
- Österlind, F. (2006). *A Sensor Network Simulator for the Contiki OS*. Sics technical report, Swedish Institute of Computer Science (SICS).
- Österlind, F., Dunkels, A., Eriksson, J., Finne, N., and Voigt, T. (2006). Cross-level sensor network simulation with cooja. In *Local Computer Networks (LCN), 2006*, pages 641–648.
- OTcl (2012). *OTcl home page*. Available from: <http://otcl-tclcl.sourceforge.net/otcl/>.
- Park, C., Lahiri, K., and Raghunathan, A. (2005). Battery discharge characteristics of wireless sensor nodes: an experimental analysis. In *Sensor and Ad Hoc Communications and Networks, 2005. IEEE SECON 2005. 2005 Second Annual IEEE Communications Society Conference on*, pages 430 – 440.
- Perla, E., Catháin, A. O., Carbajo, R. S., Huggard, M., and Mc Goldrick, C. (2008). Powertossim z: realistic energy modelling for wireless sensor network environments. In *Performance Monitoring, Measurement and Evaluation of Heterogeneous Wireless*

- and Wired Networks Workshop (PM2HW2N) 2008*, pages 35–42, New York, NY, USA. ACM.
- Polastre, J., Szewczyk, R., and Culler, D. (2005). Telos: enabling ultra-low power wireless research. In *ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN), 2005*, Piscataway, NJ, USA. IEEE Press.
- Pop, V., Bergveld, H., Notten, P., and Regtien, P. (2005). State-of-the-art of battery state-of-charge determination. *Measurement Science and Technology*, 16(12):R93–R110.
- Prabhakar, T., Venkatesh, S., Sujay, M., Kuri, J., and Praveen, K. (2008). Simulation blocks for tossim-t2. In *International Conference on COMmunication System softWARe and middlewaRE (COMSWARE), 2008*, pages 17–23.
- Sargent, R. G. (1998). Verification and validation of simulation models. pages 55–64.
- Shnayder, V., Hempstead, M., Chen, B.-r., Allen, G. W., and Welsh, M. (2004). Simulating the power consumption of large-scale sensor network applications. In *ACM Conference on Embedded Networked Sensor Systems (SenSys), 2004*, pages 188–200, New York, NY, USA. ACM.
- Siewiorek, D. P., Bell, G., and Newell, A. C. (1982). *Computer Structures: Principles and Examples*. McGraw-Hill, Inc., New York, NY, USA.
- Sky, T. (2012). *Ultra Low Power IEEE 802.15.5 Compliant Wireless Sensor Module*. Available from: <http://www.eecs.harvard.edu/~konrad/projects/shimmer/references/tmote-sky-datasheet.pdf>.
- Sobeih, A., Hou, J. C., Kung, L.-C., Li, N., Zhang, H., Chen, W.-P., Tyan, H.-Y., and Lim, H. (2006). J-sim: a simulation and emulation environment for wireless sensor networks. *Wireless Commun.*, 13(4):104–119.
- Stargate (2012). *Stargate*. Available from: <http://platformx.sourceforge.net/>.
- Stecklina, O., Vater, F., Basmer, T., Bergmann, E., and Menzel, H. (2011). Hybrid simulation environment for rapid MSP430 system design test and validation using MSPsim and SystemC. In *IEEE Symposium On Design And Diagnostics Of Electronic Circuits And Systems (DDECS), 2011*, pages 167–170.
- Stehlik, M. (2011). *Comparison of Simulators for Wireless Sensor Networks*. PhD thesis, Masaryk University.

- TinyOS (2012). *TinyOS Documentation Wiki*. Available from: http://docs.tinyos.net/tinywiki/index.php/Main_Page.
- TinyOSWiki-Avrora (2012). *TinyOS Documentation Wiki*. Available from: <http://docs.tinyos.net/tinywiki/index.php/Avrora>.
- Titzer, B. L., Lee, D. K., and Palsberg, J. (2005). Avrora: scalable sensor network simulation with precise timing. In *ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN), 2005*, Piscataway, NJ, USA. IEEE Press.
- TOSSIM (2012). *TinyOS Documentation Wiki*. Available from: <http://docs.tinyos.net/tinywiki/index.php/TOSSIM>.
- TR1100-datasheet (2012). *916.50 MHz Hybrid Transceiver*. Available from: <http://www.rfm.com/products/data/tr1100.pdf>.
- van Dam, T. and Langendoen, K. (2003). An adaptive energy-efficient mac protocol for wireless sensor networks. In *ACM Conference on Embedded Networked Sensor Systems (SenSys), 2003*, pages 171–180, New York, NY, USA. ACM.
- Varga, A. and Hornig, R. (2008). An overview of the omnet++ simulation environment. In *SIMUTools, 2008*, pages 60:1–60:10, ICST, Brussels, Belgium, Belgium. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).
- Vieira, M., Coelho, Claudionor.N., J., da Silva, Diógenes.Cecílio., J., and da Mata, J. (2003). Survey on wireless sensor network devices. In *IEEE Conference on Emerging Technologies and Factory Automation (ETFA), 2003*, volume 1, pages 537 – 544 vol.1.
- Warneke, B. and Pister, K. (2002). Mems for distributed wireless sensor networks. In *ICECS, 2002*, volume 1, pages 291 – 294 vol.1.
- Wismote (2012). *WiSMote Documents*. Available from: <http://www.aragossystems.com/en/document-center/wismote-docs-center-detail.html>.
- Ye, W., Heidemann, J., and Estrin, D. (2002). An energy-efficient mac protocol for wireless sensor networks. In *IEEE International Conference on Computer Communications (INFCOM), 2002*, volume 3, pages 1567 – 1576 vol.3.

Yick, J., Mukherjee, B., and Ghosal, D. (2008). Wireless sensor network survey. *Comput. Netw.*, 52:2292–2330.

Z1 (2012). *Z1 Platform*. Available from: <http://www.zolertia.com/ti>.

Zheng, J. Jamelipour, A. (2009). *Wireless Sensor Networks: A Networking Perspective*. Wiley-IEEE Press.

Appendix A

Temperature versus Capacity Loss Look up Table

Temperature	Capacity Loss	Temperature	Capacity Loss	Temperature	Capacity Loss
20,0	0.035	33.4	0.072	46.8	0.193
20.1	0.035	33.5	0.073	46.9	0.194
20.2	0.035	33.6	0.073	47.0	0.196
20.3	0.036	33.7	0.074	47.1	0.198
20.4	0.036	33.8	0.074	47.2	0.199
20.5	0.036	33.9	0.074	47.3	0.201
20.6	0.036	34.0	0.075	47.4	0.203
20.7	0.036	34.1	0.075	47.5	0.204
20.8	0.036	34.2	0.076	47.6	0.205
20.9	0.037	34.3	0.076	47.7	0.206
21.0	0.037	34.4	0.076	47.8	0.207
21.1	0.037	34.5	0.077	47.9	0.208
21.2	0.037	34.6	0.077	48.0	0.209
21.3	0.037	34.7	0.078	48.1	0.210
21.4	0.037	34.8	0.078	48.2	0.211
21.5	0.038	34.9	0.078	48.3	0.212
21.6	0.038	35.0	0.079	48.4	0.213
21.7	0.038	35.1	0.079	48.5	0.214
21.8	0.038	35.2	0.080	48.6	0.215

21.9	0.038	35.3	0.080	48.7	0.216
22.0	0.039	35.4	0.081	48.8	0.217
22.1	0.039	35.5	0.081	48.9	0.218
22.2	0.039	35.6	0.081	49.0	0.219
22.3	0.039	35.7	0.082	49.1	0.220
22.4	0.039	35.8	0.082	49.2	0.221
22.5	0.039	35.9	0.083	49.3	0.222
22.6	0.040	36.0	0.083	49.4	0.223
22.7	0.040	36.1	0.084	49.5	0.224
22.8	0.040	36.2	0.084	49.6	0.225
22.9	0.040	36.3	0.085	49.7	0.226
23.0	0.040	36.4	0.085	49.8	0.227
23.1	0.041	36.5	0.086	49.9	0.228
23.2	0.041	36.6	0.086	50.0	0.229
23.3	0.041	36.7	0.086	50.1	0.231
23.4	0.041	36.8	0.087	50.2	0.233
23.5	0.042	36.9	0.087	50.3	0.234
23.6	0.042	37.0	0.088	50.4	0.236
23.7	0.042	37.1	0.088	50.5	0.238
23.8	0.042	37.2	0.089	50.6	0.239
23.9	0.042	37.3	0.089	50.7	0.241
24.0	0.043	37.4	0.090	50.8	0.243
24.1	0.043	37.5	0.090	50.9	0.244
24.2	0.043	37.6	0.090	51.0	0.246
24.3	0.043	37.7	0.091	51.1	0.248
24.4	0.043	37.8	0.091	51.2	0.249
24.5	0.044	37.9	0.092	51.3	0.251
24.6	0.044	38.0	0.092	51.4	0.252
24.7	0.044	38.1	0.092	51.5	0.254
24.8	0.044	38.2	0.093	51.6	0.256
24.9	0.044	38.3	0.093	51.7	0.257
25.0	0.045	38.4	0.094	51.8	0.259
25.1	0.045	38.5	0.094	51.9	0.261
25.2	0.045	38.6	0.094	52.0	0.262
25.3	0.045	38.7	0.095	52.1	0.264

25.4	0.046	38.8	0.095	52.2	0.266
25.5	0.046	38.9	0.096	52.3	0.267
25.6	0.046	39.0	0.096	52.4	0.269
25.7	0.046	39.1	0.096	52.5	0.271
25.8	0.046	39.2	0.097	52.6	0.272
25.9	0.047	39.3	0.097	52.7	0.273
26.0	0.047	39.4	0.098	52.8	0.274
26.1	0.047	39.5	0.098	52.9	0.275
26.2	0.047	39.6	0.098	53.0	0.277
26.3	0.047	39.7	0.099	53.1	0.278
26.4	0.048	39.8	0.099	53.2	0.279
26.5	0.048	39.9	0.100	53.3	0.280
26.6	0.048	40.0	0.100	53.4	0.281
26.7	0.048	40.1	0.101	53.5	0.282
26.8	0.049	40.2	0.103	53.6	0.284
26.9	0.049	40.3	0.104	53.7	0.285
27.0	0.049	40.4	0.105	53.8	0.286
27.1	0.049	40.5	0.106	53.9	0.287
27.2	0.049	40.6	0.108	54.0	0.288
27.3	0.050	40.7	0.109	54.1	0.289
27.4	0.050	40.8	0.110	54.2	0.291
27.5	0.050	40.9	0.111	54.3	0.292
27.6	0.050	41.0	0.113	54.4	0.293
27.7	0.051	41.1	0.114	54.5	0.294
27.8	0.051	41.2	0.115	54.6	0.295
27.9	0.051	41.3	0.116	54.7	0.296
28.0	0.052	41.4	0.118	54.8	0.298
28.1	0.052	41.5	0.119	54.9	0.299
28.2	0.052	41.6	0.120	55.0	0.300
28.3	0.053	41.7	0.121	55.1	0.302
28.4	0.053	41.8	0.123	55.2	0.305
28.5	0.054	41.9	0.124	55.3	0.307
28.6	0.054	42.0	0.125	55.4	0.310
28.7	0.054	42.1	0.126	55.5	0.312
28.8	0.055	42.2	0.128	55.6	0.314

28.9	0.055	42.3	0.129	55.7	0.317
29.0	0.055	42.4	0.130	55.8	0.319
29.1	0.056	42.5	0.131	55.9	0.322
29.2	0.056	42.6	0.133	56.0	0.324
29.3	0.056	42.7	0.134	56.1	0.326
29.4	0.057	42.8	0.135	56.2	0.329
29.5	0.057	42.9	0.136	56.3	0.331
29.6	0.057	43.0	0.138	56.4	0.334
29.7	0.058	43.1	0.139	56.5	0.336
29.8	0.058	43.2	0.140	56.6	0.338
29.9	0.058	43.3	0.141	56.7	0.341
30.0	0.059	43.4	0.143	56.8	0.343
30.1	0.059	43.5	0.144	56.9	0.346
30.2	0.060	43.6	0.145	57.0	0.348
30.3	0.060	43.7	0.146	57.1	0.350
30.4	0.060	43.8	0.148	57.2	0.353
30.5	0.061	43.9	0.149	57.3	0.355
30.6	0.061	44.0	0.150	57.4	0.358
30.7	0.062	44.1	0.151	57.5	0.360
30.8	0.062	44.2	0.153	57.6	0.361
30.9	0.062	44.3	0.154	57.7	0.363
31.0	0.063	44.4	0.155	57.8	0.364
31.1	0.063	44.5	0.156	57.9	0.366
31.2	0.064	44.6	0.158	58.0	0.367
31.3	0.064	44.7	0.159	58.1	0.369
31.4	0.064	44.8	0.160	58.2	0.370
31.5	0.065	44.9	0.161	58.3	0.372
31.6	0.065	45.0	0.163	58.4	0.373
31.7	0.066	45.1	0.164	58.5	0.375
31.8	0.066	45.2	0.166	58.6	0.376
31.9	0.066	45.3	0.168	58.7	0.378
32.0	0.067	45.4	0.169	58.8	0.379
32.1	0.067	45.5	0.171	58.9	0.381
32.2	0.068	45.6	0.173	59.0	0.382
32.3	0.068	45.7	0.174	59.1	0.384

32.4	0.068	45.8	0.176	59.2	0.385
32.5	0.069	45.9	0.178	59.3	0.386
32.6	0.069	46.0	0.179	59.4	0.388
32.7	0.070	46.1	0.181	59.5	0.389
32.8	0.070	46.2	0.183	59.6	0.391
32.9	0.070	46.3	0.184	59.7	0.392
33.0	0.071	46.4	0.186	59.8	0.394
33.1	0.071	46.5	0.188	59.9	0.395
33.2	0.072	46.6	0.189	60.0	0.397
33.3	0.072	46.7	0.191		

Table A.1: Temperature ($^{\circ}C$) versus Capacity Loss Look up Table

Appendix B

Effective Capacity Table

Current (mA)	Shelf time (hours)	Effective capacity (mAh)
2.0	341.67	683.34
2.1	329.27	691.47
2.2	316.87	697.11
2.3	304.46	700.26
2.4	292.06	700.94
2.5	279.66	699.15
2.6	267.26	694.88
2.7	254.86	688.12
2.8	242.45	678.86
2.9	230.05	667.15
3.0	217.65	652.95
3.1	212.66	659.25
3.2	207.67	664.54
3.3	202.68	668.84
3.4	197.69	672.15
3.5	192.69	674.42
3.6	187.70	675.72
3.7	182.71	676.03
3.8	177.72	675.34
3.9	172.73	673.65
4.0	167.74	670.96
4.1	165.15	677.12

4.2	162.56	682.75
4.3	159.97	687.87
4.4	157.38	692.47
4.5	154.79	696.56
4.6	152.19	700.07
4.7	149.60	703.12
4.8	147.01	705.65
4.9	144.42	707.66
5.0	141.83	709.15
5.1	139.24	710.12
5.2	136.65	710.58
5.3	134.06	710.52
5.4	131.47	709.94
5.5	128.88	708.84
5.6	126.28	707.17
5.7	123.69	705.03
5.8	121.10	702.38
5.9	118.51	699.21
6.0	115.92	695.52
6.1	113.33	691.31
6.2	110.74	686.59
6.3	108.15	681.35
6.4	105.56	675.58
6.5	102.97	669.31
6.6	100.37	662.44
6.7	97.78	655.13
6.8	95.19	647.29
6.9	92.60	638.94
7.0	90.01	630.07
7.1	89.16	633.04
7.2	88.09	634.25
7.3	87.04	635.39
7.4	86.00	636.40
7.5	84.97	637.28
7.6	83.95	638.02

7.7	82.94	638.64
7.8	81.95	639.21
7.9	80.97	639.66
8.0	80.00	640.00
8.1	79.04	640.22
8.2	78.09	640.34
8.3	77.16	640.43
8.4	76.23	640.33
8.5	75.32	640.22
8.6	74.42	640.01
8.7	73.53	639.71
8.8	72.65	639.32
8.9	71.78	638.84
9.0	70.92	638.28
9.1	70.07	637.64
9.2	69.23	636.92
9.3	68.40	636.12
9.4	67.58	635.25
9.5	66.77	634.32
9.6	65.97	633.31
9.7	65.18	632.25
9.8	64.40	631.12
9.9	63.63	629.94
10.0	62.87	628.70
10.1	62.39	630.14
10.2	61.92	631.58
10.3	61.45	632.94
10.4	60.99	634.30
10.5	60.53	635.57
10.6	60.07	636.74
10.7	59.61	637.83
10.8	59.16	638.93
10.9	58.72	640.05
11.0	58.28	641.08
11.1	57.84	642.02

11.2	57.40	642.88
11.3	56.97	643.76
11.4	56.54	644.56
11.5	56.11	645.27
11.6	55.69	646.00
11.7	55.27	646.66
11.8	54.85	647.23
11.9	54.43	647.72
12.0	54.02	648.24
12.1	53.62	648.80
12.2	53.21	649.16
12.3	52.81	649.56
12.4	52.41	649.88
12.5	52.02	650.25
12.6	51.62	650.41
12.7	51.23	650.62
12.8	50.85	650.88
12.9	50.46	650.93
13.0	50.08	651.04
13.1	49.70	651.07
13.2	49.33	651.16
13.3	48.96	651.17
13.4	48.59	651.11
13.5	48.22	650.97
13.6	47.86	650.90
13.7	47.50	650.75
13.8	47.14	650.53
13.9	46.78	650.24
14.0	46.43	650.02
14.1	46.08	649.73
14.2	45.73	649.37
14.3	45.38	648.93
14.4	45.04	648.58
14.5	44.70	648.15
14.6	44.36	647.66

14.7	44.03	647.24
14.8	43.70	646.76
14.9	43.37	646.21
15.0	43.04	645.60
15.1	42.72	645.07
15.2	42.39	644.33
15.3	42.07	643.67
15.4	41.76	643.10
15.5	41.44	642.32
15.6	41.13	641.63
15.7	40.82	640.87
15.8	40.51	640.06
15.9	40.20	639.18
16.0	39.90	638.40
16.1	39.60	637.56
16.2	39.30	636.66
16.3	39.00	635.70
16.4	38.71	634.84
16.5	38.42	633.93
16.6	38.13	632.96
16.7	37.84	631.93
16.8	37.55	630.84
16.9	37.27	629.86
17.0	36.99	628.83
17.1	36.71	627.74
17.2	36.43	626.60
17.3	36.16	625.57
17.4	35.89	624.49
17.5	35.61	623.18
17.6	35.35	622.16
17.7	35.08	620.92
17.8	34.81	619.62
17.9	34.55	618.45
18.0	34.29	617.22
18.1	34.03	615.94

18.2	33.78	614.80
18.3	33.52	613.42
18.4	33.27	612.17
18.5	33.02	610.87
18.6	32.77	609.52
18.7	32.52	608.12
18.8	32.27	606.68
18.9	32.03	605.37
19.0	31.79	604.01
19.1	31.55	602.61
19.2	31.31	601.15
19.3	31.07	599.65
19.4	30.84	598.30
19.5	30.61	596.90
19.6	30.38	595.45
19.7	30.15	593.96
19.8	29.92	592.42
19.9	29.69	590.83
20.0	30.00	600.00
20.1	29.86	600.19
20.2	29.73	600.55
20.3	29.59	600.68
20.4	29.46	600.98
20.5	29.32	601.06
20.6	29.19	601.31
20.7	29.06	601.54
20.8	28.92	601.54
20.9	28.79	601.71
21.0	28.66	601.86
21.1	28.53	601.98
21.2	28.40	602.08
21.3	28.27	602.15
21.4	28.14	602.20
21.5	28.01	602.22
21.6	27.89	602.42

21.7	27.76	602.39
21.8	27.63	602.33
21.9	27.51	602.47
22.0	27.38	602.36
22.1	27.26	602.45
22.2	27.13	602.29
22.3	27.01	602.32
22.4	26.89	602.34
22.5	26.76	602.10
22.6	26.64	602.06
22.7	26.52	602.00
22.8	26.40	601.92
22.9	26.28	601.81
23.0	26.16	601.68
23.1	26.04	601.52
23.2	25.92	601.34
23.3	25.80	601.14
23.4	25.69	601.15
23.5	25.57	600.90
23.6	25.45	600.62
23.7	25.34	600.56
23.8	25.22	600.24
23.9	25.11	600.13
24.0	24.99	599.76
24.1	24.88	599.61
24.2	24.76	599.19
24.3	24.65	599.00
24.4	24.54	598.78
24.5	24.43	598.54
24.6	24.32	598.27
24.7	24.21	597.99
24.8	24.09	597.43
24.9	23.99	597.35
25.0	23.88	597.00
25.1	23.77	596.63

25.2	23.66	596.23
25.3	23.55	595.82
25.4	23.44	595.38
25.5	23.34	595.17
25.6	23.23	594.69
25.7	23.12	594.18
25.8	23.02	593.92
25.9	22.91	593.37
26.0	22.81	593.06
26.1	22.71	592.73
26.2	22.60	592.12
26.3	22.50	591.75
26.4	22.40	591.36
26.5	22.30	590.95
26.6	22.19	590.25
26.7	22.09	589.80
26.8	21.99	589.33
26.9	21.89	588.84
27.0	21.79	588.33
27.1	21.69	587.80
27.2	21.59	587.25
27.3	21.50	586.95
27.4	21.40	586.36
27.5	21.30	585.75
27.6	21.20	585.12
27.7	21.11	584.75
27.8	21.01	584.08
27.9	20.91	583.39
28.0	20.82	582.96
28.1	20.72	582.23
28.2	20.63	581.77
28.3	20.54	581.28
28.4	20.44	580.50
28.5	20.35	579.98
28.6	20.26	579.44

28.7	20.16	578.59
28.8	20.07	578.02
28.9	19.98	577.42
29.0	19.89	576.81
29.1	19.80	576.18
29.2	19.71	575.53
29.3	19.62	574.87
29.4	19.53	574.18
29.5	19.44	573.48
29.6	19.35	572.76
29.7	19.26	572.02
29.8	19.18	571.56
29.9	19.09	570.79
30.0	19.00	570.00
30.1	18.95	570.40
30.2	18.90	570.78
30.3	18.85	571.16
30.4	18.80	571.52
30.5	18.76	572.18
30.6	18.71	572.53
30.7	18.66	572.86
30.8	18.61	573.19
30.9	18.56	573.50
31.0	18.52	574.12
31.1	18.47	574.42
31.2	18.42	574.70
31.3	18.37	574.98
31.4	18.33	575.56
31.5	18.28	575.82
31.6	18.23	576.07
31.7	18.18	576.31
31.8	18.14	576.85
31.9	18.09	577.07
32.0	18.04	577.28
32.1	18.00	577.80

32.2	17.95	577.99
32.3	17.90	578.17
32.4	17.86	578.66
32.5	17.81	578.83
32.6	17.77	579.30
32.7	17.72	579.44
32.8	17.67	579.58
32.9	17.63	580.03
33.0	17.58	580.14
33.1	17.54	580.57
33.2	17.49	580.67
33.3	17.45	581.09
33.4	17.40	581.16
33.5	17.36	581.56
33.6	17.31	581.62
33.7	17.27	582.00
33.8	17.22	582.04
33.9	17.18	582.40
34.0	17.13	582.42
34.1	17.09	582.77
34.2	17.05	583.11
34.3	17.00	583.10
34.4	16.96	583.42
34.5	16.91	583.40
34.6	16.87	583.70
34.7	16.83	584.00
34.8	16.78	583.94
34.9	16.74	584.23
35.0	16.70	584.50
35.1	16.65	584.42
35.2	16.61	584.67
35.3	16.57	584.92
35.4	16.53	585.16
35.5	16.48	585.04
35.6	16.44	585.26

35.7	16.40	585.48
35.8	16.36	585.69
35.9	16.31	585.53
36.0	16.27	585.72
36.1	16.23	585.90
36.2	16.19	586.08
36.3	16.15	586.25
36.4	16.10	586.04
36.5	16.06	586.19
36.6	16.02	586.33
36.7	15.98	586.47
36.8	15.94	586.59
36.9	15.90	586.71
37.0	15.86	586.82
37.1	15.82	586.92
37.2	15.77	586.64
37.3	15.73	586.73
37.4	15.69	586.81
37.5	15.65	586.88
37.6	15.61	586.94
37.7	15.57	586.99
37.8	15.53	587.03
37.9	15.49	587.07
38.0	15.45	587.10
38.1	15.41	587.12
38.2	15.37	587.13
38.3	15.33	587.14
38.4	15.29	587.14
38.5	15.25	587.13
38.6	15.21	587.11
38.7	15.18	587.47
38.8	15.14	587.43
38.9	15.10	587.39
39.0	15.06	587.34
39.1	15.02	587.28

39.2	14.98	587.22
39.3	14.94	587.14
39.4	14.90	587.06
39.5	14.86	586.97
39.6	14.83	587.27
39.7	14.79	587.16
39.8	14.75	587.05
39.9	14.71	586.93
40.0	14.67	586.80
40.1	14.64	587.06
40.2	14.60	586.92
40.3	14.56	586.77
40.4	14.52	586.61
40.5	14.49	586.85
40.6	14.45	586.67
40.7	14.41	586.49
40.8	14.37	586.30
40.9	14.34	586.51
41.0	14.30	586.30
41.1	14.26	586.09
41.2	14.23	586.28
41.3	14.19	586.05
41.4	14.15	585.81
41.5	14.12	585.98
41.6	14.08	585.73
41.7	14.04	585.47
41.8	14.01	585.62
41.9	13.97	585.34
42.0	13.93	585.06
42.1	13.90	585.19
42.2	13.86	584.89
42.3	13.83	585.01
42.4	13.79	584.70
42.5	13.76	584.80
42.6	13.72	584.47

42.7	13.68	584.14
42.8	13.65	584.22
42.9	13.61	583.87
43.0	13.58	583.94
43.1	13.54	583.57
43.2	13.51	583.63
43.3	13.47	583.25
43.4	13.44	583.30
43.5	13.40	582.90
43.6	13.37	582.93
43.7	13.34	582.96
43.8	13.30	582.54
43.9	13.27	582.55
44.0	13.23	582.12
44.1	13.20	582.12
44.2	13.16	581.67
44.3	13.13	581.66
44.4	13.10	581.64
44.5	13.06	581.17
44.6	13.03	581.14
44.7	13.00	581.10
44.8	12.96	580.61
44.9	12.93	580.56
45.0	12.90	580.50
45.1	12.86	579.99
45.2	12.83	579.92
45.3	12.80	579.84
45.4	12.76	579.30
45.5	12.73	579.22
45.6	12.70	579.12
45.7	12.66	578.56
45.8	12.63	578.45
45.9	12.60	578.34
46.0	12.57	578.22
46.1	12.53	577.63

46.2	12.50	577.50
46.3	12.47	577.36
46.4	12.44	577.22
46.5	12.41	577.07
46.6	12.37	576.44
46.7	12.34	576.28
46.8	12.31	576.11
46.9	12.28	575.93
47.0	12.25	575.75
47.1	12.21	575.09
47.2	12.18	574.90
47.3	12.15	574.70
47.4	12.12	574.49
47.5	12.09	574.28
47.6	12.06	574.06
47.7	12.03	573.83
47.8	12.00	573.60
47.9	11.96	572.88
48.0	11.93	572.64
48.1	11.90	572.39
48.2	11.87	572.13
48.3	11.84	571.87
48.4	11.81	571.60
48.5	11.78	571.33
48.6	11.75	571.05
48.7	11.72	570.76
48.8	11.69	570.47
48.9	11.66	570.17
49.0	11.63	569.87
49.1	11.60	569.56
49.2	11.57	569.24
49.3	11.54	568.92
49.4	11.51	568.59
49.5	11.48	568.26
49.6	11.45	567.92

49.7	11.42	567.57
49.8	11.39	567.22
49.9	11.36	566.86
50.0	11.33	566.50
50.1	11.30	566.13
50.2	11.27	565.75
50.3	11.24	565.37
50.4	11.22	565.49
50.5	11.19	565.10
50.6	11.16	564.70
50.7	11.13	564.29
50.8	11.10	563.88
50.9	11.07	563.46
51.0	11.04	563.04
51.1	11.01	562.61
51.2	10.99	562.69
51.3	10.96	562.25
51.4	10.93	561.80
51.5	10.90	561.35
51.6	10.87	560.89
51.7	10.85	560.95
51.8	10.82	560.48
51.9	10.79	560.00
52.0	10.76	559.52
52.1	10.73	559.03
52.2	10.71	559.06
52.3	10.68	558.56
52.4	10.65	558.06
52.5	10.62	557.55
52.6	10.60	557.56
52.7	10.57	557.04
52.8	10.54	556.51
52.9	10.51	555.98
53.0	10.49	555.97
53.1	10.46	555.43

53.2	10.43	554.88
53.3	10.41	554.85
53.4	10.38	554.29
53.5	10.35	553.73
53.6	10.33	553.69
53.7	10.30	553.11
53.8	10.27	552.53
53.9	10.25	552.48
54.0	10.22	551.88
54.1	10.19	551.28
54.2	10.17	551.21
54.3	10.14	550.60
54.4	10.11	549.98
54.5	10.09	549.91
54.6	10.06	549.28
54.7	10.04	549.19
54.8	10.01	548.55
54.9	9.98	547.90
55.0	9.96	547.80
55.1	9.93	547.14
55.2	9.91	547.03
55.3	9.88	546.36
55.4	9.86	546.24
55.5	9.83	545.57
55.6	9.81	545.44
55.7	9.78	544.75
55.8	9.76	544.61
55.9	9.73	543.91
56.0	9.70	543.20
56.1	9.68	543.05
56.2	9.65	542.33
56.3	9.63	542.17
56.4	9.61	542.00
56.5	9.58	541.27
56.6	9.56	541.10

56.7	9.53	540.35
56.8	9.51	540.17
56.9	9.48	539.41
57.0	9.46	539.22
57.1	9.43	538.45
57.2	9.41	538.25
57.3	9.38	537.47
57.4	9.36	537.26
57.5	9.34	537.05
57.6	9.31	536.26
57.7	9.29	536.03
57.8	9.26	535.23
57.9	9.24	535.00
58.0	9.22	534.76
58.1	9.19	533.94
58.2	9.17	533.69
58.3	9.14	532.86
58.4	9.12	532.61
58.5	9.10	532.35
58.6	9.07	531.50
58.7	9.05	531.24
58.8	9.03	530.96
58.9	9.00	530.10
59.0	8.98	529.82
59.1	8.96	529.54
59.2	8.93	528.66
59.3	8.91	528.36
59.4	8.89	528.07
59.5	8.87	527.77
59.6	8.84	526.86
59.7	8.82	526.55
59.8	8.80	526.24
59.9	8.77	525.32
60.0	9.00	540.00
60.1	8.99	540.30

60.2	8.98	540.60
60.3	8.97	540.89
60.4	8.96	541.18
60.5	8.95	541.48
60.6	8.94	541.76
60.7	8.93	542.05
60.8	8.92	542.34
60.9	8.91	542.62
61.0	8.90	542.90
61.1	8.89	543.18
61.2	8.88	543.46
61.3	8.87	543.73
61.4	8.86	544.00
61.5	8.85	544.28
61.6	8.84	544.54
61.7	8.83	544.81
61.8	8.82	545.08
61.9	8.81	545.34
62.0	8.80	545.60
62.1	8.79	545.86
62.2	8.78	546.12
62.3	8.77	546.37
62.4	8.76	546.62
62.5	8.75	546.88
62.6	8.74	547.12
62.7	8.73	547.37
62.8	8.72	547.62
62.9	8.71	547.86
63.0	8.70	548.10
63.1	8.69	548.34
63.2	8.68	548.58
63.3	8.67	548.81
63.4	8.66	549.04
63.5	8.65	549.28
63.6	8.64	549.50

63.7	8.63	549.73
63.8	8.62	549.96
63.9	8.61	550.18
64.0	8.60	550.40
64.1	8.59	550.62
64.2	8.58	550.84
64.3	8.57	551.05
64.4	8.56	551.26
64.5	8.55	551.48
64.6	8.54	551.68
64.7	8.53	551.89
64.8	8.52	552.10
64.9	8.51	552.30
65.0	8.50	552.50
65.1	8.49	552.70
65.2	8.48	552.90
65.3	8.47	553.09
65.4	8.46	553.28
65.5	8.45	553.48
65.6	8.44	553.66
65.7	8.43	553.85
65.8	8.42	554.04
65.9	8.41	554.22
66.0	8.40	554.40
66.1	8.39	554.58
66.2	8.38	554.76
66.3	8.37	554.93
66.4	8.36	555.10
66.5	8.35	555.28
66.6	8.34	555.44
66.7	8.33	555.61
66.8	8.32	555.78
66.9	8.31	555.94
67.0	8.30	556.10
67.1	8.29	556.26

67.2	8.28	556.42
67.3	8.27	556.57
67.4	8.26	556.72
67.5	8.25	556.88
67.6	8.24	557.02
67.7	8.23	557.17
67.8	8.22	557.32
67.9	8.21	557.46
68.0	8.20	557.60
68.1	8.19	557.74
68.2	8.18	557.88
68.3	8.17	558.01
68.4	8.16	558.14
68.5	8.15	558.28
68.6	8.14	558.40
68.7	8.13	558.53
68.8	8.12	558.66
68.9	8.11	558.78
69.0	8.10	558.90
69.1	8.09	559.02
69.2	8.08	559.14
69.3	8.07	559.25
69.4	8.06	559.36
69.5	8.05	559.48
69.6	8.04	559.58
69.7	8.03	559.69
69.8	8.02	559.80
69.9	8.01	559.90
70.0	8.00	560.00
70.1	7.99	560.10
70.2	7.98	560.20
70.3	7.97	560.29
70.4	7.96	560.38
70.5	7.95	560.48
70.6	7.94	560.56

70.7	7.93	560.65
70.8	7.92	560.74
70.9	7.91	560.82
71.0	7.90	560.90
71.1	7.89	560.98
71.2	7.88	561.06
71.3	7.87	561.13
71.4	7.86	561.20
71.5	7.85	561.28
71.6	7.84	561.34
71.7	7.83	561.41
71.8	7.82	561.48
71.9	7.81	561.54
72.0	7.80	561.60
72.1	7.79	561.66
72.2	7.78	561.72
72.3	7.77	561.77
72.4	7.76	561.82
72.5	7.75	561.88
72.6	7.74	561.92
72.7	7.73	561.97
72.8	7.72	562.02
72.9	7.71	562.06
73.0	7.70	562.10
73.1	7.69	562.14
73.2	7.68	562.18
73.3	7.67	562.21
73.4	7.66	562.24
73.5	7.65	562.28
73.6	7.64	562.30
73.7	7.63	562.33
73.8	7.62	562.36
73.9	7.61	562.38
74.0	7.60	562.40
74.1	7.59	562.42

74.2	7.58	562.44
74.3	7.57	562.45
74.4	7.56	562.46
74.5	7.55	562.48
74.6	7.54	562.48
74.7	7.53	562.49
74.8	7.52	562.50
74.9	7.51	562.50
75.0	7.50	562.50
75.1	7.49	562.50
75.2	7.48	562.50
75.3	7.47	562.49
75.4	7.46	562.48
75.5	7.45	562.48
75.6	7.44	562.46
75.7	7.43	562.45
75.8	7.42	562.44
75.9	7.41	562.42
76.0	7.40	562.40
76.1	7.39	562.38
76.2	7.38	562.36
76.3	7.37	562.33
76.4	7.36	562.30
76.5	7.35	562.28
76.6	7.34	562.24
76.7	7.33	562.21
76.8	7.32	562.18
76.9	7.31	562.14
77.0	7.30	562.10
77.1	7.29	562.06
77.2	7.28	562.02
77.3	7.27	561.97
77.4	7.26	561.92
77.5	7.25	561.88
77.6	7.24	561.82

77.7	7.23	561.77
77.8	7.22	561.72
77.9	7.21	561.66
78.0	7.20	561.60
78.1	7.19	561.54
78.2	7.18	561.48
78.3	7.17	561.41
78.4	7.16	561.34
78.5	7.15	561.28
78.6	7.14	561.20
78.7	7.13	561.13
78.8	7.12	561.06
78.9	7.11	560.98
79.0	7.10	560.90
79.1	7.09	560.82
79.2	7.08	560.74
79.3	7.07	560.65
79.4	7.06	560.56
79.5	7.05	560.48
79.6	7.04	560.38
79.7	7.03	560.29
79.8	7.02	560.20
79.9	7.01	560.10
80.0	7.00	560.00
80.1	6.99	559.90
80.2	6.98	559.80
80.3	6.97	559.69
80.4	6.96	559.58
80.5	6.95	559.48
80.6	6.94	559.36
80.7	6.93	559.25
80.8	6.92	559.14
80.9	6.91	559.02
81.0	6.90	558.90
81.1	6.89	558.78

81.2	6.88	558.66
81.3	6.87	558.53
81.4	6.86	558.40
81.5	6.85	558.28
81.6	6.84	558.14
81.7	6.83	558.01
81.8	6.82	557.88
81.9	6.81	557.74
82.0	6.80	557.60
82.1	6.79	557.46
82.2	6.78	557.32
82.3	6.77	557.17
82.4	6.76	557.02
82.5	6.75	556.88
82.6	6.74	556.72
82.7	6.73	556.57
82.8	6.72	556.42
82.9	6.71	556.26
83.0	6.70	556.10
83.1	6.69	555.94
83.2	6.68	555.78
83.3	6.67	555.61
83.4	6.66	555.44
83.5	6.65	555.28
83.6	6.64	555.10
83.7	6.63	554.93
83.8	6.62	554.76
83.9	6.61	554.58
84.0	6.60	554.40
84.1	6.59	554.22
84.2	6.58	554.04
84.3	6.57	553.85
84.4	6.56	553.66
84.5	6.55	553.48
84.6	6.54	553.28

84.7	6.53	553.09
84.8	6.52	552.90
84.9	6.51	552.70
85.0	6.50	552.50
85.1	6.49	552.30
85.2	6.48	552.10
85.3	6.47	551.89
85.4	6.46	551.68
85.5	6.45	551.48
85.6	6.44	551.26
85.7	6.43	551.05
85.8	6.42	550.84
85.9	6.41	550.62
86.0	6.40	550.40
86.1	6.39	550.18
86.2	6.38	549.96
86.3	6.37	549.73
86.4	6.36	549.50
86.5	6.35	549.28
86.6	6.34	549.04
86.7	6.33	548.81
86.8	6.32	548.58
86.9	6.31	548.34
87.0	6.30	548.10
87.1	6.29	547.86
87.2	6.28	547.62
87.3	6.27	547.37
87.4	6.26	547.12
87.5	6.25	546.88
87.6	6.24	546.62
87.7	6.23	546.37
87.8	6.22	546.12
87.9	6.21	545.86
88.0	6.20	545.60
88.1	6.19	545.34

88.2	6.18	545.08
88.3	6.17	544.81
88.4	6.16	544.54
88.5	6.15	544.28
88.6	6.14	544.00
88.7	6.13	543.73
88.8	6.12	543.46
88.9	6.11	543.18
89.0	6.10	542.90
89.1	6.09	542.62
89.2	6.08	542.34
89.3	6.07	542.05
89.4	6.06	541.76
89.5	6.05	541.48
89.6	6.04	541.18
89.7	6.03	540.89
89.8	6.02	540.60
89.9	6.01	540.30
90.0	6.00	540.00
90.1	5.99	539.70
90.2	5.98	539.40
90.3	5.97	539.09
90.4	5.96	538.78
90.5	5.95	538.48
90.6	5.94	538.16
90.7	5.93	537.85
90.8	5.92	537.54
90.9	5.91	537.22
91.0	5.90	536.90
91.1	5.89	536.58
91.2	5.88	536.26
91.3	5.87	535.93
91.4	5.86	535.60
91.5	5.85	535.28
91.6	5.84	534.94

91.7	5.83	534.61
91.8	5.82	534.28
91.9	5.81	533.94
92.0	5.80	533.60
92.1	5.79	533.26
92.2	5.78	532.92
92.3	5.77	532.57
92.4	5.76	532.22
92.5	5.75	531.88
92.6	5.74	531.52
92.7	5.73	531.17
92.8	5.72	530.82
92.9	5.71	530.46
93.0	5.70	530.10
93.1	5.69	529.74
93.2	5.68	529.38
93.3	5.67	529.01
93.4	5.66	528.64
93.5	5.65	528.28
93.6	5.64	527.90
93.7	5.63	527.53
93.8	5.62	527.16
93.9	5.61	526.78
94.0	5.60	526.40
94.1	5.59	526.02
94.2	5.58	525.64
94.3	5.57	525.25
94.4	5.56	524.86
94.5	5.55	524.48
94.6	5.54	524.08
94.7	5.53	523.69
94.8	5.52	523.30
94.9	5.51	522.90
95.0	5.50	522.50
95.1	5.49	522.10

95.2	5.48	521.70
95.3	5.47	521.29
95.4	5.46	520.88
95.5	5.45	520.48
95.6	5.44	520.06
95.7	5.43	519.65
95.8	5.42	519.24
95.9	5.41	518.82
96.0	5.40	518.40
96.1	5.39	517.98
96.2	5.38	517.56
96.3	5.37	517.13
96.4	5.36	516.70
96.5	5.35	516.28
96.6	5.34	515.84
96.7	5.33	515.41
96.8	5.32	514.98
96.9	5.31	514.54
97.0	5.30	514.10
97.1	5.29	513.66
97.2	5.28	513.22
97.3	5.27	512.77
97.4	5.26	512.32
97.5	5.25	511.88
97.6	5.24	511.42
97.7	5.23	510.97
97.8	5.22	510.52
97.9	5.21	510.06
98.0	5.20	509.60
98.1	5.19	509.14
98.2	5.18	508.68
98.3	5.17	508.21
98.4	5.16	507.74
98.5	5.15	507.28
98.6	5.14	506.80

98.7	5.13	506.33
98.8	5.12	505.86
98.9	5.11	505.38
99.0	5.10	504.90
99.1	5.09	504.42
99.2	5.08	503.94
99.3	5.07	503.45
99.4	5.06	502.96
99.5	5.05	502.48
99.6	5.04	501.98
99.7	5.03	501.49
99.8	5.02	501.00
99.9	5.01	500.50
100.0	5.00	500.0

Table B.1: Effective Capacity

Appendix C

CoojaED Extension source code

```
1 private MicrosecondObservable microsecondObservable =
2         new MicrosecondObservable ();
3 private class MicrosecondObservable extends Observable {
4     private void newMicrosecond(long time) {
5         setChanged ();
6         notifyObservers (time);
7     }
8 }
9 public void addMicrosecondObserver (Observer newObserver) {
10     microsecondObservable.addObserver (newObserver);
11     hasMicrosecondObservers = true;
12
13     invokeSimulationThread (new Runnable () {
14         public void run () {
15             if (!microsecondEvent.isScheduled ()) {
16                 scheduleEvent (
17                     microsecondEvent,
18                     currentSimulationTime - (currentSimulationTime
19                         \% MICROSECOND) + simTimeLog + MICROSECOND);
20             }
21         }
22     });
23 }
24 public void deleteMicrosecondObserver (Observer observer) {
25     microsecondObservable.deleteObserver (observer);
26     hasMicrosecondObservers = microsecondObservable.countObservers () > 0;
27 }
```

Listing C.1: New observer

```
1 private double radioOutputPower(){
2     double outputPower;
3     int radioOutputPower = radio.getOutputPower();
4
5     switch(radioOutputPower){
6         case -1: outputPower = 16.5;
7             break;
8         case -3: outputPower = 15.2;
9             break;
10        case -5: outputPower = 13.9;
11            break;
12        case -7: outputPower = 12.5;
13            break;
14        case -10: outputPower = 11.2;
15            break;
16        case -15: outputPower = 9.9;
17            break;
18        case -25: outputPower = 8.5;
19            break;
20        default: outputPower = 17.4;
21            break;
22    }
23    return (outputPower);
24 }
```

Listing C.2: Track the currently radio power transmission and their corresponding typical current consumption

```
1 private void calculateCurrentConsumption() {
2
3 simTime = simulation.getSimulationTimeMillis();
4 simTimeSec = simTime / 1000;
5 modeRadio = radio.getMode();
6 modeCPU = cpu.getMode();
7
8 radioCurrent = 0;
9 cpuCurrent = 0;
10 ledRedCurrent = 0;
11 ledGreenCurrent = 0;
12 ledBlueCurrent = 0;
13 ledCurrent = 0;
14 cpuRadioCurrent = 0;
15
16
17 if(currentConfiguration == 0){
18     if (modeRadio == 1 && modeCPU == 0){
19         cpuRadioCurrent = currentCPUOnRadioRx;
20     }
21     else if(modeRadio == 2 && modeCPU == 0){
22         cpuRadioCurrent = currentCPUOnRadioTx;
23     }
24     else if (modeCPU == 0 && (modeRadio == 3 || modeRadio == 0)){
25         cpuRadioCurrent = currentCPUOnRadioOff;
26     }
27     else if (modeCPU != 0 && modeRadio == 3){
28         cpuRadioCurrent = currentCPUOffRadioOff;
29     }
30     else if(modeRadio == 1 && modeCPU != 0){
31         cpuRadioCurrent = currentCPUOffRadioRx;
32     }
33     else if(modeRadio == 2 && modeCPU != 0){
34         cpuRadioCurrent = currentCPUOffRadioTx;
35     }
36 }
37 else{
38     if(modeRadio == 3){
39         if(radio.getState() == CC2420.RadioState.VREG_OFF)
40             radioCurrent = currentRadioMode_3_1;
41         else if (radio.getState() == CC2420.RadioState.POWER_DOWN)
42             radioCurrent = currentRadioMode_3_2;
43     }
```



```
44     else if(modeRadio == 1){
45         radioCurrent = currentRadioMode_1;
46     }
47     else if(modeRadio == 0){
48         radioCurrent = currentRadioMode_0;
49     }
50     else if(modeRadio == 2) {
51         radioCurrent = currentRadioMode_2; //radioOutputPower();
52     }
53
54     switch (modeCPU){
55         case 0: cpuCurrent = currentCPUModeActive;
56             break;
57         case 1: cpuCurrent = currentCPUModeLpm_0;
58             break;
59         case 3: cpuCurrent = currentCPUModeLpm_2;
60             break;
61         case 4: cpuCurrent = currentCPUModeLpm_3;
62             break;
63         case 5: cpuCurrent = currentCPUModeLpm_4;
64             break;
65         default: cpuCurrent = currentCPUModeLpm_1;
66             break;
67     }
68
69 }
70 if (moteLEDs.isRedOn()){
71     modeLED1 = 1;
72     ledRedCurrent = currentLedRed;
73 }
74 else { modeLED1 = 0;}
75 if (moteLEDs.isGreenOn()){
76     modeLED2 = 1;
77     ledGreenCurrent = currentLedGreen;
78 }
79 else {modeLED2 = 0;}
80 if(moteLEDs.isYellowOn()){
81     modeLED3 = 1;
82     ledBlueCurrent = currentLedBlue;
83 }
84 else { modeLED3 = 0;}
85
86 ledCurrent = ledRedCurrent + ledGreenCurrent + ledBlueCurrent;
87
```

```
88 if(currentConfiguration == 0){
89     currentConsumptionTotal = cpuRadioCurrent + ledCurrent;
90 }
91 else{
92     currentConsumptionTotal = cpuCurrent + radioCurrent + ledCurrent;
93 }
94
95 CurrentConsumptionLog.add(simTimeSec + ", " +
96                             Double.toString(currentConsumptionTotal));
97 }
```

Listing C.3: Instantaneous current consumption estimation

```

1 private static void readEffectiveCapacity(String inputFile){
2     String row = "";
3
4     try {
5         File inputWorkbook = new File(inputFile);
6         workbook w;
7         w = Workbook.getWorkbook(inputWorkbook);
8         Sheet sheet = w.getSheet(0);
9
10        for (int i = 0; i < sheet.getRows(); i++){
11            for (int j = 0; j < (sheet.getColumns() - 1); j++){
12                Cell cell = sheet.getCell(j, i);
13                row += cell.getContents() + "␣";
14            }
15            effectiveCapacity.add(row);
16            row = "";
17        }
18    } catch (BiffException e) {
19        e.printStackTrace();
20    } catch (Exception e) {
21        System.err.println("Error:␣" + e.getMessage());
22        logger.fatal("Could␣not␣read␣to␣file:␣" +
23            "EffectiveCapacity.xls");
24        return;
25    }
26 }
27
28 private double searchEffectiveCapacity(String im){
29     double ceff = 0.0;
30     String [] line;
31
32     for (int i = 0; i < capacidadeEfetiva.size() - 1; i++){
33         line = capacidadeEfetiva.get(i).split("␣");
34
35         if(Double.parseDouble(line[0].trim()) == Double.parseDouble(im)){
36             ceff = Double.parseDouble(line[2].trim());
37             break;
38         }
39     }
40     return (ceff);
41 }

```

Listing C.4: Read effective capacity

```

1 private static void readCapacityLoss(String inputFile){
2     String row = "";
3
4     try {
5         File inputWorkbook = new File(inputFile);
6         Workbook w;
7         w = Workbook.getWorkbook(inputWorkbook);
8         Sheet sheet = w.getSheet(0);
9
10        for (int i = 0; i < sheet.getRows(); i++)        {
11            for (int j = 0; j < (sheet.getColumns() - 1); j++) {
12                Cell cell = sheet.getCell(j, i);
13                row += cell.getContents() + "\n";
14            }
15            capacityLoss.add(row);
16            row = "";
17        }
18    } catch (BiffException e) {
19        e.printStackTrace();
20    } catch (Exception e) {
21        System.err.println("Error:\n" + e.getMessage());
22        logger.fatal("Could not read to file:\n" + "CapacityLoss.xls");
23        return;
24    }
25 }
26
27 private double searchCapacityLoss(double t){
28     double pc = 0.0;
29     String [] line;
30
31     for (int i = 0; i < perdaCapacidade.size(); i++)
32     {
33         line = perdaCapacidade.get(i).split("\n");
34         if(Double.parseDouble(line[0].trim()) == t){
35             pc = Double.parseDouble(line[1].trim());
36             break;
37         }
38     }
39     return (pc);
40 }

```

Listing C.5: Read capacity loss

```
1 private double recoveryCapacity(double toff)
2 {
3     double crt = 0;
4
5     if(toff < 1)
6         crt = 0;
7     else if(toff >= 1 && toff <= 4)
8         crt = (0.051 * toff) + 0.445;
9     else if (toff > 4)
10        crt = 0.540;
11    return (crt);
12 }
```

Listing C.6: Recovery Capacity

```
1 private double capacityLoss(double pc, long years)
2 {
3     double pc1 = pc * (years / secondToYears);
4     return (pc1);
5 }
```

Listing C.7: Capacity Loss

```

1 private void batteryModelBMAB(ArrayList<String> logFile ,
2                               double time_sample)
3 {
4     double current = 0, relaxationEffect , sample = 0;
5     int size;
6     String [] line;
7     size = logFile.size();
8     sample = time_sample / 1000;
9
10    if (meanFlag == false){
11        meanCurrent(logFile);
12        ceff = searchEffectiveCapacity(imS);
13        meanFlag = true;
14    }
15
16    try{
17        for (int i = 0; i < logFile.size(); i++)
18            {
19                line = logFile.get(i).split(",");
20                current = Double.parseDouble(line[1].trim());
21
22                if(current >= current_cutoff){
23                    sum_1 = sum_1 + (current * sample);
24                }
25
26                if(current <= current_cutoff){
27                    if(flagPeriods == false){
28                        flagPeriods = true;
29                    }
30                    toff = toff + sample;
31                    flagSum = true;
32                }
33            else{
34                flagPeriods = false;
35                if(flagSum == true){
36                    sum_2 = sum_2 + (sleep_current * toff * 1);
37                    crt = recoveryCapacity(toff);
38                    sum_3 = sum_3 + (crt * 1);
39                    toff = 0;
40                }
41            }
42
43            if(i == (timeShowLog - 1)){

```

```
44         toff = toff / 1000;
45         simTime = simulation.getSimulationTimeMillis();
46         years = simTime /1000;
47
48         pc = searchCapacityLoss(temperature);
49         pca = capacityLoss(pc,years);
50         rca = capacityRetention(pca);
51         ceffa = finalEffectiveCapacity(rca, ceff);
52
53         sum_1 = sum_1 /3600;
54         sum_2 = sum_2 /1000;
55         sum_2 = sum_2 /3600;
56         sum_3 = sum_3 /1000;
57
58         sum_1Total = sum_1Total + sum_1;
59         sum_2Total = sum_2Total + sum_2;
60         sum_3Total = sum_3Total + sum_3;
61
62         energyTotal = energyTotal + (sum_1 + sum_2);
63         energyAvailable = (ceffa - (energyTotal + sum_2Total)) + sum
64         sum_1 = 0; sum_2 = 0; sum_3 = 0;
65     }
66     }
67     }catch (Exception e){
68         System.err.println("Error:␣" + e.getMessage());
69     }
70 }
71 }
```

Listing C.8: Battery remaining capacity