GUSTAVO ALONSO NUÑEZ SEGURA

**Energy consumption prediction in software-defined
wireless sensor networks**

São Paulo
2018

GUSTAVO ALONSO NUÑEZ SEGURA

**Energy consumption prediction in software-defined
wireless sensor networks**

Dissertation submitted to Escola Politécnica
da Universidade de São Paulo in partial
fulfillment of the requirements for the degree
of Master of Science.

São Paulo
2018

GUSTAVO ALONSO NUÑEZ SEGURA

**Energy consumption prediction in software-defined
wireless sensor networks**

Dissertation submitted to Escola Politécnica
da Universidade de São Paulo in partial
fulfillment of the requirements for the degree
of Master of Science.

Concentration area:

Computer Engineering

Advisor:

Profa. Dra. Cintia Borges Margi

São Paulo
2018

Este exemplar foi revisado e corrigido em relação à versão original, sob responsabilidade única do autor e com a anuência de seu orientador.

São Paulo, _____ de _____ de _____

Assinatura do autor: _____

Assinatura do orientador: _____

Catálogo-na-publicação

Núñez Segura, Gustavo Alonso

Energy consumption prediction in software-defined wireless sensor networks / G. A. Núñez Segura -- versão corr. -- São Paulo, 2018.
131 p.

Dissertação (Mestrado) - Escola Politécnica da Universidade de São Paulo. Departamento de Engenharia de Computação e Sistemas Digitais.

1.Redes de sensores sem fio 2.Redes definidas por software 3.Consumo de energia I.Universidade de São Paulo. Escola Politécnica. Departamento de Engenharia de Computação e Sistemas Digitais II.t.

ACKNOWLEDGEMENTS

I would first like to thank my mother Marlene and my brother Ignacio for all their support. They were always there, motivating me throughout my years of study.

I would like to thank my advisor, Cintia Borges Margi, for all her support, patience, and valuable guidance. I would also like to thank Tamires Santana for her motivation and excellent portuguese lessons.

I would also like to thank my labmates from the SEMBEI (Security and Embedded Systems Engineering group). Thank you all for your frienship and collaboration. And finally, I want to express my gratitude to Universidad de Costa Rica (UCR) and Coordenação de Aperfeiçoamento de Pessoal de Nível Superior for their support and financial aid from CAPES-PROEX # 0212083.

RESUMO

A conservação da energia é uma das principais preocupações nas Redes de Sensores Sem Fio (WSN, do inglês *Wireless Sensor Networks*). Para reduzir o consumo de energia, é importante saber como a energia é gasta e quanta energia há disponível durante o funcionamento da rede. Diversos trabalhos anteriores propuseram modelos de consumo de energia focados no módulo de comunicação, ignorando o consumo por tarefas de processamento e sensoriamento. Outros trabalhos apresentam modelos mais completos e complexos, mas carecem de experimentos que demonstrem a exatidão em dispositivos reais. O objetivo principal deste trabalho é projetar e avaliar um modelo de consumo de energia para WSN que considere o consumo por sensoriamento, processamento e comunicação. Este modelo foi utilizado para implementar dois mecanismos de previsão de consumo de energia, um deles baseado em cadeias de Markov e o outro baseado em séries temporais. As métricas para avaliar o desempenho do modelo e dos mecanismos de previsão de consumo de energia foram: exatidão da estimativa de consumo de energia, exatidão da previsão de consumo de energia e uso dos recursos de comunicação e processamento do nó. O desempenho dos mecanismos de previsão de consumo de energia foram comparados utilizando dois esquemas de implementação: rodando o algoritmo de previsão no nó sensor e rodando o algoritmo de previsão em um controlador de rede definida por software. A implementação foi conduzida utilizando IT-SDN, um arcabouço de desenvolvimento de redes de sensores sem fio definidas por software. A avaliação foi feita com simulações e emulações utilizando o simulador COOJA e ensaios com dispositivos reais utilizando o TelosB. Os resultados mostraram que considerando o consumo de energia por sensoriamento, processamento e comunicação, é possível fazer uma estimativa de consumo de energia em redes de sensores sem fio com uma boa exatidão. Ainda, o uso de um controlador de rede definida por software para processamento de algoritmos de previsão complexos pode aumentar a exatidão da previsão.

Palavras-chave: Redes de sensores sem fio. Consumo de energia. Redes definidas por software.

ABSTRACT

Energy conservation is a main concern in Wireless Sensor Networks (WSN). To reduce energy consumption it is important to know how it is spent and how much is available during the node and network operation. Several previous works have proposed energy consumption models focused on the communication module, while neglecting the processing and sensing activities. Other works presented more complex and complete models, but lacked experiments to demonstrate their accuracy in real deployments. The main objective of this work is to design and to evaluate an accurate energy consumption model for WSN, which considers the sensing, processing, and communication modules usage. This model was used to implement two energy consumption prediction mechanism. One mechanism is based in Markov chains and the other one is based in time series analysis. The metrics to evaluate the model and prediction mechanisms performance were: energy consumption estimation accuracy, energy consumption prediction accuracy, and node's communication and processing resources usage. The energy consumption prediction mechanisms performance was compared using two implementation schemes: running the prediction algorithm in the sensor node and running the prediction algorithm in a Software-Defined Networking controller. The implementation was conducted using IT-SDN, a Software-Defined Wireless Sensor Network framework. For the evaluation, simulation and emulation used COOJA, while testbed experiments used TelosB devices. Results showed that considering the sensing, processing, and communication energy consumption into the model, it is possible to obtain an accurate energy consumption estimation for Wireless Sensor Networks. Also, the use of a Software-Defined Networking controller for processing complex prediction algorithms can improve the prediction accuracy.

Keywords: Wireless sensor networks. Energy consumption. Software-Defined Networking.

LIST OF FIGURES

1	IT-SDN software architecture	27
2	PC conected with a WSN device	28
3	ContikiMAC normal operation	57
4	The proposed tool working upon ContikiMAC	58
5	Proposed tool algorithm	60
6	Toplogy configuration	61
7	Results for all scenarios	62
8	Energy consumption prediction model algorithm	65
9	Simulation's topology	66
10	Processing state energy consumption	67
11	Radio module energy consumption	68
12	Sensing module front-end	71
13	Simplified block diagram of the communication module	74
14	Domain transitions behavior	75
15	<i>Sensor node state</i> transition example	77
16	Seasonality and trending simulation topology	81
17	Node 3 energy consumption for periodical messages exchange	82
18	Node 5 energy consumption when including new nodes in the network	83
19	Node 3 energy consumption for periodical messages exchange	83
20	Node 5 energy consumption when including new nodes in the network	84
21	Controller node	86
22	Measurement setup diagram	90
23	Network deployed for performance measurements	93
24	State transitions' energy consumption percentage	95
25	Sensing domain energy consumption percentage	96
26	Markov chain MAPE for sensor node scheme and controller scheme	99
27	Markov chain prediction accuracy for 12 nodes	100
28	Markov chain prediction accuracy for 25 nodes	101

29	Markov chain prediction accuracy for 36 nodes	102
30	Energy consumption and prediction for 25 nodes using Markov chain	103
31	Energy consumption when using and not using the Markov chain prediction method	104
32	Average energy consumption when using and not using the Markov chain prediction method	105
33	Controller scheme time series algorithm	106
34	ARIMA's prediction MAPE for controller scheme and sensor node scheme	107
35	ARIMA's prediction MAPE for 12 nodes	108
36	ARIMA's prediction MAPE for 25 nodes	109
37	ARIMA's prediction MAPE for 36 nodes	110
38	ARIMA's prediction MAPE for 49 nodes	111
39	Energy consumption when using and not using the ARIMA prediction method	112
40	Prediction error of Markov chain and time series prediction mechanisms	112
41	Energy consumption of Markov chain and time series prediction mechanisms	113
42	Sensing state current consumption	125
43	Low-power mode 1 and processing current consumption	126
44	Low-power mode 3 and processing current consumption	126
45	Transmitting state current consumption	126
46	Receiving state current consumption	127

LIST OF TABLES

1	IEEE 802.15.4 frequency bands characteristics	21
2	Summary of ARMA models properties	35
3	Energy models summary	52
4	Forecasting simulation parameters	66
5	Code size on the sensor node including IT-SDN for both schemes . . .	67
6	Energy consumption balance	69
7	<i>Sensor node states</i> configuration	78
8	MSP430F1611's microcontroller operation modes	87
9	CC2420's communication module operation modes	88
10	States average current consumption	91
11	Sensing domain transitions average consumption	91
12	Energy consumption estimation accuracy	94
13	Transitions cost matrix in μJ	94
14	Markov chain states configuration for TelosB	96
15	Average increasing of sensor node resources	102
16	Average energy consumption of a real devices network running the Markov chain prediction method	103
17	Average energy consumption of a real devices network running the time series prediction method	109
18	ARIMA models tested in the controller	128

LIST OF ABBREVIATIONS AND ACRONYMS

ACF	Autocorrelation Function
APS	Applications Support
APOs	Application Objects
ACLK	Auxiliary Clock
BLE	Bluetooth Low Energy
CPU	Central Processing Unit
CTP	Collection Tree Protocol
CMOS	Complementary Metal Oxide Semiconductor
CoAP	Constrained Application Protocol
CTMC	Continuous-Time Markov Chain
CO	Crystal Oscillator
DCO	Digitally Controlled Oscillator
DTMC	Discrete-Time Markov Chain
DSN	Distributed Sensor Networks
FORM	First Order Radio Model
FFD	Full-Function Device
IETF	Internet Engineering Task Force
IPv6	Internet Protocol version 6
KB	Kilobyte
LLN	Low-Power and Lossy Networks
MCLK	Main Clock
MAPE	Mean Absolute Percentage Error
MAC	Media Access Control

MCU	Microcontroller
MEMS	Microelectromechanical systems
PHY	Physical Layer
RAM	Random-Access Memory
RDC	Radio Duty Cycle
RFD	Reduce-Function Device
ROM	Read-Only Memory
SDN	Software-defined networking
SMCLK	Sub-Main Clock
TCP	Transmission Control Protocol
USB	Universal Serial Bus
UDP	User Datagram Protocol
WSN	Wireless Sensor Networks
ZDO	Zigbee Device Object

CONTENTS

1	Introduction	13
1.1	Motivation	15
1.2	Goals	16
1.3	Methods	16
1.4	Publications and original contributions	18
1.5	Document organization	18
2	Background	20
2.1	Wireless Sensor Networks	20
2.1.1	IEEE 802.15.4	21
2.1.2	ZigBee	22
2.1.3	Other MAC and PHY Standards	23
2.1.4	Routing in WSN	24
2.1.5	Software defined networking	25
2.1.5.1	IT-SDN: Improved architecture for SDWSN	26
2.2	Prediction methods	28
2.2.1	Markov chains	28
2.2.2	Time series	31
2.2.2.1	Stationarity	31
2.2.2.2	Autocovariance and Autocorrelation	32
2.2.2.3	Autoregressive Moving Average (ARMA) Models	33
2.2.2.4	Autoregressive Integrated Moving Average (ARIMA) Models	34

2.2.2.5	Model Identification	34
2.3	Summary	38
3	Literature Review	39
3.1	WSN energy consumption models	39
3.1.1	Sensing	40
3.1.2	Processing	41
3.1.3	Communication	44
3.1.4	Global analysis	51
3.2	Energy consumption prediction in WSN	51
3.2.1	Markov chain approaches	52
3.2.2	Time series	53
3.2.3	Summary	54
4	Preliminary Results	56
4.1	Duty Cycle Based Energy Management Tool for Wireless Sensor Networks	56
4.1.1	The tool proposed	57
4.1.2	Implementation and simulation	58
4.1.3	Results	60
4.2	Energy Map Model for Software-Defined Wireless Sensor Networks .	63
4.2.1	Energy consumption prediction model	63
4.2.2	Implementation and simulation	64
4.2.3	Results	66
4.3	Summary	69
5	Energy consumption and prediction model	70
5.1	Energy consumption model proposal	70

5.1.1	Sensing domain	71
5.1.2	Processing domain	72
5.1.3	Communication domain	73
5.1.4	Transitions between domains	75
5.1.5	Model considerations	76
5.2	Energy consumption prediction with Markov chain	76
5.3	Energy consumption prediction with time series	80
6	Proposal Evaluation	85
6.1	General aspects	85
6.2	Energy consumption model	86
6.2.1	TelosB modeling	87
6.2.2	Implementation	89
6.3	Markov chain	95
6.3.1	Markov chain schemes	97
6.4	Time series	104
6.4.1	Time series experiments	106
6.5	Summary	110
7	Conclusions	114
7.1	Future work	115
	References	116
	Appendix A – TelosB energy consumption measurements	125
	Appendix B – ARIMA models	128

1 INTRODUCTION

Wireless sensor networks (WSN) are formed by interconnected nodes with the capacity of sensing, processing, and communicating. Advances in wireless communication, digital electronics, and the popularity gained by microelectromechanical systems (MEMS) in the past two decades, have helped to enhance its popularity and become a technology exploited in a vast number of applications nowadays, such as: environmental, industrial and health monitoring; tracking and detection in military battlefields; home automation; security; and also a key technology in the development of the Internet of Things (CULLER; ESTRIN; SRIVASTAVA, 2004; MAINETTI; PATRONO; VILEI, 2011).

This kind of networks are characterized by the resources constraints on the sensor nodes. They normally have: low processing power, low storage capacity, limited bandwidth, and the main energy source is batteries. Those restrictions are important to reduce the cost of the hardware and reduce the energy consumption, but also they become a challenge for the scientific and industrial community involved. For example, the AS-XM1000 is an IEEE 802.15.4 mote based on the popular TelosB (MEMSIC Inc., 2003). It has a 16-bit microcontroller of 8 MHz, 8 kB of random-access memory (RAM), 116 kB of program flash memory and uses two AA batteries (ADVANTIC-SYS, 2012). Another example is the IRIS mote, an IEEE 802.15.4 mote that has an 8-bit microcontroller able to work at 16 MHz, 128 kB of flash memory, 8 kB of internal SRAM and uses two AA batteries as main energy source (MEMSIC, 2014). Thus, as several WSN applications are deployed in hostile environments, where batteries replacement or recharge could be impractical, WSN are primarily focused on power conservation instead of quality of service (AKYILDIZ et al., 2002).

These challenges, mainly caused by the energy restrictions, open a wide research area that aims on improving the network lifetime. There are many network lifetime definitions that can be found in the literature (YETGIN et al., 2017), but it can be un-

derstood as the total time during which the network still has its full functionality. Some of the network lifetime definitions are focused on the individual node lifetime (CHEN; ZHAO, 2005), others are focused on the coverage and connectivity of a certain area (CARDEI et al., 2005), or even transmission metrics can be used to define when the network lifetime is over (BAYDERE; SAFKAN; DURMAZ, 2005; LI; ALREGID, 2009). Thus, different techniques have been developed to maximize the network lifetime. The objective of each of them depends on the network lifetime definition that was considered. Some examples of those techniques are based on resource allocation, sleep-wake duty cycle scheduling, routing and clustering, coverage and connectivity, and mobile sinks.

The study of the network lifetime is strongly linked with the study of the accurate estimation of residual energy in the node, but also with its behavior over time (MINI et al., 2005; WANG; YANG, 2007). To obtain an accurate estimation, it is necessary to employ a proper energy consumption model and to understand the behavior of the main energy source. In the case of the consumption model, there is the difficulty in modeling the radio communication, specially due the probabilistic nature of the wireless communications (JALSAN; FLOURI; FELTRIN, 2014); and in the case of batteries as main energy source, its lifetime depends on the usage pattern (RAO; VRUDHULA; RAKHMATOV, 2003). Those factors and their complexity, plus the fact that this process has to be continuously running on the node, become a major problem given the computational capabilities and limited energy in WSN devices. In this manner, mechanisms to improve the WSN's energy efficiency and the computational capabilities, can help to implement more complex energy consumption models to enhance the energy consumption estimation accuracy.

Kobo, Abu-mahfouz and Hancke (2017) assert that to solve WSN's inherent problems, including energy and computational constraints, it is necessary an holistic solution instead of isolated solutions. Recent works propose the integration of WSN with Software Defined Networking (SDN) as a potential solution to several WSN challenges. The main idea of SDN is to separate the control plane from the data plane. The control logic of the network is logically centralized in what is called: SDN controller. The network devices (switches and routers) become forwarding devices (MCKEOWN et al., 2008). Those characteristics have motivated different authors to study the benefits of integrating WSN and SDN, and there are already proposals on the literature (LUO; TAN; QUEK, 2012; TREVIZAN; BATISTA; BORGES, 2015; GALLUCCIO

et al., 2015; ALVES et al., 2017).

The integration of WSN and SDN is known as Software Defined Wireless Sensor Networks (SDWSN). In this case, there is an SDN controller, which is a node in the network in charge of the WSN control plane, and the sensor nodes are sensing and forwarding devices (LUO; TAN; QUEK, 2012). We believe this control plane and data plane separation can be useful to alleviate the processing overhead in WSN, bringing more processing resources to use them for energy consumption estimation.

This work comprises the study of energy consumption models and energy consumption prediction in a WSN. It also involves the study and implementation of those concepts applied to SDWSN.

1.1 Motivation

Energy efficiency has been a major concern in the research of WSN. This concern has prompted the development of different approaches and techniques, aiming to maximize the network lifetime.

An important information to maximize the network lifetime is the knowledge of the residual energy in the sensor node. Najimi et al. (2014) and Yetgin et al. (2017) propose techniques to balance the energy consumption among all the sensor nodes to improve the network lifetime. In this manner, the accurate estimation of the residual energy is relevant because a considerable error in the estimation could affect a decision in the algorithm or gives wrong information about the state of the node. For example, the estimation algorithm could indicate that a node has ran out of energy when it is still alive, or otherwise.

The estimation of the residual energy and the transmission of this information to a central node have an energy cost that is important to take into account. Previous works address this problem implementing prediction algorithms, reducing the use of the communication module but increasing the overhead in processing (MINI et al., 2005). For this reason, to execute a prediction algorithm in a controller node, which does not have energy and processing power constraints, increase its relevance. In an SDN scheme, where the controller has a global view of the network, this information could be an interesting parameter for resource allocation or dynamic routing algorithms (YETGIN et al., 2017).

1.2 Goals

The main goal of this work is to specify and evaluate the performance of an energy consumption model for WSN, which allows a node, or a controller, to estimate its remaining energy. A specific goal will be the implementation of this model into an SDWSN. With that, it will be possible to collect the energy information of the whole network in an SDN controller, leveraging from its processing power and its energy independence, and easing the access to this information to other control functions. Also, the implementation in an SDWSN will enable the comparison of the performance of implementing the model on each node and implementing it in a central controller. The accuracy of the energy consumption estimation, the accuracy of the prediction of the energy consumption, and the nodes processing and communication resources usage are key metrics in this work.

1.3 Methods

The first step to elaborate this model was the detailed survey of the literature about energy consumption in WSN. From this revision, it was identified the main elements to take into account for the estimation.

Two forecasting mechanisms were studied with the objective of evaluating their performance for the energy consumption prediction in a Software-Defined Wireless Sensor Network. The first one was a statistical approach that forecasts the energy consumption using time series analysis. The WSN energy consumption behavior was characterized to select the best suited time series model. The second prediction mechanism was Markov chain. The Markov chain prediction mechanism is based on states transitions probability. To implement the Markov chain prediction algorithm, the WSN node energy consumption was expressed in consumption states, based on the hardware characteristics and operation system behavior.

The model was implemented into a WSN application using the operating system Contiki (DUNKELS; GRONVALL; VOIGT, 2004) and the SDWSN implementation IT-SDN (ALVES et al., 2017). For the simulation and emulation the software used was COOJA (ÖSTERLIND et al., 2006). This software allows to emulate different commercial nodes and has a graphical interface that allows to monitor the network in real time and eases to obtain metrics. For the real implementation, experiments were

carried out using the TelosB mote (MEMSIC Inc., 2003).

Two different schemes were employed to test the model and to measure its performance. In the first scheme, most of the algorithm implemented was processed on the node. In the second scheme most of the algorithm was processed in the SDN controller. The metrics used to evaluate the model and predictions mechanism performance were: energy consumption estimation accuracy, the energy consumption prediction accuracy, and the the nodes processing and communication resources usage due to the implementation of the model. To evaluate the energy consumption estimation accuracy, the results were compared with real devices consumption, measured with external devices (multimeter). And finally, prediction mechanisms' energy consumption performance was compared, for both schemes, running the same WSN application. The objective of this comparison was to identify which scheme was the best option in order to extend the network lifetime.

1.4 Publications and original contributions

The two following works were conducted in collaboration with the advisor. Both works show the preliminary results of the main research.

- *Duty Cycle Based Energy Management Tool for Wireless Sensor Networks*: This work was published in the XXXIV Brazilian Communications and Signal Processing Symposium, 2016. It was written by Gustavo A. Núñez and Cintia B. Margi. This work propose an energy management tool which allows to control the node communication module duty cycle. The main objective of this tool is to manage the communication module usage in order to increase the network lifetime.
- *Energy Map Model for Software-Defined Wireless Sensor Networks*: This work was published in the XXXV Brazilian Communications and Signal Processing Symposium, 2017. It was written by Gustavo A. Núñez and Cintia B. Margi. This work presents a method to construct an energy map into a SDWSN. It is inspired in the Markov chain approach proposed by Mini et al. (2005).

The next work was a collaboration with other postgraduate students and the advisor. This work is not part of preliminary results but it was important to carry out the research implementation.

- *ITSDN: Improved architecture for SDWSN*: A demo session work presented in the XXXV Brazilian Symposium on Computer Networks and Distributed Systems, 2017. It was written by Renan C. A. Alves, Doriedson A. G. Oliveira, Gustavo A. Núñez and Cintia B. Margi. This work describes IT-SDN, an open SDWSN tool which aims to address many of the problems in previous SDWSN approaches. It was inspired by TinySDN (TREVIZAN; BATISTA; BORGES, 2015) but improving the architecture, protocols and implementation.

1.5 Document organization

The remainder of this work is organized as follows. Chapter 2 presents an introduction to WSN, SDWSN, and prediction methods. In Chapter 3 it is analyzed the

related work. Chapter 4 presents original contributions and preliminary results. Then, Chapter 5 presents the energy consumption and prediction model design. Chapter 6 details the implementation, experiments, and results of the energy consumption and prediction models. Chapter 7 concludes the document with final considerations and future work.

2 BACKGROUND

This chapter presents a brief overview of WSN concepts and the evolution of this technology over the time. Then, the concept of SDWSN is presented, and the SDWSN framework IT-SDN is introduced. Finally, there is an overview of the mathematics used for the Markov chain and time series modelling

2.1 Wireless Sensor Networks

Wireless Sensor Networks (WSN) are typically formed by a large number of nodes with the capacity of sensing, processing, and communicating. Such networks have specific characteristics and constraints that differentiates them from other wireless networks, such as cellular systems or mobile ad hoc networks (MANET). WSN are usually application specific, are densely deployed into hostile environments, can suffer topology changes due to node failure, and also have several energy, processing, and storage constraints.

First WSN implementations are registered from the Cold War era. In the early fifties, an undersea surveillance system was deployed at the bottom of Eleuthera with the objective to detect and track submarines (URICK, 1975). Around the eighties, the WSN development continued growing with the creation of the Distributed Sensor Networks (DSN) program at the Defense Advanced Research Projects Agency (DARPA), but the absence of technology to develop small sensors was a drawback (CHONG; KUMAR, 2003). Following advances in computing, communication, and microelectromechanical systems have empowered the WSN evolution, helping to solve some of the principal issues that faced the first implementations and broaden the range of possible applications.

In our days, WSN technology embraces multiple areas. For example, in the military industry for tracking and detecting functions in hostile environments, environ-

mental monitoring, health applications such as patient monitoring and diagnostics, industrial applications such as preventive maintenance and structural health monitoring. Also it is an important technology on the development of the Internet of Things (IoT) (AKYILDIZ; VURAN, 2010).

The broad range of applications, the network characteristics, and hardware resources constraints, establish a number of requirements and objectives when designing sensor networks. The low power consumption is a mandatory requirement in most applications. Hostile environment deployments increase the complexity in the network maintenance, being necessary to extend the network lifetime as much as possible. Those deployments also expose sensor nodes to different circumstances that can affect their integrity and fault tolerance, what becomes an important design requirement. The large number and density of nodes in WSN deployments helps to counteract its distance communication constraint and gives redundancy in case of a node fault. In this case, the network adaptability and scalability are important requirements. If a node have operation problems, or new nodes enter the network, the protocols used should be capable to adapt the network to its new conditions.

2.1.1 IEEE 802.15.4

In 2003, it was published the standard IEEE 802.15.4 (IEEE, 2003). It specifies the physical layer (PHY) and media access control (MAC) sublayer for Low-Rate Wireless Personal Area Networks (LR-WPAN). The specification is focused on low cost and flexible communication, obtaining low energy consumption, but also low data transfer rate.

The first version of IEEE 802.15.4 defines three licence-free frequency bands for the physical layer: 868 MHz, 915 MHz and 2450 MHz. Table 1 summarizes their main characteristics. Later revisions added new frequency bands for China and Japan.

Table 1: IEEE 802.15.4 frequency bands characteristics

Frequency band (MHz)	Channels	Nominal data rate (kbit/s)	Access mode	Modulation
868	1	20	DSSS	BPSK
915	10	40	DSSS	BPSK
2450	250	16	DSSS	O-QPSK

Adapted from IEEE (2003)

The PHY provides a data service and a management service. The PHY main functionalities are the radio activation and deactivation, packets transmission and reception, energy detection measurement, channel selection, link quality estimation, and clear channel assessment. The MAC sublayer provides the MAC data service and MAC management service. Its main functionalities are channel access, beacon management, frame validation, acknowledged frame delivery, guaranteed time slot management, association and dissociation (IEEE Computer Society, 2011).

The standard defines two operation topologies and two different devices that can participate in the network. The full-function device (FFD) has characteristics that enables it to be network coordinator. The reduce-function device (RFD) is intended for simple applications and is not capable to work as coordinator. The topologies defined in the standard are: star and peer-to-peer. In the star topology there is a personal area network (PAN) coordinator and all nodes around establish communication with it. In the peer-to-peer topology each device can communicate with any other device if they are in range of one another. This topology have also a PAN coordinator (IEEE Computer Society, 2011).

2.1.2 ZigBee

ZigBee Alliance is a non-profit consortium of semiconductor manufacturers established in 2002¹. They created the Zigbee standard and they are in charge of maintaining and updating it.

The ZigBee standard is defined in conjunction with the IEEE 802.15.4 standard. The IEEE 802.15.4 standard defines the PHY and MAC layers and ZigBee defines the network layer and the application framework (ZigBee Alliance, 2012). The ZigBee network layer plays management functionalities. It is in charge of establishing a new network and the process to allow or drop out a device's network membership (ZigBee Alliance, 2012). ZigBee supports mesh and tree topologies, besides the ones mentioned in the IEEE 802.15.4. ZigBee defines three types of devices: the end device, the router, and the coordinator. The end device can be a RFD or a FFD, because is intended for simple tasks. The router is a FFD with routing capabilities and the coordinator is a FFD that manages the whole network (ZHENG; JAMALIPOUR, 2009).

The application layer defines the framework used to develop applications. It is

¹<http://www.zigbee.org/>

composed by the application support (APS) sublayer, the ZigBee device object (ZDO), and the manufacturer-defined application objects (APOs). The APS provides the services required for APOs and ZDO to communicate them with the network layer. The APOs define inputs and outputs to the APS, for example, a switch can be an input and the light the switch controls is the output. The ZDO has control and management functions as APS, network layer, and security provider initialization. Also it controls the network layer indicating when to form, to join, or to leave a network.

2.1.3 Other MAC and PHY Standards

WirelessHART (CHEN; NIXON; MOK, 2010) is an open standard based on the HART protocol (BOWDEN, 1999), intended for processes measurements and control applications. Its PHY characteristics are defined by the IEEE 802.15.4 using the 2,4 GHz band and it employs a time domain multiple access (TDMA) MAC protocol (AKYILDIZ; VURAN, 2010). In terms of networking, all WirelessHART devices have routing capability and use two different routing protocols: graph routing and source routing (LENNVALL; SVENSSON; HEKLAND, 2008). The transport layer provides end-to-end reliability communication and the application layer use the HART (BOWDEN, 1999) existing solution.

Similar as WirelessHART, ISA100.11A was designed for industrial and control applications. The PHY uses IEEE 802.15.4 and the data link layer implements frequency hopping and TDMA. The network layer is compatible with the Internet Engineering Task Force (IETF) 6LoWPAN standard (MONTENEGRO et al., 2007), and the transport layer supports a connectionless service based on User Datagram Protocol (UDP). The application layer defines objects and the services to communicate those objects among distributed applications. The specifications does not define the operation of the distributed applications (International Society for Automation, 2009).

Bluetooth Low Energy (BLE) is an energy aware, low cost, and low complexity Bluetooth version. BLE was designed for control and monitoring applications, and launched in the Bluetooth 4.0 specification (BLUETOOTH SPECIAL INTEREST GROUP, 2010).

BLE has been proposed as a WSN solution, mainly for its low energy consumption. Siekkinen et al. (2012) and Mikhaylov, Plevritakis and Tervonen (2013) compared BLE with IEEE 802.15.4, and both concludes that BLE has better energy con-

sumption efficiency, but also they conclude that its network characteristics are not adequate for WSN. One of those characteristics is the topologies supported because, as mentioned by BLUETOOTH SPECIAL INTEREST GROUP (2014) the BLE standard can just operate as a star topology or broadcasting (BLUETOOTH SPECIAL INTEREST GROUP, 2014), and this is a limitation for WSN and IoT. To offset this restriction, some works propose mechanisms to obtain mesh topologies. Hortelano et al. (2017) propose a packet format and a mesh topology with two configurations: individual mesh and collaborative mesh. In the collaborative mesh, sensor devices transmit their data packets and also retransmit data packets from other devices to reach its destiny. In the individual mesh, sensor devices do not retransmit data packets. Patti, Leonardi and Bello (2016) proposed a mesh network formed by multiple networks (called sub-networks). Each sub-network share one or more nodes with another sub-network. Those nodes shared can be masters or slaves and the configuration has to follow the next two rules. First, a slave node can establish communications with no more than two masters. Second, a master node can establish communication with at most another master. In that case, one of the masters have to play a slave role.

On July 2017, Bluetooth SIG announced the mesh networking capability which emerges as a solution for the limitations of BLE in WSN and IoT mentioned before.

2.1.4 Routing in WSN

Routing is an important issue in WSN and it is challenging due to several network constraints. These constraints are not only by individual sensor node characteristics. The behavior of the network, the nature of sensor fields, and the metric requirements of a sensing application also have influence in the routing design (GOYAL; TRIPATHY, 2012).

Akkaya and Younis (2005) classify WSN routing protocols considering the following characteristics: data-centric, hierarchical, location-based, QoS, network-flow, and data aggregation. Data-centric protocols aims to reduce the data redundancy. To achieve this, data-centric routing protocols are able to select a set of sensor nodes and use data aggregation during the relaying of data. SPIN (HEINZELMAN; KULIK; BALAKRISHNAN, 1999) is the first data-centric protocol, which considers data negotiation between nodes. Hierarchical routing protocols are meant to efficiently maintain the energy consumption in a clustering organized network, performing data aggrega-

tion. LEACH (HEINZELMAN; CHANDRAKASAN; BALAKRISHNAN, 2000) is a hierarchical routing protocol that forms clusters based on the received signal strength. Location information is worth for routing protocols. It could be used to estimate the distance between two nodes and to estimate the energy consumption. Also, using location information, queries can be diffused to specific regions. GAF (XU; HEIDEMANN; ESTRIN, 2001) is an example of energy-aware location-based routing algorithm. This algorithm helps to conserve energy by turning off nodes in the network without affecting the routing fidelity. QoS-aware protocols set up the path considering end-to-end delay requirements. SAR (SOHRABI et al., 2000) is a WSN protocol that considers QoS in its routing decisions. This protocol creates trees considering QoS metric, energy resource and packet's priority level. Those trees are rooted at one-hop neighbors of the sink (SOHRABI et al., 2000). The use of network-flow to set up route paths has also been researched, as well Chang and Tassiulas (2004) define link costs as a function of node residual energy and link transmission cost. The main objective is to maximize the network lifetime.

More recent routing protocols are Collection Tree Protocol (CTP) (GNAWALI et al., 2009) and RPL (WINTER, 2012). CTP is a tree-based collection protocol, which builds and maintains minimum-cost trees to tree roots. Also, CTP is address-free, which means that nodes do not send packets to a particular root; but it is implicitly chosen when the node chose a next hop. The metric to establish the next hop and routes is expected transmissions (ETX).

RPL (WINTER, 2012) is an IPv6 routing protocol for *Low Power and Lossy Networks* (LLN) defined in RFC6550. RPL specifies how to build a Destination Oriented Directed Acyclic Graph (DODAG) using an objective function and a set of metrics and constraints. DODAG is a set of vertices that are connected by directed edges with no cycles. There could be more than one DODAG and each one is optimized according different QoS requirements (WINTER, 2012). Another important characteristic is that RPL supports three traffic patterns: multipoint-to-point, point-to-multipoint, and point-to-point.

2.1.5 Software defined networking

Software Defined Networking (SDN) is a paradigm that separates the control plane from the data plane. In order to accomplish this separation, it logically centralizes the

control logic decisions in one or more programmable nodes. This node, or nodes, are known as SDN controller. Then, the network devices become forwarding devices, which means they do not interfere in the control decisions. This scheme allows the controller to have a global view of the network, giving the opportunity to adequate the routing algorithms (KREUTZ et al., 2015). McKeown et al. (2008) proposed OpenFlow, the first southbound protocol to establish communication between an SDN controller and forwarding devices, focusing on wired networks.

Applying the SDN paradigm into WSN is known as Software Defined Wireless Sensor Networks (SDWSN), and it has been presented as an efficient solution for different issues of the latter. Energy consumption, network management, mobility, interoperability and security are among the fields where SDWSN is being studied (KOBO; ABU-MAHFOUZ; HANCKE, 2017). There are already several proposals in the literature, but they have some issues. Sensor OpenFlow (LUO; TAN; QUEK, 2012) have issues concerning frame sizes. SDN-Wise (GALLUCCIO et al., 2015) implementation provided is not complete and TinySDN (TREVIZAN; BATISTA; BORGES, 2015) is highly dependable of the operating system. IT-SDN (ALVES et al., 2017) is another SDWSN proposal and it is the more recent of works mentioned before. IT-SDN is an SDWSN framework inspired by TinySDN (TREVIZAN; BATISTA; BORGES, 2015) but improving the architecture, protocols, and implementation.

IT-SDN was the framework used to implement the research model. It was chosen because its code is totally open and available, and it addresses some of the issues in the other SDWSN frameworks. IT-SDN main characteristics are detailed next.

2.1.5.1 IT-SDN: Improved architecture for SDWSN

IT-SDN (ALVES et al., 2017), is an open SDWSN tool which aims to address many of the problems in previous SDWSN approaches mentioned in Section 2.1.5. The IT-SDN architecture has three main components: southbound protocol, neighbor discovery protocol, and controller discovery protocol. The southbound protocol defines the communication procedure between controller and SDN-enabled devices. The neighbor discovery protocol obtains and maintains the neighborhood information, and the controller discovery protocol identifies a next hop candidate to reach the controller.

The architecture was instantiated into a compliant software architecture targeting Contiki. Contiki is an operating system designed for constrained devices. Contiki's

kernel is event-driven, but its system supports multi-threading. Also, Contiki is implemented in the C language and it can be used in different microcontriller architectures, such as MSP430 and Atmel AVR.

Figure 1 shows the IT-SDN software architecture. The southbound is covered by the *core*, which coordinates all other modules, and the *process packet* module. Flow tables, neighbor table, RX queue, and TX queue are four auxiliary components. The flow tables store routing information set by the controller. The neighbor table brings a common data structure for the neighbor discovery protocol, which is used to read and to transmit neighbor information. The RX and TX queues are buffers to avoid dropping packets in case of network congestion or long delays in packets processing.

Figure 1: IT-SDN software architecture

Source: Alves et al. (2017)

The SDWSN protocol was implemented as a Contiki network driver, using C language, and it could replace the existing layer 3 protocols available in the operating system. Also, it is made compatible with the Rime stack, which allowed to use the Contiki Collect protocol implementation as neighbor and controller discovery protocol. Also, the Contiki Link Layer Security (LLSEC) stack, ContikiSec (CASADO; TSIGAS, 2009), is used as a decision point to forward packets to SDN or Rime stack. ContikiSec provides confidentiality, authentication, and integrity. Its design is based in existing primitives, but it tries to balance security with low energy consumption.

The controller station is formed by a computer and a Contiki node. The last one works as a bridge between the wireless network and the software in the PC. This com-

munication is carried out through a serial connection, using TCP for COOJA simulations and USB connection when working with real devices (Figure 2). The PC software constructs a graph according to the neighbor report and uses it to calculate flows, when node request, and to update flow tables. It was implemented using the Dijkstra's shortest path algorithm.

Figure 2: PC conected with a WSN device

Source: Author

This SDWSN framework was tested both in COOJA simulator and with real devices.

2.2 Prediction methods

This section covers the main concepts of Markov chains and time series, which are then used in Chapter 5 to propose two prediction methods for energy consumption in WSN.

2.2.1 Markov chains

Markov chains are a class of stochastic processes with different variants: discrete-time Markov chains (DTMC), continuous-time Markov chains (CTMC), finite and countably infinite states (MODICA; POGGIOLINI, 2012). The DTMC with finite states were chosen to develop this work and in this section their main properties are explained.

Before defining a DTMC, some concepts need to be defined. A stochastic process is a family of random variables X_t , and each random variable is indexed by paratemer

$t \in T$ where $T = 0, 1, 2, \dots$ and is usually called the *time-parameter*. The possible values of X_t is known as the state-space S and its elements are called *states*. The Markov property establishes that the conditional probability of X_{t+1} depends only on the previous value X_t . The Markov property is shown in Equation 2.1.

$$P(X_{t+1} = i_{t+1} | X_t = i_t, \dots, X_{k+1} = i_{k+1}, X_k = i_k) = P(X_{t+1} = i_{t+1} | X_t = i_t) \quad (2.1)$$

In this manner, DTMC is a stochastic process which satisfies the Markov property (MODICA; POGGIOLINI, 2012), restricted to a discrete state-space S and a discrete-parameter state T (MODICA; POGGIOLINI, 2012). In a DTMC, each element of T is called time-step.

The Markov chains can be time-homogeneous and time-inhomogeneous. A Markov chain is time-homogeneous if the transition probabilities do not depend on time, otherwise, it is considered a time-inhomogeneous Markov chain (LEVIN; PERES; WILMER, 2009). The time-homogeneous property is shown in Equation 2.2.

$$P_{ij}^1 = P_{ij}^1(n) = P(X_{n+1} = j | X_n = i) = P(X_1 = j | X_0 = i), \forall n \in T \quad (2.2)$$

For the remain of this section, it is considered only time-homogeneous discrete-time Markov chains.

The dynamics of a Markov chain can be represented by a probability matrix, $\mathbf{P}$, or by a transition diagram. The probability matrix dimension depends on the amount of elements, or states, in the state-space. In the case the state-space has k elements, the probability matrix size will be $k \times k$, as shown next.

$$\mathbf{P} = \begin{pmatrix} p_{11} & p_{12} & \cdots & p_{1k} \\ p_{21} & p_{22} & \cdots & p_{2k} \\ \vdots & \vdots & \ddots & \vdots \\ p_{k1} & p_{k2} & \cdots & p_{kk} \end{pmatrix}$$

The matrix's elements are the probabilities distribution of going from one state to another for one step. For example, the element p_{12} is the probability the next state will be the number '2' if the current state is the number '1'. Also, as a stochastic process,

it has to satisfy the conditions: $P_{ij} \geq 0$, $\sum_{j=1}^N P_{ij} = 1 \forall i, j$.

Having the one-step probabilities distribution together with the total probability theorem and the Markov property, allows to compute the probability distribution for n time-steps (BOLCH et al., 1998). Assuming X_0 is the initial state-space distribution and X_1 is the state-space distribution in the next time-step, thus $X_1 = X_0 P$. Then, $X_2 = X_1 P = X_0 P P$. Repeating this process n times, the relation obtained is shown in Equation 2.3.

$$X_n = X_0 P^n \quad (2.3)$$

In the case the distribution X_n is the same for all n , it is known as steady-state or stationary distribution. Thus, this vector X_n satisfies the Equation 2.4

$$X_n = X_n P \quad (2.4)$$

The states in a Markov chain can be classified as recurrent state or transient state. If a Markov chain starts in a certain state and the probability of eventually return to this state is 1, the state is recurrent. Otherwise, the state is transient (TOLVER, 2016). To calculate the number of visits to state j when the markov chain starts in the state i , in the next n time-steps, is shown in Equation 2.5:

$$N_j = \sum_{t=1}^n P_{ij}^t \quad (2.5)$$

In the same way, to calculate the number of visits to all states, Equation 2.5 can be expressed in a general manner as shown in Equation 2.6.

$$N = \sum_{t=1}^n P^t \quad (2.6)$$

The DTMC characteristics and properties explained before, are important to understand next chapters. In Chapter 5, it is explained how DTMC is used to devise a prediction method for energy consumption in WSN. Then, it is implemented in a SDWSN.

2.2.2 Time series

A time series is a collection, or sequence, of observations x_t indexed by time. Those observations might be made in a discrete domain, which are called discrete-time time series, or in a continuous domain, which are called continuous-time time series (BROCKWELL; DAVIS, 2002).

Time series analysis constitute a set of methods for analyzing time series, and its main objective is to develop mathematical models to describe the sample data (SHUMWAY; STOFFER, 2006). Time series analysis is currently used in different areas, such as economy, social behaviors, and health problems. For example, a time series of the gross national product of a country, the amount of births in a community, or the cases of a certain disease observed in a time period.

There are two different analysis approaches for time series: time-domain approach and frequency-domain approach (SHUMWAY; STOFFER, 2006). The time-domain approach focused in how does the current variable behavior will affect the future behavior. The frequency-domain approach study the cycles of the variable behavior. Both analysis are not necessarily mutually exclusive. In this section the main characteristics and models of discrete-time time series and time domain approach analysis are explained.

2.2.2.1 Stationarity

Stationarity is the basis of time series analysis because most of statistical forecasting methods are based on this property. On the other hand, nonstationary time series are very common in many applications, for example, bussiness, economics or industrial processes without control actions to ensure stationarity.

The time series stationarity can be classified as weak and strict. Brockwell and Davis (2002) defines both stationarities as follows. A time series $X_t, t = 0, \pm 1, \dots$ is weakly stationary if the vectors $(X_1, \dots, X_n)$ and $(X_{1+h}, \dots, X_{n+h})$ have the same mean vectors and covariance matrices for all integers h and $n > 0$. If in the same h and n , those two vectors have the same joint probability distribution, X_t is strictly stationary.

2.2.2.2 Autocovariance and Autocorrelation

Before defining autocovariance and autocorrelation, it is important to define covariance and correlation. Covariance is the measure of the linear relationship between two random variables. Correlation is a normalized covariance. Assuming two random variables X and Y the covariance and correlation between them are expressed in Equations 2.7 and 2.8 respectively, where $E(X)$ and $E(Y)$ are the means of the probability distribution, and σ_X^2 and σ_Y^2 are the variances of X and Y (BISGAARD; KULAHCI, 2011). The mean of the probability distribution is also called the *expected value* of the distribution (BRASE; BRASE, 2016).

$$Cov(X, Y) = E[(X - E(X))(Y - (E(Y)))] \quad (2.7)$$

$$Corr(X, Y) = \frac{Cov(X, Y)}{\sqrt{\sigma_X^2} \sqrt{\sigma_Y^2}} \quad (2.8)$$

To calculate the covariance and correlation for a single time series, it is necessary to form *sub-series* and to treat each one as a different variable. That means, if a time series has 10 observations, the first variable z_t could be a *sub-series* taking from the first to the ninth value, and the second variable z_{t+1} a *sub-series* taking from the second to the tenth value. Thus, as the covariance and correlation calculation is on the same data set, those are also called autocovariance and autocorrelation (BISGAARD; KULAHCI, 2011).

As mentioned before, the covariance in a stationary time series does not change over time. Thus, the autocovariance can be defined as a function of the time lag. For a time lag k and a mean value μ , the autocovariance $\gamma(k)$ can be expressed as Equation 2.9:

$$\gamma(k) = E[(z_{t+k} - \mu)(z_t - \mu)] \quad (2.9)$$

Then, defining the time series variance as $\gamma(0)$, the autocorrelation function (ACF),

in dependence of k , can be expressed as showing next:

$$\rho(k) = \frac{\gamma(k)}{\sqrt{\gamma(0)}\sqrt{\gamma(0)}} = \frac{\gamma(k)}{\gamma(0)} \quad (2.10)$$

The ACF is imperative in the time series model identification because it allows to express the correlation of observations, that are k lags apart, in a function of k (BISGAARD; KULAHCI, 2011). On the other hand, for some time series models it is necessary the information of the correlation considering the effect of the intermediate values. To obtain this information it is necessary to use partial autocorrelation (PACF) (SHUMWAY; STOFFER, 2006). The PACF ϕ_{hh} for a stationary process x_t is shown in Equation 2.11

$$\phi_{hh} = \text{Corr}(x_{t+h} - \hat{x}_{t+h}, x_t - \hat{x}_t), \quad h \geq 2 \quad (2.11)$$

In Equation 2.11 $\hat{x}_{t+h}$ and $\hat{x}_t$ denote the regression of x_{t+h} on $x_{t+h-1}, x_{t+h-2}, \dots, x_{t+1}$, and x_t on $x_{t+1}, x_{t+2}, \dots, x_{t+h-1}$ on respectively (SHUMWAY; STOFFER, 2006).

2.2.2.3 Autoregressive Moving Average (ARMA) Models

The autoregressive moving average (ARMA) model for stationary time series, is formed by the combination of an autoregressive (AR) model and a moving average (MA) model. The AR model models the linear dependency of current observations with lagged observations. The MA model relates the current observations to past and present noise terms. Denoting the stationary time series w_t , an $\text{AR}(p)$ model is shown in Equation 2.12 and a $\text{MA}(q)$ model is shown in Equation 2.13 (SHUMWAY; STOFFER, 2006).

$$w_t = \phi_1 w_{t-1} + \phi_2 w_{t-2} + \dots + \phi_p w_{t-p} + a_t \quad (2.12)$$

$$w_t = a_t - \theta_1 a_{t-1} - \theta_2 a_{t-2} - \dots - \theta_q a_{t-q} \quad (2.13)$$

Combining both models, the result is an $\text{ARMA}(p, q)$ model, as shown in Equation

2.14.

$$w_t = \phi_1 w_{t-1} + \phi_2 w_{t-2} + \dots + \phi_p w_{t-p} + a_t - \theta_1 a_{t-1} - \theta_2 a_{t-2} - \dots - \theta_q a_{t-q} \quad (2.14)$$

2.2.2.4 Autoregressive Integrated Moving Average (ARIMA) Models

The ARMA models explained in Section 2.2.2.3 applies only to stationary time series. Since stationarity is not natural (BISGAARD; KULAHCI, 2011) it is common to find a nonstationary behavior in processes without control system. One way to model nonsationary time series, is by stationarizing them and then using the ARMA models. There are different methods to stationarize a time series, such as: square roots transformation, de-trending, and differencing.

Differencing data is basically to compute the differences between consecutive observations as shown in Equation 2.15. After differencing the data and verifying its stationarity, it is possible to use the ARMA models to forecast the time series. This process is called autoregressive integrated moving average, ARIMA(p, d, q). Same as the ARMA model, ARIMA has the parameters p and q , but also includes the parameter d which indicates the times the series need to be differenced to reach stationarity, as shown in Equation 2.16 (BISGAARD; KULAHCI, 2011).

$$w_t = \nabla z_t = z_t - z_{t-1} \quad (2.15)$$

$$w_t = \nabla_d z_t \quad (2.16)$$

2.2.2.5 Model Identification

When talking about time series model identification, it is common to use the term “adequate model”, keeping the others as possible solutions too. Related to this, Bisgaard and Kulahci (2011) state that: “The models may be “useful” and “adequate”, but it would be inappropriate to “assume” that they are “true” or that there is anything like a “correct” model for a given time series”.

Box et al. (2008) proposed an iterative method to aid in the time series adequate model identification. This method was adapted to ARIMA models by Bisgaard and

Kulahci (2011), following the next steps:

1. consider a general ARIMA model;
2. identify the appropriate degree of differencing;
3. find a tentative model using ACF and PACF;
4. estimate the parameterers of the model (software);
5. perform de the residual analysis;
6. verify if the chosen model was the adequate, if it was, start forecasting, if not, go back to step 2.

Following the step 1, when considering ARIMA models, it is necessary to estimate p , q and d . The step 2 is related with d because this parameter indicates how many times the time series need to be differenced to be stationary. Thus, if the time series becomes stationary after differenced it one time, the model will be $ARIMA(p, 1, q)$. Then p and q are estimated.

Once with the differenced time series, the next step is to analyze the ACF and PACF. To analyze the ACF and PACF, the first step is to calculate the confidence interval at 95%, which is defined as: $\bar{x} \pm \frac{1.96s_x}{\sqrt{N}}$, where $\bar{x}$ is the mean of the time series x , s_x its standard deviation and N the total of observations. This confidence interval defines if the autocorrelation or partial autocorrelation in a certain lag is significant or not. If the lag is within the interval, it is not significant, and if it is outside the confidence interval, it is significant. Then, it is necessary to compare the ACF and PACF behavior with the characteristics described in Table 2, and to identify if they fit with an $AR(p)$, $MA(q)$ or $ARMA(p, q)$ model.

Table 2: Summary of ARMA models properties

	$AR(p)$	$MA(q)$	$ARMA(p, q)$
Model	$w_t = \phi_1 w_{t-1} + \dots + \phi_p w_{t-p} + a_t$	$w_t = a_t - \theta_1 a_{t-1} - \dots - \theta_q a_{t-q}$	$w_t = \phi_1 w_{t-1} + \dots + \phi_p w_{t-p} - \theta_1 a_{t-1} + \dots - \theta_q a_{t-q} + a_t$
Autocorrelation function (ACF)	Infinite; damped exponentials and/or damped sine waves; Tails off	Finite; cuts off after q lags	Infinite; damped exponentials and/or damped sine waves; Tails off
Partial autocorrelation function (PACF)	Finite; cuts off after q lags	Infinite; damped exponentials and/or damped sine waves; Tails off	Infinite; damped exponentials and/or damped sine waves; Tails off

Adapted from Box et al. (2008)

The next step is to identify p and q . If the time series fits with an AR model, then p is equal to the amount of significant lags before the PACF cuts off. In the case of an MA model, q is equal to the amount of significant lags before the ACF cuts off. Finally, if the model is an ARMA, it is necessary to use other tools to complete the model (BISGAARD; KULAHCI, 2011). One way is using the extended sample autocorrelation function (ESACF) proposed by Tsay and Tiao (1984), which applies for stationary and nonstationary ARMA models. The ESACF is based on the Yule-Walker equation 2.17, which establishes a relation between the first p autocorrelations and the process parameters.

$$\phi = R^{-1}\rho \quad (2.17)$$

$$\phi = [\phi_1, \dots, \phi_p], \quad \rho = [\rho_1, \dots, \rho_p], \quad R = \begin{pmatrix} 1 & \rho_1 & \cdots & \rho_{p-1} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{p-1} & \rho_{p-2} & \cdots & 1 \end{pmatrix}$$

The ARMA order identification using ESACF can be divided in two steps. The first step attempts to obtain consistent estimates of AR coefficients. Then, with such coefficients, it is possible to transform the ARMA series into a pure MA. In the second step, it is calculated the ACF of the transformed MA to identify the MA order.

Then, it is necessary to estimate the parameters to complete the model. For this purpose, there are different techniques, such as: method of moment estimators, least squares method and maximum likelihood (SHUMWAY; STOFFER, 2006). The method of moment estimators is based on the Yule-Walker equations. The least squares and maximum likelihood are based on the computation of errors ϵ_t . Next, there is a brief explanation of those methods.

The least squares method seeks to minimize the sum of the squares of the errors (or residuals²) S (VERBEEK, 2008). For an AR(1) model, the sum of squares is shown in Equation 2.18. This method requires to know the initial values to obtain a more realistic model. Another mechanism is to start the sum after the first p observations, assuming that previous errors are equal to zero. This least square estimation is known as *conditional least squares*, because it is conditioned to the first p observations (VERBEEK, 2008). In an AR(p) model with a zero mean, this assumption is shown in

²The difference between the dependent variable and the value predicted by the model

Equation 2.19.

$$S = \sum_t^n \epsilon_t^2 = \sum_t^n (x_t - x_{t-1})^2 \quad (2.18)$$

$$\epsilon_t = x_t - \phi_1 x_{t-1} - \dots - \phi_p x_{t-p} \quad (2.19)$$

Expressing the AR(p) model in matrix notations as $X_t = X_{t-1}^T \Phi + \epsilon$ where, $X_{t-1} = (x_{t-1}, x_{t-2}, \dots, x_{t-p})^T$ and $\Phi = (\phi_1, \dots, \phi_p)^T$ are both $p \times 1$ vectors. The sum of the squares is expressed in Equation 2.20.

$$S = \sum_{t=p+1}^n (X_t - \Phi X_{t-1})^2 \quad (2.20)$$

Then, solving for Φ

$$\hat{\Phi} = \left(\sum_{t=p+1}^n X_{t-1} X_{t-1}^T \right)^{-1} \left(\sum_{t=p+1}^n X_{t-1} X_t \right) \quad (2.21)$$

Remembering an MA(1) model $x_t = \epsilon_t + \theta_1 \epsilon_{t-1}$, since ϵ_{t-1} is not observed, it is not possible to use the least square method used for AR models. The solution is to use a nonlinear least square method (VERBEEK, 2008). Thus, the first step is to express ϵ_{t-1} as a function of observed x_t , as showing in equation 2.22. This step is possible only if the MA polynomial is invertible.

$$\epsilon_{t-1} = \sum_{j=0}^{\infty} (-\theta)^j (x_{t-j-1}) \quad (2.22)$$

Then, substituting Equation 2.22 in 2.20:

$$S = \sum_{t=2}^n \left(x_t - \theta \sum_{j=0}^{\infty} (-\theta)^j (x_{t-j-1}) \right)^2 \quad (2.23)$$

Finally, since x_t is not observed for $t = 0, -1, \dots$, it is necessary to cut off the infinite

sum up to $t - 2$.

$$S = \sum_{t=2}^n \left(x_t - \theta \sum_{j=0}^{t-2} (-\theta)^j (x_{t-j-1}) \right)^2 \quad (2.24)$$

The maximum likelihood is a method based on the data's probability distribution (VERBEEK, 2008). The likelihood function is defined as shown in 2.25, where $f(x_i; p)$ is the probability distribution for x_i , with parameter p .

$$L(p|x) = \prod_{i=1}^n f(x_i; p) \quad (2.25)$$

Then, Verbeek (2008) argues that it is better to work with the loglikelihood (Equation 2.26) to ease computational tasks. For ARMA models, Verbeek (2008) proposes to assume a normal distribution for ϵ_t . Assuming this, it is possible to obtain consistent estimators³, even when ϵ_t is not normal. Then, conditioned upon an initial value, the loglikelihood for ARMA models is shown in Equation 2.27.

$$\log L(p|x) = \log \prod_{i=1}^n f(x_i; p) \quad (2.26)$$

$$\log L(\phi, \theta, \sigma) = -\frac{T-1}{2} \log(2\pi\sigma^2) - \frac{1}{2} \sum_{t=2}^T T \frac{\epsilon^2}{\sigma^2} \quad (2.27)$$

2.3 Summary

In Chapter 2, an overview of WSN and SDN main concepts was presented. Then, the SDWSN concept was introduced and the IT-SDN's implementation and operation details were explained, which is the framework chosen to carry out this research. Finally, the basic Markov chain and time series mathematical theory, that was used in Chapter 5 to model the prediction mechanisms, was explained.

³For an estimate $\hat{\sigma}$ and σ_0 , $\hat{\sigma}$ is consistent if $\hat{\sigma} \rightarrow \sigma_0$ in probability as the sample size $n \rightarrow \infty$ (BROCKWELL; DAVIS, 2013)

3 LITERATURE REVIEW

This chapter presents the literature review, encompassing energy consumption models and energy consumption prediction in WSN.

3.1 WSN energy consumption models

The energy consumption in WSN depends on many hardware, software, and electrical characteristics. Therefore, hardware, software and energy sources are constantly evolving with the goal of improving the energy efficiency (CHEOUR; LAHMAR; ABID, 2011). In this manner, energy consumption models are constantly adapting to those changes.

The energy consumption in WSN nodes can be divided in three domains: sensing, processing, and communicating. Each module has specific characteristics and different modes of operation, which is why they are typically analyzed separately. Some works do analyze just one of the three domains, and others consider the three of them.

The analysis is limited to networks that used the standard IEEE 802.15.4 (IEEE, 2003) for communication. Even though other constrained devices standards are currently been used for WSN, such as BLE, the IEEE 802.15.4 standard has more than fifteen years since its first release and it is still a dominant technology. In this manner, most of the current WSN devices are IEEE 802.15.4 compliant, which enhances this research contribution.

The first IEEE 802.15.4 standard publication was in 2003, thus the period analyzed comprises from 2005 to 2017. This analysis starts reviewing each domain separately and then it does a global analysis of all works presented.

3.1.1 Sensing

The sensing module consumption varies with the type of sensor used in the application, but it also depends on the analog-to-digital converter (ADC) subsystem. The sensor front-end consist of five submodules (CALHOUN et al., 2005): sensor, preamplifier, anti-aliasing filter, analog-to-digital converter, and digital signal processor. Thus, the sensing module energy consumption depends on various factors such as sampling rate and the effective number of bits.

Zhou et al. (2011) proposed a sensing module energy consumption model that considers the number of sensors in the node, state transitions, and the total current consumption while sensing. Equation 3.1 shows the model, where N is the number of sensors, e_{on-off} is the energy consumption when the sensor changes from activate state to deactivate state, e_{off-on} is the energy consumption when the sensor pass from deactivate state to activate state, V_s is the working voltage, I_s the working current, and T_s is the interval of sensing operation.

$$E_{sensor} = N(e_{on-off} + e_{off-on} + V_s I_s T_s) \quad (3.1)$$

Ahmad et al. (2016) proposed a similar model but detailing the consumption while the sensor is operating. The Equation 3.2 shows the proposal, where E_{01} is the energy consumption when the sensor changes from deactivate state to activate state, E_{10} is the energy consumption when the sensor pass from activate state to deactivate state, and E_{11} is the consumption while the sensor is operating.

$$E_{SM} = E_{01} + E_{10} + E_{11} \quad (3.2)$$

Equation 3.3 details the consumption when the sensor is operating, where E_{ss} is the energy consumption due to the signal sampling, E_{ad} is the energy consumption by the ADC, and E_m is the consumption due to the signal modulation.

$$E_{11} = E_{ss} + E_{ad} + E_m \quad (3.3)$$

The level of detail presented by Ahmad et al. (2016) model could help to improve the estimation accuracy, but also it requires to gather more hardware usage information

than Zhou et al. (2011) model, which means more energy required to estimate the energy consumption. In the case of Zhou et al. (2011) model, it multiplies the energy consumption by the total amount of sensors. This assumes that all sensors working in one node have same power consumption, which is not always true. The methods were not tested with real devices to validate the results.

Two energy consumption tools that consider the sensing domain are AEON (LANDSIEDEL; WEHRLE; GOTZ, 2005) and Energest (DUNKELS; OSTERLIND; TSIFTES, 2007). AEON energy model is based on the Atmega128 microprocessor. For sensing, it considers just the energy consumption when the sensor is activated. Energest is another energy estimation tool developed for Contiki (DUNKELS; GRONVALL; VOIGT, 2004). This tool is based on the work by Dunkels, Osterlind and Tsiftes (2007) and it basically counts the time the node remains on each state and then the user sets the energy consumption of each state to obtain the total energy consumption. Same as AEON, Energest model for sensing domain is simple.

Summarizing, the energy tools consider just the consumption when the sensor is activated but they neglect the activation and deactivation consumption. (ZHOU et al., 2011) and Ahmad (AHMAD et al., 2016) proposals consider the three factors. The last ones present a model that better represent the real energy consumption.

3.1.2 Processing

The processing unit consist of a microprocessor or microcontroller (MCU) with memory. This unit has the control of the sensor node and it processes the information received (ZHENG; JAMALIPOUR, 2009).

The microprocessor energy consumption has two components: transistors switching power, and dissipation by leakage currents. *Complementary Metal Oxide Semiconductor* (CMOS) is a transistors technology used to fabricate low-cost microprocessors for sensor platforms and its power dissipation depends on the clock frequency, device capacitance, and square of the voltage switching. In this way, the energy consumption for a single task can be represented by the sum of the transistors dissipation and the dissipation due to leakage currents as shown in Equation 3.4 (AKYILDIZ; VURAN, 2010), where N is the number of clock cycles for the task analyzed, C is the total switching capacitance, V_{dd} is the supply voltage, I_O is the leakage current, n is a constant that depends on the processor hardware, V_T is the threshold voltage, and f is the

operation frequency.

$$E_p = NC(V_{dd})^2 + V_{dd}(I_O e^{V_{dd}/nV_T}) \left(\frac{N}{f} \right) \quad (3.4)$$

Zhou et al. (2011) proposed a model that considers the energy consumption while the microprocessor is operating in a certain state, and also the consumption when changing to another state. This model defines three states: sleep, idle, and run. The microprocessor can pass from any of those states to another, that means five different transitions. Equation 3.5 shows the total energy consumption by processing, where $P_{cpu-state}(i)$ is the power of the state i , $T_{cpu-state}(i)$ is the time in the state i , $N_{cpu-change}(j)$ is the frequency of state transition j , and $e_{cpu-change}(j)$ is the energy consumption of state transition j .

$$E_{cpu} = \sum_{i=1}^m P_{cpu-state}(i) T_{cpu-state}(i) + \sum_{j=1}^n N_{cpu-change}(j) e_{cpu-change}(j) \quad (3.5)$$

Typically, manufacturers provide the information about the power consumption of different states of operation, but do not provide the energy consumption of state transitions. Zhou et al. (2011) propose equation 3.6 to calculate it, where $T_{init-end}(j)$ is the time to complete the transition j , $P_{init}(j)$ is the power of the initial state, and $P_{end}(j)$ is the power of the final state.

$$e_{cpu-change}(j) = T_{init-end}(j) \frac{P_{init}(j) + P_{end}(j)}{2} \quad (3.6)$$

Ahmad et al. (2016) propose a similar approach to Zhou et al. (2011). It considers the consumption on each processor's state and state transitions, but it does not specify the state transitions consumption or how to calculate them.

Both works lack performance evaluation comparing them with real experiments, which difficults a comparison of both models. Also, Ahmad et al. (2016) do not specify how to calculate the consumption by state transitions. In the case of Zhou et al. (2011), it is assumed that state transitions power consumption is the average of the power consumption of both states in the transition. Also this assumption was not verified with measurements in real devices.

Lu et al. (2017) proposed a model for networks that used data aggregation. Lu et al. (2017) assert that in this kind of networks, data processing is embodied on aggregating

and compressing tasks. In this manner, the energy consumption of data aggregation can be calculated as shown in Equation 3.7, where E_{ec} is the energy consumption of computation, x is the number of data inputs, k_{in}^j is the size of the j th data input.

$$E_{Da} = \begin{cases} E_{ec} \sum_{j=1}^x k_{in}^j & x > x \\ 0 & x = 1 \end{cases} \quad (3.7)$$

Lu et al. (2017) conducted several simulations using the model proposal. Simulations were implemented using MiXim, a modelling framework. The energy model was not compared with other energy models neither in real devices.

Similar to Landsiedel, Wehrle and Gotz (2005), PowerTOSSIM z¹ (PERLA et al., 2008) use a model based on the Atmega128 microprocessor. To estimate the consumption, both models consider the active state and the 6 low-power modes. The main difference between them is that the first one do a linear estimation adding up all the components in a certain time, and the second one use a non-linear battery model to estimate the residual energy. An issue is that neither of them consider the consumption by state transitions.

Energest (DUNKELS; OSTERLIND; TSIFTES, 2007) defines two main states: active and low power mode. Also, it can measure the flash memory write and read time, and the serial port use. Similar to AEON and PowerTOSSIM z, Energest does not considers the consumption by state transitions.

Haas, Wilke and Stöhr (2012) studied the energy consumption estimation accuracy of Avrora (TITZER; LEE; PALSBERG, 2005). Avrora is a simulation and analysis tool for programs written for the AVR microcontroller produced by Atmel and the Mica2 sensor nodes². The energy consumption results using Avrora were compared with the results obtained by a testbed with SANDBed (HERGENRÖDER; HORNEBER; WILKE, 2009) and using MICAz nodes.

Haas, Wilke and Stöhr (2012) identified three main issues in Avrora (TITZER; LEE; PALSBERG, 2005): incorrect calibration of energy model, the cost of state transitions is neglected, and hardware variations in same model platforms. Avrora uses datasheet information to estimate the consumption but the testbed measurements

¹PowerTOSSIM z is the improved version of PowerTOSSIM (SHNAYDER et al., 2004) <http://www.eecs.harvard.edu/shnayder/ptossim/>

²<http://compilers.cs.ucla.edu/avrora/>

showed a considerable difference in all results. In the case of state transitions, Avrora assumes they occur instantaneously and testbed measurements showed that those transitions implies considerable consumption. Related to the third point, the testbed showed a high variance in energy consumption for different MICA nodes when measuring the same state.

To improve the energy consumption accuracy, Haas, Wilke and Stöhr (2012) proposed Avrora+. This Avrora version addressed the three issues mentioned before. In terms of modeling, Avrora+ propose a five-state machine for the microcontroller. Those five states are: active, idle, power save, power down, and ADC. This approach join the consumption of processing and sensing, but considering only the ADC consumption for the sensing domain. It also considers the consumption by state transitions, including them as new states.

3.1.3 Communication

Communication domain can be divided in two main tasks: transmitting data, and receiving data. The transceiver is the main hardware for communication in WSN, and it is typically composed by a mixer, a frequency synthetizer, voltage-controlled oscillator (VCO), phase-locked loop (PLL), demodulator, and power amplifiers (AKYILDIZ; VURAN, 2010). All this circuitry implies a significant consumption, making the communication the most energy costly domain.

Wang, Hempstead and Yang (2006) proposed a power consumption model for communication, including the basic transceiver parts and a channel model. Equation 3.8 represents the power consumption by transmitting data in function of the transmission range. Equation 3.9 represents the power consumption by receiving data.

$$P_T(d) = P_{TB} + P_{TRF} + P_A(d) = P_{TO} + P_A(d) \quad (3.8)$$

$$P_R = P_{RB} + P_{RRF} + P_L = P_{RO} \quad (3.9)$$

In equation 3.8 P_{TB} represents the power consumption in baseband DSP for transmitting, P_{TRF} represents the power consumption by front-end circuit for transmitting, and $P_A(d)$ is the power consumption by the power amplifier in function of the trans-

mission range. Since P_{TB} and P_{TRF} do not depend on the transmission range, both are substituted by the constant P_{TO} . In equation 3.9 P_{RB} represents the power consumption in baseband DSP for receiving, P_{RRF} represents the power consumption by front-end circuit for receiving, and P_L is the consumption by the low-noise amplifier of the receiving circuit. The three terms in Equation 3.9 are constants and are substituted by one constant P_{RO} .

To guarantee a reliable communication, Wang, Hempstead and Yang (2006) define the minimum power required in the transmitting antenna. To calculate it, they first define the relation between the power in the receiving antenna (P_{RX}) and the power in the transmitting antenna (P_{TX}), using a channel model considering only path loss. Equation 3.10 shows the relation where A is defined by the antenna's characteristics and α represents the path loss exponent.

$$P_{RX} = \frac{P_{TX}}{A \times d^\alpha} \quad (3.10)$$

Then, considering the radio power amplifier drain efficiency η , the relation between the power consumed by the amplifier and the power received by the antenna to transmit is shown in equation 3.11

$$\eta = \frac{P_{TX}}{P_A} \quad (3.11)$$

Substituting equation 3.11 in equation 3.8, and considering the minimum receiving power for reliable communication, the power consumption by transmitting is shown in equation 3.12.

$$P_T(d) = P_{TO} + \frac{P_{RX-min} \times A \times d^\alpha}{\eta} \quad (3.12)$$

One year later, Wang and Yang (2007) proposed *CSECM*, a energy consumption model for communication in WSN. Unlike their previous work (WANG; HEMPSTEAD; YANG, 2006), *CSECM* considers the consumption due to state transitions and it is based in a five-state state machine, including: *Power Off*, *Power Down*, *Power Save*, *TX*, *RX*. Equation 3.13 represents the energy consumption for a certain period of

time.

$$E_D = \sum_{k=1}^n T_{S(k)} I_{S(k)} V_{S(k)} + \sum_{k=0}^{n-1} T_{S(k) \rightarrow S(k+1)} I_{S(k) \rightarrow S(k+1)} V_{S(k) \rightarrow S(k+1)} \quad (3.13)$$

$$S(k) \in \{Power_{Off}, Power_{Down}, Power_{Save}, TX, RX\}$$

In Equation 3.13, E_D is the total energy consumption for a certain period. E_D estimation includes the consumption by the time remained on each state and the consumption by the state transitions.

Kan et al. (2007) proposed an energy model that omitted the processor consumption and differs from *CSECM* because it is not completely based on operating states. Kan et al. (2007) study the CC1000 transceiver hardware and proposed an equation to describe it. This equation includes the energy consumption by transmitting and receiving, and the power amplifier energy consumption. Also, this model considers the transceiver switching consumption. The model proposed was used to evaluate the consumption performance when varying the data rate. Results showed that it is possible to reduce the energy per bit choosing the correct data rate. Results were not verified with measurements on real devices.

Similar to Wang and Yang (2007), Zhou et al. (2011) proposed a five-state model, including: transmitting (TX), receiving (RX), idle, sleeping, and Clear Channel Assessment-Energy detect (CCA-ED). The model considers the consumption of each state and by state transitions, as shown in equation 3.14. Unlike Wang and Yang (2007) proposal, it calculates the consumption of transmitting and receiving using the packet length and data transferring rate. For idle, sleeping, and CCA-ED they use the time remained on each state. The states energy consumption can be expressed as equation 3.15.

$$E_{transceiver} = E_{transceiver-state} + E_{transceiver-change} \quad (3.14)$$

$$E_{transceiver-state} = \sum_{i=1}^{N_{TX}} \frac{P_{TX} L_i}{R} + \sum_{i=1}^{N_{RX}} \frac{P_{RX} L_i}{R} + P_{idle} T_{idle} + P_{sleep} T_{sleep} + P_{CCA} T_{CCA}$$

$$E_{transceiver-state} = \sum_{i=1}^{N_{TX}} \frac{V_{tr} I_{TX} L_i}{R} + \sum_{i=1}^{N_{RX}} \frac{V_{tr} I_{RX} L_i}{R} + V_{tr} (I_{idle} T_{idle} + I_{sleep} T_{sleep} + I_{CCA} T_{CCA}) \quad (3.15)$$

In equation 3.15 N_{TX} and N_{RX} are the total transmitted and received packets. L_i is

the size length of the i^{th} packet and R is the data transferring rate. V_{tr} is the operating voltage. I_{TX} , I_{RX} , I_{idle} , I_{sleep} , and I_{CCA} are the electric current on each state. T_{idle} , T_{sleep} , and T_{CCA} are the time spent in idle, sleeping, and CCA-ED states.

To complete equation 3.14, Zhou et al. (2011) proposed equation 3.16 to calculate the consumption by state transitions

$$E_{transceiver-change} = \sum_{j=1}^n N_{transceiver-change}(j) e_{transceiver-change}(j) \quad (3.16)$$

$$e_{transceiver-change}(j) T_{init-end}(j) \frac{P_{init}(j) + P_{end}(j)}{2} \quad (3.17)$$

Jalsan, Flouri and Feltrin (2014) proposed a discrete-event approach, which the main goal is to balance the trade-off between the accuracy and computational complexity of the energy estimation. Different from Wang and Yang (2007), and Zhou et al. (2011), Jalsan, Flouri and Feltrin (2014) propose to use primitive task, which the energy consumption depends on certain parameters. Then, to calculate the energy consumption for a certain period of time, it is necessary to count the times each task were executed, multiply it by the corresponding task consumption, and then to add up all the results. Equation 3.18 shows the process described before where n_t is the number of executions of task t and E_t is the energy consumption of task t .

$$E_{node} = \sum_{t \in T_p} n_t E_t \quad (3.18)$$

The model was simulated and compared with another energy model that they named *simple model* (JALSAN; FLOURI; FELTRIN, 2012). The results showed that using the discrete-event model, the energy consumption calculated is significantly higher than using the *simple model* because the assumptions of the second one produce incorrect estimations. The accuracy of the models were not verified with measurements on real devices.

Ahmad et al. (2016) propose a consumption model for communication based on First Order Radio Model (FORM). The model considers two radio amplifier types: free space and multipath. Also it considers the packet length (l), coding rate (R_{code}), bit rate (R), and decoding ($E_{dec-bit}$). Equation 3.19 represents the consumption by transmitting and Equation 3.20 represents the consumption by receiving, where $P_{tx-elec}$ is the power

per bit to run the electronic transmitter circuitry, P_0 and P_1 represent the power of the amplifier for free space and the amplifier for multipath respectively. d_0 is the reference distance, defined as $d_0 = \sqrt{\frac{\epsilon_0}{\epsilon_1}}$, where ϵ_0 and ϵ_1 are the amplification factors for free space and multipath transmitters respectively.

$$E_{Tx} = \begin{cases} \frac{l}{RR_{code}}(P_{tx-elec} + P_0 d^2) & d \leq d_0 \\ \frac{l}{RR_{code}}(P_{tx-elec} + P_1 d^4) & d \geq d_0 \end{cases} \quad (3.19)$$

$$E_{Rx} = \frac{l}{RR_{code}}(P_{rx-elec}) + (lE_{dec-bit}) \quad (3.20)$$

Ahmad et al. (2016) compares A-ECM with a FORM which does not considers energy consumption by processing and sensing. Also, A-ECM includes data rate, coding rate, and decoding in the communication model, which typically are ignored in FORM. Results show that A-ECM estimates an energy consumption significantly higher than FORM. That means that consumption by processing, sensing, and the modifications on the communication model have a significant affect. An issue on this work is that results were not compared with real energy consumption measurements, thus it is not possible to determine the model accuracy.

Abo-Zahhad et al. (2015) provide an energy model considering physical and MAC layer parameters. The main goal is to determine the transceiver energy consumption necessary for transmitting one bit of data successfully. This work considers just the consumption by the transceiver, neglecting the consumption by sensing and processing. The energy consumption by the transceiver for transmitting and receiving is represented in equations 3.21 and 3.22 respectively.

$$\begin{aligned} P_{tx} &= P_{Dct} + P_{Act} + P_{amp} + P_t \\ P_{tx} &= P_{to} + P_t(\alpha + 1) \end{aligned} \quad (3.21)$$

$$P_{rx} = P_{Dcr} + P_{Acr} + P_{LNA} = P_{ro} \quad (3.22)$$

In Equation 3.21, P_{Dct} and P_{Act} are the power consumption of the digital and analog circuit in the transmitter side. P_{amp} is the power consumption of the power amplifier and P_t the transmitted power. Then $\alpha = (\xi/\eta) - 1$, where ξ is the drain efficiency and η the peak-to-average ratio. In Equation 3.22 P_{Dcr} and P_{Acr} are the digital and analog

circuit power consumption in the receiver side, and P_{LNA} is the low noise amplifier power consumption. To calculate the energy consumption for a MAC protocol, this model considers three activities: transmitting, receiving, and sleeping. Thus, if the node is neither transmitting nor receiving a packet, it is sleeping. Normalizing the time spent on each activity, the total power consumption for a MAC protocol is represented in Equation 3.23.

$$P_{MAC} = t_{tx}P_{tx} + t_{rx}P_{rx} + (1 - t_{tx} - t_{rx})P_{slp} \quad (3.23)$$

In this manner, knowing the power consumption, the time necessary to send a packet, and the total amount of data delivered, the energy per successfully transferred bit can be calculated using Equation 3.24.

$$E_b = \frac{\text{Total energy consumed}}{\text{Total data delivered}} \quad (3.24)$$

Abo-Zahhad et al. (2015) also presented the results of simulating two low power transceivers and calculating the average energy consumption per bit in function of the power transmission for a fixed distance. The software used for simulations was MATLAB³. The results showed the optimal transmission power for a fixed distance. Then, four different modulation schemes (BPSK, 4-PSK, 4-QAM, and 8-QAM) were simulated, varying the distance between both radios. Results shown that increasing the distance the optimal transmission power increases too. The behavior was the same for all modulation schemes tested. The work presented interesting results to optimize the MAC protocol energy consumption, but it does not compare the results obtained in the simulations with real energy consumption measurements.

Lu et al. (2017) proposed an energy model where the communication energy consumption is divided in: energy consumption by emitter electronics and energy consumption by receiver electronics. The emitter electronics energy consumption is shown in Equation 3.25, where k_{out} is the message's size, E_{et} is the energy consumption of radio dissipation, E_{af} is the energy consumption of the radio amplifier, and d is the transmission distance.

$$E_{Tx} = k_{out}(E_{et} + E_{af}d^2) \quad (3.25)$$

³<https://www.mathworks.com/products/matlab.html>

The receiver electronics energy consumption is shown in Equation 3.26, where k'_{in} is the input's size.

$$E_{Rx} = k'_{in} E_{et} \quad (3.26)$$

Wu, Xiong and Wu (2017) proposed a clustering algorithm based on nodes' residual energy. To estimate the residual energy, a energy consumption model that considers only the consumption by communication is proposed. The energy consumption is divided in transmitting consumption and receiving consumption. The transmitting consumption is shown in Equation 3.27, where $E_{Tx-elec}$ is the energy consumption by the transmitter circuitry, E_{Tx-amp} is the consumption by the amplifier circuit, m is the number of bits to transmit, and d is the distance to the receiving node. The receiving energy consumption is shown in Equation 3.28.

$$E_{Tx}(m, d) = E_{Tx-elec}(m) + E_{Tx-amp}(m, d) \quad (3.27)$$

$$E_{Rx}(m) = mE_{Tx-elec} \quad (3.28)$$

AEON, PowerTOSSIM z, and Energest are very similar in terms of communication modeling. All of them divide the transceiver usage in receiving and transmitting. In the case of Energest, it could be used in cooperation with *Compower* Contiki's library⁴, and calculate the time the transceiver remains in idle listening. Same as mentioned in previous sections, those tools do not consider the state transitions consumption into the communication module.

In the case of Avrora simulation tool, it models the communication consumption using four states: listen, transmitting, receiving, and idle. This work tested its model performance comparing its results with real measures and also it considers the consumption by state transitions. Global results (processing and communication) showed and average error less than 5%. Its measures on real devices helped to calibrate the model and, in terms of communication, to adequate it according to platform behavior. For this work the mote used was MICAz.

⁴<https://github.com/contiki-os/contiki/blob/master/core/sys/compower.h>

3.1.4 Global analysis

The analysis presented in previous sections is based on 13 works related with WSN energy consumption. The constant in all works is the study of the energy consumption by the communication module. This is because it is the most energy-costly module in the WSN platform. The processing module consumption is considered in seven works, and the sensing module consumption just in four of them. That means that only the 30% percent of those works consider the total energy consumption sources on its models.

The impact of considering only the communication module instead of the three modules is shown by Zhou et al. (2011) and Abo-Zahhad et al. (2015). In the case of Zhou simulation results, the difference when considering all domains and considering only the radio module is around 15%. Another detail that changed over the time was the consideration of state transitions consumption into the model. First works, including three of the four energy tools, neglected this consumption. Then, Avrora+ (HAAS; WILKE; STÖHR, 2012) showed that this consumption can be significant in the final result.

In this manner, an accurate energy consumption model has to consider the consumption by sensing, processing, communicating, and the consumption by state transitions. Also, its performance has to be evaluated in real testbeds.

Table 3 summarizes the main energy models and the energy consumption domains considered.

3.2 Energy consumption prediction in WSN

One way to reduce the energy consumption in WSN is by using prediction approaches. The main idea is to reduce the usage of high energy-cost components or processes, by predicting its behavior. For example, a cluster head can run a prediction model to approximate sensing values of nodes that are linked to it. Accurate forecasting allows to reduce the transceiver usage for both the nodes: cluster members and the cluster head. Prediction models are also used to forecast the node energy consumption and the network lifetime. As shown by Egea-Lopez et al. (2006) processes, as energy models, that run almost continuously and independently on every node, require significant processing resources. In this way, using prediction models is an alternative to

Table 3: Energy models summary

Energy model	Sensing	Processing	Communication	State transitions	Performance Evaluation
AEON	X	X	X		Testbed using TinyOS applications
Wang, Hempstead and Yang (2006)			X		Not specified
Wang and Yang (2007)			X	X	Experiments with real devices
Kan et al. (2007)			X		Mathematical analysis and simulations
Energest	X	X	X		Experiments with real devices
PowerTOSSIM z		X	X		Simulations using TinyOS applications
Zhou et al. (2011)	X	X	X	X	Simulations using OPNET
Haas, Wilke and Stöhr (2012)		X	X	X	Testbed using Micaz platform and TinyOS
Jalsan, Flouri and Feltrin (2014)			X		Simulations using Tossim Simulator
Abo-zahhad et al. (2015a)			X		Simulations using Matlab
Ahmad et al. (2016)	X	X	X	X	Simulations. Simulator not specified
Lu et al. (2017)		X	X		Simulations MiXim
Wu, Xiong and Wu (2017)			X		Simulations. Simulator not specified

Source: Author

alleviate the process overhead, which also can help to reduce the energy consumption.

Two techniques used in WSN for energy prediction are Markov chains and time series. Section 3.2.1 and Section 3.2.2 review the literature related to Markov chain and time series approaches used to predict energy consumption in WSN.

3.2.1 Markov chain approaches

Mini et al. (2005) modeled the node energy consumption using 4 states: sensing off and radio off, sensing on and radio off, sensing on and radio receiving, and sensing on and radio transmitting. Then, knowing the transition probabilities between states, each node forecasts its behavior to estimate the energy consumption rate for a certain future time. To calculate those probabilities, each node runs a training period to count the transitions between states and the total-step amount on each state. Then, each node informs to a monitoring node its energy consumption rate and available energy. To reduce the communication overhead, the node updates its parameters when the prediction model has an error higher than a threshold previously defined.

The performance of this approach was compared with a statistical approach based on time series, and with a naive approach, which does not implement a prediction model. The Markov chain proposal obtained the best prediction power and a better energy efficiency than the time series approach. Details such as comparing the energy consumption prediction with real measures and comparing the processing overhead of

each method were not considered in this work. Also, the consumption model ignores the consumption by state transitions and by processing.

Achir and Ouvry (2004) proposed another energy consumption and prediction model using Markov chains (ACHIR; OUVRY, 2004). Unlike Mini et al. (2005), this work used the Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) protocol to model the node energy consumption. This Markov chain model considers five states: idle mode, active (sensing, transmission attempt), transmission, backoff, and collision. It is possible to have several backoff and collision states, therefore those are considered in the model.

The Achir and Ouvry (2004) model performance was compared with two adaptive filters: simple average filter, and Least Mean Square (LMS) filter. The three prediction model were compared with real devices energy measurements. Results showed a better precision using the Markov chain model.

Lin et al. (2007) proposed a prediction model using a Markov chain with 5 states: sending, receiving, monitoring, apperceiving (defined as: sensor module turned on and radio module in sleep mode), and sleeping. Those states consider just the sensor module and the communication module. Then, Lin et al. (2007) proposed EPCH, a dynamic cluster-based routing protocol. EPCH uses the remaining energy to chose the next cluster head. EPCH was compared with LEACH and LEACH-F protocols and results showed that EPCH consumption rate is the lowest and improve the network lifetime. The results were based on simulations.

3.2.2 Time series

Mini et al. (2005) use the Autoregressive Integrated Moving Average (ARIMA) model as a forecasting approach to create an energy map. ARIMA is a model to forecast time series proposed by Box and Jenkins (1970) and it is a systematic methodology that could incorporate autoregressive (AR) and moving average (MV) approaches.

To implement an ARIMA model, it is necessary to identify three parameters: p , d , and q . p is the number of autoregressive terms, d is the number of nonseasonal differences needed for stationarity, and q is the number of lagged forecast errors in the prediction equation. Also it is necessary to identify the AR and MV coefficients. In Mini et al. (2005) proposal all nodes identify their parameters and send them to a

monitoring node. Then, this node does the forecasting and creates the energy map. The energy model used consider four states: sensing off and radio off, sensing on and radio off, sensing on and radio receiving, and sensing on and radio transmitting.

The performance of this approach was compared with a Markov chain approach, and with a naive approach, which does not implement a prediction model. The Markov chain approach estimation accuracy and energy efficiency obtained was better than ARIMA model. The energy efficiency was measured according to the transmitting overhead. ARIMA overhead is of 40 bytes and Markov chain overhead just 8 bytes. This work did not consider the consumption by state transitions and processing. Also, it did not conduct a performance analysis with real devices.

Ali et al. (2010) presented an event level prediction framework which is capable of predicting nodes energy levels, but not limited to it. The framework uses the ARIMA model and same as Mini et al. (2005), nodes send the parameters to the sink and then it runs the prediction algorithm.

This work does not explain how the energy consumption is calculated, it assumes the node already knows how to calculate it. The framework performance was tested with simulations and using MATLAB⁵.

3.2.3 Summary

The works presented in section 3.2.1 and 3.2.2 are examples of Markov chain and time series approaches used to predict the energy consumption in WSN. Mini et al. (2005) compare both approaches, using the prediction accuracy and energy consumption as main metrics. Markov chain approach obtained a better performance than time series in both metrics. In the case of Achir and Ouvry (2004) proposal, Markov chain was compared with two adaptative filters, obtaining the best precision of three approaches.

A similarity in Mini et al. (2005), Achir and Ouvry (2004), and Lin et al. (2007) Markov chain proposals is that each node estimates and predict its energy consumption. Then, if required, it informs to other nodes about its energy level or consumption rate. A difference of those three works is that they contemplate different energy states. Mini et al. (2005) time series proposal and Ali et al. (2010) have two similarities: both

⁵<https://www.mathworks.com/products/matlab.html>

implemented the ARIMA prediction model (BOX; JENKINS, 1970), and both separate the processing of the prediction algorithm. This separation is explained next: first, each node calculates the model parameters. Then, the nodes send those parameters to a central node with better capabilities. Finally, the central node execute the energy consumption forecasting.

Different from Achir and Ouvry (2004), the other works did not perform real experiments to measure the prediction accuracy. Also, Ali et al. (2010) do not specify how the energy consumption was calculated. Finally, the consumption by state transitions was neglected in all these works.

Summarizing, there is a lack of experiments with real devices running energy prediction models. The energy model used to predict the consumption can be improved by considering factors as state transitions. Finally, there is not an implementation of Markov chain and time series to predict the energy consumption in a SDWSN.

4 PRELIMINARY RESULTS

This chapter presents the preliminary results of this research as follows.

- *Duty Cycle Based Energy Management Tool for Wireless Sensor Networks*: Presented in the XXXIV Brazilian Communications and Signal Processing Symposium, 2016. Authors: Gustavo A. Núñez and Cintia B. Margi.
- *Energy Map Model for Software-Defined Wireless Sensor Networks*: Submitted to XXXV Brazilian Communications and Signal Processing Symposium, 2017. Authors: Gustavo A. Núñez and Cintia B. Margi.

4.1 Duty Cycle Based Energy Management Tool for Wireless Sensor Networks

Different approaches have been explored to reduce the energy consumption in WSN, and duty cycle based algorithms for energy management can be pointed as one of them. Zhang et al. (2010) derive a relation between the duty cycle and the distance of the node with the sink. The method proposed was compared with a constant duty cycle and results showed an improvement by up 32%. Pereira and Margi (2012) proposed EM, an energy management tool. The tool intends to reduce the energy consumption limiting the node capabilities and reducing the duty cycle. The node capabilities limitation and the duty cycle depends on the node's energy level. The energy management tool proposed follows the Pereira and Margi (2012) approach, limiting the node capabilities, but in this case restricted to the communication module. Unlike Zhang et al. (2010) and Pereira and Margi (2012), the energy management tool performance is analyzed in two different traffic flows, two routing protocols and three node roles: generating traffic, routing packets, being a sink.

4.1.1 The tool proposed

The energy management tool proposed allows to control the node communication module duty cycle. The main objective of this tool is to manage the communication module usage in order to increase the network lifetime. To reduce the duty cycle could affect communication performance metrics, for example: delay, reliability, and delivery rate (PARK et al., 2013). For this reason, this tool allows to gradually reduce it according to the remaining energy. In this manner, the tool can be calibrated to balance the communication/lifetime performance, according to the specific application.

The tool was programmed using Contiki (DUNKELS; GRONVALL; VOIGT, 2004) as operating system, and using ContikiMAC (DUNKELS, 2011), which is a radio duty cycle (RDC) mechanism with three main characteristics: a power efficient timing scheme, a phase-lock mechanism to make transmission efficient, and a fast sleep mechanism. Thus, a node with ContikiMAC is periodically waking up to listen for packet transmissions. When a packet is detected, the radio remains on to receive the packet. Then, it sends a link layer acknowledgement to indicate the packet was successfully received. During this time, the transmitter is repeatedly sending the packet until it receives the acknowledgement. The ContikiMAC normal operation is shown in Figure 3.

Figure 3: ContikiMAC normal operation

Source: Author

The energy management tool was programmed upon ContikiMAC, duty-cycling the radio module and ContikiMAC operation at the same time. Which means that when the tool turns off or turns on the radio module, it does the same action with the ContikiMAC. This dynamic is depicted in Figure 4 where the downside graphic represent a control signal. When the control signal is “up”, it enables the ContikiMAC, and when the control signal is “down”, it disables the ContikiMAC.

Figure 4: The proposed tool working upon ContikiMAC

Source: Author

4.1.2 Implementation and simulation

The algorithm implemented is depicted in Figure 5. First, the program sets the parameters that were previously defined by the user. Those parameters are: the amount of states, the threshold of each one, the total energy amount, the average current of each state, and the operating voltage. Then, the program initializes four timers. Two of them are to set control signal frequency and duty cycle, the third one sets the frequency for the *CalculateConsumption* function, and the fourth one sets the frequency for the *CalculateState* function. Previous functions and the *ControlSignal* function are explained next:

- *CalculateConsumption*: this function is in charge of calculating the total energy consumption within a certain period. The energy consumption model is represented in Equation 4.1, where E_T is the total energy consumption in a certain period, i represents the states of the node, k is the total of states considered, t_i is the time the node spends in the state i , V_s is the operating voltage, and I_i is the average current consumption in the state i . The energy model considers four states: transmitting, listening, processing, and low power mode. This function uses Energest (DUNKELS; OSTERLIND; TSIFTES, 2007) to measure the time spent on each state. Then, the function calculates the equivalent energy for those

intervals and subtracts the total from the total energy level.

$$E_T = \sum_{i=1}^k t_i V_s I_i \quad (4.1)$$

- *CalculateState*: this function checks the remaining energy level and compares it with the next state threshold. If the current energy level is lower than the next state threshold, the state changes to the next one.
- *ControlSignal*: this function checks whether the mote has a task in the queue, if so, the mote waits until all the tasks are finished. When the mote is finally idle, the function deactivate the ContikiMAC and sends the mote to sleep. The checking frequency and the sleeping period depends on the control signal timers.

To measure the tool performance, it was tested in three different scenarios. The topology configuration used in the experiments is shown in Figure 6. The scenarios configuration is explained next:

- Scenario 1: A broadcasting scenario with three nodes. Node 2 sends a message every 60 seconds and nodes 1 and 3 remain listening to receive it. The communication use the Rime stack (DUNKELS, 2007).
- Scenario 2: In this scenario node 1 is sending packets to node 3, which works as a sink. Node 2 works as a router. This scenario also use the Rime stack.
- Scenario 3: Same as Scenario 2 but using the Routing Protocol for Low-Power and Lossy Networks (RPL - RFC 6550).

The tool was configured in 5 states and the thresholds were defined as follow: 100%, 80%, 60%, 40%, 20%. Simulation was conducted using COOJA Simulator, emulating the TelosB mote.

Figure 5: Proposed tool algorithm

Source: Author

4.1.3 Results

This section shows the results of simulating the three scenarios when using ContikiMAC and when using ContikiMAC and the tools proposed. Each scenario was simulated ten times, running during one hour.

Figure 7 show the results for all scenarios. The histogram is divided by scenarios

Figure 6: Toplogy configuration

Source: Author

and for each scenario it is shown the energy consumption of each node (Figure 6) for all duty-cycle states. In the Scenario 1, node 2 saves 29,32% of energy per hour when operating in State 5 (20% duty cycle). Node 1 and node 3 save 27,21% of energy per hour, with a standard deviation of 1,5 %, also operating in State 5. Scenario 2 results show that node 1 and node 3 reduce its energy consumption up to 28,54% and 28,65% respectively, with a standard deviation of 2 %. The low standard deviation is due the few nodes in the network, thus, the energy consumption does not have significant fluctuations. For node 2 (router) the consumption is always higher when using the tool than the consumption when using just ContikiMAC.

Analyzing the time measured for transmitting and listening for the node 2 when using ContikiMAC and when using the tool in State 2 (state with maximum consumption), the average transmitting time per hour increases from 910 ms to 6940 ms, and listening time increases from 21090 ms to 26060 ms. On the other hand, for the other two nodes, listening time decreases from 22690 ms to 18590 ms per hour for node 1, and from 20170 ms to 18060 ms for node 3.

Lastly, the results for Scenario 3 show it is not possible to reduce the energy consumption for node 1 and 2. The node 3, reduce the energy consumption when reached State 3. In the case of node 1, the average transmitting time per hour increases from 6030 ms to 31990 ms, and listening time increases from 21090 ms to 26060 ms when the tool runs over State 3 (state with maximum consumption for this node). In the case of node 2, the average transmitting time per hour increases from 3660 ms to 8620 ms, and listening time increases from 21920 ms to 24500 ms when the tool runs over the State 2 (state with maximum consumption for this node).

Figure 7: Results for all scenarios

The main reason for the increasing in the energy consumption was the loss of packets. This happens because the proposed tool does not coordinate the node's sleeping schedule with its neighbors sleeping schedule. Thus, it is possible that a node sends a message to a neighbor which is sleeping and with the ContikiMAC deactivated, being impossible to receive the message, and consequently, increasing the retransmissions. This loss of packets has more impact in Scenario 3 because of control message overhead in RPL.

Summarizing, the tool proposed obtains good results working together with ContikiMAC and Rime stack, but this version does not work well with RPL. The implementation of this tool was helpful to have a first experience with energy models and operating systems in WSN. As future work, it is recommended to test the tool with more nodes. Also, it could be interesting to try a synchronization between neighbors duty cycle to reduce packets loss.

4.2 Energy Map Model for Software-Defined Wireless Sensor Networks

This section presents a method to construct an energy map into a SDWSN. It is inspired in the Markov chain approach proposed by Mini et al. (2005), but instead of running the prediction model on each node, the controller obtains information of each node behavior and then it estimates the energy consumption rate. To measure the model performance, it was compared with an energy map similar to Mini et al. (2005) method, which executes the prediction model on each node.

4.2.1 Energy consumption prediction model

Sensor nodes can be modeled as a device with different states of operation, such as: processing, transmitting, listening, sensing, and different low power modes. Therefore, it is possible to model each sensor node as a Markov chain, where each operation mode can be represented as a state of the Markov chain.

It was defined that the relation $X_n = i$ represents a node in a operation mode i at time-step n . The probability that the next state be j can be represented as P_{ij} . For a Markov chain defined by M states, the probability that a node in the state i will be in state j after n -step transitions is given by the *Chapman-Kolmogorov* equation (LEVIN; PERES; WILMER, 2009), and it is defined as:

$$P_{ij}^n = \sum_{k=1}^M P_{ik}^r P_{kj}^{(n-r)} \quad for \quad 0 < r < n \quad (4.2)$$

For a current state $i(X_0 = i)$, the total of times-steps a node will remain in state s during T time-steps is defined as:

$$\sum_{t=1}^T P_{is}^t \quad (4.3)$$

Then, if a node is in state i and E_s is the energy dissipated during one time-step in state s , thus the expected amount of energy the node will spend in the next T time-step

is expressed in the equation (4.4):

$$E^T = \sum_{s=1}^M \left(\sum_{t=1}^T P_{is}^t \right) E_s \quad (4.4)$$

In a system with M states, the transition probabilities among all states are represented by a $M \times M$ matrix. Thus, to forecast the energy consumption following the equation 4.4, the probability P is substituted by the probability matrix. The energy model is represented in Equation 4.5, where E_T is the total energy consumption in a certain period, i represents the states of the node, M is the total of states considered, t_i is the time the node spent in the state i , V_s is the operating voltage, and I_i is the average current consumption in the state i . The four states used in this model are: transmitting, listening, processing, and low power mode.

$$E_T = \sum_{i=1}^M t_i V_s I_i \quad (4.5)$$

4.2.2 Implementation and simulation

To predict the energy consumption, both models follow four steps: transitions matrix construction, probability matrix calculation, energy consumption forecasting, and error calculation. The node behavior can change over time, thus, those steps are periodically running. First, the node monitors its behavior for a certain time to construct the transition matrix. Second, using the transition matrix and the time remained on each state, it calculates the probability matrix. Then, using the probability matrix it forecasts the consumption. Finally, it calculates the error and goes back to the first step. This error can be used to decide if it is necessary to update the consumption rate, comparing it with a threshold. When the error is higher than the threshold, the consumption rate is updated. The entire prediction method is shown in Figure 8.

Figure 8: Energy consumption prediction model algorithm

Source: Author

The energy map was implemented and tested using IT-SDN (ALVES et al., 2017). The implementation comprises two different schemes: Scheme 1, which is based on Mini et al. (2005) method; and Scheme 2, which is the model and method proposed. In Scheme 1, each node processes its own prediction algorithm and sends its consumption rate and remaining energy to the controller. The controller constructs the energy map and uses the consumption rate to update it periodically. In Scheme 2, each node constructs the transitions matrix, calculates the total transitions for each state, and sends them and its remaining energy to the controller. With this information, the controller executes the forecasting algorithm and calculates the consumption rate of each node. Then, it constructs the energy map and uses the consumption rate to do periodical updates. For both cases, each node is monitoring its behavior. When detecting an important change on it, the node sends an update to the controller, if necessary.

Both schemes were simulated in COOJA (ÖSTERLIND et al., 2006), emulating TelosB motes (MEMSIC Inc., 2003), and using a topology with six nodes: one controller, one sink, and four sensor nodes. The topology implemented is represented in Figure 9. The node 1 is the controller, node 2 is the sink, and node 3, 4, 5, 6 are the sensor nodes. The arrows represent the communication links established. Other simulation parameters are shown in Table 4.

Figure 9: Simulation's topology

Table 4: Forecasting simulation parameters

Parameter	Value
Forecasting	
Time-step	10 ms
Forecasting time	100 time-step
Error threshold	5%
Energy Consumption (PRAYATI et al., 2010)	
Processing (4 MHz)	2,33 mA
Low power mode	0,180 mA
Transmitting	21,7 mA
Listening	22,8 mA

Source: Author

4.2.3 Results

The performance of both schemes were compared using two metrics: memory space and energy consumption. The energy consumption is based in the processing time and radio usage. The memory space results are shown in Table 5. Since the TelosB mote has 48 kB of read-only memory (ROM) and 10 kB of RAM, the IT-SDN code takes up the 79% of the total ROM and the 82,2% of the total RAM. The remaining 21% and 17,8% are available for applications, including the prediction code. Using the Scheme 2 implementation, it saves 12% of ROM that can be used for other applications.

The processing time measurements are shown in Figure 10. These results show a reduction when using the Scheme 2. The reduction was around 8% in all the sensor nodes and for both transition matrix update periods. The radio usage time measure-

Table 5: Code size on the sensor node including IT-SDN for both schemes

Scheme	ROM	RAM	Total
Memory usage without prediction model	37944	8222	46358
Memory usage for Scheme 1	40556	8474	49246
Memory usage for Scheme 2	39336	8484	48036

Source: Author

ments are shown in Figure 11. In the radio usage case, it increased its energy consumption in all nodes when using the Scheme 2. This result was expected due to the great difference in the message payload size. While Scheme 1 sends its remaining energy and consumption rate, Scheme 2 sends the total transitions of each state and the transitions matrix. Nodes 3 and 5 were the ones with higher increase on the energy consumption, since they have to forward messages from node 6 to the controller.

Figure 10: Processing state energy consumption

(a) Energy consumption using 10 minutes prediction periods

(b) Energy consumption using 20 minutes prediction periods

Source: Author

Figure 11: Radio module energy consumption

(a) Energy consumption using 10 minutes prediction periods

(b) Energy consumption using 20 minutes prediction periods

Source: Author

Table 6 shows the energy balance of both schemes for two different configurations. One configuration updates the transitions matrix every 10 minutes and the other every 20 minutes. The information on *Processing* and *Radio* columns shows the difference of energy consumption of Scheme 1 and Scheme 2. When the number is positive, it means Scheme 2 had a better performance. When the number is negative, it means Scheme 1 had a better performance. Finally, the energy consumption balance reveals a small improvement in the sensor node performance when using the Scheme 2.

This work is the first implementation using IT-SDN and an energy consumption prediction model. Results show that is possible to reduce processing and reduce memory space in sensor nodes, but this means an increase in the communication usage. As communication is the most energy expensive module, the trade-off will depend on the processing overhead and code size.

Table 6: Energy consumption balance

Node ID	10 minutes update		Total (mJ)	20 minutes update		Total (mJ)
	Processing (mJ)	Radio (mJ)		Processing (mJ)	Radio (mJ)	
3	22,71	-10,89	11,82	32,74	-28,02	4,72
4	13,70	-12,19	1,51	32,56	-24,57	7,99
5	21	-19,93	1,07	41,26	-31,07	10,19
6	18,5	-1,04	17,46	34,09	-28,49	5,6

Source: Author

4.3 Summary

The main contributions of these preliminary works are: the study of energy consumption in WSN and the implementation of a Markov chain approach to forecast the energy consumption in a WSN.

The energy management tool work (NUNEZ; MARGI, 2016) was the first experience with energy models and operating systems in WSN. Also, it was useful to experiment with Energest (DUNKELS; OSTERLIND; TSIFTES, 2007), a Contiki's tool to measure the energy consumption in WSN. The energy model used in this work considers four states: transmitting, listening, processing, and low power mode. Results show the tool can work together with ContikiMAC, in order to reduce the energy consumption when using the Rime stack. When using RPL it did not obtain good results.

The work explained in Section 4.2 proposes a method to construct an energy map in an SDWSN using a Markov chain approach to predict the energy consumption. It was implemented using IT-SDN (ALVES et al., 2017) framework. Results show that is possible to reduce the processing overhead in the node when running the Markov chain in the controller, instead of running it on each node. On the other hand, this approach could increase the communication module usage.

5 ENERGY CONSUMPTION AND PREDICTION MODEL

This chapter presents the details of the energy consumption and prediction model design. First, Section 5.1 discusses the shortcomings and lacks of previous works models and how this work attempts to address them, then it explains the design decisions and presents the energy consumption model proposal. Section 5.2 explains the energy consumption prediction using Markov chain approach. The mathematics and terms used in this section are explained in Section 2.2.1. Finally, Section 5.3 explains the energy consumption prediction using time series approach.

5.1 Energy consumption model proposal

The analysis of energy models in Section 3 brings relevant information. First, the energy consumption in WSN can be divided in three domains: sensing, processing, and communication. Each domain has states and the transition between states also consumes energy. Second, Zhou et al. (2011) and Ahmad et al. (2016) states that neglecting the the processing and sensing domains has an impact in the energy consumption estimation around 15%. Third, Haas, Wilke and Stöhr (2012) argue that the consumption by states transitions can be significant in the final result. Thus, in order to design an accurate energy consumption model, it should be based on the next three main points. First, the WSN device energy consumption is divided in three groups or consumption domains: sensing, processing, and communication. Second, each domain has states and will be considered the consumption of each state and the consumption by the transitions between them. Equation 5.1 expresses the consumption of each domain where J is the number of states of the domain i , and K is the number of possible transitions in the domain i . Finally, there is energy consumption by transitions between states of different domains that will be considered as one more addend in the total energy consumption equation. Regarding the previous ideas, the total energy

consumption can be modeled as shown in Equation 5.2 where: E_{CPU} is the consumption by the processing domain, E_{Sen} is the consumption by the sensing domain, E_{Com} is the consumption by the communication domain, and $Edtran$ is the consumption by transitions between domains for the M different transitions.

$$E_{i-domain} = \sum_{j=1}^J State_j + \sum_{k=1}^K Transitions_k \quad (5.1)$$

$$E_{total} = E_{CPU} + E_{Sen} + E_{Com} + \sum_{m=1}^M Edtran_m \quad (5.2)$$

Next sections detail the design decisions and other considerations about each consumption domain.

5.1.1 Sensing domain

In Section 3.1.1 it was mentioned that the sensing module front-end is composed by five submodules. As shown in Figure 12, the first submodule is the transducer, the second and third are signal aconditioners, the fourth is the ADC, and the fifth is the digital signal processor. There are sensor modules that are integrated with an ADC, but in other cases they just bring an aconditioned analog signal output. In the first case, the communication with the MCU is through a digital input/output (I/O). In the second case, the MCU uses one of its ADC or an external ADC, if the MCU has not one. One example of those two cases can be found in the TelosB mote (MEMSIC Inc., 2003). The TelosB has a dual humidity-temperature sensor (Sensirion SHT11), which has an ADC integrated and communicates with the MCU through two digital I/O pins. Then, the TelosB also has a visible light sensor (Hamamatsu S1087) and a visible to infrared sensor (Hamamatsu S1087-01), which are connected to the MCU ADC.

Figure 12: Sensing module front-end

Source: Adapted from Calhoun et al. (2005)

Another issue to consider is the standby or sleep consumption. Typically this consumption is very low and oftenly neglected. For example, the Sensirion SHT11 sensor has a typical and maximum sleep current consumption of $0,3\mu A$ and $1,5\mu A$ respectively. Another example is the DS18B20 temperature sensor from MAXIM¹ which has a typical and maximum standby current consumption of $0,75\mu A$ and $1\mu A$.

Considering the details discussed before, the sensing module energy consumption, for one sensor, can be calculated as shown in Equation 5.3. The term E_{ADC} can be omitted if the sensor module has an ADC integrated. Then, $E_{sen-act}$ is the energy consumption by the *active* state, $E_{sen-sleep}$ is the consumption by the *sleep*, and E_{on} and E_{off} are the energy consumptions by turning on and turning off the sensor.

$$E_{Sen} = E_{sen-act} + E_{sen-on} + E_{sen-off} + E_{sen-sleep} + E_{ADC} \quad (5.3)$$

To calculate the energy consumption for a certain period of time, the terms $E_{sen-act}$, $E_{sen-sleep}$ and E_{ADC} of Equation 5.3 can be expressed as the average power consumption multiply by the time of usage. Then, counting the times the sensor is turned on and turned off in that period, and knowing the average energy consumption of one transition, it is possible to calculate E_{sen-on} and $E_{sen-off}$. Thus, E_{Sen} can be expressed as shown in equation 5.4.

$$E_{Sen} = P_{sen-act}t_{sen-act} + P_{sen-sleep}t_{sen-sleep} + P_{ADC}t_{ADC} + N_{on}e_{sen-on} + N_{off}e_{sen-off} \quad (5.4)$$

5.1.2 Processing domain

The processing domain consumption model proposal is based on Zhou et al. (2011) work, explained in 3.1.2, but considering the possibility of more states. Zhou et al. (2011) model consider three states: active, idle, and sleep. Thus, the sleep state is changed for *power-saving* state, and it will depend on the MCU used in the mote. Some examples are the TelosB and WiSense, that have an MSP430 family MCU (TEXAS INSTRUMENTS, 2011) with five power-saving modes, including the idle state. Another examples are Mica2 and Iris motes, which have an Atmel family MCU, ATmega128L (ATMEL, 2011) and ATmega1280 (ATMEL, 2014) respectively. Both Atmel MCUs

¹ <<https://www.maximintegrated.com/en.html>>

have six power-saving modes, including the idle state. With this modification, the processing domain consumption can be calculated as shown in Equation 5.5.

$$E_{CPU} = E_{cpu-act} + E_{cpu-idle} + \sum_{s=1}^S Eps_s + \sum_{g=1}^G ECPUtran_g \quad (5.5)$$

The term $E_{cpu-act}$ is the energy consumption by the active state, which means, the CPU is active and processing. The term $E_{cpu-idle}$ is the energy consumption by the idle state. In this state the CPU is enabled but not active. Then, the term Eps_s is the energy consumption for the *power-saving* mode m , for $m \in [1, S]$, and S is the number of power-saving modes. Finally, the term $ECPUtran_n$ is the energy consumption by the state transition n , for $g \in [1, G]$, where G is the number of possible transitions. N is calculated using Equation 5.6, where P is the total of possible states.

$$G = P(P - 1) = (S + 2)(S + 1) \quad (5.6)$$

To calculate the energy consumption for a certain period of time, the terms $E_{cpu-act}$, $E_{cpu-idle}$, and Eps_s of Equation 5.5 can be expressed as the average power consumption multiplied by time of usage. Then, counting all transitions in the same period, and knowing the average energy consumption of each transition, it is possible to calculate the $ECPUtran_g$ sum. Thus, E_{CPU} can be expressed as shown in equation 5.7.

$$E_{CPU} = P_{cpu-act}t_{cpu-act} + P_{cpu-idle}t_{cpu-idle} + \sum_{s=1}^S tps_s Pps_s + \sum_{g=1}^G N_g PCPUtran_g \quad (5.7)$$

5.1.3 Communication domain

There are two approaches that are commonly used to calculate the energy consumption in the communication module. One approach is modeling the energy consumption per bit transmitted and received (ABO-ZAHHAD et al., 2015; AHMAD et al., 2016). The other approach is establishing operation states and measuring the time spent on each state (WANG; YANG, 2007; ZHOU et al., 2011). This work will follow the second approach because it is aligned with the methodology used in the two previous consumption domains, and also it is more suitable for the Markov chain prediction method.

Figure 13: Simplified block diagram of the communication module

Source: Author

Figure 13 is a simplified block diagram of a typical wireless communication module. Following Figure 13, are proposed four states to model the communication energy consumption: *transmitting*, *receiving* or *listening*, *idle*, and *sleeping*. In the *transmitting* state, only the transmitter circuitry and the control logic block are active. An important detail is that several communication modules brings the option of adjusting the transmitting power. The transmitting power options are commonly specified in decibel-milliwatt (dBm) in the datasheets, thus the transmitting energy consumption will be in function of dBm. In the *listening* state, only the receiver circuitry and the control logic block are active. In the *idle* state, the receiver and transmitter circuitry are turned off and the control logic block is turned on and can receive instructions. Finally, in the *sleeping* state, both transmitter and receiver circuitry are turned off and the control logic block is sleeping. Thus, the energy consumption by the communication domain can be calculated using Equation 5.8. The terms $E_{TX}(dBm)$, E_{RX} , $E_{Com-idle}$, $E_{Com-sleep}$, and $E_{Comtran_q}$ are the energy consumption by the *transmitting* state, *listening* state, *idle* state, and *sleeping* state respectively. This communication domain model is based on the characteristics of four IEEE 802.14 compliant radio modules: CC2420 (TEXAS INSTRUMENTS, 2017), CC2520 (TEXAS INSTRUMENTS, 2007), MRF24J40 (MICROCHIP, 2010), and CC1120 (TEXAS INSTRUMENTS, 2015).

$$E_{Com} = E_{TX}(dBm) + E_{RX} + E_{Com-idle} + E_{Com-sleep} + \sum_{q=1}^Q E_{Comtran_q} \quad (5.8)$$

Then, substituting each energy term in Equation 5.8 by the average power and the time usage, the energy consumed by the communication domain is shown in Equation

5.9

$$E_{Com} = P_{TX}(dBm)t_{TX} + P_{RX}t_{RX} + P_{Com-idle}t_{Com-idle} + P_{Com-sleep}t_{Com-sleep} + \sum_{q=1}^Q N_q P_{Comtran_q} \quad (5.9)$$

5.1.4 Transitions between domains

The processing domain is formed by a microcontroller, which has a microprocessor responsible to control all the components in the sensor node (ZHENG; JAMALIPOUR, 2009). In this manner, the components that form the sensing and communication domain are peripherals of the microcontroller. Thus, related to the communication between domains, the processing domain works as a master and the sensing and communication domain are slaves (RUSSELL, 2010). This means that there is not direct communication between the sensing and communication domain, and as a consequence, there are not transitions between both domains states. Figure 14 depicts the possible domain transitions.

Figure 14: Domain transitions behavior

Source: Author

According to Figure 14, the energy consumption by domain transitions can be calculated as shown in Equation 5.10. The terms $E_{CPU-Com}$, $E_{Com-CPU}$, $E_{CPU-Sens}$ and $E_{Sens-CPU}$ are the energy consumption by the transitions between: processing to communication, communication to processing, processing to sensing, and sensing to processing, respectively.

$$E_{DT} = E_{CPU-Com} + E_{Com-CPU} + E_{CPU-Sens} + E_{Sens-CPU} \quad (5.10)$$

To calculate the energy consumption for a certain period, it is necessary to count the number of transitions between domains in this period. Then, the number of transitions of each type of domain-transition is multiplied by the average domain-transition

consumption. Finally, the energy consumption is calculated as shown in Equation 5.11. $N_{CPU-Com}$, $N_{Com-CPU}$, $N_{CPU-Sens}$, and $N_{Sens-CPU}$ are the number of transitions from processing to communication, communication to processing, processing to sensing, and sensing to processing, respectively. $E_{CPU-Com}$, $E_{Com-CPU}$, $E_{CPU-Sens}$, and $E_{Sens-CPU}$ are the average energy consumption from processing to communication, communication to processing, processing to sensing and sensing to processing, respectively.

$$E_{DT} = N_{CPU-Com}E_{CPU-Com} + N_{Com-CPU}E_{Com-CPU} + N_{CPU-Sens}E_{CPU-Sens} + N_{Sens-CPU}E_{Sens-CPU} \quad (5.11)$$

5.1.5 Model considerations

From the analysis conducted in Section 3, it was identified the main elements considered in the literature to model the energy consumption, but also it was identified some details neglected. The model proposed, and explained along the Section 5.1, intends to improve the energy consumption estimation accuracy by including the aspects neglected in previous works, such as: consumption by transitions between energy domains and different low power modes in the processing domain.

To calculate the whole energy consumption for a sensor node, it is necessary to substitute equations 5.4, 5.7, 5.9 and 5.11 in Equation 5.2. After doing the substitution, the equation obtained is showing next:

$$\begin{aligned} E_{total} = & P_{sen-act}t_{sen-act} + P_{sen-sleep}t_{sen-sleep} + P_{ADC}t_{ADC} + N_{on}e_{sen-on} + N_{off}e_{sen-off} \\ & + P_{cpu-act}t_{cpu-act} + P_{cpu-idle}t_{cpu-idle} + \sum_{s=1}^S tps_s Pps_s + \sum_{g=1}^G N_g PCPUtran_g \\ & + P_{TX}(dBm)t_{TX} + P_{RX}t_{RX} + P_{Com-idle}t_{Com-idle} + P_{Com-sleep}t_{Com-sleep} + \sum_{q=1}^Q N_q PComtran_q \\ & + N_{CPU-Com}E_{CPU-Com} + N_{Com-CPU}E_{Com-CPU} + N_{CPU-Sens}E_{CPU-Sens} + N_{Sens-CPU}E_{Sens-CPU} \end{aligned} \quad (5.12)$$

5.2 Energy consumption prediction with Markov chain

In Section 5.1, the energy consumption was modeled as the sum of the consumption of three domains and the cost of transitions between domains. This approach can give the impression that the sensor node is constantly jumping among all the states,

being in only one state at the same time. But, what really happens is that the node state is a combination of three states, one state of each domain.

To predict the energy consumption with the Markov chain approach, based on the energy model proposed, the first step is to determine which are the combinations of states that represents typical sensor node behavior. Then, each combination is defined as a *sensor node state*, and the energy consumption of each *sensor node state* is calculated as the sum of the consumption of the three states that constitutes it. In the same way, the consumption by transitions between *sensor node states* is calculated as the sum of the consumption of the internal transitions involved. To better explain the state transitions consumption, Figure 15 shows an example of a *sensor node state* transition. In this example, the consumption by the transition from *sensor node state 1* to *sensor node state 2*, is equal to the sum of the consumption of the next transitions: CPU_{idle} to CPU_{active} and Com_{TX} to Com_{sleep} . For this specific example, the sensing domain does not contribute in the energy consumption transition because it remains in sleep mode.

Figure 15: *Sensor node state* transition example

Source: Author

The states possible combinations depend on different factors, for example: the operating system, which may have specific operation or energy aware politics; and the applications programmed in the sensor node, which may implement or may require specific functions. For this reason, to explain the prediction process, we supposed there are *sensor node states*: Sensing, Processing, Transmitting, Receiving, and Low-power mode. The configuration of each *sensor node states* is shown in Table 7, where in the first column are the *sensor node states* and in the first row are the energy consumption domains.

Once the *sensor node states* are defined, the next step is to construct the probability

Table 7: *Sensor node states* configuration

	Sensing	Processing	Communication
Sensing	active	active	sleeping
Processing	sleeping	active	sleeping
Transmitting	sleeping	active	transmitting
Receiving	sleeping	active	receiving
Low-power mode	sleeping	sleeping	sleeping

Source: Author

matrix. The probability matrix depends on the sensor node's behavior, and also it can vary over time. Before constructing the probability matrix, it is necessary to run a training period. In this period, the sensor node monitors and counts the transitions between all states to create a transitions matrix and a vector with the total transitions of each state. This matrix shows how many times the node remains in the same state after a time-step, and how many times it passes to another state. Equation 5.13 shows the transitions matrix A for five *sensor node states*, enumerated from 1 to 5, where: a_{11} is the number of transitions the node went from state 1 to state 1, a_{12} is the number of transitions the node went from state 1 to state 2 and so on. Equation 5.14 and Equation 5.15 show the total transitions vector V and how to calculate it.

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} \end{pmatrix} \quad (5.13)$$

$$V = \begin{pmatrix} v_1 & v_2 & v_3 & v_4 & v_5 \end{pmatrix} \quad (5.14)$$

$$\begin{aligned} v_1 &= a_{11} + a_{12} + a_{13} + a_{14} + a_{15} \\ v_2 &= a_{21} + a_{22} + a_{23} + a_{24} + a_{25} \\ v_3 &= a_{31} + a_{32} + a_{33} + a_{34} + a_{35} \\ v_4 &= a_{41} + a_{42} + a_{43} + a_{44} + a_{45} \\ v_5 &= a_{51} + a_{52} + a_{53} + a_{54} + a_{55} \end{aligned} \quad (5.15)$$

Then, the probability matrix P is calculated using the transitions matrix and the

total transitions vector, as shown in Equation 5.16.

$$P = \begin{pmatrix} a_{11}/v_1 & a_{12}/v_1 & a_{13}/v_1 & a_{14}/v_1 & a_{15}/v_1 \\ a_{21}/v_2 & a_{22}/v_2 & a_{23}/v_2 & a_{24}/v_2 & a_{25}/v_2 \\ a_{31}/v_3 & a_{32}/v_3 & a_{33}/v_3 & a_{34}/v_3 & a_{35}/v_3 \\ a_{41}/v_4 & a_{42}/v_4 & a_{43}/v_4 & a_{44}/v_4 & a_{45}/v_4 \\ a_{51}/v_5 & a_{52}/v_5 & a_{53}/v_5 & a_{54}/v_5 & a_{55}/v_5 \end{pmatrix} \quad (5.16)$$

The energy consumption prediction using Markov chain approach will be separated in two terms: E_R is the energy consumption prediction for the usage time in all the energy domains, and E_B is the energy consumption prediction for the transitions. In this way, the energy consumption prediction E_{MC} is shown in Equation 5.17.

$$E_{MC} = E_R + E_B \quad (5.17)$$

To calculate E_R , there are calculated the number of visits to each *sensor node state* using Equation 5.18, explained in Section 2.2.1. As explained in Section 5.1.4, the processing module has a role of master, thus, it is defined an initial distribution $x_0 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \end{pmatrix}$. Then, the number of visits to each state is multiplied by its average energy consumption. In this manner, the energy consumption E_R for the five *sensor node states* shown in Table 7 can be calculated as shown in Equation 5.18. The term E_s is a vector of the average energy consumption of the *sensor node states*, and T is the prediction time in time-steps.

$$E_R = x_0 \left(\sum_{t=1}^T P^t \right) E_s \quad (5.18)$$

To calculate E_B , a transition cost matrix is defined. This matrix is similar to the probability matrix, but showing the energy cost of passing from one *sensor node state* to another. A transition cost matrix B for five states is shown in Equation 5.19, where e_{12} is the energy consumption of going from *sensor node states* 1 to *sensor node states* 2, e_{13} is the energy consumption of going from *sensor node states* 1 to *sensor node*

states 3 and so on.

$$B = \begin{pmatrix} e_{11} & e_{12} & e_{13} & e_{14} & e_{15} \\ e_{21} & e_{22} & e_{23} & e_{24} & e_{25} \\ e_{31} & e_{32} & e_{33} & e_{34} & e_{35} \\ e_{41} & e_{42} & e_{43} & e_{44} & e_{45} \\ e_{51} & e_{52} & e_{53} & e_{54} & e_{55} \end{pmatrix} \quad (5.19)$$

Then, the number of visits to each *sensor node state* are calculated and multiplied by the vector K_s , which represents the transitions cost. The vector K_s is the result of the element-wise product of the probability matrix and the transition cost matrix, x_0 is the initial distribution vector, and e is a column vector with all entries 1. Thus, E_B can be calculated as shown in Equation 5.21.

$$K_s = P \circ B \quad (5.20)$$

$$E_B = x_0 \left(\sum_{t=1}^T P^t \right) K_s e \quad (5.21)$$

5.3 Energy consumption prediction with time series

In this section a proposal to forecast the energy consumption in WSN nodes using time series analysis is explained. Different than the Markov chain forecasting method presented before, which is based on the probability of state transitions in the node, the time series analysis forecasting method is based on the behavior of a variable observed. The time series to be analyzed are formed by observations of sensor nodes' energy consumption, and the energy consumption is calculated using the model proposed in Section 5.1

As explained in Section 3.2.2, time series can be stationary and nonstationary. When a time series is stationary, its mean and covariance are constant overtime. Two behaviors that could affect the stationarity are trending and seasonality. Both behaviors may appear in the energy consumption data of WSN nodes in typical WSN scenarios. For example, when a new node is included in the network it could affect the energy consumption mean of its neighbors. If new nodes are periodically included, it could set

a trend in the energy consumption. Another typical behavior is the periodical messages exchange, for example, when sensor nodes are programmed to periodically send new data to the sink. This behavior could increment the energy consumption mean for a certain period of time. To illustrate those behaviors, both situations were simulated in a SDWSN, using a nine nodes grid topology with one sink node, one SDN controller, and seven nodes generating traffic. The network nodes distribution is shown in Figure 16.

Figure 16: Seasonality and trending simulation topology

Figures 17 and 18 show the node 3 and node 5 energy consumption, and its moving average, when the sensor nodes are programmed to periodically send information to the sink. The results show the seasonality caused by the periodical messages exchange.

Then, Figures 19 and 20 show the node 3 and node 5 energy consumption, and its moving average, when including new nodes in the network. It was included four new nodes, located as node 3 neighbors. The results show the trending effect in the energy consumption of both node 3 and node 5. Since node 5 routes the node 3 data packets, and node 3 routes the new nodes data packets, both nodes energy consumption is strongly affected.

The basis of forecasting a time series is the modelling. The time series modelling choosen for this work was ARIMA models, which was explained in Section 2.2.2.4. The reasons to choose ARIMA were:

- it allows to work with nonstationary time series;

Figure 17: Node 3 energy consumption for periodical messages exchange

Source: Author

- it enables to model trending and seasonal behaviors;
- and there are several works that successfully implemented an ARIMA prediction model in a WSN, despite the WSN limitations.

Also, there are special cases of ARIMA models that are equivalent to models of other methods (DIAS, 2016), for example:

- ARIMA(0,1,0) is equivalent to the *Constant* method;
- ARIMA(0,2,0) is equivalent to the *Linear* method
- ARIMA(0,1,1), ARIMA(0,2,2), ARIMA(0,1,2), and ARIMA(1,1,2) are equivalent to some models of the *Exponential Smoothing* method.

In this manner, using ARIMA there are also considered other prediction methods.

In this work there are considered scenarios with the number of neighbors fixed. This means that there are not included new nodes in the network during the testing time. This restriction reduces the trending components in the time series.

Figure 18: Node 5 energy consumption when including new nodes in the network

Source: Author

Figure 19: Node 3 energy consumption for periodical messages exchange

Source: Author

Figure 20: Node 5 energy consumption when including new nodes in the network

Source: Author

6 PROPOSAL EVALUATION

In order to evaluate the performance of the energy consumption and prediction models proposed in Chapter 5, different experiments were conducted, divided in three groups: energy consumption model, Markov chain, and time series. For each group, there are two subgroups: simulations and real hardware experiments. The objective of testing the models through simulations and real hardware experiments is to compare the results and argue about the reliability of the models.

This chapter explains the details of the implementation of the models for the simulation and real testbed experiments. Also, it presents the results of those experiments. This chapter is organized as follow. Section 6.1 explains general aspects of the implementation of the three groups of experiments. Section 6.2 explains the energy consumption model implementation, experiments conducted and its results. Section 6.3 explains the Markov chain implementation, experiments and results. Then, Section 6.4 explains time series implementation, experiments and results. Finally, Section 6.5 summarize the main results.

6.1 General aspects

One specific objective of this work is the implementation of the energy consumption and prediction model in an SDWSN. To attain this objective, the whole implementation is conducted using IT-SDN, explained in Section 2.1.5.1. The current implementation of IT-SDN works upon Contiki 3.0¹, thus, it is the operating system used to program each node of the network.

To identify the nodes in the network, the next classification is used: sensor node, sink node, and controller node. Sensor nodes are the nodes which generates data traffic. The sink node is the destiny of data traffic. The controller node is the node in charge of

¹ <<https://github.com/contiki-os/contiki/releases/tag/3.0>>

control tasks and also used to run energy consumption prediction algorithms. The simulations were conducted using COOJA (ÖSTERLIND et al., 2006). The sensor nodes and sink nodes are emulated as sky motes, which is the equivalent of TelosB mote. The controller software is running outside of the COOJA environment but connected via TCP with a sensor node of the network.

The experiments with real hardware were conducted using TelosB motes for both sensor nodes and sink nodes. For the controller node, the controller software were running in a laptop computer (Core i7, 2,40 GHz), using a sensor node to communicate with the wireless network. The communication between the computer and the sensor node is through serial connection (USB) as shown in Figure 21.

Figure 21: Controller node

Source: Author

Lastly, network's characteristics and configurations for all experiments are:

- Topology: grid topology of 12, 25, 36 and 49 nodes, one controller and one sink in all topologies.
- Radio duty cycle mechanism: ContikiMAC, with a 16 Hz waking-up frequency.
- Transmitting power output: 0 dBm

6.2 Energy consumption model

In this section an equation to calculate the energy consumption of a TelosB mote is established, following the model proposed in Section 5.1. Then, it is explained the code implementation to enable the node to calculate its energy consumption. Finally, the results of the implementation are presented.

6.2.1 TelosB modeling

The first step to establish the energy consumption equation is to analyze the hardware chosen. This analysis is divided by domains in the next order: sensing, processing, and communication. The Telosb mote has four sensors: visible light, visible to infrared, humidity, and temperature. Then, the application programmed was setted to sense only temperature, thus the analysis ignores the other three sensors. The temperature sensor in the TelosB is the Sensirion STH11. This sensor has an ADC embedded, which means that the ADC consumption is included in the *active* consumption. Also, the sensor specifies one sleep mode. With this information and following the model explained in 5.1.1, the energy consumption for the sensing domain is calculated as shown in equation 6.1.

$$E_{sentelos} = E_{sen-act} + E_{sen-on} + E_{sen-off} + E_{sen-sleep} \quad (6.1)$$

In the processing domain, the microcontroller characteristics were analyzed. The TelosB has an MSP430F1611 microcontroller, which has one active mode and five power-saving modes. On each mode there are different components that are enabled and disabled. Those componentes are: the CPU, the main clock (MCLK), the sub-main clock (SMCLK), the auxiliary clock (ACLK), the digitally controlled oscillator (DCO), and the crystal oscillator (CO). The configurations of all modes are summarized in Table 8 (TEXAS INSTRUMENTS, 2011).

Table 8: MSP430F1611's microcontroller operation modes

Mode	CPU	MCLK	SMCLK	ACLK	DCO	CO
Active	X	X	X	X	X	X
LPM0			X	X	X	X
LPM1			X	X	* 2	X
LPM2				X	X	X
LPM3				X		X
LPM4						

Source: Adapted from Texas Instruments (2011)

Then, the sky mote main method in Contiki was programmed to put the microcontroller in power-saving mode whenever there is no processing to be done. The algorithm first checks if the DCO needs to be on. In that case, the microcontroller en-

²The DCO's DC generator is disabled if it is not used in active mode

ters in LPM1. Otherwise, it enters in LPM3. In this manner, there are considered three states for the processing domain: *Active*, LPM1, and LPM3. The energy consumption can be calculated as shown in Equation 6.2.

$$E_{cputelos} = E_{cpu-act} + E_{lpm1} + E_{lpm3} + \sum_{g=1}^5 E_{CPUtran_g} \quad (6.2)$$

The TelosB has a CC2420 communication module. This module operation can be classified in five modes: *transmitting*, *receiving*, *idle*, *power down*, and *voltage regulator off*. Same as explained for the MSP430 microcontroller, there are different components that are enabled and disabled on each mode. The main components are: transmitter circuitry, receiver circuitry, frequency synthesizer, and voltage regulator (TEXAS INSTRUMENTS, 2017). The relation between modes and components is summarized in Table 9.

Table 9: CC2420's communication module operation modes

Mode	Transmitter circuitry	Receiver circuitry	Frequency synthesizer	Voltage regulator
Transmitting	X		X	X
Receiving		X	X	X
Idle			X	X
Power down				X
Voltage regulator off				

Source: Adapted from Texas Instruments (2017)

For this implementation, the communication module behavior is defined by ContikiMAC (DUNKELS, 2011), but also by the CC2420 Contiki's implementation. ContikiMAC is an RDC mechanism, which its main objective is to keep the transceiver turned off as long as possible to reduce the energy consumption. To attain this objective, ContikiMAC is constantly turning on the node's radio to check up if another node is sending a message to it. In that case, the node remains with the radio on, otherwise, it turns the radio off. An important issue is that using ContikiMAC, the communication module does not enter in *idle* state. Also, every time ContikiMAC turns off the radio, the frequency synthesizer is disabled. This coincides with the *power down* state specification. Thus, based on the characteristics described, the TelosB communication domain energy consumption using ContikiMAC can be calculated as shown in Equation 6.3. Since the *voltage regulator off* state is activated and deactivated using an

external pin, it is not considered for this work implementation.

$$E_{comtelos} = E_{TX}(dBm) + E_{RX} + E_{powerdown} + \sum_{q=1}^5 EComtran_q \quad (6.3)$$

Finally, using Equations 6.1, 6.2, and 6.3. TelosB total energy consumption can be calculated as shown in Equation 6.4.

$$E_{TelosB} = E_{sentelos} + E_{cputelos} + E_{comtelos} \quad (6.4)$$

6.2.2 Implementation

The model proposed in Section 5.1 established that the energy consumption of each state is calculated by the multiplication of the average power consumption and the time period the node remained in that state. In the case of state transitions, the average energy consumption of each transition is multiplied by the number of times the transition occurred in a certain period of time. This means that to calculate Equation 6.4, it is necessary to monitor the time spent on each state, the average power consumption of each state, the number of transitions, and the average energy consumption of each transition type.

The components' datasheets normally brings basic power consumption information, but it has been shown that real measurements can have significant differences with the datasheet information. For example, Prayati et al. (2010) compared the TelosB datasheet information with real measurements, reporting significant differences and bringing information not available in the datasheet. Thus, it was measured the power consumption of the TelosB running pre-defined routines and the equipment available in the laboratory.

The equipment employed for the measurements were: one power supply Agilent E3631A (AGILENT, 2002), one digital multimeter Agilent 34401A (AGILENT, 2007), and one computer running LabView (NATIONAL INSTRUMENTS, 2002). The power supply was configured to provide 3,00 V. The multimeter measures the current provided by the power supply and sends the information to the computer running LabView. Figure 22 shows a diagram of the equipment connections.

A constraint of using the Agilent 34401A is that its maximum sample frequency is

1 kHz, which is not enough to correctly sample some TelosB state transitions. For example, the CC2420 radio datasheet notify a maximum startup time of $192\mu s$. Another examples is the TelosB microcontroller which has a typical wakeup time of $1,5\mu s$. Since those transitions are much quicker than multimeter sample period, those were not considered for the model proposal evaluation. The only transitions considered were the on and off sensing domain transitions.

Figure 22: Measurement setup diagram

Source: Author

To carry out the measurements, a LabView script was used. This script allows to set two different sample frequencies: 55 Hz and 500 Hz. The first frequency allows to obtain accurate mean values and the second one allows to see the current consumption dynamic. Also, different routines were programmed in the TelosB. Those routines are explained next:

- Sensing routine: the TelosB senses the temperature every 15 seconds. Before each temperature measurement, the yellow LED is turned on for 2 seconds, then it waits 3 seconds and do the measurement. This routine was executed using both LPM1 and LPM3, and doing a basic processing task before and after the sensing measurement. From this routine it was taken the sensing state average current consumption and sensing transitions energy consumption.
- Processing routine 1: the TelosB runs a routine of matrix multiplications. The routine is repeated every five seconds, turning on the red LED between two routines and forcing the TelosB to enter in LPM3. From this routine it was taken the processing state and LPM3 average current consumption.

- Processing Routine 2: Same as routine processing 1 but forcing the TelosB to enter in LPM1. From this routine it was taken the LPM1 average current consumption.
- Communication routine 1: there is one node working as a sink and other node working as a sensor node. The sensor node sends a message to the sink every 10 seconds. Before sending the message, the red LED blinks. When the sink receives the messages, sink's green LED blinks. From this routine it was taken the transmitting state and receiving state average current consumption.
- Communication routine 2: Same as communication routine 1, but keeping the radio turned on. From this routine it was taken the listening state average current consumption.

The results of states average current consumption is shown in Table 10. Appendix A shows the TelosB energy consumption captured in the LabView software.

Table 10: States average current consumption

State	Average current consumption (mA)	Standard deviation (mA)
Sensing	2,5000	0,0271
Processing	2,2129	0,0041
LPM1	0,6782	0,0041
LPM3	0,5262	0,0236
Transmitting	19,7008	0,0662
Receiving	18,9060	0,2810
Listening	18,8801	0,0201

Source: Author

The sensing domain transitions average consumption is shown in Table 11.

Table 11: Sensing domain transitions average consumption

Sensing domain transitions	Average energy consumption (μJ)	Standard deviation (μJ)
On	14,90	0,15
Off	14,60	0,22

Source: Author

The results of current consumption tests and the routines programmed provided important information: The current consumption measured when the TelosB micro-

processor is in Low-Power and it is neither sensing nor communication, is much higher than reported in datasheets and by Prayati et al. (2010). The reason of this difference is because Contiki does not deactivate the radio module crystall oscillator, thus the current measured includes both the microprocessor's Low-Power mode consumption and the communication crystall oscillator consumption. Then, in normal operation the node does not enter into LPM3, just when the application forces the node to enter into LPM3. Also, the Listening and Receiving states average current consumption measured were nearly equal. The difference defined for both states is that Receiving state includes the processing consumption but measurements do not evidence this difference. Analyzing the states behavior, it was noted that when the node is receiving a message the microprocessor is in Low-Power mode most of time, just like Listening state. Then, instants before exit Receiving state, the microprocessor enters in processing state. This means that most of time both states energy consumption are equal.

Considering previous information, the seven states proposal can be reduced to five states. First, it was considered only LPM1 and LPM3 was excluded. Second, Listening and Receiving state were fused in one state called Listening state. Thus, the five states are: Sensing, Processing, LPM1, Transmitting, Listening.

Using the energy consumption model, an application for Contiki to calculate the energy consumption was implemented, considering the five states mentioned before and the sensing domain transitions. To monitor the time spent on each state it was used Energest (DUNKELS; OSTERLIND; TSIFTES, 2007), a Contiki's energy management tool. Energest defines generic states for sensing, processing, and communication. When the node enters in a certain state, it checks the system clock and saves the time reported. Then, when the node exits the state, it checks the system clock again and calculates the total time. To accomplish this, Energest has two main function-like macros: `ENERGEST_ON(type)` and `ENERGEST_OFF(type)`. `ENERGEST_ON(type)` checks the time when entering the state "type" state and `ENERGEST_OFF(type)` checks the time when exiting it. Energest brings the time remained on each state but it does not bring the information of states transitions. Due to this, a transitions matrix where every transition is registered was included. The transition between two states is identified using the Energest macros and two variables: *former_state* and *current_state*. Also, each state is identified with a number from 0 to $n - 1$, where n is the number of states. When the `ENERGEST_OFF(type)` macro is called up, the state number (the state from it just come out) is stored in the *former_state* variable. Then, when the `ENERGEST_`-

ON(type) macro is called up, the current state number is stored in the *current_state* variable. Next, one transition is added to the entry [*former_state*][*current_state*] of the transitions matrix .

To evaluate the energy consumption model accuracy it was tested in a real network composed by 6 nodes and using TelosB motes. The network was composed by one controller node, one sink node, and four sensor nodes. Each sensor node senses the temperature, sends the information to the sink, and then calculate the energy consumption and send its estimation to the controller. The estimation sent to the controller was compared with measurements made with the digital multimeter and the LabView application. The network deployed is shown in Figure 23. The red circles represent the sensor nodes, the green circle represents the sink node, and the blue circle represents the controller node.

Figure 23: Network deployed for performance measurements

Source: Author

The experiment was repeated six times, taking between ten and fifteen energy estimations per experiment. The main indicator used to measure the accuracy was the Mean Absolute Percentage Error (MAPE) which is defined in Equation 6.5 where n is the number of values (errors), A_t is the real energy consumption and F_t is the value estimated. The results obtained are shown in Table 12.

$$M = \frac{100}{n} \sum_{t=1}^n \left| \frac{A_t - F_t}{A_t} \right| \quad (6.5)$$

To measure the difference in the energy consumption estimation when considering state transitions, simulations varying the topology size were carried out. On each simulation, each node calculates the energy consumption by the time remaining on all states

Table 12: Energy consumption estimation accuracy

Indicator	Value
MAPE	6,60 %
Error variance	2,36 %
Error confidence interval (95 %)	4,13 % - 9,15 %

Source: Author

and by state transitions separately. Then, the percentage of transitions' energy consumption over the total energy consumption was calculated. Since it was not possible to measure real state transitions consumption, a theoretical energy consumption was calculated using the trapezoidal rule ³ as shown in Equation 6.6. Δt is the transition time, V_{op} is the operation voltage, I_1 is the current consumption before the transition, and I_2 is the current consumption after the transition. The transitions time was taken from the datasheet of each component: MSP430 microcontroller, CC2420 radio module, and the SHT11 sensor. The transitions energy consumption calculated were: from LPM1 to Processing, from LPM1 to Transmitting, from LPM1 to Listening, from Processing to Transmitting, and from Processing to Receiving. For the rest of the state transitions, datasheets do not bring the information required to calculate the energy consumption. The transitions cost matrix obtained is shown in Table 13.

$$E_{tran} = \frac{\Delta t}{2} V_{op} (I_1 + I_2) \quad (6.6)$$

Figure 24 shows the results simulating for 12, 25, 36 and 49 nodes. In all the cases, the energy consumption by state transitions is similar, with values between 1,09% and 1,12%. From these results can be concluded that energy consumption by transitions is low and it does not depend on the topology size.

Table 13: Transitions cost matrix in μJ

	Sensing	Processing	LPM1	Transmitting	Listening
Sensing	0	14,6	0	0	0
Processing	14,9	0	0	6,3	6,08
LPM1	0	0,0064	0	5,86	5,63
Transmitting	0	0	0	0	0
Listening	0	0	0	0	0

Source: Author

³ <https://ccrma.stanford.edu/~jos/pasp/Trapezoidal_Rule.html>

Figure 24: State transitions' energy consumption percentage

Source: Author

Another domain that is seldom discussed in previous work (Chapter 3) is the energy consumption of the sensing domain. In Figure 25 the percentage of the sensing domain energy consumption over the total energy consumption is shown. According to the results, the sensing domain usage represents between 0,19% and 0,05% of the total network consumption. In this case, this energy consumption percentage is within the error margin and can be considered as negligible. In other way, the sensing domain consumption depends on the network application, for example, it depends on the number of sensors used and the sensing periods. For those reasons, it can not be concluded that the sensing domain energy consumption can be neglected in all the cases.

6.3 Markov chain

Following the TelosB modeling proposed in Section 6.2.1, there are considered 5 states for the Markov chain modeling: Sensing, Processing, LPM1, Transmitting, and Listening. Table 14 shows each state configuration, where the first column represents the state and the first row represents the consumption domains.

The probability matrix is constructed running a training process and using En-ergest. This process runs during a certain period of time to create the transitions matrix

Figure 25: Sensing domain energy consumption percentage

Source: Author

Table 14: Markov chain states configuration for TelosB

	Sensing	Processing	Communication
Sensing	active	active	sleeping
Processing	sleeping	active	sleeping
LPM1	sleeping	LPM1	sleeping
Transmitting	sleeping	active	transmitting
Listening	sleeping	LPM1	receiving

Source: Author

and a vector with the total transitions of each state. Using the transitions matrix and the total transitions vector, it is calculated the probability matrix as explained in Section 5.2 and shown in Equation 5.16.

To construct the transitions matrix, a process similar to the one described in Section 6.2.2 using Energest was implemented. The difference is that in this case there were defined two vectors with three entries: *former_state_vector* and *current_state_vector*. Each entry represents one energy consumption domain. Thus, when the `ENERGEST_ON(type)` macro is executed, it compares both vectors to determine the two states involved in the transition. Then, the information is stored in the variables *former_state* and *current_state*, and finally one transition is added to the entry `[former_state][current_state]` of the transitions matrix. When the transitions matrix

construction is done, the total transitions vector is calculated using Equation 5.15. Finally, the probability matrix is calculated. At this point, the sensor node or the controller node have the necessary information to do the energy consumption prediction.

The time-step for this implementation was defined in 1 *ms* because it can follow the sensor node's dynamic and is manageable by the program implemented. Previous works used higher time-steps values, for example, Lin et al. (2007) and Mini et al. (2005) define the time-step as 0, 1s and 1s respectively. The problem of using those values in our work is that state transitions are faster thus it is not possible to take the nodes' behavior dynamic. The forecasting time was defined in 100.

The Markov chain prediction method was implemented in two different schemes. In the first scheme (sensor node scheme) each sensor node runs the prediction algorithm. In the second scheme (controller scheme), the controller receives information off all nodes in the network to predict the energy consumption of each of them. Also, both schemes were using two energy model configurations: including state transitions consumption and not including them.

The implementation details, experiments, and results of both schemes are shown in Section 6.3.1.

6.3.1 Markov chain schemes

In the sensor node scheme, each sensor node uses the energy consumption model explained in 6.2.2 to periodically estimate its energy consumption. Then, the sensor node runs the Markov chain algorithm to do a prediction of its energy consumption and then estimate a consumption rate. The sensor node monitors its energy consumption since it is turned on and the training process starts when the node already has a flow route to send data to the sink. In this way the training process is not polluted with the network's initial behavior. The training process runs during 5 sensing periods. Then the probability matrix is calculated and the energy consumption for the next 100 time-steps is predicted. Using the periodically energy consumption estimations, the energy remaining in the node is calculated; and using the energy consumption prediction an energy consumption rate is calculated. Then, this information is sent to the controller. Since the sensor node's behavior may change over time and the controller is not aware about it, the sensor node periodically compares its energy consumption and the energy consumption predicted. If the error is higher than a threshold defined, the sensor node

starts a new training process, calculates a new consumption rate, and sends the new energy consumption rate and the energy remaining to the controller.

In the controller scheme, each node periodically estimates its energy consumption and monitors the state transitions. When the period defined ends, the node sends the state transitions matrix and state's total time to the controller. The controller runs the Markov chain algorithm and sends the prediction to the sensor node. Then, the sensor node compares the prediction with the calculated energy consumption. If the error is below the defined threshold, the prediction is considered as valid. If the error is above the threshold, the node sends a new update to the controller to estimate a new prediction. For both schemes a 5% threshold was used.

Both schemes were simulated for 12, 25, 36, and 49 nodes. To simulate a real WSN behavior, a temperature monitoring application was programmed where each node senses the temperature and sends the information to the sink. The sensing period depends on the network size. For 12 nodes, each node senses the temperature every 10 seconds, for 25 nodes every 20 seconds, and for 36 nodes every 30 seconds. For 49 nodes, the controller was not able to handle the amount of packets received, bursting the packets buffer programmed for IT-SDN. It was attempted to increase the updating period to alleviate the controller's packet buffer, but for the case of the controller scheme this increase requires larger data types variables to store the information, surpassing the IEEE 802.15.4 message's maximum size. From those simulations the prediction accuracy and the energy consumption were measured. The prediction accuracy is represented by the prediction MAPE and the energy consumption is shown in milijoules.

Figure 26 shows the prediction MAPE (error percentage) of each topology size and both schemes. Then, it was analyzed the relation of the prediction error and the hops distance between the node and the sink. Figures 27, 28, 29, show the MAPE for all topology sizes and both schemes, sorted by number of hops to the sink.

From Figure 26 it is observed that the sensor node scheme has a lower average prediction error for 12 nodes, than the obtained for the controller scheme. For 25 and 36 nodes the controller scheme shows a better performance. On the other hand, the standard deviation for all the cases is too high, and it does not allows to conclude which scheme is better. Analyzing Figure 27, the node scheme has a similar average prediction error for all nodes in the network. In the case of the controller scheme,

Figure 26: Markov chain MAPE for sensor node scheme and controller scheme

Source: Author

the average prediction accuracy has a significative improve from one hop to two and three hops. Even that, it is not possible to guarantee which one is better due to the high standar deviation. What makes the node scheme to have a better accuracy for 12 nodes, is the 5% difference with the controller scheme for one hop distance. This behavior is repeated for 36 nodes network, as shown in Figure 29, but in this case the difference between schemes' prediction accuracy for one hop is smaller than the difference in the 12 nodes network.

The reason why the prediction accuracy becomes worse for nodes close to the sink is the energy consumption dynamic. Since all sensor nodes are periodically sending messages to the sink, the sensor nodes nearby the sink have higher and more variable energy consumption than the rest of the sensor nodes. This behavior creates a funnel effect in the energy consumption dynamic which produces the behavior observed in Figure 28 and Figure 29 for 25 and 36 nodes, where the prediction error progressively decreases.

Another reason that could be influecing the error prediction obtained is the time-step selection and the time-step forecasted. A time-step of $1ms$ was chosen because it is more close to real motes' and WSN's dynamic than time-steps used in previous works (LIN et al., 2007; MINI et al., 2005). Then, forecasting 100 time-steps will give the equivalent of forecasting $100ms$ which can not be enough time to estimate an accurate consumption rate. One option could be increasing the time-step, missing

Figure 27: Markov chain prediction accuracy for 12 nodes

Source: Author

part of mote's dynamic; or increasing the forecasting time, increasing the processing overhead in the node.

Another characteristic observed is the high standard deviation obtained in all simulations. The behavior observed during simulations and results analysis show that the energy consumption by processing the energy consumption prediction in the node (in the sensor node scheme) and sending the transitions matrix to the controller (in the controller scheme) is significant if compared with the node's average energy consumption per minute. This difference causes immediate increments in the energy consumption when is necessary a prediction update, generating the high variance in the prediction error. Figure 30 shows the energy consumption of one node from each distance, compared with the prediction made for this node. What this figure shows is that the energy consumption of the nodes close to the sink has bigger and faster variations than energy consumption of the nodes at three or four hops away from the sink. Also, it is possible to observe that prediction for nodes at three or four hops away from the sink has a better performance than prediction of the nodes at one hop from the sink.

To measure the impact of both schemes, the network's energy consumption was compared using and not using the Markov chain prediction method. In the case when not using the Markov chain method, the node informs its energy consumption to the controller every 60 seconds. In the case when using the Markov chain method, the error checking period is set to 60 seconds for both schemes. Figure 31 shows the

Figure 28: Markov chain prediction accuracy for 25 nodes

Source: Author

average energy consumption per minute, in milijoules, for topologies of 12, 25, and 36 nodes.

Results show that using the Markov chain prediction method, the energy consumption increases in all scenarios, which is the opposite to what is expected for an energy consumption prediction mechanism. What this means is that the prediction mechanism is not obtaining the enough accuracy to keep the prediction error below the threshold to reduce the radio usage by energy messages. Even that, a reduction in the energy consumption was observed for nodes farther from the sink. As it is depicted in Figure 32, the energy consumption for nodes at one hop from the sink is much higher when using the Markov chan method than when not using a prediction method. For the case of two and three hops from the sink, the energy consumption is almost the same. And finally, for nodes at four hops from the sink, the energy consumption when using the Markov chain method is lower than when not using the prediction method.

Table 15 gives more details about the energy consumption when using and not the Markov chain prediction mechanism. The table shows the increasing in processing and communication resources when using the prediction mechanism with regard to the scheme without prediction.

Table 15 shows that by using the controller scheme, the resources overhead mainly goes to the transmitting time. When using the sensor node scheme, the overhead is pre-

Figure 29: Markov chain prediction accuracy for 36 nodes

Source: Author

Table 15: Average increasing of sensor node resources

	Controller scheme			Sensor node scheme		
	12 nodes	25 nodes	36 nodes	12 nodes	25 nodes	36 nodes
Processing	15,42 %	63,18 %	29,84 %	89,85 %	169,62 %	115,80 %
Transmitting	110,87 %	127,75 %	107,26 %	82,01 %	87,61 %	61,80 %
Receiving	60,31 %	82,06 %	49,84 %	57,28 %	82,55 %	38,19 %

Source: Author

dominantly in the processing time. The receiving time has a similar increase for both schemes. This behavior was the expected for both schemes. The controller scheme concentrate the processing in the controller in exchange of sending long messages to the controller; and the sensor node scheme sends short messages to the controller in exchange of running the entire algorithm in the node. The energy consumption of the sensing module was not included in Table 15 because it does not depend on the prediction algorithm. In other words, the sensing applications is independent of the energy prediction application.

Finally, the energy consumption was compared for both Markov chain prediction schemes running in real devices. The network deployed is shown in Figure 23. The average node's energy consumption per minute and its standard deviation are shown in Table 16. The results show that the controller scheme has a better energy consumption performance than sensor node scheme. These results coincide with the simulation

Figure 30: Energy consumption and prediction for 25 nodes using Markov chain

Source: Author

results shown in Figure 31.

Table 16: Average energy consumption of a real devices network running the Markov chain prediction method

	Average energy consumption (mJ)	Standard deviation (mJ)
Sensor node scheme	245,86	18,33
Controller scheme	209,54	21,39

Source: Author

Summarizing the Markov chain section results, the algorithm implemented and both schemes simulated do not aid to reduce the energy consumption in a WSN where each node estimates and informs its energy consumption to a controller. The prediction accuracy obtained is not enough to reduce the communication and processing overhead when running the prediction algorithm. Comparing both schemes, the controller scheme has a better prediction accuracy and smaller energy consumption than the sensor node scheme. Also, it was observed that nodes close to the sink have worse prediction accuracy than nodes 3 or 4 hops away from the sink. One scenario that could be studied is to run the prediction algorithm only on nodes that are 3 or more hops away from the sink.

Figure 31: Energy consumption when using and not using the Markov chain prediction method

Source: Author

6.4 Time series

Similar to the Markov chain implementation and evaluation, two schemes were implemented to evaluate the time series prediction method. One scheme doing the prediction in the sensor node (sensor node scheme) and the other one doing the prediction in the controller (controller scheme). One difference in the time series' implementation, compared to the Markov chain's implementation, was that the controller has processing participation in both schemes. It was implemented in this way because the TelosB mote has not enough memory to store the entire time series' program.

The sensor node scheme and controller scheme algorithm first steps are the same. Each sensor node calculates its energy consumption and stores it to construct a time series. When it completes thirty samples, it sends those samples to the controller. It was implemented with thirty samples because it takes the energy consumption information of several sensing periods and it does not overflow the message size. Then, the controller calculates 26 different ARIMA models and chose the one with the least Akaike information criterion (AIC). The list of ARIMA models test is shown in Appendix B. On this point, the predictions schemes take different ways. In the sensor node scheme, the controller sends the model parameters to the sensor node. The sensor node uses those parameters to forecast the next ten energy consumption values. The prediction

Figure 32: Average energy consumption when using and not using the Markov chain prediction method

Source: Author

is compared with the real energy consumption and if the error is above the threshold defined, the node sends a new time series to the controller. In the controller scheme, the controller uses the ARIMA model chosen to forecast the next ten energy consumption values and then it sends the prediction estimated to the sensor node. The sensor node compares the energy consumption prediction with the real energy consumption and if the error is above the threshold defined, the node sends a new time series to the controller. Both algorithms are depicted in Figure 33. The red rectangles highlight the difference between both algorithms.

To run the ARIMA modeling and forecasting on the controller, it was used Gretl library (JR, 2009) because it is open source and it is programmed in C language. Both characteristics eased its integration with IT-SDN. When the forecasting runs on the sensor node, it is not possible to use the Gretl library due to compilation compatibility and memory constraints. Thus, a simple version of the Gretl's forecasting algorithm was implemented as a Contiki application.

The experiments details and results are shown in Section 6.4.1.

Figure 33: Controller scheme time series algorithm

Source: Author

6.4.1 Time series experiments

For the time series prediction mechanism, the same experiments were simulated for the Markov chain prediction mechanism. The only difference is that in this case it was possible to run experiments for 49 nodes. To create the energy consumption time series, an energy consumption period of 10 seconds was set. It means that each node checks the prediction error and updates the energy consumption prediction every three hundred seconds (five minutes), if required. The temperature application was configured with the same sensing periods used in Markov chain experiments, but including the 49 nodes network: for 12 nodes, each node senses the temperature every 10 seconds, for 25 nodes every 20 seconds, for 36 nodes and 49 nodes every 30 seconds. From those simulations, the prediction accuracy and the energy consumption were measured. The prediction accuracy is represented by the prediction MAPE and the energy consumption is shown in millijoules.

Figure 34 shows the prediction MAPE of each network size and both schemes. The

information obtained shows two important behaviors: the controller scheme prediction error is considerable smaller for all network sizes, and as the network size increases the prediction error increases too.

Figure 34: ARIMA's prediction MAPE for controller scheme and sensor node scheme

Source: Author

One main factor that may be influencing on the high prediction error obtained for the node scheme, and its considerable difference with the controller scheme, is the way the ARIMA forecasting was implemented after the model selection. Different than the Markov chain prediction mechanism, where both controller and node runs exactly the same program to carry out the prediction, in the time series mechanism the ARIMA forecasting algorithm programmed in the node is a very simple version of the programm running in the controller. The ARIMA prediction running on the controller is powered by Gretl library, and it brings a series of tools to detect and to fix prediction errors. In the case of the node scheme, due to resources constraints, the node is not able to use the Gretl library thus the simple version programmed does not include the mathematical tools to detect and to fix possible prediction errors.

Figures 35, 36, 37, and 38 show the prediction MAPE obtained on each network size sorted between the sensor node and the sink in hops.

Analyzing Figures 35, 36, 37, and 38 it is observed that in most of cases, the controller scheme has a smaller prediction error than obtained using the node scheme. The only exception is the prediction error of nodes from one hop to the sink in the 25 nodes network. Also, different than observed for the Markov chain mechanism, there

Figure 35: ARIMA's prediction MAPE for 12 nodes

Source: Author

is not a trending defined between the prediction error and the distance between the sensor nodes and the sink. In the case of 12 and 49 nodes, the prediction error obtained using the controller scheme increases when the distance to the sink increases. On the other hand, for 25 nodes network, the prediction error decreases when the distance to the sink increases, similar to the behavior observed in the Markov chain experiments. In the case of using the sensor node scheme, the prediction error oscillates but it does not follow the same behavior in all experiments.

Figure 39 shows the average energy consumption per minute, in milijoules, when using the time series prediction mechanism and when not using a prediction mechanism. Results show that by using the time series mechanism the increase in the energy consumption is lower than using the Markov chain mechanism. For example, for 36 nodes the increase in the energy consumption using the Markov chain mechanism is 29,31% and 25,26% for the sensor node scheme and the controller scheme respectively. The increase using the time series mechanism is 3,04% and 4,12% for the sensor node scheme and the controller scheme respectively. There is also the case where the time series mechanism reduces the energy consumption. For 25 nodes it is possible to obtain a reduction of 2,79% using the controller scheme and 7,5% using the sensor node scheme. On the other hand, the network with 49 nodes has a very significative increase in the energy consumption when using the time series prediction mechanism and the sensor node scheme. This increase coincides with the high pre-

Figure 36: ARIMA's prediction MAPE for 25 nodes

Source: Author

diction error obtained in the same scenario. A high prediction error means that it was necessary to run multiple times the prediction algorithm, increasing the processing and communication resources usage.

Finally, the energy consumption was compared for both time series prediction schemes running in real devices. The network deployed is shown in Figure 23. The average node's energy consumption per minute and the standard deviation are shown in Table 17. The energy consumption for both schemes is about equal, similar to the behavior observed in Figure 35, which is the network with a dynamic similar to the network deployed with real devices. What these results show is that for networks with low messages traffic, there is not a considerable difference in the energy consumption if using the controller scheme or the sensor node scheme.

Table 17: Average energy consumption of a real devices network running the time series prediction method

	Average energy consumption (mJ)	Standard deviation (mJ)
Sensor node scheme	188,2	5,9
Controller scheme	193,6	18,1

Source: Author

Summarizing the time series prediction mechanism results, it can be concluded that the algorithm implemented does not help to reduce the energy consumption in a

Figure 37: ARIMA's prediction MAPE for 36 nodes

Source: Author

WSN where each node estimates and informs its energy consumption to a controller. Comparing both schemes, the controller scheme display a better prediction accuracy than the sensor node scheme. Even though, the energy consumption of both schemes is quite similar.

6.5 Summary

In Chapter 6 the accuracy of the energy consumption model proposed was evaluated. Then, two energy consumption prediction mechanism were implemented, one using the Markov chains and the other using time series. The prediction accuracy was measured comparing the value predicted with the value estimated using the energy energy consumption model.

The energy consumption model was used to model the TelosB energy consumption when working with Contiki. The model implemented considered the TelosB hardware characteristics and the Contiki behavior. Initially, a seven-state model was proposed, but then it was simplified to five states. The model accuracy was tested measuring real devices consumption, obtaining a mean absolute percentage error of 6,6%. Then the influence of state transitions in the total energy consumption was measured. Since it was not possible to measure the TelosB state transitions consumption, a theoretical energy consumption with the information available in datasheets was calculated. Results

Figure 38: ARIMA's prediction MAPE for 49 nodes

show that state transitions are around 1% of total energy consumption.

The energy consumption prediction mechanisms results show that the prediction accuracy obtained was not enough to compensate the energy consumption by the processing and communication overhead required for their implementation. The prediction errors obtained were between 10% and 40%. For both Markov chain and time series models, the controller scheme had better prediction accuracy than the sensor node scheme. About the energy consumption, the controller scheme had better performance in the Markov chain implementation. For the time series implementation, both schemes obtained similar results, excluding the 49 nodes network simulation.

Figure 40 shows the prediction error of both prediction mechanism and Figure 41 shows their energy consumption.

Figure 39: Energy consumption when using and not using the ARIMA prediction method

Source: Author

Figure 40: Prediction error of Markov chain and time series prediction mechanisms

Source: Author

Figure 41: Energy consumption of Markov chain and time series prediction mechanisms

Source: Author

7 CONCLUSIONS

This research proposed an energy consumption model for WSN that considers the energy consumption by processing, sensing, communication, and state transitions. To construct this model, the main characteristics of the typical hardware used for WSN and the literature related with WSN energy consumption models were analyzed.

The TelosB's energy consumption was modeled when working with the operating system Contiki. The model was implemented as a Contiki application and tested in real devices, obtaining a mean percentage error of 6,6% and a standard deviation of 2,36%. Also, the percentage of energy consumption by state transitions and by the sensing module usage were measured. Results show that state transitions represent 1% of the total energy consumption, considering the transitions from processing to sensing and from processing to radio module. The state transitions energy consumption were estimated using the information provided by the TelosB's components datasheets. About the sensing module usage, results show that it represents between 0,19% and 0,05% of the total energy consumption. For this specific application, where only one environmental variable was monitored, the state transitions and sensing module energy consumption could be neglected to reduce the model complexity without significantly affecting the final result.

The energy consumption model and the TelosB implementation was employed to construct two energy prediction mechanisms. One mechanism is based on Markov chain and the second one is based on time series analysis. Both prediction mechanisms were tested in a SDWSN and using two different schemes. One scheme concentrates the processing on each sensor node and the second one concentrates the consumption in the SDN controller. The tests were carried out by simulations and with real devices.

The Markov chain energy prediction mechanism results show that concentrating the prediction algorithm processing in the controller, a lower average prediction error is obtained and also it is more power efficient than processing it in the sensor node.

Also, it was observed that for nodes located at four and more hops away from the sink, the Markov chain prediction method reduces its energy consumption. But, for nodes located at one hop from the sink, using the prediction method increases it. Doing a network energy balance, the prediction accuracy obtained was not enough to reduce the network's energy consumption.

The time series prediction mechanism was implemented using ARIMA modeling. Similar to the Markov chain implementation, running the entire algorithm in the controller obtains a lower average prediction error. In the case of energy consumption, both schemes have similar performance. One difference with the Markov chain implementation is that it was not feasible to run the entire ARIMA modeling and forecasting in the sensor node due to resources constraints. To solve this issue, the algorithm was divided, giving the simple part to the sensor node. Comparing both methods, the time series method is more power efficient than the Markov chain method.

Summarizing the results obtained, it can be concluded that considering the processing, sensing, and communication energy consumption, it is possible to do an accurate energy consumption estimation in WSN. The prediction algorithms implementation involve an overhead of processing and communication that considerably increases the energy consumption in WSN. Also, the SDN controller can be used to process complex algorithms that are unfeasible to process in the sensor nodes, and then sends back the information required.

7.1 Future work

- Test the energy consumption model using real state transitions consumption measurements. To carry out those measurements, it is necessary a digital multimeter with a frequency sample of 20 kHz or more.
- Test the network's energy consumption performance running the prediction methods only on selected nodes, for example, only on nodes at three or more hops away from the sink.
- Include more controllers or nodes with high processing capabilities to balance the prediction method processing overhead.
- Expand the energy consumption model considering energy sources, such as batteries or energy harvesting.

REFERENCES

- ABO-ZAHHAD, M. et al. An energy consumption model for wireless sensor networks. In INTERNATIONAL CONFERENCE ON ENERGY AWARE COMPUTING SYSTEMS APPLICATIONS, 5. **Proceedings...** Cairo, Egypt, 2015. p. 1–4. ISBN 978-1-4799-1771-6.
- ACHIR, M.; OUVRY, L. Power consumption Prediction In Wireless Sensor Networks. In ITC SPECIALIST SEMINAR ON PERFORMANCE EVALUATION OF WIRELESS AND MOBILE SYSTEMS, 16. **Proceedings...** [S.l.], 2004.
- ADVANTICSYS. **AS-XM1000**. 2012. Accessed on February 27th, 2017. Available from Internet: <<https://www.advanticsys.com/shop/asxm1000-p-24.html>>.
- AGILENT. **E363xA Series Programmable DC Power Supplies**. [S.l.], 2002.
- _____. **Agilent 34401A Multimeter**. [S.l.], 2007.
- AHMAD, A. et al. An advanced energy consumption model for terrestrial wireless sensor networks. In INTERNATIONAL WIRELESS COMMUNICATIONS AND MOBILE COMPUTING CONFERENCE (IWCMC). **Proceedings...** [S.l.], 2016. p. 790–793.
- AKKAYA, K.; YOUNIS, M. A survey on routing protocols for wireless sensor networks. **Ad Hoc Networks**, vol. 3, no. 3, p. 325–349, 2005. ISSN 15708705.
- AKYILDIZ, I. F. et al. A survey on sensor networks. **IEEE Communications Magazine**, vol. 40, no. 8, p. 102–105, 2002. ISSN 01636804.
- AKYILDIZ, I. F.; VURAN, M. C. **Wireless sensor networks**. [S.l.]: John Wiley & Sons, 2010. vol. 4.
- ALI, A. et al. MPM: Map based Predictive Monitoring for Wireless Sensor Networks. **Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering**, vol. 23 LNICST, p. 79–95, 2010. ISSN 18678211.
- ALVES, R. C. A. et al. IT-SDN: Improved architecture for SDWSN. In XXXV BRAZILIAN SYMPOSIUM ON COMPUTER NETWORKS AND DISTRIBUTED SYSTEMS. **Proceedings...** [S.l.], 2017.
- ATMEL. **8-bit Atmel Microcontroller with 128KBytes In-System Programmable Flash**. [S.l.], 2011.
- _____. **8-bit Atmel Microcontroller with 16/32/64KB In-System Programmable Flash**. [S.l.], 2014.

BAYDERE, S.; SAFKAN, Y.; DURMAZ, O. Lifetime analysis of reliable wireless sensor networks. **IEICE Transactions on Communications**, E88-B, no. 6, p. 2465–2472, 2005. ISSN 09168516.

BISGAARD, S.; KULAHCI, M. **Time series analysis and forecasting by example**. [S.l.: s.n.], 2011. 366 p. ISBN 9780470540640.

BLUETOOTH SPECIAL INTEREST GROUP. **Bluetooth Core Specification Version 4.0 Master Table of Contents & Compliance Requirements**. [S.l.], 2010.

_____. **Bluetooth Core Specification Version 4.2**. [S.l.], 2014. 2684 p. Accessed february 2017. Available from Internet: <https://www.bluetooth.org/en-us/specification/adopted-specifications>.

BOLCH, G. et al. Queueing Networks and Markov Chains. **Journal of Chemical Information and Modeling**, vol. 53, no. 9, p. 1689–1699, 1998. ISSN 1098-6596. Available from Internet: <http://doi.wiley.com/10.1002/0471200581>.

BOWDEN, R. **HART Field Communication Protocol: A technical Overview**. [S.l.]: Fisher-Rosemount, 1999.

BOX, G. E. et al. **Time series analysis: forecasting and control**. [S.l.]: Prentice Hall, 2008.

BOX, G. E. P.; JENKINS, G. M. **Time series analysis: forecasting and control**, 1976. ISBN: 0-8162-1104-3, 1970.

BRASE, C. H.; BRASE, C. P. **Understandable Statistics: Concepts and Methods, Enhanced**. [S.l.]: Cengage Learning, 2016.

BROCKWELL, P.; DAVIS, R. **Introduction to Time Series and Forecasting**. [s.n.], 2002. 437 p. ISSN 00401706. ISBN 0387953515. Available from Internet: <http://scholar.google.com/scholar?hl=en{&}btnG=Search{&}q=intitle:Introduction+to+Time+Series+and+Forecasting{\\#}0{\\%}5Cnhttp://www.jstor.org/stable/1271510?ori>.

BROCKWELL, P. J.; DAVIS, R. A. **Time series: theory and methods**. [S.l.]: Springer Science & Business Media, 2013.

CALHOUN, B. H. et al. Design considerations for ultra-low energy wireless microsensor nodes. **IEEE Transactions on Computers**, vol. 54, no. 6, p. 727–740, 2005. ISSN 00189340.

CARDEI, M. et al. Energy-efficient target coverage in wireless sensor networks. In IEEE ANNUAL JOINT CONFERENCE OF THE IEEE COMPUTER AND COMMUNICATIONS SOCIETIES, 24. **Proceedings...** [S.l.], 2005. vol. 3, p. 1976–1984.

CASADO, L.; TSIGAS, P. Contikisec: A secure network layer for wireless sensor networks under the contiki operating system. In SPRINGER NORDIC CONFERENCE ON SECURE IT SYSTEMS. **Proceedings...** [S.l.]: Institution of Engineering and Technology, 2009. p. 133–147.

- CHANG, J.-H.; TASSIULAS, L. Maximum lifetime routing in wireless sensor networks. **IEEE/ACM Transactions on networking**, IEEE, vol. 12, no. 4, p. 609–619, 2004.
- CHEN, D.; NIXON, M.; MOK, A. **WirelessHART:Real-Time Mesh Network for Industrial Automation**. [S.l.]: Springer US, 2010.
- CHEN, Y.; ZHAO, Q. On the lifetime of wireless sensor networks. **IEEE Communications Letters**, vol. 9, no. 11, p. 976–978, 2005. ISSN 10897798.
- CHEOUR, R.; LAHMAR, K.; ABID, M. Evolution of wireless sensor networks and necessity of power management technique. In FAIBLE TENSION FAIBLE CONSOMMATION (FTFC). **Proceedings...** [S.l.], 2011. p. 75–78. ISBN 9781612846477.
- CHONG, C. Y.; KUMAR, S. P. Sensor networks: Evolution, opportunities, and challenges. **Proceedings of the IEEE**, vol. 91, no. 8, p. 1247–1256, 2003. ISSN 00189219.
- CULLER, D.; ESTRIN, D.; SRIVASTAVA, M. Overview of sensor networks. **IEEE Computer Society**, vol. 37, no. 8, p. 41–49, 2004. ISSN 00189162.
- DIAS, G. M. **Prediction-based strategies for reducing data transmissions in the iot**. PhD Thesis (PhD) — Universitat Pompeu Fabra, 2016.
- DUNKELS, A. Rime-a lightweight layered communication stack for sensor networks. In EUROPEAN CONFERENCE ON WIRELESS SENSOR NETWORKS (EWSN). **Proceedings...** Delft, The Netherlands, 2007.
- _____. The contikimac radio duty cycling protocol. Swedish Institute of Computer Science, 2011.
- DUNKELS, A.; GRONVALL, B.; VOIGT, T. Contiki-a lightweight and flexible operating system for tiny networked sensors. In ANNUAL IEEE INTERNATIONAL CONFERENCE ON, 29. **Proceedings...** [S.l.], 2004. p. 455–462.
- DUNKELS, A.; OSTERLIND, F.; TSIFTES, N. Software-based on-line energy estimation for sensor nodes. In WORKSHOP ON EMBEDDED NETWORKED SENSORS, 4. **Proceedings...** [S.l.], 2007. p. 28. ISBN 9781595936943. ISSN 9781595936943.
- EGEA-LOPEZ, E. et al. Simulation scalability issues in wireless sensor networks. **IEEE Communications Magazine**, vol. 44, no. 7, p. 64–73, 2006. ISSN 01636804.
- GALLUCCIO, L. et al. SDN-WISE: Design, prototyping and experimentation of a stateful SDN solution for Wireless Sensor networks. In IEEE CONFERENCE ON COMPUTER COMMUNICATIONS (INFOCOM). **Proceedings...** [S.l.], 2015. vol. 26, p. 513–521. ISBN 9781479983810. ISSN 0743166X.
- GNAWALI, O. et al. Collection tree protocol. In ACM CONFERENCE ON EMBEDDED NETWORKED SENSOR SYSTEMS, 7. **Proceedings...** [S.l.], 2009. p. 1–14. ISBN 9781605585192. ISSN 9781605585192.

GOYAL, D.; TRIPATHY, M. R. Routing protocols in wireless sensor networks: A survey. In IEEE INTERNATIONAL CONFERENCE ON ADVANCED COMPUTING COMMUNICATION TECHNOLOGIES (ACCT), 2. **Proceedings...** [S.l.], 2012. p. 474–480.

HAAS, C.; WILKE, J.; STÖHR, V. Realistic simulation of energy consumption in wireless sensor networks. In EUROPEAN CONFERENCE ON WIRELESS SENSOR NETWORKS (EWSN). **Proceedings...** [S.l.], 2012. p. 82–97. ISBN 978-3-642-28168-6. ISSN 03029743.

HEINZELMAN, W. R.; CHANDRAKASAN, A.; BALAKRISHNAN, H. Energy-efficient communication protocol for wireless microsensor networks. In ANNUAL HAWAII INTERNATIONAL CONFERENCE ON SYSTEM SCIENCES. **Proceedings...** Maui, HI, USA: IEEE, 2000. vol. 2, p. 10–pp.

HEINZELMAN, W. R.; KULIK, J.; BALAKRISHNAN, H. Adaptive protocols for information dissemination in wireless sensor networks. In ANNUAL ACM/IEEE INTERNATIONAL CONFERENCE ON MOBILE COMPUTING AND NETWORKING (MOBICOM). **Proceedings.** Seattle, Washington, USA, 1999. p. 174–185. ISBN 1581131429. Available from Internet: <http://portal.acm.org/citation.cfm?doid=313451.313529>.

HERGENRÖDER, A.; HORNEBER, J.; WILKE, J. SANDBed: A WSAN Testbed for Network Management and Energy Monitoring. **Proceedings of the 8th GIITG KuVS Fachgespräch Drahtlose Sensornetze**, p. 71–73, 2009. Available from Internet: <http://doc.tu-berlin.de/2009/hergenroeder-sandbed-fgsn09.pdf>.

HORTELANO, D. et al. From Sensor Networks to Internet of Things. Bluetooth Low Energy, a Standard for This Evolution. **Sensors**, vol. 17, no. 2, p. 372, 2017. ISSN 1424-8220. Available from Internet: <http://www.mdpi.com/1424-8220/17/2/372>.

IEEE. **805.15.4-2003-IEEE standards for physical and data communications.** [S.l.], 2003. vol. 30, no. 2, 172–5 p. Available from Internet: <http://www.ncbi.nlm.nih.gov/pubmed/21097060> <http://www.mendeley.com/research/ieee-standards-physical-data-communications>.

IEEE Computer Society. **802.15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs).** [S.l.], 2011. 314 p.

International Society for Automation. **ISA-100.11a Wireless Systems for Industrial Automation: Process Control and Related Applications.** [S.l.], 2009.

JALSAN, K.; FLOURI, K.; FELTRIN, G. Bi-objective layout optimization of a wireless sensor network for footbridge monitoring. In INTERNATIONAL CONFERENCE ON BRIDGE MAINTENANCE, SAFETY AND MANAGEMENT, 6. **Proceedings...** [S.l.], 2012.

JALSAN, K.-E.; FLOURI, K.; FELTRIN, G. Energy Consumption Estimation for Wireless Sensor Network Layout Optimization. In INTERNATIONAL CONFERENCE ON UBI-MEDIA COMPUTING AND WORKSHOPS (UMEDIA), 7. **Proceedings...** [S.l.], 2014. p. 238–242. ISBN 978-1-4799-4266-4.

JR, J. M. Gretl: an econometrics package for teaching and research. **Managerial Finance**, Emerald Group Publishing Limited, vol. 36, no. 1, p. 71–81, 2009.

KAN, B. et al. Energy efficient design of WSN based on an accurate power consumption model. In INTERNATIONAL CONFERENCE ON WIRELESS COMMUNICATIONS, NETWORKING AND MOBILE COMPUTING (WICOM). **Proceedings...** Shangai, China, 2007. p. 2751–2754. ISBN 1424413125.

KOBO, H. I.; ABU-MAHFOUZ, A. M.; HANCKE, G. P. A Survey on Software-Defined Wireless Sensor Networks: Challenges and Design Requirements. **IEEE Access**, vol. 3536, p. 1872–1899, 2017. ISSN 2169-3536.

KREUTZ, D. et al. Software-defined networking: A comprehensive survey. **Proceedings of the IEEE**, vol. 103, no. 1, p. 14–76, 2015. ISSN 00189219. Available from Internet: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6994333>.

LANDSIEDEL, O.; WEHRLE, K.; GOTZ, S. Accurate prediction of power consumption in sensor networks. In IEEE WORKSHOP ON EMBEDDED NETWORKED SENSORS, 2. **Proceedings...** 2005. p. 37–44. ISBN 0-7803-9246-9. Available from Internet: <http://dl.acm.org/citation.cfm?id=1251990.1253399>.

LENNVALL, T.; SVENSSON, S.; HEKLAND, F. A comparison of wireless hart and zigbee for industrial applications. In IEEE INTERNATIONAL WORKSHOP ON FACTORY COMMUNICATION SYSTEMS. **Proceedings...** Dresden, Germany, 2008. p. 85–88.

LEVIN, D. a.; PERES, Y.; WILMER, E. L. **Markov Chains and Mixing Times**. [S.l.]: American Mathematical Society, 2009. 371 p. ISSN 14698714. ISBN 978-0821847398.

LI, J.; ALREGID, G. Network Lifetime Maximization for Estimation in Multihop Wireless Sensor Networks. **IEEE Transactions on Signal Processing**, vol. 57, no. 7, p. 2456–2466, 2009. ISSN 1053-587X. Available from Internet: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4799111>.

LIN, K. et al. An energy prediction clustering hierarchy for wireless sensor networks. In IEEE CONFERENCE ON INDUSTRIAL ELECTRONICS AND APPLICATIONS (ICIEA), 2. **Proceedings...** Harbin, China, 2007. p. 2268–2272.

LU, Y. et al. Benefits of data aggregation on energy consumption in wireless sensor networks. **IET Communications**, IET, 2017.

LUO, T.; TAN, H. P.; QUEK, T. Q. S. Sensor openflow: Enabling software-defined wireless sensor networks. **IEEE Communications Letters**, vol. 16, no. 11, p. 1896–1899, 2012. ISSN 10897798.

MAINETTI, L.; PATRONO, L.; VILEI, A. Evolution of wireless sensor networks towards the Internet of Things: A survey. In INTERNATIONAL CONFERENCE ON SOFTWARE, TELECOMMUNICATIONS AND COMPUTER NETWORKS(SOFTCOM), 19. **Proceedings...** [S.l.], 2011. p. 1–6. ISBN VO -.

MCKEOWN, N. et al. OpenFlow: enabling innovation in campus networks. **ACM SIGCOMM Computer Communication Review**, vol. 38, no. 2, p. 69, 2008. ISSN 01464833. Available from Internet: <http://portal.acm.org/citation.cfm?doid=1355734.1355746>.

MEMSIC. **MEMSIC, Inc - WSN Nodes | XM2110CB**. 2014. Last accessed on February 27th, 2017. Available from Internet: <http://www.memsic.com/wireless-sensor-networks/XM2110CB>.

MEMSIC Inc. TelosB datasheet: Document Part Number: 6020-0094-02 Rev B. p. 2, 2003. Available from Internet: http://www.memsic.com/userfiles/files/Datasheets/WSN/6020-0094-02{_}B{_}TELO.

MICROCHIP. **IEEE 802.15.4 2.4 GHz RF Transceiver Features**. [S.l.], 2010.

MIKHAYLOV, K.; PLEVITAKIS, N.; TERVONEN, J. Performance Analysis and Comparison of Bluetooth Low Energy with IEEE 802.15.4 and SimpliciTI. **Journal of Sensor and Actuator Networks**, vol. 2, no. 3, p. 589–613, 2013. ISSN 2224-2708. Available from Internet: <http://www.mdpi.com/2224-2708/2/3/589>.

MINI, R. A. F. et al. Prediction-based energy map for wireless sensor networks. **Ad Hoc Networks**, vol. 3, no. 2, p. 235–253, 2005. ISSN 15708705.

MODICA, G.; POGGIOLINI, L. **A First Course in Probability and Markov Chains**. [S.l.: s.n.], 2012. 1–334 p. ISSN 1098-6596. ISBN 9781118477793.

MONTENEGRO, G. et al. **Transmission of IPv6 Packets over IEEE 802.15.4 Networks**. [S.l.], 2007. <http://www.rfc-editor.org/rfc/rfc4944.txt>. Available from Internet: <http://www.rfc-editor.org/rfc/rfc4944.txt>.

NAJIMI, M. et al. Lifetime Maximization in Cognitive Sensor Networks Based on the Node Selection. **IEEE Sensors Journal**, vol. 14, no. 7, p. 2376–2383, 2014. ISSN 1530437X.

NATIONAL INSTRUMENTS. **LabView**. [S.l.], 2002.

NUNEZ, G. A.; MARGI, C. B. Duty Cycle Based Energy Management Tool for Wireless Sensor Networks. In BRAZILIAN TELECOMMUNICATIONS AND SIGNAL PROCESSING SYMPOSIUM, 34. **Proceedings...** [S.l.], 2016.

ÖSTERLIND, F. et al. Cross-level sensor network simulation with COOJA. In IEEE CONFERENCE ON LOCAL COMPUTER NETWORKS (LCN), 31. **Proceedings...** [S.l.], 2006. p. 641–648. ISBN 1424404185. ISSN 0742-1303.

PARK, P. et al. Duty-cycle optimization for ieee 802.15. 4 wireless sensor networks. **ACM Transactions on Sensor Networks (TOSN)**, ACM, New York, USA, vol. 10, no. 1, p. 12, 2013.

PATTI, G.; LEONARDI, L.; BELLO, L. L. A Bluetooth Low Energy real-time protocol for industrial wireless mesh networks. In ANNUAL CONFERENCE OF THE IEEE INDUSTRIAL ELECTRONICS SOCIETY (IECON). **Proceedings...** [S.l.], 2016. p. 0–5. ISBN 9781509034741.

PEREIRA, A. H.; MARGI, C. B. Em: Energy management tool for wireless sensor networks. In BRAZILIAN SYMPOSIUM ON COMPUTER NETWORKS AND DISTRIBUTED SYSTEMS, 30. **Proceedings...** [S.l.], 2012.

PERLA, E. et al. Powertossim z: realistic energy modelling for wireless sensor network environments. In ACM WORKSHOP ON PERFORMANCE MONITORING AND MEASUREMENT OF HETEROGENEOUS WIRELESS AND WIRED NETWORKS, 3. **Proceedings...** [S.l.]: ACM, 2008. p. 35–42.

PRAYATI, A. et al. A modeling approach on the TelosB WSN platform power consumption. **Journal of Systems and Software**, Elsevier Inc., vol. 83, no. 8, p. 1355–1363, 2010. ISSN 01641212. Available from Internet: <<http://dx.doi.org/10.1016/j.jss.2010.01.015>>.

RAO, R.; VRUDHULA, S.; RAKHMATOV, D. N. Battery Modeling for Energy-Aware System Design. **Computer**, vol. 36, no. 12, p. 18, 2003. ISSN 00189162. Available from Internet: <<http://doc.utwente.nl/64556/>>.

RUSSELL, D. J. Introduction to embedded systems: using ansi c and the arduino development environment. **Synthesis Lectures on Digital Circuits and Systems**, Morgan & Claypool Publishers, vol. 5, no. 1, p. 1–275, 2010.

SHNAYDER, V. et al. Simulating the power consumption of large-scale sensor network applications. In INTERNATIONAL CONFERENCE ON EMBEDDED NETWORKED SENSOR SYSTEMS, 2. **Proceedings...** 2004. p. 188. ISBN 1581138792. ISSN 0303-2434. Available from Internet: <<http://portal.acm.org/citation.cfm?doid=1031495.1031518>>.

SHUMWAY, R. H.; STOFFER, D. S. **Time Series Analysis and Its Applications**. [s.n.], 2006. vol. 102. 618 p. ISSN 01621459. ISBN 9780387293172. Available from Internet: <<http://books.google.com/books?id=9tv0taI8l6YC>>.

SIEKKINEN, M. et al. How low energy is bluetooth low energy? Comparative measurements with ZigBee/802.15.4. In IEEE WIRELESS COMMUNICATIONS AND NETWORKING CONFERENCE WORKSHOPS (WCNCW). **Proceedings...** [S.l.], 2012. p. 232–237. ISBN 9781467306829.

SOHRABI, K. et al. Protocols for self-organization of a wireless sensor network. **IEEE personal communications**, IEEE, vol. 7, no. 5, p. 16–27, 2000.

TEXAS INSTRUMENTS. **2.4 GHz IEEE 802.15.4/ZigBee-ready RF Transceiver Applications**. [S.l.], 2007.

_____. **Mixed Signal Microcontroller**. [S.l.], 2011.

_____. **CC1120 High-Performance RF Transceiver for Narrowband Systems**. [S.l.], 2015.

_____. **2.4 GHz IEEE 802.15.4/ZigBee-ready RF Transceiver Applications**. [S.l.], 2017.

- TITZER, B. L.; LEE, D. K.; PALSBERG, J. Aurora: Scalable Sensor Network Simulation with Exact Timing. In INTERNATIONAL CONFERENCE ON INFORMATION PROCESSING IN SENSOR NETWORKS (IPSN), 4. **Proceedings...** [S.l.], 2005. p. 477–482. ISBN 0780392027.
- TOLVER, A. An Introduction To Markov chains. 2016.
- TREVIZAN, B.; BATISTA, G.; BORGES, C. TinySDN: Enabling multiple controllers for software-defined wireless sensor networks. **IEEE Latin America Transactions**, vol. 13, no. 11, p. 3690–3696, 2015. ISSN 15480992.
- TSAY, R. S.; TIAO, G. C. Consistent estimates of autoregressive parameters and extended sample autocorrelation function for stationary and nonstationary arma models. **Journal of the American Statistical Association**, Taylor and Francis Group, vol. 79, no. 385, p. 84–96, 1984.
- URICK, R. J. **Principles of Underwater Sound**. Second. [S.l.]: McGraw-Hill, 1975.
- VERBEEK, M. **A guide to modern econometrics**. [S.l.]: John Wiley and Sons, 2008.
- WANG, Q.; YANG, W. Energy consumption model for power management in wireless sensor networks. In ANNUAL IEEE COMMUNICATIONS SOCIETY CONFERENCE ON SENSOR, MESH AND AD HOC COMMUNICATIONS AND NETWORKS, 4. **Proceedings...** 2007. p. 142–151. ISBN 1424412684 | 9781424412686. Available from Internet: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4292826>.
- WANG, Q. W. Q.; HEMPSTEAD, M.; YANG, W. A Realistic Power Consumption Model for Wireless Sensor Network Devices. In ANNUAL IEEE COMMUNICATIONS SOCIETY ON SENSOR AND AD HOC COMMUNICATIONS AND NETWORKS, 3. **Proceedings...** [S.l.], 2006. vol. 1, p. 286 – 295. ISBN 1-4244-0626-9.
- WINTER, T. **RPL: IPv6 routing protocol for low-power and lossy networks**. [S.l.], 2012.
- WU, W.; XIONG, N.; WU, C. An improved clustering algorithm based on energy consumption in wsns. **IET Networks**, IET networks, vol. 8, 2017.
- XU, Y.; HEIDEMANN, J.; ESTRIN, D. Geography-informed energy conservation for ad hoc routing. In ANNUAL INTERNATIONAL CONFERENCE ON MOBILE COMPUTING AND NETWORKING, 7. **Proceedings...** New York, NY, USA: ACM, 2001. (MobiCom '01), p. 70–84. ISBN 1-58113-422-3. Available from Internet: <http://doi.acm.org/10.1145/381677.381685>.
- YETGIN, H. et al. A Survey of Network Lifetime Maximization Techniques in Wireless Sensor Networks. **IEEE Communications Surveys Tutorials**, vol. 19, no. 2, 2017. ISSN 1553-877X.
- ZHANG, Y. et al. Energy-efficient duty cycle assignment for receiver-based convergecast in wireless sensor networks. In GLOBAL TELECOMMUNICATIONS CONFERENCE (GLOBECOM). **Proceedings...** [S.l.]: IEEE, 2010. p. 1–5.

ZHENG, J.; JAMALIPOUR, A. **Wireless sensor networks: a networking perspective**. [S.l.]: John Wiley and Sons, 2009. ISBN 978-0-470-16763-2.

ZHOU, H.-Y. et al. Modeling of Node Energy Consumption for Wireless Sensor Networks. **Wireless Sensor Network**, vol. 03, no. 01, p. 18–23, 2011. ISSN 1945-3078.

ZigBee Alliance. **ZigBee Specification**. [S.l.], 2012.

APPENDIX A - TELOS B ENERGY CONSUMPTION MEASUREMENTS

Figures 42, 43, 44, 45, 46 show the TelosB current consumption measurements when running the routines explained in Section 6.2.2.

Figure 42: Sensing state current consumption

Figure 43: Low-power mode 1 and processing current consumption

Figure 44: Low-power mode 3 and processing current consumption

Figure 45: Transmitting state current consumption

Figure 46: Receiving state current consumption

APPENDIX B - ARIMA MODELS

Table 18: ARIMA models tested in the controller

Model ID	Parameters		
	p	d	q
Model 1	1	0	1
Model 2	2	0	1
Model 3	3	0	1
Model 4	3	0	2
Model 5	4	0	1
Model 6	4	0	2
Model 7	4	0	3
Model 8	1	0	2
Model 9	1	0	3
Model 10	2	0	3
Model 11	1	0	4
Model 12	2	0	4
Model 13	3	0	4
Model 14	1	1	1
Model 15	2	1	1
Model 16	3	1	1
Model 17	3	1	2
Model 18	4	1	1
Model 19	4	1	2
Model 20	4	1	3
Model 21	1	1	2
Model 22	1	1	3
Model 23	2	1	3
Model 24	1	1	4
Model 25	2	1	4
Model 26	3	1	4