TÉCNICO LISBOA

Cloud-based Wireless Sensor Network System for Smart Livestock Monitorization

João Carlos Duarte Monteiro Ambrósio

Thesis to obtain the Master of Science Degree in

Telecommunications and Informatics Engineering

Supervisor: Prof. Artur Miguel do Amaral Arsénio

Examination Committee

Chairperson: Prof. Paulo Jorge Pires Ferreira
Supervisor: Prof. Artur Miguel do Amaral Arsénio
Member of the Committee: Prof. Ricardo Jorge Feliciano Lopes Pereira

January 2015

Esta dissertação é dedicada à minha Avó Beatriz.

Acknowledgments

Agradecimentos...

Em primeiro lugar à minha Mãe e ao meu Pai por me acompanharem desde sempre em tudo e por estarem sempre presentes para me ajudar. Ao meu Irmão, o meu eterno orgulho, um obrigado especial. À minha Tia por nunca se esquecer de me trazer doces. À minha Avó Beatriz pelo muito que me ensinou e pelo seu exemplar e corajoso percurso de vida. Aos meus avós. Enfim, um obrigado a toda a minha família que representa um enorme pilar da minha vida.

Aos amigos. À Bia, amiga e companheira de todos os momentos, um obrigado especial. Ao Nelito, parceiro de todos os projectos na faculdade, toda a amizade e companheirismo honesto. Sem o seu apoio a realização desta tese teria sido muito mais difícil. Ao Paizinho, Chico, Pedro, Hugo, Mariana e Andreia pelas conversas aleatórias, campeonatos de ragdoll masters e companhia em dias e noites de trabalho. Um agradecimento especial à senhora das senhas. Ao Alexandre, companheiro de todas as corridas. À Catarina, uma nota especial pela motivação do outro lado do Mundo. Ao meu padrinho Zé pela sua importante ajuda. Um especial agradecimento à família Maricato, Pais, Saraiva, Covas e à minha Vizinha e ao meu Vizinho.

Ao professor Artur Arsénio e ao Orlando Remédios pelo acompanhamento, ajuda e sugestões prestadas durante a realização desta tese. A toda a equipa da Sensefinity, em especial ao Tiago e ao Marco. Ao Sr. Amadeu e à Sra. Sésinha, bem como ao Sr. Pedro e restante staff da Queijaria Ribeira de Alpreade por toda a hospitalidade e ajuda. À Sara e ao Sr. José Romana da Câmara Municipal de Cascais, bem como restante pessoal responsável, pela colaboração neste projecto. Por fim, um agradecimento ao Eng. Francisco Borba e restante staff da Herdade de Gâmbia.

Resumo

Avanços na miniaturização da tecnologia e a contínua evolução da Internet permitiram que cada vez mais e mais objectos estejam conectados à Internet, o que fez surgir um novo paradigma chamado Internet das Coisas. Ao mesmo tempo as comunicações estão também a expandir-se para outras formas, tais como as comunicações Máquina-Máquina. Juntamente com estes conceitos, novas aplicações têm surgido a nível Mundial baseadas em Redes de Sensores Sem Fios. Compostas por pequenos nós, este tipo de redes conseguem recolher informação do ambiente à sua volta e comunicá-la sem fios. A quantidade de dados gerado por este tipo de redes requer alta capacidade de armazenamento e processamento, nos quais a Computação em Nuvem poderá dar uma valiosa contribuição. Todos os conceitos apresentados têm potencial para abalar positivamente a forma como a monitorização é feita no sector da Pecuária. Neste documento é proposto um sistema de Redes de Sensores Móveis baseado numa plataforma de Computação em Nuvem. Este sistema foi desenhado para ser aplicado no ramo da Pecuária, mais especificamente na monitorização de ovelhas leiteiras. Esta monitorização tem como base a recolha e processamento da localização de cada animal com o intuito de extrair informação de valor acrescido para os responsáveis da quinta (p.e., alteração do comportamento normal de cada animal, propagação de eventuais doenças no rebanho ou roubos) e ainda combiná-la com outras fontes de informação (p.e., sistemas de ordenha, etc.). Desta forma, permitindo oferecer aos responsáveis de quintas uma visão mais global do negócio. Neste contexto foi implementado o primeiro protótipo funcional do sistema que foi submetido como prova de conceito em cenários de aplicação reais.

Palavras-chave: Internet das Coisas, Comunicações Máquina-Máquina, Redes de Sensores Sem Fios, Computação em Nuvem, Pecuária, Localização.

Abstract

Advances in technology miniaturization and the continuous Internet evolution has boosted the connection of more and more objects to the Internet, leading to a new paradigm named Internet of Things. At the same time, the communications are also expanding to other forms, such as Machine-to-Machine communications. Together with these concepts, new applications have been emerging worldwide based on Wireless Sensor Networks. Composed by small sensing nodes, these kind of networks are able to collect information about the environment around them and communicate it wirelessly. The amount of data that is generated by such networks require high storage and processing power, for which Cloud Computing can give an effective contribution. All these concepts have the potential to positively impact the way monitorization is done on the Livestock field. In this report, a cloud-based WSN system is proposed in order to be applied to the Livestock sector, more specifically for the monitorization of dairy sheep. The monitorization was centered on the collection and processing of each animal's location, in order to extract valuable information to the farmer (e.g, alterations in the normal bustle of each animal, the propagation of possible diseases in the flock of sheep or thefts) and even combining it with other sources of information (e.g., milking systems, etc.). Such solution gives the farmer a broader vision of the business. In this context, it was implemented a functional system prototype experimentally evaluated in real deployment scenarios.

Keywords: Internet of Things, Machine-to-Machine Communications, Wireless Sensor Networks, Cloud Computing, Livestock, Localization.

Contents

Acknowledgments v
Resumo vii
Abstract ix
List of Figures xiv
List of Tables xv
Acronyms xx

1 Introduction **1**
1.1 The Context 1
1.2 The Problem 1
1.3 The Solution 1
1.4 Goals and Contributions 3
1.5 Document Organization 4

2 State of the Art **5**
2.1 Internet of Things 5
2.2 Machine-to-Machine 7
2.3 Wireless Sensor Networks 7
2.3.1 Applications 9
2.3.2 Design Aspects 11
2.4 Cloud Computing 22
2.5 M2M and WSNs with Cloud Computing 23
2.5.1 M2M Cloud-based Platforms 24
2.6 Related Architectures 26
2.6.1 ZebraNet 26
2.6.2 Monitoring and Classifying Animal Behaviour 27
2.6.3 WSANs on the Farm 29
2.6.4 Glacsweb 30
2.7 Summary 31

3 Architecture **33**
3.1 Overview 33

3.2 Requirement Analysis . . . 34
3.3 Architecture Design . . . 34
3.3.1 WSN Infrastructure . . . 35
3.3.2 Cloud Computing Platform . . . 39
3.3.3 Web Application . . . 39
3.4 Summary . . . 40

4 Implementation **41**
4.1 The Context . . . 41
4.2 WSN Infrastructure . . . 43
4.2.1 *MSP430* Microcontroller . . . 44
4.2.2 *CC2500* Radio . . . 46
4.2.3 *SIM908* GPRS Module . . . 48
4.2.4 Data Sensing . . . 50
4.2.5 The Ultimate Node Solution . . . 55
4.3 Network Protocol . . . 56
4.3.1 Network Configuration Phase . . . 57
4.4 Cloud Computing Platform . . . 67
4.4.1 Smart Cloud Data Processing . . . 69
4.5 Web Application . . . 71
4.6 Summary . . . 71

5 Evaluation **73**
5.1 System Components Testing . . . 73
5.1.1 WSN Infrastructure . . . 73
5.1.2 Cloud Computing Platform . . . 78
5.2 System-wide Field Testing . . . 79
5.2.1 *Queijaria Ribeira de Alpreade*'s Experiments . . . 79
5.2.2 Cascais Municipality's Experiments . . . 87
5.2.3 Herdade de Gâmbia's Experiments . . . 89
5.3 Summary and Conclusions . . . 91

6 Conclusions **93**
6.1 Future Work . . . 94

A *CC2500* Radio's *PATABLE* Settings **95**

B *Queijaria Ribeira de Alpreade*'s Test **96**
B.1 Data Uploading Latency Times . . . 96
B.2 *Machinates*® Data Processing Latency Times . . . 97

Bibliography **104**

List of Figures

1.1 From visual animal monitorization to technological monitorization 2

2.1 Data transformation phases 6
2.2 Schematic of a typical wireless sensor node 8
2.3 Star topology 12
2.4 Mesh topology 12
2.5 Star-mesh hybrid topology 13
2.6 Greedy forwarding strategies 16
2.7 Initialization phase in gradient-based routing schemes 17
2.8 Localization techniques 20
2.9 ZebraNet's architecture design 27
2.10 System's architecture design 28
2.11 Architecture Design 30
2.12 Glacsweb's architecture design 31

3.1 Generic system's arquitecture design scheme 35

4.1 The use-case illustrated 43
4.2 WSN node's components schematic 44
4.3 *eZ430-RF2500* development kit 45
4.4 *MSP430F5419A* module built-in the *Butterfinger®* board 46
4.5 *eZ430-RF2500* development tool module 47
4.6 Typical transmitted packet format 47
4.7 Typical received packet format after being processed 48
4.8 *SIM908* module built-in the *Butterfinger®* with a GSM/GPRS and GPS antennas 48
4.9 NMEA 0183 output from a *Adafruit Ultimate GPS* board 51
4.10 GPS data packet format 52
4.11 *Adafruit Ultimate GPS* 53
4.12 *Adafruit Ultimate GPS* with the *MSP430F2274* MCU schematic 54
4.13 Battery data packet format 55
4.14 The ultimate solution; a) protective case; b) the hardware disposition 56
4.15 *CONF* packet format 58

4.16 *CONF* packet network flow . . . 59
4.17 Network topology with depth 3 . . . 61
4.18 Network configuration phase timeline for the network presented in figure 4.17 . . . 61
4.19 Mesh node networking tables . . . 62
4.20 *DATA* packet format . . . 63
4.21 *ACK* packet format . . . 64
4.22 Data exchange phase; a) flowchart of the pickup phase; b) flowchart of the dispatch phase 65
4.23 *Machinates*® back-end logic structure . . . 67
4.24 *Columbus*® messages format . . . 68
4.25 *Machinates*® data message's states . . . 68
4.26 Jordan Curve Theorem illustrated . . . 70
4.27 Total travelled distance calculation . . . 70
4.28 *Machinates*® GUI login page . . . 71
4.29 *Machinates*® GUI dashboard page . . . 72

5.1 Geographic information collected and reported by a mesh node . . . 74
5.2 RSSI variation according to the communication distance . . . 75
5.3 Transmission output power adaptation according to the communication distance . . . 77
5.4 Signal strength adaptability impact on the current consumption . . . 78
5.5 Variation of the processing latency mean times according with the number of messages . . 79
5.6 Monitorized area in the Portuguese village Zebras . . . 80
5.7 Devices in different regional sheep collars . . . 81
5.8 Sheep wearing a monitorization collar . . . 81
5.9 Overall vision of the system illustrated on the *Machinates*® GUI . . . 82
5.10 Animal's location heat map during 24 hours . . . 82
5.11 Geo-fencing alarm illustrated on the *Machinates*® GUI . . . 83
5.12 Mesh node network connectivity during the daytime and nighttime period . . . 84
5.13 Latency times on data uploading to the *Machinates*® back-end . . . 85
5.14 *Machinates*® data processing latency times . . . 85
5.15 Battery consumption over time . . . 86
5.16 Monitorized area in the natural park of Quinta do Pisão . . . 88
5.17 Total travelled distance by hour; a) experimental day 1; b) experimental day 2 . . . 88
5.18 Location variation; a) Latitude variation along time; b) Longitude variation along time; c) Latitude and longitude variation . . . 89
5.19 Illustration of the statistical information collected from the two experimental days . . . 90

List of Tables

2.1 Comparison of different network topologies 13
2.2 Comparison of different wireless communication standards 15
2.3 Overall comparison of the surveyed related architectures 32

4.1 Sequence of AT commands used by the GPRS module 49
4.2 RMC sentence fields description 52
4.3 Sequence of AT commands used by the GPS module 54
4.4 AT command used by the battery application 55
4.5 Description of the *CONF* packet fields 58
4.6 Description of the *DATA* packet fields 63
4.7 Description of the *ACK* packet fields 64
4.8 Description of the *Columbus*® messages fields 69

5.1 Mean and standard deviation of the data represented in figure 5.2 76
5.2 Mean and standard deviation of the data represented in figure 5.3 77
5.3 Mean and standard deviation of the data represented in figure 5.14 78
5.4 Statistical information collected from the two experimental days 90

A.1 Optimum *PATABLE* settings for the possible output power levels 95

B.1 Mean and standard deviation of the data represented in figure 5.13 96
B.2 Mean and standard deviation of the data represented in figure 5.14 97

Acronyms

ACK Acknowledgement

ACLK Auxiliary Clock

ANN Artificial Neural Network

AoA Angle of Arrival

AODV Ad hoc On-Demand Distance Vector

APN Access Point Name

APO Application Objects

ASCII American Standard Code for Information Interchange

AT ATtention

AWP Axeda Wireless Protocol

AWS Amazon Web Services

CDS Connected Dominating Set

CID Connection ID

CMOS Complementary Metal–Oxide Semiconductor

CRC Cyclic Redundancy Check

CR Carriage Return

CSMA/CA Carrier Sense Multiple Access with Collision Avoidance

DSDV Destination-Sequenced Distance-Vector

DSR Dynamic Source Routing

ECG Electrocardiography

FCS Frame Check Sequence

FFD Full-Function Device

FIFO First In, First Out

GLS Grid Location Service

GPRS General Packet Radio Service

GPSR Greedy Perimeter Stateless Routing

GPS Global Positioning System

GSM Global System for Mobile Communications

GTS Guaranteed Time Slots

GUI Graphical User Interface

H2H Human-to-Human

HTTP HyperText Transfer Protocol

IaaS Infrastructure as a Service

IDE Integrated Development Environment

IMEI International Mobile Equipment Identity

IoT Internet of Things

IoUT Internet of Underwater Things

IRP Implicit Routing Protocol

ISDN Integrated Services Digital Network

ISM Industrial, Scientific and Medical

JSON JavaScript Object Notation

LF Line Feed

LPM3 Low-Power Mode 3

LQI Link Quality Indicator

LR-WPAN Low-Rate Wireless Personal Area Networks

M2M Machine-to-Machine

MAC Medium Access Control

MANET Mobile Ad Hoc Network

MCU Microcontroller

MEMS Micro-Electro-Mechanical Systems

MFR Most Forward within r

MRFI Minimal RF Interface

MULE Mobile Ubiquitous LAN Extension

MVC Model–View–Controller

NFP Nearest with Forward Progress

NMEA National Marine Electronics Association

OLSR Optimized Link State Routing

OTAP Over-The-Air Programming

PaaS Platform as a Service

PAN Personal Area Network

PC Personal Computer

PLF Precision Livestock Farming

PRR Packet Reception Rate

RAM Random-Access Memory

REST REpresentational State Transfer

RFD Reduced-Function Device

RF Radio Frequency

RISC Reduced Instruction Set Computing

RMC Recommended Minimum Navigation Information

ROM Read-Only Memory

RSSI Received Signal Strength Indicator

RTC Real-Time Clock

SaaS Software as a Service

SensePnPLayer Sensefinity Plug and Play Layer

SIM Subscriber Identity Module

SMS Short Message Service

SNS Sensor Network Server

SOAP Simple Object Access Protocol

SRD Short Range Device

TDoA Time Difference of Arrival

ToA Time of Arrival

UART Universal Asynchronous Receiver/Transmitter

URL Uniform Resource Locator

USB Universal Serial Bus

UTC Coordinated Universal Time

VLO Very Low Oscilator

WDT Watchdog Timer

WSAN Wireless Sensor and Actuator Network

WSN Wireless Sensor Network

ZDO ZigBee Device Object

Chapter 1

Introduction

This chapter introduces the motivation for this dissertation, the problem to be addressed, as well as the basic concept that is behind the solution. It will be also stated the main goals and contributions of this thesis and finally, it will be described the organizational structure of this document.

1.1 The Context

Livestock represents a vital sector for the economy of several countries. Indeed, it is estimated that Livestock provides over one half of the total value of the agricultural gross output in the industrialized countries, and about one third in the developing countries [1]. In recent decades Livestock production has experienced a tremendous growth stimulated by the population growth and changes in dietary preferences [2]. Nowadays Livestock industry has associated with it several requirements, a lot due to the demands of our modern society [3]. For instance, food safety and quality, sustainable and efficient animal farming, animal health and welfare as well as acceptable environmental impact are some of those requirements.

1.2 The Problem

In order to give an efficient response to the presented demanding requirements, it is necessary to monitor a large number of variables related to this sector. However, together with the increasing scale of the farms and the high number of animals that compose it, it is infeasible to continuously monitor the animals through visual observation during twenty four hours a day [3]. In the past, Livestock management was based on farmer's observation, judgement and experience. Indeed, the only way to locate the animals was to have people watching them, which is a time consuming and expensive approach. In this context, a new concept of management named Precision Livestock Farming (PLF) arises.

1.3 The Solution

Technology is a key part of the PLF vision. Nowadays, the technology is evolving at a very fast pace and spreading to many sectors of our society. Indeed, the Livestock sector is no exception, with the lower

costs and miniaturization of the sensor boards, using WSNs' technology the Livestock monitorization can be made more effectively than using human observation alone (see figure 1.1). It allows to continuously and remotely monitor and capture measurements related with the condition of individual animals in a very detailed level as well as reporting these data to the farm manager [4].

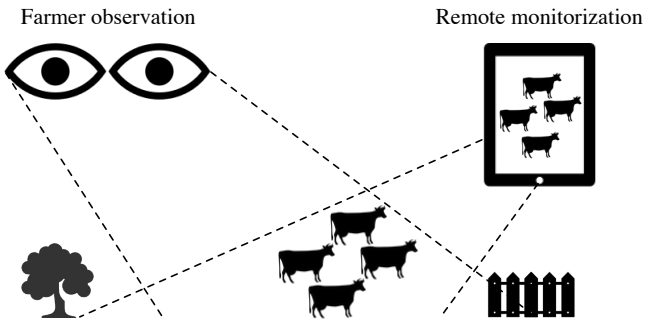


Figure 1.1: From visual animal monitorization to technological monitorization

The adoption of technologies for tracking Livestock animals is an example with plenty of potential that illustrates very well the benefits of allying WSNs with the Livestock sector [5]. In this field, if there were for instance animal's collars that allow to capture the location of each animal, it would be possible to track real-time free-grazing animals' movements in large dimension pastures area, without the need of observations on the ground. This location information can be employed by different applications. For instance, it could be possible to deduce automatically the grazing habits of free ranging animals, which can be very useful to the farmer for making better decisions on the efficient usage of land. Additionally, it could also be possible to know the path of the animals in a certain time, and so determine exactly the utilized grazing area. This could be very useful, when the farmer pays for the grazing land that was used, avoiding this way costs for unused resources. Collecting the location of each animal can be also very interesting when it is detected an infectious disease in an animal, isolating only those animals that had direct or indirect contact with it, avoiding slaughtering healthy animals. On the other hand, knowing in real-time the location of each animal may be very useful when the farmer wants to fetch free-grazing animals at the end of a season, avoiding searching for them in a certain region. In certain countries, insurance companies pay in case of death of an animal if the body is found. Therefore knowing the location of the dead animal body could mean recover some money. Besides these facts, the cost of building and maintaining fences is expensive, and it is one of the most expensive costs associated with Livestock grazing [6]. Therefore, two interesting concepts can be used in order to minimize these costs: geo-fencing and virtual-fencing. In both of them, there are no physical fence, which makes them very advantageous solutions in terms of cost savings.

Geo-Fencing It is a technology used to monitor mobile assets (e.g., trucks, animals, etc.). In the Livestock case, the location of an animal is compared to a virtual boundary (geo-fence), which delimits a particular geographic area of interest. Whenever an animal crosses the geo-fence an alert is generated [7]. This could be very useful in order to send alerts in real-time to the farmer indicating a possible theft.

Virtual-Fencing It is associated with the idea of keeping the Livestock animals away from a certain area without a physical fence. The animal movement is controlled through auditory and tactile stimuli delivered by a device located on its neck. This way, when an animal approaches to the virtual boundary, stimuli are triggered in order to discourage the animal moving into the exclusion zone. This approach can even be used combined with information about the soil moisture to automatically guide the animals to the best grazing lands, and so increase the productivity of the system [8, 9].

These presented concepts become even more interesting when the target grazing areas assume large dimension zones. For instance, animals are often set free in certain geographic areas as a natural mechanism for minimizing the risk of fire in that zones. Therefore, delimiting these large areas with fences may represent unaffordable investments that the geo-fencing and virtual-fencing concepts could substantially reduce.

This new monitoring paradigm of networked Livestock leads to an enormous quantity of information that must be efficiently addressed. For instance, applying this paradigm to the cattle monitorization, each cow will generate on average about 200MB of information per year [10]. Making the calculation for the entire cattle, it becomes clear that there is more information than a local server can manage. Based on this fact, how to deal with this quantity of data? Cloud Computing offers an attractive solution for this issue, making it the solution data back-end. Indeed, it allows the reduction of the initial costs associated with the computational infrastructure and at the same time the farmer does not have to worry about infrastructure issues, thus permitting the farmer to focus on his core business. Another relevant aspect is that the Cloud Computing resources are easily and automatically adjustable according to the real necessities of the Livestock situation. This way, the computational resources are easily scalable following the growth of the Livestock business, so that the farmer does not need to plan in advance its computational infrastructure. Another important point is related with the fact that the customer only pays for the Cloud resources that he actually used, therefore he does not have the problem of paying for resources that were not used. All these advantages have an end result: a substantial reduction of costs, making this kind of technological solutions attractive for big Livestock producers as well as to medium and small scale producers.

1.4 Goals and Contributions

The main purpose of this work is to design, implement and test a WSN-based system for the monitorization of Livestock animals. More specifically, it is intended to design a WSN that essentially captures the geographic location of each node of the system and based on the collected data provides the maximum useful information for the farmer. To accomplish this it is necessary to report the collected data to a back-end infrastructure in order to be treated. This back-end infrastructure will correspond to a Cloud Computing platform, which will allow to obtain a highly scalable and cost-effective solution that will store, process and correlate the sensed information, in order to detect particular events (e.g., health diseases, geo-fencing, etc.). Furthermore, a Graphical User Interface (GUI) will be available by the Cloud

platform in order to streamline the process of data visualization and interaction with the system by the farmer.

The expected goals of this work are the following:

- The design of a solution and architecture that solve the proposed problem;
- The development of a functional prototype for the proposed solution that allows to demonstrate the potentialities of the WSNs in the Livestock area as well as other areas of our society;
- The deployment and evaluation of the developed prototype on real application scenarios. Particularly, it will be evaluated in the monitorization of free-ranging dairy sheep.

Besides the stated main goals, this thesis has the following contributions:

- An analysis on the state of the art concerning the integration of WSN and Cloud Computing technologies for monitoring applications;
- A comparative evaluation of currently related architectures;
- The design of a cost-effective, scalable as well as flexible communication architecture;
- The development of a solution that integrates a WSN with a Cloud Computing platform;
- The implementation of Cloud-based algorithms related to the Livestock animals' monitorization field.

1.5 Document Organization

This dissertation is organized into six chapters. This chapter introduced the problem addressed in this thesis, presenting the main issues and an overview for its solution. In chapter 2 it is presented the state of the art in the scope of this dissertation, as well as some related works done in this area. Then, chapter 3 states the main general system requirements and it is proposed a generic architecture that provides an adequate solution for the identified requirements. Chapter 4 describes in detail the implemented prototype that follows the proposed architecture. The experimental setup, together with the results obtained during the evaluation phase are presented and discussed in chapter 5. Finally, the main conclusions of this dissertation are stated in chapter 6.

Chapter 2

State of the Art

This section starts with the explanation of a promising paradigm, named Internet of Things. This paradigm is presented in section 2.1. Several potentialities and some application scenarios related to it will be stated. In section 2.2 it is described a new communication paradigm, called Machine-to-Machine (M2M) as well as its main challenges. Then, in section 2.3 it is presented an emerging technology named Wireless Sensor Networks (WSNs). In the beginning at this section it will be presented some interesting works in distinct application areas. Afterwards, it will be described and analysed the fundamental technical aspects of this promising technology that must be considered when designing a WSN, presenting at the same time some interesting approaches and its several challenges. The Cloud Computing area is presented in section 2.4 as well as its fundamental concepts and potentialities. Next, in section 2.5 it is described the impact that the Cloud Computing concepts and its characteristics could bring to the M2M and WSNs world, presenting and analysing at the same time some Cloud-based platforms. In section 2.6 it is presented and analysed some interesting related works, giving in particular focus to their architectures. Finally, section 2.7 concludes this chapter presenting the main guidelines that can be drawn from the analysis of the previous topics.

2.1 Internet of Things

Nowadays the majority of Internet connections come from devices used directly by humans, but in a near future, people may become the minority of generators and receivers of traffic [11]. Indeed, our physical everyday objects will also communicate in the virtual world of the Internet, and so the Internet of computers will become the Internet of Things (IoT).

IoT presents a new paradigm that will have a huge impact on our lives and its importance has already been recognized worldwide [12]. But, before describing this promising technology, it is important to define what really is IoT. IoT is simply the point in time when the number of "things or objects" connected to the Internet exceeds the number of people on Earth [10]. According to the Cisco Internet Business Solutions Group, IoT appeared between 2008 and 2009, much due to the impressive growth of smartphone's and tablet's markets. The number of connected devices is still massively growing and it is

estimated that this number reaches 50 billions by 2020. The principle behind IoT is simple. It is based on the presence around us of a variety of "things or objects", which are able to interact and cooperate one another in order to reach common goals [13].

> *"As cows, water pipes, people, and even shoes, trees, and animals become connected to IoT, the world has the potential to become a better place."*
>
> Dave Evans

Nature, humans and objects have been always an enormous source of data but, traditional IT systems are not able to capture, process and understand it in an efficient way. IoT is all about that. In figure 2.1 [10] it is possible to observe the transformation phases that raw data may take to become useful to people. In spite of the tremendous quantity of data that IoT will provide, it will be necessary layers of intelligence to generate wisdom [14]. The more data collected, the more wisdom people can get.

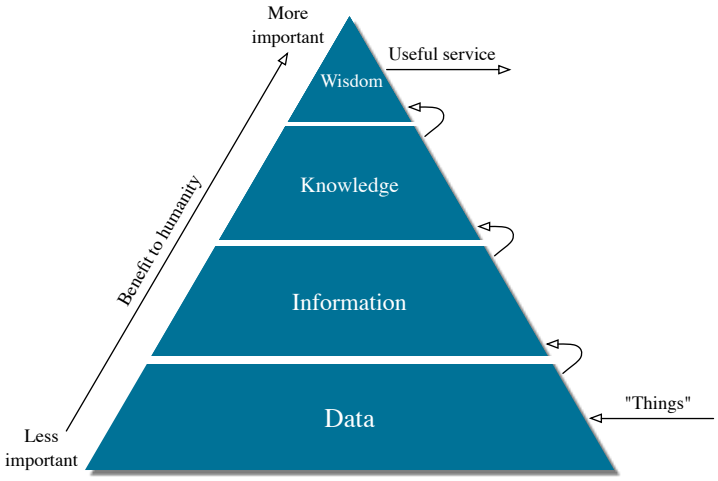


Figure 2.1: Data transformation phases

IoT offers people an infinite universe of possibilities, which can be deployed on many areas. The author of [15] presents a vision of how IoT can provide some technology tools to people with disabilities in order to help them on their daily life activities, increasing their autonomy. So IoT can tell people with disabilities the location of the products they want to buy when they are shopping in the supermarket, help kids with disabilities learning at school bringing them new specialized technological ways to learn (so each kid can learn at his/her own rhythm), and so on. In the healthcare field, Proteus Biomedical designed a system based on pills, which contains a microchip that when swallowed sends through the body's tissues a high frequency electrical current that identifies the pill [16]. This signal is received by a receptor that is also monitoring the vital signs of the patient. All of this information is uploaded to a healthcare professional so that they can follow more efficiently and individually their patients. Even in underwater places, IoT has many potential and is called Internet of Underwater Things (IoUT). IoUT consists of a network of smart interconnected underwater objects that cooperate one another in order to collect information about the underwater universe with educational, scientific, environmental, industrial or military purposes [17].

But if IoT brings great benefits, why has it not been deployed yet? Many are the roadblocks that slow IoT emergence. This new vision of Internet rises serious challenges for scalability, manageability, addressing, identity and robustness, which must be solved [18]. Energy issues and the lack of global common set of standards, especially in the areas of architecture, communications, security and privacy, are some of the delay factors [10].

IoT has the potential of modifying people behaviour so that they can become more proactive and less reactive. Indeed, the potential of IoT in making our world more efficient is only limited by people's imagination.

2.2 Machine-to-Machine

Machine-to-Machine (M2M) communications appeared in the peak period of the analog cellular networks, when it was discovered that digital communications could run over analog cellular networks [19]. The term M2M is associated with communication between machines, through a wired or wireless communication channel and without or little human intervention [20]. In this concept, devices collect and transmit data in an automatic manner, interacting and cooperating one another. For instance, when crash vehicle sensors act collaboratively to detect damage extent and the location of a crash, to facilitate car crash reporting to emergency services. Still in the car monitoring area, the sensors could report engine malfunctions in an advanced way and also indicate when the next maintenance is needed. Besides this, the driver profile and characteristics could be reported in order to identify possible dangerous drivers. Perhaps if a vehicle is stollen it could be possible to obtain the car location more frequently and so the authorities can intercept the vehicle more easily and quickly [19, 21].

> *"The possibilities that arise with virtually instant and automated communications are numerous and surprising in depth and breadth."*
>
> David Crowe

M2M communications have many technology challenges to solve until it can finally proliferate in a global way. Current wireless networks are essentially optimized for Human-to-Human (H2H) communications, so it is necessary to support in an efficient way M2M communications [22]. Nowadays, M2M markets are highly segmented and based on proprietary solutions. Thus, standartization is a key word in M2M future [21].

According to the IoT vision, M2M is not just only machines communicating with machines, but also the integration of other systems, in order to connect data, "things", people, processes and knowledge, having much more increased value. Indeed, it is essential to obtain real collective intelligence and real-life applicability [18].

2.3 Wireless Sensor Networks

The origin of the Wireless Sensor Network (WSN) concept is related with military applications and its appearance was motivated by the recent advances in Micro-Electro-Mechanical Systems (MEMS)

area that has enabled the production of low cost, low power and multifunctional sensor nodes [23]. In general, WSN concept concerns to a large number of inexpensive small sensing self-powered nodes, densely distributed in the region of interest, that collect information or detect special events and communicate in a wireless way, with the final goal of deliver their data to a sink node (i.e., base station) which may be connected to other networks (e.g., the Internet) [24].

> *"Wireless sensor networks currently exhibit an incredible research momentum. Computer scientists and engineers from all flavors are embracing the area."*
>
> Roger Wattenhofer

Typically, wireless sensor nodes have batteries as energy sources and can be equipped with a large variety of sensing units (which enables to monitor an enormous diversity of physical or environmental conditions, such as temperature, moisture, pressure, vibration, motion, etc.). In WSNs, aspects like accuracy and calibration are critically relevant; it is necessary that the data collected by the sensor nodes accurately reflect the state of the environment [25]. Besides this, they generally have a processor, memory and, logically, a wireless communication interface. In a simple manner, a wireless sensor node can be represented with four main components as illustrated in figure 2.2 [14].

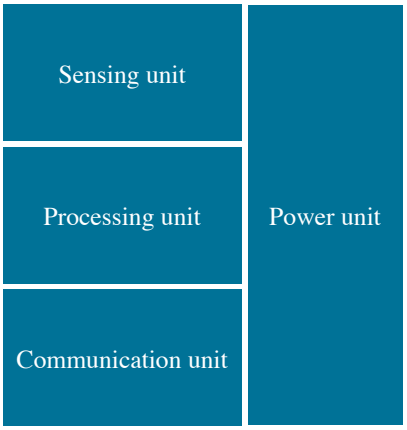


Figure 2.2: Schematic of a typical wireless sensor node

As this kind of sensors may tend to have small dimensions (e.g. the Smart Dust millimeter-scale nodes [26]), this fact imposes some limitations on its resources, such as energy, processor and communication capabilities [27]. In general, existing wireless sensor nodes typically use 8 or 16 bits microcontrollers with tens of kilobytes of Random-Access Memory (RAM), hundreds of kilobytes of Read-Only Memory (ROM) for program storage as well as low-power radios for communication [28]. As its average power consumption is very low it is possible to have long-term operation with just only two AA batteries.

Besides the sensors, a WSN can also have actuators in order to control particular elements of the system, and so in this case it is named Wireless Sensor and Actuator Network (WSAN) [23]. In this kind of networks, sensors gather information of the surrounding environment, while the actuators perform specific actions upon this environment, according with the sensors measurements. For instance, the actuation could be made triggering the irrigation of a cultivation field, based on its moisture measurements [29] or

by guiding the animals of a cattle to a specific location, applying different kind of stimuli, based on their actual location [9].

It is necessary great progresses in four critical fields in order to reduce the costs associated with the WSNs deployment while increasing its functionality; these four areas are: sensors, complementary metal–oxide semiconductor (CMOS) based devices, networking protocols as well as energy storage and harvesting technologies. The massive deployment of WSNs is dependent on the progress made in these fields and achieving this, it could finally make the IoT vision a reality [30]. In fact, WSN provides endless opportunities and has the potential of changing the way we live and work. Specially, WSN has the potential of revolutionize several areas, such as environmental, military, industrial and healthcare area as well as at our homes [23]. In particularly, WSN can be quite useful in systems with more limited human intervention.

2.3.1 Applications

As mentioned before, the applicability that was born from the combination of sensing, processing and radio capabilities, given by the WSNs is extremely high and so this concept is of the interest of the most diverse fields of our society. Some application areas will be listed below and for each of them some relevant works will be briefly described.

Environment The potentialities that the WSN's technologies have been bringing in the past years to the environmental field represent a valuable tool to this area, much due to its implementation facilities in remote sites that may correspond to wide areas. Typically, the environmental monitorization applications can be divided into two main categories [31]: indoor and outdoor monitoring.

Indoor monitoring typically includes the monitorization of buildings and the measured physical variables could be temperature, light, air quality, etc. An interesting example of an indoor use case is the incorporation of sensors inside cement blocks or attached to structural units that sense and record the vibrations of the building [32]. In the case of a earthquake scenario, beyond the typical analysis of cracks and damages, the analysis after an earthquake of real data collected by the sensors could be very useful for a more precise inspection of repairs.

On the other hand there is the outdoor monitoring; in this case this kind of monitorization includes habitat monitoring, earthquake and flooding detection, volcano eruptions, weather forecast, precision agriculture, etc. One of the first implementations of a large scale WSN with an environmental monitorization purpose was made in Great Duck Island [33]. The goal of this project was the monitorization of seabird nesting in a non-intrusive way. To accomplish this, some nodes were deployed inside birds burrows to detect its presence and the rest of them were installed in the surrounding area in order to monitor the local environment conditions. Moving to the precision agriculture field, Mafuta et al. [29] developed a smart irrigation management system in Malawi through the deploying of a WSAN. In this work the irrigation is triggered based on the temperature and moisture measurements of the soil in a maize plantation, in order to turn the irrigation process more efficient, irrigating only when it is necessary.

Military Military applications can be essentially classified into three types; information collection (e.g., enemy tracking), battlefield surveillance or target classification [34].

In the Defense Advanced Research Projects Agency's (DARPA) self-healing minefield project, a self-organized network of sensors that communicate with anti-tank mines is used with the purpose of automatically redistribute the mines, healing gaps in order to complicate the enemy progress in the battle field [24]. PinPtr is a system composed by an ad hoc WSN that can detect acoustic events, identifying this way muzzle blasts and acoustic shockwaves originated from a sound of a gunfire. Measuring their time of arrival and combining it with the measurements taken by other nodes of the network it is possible to estimate the location of the shooter [35].

Industry In the industrial area, the management of the assets (e.g., pieces of equipment, machinery, stock, etc.) is a crucial aspect for the companies and particularly, the tracking of the assets is seen as a relevant feature. Specifically, in process monitoring and control, the collection of physical measurements by sensing units and its wireless delivery to a control system, given by the WSNs' systems provide great advantages over traditional systems, such as no wiring constraints, easy maintenance, reduced costs as well as better performance [36].

For instance the author of [37] presents the deployment of small sensor nodes in the heating cylinders used in the drying process of paper production, in order to measure the temperature and to control the heating cylinders. Krishnamurthy et al. [38] have demonstrated the WSN's potentialities in the predictive maintenance of the equipment in an industrial context. In order to achieve this, vibration signatures are gathered by sensors in order to predict equipment failures. Experiments were performed in two different scenarios; in a semiconductor fabrication plant and onboard an oil tanker in the North Sea. It was demonstrated that predictive maintenance is a viable application of WSNs among others and it can bring high quality data at a relatively low cost in installation and operation.

Healthcare WSNs' technologies applied in the healthcare area have the ability to improve our actual healthcare system and to make patient monitorization more proactive.

Sudden Infant Death Syndrome (SIDS) is responsible for the death of many infant each year. One great factor that increases the incidence of SIDS is allowing the infant to sleep on their stomach. Based in this fact, Baker et al. [39] developed a prototype, called SleepSafe that monitors the sleeping position of the infant through a sensor that is attached to the infant's clothing. When the infant is lying on its stomach, the parents are immediately alerted, so they don't need to constantly watch their child when it sleeps.

In the hospitals, WSNs can help in the vital signs monitorization by replacing the wired sensors that are commonly found in hospitals and emergency rooms [28]. This way WSN can help reducing the wires and the patients anxiety, reducing at the same time the error occurrence.

Home Nowadays smart sensors can be incorporated in almost every domestic devices (e.g. microwaves, refrigerators, etc.) that can interact with each other, enabling the end-user to manage these devices in a more easily way [23].

An interesting example of the use of the WSNs in the home context is proposed in [40] and its main goal is the reduction of the energy consumption at home. In this project, home appliances are automatically controlled according to the user habits and profile. The actual user preferences are predicted based on its previous behaviour observed by the system. Also based on its previous actions it is possible to estimate the user behaviour for future periods (e.g. heating the room at the desired temperature before the user come in).

2.3.2 Design Aspects

Most of the WSNs' deployments came from research attempts or made for a very specific application scenario. There are already limited generic solutions and so, typically it is necessary to adjust and choose the WSN's design and technologies that best fit into a specific application scenario and that fulfils its requirements. Below it will be stated and analysed some relevant design aspects that play an important role when designing and deploying a WSN and that may vary according to each specific application scenario.

2.3.2.1 Network Topologies

Before describing some existing topologies, it is convenient to present the various types of nodes that participate in the system. Sensor nodes, which can be static or mobile, collect information and communicate through wireless links. Collected data is sent to a sink node (i.e., base station), which may be connected to other networks (e.g., the Internet) through a gateway [24].

Topologies will define the configuration of the various types of nodes in the system and how data is transmitted through that configuration. Based on the following works [41, 42], some important network topologies are described below. Their success and applicability strongly depends on the specific application requirements and environmental constraints.

At the end of this topic it is presented the summary table 2.1 that compares some characteristics of the network topologies that will be discussed along this section.

Star Topology Star networks are one of the most common and simple communication topologies and is based on a single hop approach. In this topology all sensor nodes are directly connected to a central node (i.e., sink node), and so this node is the only one that they are able to communicate with (see figure 2.3). So that, in this situation, sensor nodes are called end nodes. Star topology is typically used in applications that have limited coverage area and few sensor nodes.

Simplicity, low latency communications between the end nodes and the sink node and also low power consumption of the nodes, are the main advantages of this topology. The fact that all sensor nodes must be within radio range of the sink node (usually 30m to 100m) is an inconvenient of this topology. Another disadvantage is the absence of alternate communication paths therefore, if one path becomes obstructed, the communication between an end node and the sink node is lost.

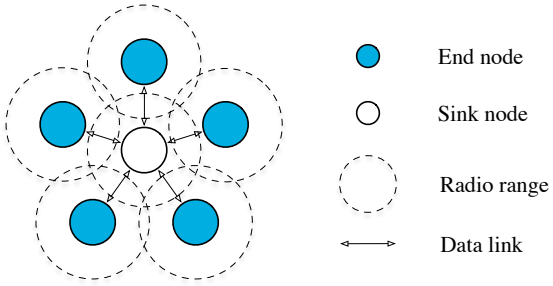


Figure 2.3: Star topology

Mesh Topology Unlike the previous topology, in a Mesh Network any node can transmit data to any other node in the network that, of course, is within its radio range, hopping data between the sensors nodes and the sink node (and on the opposite direction), see figure 2.4. This allows for multi-hop communications, which consists of a node communicating with another one that is out of radio communication range, forwarding data through intermediate neighbour nodes until it finally reach the desired node. So that, besides sensor nodes send and receive their own messages, will also relay messages to or from their neighbours, acting this way also as a router, and so are called mesh nodes. Mesh topology fits in an adequate manner on networks with mobile nodes requirement.

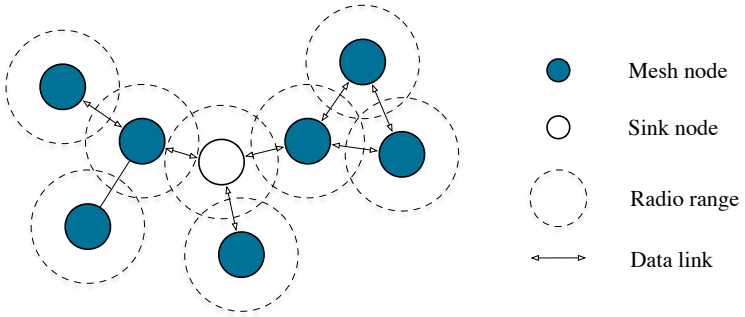


Figure 2.4: Mesh topology

This flexible topology offers redundancy, so if one node fails or a radio link experiences interference, there is always the possibility to deliver the message to another neighbour node, so that it is established an alternative path to reach the desired node. If a sensor node fails, the network will automatically reconfigure itself around the failed node. Scalability is another advantage of this topology due to the fact that it is possible to extend the range of the network, in theory to an unlimited range, simply adding more nodes. On the other hand, this topology requires more power consumption. Besides this fact, when the number of hops to deliver a message increases, the higher the latency of the system.

Star-mesh Hybrid Topology This topology takes advantage of the simplicity and low power of the star topology and the robust and scalability characteristics of a mesh topology. The star-mesh hybrid topology organises end nodes, which are disabled to forward messages, in a star topology around the mesh node. These mesh nodes, in turn, are organized themselves in a mesh network (see figure 2.5).

Mesh nodes are enabled with the capability of forward messages to or from end nodes to other nodes on the network, particularly to the sink node through other mesh nodes, so that network's range is easily extensible.

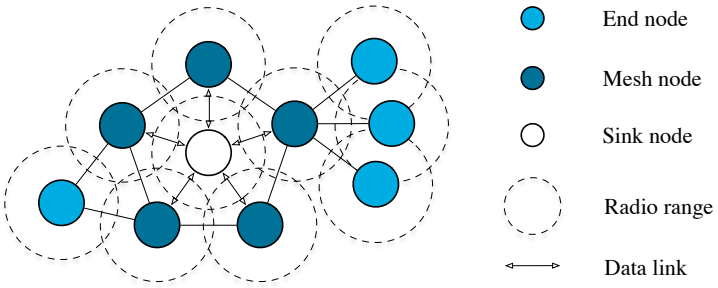


Figure 2.5: Star-mesh hybrid topology

Besides fault tolerance given by mesh nodes, more alternatives paths can be added to the system since end nodes can also be in range of more than one router node, and so establish redundant paths. Comparing with the mesh topology, star-mesh hybrid topology presents low energy consumptions, even more if considering that end nodes can be asleep a significant amount of time, which makes this topology an attractive solution for many implementations of a WSN.

Topology	**Star**	**Mesh**	**Star-mesh Hybrid**
Scalability	No	Yes	Yes
Redundant Paths	No	Yes	Yes
Node Mobility	Partial	Yes	Yes
Energy Efficiency	Yes	No	Yes

Table 2.1: Comparison of different network topologies

2.3.2.2 Data Reporting

In WSNs, the data reporting model is dependent on the application and can be categorized as either time-driven, event-driven, query-driven as well as a hybrid of all these models [43]. In the time-driven model, sensor nodes periodically switch on their sensors and transmitters and transmit the collected data according to a constant periodic time intervals. In the event-driven model, sensor nodes send data to the sink node in face of drastic changes related to a sensed attribute due to the occurrence of certain events. Lastly, in the query-driven model, the sensor node sends its data in response of a query generated by the sink node or another network node.

2.3.2.3 Wireless Communication Standards

This section intends to be a brief overview of some of the typically wireless communication technologies used in WSNs, presenting at the same time a summarized explanation of their main proposes and characteristics. At the end of this topic it is presented the summary table 2.2 [44] that compares some characteristics of the wireless communication standards that will be discussed along this section.

IEEE 802.15.1 (Bluetooth) This IEEE standard [45], typically known as Bluetooth, is a formalization of Bluetooth wireless technology and defines the physical and Medium Access Control (MAC) layers specifications for transmitting information over short distances, between a private group of devices. In this standard, some devices (named slaves) are synchronized to one master device, forming this way a piconet. In a piconet, slaves are not able to exchange information directly between them, so that they only communicate with their master. Slave devices could be connected to more than one piconet, forming this way interconnected piconets (i.e., scatternet). Devices transmit data in packets, during time units, named as slots. Depending on the packet, it can be sent using consecutive slots. The system employs a frequency hop transceiver to minimize interference and fading phenomena, and for that, the frequency hopping pattern's device is determined using the device address and the clock of the master. Frequency hopping occurs between the transmission or the reception of packets.

IEEE 802.15.4 (ZigBee) This IEEE standard [46], simply known as ZigBee, defines the physical layer and MAC sublayer specifications for Low-Rate Wireless Personal Area Networks (LR-WPAN). Its purpose is to be used for low data rate exchange, using for that Radio Frequency (RF) transmissions that cover relatively short distances, between static, portable and moving devices. These devices have very limited power consumption requirements and participate on networks with little or no infrastructure. There are two types of devices that can participate in an IEEE 802.15.4 network: Full-Function Device (FFD) and Reduced-Function Device (RFD). The difference between them resides on the fact that the first one is capable of serving as a Personal Area Network (PAN) coordinator or a coordinator, unlike the second one. A coordinator may operate with a superframe structure or without it. In the first case, devices can communicate during the contention access period, and for that, this standard employs slotted Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA) and ALOHA mechanisms. Applications with special requirements (e.g., low latency) can use Guaranteed Time Slots (GTS), which form the contention-free period. In the operation case without a superframe, the communication is based on an unslotted CSMA/CA or ALOHA mechanism. As stated before, this standard only defines the physical and MAC layers, and so upper layers must be defined by other specifications that will be built upon these standard layers.

ZigBee [47] is an open specification for very low cost and very low power consumption wireless communications that complements the IEEE 802.15.4 standard with network and security layers and provides a framework for application programming in the application layer. The ZigBee network layer has the capability of organizing and providing routing over a multi-hop network. This layer supports star, tree and mesh topologies. A ZigBee application consists of a set of Application Objects (APO) that are spread over several nodes. These APOs consists in a piece of software that enable to control the hardware components of a device, so that other nodes in the network can interact with it. The ZigBee Device Object (ZDO) offers services to the APOs, such as allowing them to discover other devices as well as the services that they implement.

IEEE 802.11 (Wi-Fi) This IEEE standard [48], typically known as Wi-Fi technology, has the purpose of allowing different kinds of devices (e.g., laptops, smartphones, tablets, etc.) to communicate one

another or with the Internet, without a cable connection. Since the appearance of the original standard, were released several amendments, such as IEEE 802.11a/b/g/n, which are widely used. Although the numerous advantages (e.g., high transmission data rates), this standard may not be advisable for applications with very strict low power requirements.

Standard	**Bluetooth** 802.15.1	**ZigBee** 802.15.4	**Wi-Fi** 802.11
Range (m)	∼10 to 100	∼10 to 100	∼100
Data Rate (Mbit/s)	∼1 to 3	∼0.25	∼2 to 54
Power Consumption	Low	Very low	Medium
Infrastructured	No	No	Yes

Table 2.2: Comparison of different wireless communication standards

2.3.2.4 Routing

Routing techniques are required in order to send the data packets between the sensor nodes and the sink nodes. WSNs require routing protocols that may be lightweight regarding both energy and memory issues, should require minimal message overhead as well as they should be tolerant to frequent topology changes and node failures [49]. Below, some routing approaches are presented and for each category it will be discussed the main characteristics and how it allows to overcome some of the WSN challenges. At the end it will be stated and described in more detail some interesting routing protocols.

Proactive Routing Proactive (or table-driven) routing protocols are based on the maintenance of routing information by each node in routing tables that are built based on the information exchanged between network nodes [50]. The routing table must be up-to-date therefore it is necessary to exchange packets between nodes in a regular basis. When it is necessary to route a packet, the source or intermediate node consults its own routing table (that are up-to-date), so that the routing information is immediately available.

Although getting the routing information needed is fast, it requires high overhead traffic. Besides this fact, it is necessary to exchange packets continuously even for routes that doesn't have data traffic flowing on it. The Destination-Sequenced Distance-Vector (DSDV) [51] as well as the Optimized Link State Routing (OLSR) [52] are examples of proactive routing protocols.

Reactive Routing Reactive (or on-demand) routing protocols are based on discovering a route whenever a packet needs to be delivered to a certain destination (e.g. in response to a query or an event), so that routes are discovered as needed [53]. In this type of routing there is no pre-built routing table, and so the route is created by a route discovery process that floods the network with route request (RREQ) packets, starting with the neighbours of the source node. When the route is finally found, a route reply (RREP) packet is sent, in the opposite direction (to the source node), informing the complete route (from the source to the destination node). After this, a route maintenance phase starts with the purpose of

verifying and maintaining the route, for the timestamp specified by the source node.

Reactive routing protocols require low control overhead traffic compared to proactive routing as well as maintain only the routes that are currently in use. An inconvenient of this kind of routing is that the source node typically has to wait for the route discovery process until it can send a message, thereby producing a delay for the first packet to be transmitted. As flooding are used by route discovery processes, this can be a roadblock on scaling to very large networks [49]. The Ad hoc On-Demand Distance Vector (AODV) [54], which is used by ZigBee (see section 2.3.2.3), and the Dynamic Source Routing (DSR) [55] are examples of reactive routing protocols.

Geographic Routing The geographic routing is based on the idea that a sender uses the geographic location of the destination to deliver a message [50, 56]. In this way, the location information replaces the network address in the routing process. In this kind of routing, each node must be aware of its own geographic location, and for that it can use Global Positioning System (GPS) or other localization method (see section 2.3.2.5). The packet sender must know the location of its destination node, such as through a location service that returns the geographic position of the desired node, as it happens in the Grid Location Service (GLS) [57]. Besides this, the sender must be aware of the location of its one-hop neighbour nodes, typically known through one-hop broadcasts.

Most of the geographic routing algorithms start with a greedy forwarding technique [56]. This technique consists of sending a data packet to one of its neighbour nodes lying in the direction of the destination, starting at the source node and repeating at each intermediate node until it finally reaches the destination. There are different strategies to forwarding the packets through the neighbour nodes using greedy forwarding; Most Forward within r (MFR) sends the packet to the closest neighbour node of the destination (minimizing this way the number of hops); Nearest with Forward Progress (NFP) transmits the packet to the closest neighbour of it (when the sender can adjust its signal strength, reducing this way the probability of packet collisions); as well as compass routing that sends the packet to the closest neighbour to straight line between the sender and the destination node (minimizing this way the spatial distance that the packet travels). The figure 2.6 [56] illustrates these three different approaches, where the source node sends the packet to node A following the NFP, or to node B following compass routing or even to node C following the MFR.

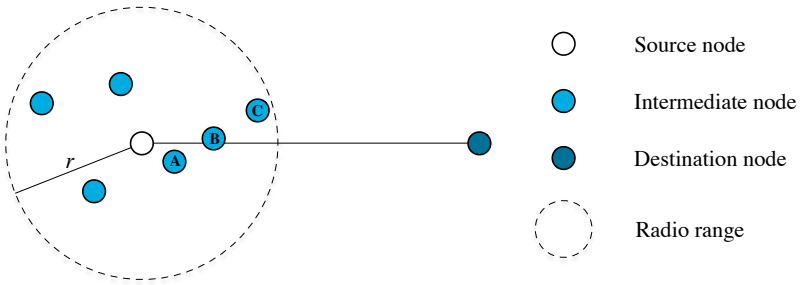


Figure 2.6: Greedy forwarding strategies

The main problem of the greedy forwarding is that is possible that in a certain point a node is closer

to the destination than any of its neighbour nodes, and so the packet gets stuck; this is called the "empty-neighbour-set problem". To solve it, most of the geographic routing algorithms trigger a recovery method (e.g., face routing) when it occurs. Greedy forwarding can be resumed when it reaches a node closer to the destination than the node where recovery method starts. The Greedy Perimeter Stateless Routing (GPSR) [58] uses this presented approach.

Geographic routing is appropriate for networks where its topology may change frequently as well as it is scalable to large networks since it does not need to maintain explicit routes. A limitation of this kind of routing may be the assumption of all nodes of the network knowing their geographic position that may not be a realistic scenario.

Relevant Routing Protocols In this section is presented some relevant routing protocols, focusing slightly in their implementation details. It were chosen some routing protocols that are used for convergent traffic, i.e. in a WSN where the network traffic consists of packets that are originated by sensor nodes and sent to a sink node.

Gradient-based Scheme In the context of the development of efficient technologies for a street lighting system, a research project team has designed and implemented a routing multi-hop protocol based on the gradient routing scheme for a large scenario of wireless sensor nodes [59]. The main idea of routing protocols based on a gradient approach is to forward a packet to a sink node, choosing for that the neighbours with the lowest height, where the height of a certain node represents the number of hops between that node and the sink node.

The network is configured in the initialization phase. In this phase, the sink node starts to broadcast a packet with its height equals to 0. The nodes that receive this packet will increment the packet's height field and broadcast the packet with the incremented height. The packet that has been originated in the sink node will progressively reach the entire network, therefore each node will have an height assigned to it, thus creating height levels on the network (see figure 2.7). The proposed protocol is proactive, despite the network being configured only once at the initialization phase. The heights of the nodes are maintained according to a "hello" message periodic mechanism.

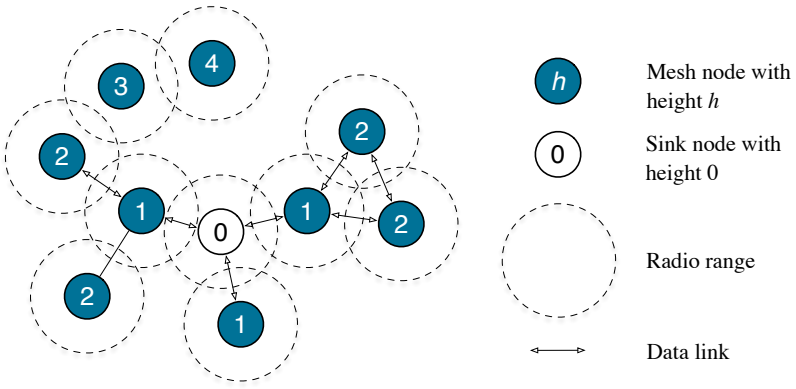


Figure 2.7: Initialization phase in gradient-based routing schemes

Unlike others, this work is based on a single-path for delivery a data packet instead of adopting a broadcast approach. Besides this fact, the authors agree that point-to-point message confirmation is crucial and so each intermediate node must acknowledge informing that the message was successfully received. Three types of packages are necessary for this protocol: setup, data and confirmation packets. When a node needs to send a data packet to the gateway node it starts to send it to its most suitable neighbour node. The packet is sent to the neighbour that has lowest height. In the case of many neighbours share the same height there are four criterions for choosing only one; the most distant neighbour, the nearest neighbour, choosing one neighbour in a random way without repeating nodes as well as in a random way where repetitions are possible. If the transmitter node does not receive the acknowledgement (ACK) after a specific timeout, it will try to retransmit the packet to another node, at this time to the second most suitable neighbour, and so on, until the maximum number of retransmissions has been reached. In this case the maximum number of retransmissions corresponds to the number of neighbours with shorter height.

After initial tests it was observed that due to the loss of confirmation packets, the data packets were copied along the path in a non intentional way, reproducing the characteristics of other protocols that are not based on a single-path approach. This fact was softened by creating a forwarding list where is kept (for a certain time) the ids of all the packets that have been forwarded. Simulation tests proved that these single-path and point-to-point confirmation strategies adopted in this protocol allow to obtain better performances compared to other reference convergent protocols. In particular the nearest neighbour strategy has a better performance compared with the others relatively with energy performance and packet delay.

Implicit Routing Protocol Kwong et al. [60] have proposed a routing protocol, called Implicit Routing Protocol (IRP), that was designed to facilitate the information reporting of animals, more particularly it is intended to be applied in a real-time cattle health monitorization system. Therefore, this protocol was designed to diminish the impact of mobility.

Similarly to the routing protocol presented in 2.3.2.4, this protocol comprises two distinct phases: the configuration phase and the data forwarding phase. In the configuration phase the base station periodically floods a *tier* message to the entire network with the base station id and with an hop count field (called *tier* id). As the animals are free to move around, it is necessary to send periodically a *tier* message in order to keep the network correctly configured. This configuration phase is very similar with the initialization phase of the protocol presented in 2.3.2.4. Although the data forwarding phase is totally different. When a node wants to send a data packet to the base station, it will broadcast the data packet containing the *tier* id and the measurement data to its neighbours. The neighbours that have higher or equal *tier* ids will immediately discard the data packet and only the neighbours that have smaller *tier* ids will acknowledge and consequently propagate the packet in the network, until it reaches the base station. This approach allows to forwarding the data packet through the shortest routing path as well as it is not necessary to maintain an explicit routing path between the source node and the base station.

The simulation results show that the network performance is severely impacted as the probability of a

node being out of the communication range of the other nodes increases, due to the consequently increase of the time that a node is disconnected of the network. It was proved that the performance is improved when the number of nodes in each *tier* also increases. In the farm these parameters corresponds to how frequent the cow changes its physical location and how many neighbouring cows remain close. It was also verified that the packet delay can be decreased by increasing the frequency of network configuration. These experiments show that the IRP is suitable for cattle monitorization.

2.3.2.5 Localization

Localization is the process which allows each sensor node that participates in a WSN to obtain its location. This location information can be used to support routing decisions (see section 2.3.2.4) as well as to support application's requirements, when measurement data need to have attached the location where the data were obtained (e.g., in several environmental monitoring applications, such as in bushfire surveillance, water quality monitoring as well as precision agriculture) [49, 61].

The most direct approach to assign physical coordinates (i.e., real geographical coordinates) to each node of a WSN is equipping each one with GPS receivers. Despite accuracy of localization given by GPS, this immediate solution has several disadvantages, such as GPS receivers cost, power consumption, size as well as problems on receiving GPS signals, due to possible obstacles [62]. Another direct approach is the manual configuration that imposes a manually assignment of the node's location, although this solution may be impracticable for large WSNs or for node mobility networks [63].

Besides direct approaches, nodes' location can be estimated through indirect approaches [63]. Indirect approaches (or typically called as relative localization) are based on the presence of a small subset of nodes, called anchors that obtain their location, through GPS receivers, manual configuration or other methods, and transmit it to their vicinity nodes sending periodically packets, which are typically called beacons. The remaining nodes, with unknown location (non-anchor nodes), can estimate their position based on the anchor nodes. Basically, there are two stages in order to estimate a node location using indirect approaches [63]; in the first stage the non-anchor node merely estimates its distance to anchor nodes and in the second stage the non-anchor node finally estimates its actual location using for that all the distances estimated in the first stage. The method used to obtain the node's actual location in the second stage depends on the technique used in the first stage, and could be, for instance triangulation, trilateration as well as multilateration.

In the indirect approach, in order to estimate the distance to a given anchor node, a non-anchor node can use location methods that fall into two categories: the range-based methods and the range-free methods [64]. In the first one, the estimation of the absolute distance between the sender and the receiver is made through the analysis of the characteristics of the communication signal. However, the accuracy of this estimation depends on the transmission medium and the surrounding environment. Besides this fact, this approach requires additional hardware and therefore the size and cost of the solution increases. The characteristics of the communication signal that are typically used for range-based methods are presented below:

Received Signal Strength Indicator (RSSI) RSSI measures the received power of the signal and, knowing the transmitted power, the propagation power loss can be calculated. Finally, based on the propagation power loss the distance from the sender can be estimated.

Time-based Methods Time-based methods comprise Time of Arrival (ToA) as well as Time Difference of Arrival (TDoA). These methods are based on the estimation of the distance from a sender through the propagation time of the signal, knowing for that the propagation speed.

Angle of Arrival (AoA) AoA estimates the angle at which signals were received. The node position can be estimated using geometric relationships.

The second category of indirect approaches is the range-free methods. Unlike the range-based methods, this kind of approach does not need to estimate the absolute distance between the sender and the receiver based on the characteristics of the communication signal. In these techniques the distance is estimated based on the implicit information provided by anchor nodes, usually through message exchanging using beacons. These information could be the number of hops between devices, radio coverage membership, among others. The most common techniques are Centroid [65], APIT [66] as well as DV-hop [67]. As in this approach the localization is made by using connectivity information, the energy consumption is optimized and the design of the hardware is significantly simplified, thus the costs are dramatically reduced.

In figure 2.8 is presented a schematic that relates the various localization techniques previously presented.

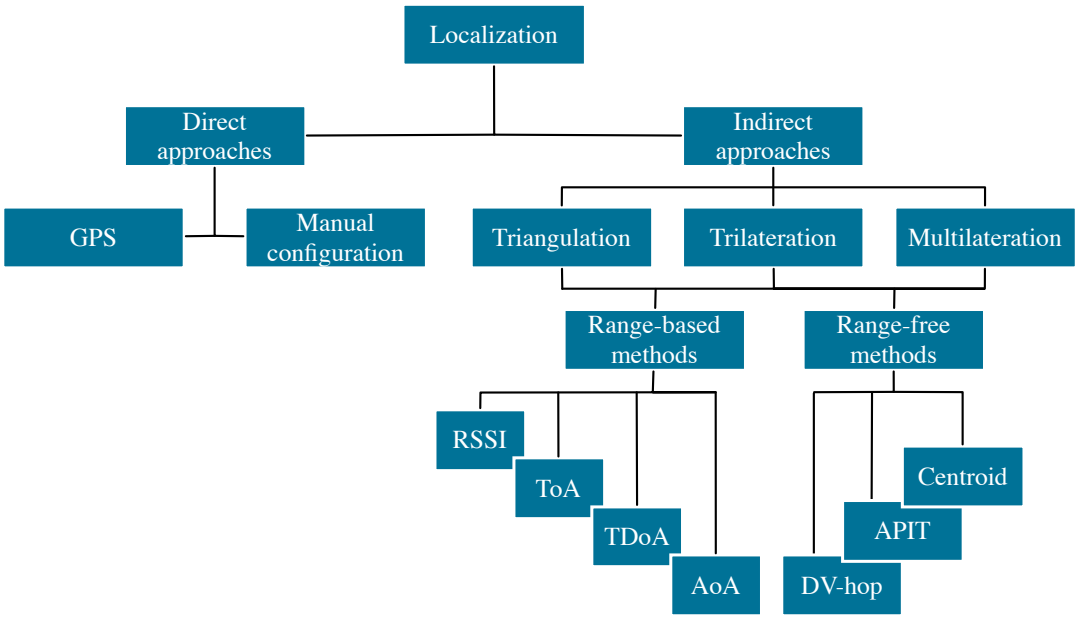


Figure 2.8: Localization techniques

As stated above, the wireless sensor nodes are limited in recourses, being the memory space available

for data storage also very limited, so regardless of the location technique used it is often necessary to store these node's location data in memory until it can finally be sent to other nodes of the network. When the wireless node has to store these location data for extended periods it is required to discard old or new node's location positions or use some kind of data compression technique. Based on the specific case of the localization, Wark et al. [68] proposed a compression algorithm to be applied in nodes location data, which was extensively tested in large-scale cattle monitoring. In this algorithm, when a network node has to store its location for a long period, the position points that contribute with the least information to the trajectory variations (e.g., points in a straight line) will be progressively overwritten in the buffer.

2.3.2.6 Energy Efficiency

In WSN systems, nodes use their limited energy sources in order to perform computations and transmit the collected information in a wireless way. Due to the limited energy capabilities of the nodes, it is essential to use techniques to prolong the batteries lifetime as much as possible. So that, energy efficiency plays an important role when designing a WSN.

It is possible to turn on/off the various hardware components of the nodes in order to save energy [49]. However, depending on the situation, this approach may require coordination with other nodes of the WSN, such as the case of turning on/off the radio component since it is needed for the communication. Indeed, the radio transceiver is one of the most power-hungry components of a WSN node, so it is necessary to manage well its utilization in order to avoid energy wasting. Keeping the radio transceiver switched off as long as possible will allow to reduce significantly the energy consumption. An interesting approach related with this issue is adopted in the work [69], named Duty-Cycling; in this approach, all the nodes of a WSN synchronously switch on their radio transceiver, and so as all of them are active at the same time, they can send the packets in a normal way. After this active period, they switch again their radio transceiver off. This way, when a node needs to send information and its radio transceiver is off, this node must save the information and wait to send it when its radio transceiver becomes active again. This approach requires that all the nodes are synchronised with each other.

Baronti et al. [49] present an approach named Connected Dominating Set (CDS). This approach can be applied in densely populated networks, where the nodes are close to each other. The principle behind this concept is the selection of some nodes (forming the network backbone) to be active all the time and so providing connectivity and message storage to its neighbouring non-backbone nodes. As the non-backbone nodes are sleeping the most of the time and so saving energy, they periodically wake up to send/receive messages to/from their neighbour backbone nodes. Typically, the nodes of the system have to alternate between backbone and non-backbone status since backbone nodes are subject to consume more energy in relation with the other nodes of the system. Besides this, backbone nodes may avoid having always their radio transceiver active, by running adequate energy efficient protocols.

There are emerging WSN applications (e.g., monitoring of critical buildings structures, environmental monitoring, etc.), whose nodes must be able to operate for long periods (e.g., years or even decades) after they are deployed, and where their batteries are hard (or even impossible) to replace [70]. So in these situations, the use of renewable energy to generate electricity (which is not a new concept and it

is called energy harvesting) has the potential of playing a relevant role as it eliminates the necessity of replacing nodes' batteries. In the WSN field, the main sources of ambient energy are solar, mechanical as well as thermal energy. The solar power is the most common form of energy harvesting and can have great potentiality in sunny environments. Although there are several issues related with this area. For instance, the success of harvesting energy strongly depends on several environmental factors as well as the energy harvesting devices must be small in size like the sensors.

2.4 Cloud Computing

Cloud Computing paradigm has recently emerged and performed an extraordinary evolution in recent times. Indeed, nowadays it is a very popular topic and its appearance has revolutionized many other areas. Before describing its characteristics and concepts, it is convenient to agree in a definition.

> *"Cloud computing is a model for enabling convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction"*
>
> NIST (National Institute of Standards and Technology)

In the presented definition there are some important cloud characteristics that deserve to be highlighted [71, 72]:

On-demand Self-service A consumer can automatically get specific computing resources (e.g., server time, network storage, etc.), without requiring human interaction with providers of these resources.

Broad Network Access These computing resources can be accessed through the Internet by heterogeneous devices, such as desktops, laptops, smartphones, tablets, etc.

Resource Pooling Using multi-tenancy or virtualization, the computing resources are "pooled" together in order to serve multiple consumers. Generally customers do not have control or knowledge over the exact location of the provided resources (e.g., storage, processing, etc.), however they are able to specify location at a higher level of abstraction (e.g., country, state or data center).

Rapid Elasticity Costumers can easily and immediately adjust the cloud computing resources, so that they can scale it up or down depending on their necessities. For the customer perspective, computing capabilities appear to be unlimited.

Measured Service The cloud infrastructure is able to measure the usage of computing resources by each individual consumer, through its metering capabilities. So that, costumers only need to pay for the volume of computing resources that they actually use.

Cloud computing services are typically categorized in the following three service models [73]:

Infrastructure as a Service (IaaS) Cloud providers offer its computing resources (e.g., processing, storage, networks, etc.), which can be obtained as a service. Virtualization plays an important role in order to integrate/decompose physical resources. Amazon Elastic Compute Cloud[1] (EC2) for processing and Amazon Simple Storage Service[2] (S3) for storage are examples of IaaS.

Platform as a Service (PaaS) With PaaS, software developers can write their applications according to a given set of platform specifications, and without having to worry about hardware infrastructure issues. PaaS can cover all phases of software development or just a specific area. Google App Engine[3] is an example of PaaS, which enables applications running on Google's infrastructures.

Software as a Service (SaaS) In SaaS, cloud providers deliver and manage software applications, which can be accessed through the Internet. This way, cloud consumers do not need to install and run the applications on their systems. Cloud consumers do not have control over the cloud infrastructure. Google Apps[4] is a SaaS productivity suite, which provides to costumers some Google applications, such as Google Mail and Google Docs.

The advantages of cloud computing are incredibly vast [74]. Indeed, cloud consumers do not have to worry about infrastructure issues (e.g., installation, power, maintenance, etc.). Another relevant point is that cloud computing provides services with lower outages, therefore providing uninterrupted services to their users. Besides this fact, all the information is appropriately backed up in order to provide disaster recovery. Cloud Computing is also viewed as a green computing form due to the fact of minimizing the electronic waste and energy consumptions of organizations, leading this way to environmental preserving. However certain aspects must be taken into consideration, such as security and privacy issues related with data storage and management that concerns, in particularly, enterprises.

2.5 M2M and WSNs with Cloud Computing

Liu et al. [75] stated some vulnerabilities derived from real deployments of WSNs. These vulnerabilities will be then highlighted.

Storage The amount of data that a WSN is able to generate is extremely high, specially considering that a WSN could have hundreds of nodes. Indeed, the amount of data can increase even more dramatically in the presence of particular events that were detected (e.g., movement of the earth surface detected by seismic sensors). In this case, the WSN nodes will collect data at a much shorter interval (e.g., once per minute instead of once per hour), and so the sampling frequency and data transfer rate will increase sharply, resulting in what are known as event and data bursts. In these critical situations, data loss

[1] `http://aws.amazon.com/ec2` (accessed last time on 19th December 2014)
[2] `http://aws.amazon.com/s3` (accessed last time on 19th December 2014)
[3] `https://cloud.google.com/products/app-engine` (accessed last time on 19th December 2014)
[4] `http://www.google.com/enterprise/apps/business` (accessed last time on 19th December 2014)

cannot occur. It was observed that a single standard PC (Personal Computer) server with moderate storage size can easily reach its storage limit, when it is in charge of saving nodes information.

Reliability In the existence of only one server, the system is not resilient to possible failures on this machine caused by hardware breakdown or software crash.

Real-time Processing Often the data collected is both stored and processed off-line. When certain events occur, timely alerts cannot be sent to the users of the systems due to the absence of continuously running algorithms for data analysis.

These issues call for a scalable and powerful storage and processing infrastructure. This is given by the Cloud. With Cloud Computing, WSN can take benefits of its services in order to solve storage, reliability and real-time data processing issues. The virtually infinite storage capacity, a scalable computing capability for data processing and agile tools for application development are some of the advantages that Cloud Computing can bring to WSNs' applications. Also, using the Cloud as the back-end infrastructure, data safety is ensured by geographically distributed data centers that perform regular data backups.

Besides the advantages of combining WSNs with Cloud Computing, there are several challenges resulting from this combination that may be taken into consideration. For instance, the design of new database mechanisms and query methods, lack of standards for data representation as well as network bandwidth restrictions for data transfer are some of those challenges.

2.5.1 M2M Cloud-based Platforms

In this section, some different M2M Cloud-enabled platforms will be presented and analysed in a briefly way.

Axeda Axeda[5] is a Cloud-based platform with the purpose of integrating M2M data as well as building and deploying M2M applications for connected products. As the requirements grow, the scalability of the infrastructure is assured, so that customers do not have to worry about scalability issues. Axeda provides a powerful development environment with flexible APIs in order to facilitate the developing of M2M applications. These APIs offer a rich set of functions that allow to access the core of the platform, providing the ability to query historical data as well as manage assets and alarms. In Axeda the web service communication is based on Simple Object Access Protocol (SOAP) and REpresentational State Transfer (REST).

Beyond basic data transfer, an agent can be included in the device, adding this way more intelligent functions on them. Another interesting feature that Axeda provides is the Axeda AnyDevice Service. This service translates a protocol running on a device to a protocol that the Axeda platform can understand. This service allows to use, on an easy way, M2M devices that communicate with a proprietary protocol and/or are incapable of running an Axeda Agent or the Axeda Wireless Protocol (AWP). This way, resulting on a more vast set of M2M devices that can be chosen.

[5] `http://www.axeda.com` (accessed last time on 19th December 2014)

Cumulocity Cumulocity[6], created by Nokia Siemens Networks (NSN), is a cloud-enabled platform with the purpose of supporting enterprises' M2M systems. The platform and applications are provided as a service, so that the enterprises do not have to worry about computation resources issues. Cumulocity provides a Software Development Kit (SDK) easing the application development process.

Cumulocity is multi-tenant, so that several enterprises (tenants) share the same instance of Cumulocity. However, each tenant has its own database (for storing tenant's users and passwords), storage area (to save tenant's M2M devices' data) as well as a set of subscribed M2M applications. The domain model of Cumulocity is formed by an inventory component and other components that handle events, alarms, readings as well as audit logs from devices. The inventory stores devices (e.g., electricity meters, GPS devices, etc.), assets (e.g., rooms with sensors, cars with GPS devices, etc.) or even business objects (e.g., households, driving routes, etc.), therefore forming the managed objects by Cumulocity.

Cumulocity uses REST interfaces, in this case JavaScript Object Notation (JSON) over HyperText Transfer Protocol (HTTP), for web service communication. For interfacing with M2M devices, Cumulocity uses a driver software called agent. Agents are in charge of three important responsibilities. They act as intermediators, converting device-specific protocol messages into the protocol that Cumulocity understands (and on the opposite direction). The agents are also responsible to transform device-specific model to the Cumulocity reference model (e.g., electricity meter may provide its readings as a parameter *received in Wh*, and so the agent will transform the readings into *total active energy in kWh* according to the Cumulocity reference model). Besides this, they are responsible to establish a secure link to the remote device. An agent is associated with one tenant, which is the owner of the devices that are managed by the agent. The agents can be deployed according to a variety of ways, such as managed agents way, when they are deployed on the Cumulocity Cloud, or non-managed agents way, when they are running on hosts (e.g., mobile phones, gateways, etc.) in the local sensor network. This way, agents simplify on an extremely way the developing of M2M applications by abstracting it from the diversities of M2M devices and protocols.

SmartM2MPT SmartM2MPT[7], which has been recently launched by Portugal Telecom (PT), is a M2M platform with the purpose of improving communication between machines in order to make decisions automatically, increasing this way the global efficiency of the systems and decreasing at the same time enterprises operating costs.

This platform is based on four basic services: connectivity, monitoring, metering as well as localization. Connectivity services are provided in order to manage and supervise the communications with the deployed devices in real-time. Monitoring services allow to interact and obtain the status of the assets, in a continuously and remotely way, such as a smart service of lightning management of the streets, enabling to monitor and control each streetlight. Metering services are used in electricity and water meter readings, eliminating this way on-site meter readings. Localization services allow to locate geographically the assets, such as to locate the vehicles of a fleet.

These services can be applied in several business sectors, such as on public administration, agriculture,

[6] `http://www.cumulocity.com` (accessed last time on 19th December 2014)
[7] `http://www.ptempresas.pt/solucoes-m2m` (accessed last time on 19th December 2014)

industry, construction, distribution as well as on healthcare.

2.6 Related Architectures

In this section some interesting works related to the environmental monitorization through WSNs systems will be described and evaluated, giving in particularly focus on their architectures. Although none of them satisfies completely the requirements of this thesis, for each work will be stated the differences and the most interesting approaches.

2.6.1 ZebraNet

ZebraNet project [76] implemented a system that captures wild animals data and brings it back to biologists, so that the research process can be done more efficiently. In this specific case, the project goal is monitoring zebras that are located over a large wild area, at the Mpala Research Centre in Kenya. Since there is no cellular service covering the region, this project is based on an ad hoc topology in order to route data across the mobile nodes (i.e., zebras) to the base station, which in this case is the final receiver. In the ZebraNet project, all nodes of the system (except the base station) are data sources, while the base station alone is a data sink. Unlike the usual, the base station is also mobile and corresponds to a researcher that periodically drives or flies through the area in order to collect the data stored on the nodes. This way, the base station corresponds to a data MULE (Mobile Ubiquitous LAN Extension). All zebras have a collar that is responsible to periodically store information about the location of the node in question, having for that a GPS receiver.

Each node has two communication interfaces therefore the system has broad control over trade-offs in energy versus communication range. One of them is a short-range radio, typically used for multi-hop communications, with very low power consumption and a communication range of only 100m. The other interface is a long-range radio, typically used for data transfers to the base station, with higher power consumption and a communication range of 8km. The way that data flows from each node to the base station is a key part of the system design. This aspect raises one question: since that not every collar is within range of the base station, how can data be transferred? Instead of sending data directly, it is necessary to send it through other collars, as intermediate hops, so that data finally reaches the base station. In ZebraNet are proposed two main approaches: the flooding protocol and the history-based protocol. In figure 2.9 is presented the system's architecture design.

The flooding protocol consists of flooding data to all neighbours whenever they are discovered. So, if one node finds a fair number of other nodes, then given enough time, data will eventually reach the base station. In this protocol, the large amount of data flooded can lead, in certain conditions, to unaffordable requirements for storage capacity, network bandwidth and energy consumption.

The history-based protocol takes into account the communication history of each node with the base station, so that based on this, each node has associated to it an hierarchy level (the higher the level, the higher the probability that a node is within range of the base station). Each time a node scans for neighbours, it will see if it is within range of the base station. If so, the node will send directly data

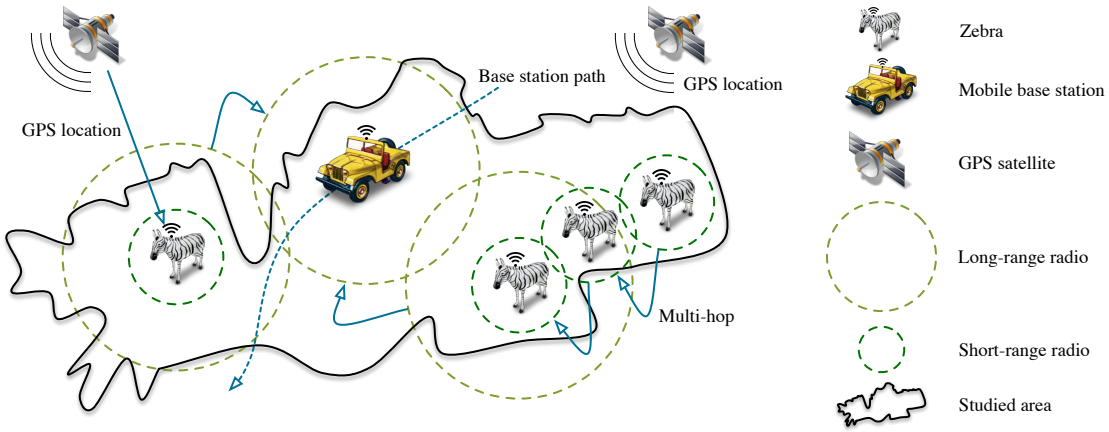


Figure 2.9: ZebraNet's architecture design

to the base station and increment its hierarchy level. Otherwise, it will send data only to the highest hierarchy level neighbour node and consequently decrement its level after d scans without being within range of the base station. The success of this protocol depends tightly on the mobility of nodes and base station. Indeed, if the network has high mobility, a node that was previously near the base station and consequently with a high hierarchy level, in a short time may no longer be the ideal communication target.

Despite the similarities of the ZebraNet project with the use-case of this thesis, there are several disparities. In particularly, the base station must be continuously active and it must not rely on human intervention for data collection. In ZebraNet, the system has high latency tolerance, unlike this thesis that, despite not being a crucial requirement, it is important to have low levels of latency. Besides the two protocols proposed, one interesting option would be to make routing decisions based on the location of the nodes (see section 2.3.2.4). An interesting approach from ZebraNet is the architecture of the system, more specifically concerning the nodes. The fact of the nodes having two communications interfaces, one for multi-hop communications and the other to communicate directly to the base station, makes it an interesting solution, even more attractive if the long-range radio would be placed only on a subset of animals.

2.6.2 Monitoring and Classifying Animal Behaviour

Nadimi et al. [77] propose a system capable of monitoring behavioural parameters of individual animals and transforming them into the corresponding behavioural mode, using for that an Artificial Neural Network (ANN). Farm animal's behaviour and physiological responses can provide important information about their health status and welfare. For instance, the length of the grazing period is an extremely relevant factor that can indicate to a farmer the sufficiency of food, which directly affects both animal welfare and production.

The purpose of this work was to monitor the behaviour of a herd in Denmark. For that, the system design consists on a ZigBee-based (see section 2.3.2.3) mobile ad hoc WSN that is responsible to measure

and monitor behaviour parameters (i.e., head movements) of each individual sheep, providing at the same time communication between the nodes of the network. In this case, each node corresponds to a sheep that carries a collar that are equipped with an accelerometer and an IRIS 2.4GHz mote from Crossbow. Once the network follows an ad hoc topology, each sensor node is a FFD and acts as a router in a multi-hop based network. Besides these nodes, the network has also one gateway and two relay nodes positioned around the field. So this way, data from the sensor nodes can be sent directly to the gateway or be rerouted through the relay nodes. In figure 2.10 is presented the system's architecture design.

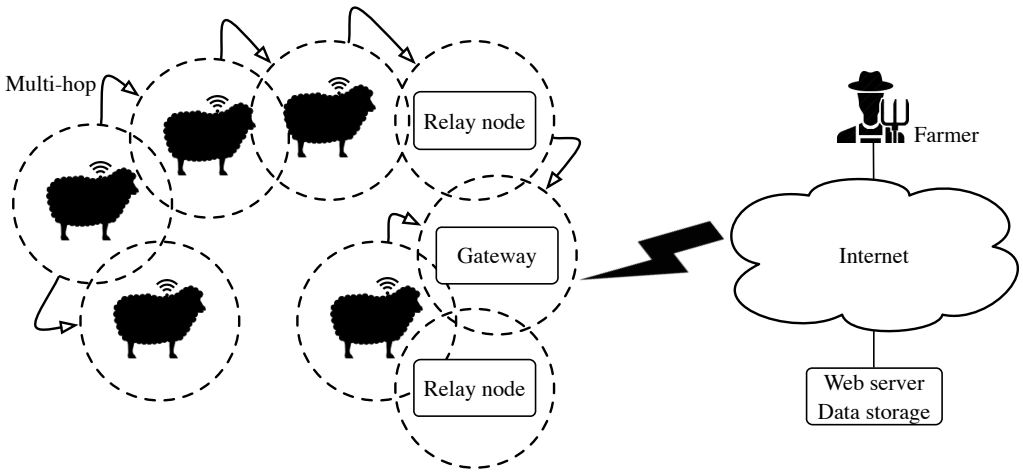


Figure 2.10: System's architecture design

The radio channel 26 (associated with frequency band 2.48GHz) was chosen since several studies demonstrated that the RF interference between ZigBee and Wi-Fi can be avoided at this radio channel. All the nodes of the network were synchronised by a network-wide time synchronization scheme. The network allows fast alert or alarm propagation to support fast response times. Each sensor node that needs to send a message to the gateway, broadcasts the message to its neighbour nodes and fires a retransmission timer. The message is retransmitted when the timer expires and the originating node does not receive the ACK message from the gateway. In this system, the maximum of retransmissions was set to eight, and so after that a new packet with new sensor measurements is sent. Given the difficulty of dismounting the sensor nodes just to update some network parameters, such as sampling rate, the system adjusts these parameters by programming the sensor nodes over the air, using this way Over-The-Air Programming (OTAP). The fact that the variation in the angle of the sheep's head (-30° when grazing to 0° when laying down or standing) and the rotation of the antenna at each node, coupled with the fact that the gateway's antenna is always set to point towards the sky, prevents the angle between the antennas of the gateway and the sensor nodes equaling 90° . Even that, it was verified that the highest percentage of packet loss occurred when the sheep were grazing. Finnally, using an ANN, the data collected from the wireless sensor nodes is processed and the behaviour mode of each animal is constructed. This behaviour mode varies into five types that are grazing, lying down, standing, walking and other modes. This classification can be very useful to the farmer and potentially improves farm management.

Despite of our use case focuses on the geographic location of the nodes and not on the behaviour of the

monitoring nodes, there are several interesting approaches presented in this work. The fact that ACK messages also contain additional information with the purpose of reprogramme sensor nodes over the air, makes it an interesting approach. Another attractive implementation is the fast alarm propagation through the network that can be useful in the case of fast reactive systems. On the other hand, the relay nodes were subject to high data traffic and network congestion, and so required a regular renovation of batteries, which can be a weakness of this system. A possible solution for this problem would be to equip these static nodes with solar panels.

2.6.3 WSANs on the Farm

Sikka et al. [9] have deployed a large WSAN on a farm in order to understand the animals behaviour and, at the same time, to improve the farms management as well as to maximize the farm production. These studies were made by two research teams on a cattle breeding station at Belmont, in Australia.

Two type of nodes have been deployed: static and mobile. Relatively to the static nodes, they consist of twelve geographically distributed soil-moisture nodes, each one collecting information at five different depths within the ground (0.001m to 1m deep), providing this way a depth profile of the moisture. These nodes are equipped with a solar panel in order to charge their batteries, therefore with this fact and due to the low energy consumption of these nodes, they can operate 24 hours per day all year round, except on extended periods of cloud and/or rain. The mobile nodes (corresponding to cows) have equipped (in their collars) Fleck 2 devices, in order to track animal movements. These Fleck 2 devices are able to communicate with a range of 500m in open outdoor environments and have equipped several sensors (e.g., GPS receiver, accelerometers, temperature sensor, etc.). With these devices, the data collected by them, such as the GPS location of an animal, is transmitted in real-time over the ad hoc network formed by the cattle as well as can be buffered on-board. Besides the tracking of animals, the authors also focused on actuation, i.e., providing animals with haptic or audio feedback. This way, based on the animal position, the system applies various stimuli to the animal in order to change its behaviour. For that, Fleck 2 devices have been modified to provide several different stimuli, such as sound, vibration as well as low-level controllable electric shock.

An Integrated Services Digital Network (ISDN) connection has been used to establish communication with the Internet, due to the poor mobile phone coverage on the zone where the farm is located. The ISDN terminal was located on the office of the farm, and as the distance between the office and the WSAN area was considerable (1.5km), it was necessary to install an high-gain antenna on the roof of the office and the relay stations (equipped with solar panels) were deployed along the field. This way the data collected by the WSAN is picked up by a relay station and sent to the office where it is finally stored. The WSAN data is also sent to a PC in Brisbane where it is stored in a database that provides a back-end for a website. This website allows users to see live and historical data from moisture sensors as well as from the animals' collars. In figure 2.11 is presented the system's architecture design.

According to the authors, cattle are very social animals, tending to herd together. The spread of the cattle varies during the day; it may be large during daytime foraging and smaller during rest periods. As observed in this work, animals favours some parts of the pasture compared to others, this is motivated

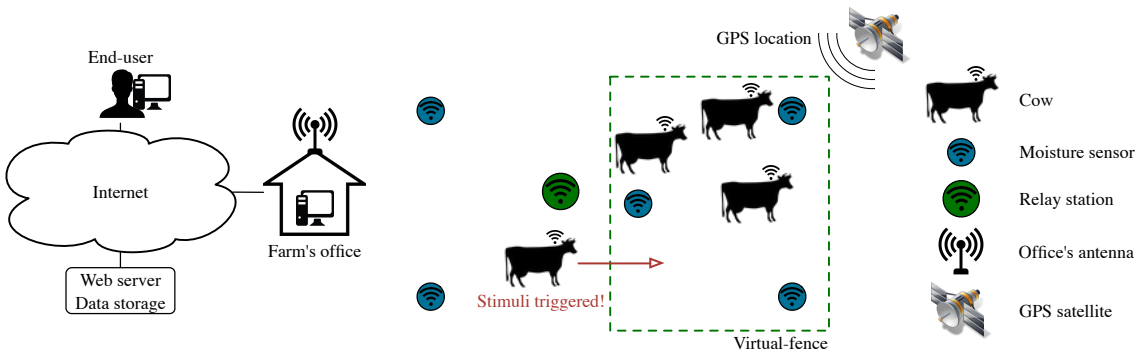


Figure 2.11: Architecture Design

by environmental features, such as water, shade, etc. One of the key goals of this work is to establish virtual fences on the farm. So that, in this concept there is no physical fence and a stimuli is triggered when an animal crosses the virtual boundary in order to encourage it to move out of the exclusion zone.

This work presents several interesting approaches. The animal actuators have the potential of influencing the environment, and this fact combined with virtual fences brings extra benefits for farm management. The long-term vision of this work is to implement an autonomous farm management system. For that, from the data collected by the moisture sensors, the system will automatically determine the areas more suitable for grazing and, according to this, it will build the virtual fences, which will contain the animals that are automatically guided to these locations through stimuli. This approach has tremendous potential and has the ability of making all the system more optimized and smarter. Beyond this, it shows perfectly the innovative power that the WSN's concept can add not only in this area but also in several ones.

2.6.4 Glacsweb

Glaciers are a key element in the sea level rise, although its behaviour and dynamics are poorly understood. Particularly, its dynamics are directly impacted by the subglacial behaviour and based on this fact, the monitorization inside the glacier plays an essential role. The raise of the sea level caused by the melting of the glaciers due the global warming has in aftermath the increased supply of fresh water into the sea that may cause the thermohaline circulation shut down which may lead to the drastic decrease of the temperatures. All these consequences affect tightly our society, so the monitorization of the glaciers is vital.

Martinez et al. [78] have developed and deployed a glacier monitorization system at Briksdalsbreen, Norway. The system is composed by four main elements: the probes that may stay in/under the glacier and are equipped with a variety of sensors (e.g., temperature, pressure, orientation, external conductivity, etc.), a base station on the ice, a reference station that acts like a gateway, measures the supra-glacial movement using a GPS and it can also be used for controlling remotely the entire system, as well as a Sensor Network Server (SNS) located at Southampton, UK, which will act as the final data back-end of the system. In figure 2.12 is presented the system's architecture design.

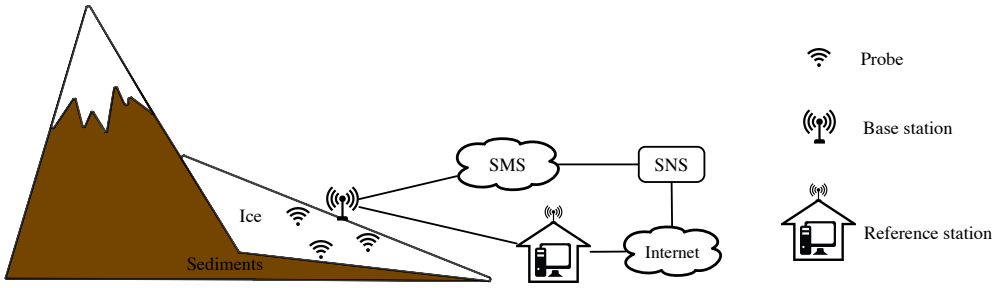


Figure 2.12: Glacsweb's architecture design

In the system the probes are passive in their communications, so that the whole system wakes up and the base station reads each probe directly and sequentially and put it back to sleep. The nodes of the system are synchronized with a communication window and can adjust their clocks through the reception of the Coordinated Universal Time (UTC) of the GPS time that is periodically broadcasted. The electronics of the probes are enclosed in a plastic and they can measure the temperature, pressure, orientation, external conductivity and strain gauge. They collect data six times a day and send these data one time a day in order to save energy. These data are stored in flash ROM in order to deal with breakdowns. The probes are inserted into the glacier (50m to 80m deep) through a hole that is made using a high pressure hot water drill. The base station node has two ways of sending the data to the SNS in the UK; directly to the SNS sending text messages using the Short Message Service (SMS) through a GSM (Global System for Mobile Communications) modem or sending through a long-range radio link to the reference station (the distance between the base station and the reference station is approximately 2,5km) that is connected to the Internet through an ISDN router.

The presence of two distinct communication possibilites in order to send the data to the SNS in the UK is an interesting approach, specially the use of the SMS. Indeed the use of text messages through SMS to deliver directly the probes data to the SNS is an interesting approach as well as the inclusion of dial-in capabilities in the final system for emergency access. The experienced difficulties to communicate with the buried probes due to the degradation of the radio signal when it goes through the glacier ice and sediments is a weakness of the system. As proposed in the paper, this less positive aspect could be overtaken in the future using an ad hoc model which would allow data to be transmitted through the probes until it finally reaches the base station.

2.7 Summary

The high sensing fidelity, fault tolerance, flexibility and low-cost characteristics of WSNs have an incredible potencial to revolutionize many of our society areas. Despite its unquestionable innovative power and the technological advances that have been made in the recent years, the WSN concept has not exploded yet in a global way and most of the WSNs' deployments came from research attempts or made for a very specific application scenario. In this context, some general design aspects has been presented throughout this chapter, focusing in particular the network topologies, data reporting models, wireless

communication standards, routing approaches, localization methods as well as some energy efficiency techniques. Based on these presented alternatives and designing the solution in function of a specific application scenario, it is possible to achieve very interesting deployments. For instance, the ZebraNet's deployment (in section 2.6.1) where the zebras' monitorization was made in a very remote location in a non-intrusive way, following a DATA mule approach, as well as the project "WSANs on the Farm" (in section 2.6.3) where it was used animal's actuators in order to guide automatically the animals to more suitable grazing areas, are very illustrative projects that express very well the potencial of the WSNs in transforming current systems, making them more optimized and smarter. Related to these, it is presented the table 2.3 that makes an overall comparison between the presented systems. One common point between them is that none of them has a Cloud platform to serve as data back-end of the WSN. Most of the WSNs expose some vulnerabilities related with storage, reliability and real-time processing issues of the back-end infrastructure. In this context, Cloud Computing solutions can solve many of these back-end issues and at the same time can help in reducing the associated costs.

Project	**ZebraNet**	**Classifying Animal Behaviour**	**WSANs on the Farm**	**Glacsweb**
Topology	Mesh	Mesh	Mesh	Star
Nodes' Mobility	Yes	Yes	Both mobile & fixed	No
Actuators	No	No	Yes	No
Nodes' Localization	GPS	No	GPS	No
Data Reporting	Time-driven	Time-driven	Time-driven	Query-driven
Internet Connection	No	Yes	ISDN	GSM & ISDN

Table 2.3: Overall comparison of the surveyed related architectures

Chapter 3

Architecture

Section 3.1 presents an overview of the system as well as the outcomes that may be obtained by allying the technological components of the proposed architecture with the Livestock monitorization field. The general requirements that were identified from pilot farm owners and business considerations are stated in section 3.2, whereas in section 3.3 introduces the system's architecture that satisfies the aforementioned requirements. Finally, this chapter is finalized in section 3.4 where it is presented the main features of the proposed architecture.

3.1 Overview

The solution is intended to be applied in the Livestock monitorization area, namely in the monitorization of Livestock animals (e.g., horses, cows, sheep, goats, etc.), specially when they are freely grazing. It is important to have a generic architecture that can be deployed in a wide variety of environments as well as easily scalable allowing to replicate the solution as the business grows by just installing and turning on a device in the new animals of the farm. Based on the concepts presented in section 2 and focusing in particularly the combination of the WSNs with Cloud Computing platforms, it becomes clear that it is possible to improve the monitorization quality done in this area, making it more efficient and available at attractive costs. So using the WSN paradigm it is possible to provide at low-costs the technological support that allows to collect the necessary information of each animal and upload it to a back-end infrastructure to be later processed. The monitorization will be centered in the collection of the animal's geographical information and through the exploitation of this information, it is intended to make important deductions about the actual status of each animal. Indeed, with this new paradigm of monitorization it is possible to extract a tremendous quantity of information that may be an enriching input for the management's tasks on farms or even simplify other tasks, employing the geo-fencing concept. By the other hand using a Cloud Computing platform as data back-end infrastructure gives to the system the flexibility demanded by the business logic by automatically adjusting the computacional resources according to the actual state of the business. This advantage and the possibility of paying only the resources that were actually used, contribute decisively to the reduction of costs. Besides these

considerations, Cloud-based platforms also allow the farmer to focus only on its core business abstracting it from other issues (e.g., back-end infrastructure management). Further, this kind of monitorization helps to reduce even more the costs due to the fact that it is possible to release the workers who were previously in charge of making the animals monitorization's tasks to perform other relevant functions. The effectiveness of this kind of solution becomes increasingly evident as the size of the business gets larger proportions.

3.2 Requirement Analysis

A requirement analysis has been made that allowed to identify the general requirements of the system. This analysis is presented above and was divided in functional and non-functional requirements. This list of requirements has conditioned the design choices made for the architecture's solution.

Functional Requirements

1. Sense periodically the geographic location of each individual animal;
2. Transmit regularly the information collected by the WSN to a back-end infrastructure. This back-end infrastructure must correspond to a Cloud-based platform;
3. Show the processed data to the farmer through a GUI. This GUI must allow the visualization of the collected information, query historical data as well as allow to interact with the system;
4. Trigger alarms and report them to the farmer when special events are detected by the system.

Non-functional Requirements

1. The number of nodes in the WSN may vary widely depending on the specific real scenario. As such, the WSN must be scalable in relation to the number of nodes that compose it;
2. The monitorized nodes have natural mobility and can move on their own due to complex natural behaviours. For these reasons the WSN must be robust to frequent network topology changes;
3. The number of nodes in the WSN is subject to change during its lifetime, therefore the WSN must tolerate node failures;
4. The equipment used by the animals should not interfere with its daily routine and must be physically robust in order to withstand the most varied environmental conditions and animal behaviour.

3.3 Architecture Design

This section presents the proposed architecture as well as its detailed description. Throughout this section all the technical design decisions related to the architecture's solution will be justified. This solution is intended to be generic, so its contextual details will be for now abstracted. Therefore, the

specific technical details may be adapted according to the characteristics of each particular application case.

This architecture can be divided into three relevant components that will be explained in detail in the following three different subsections. The first one is related to the WSN and it will describe the network structure as well as the communication details related to it. The second part corresponds to the Cloud Computing platform that will act as the final processing WSN's data back-end. This way, only the functions that are critically needed for data collection will be carried out by the WSN structure, leaving the heavy processing for the Cloud Computing platform. Finally, the third part corresponds to the Web application that will allow the farmer to visualize the system status as well as to interact with it.

An architecture design scheme is presented in figure 3.1 that illustrates the three main system's components previously mentioned as well as its subcomponents.

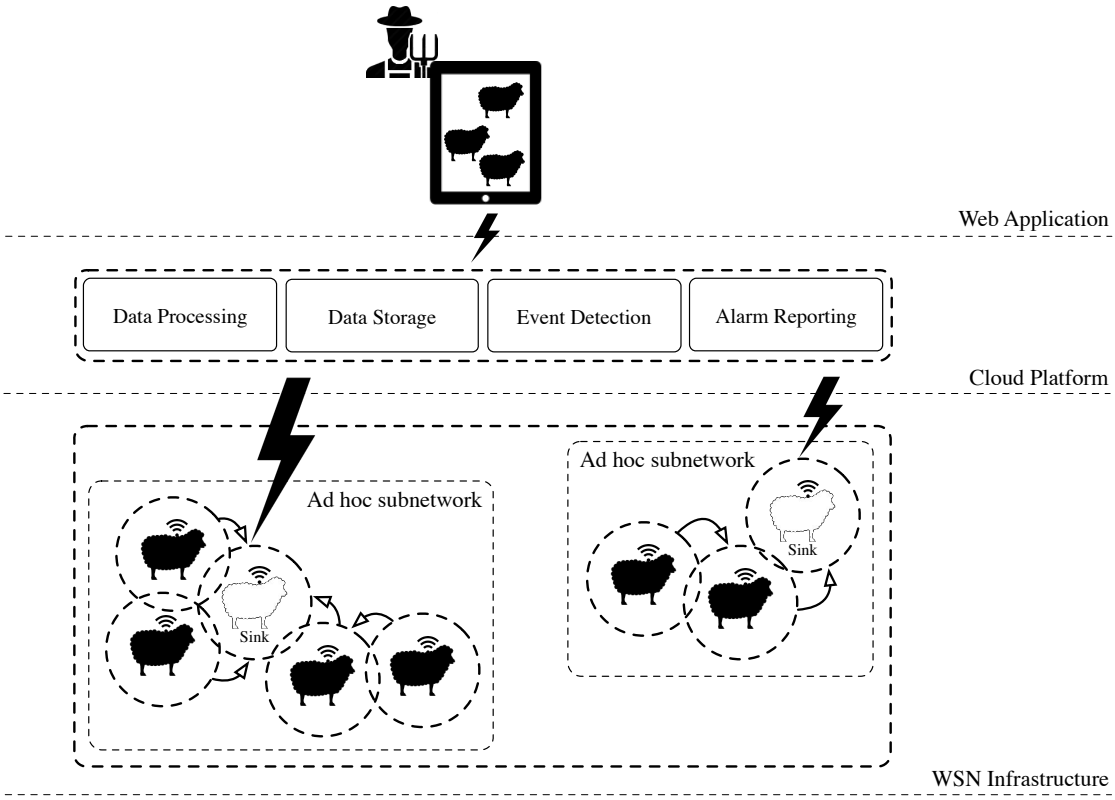


Figure 3.1: Generic system's arquitecture design scheme

3.3.1 WSN Infrastructure

Each node of the WSN infrastructure will correspond to an animal that will be equipped with a device that provides sensing, computation as well as communication capabilities. Therefore, in this system, each animal will correspond to a mobile node that communicates wiressly with the other system nodes. Additionally, it will sense its current geographic location (functional requirement 1), through GPS or other localization method (see section 2.3.2.5), following a time-driven data reporting model (see section 2.3.2.2). All this sensed information is aggregated in a sink node and then uploaded to a

back-end infrastructure to be later processed (functional requirement 2). In order to accomplish these functionalities, each animal will have to use a technological device that must not interfere with its normal activity and at the same time must be robust to the surrounding environmental conditions (non-functional requirement 4). The technological device installation depends on the specific use-case as well as on the animal that is intended to be monitored. For instance, in cow monitorization it can be used a collar, which was extensively tested in several other past projects, or even a device inserted in a bolus that is ingested and remains in the animal's rumen.

The pasture areas vary widely depending on the specific use-case but may correspond to spatial regions of large size. As the nodes are mobile, they may be positioned anywhere on the area. Due to these facts and further adding the limited transmission range of each node, it may become a hard task to assure connectivity with all the animals. Depending on the wireless communication technology used, one possible solution for this problem would be to add several static nodes to the pasture with the purpose of adding connectivity to all or almost all the desired area. However, this solution may represent a large investment and the costs may become even higher as the dimension of the area increases. Besides these facts, it involves a detailed prior study of the pasture area. Therefore, it was chosen an ad hoc approach for the network, and so using a multi-hop routing model, the messages can be propagated from animal to animal as they become within transmission range, so that each node besides collecting data must act as a router (see section 2.3.2.1). This way, instead of trying to cover all the pasture area, it is only given connectivity between the animals, which corresponds to the monitorization targets. Therefore, the monitorized animals will form a Mobile Ad Hoc Network (MANET), providing high scalability and robustness to the network (non-functional requirements 1, 2 and 3). For instance in the cows' monitorization field, some authors [9, 79] support this model by showing evidence that the animals of a cattle remain close to each other (typically herd together), so that an overall connectivity between them was maintained using a multi-hop approach. In another work [79], it was also shown that with solely one-hop connections, the connectivity between all the animals cannot be assured due to the fact that generally the herd do not stay sufficiently clustered. Due to the fact that the presented network will not have any kind of actuation and the data will be only uploaded from the ad hoc network to the back-end infrastructure, it was chosen a convergent protocol (such as presented in the subsection 2.3.2.4) in order to route the data to the back-end infrastructure using a multi-hop approach.

Besides adding connectivity between the WSN's nodes, another issue arises; how can the data collected from the ad hoc network be uploaded to the back-end infrastructure through a gateway, after reaching the sink node? First of all it is important to define what will be the sink node in this architecture. One interesting approach for addressing this issue was used in the ZebraNet's project (analysed on section 2.6.1). In this work a data MULE, corresponding to a mobile base station (i.e., the sink node), was used in order to collect the data from the ad hoc network. With the ZebraNet's solution it is not possible to know whenever a data MULE will appear to collect the information from the ad hoc network, therefore it is not possible to estimate the instant of time that the sensed data will be uploaded to the back-end infrastructure. This uncertainty is not tolerated by the projected system, which makes this solution unfeasible. Another solution for this problem is the sink node being an animal of the ad hoc network.

Therefore, in this approach the only difference to the other nodes of the ad hoc network is that the sink node, as the name implies, will also gather the information of the other nodes (centralizing it) and has the capability to upload all this information to the back-end infrastructure through a long-range radio. All the nodes of the system will use the short-range radio with low power consumption, in order to communicate with the other animals inside the WSN (for instance, using ZigBee, see section 2.3.2.3, as presented in the work in section 2.6.2) and only the sink node will upload the data collected from the WSN's nodes to the Cloud platform, through a gateway, using for that a long-range communication interface.

Due to the natural mobility constraints of this system, even using a multi-hop communication approach, some nodes eventually may become disconnected from each other, therefore forming independent and isolated ad hoc networks. Despite the low vulnerability to disconnections presented in similar deployments [80], it is necessary to attenuate the consequences of this situation. Therefore given these conditions, if the sink node corresponds to a node from the ad hoc network and some nodes may become isolated in separated ad hoc networks, it is not possible to assure connectivity with the back-end continuously in time. Depending on the application use-case scenario, this situation can be tolerated or not. For instance, if the farmer wants to detect possible cattle thefts, the connectivity of all the WSN's nodes with the data back-end infrastructure should be assured continuously, so that if the cow is stolen, it can immediately warn the farmer about this fact. On the other hand, if the farmer does not have strict latency requirements, delayed uploads may be tolerated (for instance, once again the connectivity with the sink node is again re-established). Identified these two scenarios, the system's architecture could follow one of the two models presented below.

Sparse Cloud Connectivity In this approach the system does not require that all the WSN's nodes maintain connectivity with the data back-end continuously in time. In this case, the network will have a certain number of sink nodes, being the remaining mesh nodes, setting this way a given ratio between sink and mesh nodes. This number of sink nodes in the network depends on the real scenario case, in particularly depends on the total number of nodes. In order to add some redundancy to the system, at least two of them will correspond to sink nodes. Since the disconnected nodes cannot propagate the collected information to the cattle's sink nodes, each network node is equipped with a buffer that can be used in order to store temporarily the collected information until the network connectivity is resumed. For instance, if the sink nodes use a General Packet Radio Service (GPRS) module in order to upload the information to the data back-end infrastructure, it is only necessary for the farmer to pay the costs associated with the data plans of the sink nodes and not for the whole network.

Full Cloud Connectivity In this case it is necessary that all the WSN's nodes maintain connectivity with the data back-end continuously in time. In this case, it is necessary each network node to be equipped with the long-range communication radio (besides the short-range radio for the ad hoc communications). The distinction between sink node and mesh node continues to be applied in this architecture, as for each sub ad hoc network only a certain number of nodes will act as sinks and turn on that long-range radio. So it is necessary that all the system's nodes could adapt themselves depending on the network's

current situation and so may alternate between sink nodes and mesh nodes in order to maintain network communication with the data back-end infrastructure. For instance, there is an ad hoc network that is composed by only one sink node, the animals had moved around in the grazing field and the network was divided into two independent networks. In this situation, the nodes from the ad hoc network that becomes without the sink node detect the absence of the sink node and elects one of its nodes to become the new network sink so that the collected data from the network continues to reach the back-end infrastructure. Hence, the system becomes robustness to node failures, tolerating the failure of any kind of nodes by automatically adjusting to each existing scenario and maintaining this way connectivity with the back-end infrastructure.

Depending on the specific real deployment scenario, there are two main approaches to communicate with the Internet (for uploading the collected data to the back-end infrastructure, through a gateway). In the ideal case, there exists a cellular service covering the system deployment area, so that the long-range communication radio could use, for instance, the GPRS technology (as adopted in the work presented in section 2.6.4), and so the node that turns on this communication interface becomes also the system gateway. This solution could be interesting in the case of extensive grazing areas. Otherwise, if there is no cellular service covering the area it must be taken a similar approach to the work presented in 2.6.3 and 2.6.4, where the sink nodes, using a long-range radio link, could deliver the sensed data to an Internet gateway, which in this situation corresponds to a node outside the ad hoc network (for instance located in a farm office). This could be done using intermediate relaying nodes, depending on the specific real scenario. For instance, if the distances between the gateway and the pasture fields are too large, it may be necessary the usage of static intermediate nodes on the field. Therefore, the ad hoc network data will be forwarded through static nodes (which may be equipped with solar panels) that will act as relay nodes, located between the monitorized animals and the gateway. Inspired in the work presented in section 2.6.3, an extra functionality would be to add moisture sensors to the fixed router nodes in order to automatically determine the best grazing areas and so optimize the usage of the pasture areas.

Concerning to the energy efficient issue, each system node periodically will turn on the sensors that allow it to sense its geographic location on the field and after this, it will turn off these sensors again in order to save battery. Depending on the target environment and how the equipment is installed on the animal, it can be used solar panels (for instance in the animal's collar) in order to harvest energy from the environment (see section 2.3.2.6).

Due to the fact that the long-range communication interface is power-hungry, the sink node will have to wait for a certain quantity of data from the ad hoc network in order to send all of them together, optimizing this way the communications with the back-end infrastructure. This way the sink nodes will have to save temporarily the information in a buffer memory for later transmission. Besides the sink nodes, the mesh nodes will also have to save in memory data from themselves and from other mesh nodes (since the data is propagated in a multi-hop approach), before propagating it to the sink node. As mentioned before the WSN's nodes have some limitations with regard to the memory space available. Therefore, it may be advisable to compress some data whenever an ad hoc node is getting without memory free space. In this specific case of nodes's geographic localization, it can be used summarization

within the buffer using for that a compression algorithm, such as the one presented in [68], which was extensively tested in large-scale cattle monitoring.

3.3.2 Cloud Computing Platform

The back-end infrastructure that supports the WSN corresponds to a Cloud Computing platform (functional requirement 2) due to the advantages that this kind of Cloud-based infrastructures present (explored in section 2.5) compared to other traditional data back-ends (e.g., local servers). This back-end platform is in charge of four relevant functions: data processing, data storage, event detection as well as alarm reporting (functional requirement 4). As this solution is Cloud-based, the Cloud Computing resources can be easily and automatically adjusted according to new application's demands, or as the application's requirements grow, without the farmer having to invest more money in back-end infrastructure (as mentioned in section 2.4).

Data Processing This component is the first one that analyzes the collected data from the WSN infrastructure. Since data sent by the gateway comes from different source nodes, this component will be in charge of identifying and splitting the received data by the source node that collected it. It is also responsible to identify the type of data that has been received.

Data Storage This component is in charge of persistently storing the collected data from each WSN node. This way, the system has a historical data structure since the system deployment. For instance, this feature can be used by the end-user in order to consult historical information (e.g., the distance travelled by an animal as well as its path, the pastures that have been used, interactions between animals, etc.), based on a temporal resolution specified by it (e.g., hour, day, week, month, etc.).

Event Detection This component is responsible for the detection of certain business specific events. This way, this Cloud component will be running algorithms for analysing the collected data to detect the occurrence of certain events (e.g., a possible animal health disease, an animal that crosses a geo-fence, etc.). These algorithms will depend of the business model.

Alarm Reporting This component is responsible, given the detection of a particular event, to send an alarm to the farmer about the detected event, for instance through email or SMS service (depending on the end-user's habits and preferences).

The Cloud Computing platform will also be in charge of sending to the end-user digests (e.g., via email) containing a summary of the monitorization done during a period of a day, week, month, etc.

3.3.3 Web Application

The end-user application will correspond to a Web application. In order to get access to the system information, the application will connect to the Cloud platform and will make requests to it. Using

this application, the farmer can visualize the last reported state of the system and of its nodes, consult historical information that have been saved, see the events that have been detected by the platform as well as interact with the system (functional requirement 3).

3.4 Summary

In this chapter it was proposed a generic architecture to be applied in the Livestock monitorization, focused in particularly in the collection of the geographic location of the animals. In a very simplified approach, in these kinds of monitorization systems the information is collected by each WSN's node and, after going through several layers of intelligence in order to perform data processing (see section 2.1), it can be finally delivered to the end-user. Based on this assertion, it is possible to extract the three major system layers: the WSN infrastructure, the Cloud Computing platform as well as the Web application.

It is possible to explain in a briefly way the proposed arquitecture simply basing on the path that the information must travel, from its "birth" in the WSN's node until it is finally "consumed" by the farmer. Firstly, a WSN's node, which corresponds to an animal that is grazing in the pasture field, turns on its sensors, senses its geographic location and stores it on its buffer. When this node has connectivity with a sink node, it will send the collected information to the sink. In order to accomplish this task, the data packet is sent in a multi-hop approach, from animal to animal through a short-range radio (e.g., for instance using ZigBee technology) until it reaches the network's sink node. Finally, when the information reaches the sink node, it is sent to the back-end infrastructure through its long-range radio interface. For instance, if there is a cellular service covering the region, it may be used the GPRS technology to upload the data to the back-end infrastructure, thus this way the sink node has also the gateway network role. The back-end infrastructure corresponds to a Cloud Computing platform that will receive the information (data processing component), stores it (data storage component) and finally analyses it running event detection algorithms (event detection component), which may request to send an alert to the farmer (alarm reporting component), if certain events were detected. After the data were correctly stored in the Cloud platform, the farmer can finally consult this information through an application accessed by an Web browser that connects to the Cloud Computing platform.

Chapter 4

Implementation

In this chapter it will be described in detail the implementation of the system that follows the concepts presented in chapter 3. In section 4.1 it will be briefly described the operational context for the system, as well as relevant information collected during the interviews with experts who work directly in the Livestock area. In section 4.2 it will be described the implementation of the various components of the WSN that enabled building the first functional prototype of the WSN's nodes. In section 4.4 it will be briefly described the structure of the Cloud Computing platform used as data back-end as well as its components that support the WSN. In section 4.5 it will be presented the Web application connected with the Cloud Computing platform that will allow the end-user to visualize the actual state of the system as well as to interact with it. Finally, this chapter is finalized in section 4.6 where it is presented the main guidelines and conclusions of this chapter.

4.1 The Context

As mentioned above in chapter 3, the system's architecture is intended to be generic, although its implementation decisions depend directly on the real application scenario where the system will be deployed. In this context, this project has the collaboration of the enterprise *Sensefinity*®[1], which is a startup company that provides M2M solutions as a service to other companies or partners in the most varied areas, allying the development of hardware and software. This collaboration allowed to support the development of this project. The enterprise *YDreams Robotics*[2] has also supported this project and allowed to test and validate the first prototype in a real application scenario, using the network of Fundão's *Fab Lab Aldeias do Xisto*. In this context, the company *Queijaria Ribeira de Alpreade*[3], located in the Portuguese village named Zebras near Fundão, has immediately demonstrated interest in this project in order to innovate the monitorization done on its assets.

A requirement analysis was made in the *Queijaria Ribeira de Alpreade*'s offices involving various experts in this field that work directly with the animals or who are directly involved in the management

[1] `http://www.sensefinity.com` (accessed last time on 19th December 2014)
[2] `http://www.ydreamsrobotics.com` (accessed last time on 19th December 2014)
[3] `http://www.ribeiradealpreade.com` (accessed last time on 19th December 2014)

activities of the business. During this phase of interviews with professionals of this sector, it was also made a field visit to the Livestock farm where the prototyping would be done that allowed to have a reliable overview of the physical conditions that would have to be supported by the system. These direct contact with professionals also allowed a better understanding of how actually works in real life the management and monitorization of the animals and, as the farmers will be the future users of this system, the identification of its real problems and necessities was vital to have a more complete list of requirements.

Through the interviews it was identified that the monitorization will be made in dairy sheep that are usually freely grazing in a large dimension pasture (tens of hectares) with cellular service covering the region. They spend most of their time alternating between grazing and resting/ruminating, presenting slow movements (except during short stress periods). Sheep graze in cohesive groups and isolation is a source of stress to them. They are social animals and form strong social hierarchies between them. Generally, in the top of the hierarchy there are some leading sheep that influence the other animals, particularly their grazing movements, therefore the other sheep tend to follow all their decisions. Usually, the farmers tag these leading sheep with a bell on their collars to easily locate them. Besides this, in order to identify unequivocally the animals, each sheep has a ear tag attached to it with an identification number.

Hence, it was identified the main functionalities that needed to be supported, according to priority requirements defined by the staff for the first system prototype, which are illustrated in figure 4.1. Although these functionalities are specific to this pilot company's interests, some of them are transversal to the branch of the farm animals monitorization and even common to other sectors of our society:

1. Determine the current position of an animal based in the last reported position;

2. Build the route taken by a sheep as well as by the whole flock of sheep, to clearly identify the pastures used as well as the ones with better quality;

3. As the sheep are freely grazing sometimes they surpass the farm limits, therefore it is intended to have a geo-fencing functionality that sends an alarm to the farmer staff indicating this occurrence;

4. Inform the farmer when the system detects a cut in the collar, indicating a possible theft;

5. Alterations in the normal bustle of an animal may indicate to the farmer valuable informations, therefore the farmers want to be informed when an animal presents movement values that are significantly above or below the mean for each individual animal, taking into account the history of that specific animal;

6. As when an animal picks a particular infectious disease it is necessary to slaughter the whole flock of sheep, therefore it is very interesting to infer the direct/indirect interactions between the various animals, based on the historical distance between them, in order to isolate only the potentially infected sheep, thereby avoiding slaughtering healthy ones that were not in danger of contagious.

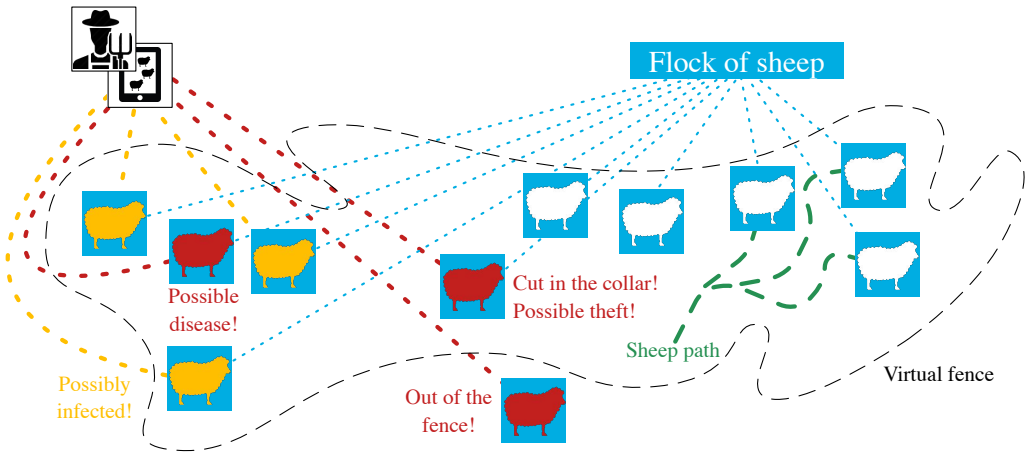


Figure 4.1: The use-case illustrated

4.2 WSN Infrastructure

The WSN infrastructure will follow the concepts presented in chapter 3. The primordial goal of the WSN is to deliver the geographical information sensed by each node of the network to the data back-end. This geographical information will be sensed through GPS receivers. As mentioned in the section 4.1, during the interviews with the farm staff it was identified that some sheep assume a leadership behaviour over the rest of the flock of sheep (they are usually tagged with a bell on their collars). These leader sheep will correspond to the sink nodes of the WSN as they tend to have more sheep close to them. The other sheep will correspond to mesh nodes that will send its data through multi-hop to the sink nodes, using for that a short-range radio. As the region is covered by a cellular service, it was chosen the GPRS technology to send the collected network information to the data back-end. Therefore, using the GPRS module, the sink nodes of the network will also play the role of gateway nodes.

Due to the natural mobility of the animals, the flock of sheep may become partitioned in various sub ad hoc networks, which may not have connectivity with the sink nodes. In order to overcome this fact and as all the sheep will be equipped with the ultimate version of the first prototype that includes a GPRS module, the mesh nodes could turn temporarily into sink nodes when they detect that its sub network has no connectivity with any sink node. This way the information collected by the network may continue to reach the data back-end infrastructure through the recently elected sink node.

It was decided that the WSN nodes structure and functionality will be as simple as possible as they have very limited resources. Indeed, having a Cloud-based platform as data back-end it is possible to transfer all the complexity to it, avoiding to overload the WSN. A low complexity WSN node solution potentially allows future cost reductions associated with the conception of the nodes. In figure 4.2 were identified some vital components for a network node to guarantee the system's goals. Each one of these components will be meticulously explained throughout this section. Note that as illustrated in the figure 4.2, each network node will be equipped with a short-range radio as well as a GPRS module. At the logical level only the current sink nodes will turn on its GPRS module in order to send the network data to the back-end infrastructure.

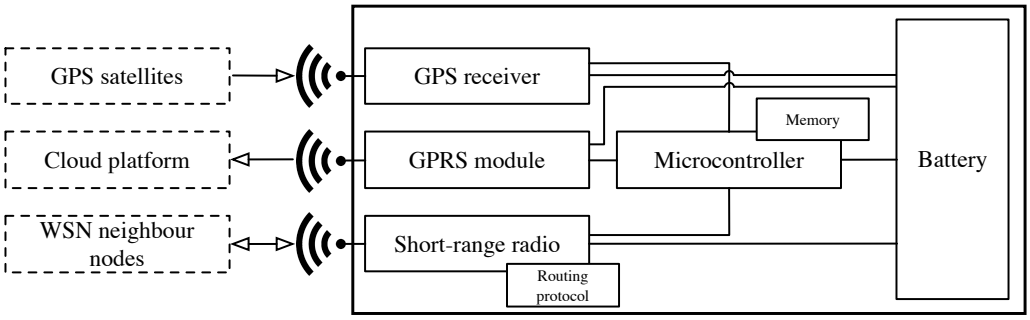


Figure 4.2: WSN node's components schematic

4.2.1 *MSP430* Microcontroller

The microcontroller unit (also known as MCU) is in a central position of the network node's structure, managing all the activity on the node. This important component is the "brain" of the network nodes and it is in charge of interacting with the other components that are connected to the node (e.g., GPS receiver, short-range radio as well as the GPRS module), through serial communication using an Universal Asynchronous Receiver/Transmitter (UART) unit.

There are two clock interrupts that allow the system to keep track of time, particularly to manage timeouts related to the operation of several system functions (e.g., collect information through a time-driven scheme, manage network operations, etc.). In order to have different time dimensions, it was programmed two clock interrupts that will be triggered approximately each millisecond and each second, respectively. The interruption that will be triggered each second is always enabled and it will allow to determine the next time to measure and collect the geographical information and battery status of the node as well as to run periodically other methods related to the management of the network configurations (which will be further explained in section 4.3). The timer that is triggered each millisecond is only enabled when it is necessary to wait a certain timeout related to some network operations of the low-range radio (e.g., transmit a network packet, receive an ACK packet, etc.).

In remote and automated applications it is essential that the system itself has the capability to detect and recover from any possible malfunction that affects its normal operation. Particularly in the Livestock area where the monitorized assets correspond to mobile nodes, this type of fault recovery is crucial. Therefore it was programmed a Watchdog Timer (WDT) that it is periodically restarted, in order to avoid that it expires. If a hardware or software malfunction occurs that precludes the MCU to restart the WDT, it eventually will expire and in consequence the MCU will restart the system.

In this project it was decided to use the *MSP430* family MCUs from *Texas Instruments* that provide a very low power MCU solution to portable applications. In the next subsections it will be described the *MSP430* MCU versions that were been programmed during the development of the first system node prototype. The MCU programming was made in the C programming language and using an Integrated Development Environment (IDE), which in this case was the *Code Composer Studio* by *Texas Instruments*.

4.2.1.1 *MSP430F2274*

The *MSP430F2274* MCU [81] was used as an initial approach to the development of a solution for the network nodes. This MCU allowed to easily start to programme and test the various components of the network nodes, using for that the UART module available. The *MSP430F2274* MCU used was part of the *eZ430-RF2500* kit [82] (illustrated in figure 4.3), which besides the MCU includes also the *CC2500* radio (presented and described in 4.2.2). This kit includes an Universal Serial Bus (USB) debugging and programming interface that allows the *MSP430F2274* MCU to send and receive data from a PC, using for that the MCU UART. The *eZ430-RF2500* kit also provides a battery expansion board that needs two AAA batteries. The *MSP430F2274* MCU has the following main technical features:

- 16-bit Reduced Instruction Set Computing (RISC) architecture;
- 32KB of flash memory;
- 1KB of RAM;
- 2 16-bit timers;
- 1 UART module;
- V_{in} range from 1,8 to 3,6VDC.

Figure 4.3: *eZ430-RF2500* development kit

The microcontroller was configured to work essentially in the Low-Power Mode 3 (LPM3), disabling the CPU, remaining uniquely active the Auxiliary Clock (ACLK), which is sourced by the Very Low Oscilator (VLO). The low power consumptions of this clock source make it suitable for the system. This component was employed to programme the two 16-bit timers, which are incremented according to this oscillator, triggering each one of them an interruption that will allow the MCU to periodically wake up from LMP3 and execute some operations depending on its current state.

4.2.1.2 *MSP430F5419A*

The *MSP430F5419A* [83] (illustrated in the figure 4.4) was used as the final MCU solution for the first prototype of the network nodes. The technical features of the *MSP430F5419A* MCU are stated below.

- 16-bit RISC Architecture;

- 128KB of flash memory;
- 16KB of RAM;
- 3 16-bit timers;
- 4 UART modules;
- Real-Time Clock (RTC) included;
- V_{in} range from 1,8 to 3,6VDC.

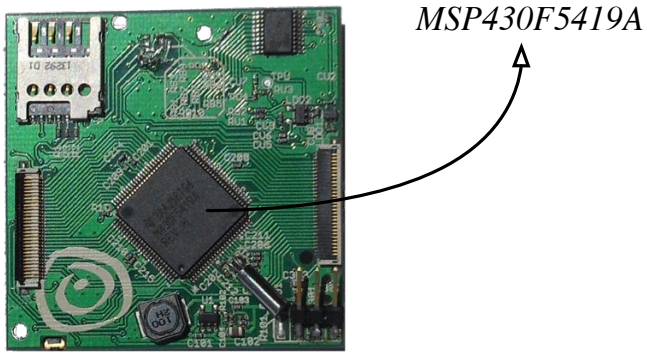


Figure 4.4: *MSP430F5419A* module built-in the *Butterfinger*® board

The *MSP430F5419A*'s available RAM compared to the *MSP430F2274* MCU is larger, allowing to easily buffer the network data packets on each WSN node. The RTC module is also a positive point of the *MSP430F5419A* MCU. This module was used to keep track of time in order to manage the operation activities on the node (e.g., collecting information from the sensors, etc.). This RTC module bring to the system a greater clock precision, which makes it a suitable solution. As also occurred in the *MSP430F2274* MCU, this RTC module will allow the MCU to periodically wake up from LMP3.

4.2.2 *CC2500* Radio

The transceiver module used for short-range communications between the network nodes inside the ad hoc network was the *CC2500* radio [84] (illustrated in figure 4.5) from *Texas Instruments*. The routing protocol (described in detail in the section 4.3) was designed and tested using this transceiver model. The CC2500 transceivers work on the 2,4GHz band and offers low-cost and low-energy consumptions for wireless applications. The circuit was designed for the 2400-2483,5MHz Industrial, Scientific and Medical (ISM) and Short Range Device (SRD) frequency bands. This radio allows to several possible configurations, which are easily changed. The *CC2500* radio has the following main technical features:

- Frequency range between 2400-2483,5MHz;
- Sensitivity up to -104dBm;
- Programmable output power from -30 to +1dBm;

- Programmable data rate from 1,2 to 500kBaud;
- Fast startup time; up to 240us from sleep to receiving/transmitting mode.

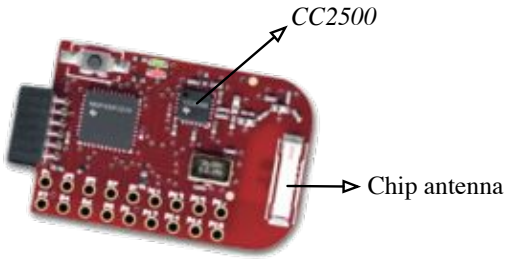


Figure 4.5: *eZ430-RF2500* development tool module

The radio output power level can be easily configurable during the normal execution of the network node through the *PATABLE* register. The list of recommended *PATABLE* settings for the possible transmission power levels is presented in table A.1 for $+25^{o}$C and 3,0V.

The CC2500 radio has one 64-byte FIFO (First In, First Out) buffer for receiving network packets (i.e., the Rx FIFO buffer) and another one to transmit them (i.e., the Tx FIFO buffer). The packet handler push/pull the packets to/from the FIFO buffers and perform configurable functions over the network packets, such as filtering, appending status bytes, etc. Every packet is framed in fields with 1 byte of size. The format of the transmitted network packets is configurable, but typically it is based on the format presented in figure 4.6. In this figure, the preamble size is 4 bytes, the sync word size is 16 bytes and it is transmitted twice, the packet length is variable and the Cyclic Redundancy Check (CRC) calculation is enabled. Every transmitted packet will be validated by the receiver with the CRC field using the 2 extra Frame Check Sequence (FCS) bytes transmitted by the sender. This way the packet handler of the receiver will automatically discard the packet if any error is detected in the received packet.

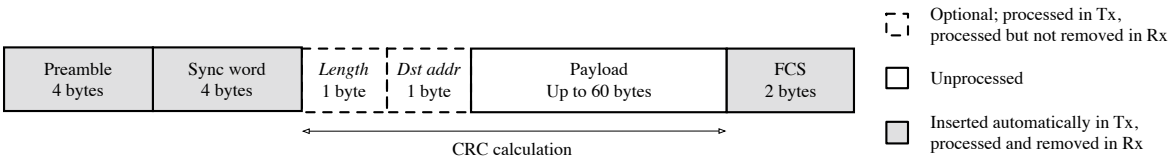


Figure 4.6: Typical transmitted packet format

The received data packet after being processed and framed typically follows the format presented in figure 4.7, when the 2 bytes status in the end of the packet are enabled. This two bytes status are appended by the packet handler and correspond to 1 byte for the RSSI, 7 bits for the Link Quality Indicator (LQI) as well as 1 bit that indicates the CRC check result.

The radio configurations were defined with the help of the *SmartRF Studio* software developed by *Texas Instruments*, particularly in the definition of the radio configuration register values. It was chosen the CRC mode activated, packet length variable and a data rate of 250kBaud.

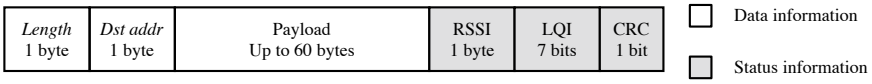


Figure 4.7: Typical received packet format after being processed

4.2.3 *SIM908* GPRS Module

The GPRS module used to communicate the information collected by the WSN to the data back-end infrastructure was the *SIM908* module [85] (shown in figure 4.8) by *SIMCom Wireless Solutions*. The GSM/GPRS u.FL antenna used is also presented in figure 4.8. It was required a micro-SIM (Subscriber Identity Module) card connected to the module. The *SIM908* unit corresponds to a quad-band GSM/GPRS module that combines these technologies with an integrated GPS receiver, in order to provide an attractive solution for tracking applications. Focusing now on the GPRS component of this module, its technical specifications are presented below.

- Quad-band 850/900/1800/1900MHz;
- GPRS multi-slot class 10;
- GPRS mobile station class B;
- Control via ATtention (AT) commands [86];
- 1 UART module;
- GPRS supply voltage range between 3,2 and 4,8V.

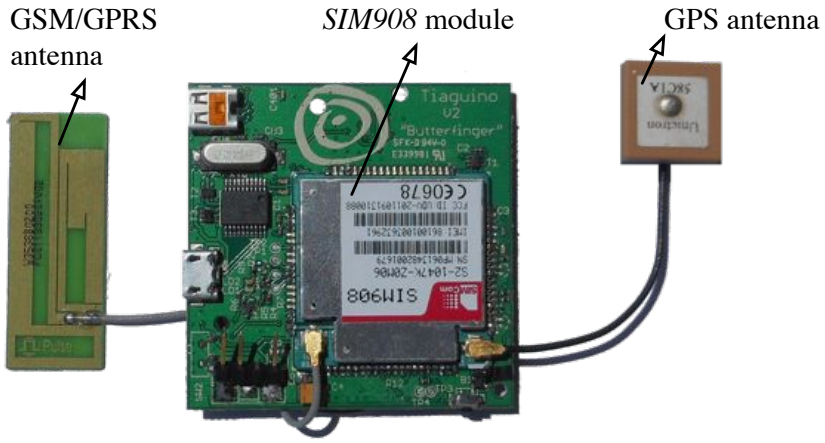


Figure 4.8: *SIM908* module built-in the *Butterfinger*® with a GSM/GPRS and GPS antennas

The *MSP430F5419A* MCU (presented in section 4.2.1.2) will control the *SIM908* module through its UART unit by transmitting AT commands to it following an internal state machine. An AT command must have the prefix "AT" at the begining and must terminate with a Carriage Return (CR) and a Line Feed (LF) control characters. The responses to the AT commands begin and terminate with CR and

LF control characters. The sequence of AT commands to send the collected information from the WSN to the data back-end is presented in table 4.1. Notice that each time an AT instruction is sent to the *SIM908* module a timeout is triggered (the timeout value depends on the specific AT command sent), after which the AT command is retried if the expected answer is not received. It was decided that every AT command is retried at most 5 times, after which the *SIM908* module is restarted.

AT Command	**Description**
AT+GSN	Get the International Mobile Equipment Identity (IMEI) number
AT+CREG?	Returns the network registration status
AT+SAPBR=3,1,"APN","*apn*"	Configures the Access Point Name (APN)
AT+SAPBR=1,1	Opens the GPRS bearer
AT+HTTPINIT	Initializes the HTTP service
AT+HTTPPARA="CID",1	Sets the Connection ID (CID) parameter
AT+HTTPPARA="URL","*url*"	Configures the Uniform Resource Locator (URL)
AT+HTTPDATA=*size*,*timeout*	Input the HTTP data
AT+HTTPACTION=1	Sets the HTTP method action (POST in this case)
AT+HTTPREAD	Reads the HTTP response
AT+HTTPTERM	Closes the opened HTTP session

Table 4.1: Sequence of AT commands used by the GPRS module

Data packets stored in the packet queues (presented on section 4.3.1.1) will be sent each T_{SIM908} (equals to 30 minutes) or each time there is a total of at least 10 packets in all the queues. Therefore, if one of these two conditions is verified, the *SIM908* module is powered on by setting the pin *P8.5*'s output to high level during 1 second. Only the current sink nodes will verify the aforementioned conditions in order to forward the WSN collected information to the final data back-end infrastructure. When the module is on, the MCU will start sending AT instructions to it, related to the initial configurations of the *SIM908* module, in order to configure the GPRS profile, register the module on the network and open an HTTP session.

After the initial configuration phase is over, the MCU will send the data collected from the WSN in an HTTP POST packet. After the HTTP has been initialized and the URL of the data back-end has been specified, the MCU will consult the packet queues in order to select the ones that will be sent. The packet queues will be scanned starting from the most priority packet queue to the least priority packet queue. During this scan, the packets will be read one by one and the message body will be constructed following the data formation imposed by the *Columbus*® model, which corresponds to the message structure that the Cloud platform (presented in section 4.4) is expecting. Note that each HTTP POST message will have a *Columbus*® header indicating, among other information, the number of sub-messages that the packet carries as well as the sub-messages that correspond to the packets collected by the WSN stored in the packet queues.

Once the header has been constructed and the data packets have been formatted to the *Columbus*® model, this binary information will be represented in an American Standard Code for Information In-

terchange (ASCII) string using a *Base64* encoder. The output string is included in the HTTP POST body and the POST request is executed. If the *SIM908* module returns a *200 OK* response, the data packets that were successfully sent are deleted from the packet queues and the *Base64* format response returned by the Cloud platform is read, after being decoded. After that, the MCU sends an AT command instruction in order to close the opened HTTP session and put the *SIM908* module in standby mode by setting the pin *P8.5*'s output to high level during 1 second.

4.2.4 Data Sensing

The information collected by each network node concerns location and battery sensing. In the following subsections it will be explained how the network nodes will collect each one of them.

4.2.4.1 Localization

Each WSN node has to be capable of reporting its current location to the data back-end. In order to accomplish this system requirement, it is necessary to adopt one localization method (see section 2.3.2.5). In order to provide location data each node was equipped with a GPS receiver.

As mentioned in section 4.1, sheep spend most of their time alternating between grazing and resting/ruminating states, presenting slow movements. Therefore it was decided that collecting periodically location information between 3 and 15 minutes (T_{GPS}) would give the system an acceptable representation of the movements and trajectory made by each sheep, and in consequence by the flock of sheep. This value is parameterized, therefore could be easily changed in the future if needed. As the GPS receivers are power-hungry, the system has to operate the GPS receiver on a duty cycle, putting it on a standby mode, and only turning it on when the geographic information is really needed. If the timeout for collecting the GPS information has been triggered and there is no available space in the respective packet queue to store it, the packet queue will be verified again after 60 seconds, and if after this time there is available space, the GPS receiver will be turned on. It was also defined that the network node will wait T_{fix} (equals to T_{GPS}) in order to get a valid fix, after which it will put the GPS receiver in standby mode again. This value is also parameterized. If the network node gets a fix before the timeout to get a valid fix expires, it will build a GPS data packet with the collected information (pushing it to the right priority queue to be sent later) and put the GPS receiver in standby mode.

A representative quantity of GPS receivers available on the market provide the collected information through serial communication, using for that the standard developed by the National Marine Electronics Association (NMEA), called NMEA 0183 [87]. NMEA 0183 devices are designated as talkers or listeners, being possible to have a single talker and several listeners on one circuit. In this specific case the talker will correspond to the GPS receiver and the listener will correspond to the MCU, which will process the information sent by the GPS sensor. Through this standard it is possible to obtain periodically sentences that contains information collected by the GPS receptor. Therefore, using this standard all the data transmitted by the GPS receiver will be transmitted in the form of sentences. Only printable ASCII characters are allowed as well as the CR and LF control characters. The general format of the sentences is presented below [87].

```
$ttsss,d1,d2,...<CR><LF>
```

According to the NMEA 0183 standard, each sentence starts with the '$'. The first two characters ("*tt*") following the '$'are the talkier identifier. The next three characters ("*sss*") are the sentence identifier, followed by the data fields separated by commas. If the data field is not available, the delimiting commas are still sent, with no space between them. It is possible to have a checksum field, which is optional, identified by the character '*'followed by two hex digits corresponding to the exclusive OR of all characters between, not including the '$'and'*'. Each sentence ends with the CR and LF control characters. Figure 4.9 shows an NMEA 0183 output example obtained from the *Adafruit Ultimate GPS* board (which will be later presented) through serial communication. It is possible to view in the figure the sequence of output lines sent by the *Adafruit Ultimate GPS* board when it is obtaining a valid GPS fix.

```
$GPRMC,111733.456,V,,,,,0.01,0.00,201014,,,N*4B
$GPRMC,111734.456,V,,,,,0.02,0.00,201014,,,N*4F
$GPRMC,111735.456,V,,,,,0.06,0.00,201014,,,N*4A
$GPRMC,111736.456,V,,,,,0.09,0.00,201014,,,N*46
$GPRMC,111738.000,A,3844.8209,N,00909.0101,W,0.33,0.00,201014,,,A*7F
$GPRMC,111739.000,A,3844.8188,N,00909.0067,W,1.36,171.15,201014,,,A*72
$GPRMC,111740.000,A,3844.8179,N,00909.0041,W,2.18,131.58,201014,,,A*74
$GPRMC,111741.000,A,3844.8171,N,00909.0065,W,5.87,82.39,201014,,,A*44
```

Figure 4.9: NMEA 0183 output from a *Adafruit Ultimate GPS* board

The geographical location collected by the network nodes is represented in a geographic coordinated system, where each location can be represented using latitude and longitude coordinates (2D fix) or using these two coordinates plus the elevation (3D fix). Besides the latitude and longitude coordinates needed to get a 2D fix, the network node also collects the current date and UTC time when the location was obtained, in order to associate them to the collected location. These parameters are obtained from the Recommended Minimum Navigation Information (RMC) sentences of the NMEA 0183 standard. The typical format of this type of sentences sent by the GPS sensors is presented below [88], being the description of each field stated in the table 4.2.

```
$ttRMC,hhmmss.sss,s,ddmm.mmmm,a,.dddmm.mmmm,b,x.xx,zzz.zz,ddmmyy,x.xx,n,m*cc<CR><LF>
```

All the sentences sent by the GPS receiver will be processed by the MCU. It was implemented a function responsible to receive this information, compare the data presented in the sentence with the checksum and parse all the information, being only used the latitude, longitude, UTC time as well as the date fields of the RMC sentence, in order to construct the GPS data packets. The format of the GPS data packets is presented in figure 4.10. These packets will have associated to it the medium priority level assigned in the *priority* field of the data packets (see figure 4.20). The date is parsed in its subcomponents (i.e., year, month and day), which are put in one packet field each. A similar approach is followed for the UTC time, being the subcomponents the hour, minute and second when the location information was sensed. Relatively to the latitude and longitude coordinates, following the sentence structure presented in the table 4.2 it is necessary two components to represent each one of them. The first component is the

Sentence Field	Example	Description
*tt*RMC	GPRMC	GPS RMC protocol header
hhmmss.sss	083951.487	UTC time
s	A	Status; 'A'= data valid, 'V'= data invalid
ddmm.mmmm	3844.8189	Latitude
a	N	Hemisphere; 'N'= north, 'S'= south
dddmm.mmmm	00909.0009	Longitude
b	W	Hemisphere; 'E'= east, 'W'= west
x.xx	0.05	Speed over ground in knots
zzz.zz	165.48	Course over ground in degrees
ddmmyy	190314	Date
x.xx	3.05	Magnetic variation in degrees
n	W	Related to the magnetic variation; 'E'= east, 'W'= west
m	A	Mode; 'A'= autonomous, 'D'= differential, 'E'= estimated
cc	2C	Checksum

Table 4.2: RMC sentence fields description

value related to the degrees and minutes of the location (e.g., *ddmm.mmmm*) and the second one is the hemisphere indicator (e.g., 'N'). Relatively to the latitude value it is necessary 2 fixed digits of degrees, 2 fixed digits of minutes and a variable number of digits for decimal fractions of minutes, whereas the longitude value follows the same model excepting the degrees that are represented with 3 fixed digits. As the point in the first component is always in a well known position, it is possible to remove it when transmitting, being put back in the data back-end infrastructure. The first component of the latitude (i.e., *ddmm.mmmm* and *ddmmmmmm* without the point) will be parsed to an integer number with 32 bits (as it is necessary a maximum of 27 bits to represent it) and the longitude's first component (i.e., *ddmmm.mmmm* and *ddmmmmmmm* without the point) will be also parsed to an integer with 32 bits (as it is necessary a maximum of 30 bits to represent it). In order to optimize the size of the packets it was decided to combine these two components by transform the first component into a negative number if the hemispheres were the south (i.e., 'S') or west (i.e., 'W') or transform it into a positive number if the hemispheres were the north (i.e., 'N') or east (i.e., 'E'). Therefore, the latitude and longitude are sent using 4 fields of the data packet each.

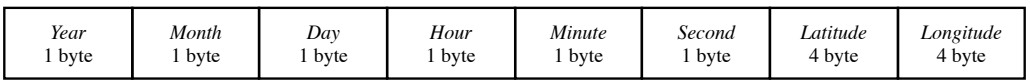


Figure 4.10: GPS data packet format

When the GPS data packet is delivered to the Cloud platform, the latitude and longitude coordinates have to be converted into decimal degrees (*dd.dddddd*). The conversion follows equation 4.1. In the case of the south (i.e., 'S') or west (i.e., 'W') hemispheres, the decimal degrees will assume a negative number.

$$dd.dddddd = dd + \frac{mm.mmmm}{60} = \left[\frac{ddmm.mmmm}{100}\right] + \frac{ddmm.mmmm - \left[\frac{ddmm.mmmm}{100}\right] \times 100}{60} \quad (4.1)$$

Adafruit Ultimate GPS As a first approach, it was used the *Ultimate GPS* board [89] (presented in figure 4.11) from *Adafruit*, in order to start collecting the geographical information of the network nodes. This board was built around the *MTK3339* GPS module that has a standard ceramic patch antenna included. The *Adafruit Ultimate GPS* board has the following typical main technical features:

- Sensitivity up to -165dBm in tracking mode and up to -145dBm during cold starts;
- Update rate from 1 to 10Hz;
- NMEA 0183 output;
- V_{in} range from 3,3 to 5,5VDC;
- Current consumptions up to 25mA in acquisition mode and 20mA during navigation.

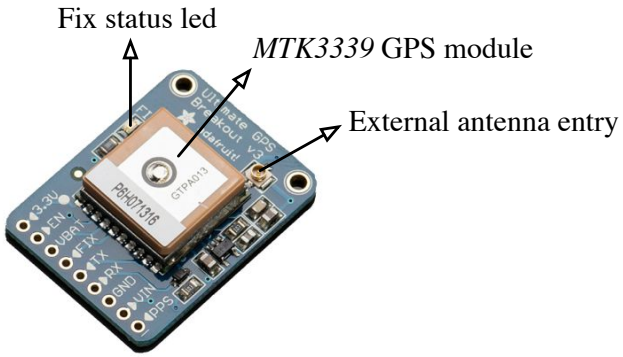


Figure 4.11: *Adafruit Ultimate GPS*

This GPS board was connected to the *MSP430F2274* MCU (previously presented in section 4.2.1.1) from the *eZ430-RF2500* development tool in order to collect the GPS sensed data, parse it and construct the GPS data packets to be later sent to the sink node. The experiments with these two components allowed to start testing the GPS data collection and parsing operations. These two components were connected following the schematic presented in figure 4.12. As the minimum voltage required by the GPS receiver is 3.3VDC and as the *eZ430-RF2500* development tool may be sourced by 2 AAA batteries (which in the maximum gives 3,0VDC in the case of being 2 AAA alkaline batteries), therefore was necessary to connect the GPS receiver to an external battery source (e.g., an 3,7V lithium-ion battery). The communication between the *Adafruit Ultimate GPS* board and the *MSP430F2274* MCU was made through its UART module.

The *MTK3339* GPS module was programmed to send only RMC sentences through the serial communication with 1Hz of periodicity. The settings of the *MTK3339* module was changed using PMTK

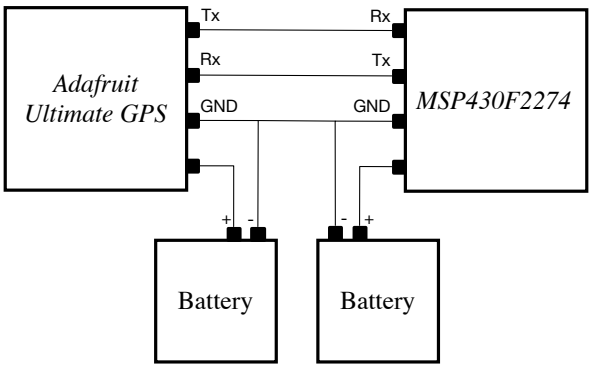


Figure 4.12: *Adafruit Ultimate GPS* with the *MSP430F2274* MCU schematic

commands [90]. In order to periodically collect GPS information, the *MSP430F2274* MCU sends each T_{GPS} an empty string to the *Adafruit Ultimate GPS*, in order to turn on the GPS receiver. After the GPS receiver has a fix or the timeout T_{fix} to get a valid fix has been expired, the *MSP430F2274* MCU will send the a PMTK standby command, in order to put the GPS receiver in a standby mode for power saving.

***SIM908* GPS Module** The final node solution employs the GPS module of the *SIM908* board (presented in section 4.2.3) in order to collect the geographical information of each node. The GPS u.FL passive antenna used is illustrated in figure 4.8. This GPS module has the following typical main technical features:

- Sensitivity up to -160dBm in tracking mode and up to -143dBm during cold starts;
- NMEA 0183 output;
- V_{in} range from 3,0 to 4,5VDC;
- Current consumptions up to 77mA in acquisition mode and 76mA during navigation.

The *MSP430F5419A* will control the GPS module of the *SIM908* board through AT commands. The typical sequence of AT commands that is made by a network node in order to get a valid GPS fix is illustrated in the table 4.3. The MCU will ask the *SIM908* GPS module for a new RMC sentence each 5 seconds, until a valid fix is obtained. As presented above the MCU will start the GPS module each T_{GPS} and it will stop it when a valid fix is obtained or the timeout T_{fix} to get a valid fix has been expired.

AT Command	**Description**
AT+CGPSPWR=1	Power on the GPS module
AT+CGPSRST=1	Resets the GPS module in autonomy mode
AT+CGPSINF=32	Gets a GPS RMC sentence
AT+CGPSPWR=0	Power off the GPS module

Table 4.3: Sequence of AT commands used by the GPS module

4.2.4.2 Battery

In order to have a remote overview of the current battery state for each node, as well as the evolution of the node's energy consumption during the runtime of the network, each WSN node has to report its current battery level to the data back-end infrastructure. Therefore it is possible to change in advance the batteries once they report low energy levels.

Each node has to report once a day its current battery level ($T_{battery}$). This value is parameterized, therefore could be easily changed in the future if needed. If the timeout for collecting the node's battery information has been triggered and there is no available space in the respective packet queue to store it, the packet queue will be verified again after 60 seconds. If after this time interval there is available space, the battery information will be collected. The format of the battery data packets is presented in figure 4.13. The battery data packets will have associated to it the low priority level assigned in the *priority* field of the data packets (see figure 4.20). The *battery level* field of the battery data packet will contain the remaining capacity of the node's battery expressed in percentage (ranging from 0% up to 100%). This information will be obtained by sending the AT command stated in table 4.4.

AT Command	**Description**
AT+CBC	Gets the battery information

Table 4.4: AT command used by the battery application

Besides this field, the network node will also send the date and UTC time when the battery information was collected. The date and UTC time information are collected from the GPS receiver, when a valid fix is available. Therefore, whenever the timeout for collecting the node's battery is triggered, the battery information is only collected when a GPS valid fix is available in order to add the temporal information associated to the time instant of the battery data acquisition. Therefore, the *SIM908* module is powered on not only to make a battery measurement, but it also waits for a new GPS data collection to perform the two measurements at the same time.

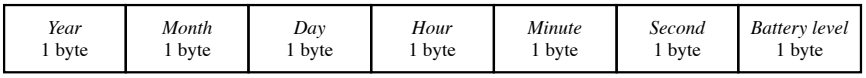


Figure 4.13: Battery data packet format

4.2.5 The Ultimate Node Solution

Combining all the approaches stated throughout this section it was possible to reach a final node solution, culminating this way with the final functional prototype for the WSN's nodes. This ultimate solution is illustrated in figure 4.14. As illustrated in the figure, the hardware is held in a waterproof plastic protective case.

Firstly and focusing exclusively on the hardware it was used the *eZ430-RF2500* module illustrated in figure 4.5 combined with the second generation node solution from *Sensefinity*®, called *Butterfinger*®

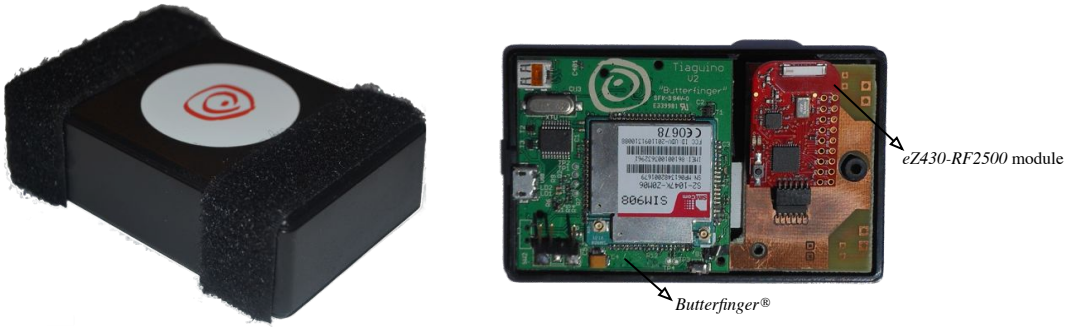


Figure 4.14: The ultimate solution; a) protective case; b) the hardware disposition

(shown in figures 4.4 and 4.8). The module is sourced by a single 880mAh rechargeable lithium-ion battery with 3,7V.

The *Butterfinger®* (already in a beta testing version) combines the *MSP430F5419A* MCU solution (presented in section 4.2.1.2) with the *SIM908* module (described in section 4.2.3). Therefore using the *SIM908* it is possible to upload the collected WSN information to the data back-end infrastructure through its GPRS module and, at the same time it allows to sense the current geographic information of the node through its GPS unit. As the GPS and battery information of the node are extracted using requests to the *SIM908* and, as this module is also used in order to upload the network information to the Cloud platform, it was necessary to synchronise all the operations made on the *SIM908*, managing the access of the various applications to it.

As the *Butterfinger®* has not a short-range radio for communications inside the WSN, it was necessary to combine it with an *eZ430-RF2500* module, which besides the *CC2500* radio (described in section 4.2.2) also contains the *MSP430F2274* (presented in section 4.2.1.1). Although this approach does not correspond to an ideal solution as it consumes more energy and increases the overall system complexity, it was necessary to use this module and in consequence the extra MCU provided by it, as it was not possible to use the *CC2500* radio in a stand-alone approach. The *eZ430-RF2500* module will be only in charge of the short-range radio operations, using the network protocol described hereafter in section 4.3 and being the received packets transferred to the packet queues located in memory in the *MSP430F5419A*. In consequence it was developed a plug and play protocol, called *SensePnPLayer®* (Sensefinity Plug and Play Layer®) that facilitates the discovery of a new connected hardware and the information exchange through the UART interface. In this case this mini protocol was used in order to exchange the data packets between the two MCUs as well as to inform the *MSP430F5419A* about the current radio configurations (e.g., the network address, the actual state of the radio, etc.).

4.3 Network Protocol

The developed network protocol, described in detail hereafter, corresponds to the network layer built upon the Minimal RF Interface (MRFI) layer, which simplifies the radio utilization by giving the necessary

support to interact with it. As there is no formal physical and data link layers, the *CC2500* radio will perform these functions, being the data received directly from the radio already framed.

This gradient-based routing protocol was designed for convergent traffic in mobile large scale networks where the data packets flow to sink nodes, which aggregate all the information collected by the network (similar to the protocols' approaches described in section 2.3.2.4). Hence, the purpose of this network protocol is to allow the upload of the collected network information to the final data back-end infrastructure, being neglected the opposite direction. Simplicity, robustness, fault tolerance, on-demand network adaptation and low message overhead were the main goals intended for this protocol, in order to yield an effective response to the non-functional requirements stated in section 3.2.

This protocol has two network phases; the first one corresponds to the network configuration phase responsible to informing to each network node the current one-hop neighbours connected to the sink nodes, in order to allow each one of them in the next phase (i.e., data exchange phase) to deliver data packets to the sinks. Note that this network protocol supports the presence of multiple sink nodes in order to give redundancy to the system. Many other gradient-based protocols are based in a broadcast approach in order to deliver its data packets to the sink nodes, in an attempt to increase the probability of a packet reaching the destination. However this solution applied to a large scale network will cause a high number of packets travelling in the network, increasing the collisions probability. Therefore it was chosen a single-path routing approach in order to forward the data packets to the sink nodes through the most adequate one-hop neighbour. It was also used a point-to-point message delivery confirmation mechanism.

All the network nodes have assigned to it a network address that is unique and identifies the WSN nodes on the network. This address can be manually assigned to each node or chosen randomly. As this first prototype is not supposed to address a large scale network, it was chosen 1 byte for the network address' length. Therefore the range of possible network addresses will be from 1 to 254, being the address 255 reserved for broadcast. The network address range space can be easily extended in the future, when the number of connected devices in the network justify it. With only one byte for representing the address of the network nodes, it is possible to decrease the size of the network packets, being only needed one field in the packet to carry one network address, which is an advantage in this phase.

4.3.1 Network Configuration Phase

This initial phase of the network protocol is responsible to configuring the network nodes. This way each node will be informed about the network topology, in particular their vicinity nodes and the position of the sink nodes in relation to it. This configuration phase will have primordial importance for the decisions made in the data exchange phase.

The beginning of this phase has origin in the sink nodes. These nodes periodically broadcast network configuration packets (called *CONF* packets) in order to announce its presence to its vicinity nodes that may be other sink nodes or mesh nodes. This periodical transmission of *CONF* packets is justified by the mobility of the network nodes. Since they move regularly it is necessary to refresh periodically the network nodes information. On the scope of this thesis, the nodes present slow movements, so it was

decided that each node currently acting like a sink has to propagate a *CONF* packet each T_{sink} time interval (equals to 60 seconds). This value is parameterized. For networks with topologies changing more frequently it may be advisable to increase the frequency for the transmission of these packets. Naturally, as the sink nodes can only communicate directly with their one-hop vicinity nodes, it is necessary that its neighbours propagate the *CONF* packet sent by the sink node, broadcasting this packet to other network nodes. Therefore, this *CONF* packet will progressively reach all the network nodes that have obviously network connectivity among them.

In the *CONF* packets there is a field with great importance in the configuration of the network nodes, which is the *height* field (see *CONF* packet format in figure 4.15 and in table 4.5). This field indicates the distance of a network node to a given sink node. This way all the sink nodes send the *CONF* packets with the *height* field equals to 0, whereas subsequently all the intermediate nodes will register this field and broadcast this *CONF* packet with its *height* field incremented in one. For instance, when a network node receives a *CONF* packet with the height of 2, it means that this *CONF* packet has travelled through 3 hops until reaches it (see the example in figure 4.16). If it only receives this *CONF* packet, it means that the minimum distance that this node is from the sink node is 3 hops and the neighbour that forwarded this packet is the only neighbour that provides connectivity with the sink node (at least according to the received information). Such information will allow the network nodes to know the neighbours that have the shortest hop distance from a given sink, therefore it will allow in the future to make well informed decisions about the most adequate neighbours in order to communicate with the sinks, minimizing this way the number of transmission hops.

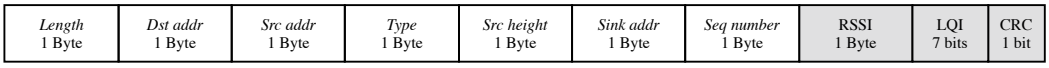


Figure 4.15: *CONF* packet format

Packet Field	**Description**
Length	Length of the *CONF* packet (equals to 6)
Dst addr	Address of the destination node (equals to 255)
Src addr	Address of the last sender
Type	Type of the *CONF* packet (equals to 0)
Src height	Height of the last sender
Sink addr	Address of the sink node
Seq number	Sequence number assigned by the sink
RSSI	RSSI value
LQI	LQI value
CRC	CRC check status (0 in the case of error and 1 if OK)

Table 4.5: Description of the *CONF* packet fields

As mentioned before, this protocol is based on a single-path for data packet delivery. Therefore it is necessary that each node has a neighbours table storing information on one-hop neighbours. This infor-

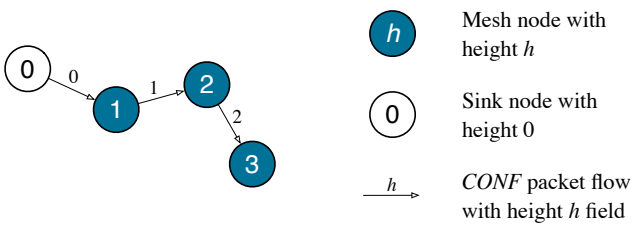


Figure 4.16: *CONF* packet network flow

mation will support future routing decisions made in the data exchange phase. The packet reception rate (PRR) is at least 85% when the RSSI is above the threshold of -87dBm [91]. Therefore, only neighbours that sent *CONF* packets with RSSI values above -87dBm were admitted in order to maintain reliable network links. Besides this table, there is another one that maintains information about the accessible sink nodes that will help to perform management operations related to the network configuration phase. These two tables will be filled during this phase, although they can be subsequently refreshed by listening for packets travelling in the network.

It is now relevant to explain how the network nodes manage and synchronize between them the *CONF* packets' sending/receiving actions for network configuration/reconfiguration. In this phase there are some few states associated to a network node, as explained hereafter. Of notice that, for each *CONF* packet originated from a given sink, each mesh node only propagates once the *CONF* packet (i.e., only transmits once), even if it has received more than one *CONF* packet from other mesh nodes connected to that sink. Note also that as the data packets will be transmitted following a single-path approach and not following a broadcast model, each mesh node is only interested in knowing information about the neighbours that have less height, in order to minimize the number of hops for data packet transmission to the sink. This way, alternative paths involving more intermediate nodes will be disregarded, whereas if there is an obstruction in the shortest path (e.g., a mesh node has moved away) it will be solved in the next network reconfiguration. As there is no strict latency requirements, this approach is acceptable. Furthermore detecting alternative paths would require more processing and would spend more memory on each node for storing it.

In order to explain the states that will be mentioned below, it is presented in figure 4.18 a network configuration phase example timeline for the network topology with depth 3 (shown in figure 4.17.

Idle This is the initial state of each mesh node in the network configuration phase. Whenever a mesh node receives a *CONF* packet, firstly it will check if the *CONF* packet's RSSI value is above the RSSI threshold (i.e., -87dBm) and if its sequence number is greater compared to the last *CONF* packet received from that sink (in order to discard old and duplicated *CONF* packets). If the *CONF* packet has passed these two tests, it means that a network configuration/reconfiguration has been initiated by a specific sink. After this, if the node receives another *CONF* packet from other sink it will drop the packet until it finishes the current network configuration phase. The first step is refreshing its actual view of the network topology in relation with the sink node that transmitted that *CONF* packet. Therefore, it will refresh the sink nodes table and delete this sink from the list of connected sinks (if present) of each

neighbour entry on the neighbours table. Besides these, it will be added the neighbour that sent the CONF packet to the neighbours table. Afterwards the node will wait for more *CONF* packets coming from other one-hop neighbours connected with this sink. As the maximum backoff time that each node delays its transmission is T_{max} (equals to 50 milliseconds), it will have to enter in the *listen more CONF packets* state and wait for T_{max} milliseconds plus a guard-time value of T_{guard} (equals to 5 milliseconds).

Listen more *CONF* packets When a mesh node receives a *CONF* packet in this state from the previous sink node it will add the neighbour that sent this *CONF* packet to the neighbours table. When the 55 milliseconds had passed since it received its first *CONF* packet from that sink, it will finally enter in the *waiting backoff time* state in order to send the *CONF* packet. Note that if the mesh node verifies that there is no available memory space to save more data packets, it will not propagate the *CONF* packet because it cannot handle more data packets from its neighbours. Therefore in this case, it will not announce its presence and so the network configuration phase will end in this point for it, returning this way to the *idle* state.

Waiting backoff time A network node enters this state to propagate a *CONF* packet through the network. Therefore, this is also the initial state for the sink nodes. Sink nodes verify if there is available memory in order to save more received data packets. If not, it will wait to send its stored data packets to the data back-end, in order to free some memory. Afterwards, the sink can finally return to this state and announce its presence to the network. The node have to wait a backoff time chosen randomly over a given time interval between 0 and T_{max} milliseconds. This avoids therefore that network nodes receiving the same *CONF* packet start to propagate it at the same time, decreasing this way the probability of collisions. In this state each received *CONF* packet will be dropped. After waiting the backoff time, the node can finally enter in the *broadcast CONF packet* state.

Broadcast *CONF* packet In this state the node transmits the *CONF* packet using the maximum transmission power in order to reach more neighbours. The *source address* field in the *CONF* packet will be filled with the network address of the sender and the *source height* will correspond to the minimum *CONF* packets' height that it received from that sink plus one (with the exception of sink nodes, for which this field is always 0). On a sink node, the *sink address* field will be filled with its network address and the sequential number will be equal to the last *CONF* packet sent plus one. These two fields together (*sink address* and the *seq number*) will identify unequivocally the *CONF* packet in the network, maintained untouchable by the remaining nodes. After sending it, the mesh node returns to the *idle* state. In the case of the node being a sink, it will wait for the next *CONF* packet transmission (i.e., after T_{sink}).

When a mesh node receives a *CONF* packet, it indicates that there is connectivity with that sink through the neighbour that propagates the *CONF* packet, which may be an intermediate mesh node or the sink node itself. As the *CONF* packets are sent by the sink nodes each T_{sink} seconds, if a mesh node does not receive another *CONF* packet from that sink after that period plus a guard-time value of 15 seconds it will consider that it becomes disconnected from that sink. Therefore, it will delete it from

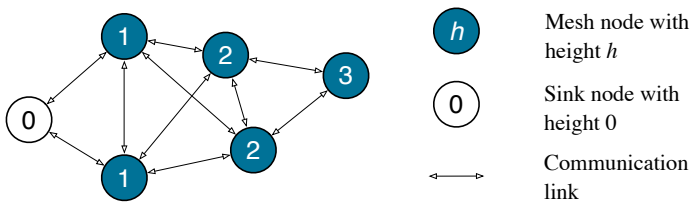


Figure 4.17: Network topology with depth 3

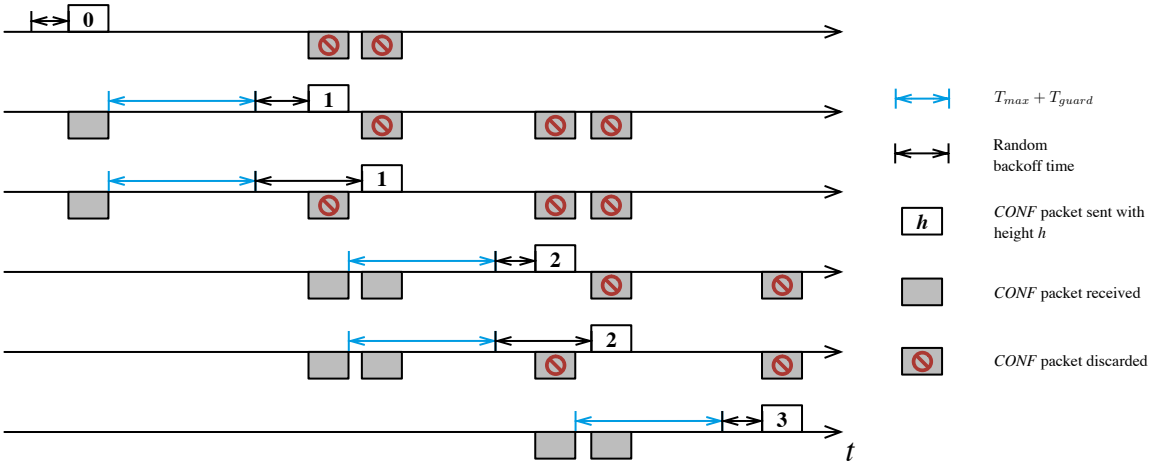
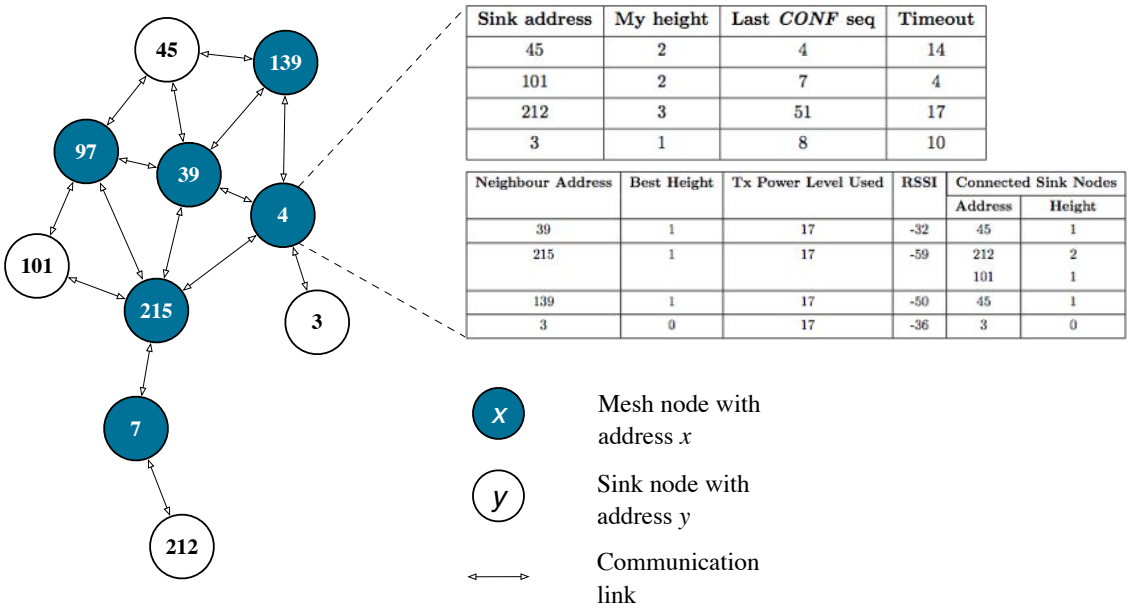


Figure 4.18: Network configuration phase timeline for the network presented in figure 4.17

its sinks list and refresh the neighbours table, deleting that sink from the list of connected sinks of each neighbour entry on the table. In figure 4.19 is illustrated an example of the tables that a network mesh node possibly has after receiving all the *CONF* packets originated by the sink nodes of the given network.

Network Nodes Adaptability If a mesh node does not have connectivity with any sink node during $T_{sink} \times 5$ seconds, it will turn temporarily into a sink node serving its disconnected ad hoc subnetwork. Therefore, the information collected by this subnetwork can continue to be transmitted through it to the back-end infrastructure. This adaptation capability of the network nodes is very interesting when the sheep move away from the flock, creating separated ad hoc subnetworks. However this adaptation will have a little nuance. Each mesh node will wait $T_{sink} \times 5$ seconds plus $5\times$ the height that it has in relation to the last sink that it had connectivity. Therefore, the mesh nodes that were previously closer to that sink will have first the opportunity to adapt themselves to sink nodes. After a mesh node temporarily turn into a sink node, it will enter in the *waiting backoff time* state in order to start propagating *CONF* packets. If a mesh node receives a *CONF* packet indicating connectivity with a sink node, it will suspend its adaptation process or turn back into a mesh node if it had already turned into a sink node. For instance, taking into consideration the example presented in figure 4.19, if the mesh node with address 4 becomes disconnected from all the sink nodes and if the last sink connected was the sink with address 212 (as it would be expected if for each entry in the sinks table the *timeout* field was decremented each second), this mesh node would have to wait $T_{sink} \times 5 + 5 \times 3$ for a new *CONF* packet. If this waiting time



Sink address	My height	Last *CONF* seq	Timeout
45	2	4	14
101	2	7	4
212	3	51	17
3	1	8	10

Neighbour Address	Best Height	Tx Power Level Used	RSSI	Connected Sink Nodes	
				Address	Height
39	1	17	-32	45	1
215	1	17	-59	212	2
				101	1
139	1	17	-50	45	1
3	0	17	-36	3	0

Figure 4.19: Mesh node networking tables

was reached and no *CONF* packet was received it would turn into a sink node, until the connectivity with a sink was again reestablished.

4.3.1.1 Data Exchange Phase

In this network phase each mesh node will transmit in a multi-hop approach its data packets to a sink node. In the data exchange phase, the sink nodes only receive data packets from other mesh nodes. The collected information is immediately saved, in order to be sent later to the data back-end infrastructure. In order to start sending its data packets, the mesh nodes must have connectivity to at least one sink node. In the absence of network connectivity they have to wait for a *CONF* packet. while buffering the data packets in memory for later transmission. Besides sending its own data packets, the mesh nodes will forward the data packets arrived from other mesh nodes. If the mesh node receives a *CONF* packet indicating a new configuration/reconfiguration of the network during the data exchange phase, it will suspend the data transmission temporarily, resuming it later. On the other hand, if a network node is in the network configuration phase and receives a data packet addressed to it, the node will save the packet in memory and confirm immediately the reception of the packet to the transmitter, remaining always in the network configuration phase.

The format of a *DATA* packet is presented in figure 4.20 and the description of each of its fields is explained in table 4.6. The address of the node that collected the information (and so that originated the *DATA* packet) as well as the packet sequence number identify the *DATA* packet and make it unique on the network. This way these two fields (*first src addr* and the *seq number*) are maintained untouchable until the packet reaches the sink node. The *DATA* packet sequence number will be filled by the node that collected the information, therefore each time a node collects information and wants to send it in a new *DATA* packet it will increment its sequence number in one. The *type* field depends on the type

of information that the *DATA* packet carries, being always higher than 1. All the *DATA* packets have associated to it a certain priority level. This way it is possible to define a fast network propagation message model, forwarding first the packets with more priority. This priority level could be one of three; *low*, *medium* or *high* priority, associated with the values 2, 1 and 0, respectively. This priority is assigned depending on the importance of the packet's information, therefore depends on the data type associated to it. In order to store the *DATA* packets for later transmission, there are three packet queues, each one associated to a certain priority level, thereby each one will only store packets of a given priority type. Each queue will follow a FIFO approach, therefore the packets in a given queue will be transmitted in the order in which they have been inserted. The first served queue will correspond to the one that has higher priority packets.

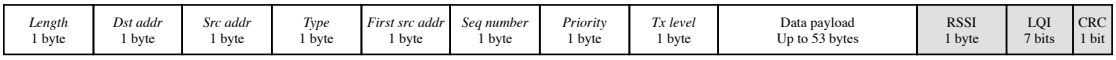


Figure 4.20: *DATA* packet format

Packet Field	**Description**
Length	Length of the DATA packet (equals to 7 plus the payload)
Dst addr	Address of the destination node
Src addr	Address of the last sender
Type	Type of the packet (depends on the data type; higher than 1)
First src addr	Address of the node that collected the information
Seq number	Sequential number of the DATA packet
Priority	Priority of the packet (depends on the data type)
Tx level	Tx level used by the last sender
Data payload	The payload organization depends on the data type
RSSI	RSSI value
LQI	LQI value
CRC	CRC check status (0 in the case of error and 1 if OK)

Table 4.6: Description of the *DATA* packet fields

The *DATA* packets could come from other network nodes through the multi-hop radio transmissions or result from the network node itself during data sensing operations. Independently of this fact, every time a *DATA* packet is successfully added to the *DATA* packet queues, the dispatch phase will be started in order to send it to the sink node (for mesh nodes) or saved in memory to be sent later to the data back-end (for sink nodes). In order to start sending packets, each mesh node will determine the most adequate node to forward its *DATA* packets. The information required for this node selection is stored in the neighbours table (filled in the network configuration phases). The criterion used was the shortest number of hops. In order to accomplish this, all the neighbours will be compared and it will be chosen the one with smaller height (i.e., the one-hop neighbour that is closer in hops to a sink node). If there are multiple neighbours with the same height, it will be chosen the one with a lower RSSI in the table.

For selected nodes having equal values of height and RSSI, it will be selected the node that first appears on the table. After this selection the *DATA* packet will be transmitted after a random backoff time of at most 50 milliseconds.

As mentioned above this routing protocol is based on a point-to-point message delivery confirmation. In order to accomplish this, after the packet sender transmits the *DATA* packet, it will wait T_{ACK} (equals to 50 milliseconds) in order to receive an *ACK* packet from the packet receiver, indicating successful packet reception. The *ACK* packet format is presented in figure 4.21 and the description of each field is specified in table 4.7.

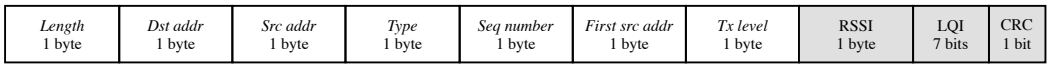


Figure 4.21: *ACK* packet format

Packet Field	**Description**
Length	Length of the ACK packet (equals to 6)
Dst addr	Address of the destination node (i.e., the node that sent the DATA packet)
Src addr	Address of the sender
Type	Type of the ACK packet (equals to 1)
First src addr	Address of the node that collected the information of the received DATA packet
Seq number	Sequential number of the previously received DATA packet
Tx level	Tx level used by the sender of the received DATA packet
RSSI	RSSI value
LQI	LQI value
CRC	CRC check status (0 in the case of error and 1 if OK)

Table 4.7: Description of the *ACK* packet fields

If the sender of the *DATA* packet does not receive the expected *ACK* packet in T_{ACK}, it will retry to send it to another neighbour (this time to the second most adequate neighbour following the same criteria stated above) after a random backoff time of at most 50 milliseconds. The maximum number of retries defined for a *DATA* packet was 6. Therefore, if the sender has less than 6 neighbours, when all the neighbours were tried it will retry to send it again to the most adequate neighbour, and so on. When the maximum number of retries were reached and the *DATA* packet was not been transmitted, the sender will keep it in the queue and wait for 15 seconds in order to repeat the transmitting process again, with the packet retries reseted. When the *DATA* packet sender finally receives the ACK packet indicating success in the *DATA* packet transmission, it will send the next highest priority packet to the last successful destination, in the case of there being another *DATA* packet to be sent. The fact of sending the packet to the last successful destination (which has received successfully the last *DATA* packet) allows to avoid sending it to neighbours who had not previously received successfully other *DATA* packets in the past. On the other hand, this approach tends to overload the last successful destination node and thus not dividing the load across the available network neighbours. A new packet destination will be

searched again when a new *DATA* packet retry was made or when a new *CONF* packet was received (indicating a network reconfiguration). Note also that every time a *CONF* packet is received it means that the network tables were recently refreshed, therefore each mesh node after terminated the network configuration phase, it will start the dispatch process and verify if there is any *DATA* packet to be sent in its queues and if so, it will send it.

In figure 4.22 are presented two flowcharts related with the *DATA* packets pickup phase and with the *DATA* packets dispatch phase.

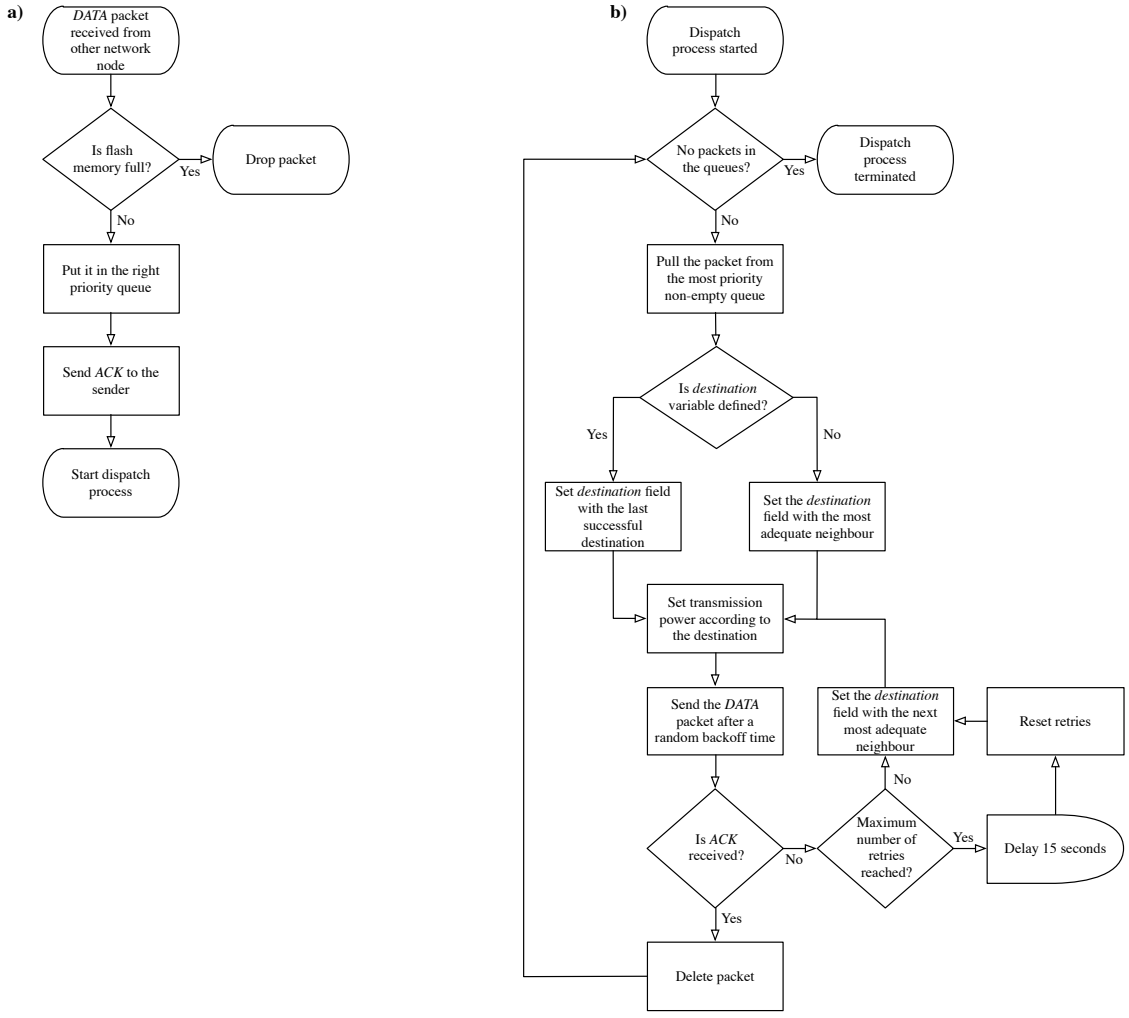


Figure 4.22: Data exchange phase; a) flowchart of the pickup phase; b) flowchart of the dispatch phase

Signal Strength Adaptability Adapting the communication signal strength when transmitting *DATA* packets may have two positive overcomes on the network and in particular in the network nodes themselves. The first benefit is saving energy; if it is known in advance the transmission power necessary to send a *DATA* packet to a given node, it is not necessary to transmit it using the maximum transmission power, therefore it is possible to save energy, optimizing this way the transmission. The second advantage is related to the network interference; if the transmission is made using less power it will be limited to a more reduced range area and thereby the number of devices involved will be lower, decreasing this way

the collisions occurrence probability.

The information that is necessary to estimate the transmission power needed is already stored in the neighbours table (during the network configuration phases), not being required more additional information for that. This information corresponds to the statistic information related to the last packet received from a given neighbour, more specifically it corresponds to the RSSI as well as the transmission power level used by the node transmitter, stored in the *RSSI* as well as in the *tx power level used* fields respectively in the neighbours table. The transmission power level used is extracted in the *tx level* field present in the *DATA* and *ACK* packets and it is known to be the maximum level in the *CONF* packets. Note that any time a packet is received (independently of being directed to a specific node or not), the node will search if that neighbour is on its list and if so it will refresh these fields in the respective table entry. This refreshing feature will enable to update these status information between networking configuration phases, allowing to adjust the signal strength proactively according to possible changes in the network context.

With only the RSSI value and the transmission power used it is possible to know the difference of power (ΔP) between the received power and the transmitted power (see equation 4.2). Knowing the ΔP value based on real measurements and the radio sensitivity (it will be equal to -89dBm for the chosen configurations) it is possible to determine the transmission power needed in order to communicate with a given network node. However, it is necessary to take into account fading effects that may damage the wireless signal. Therefore, it is necessary to add a certain amount of extra signal, called fade margin (see equation 4.3 [92]). In order to ensure reliable link performance, most wireless equipment manufacturers recommend a minimum fade margin of at least +10dB [92]. Based on these facts it is possible to deduct the inequation presented in 4.4. Therefore, each node of the network will adapt its transmission signal strength to the minimum power (between the transmission powers presented in table A.1) that satisfies the presented inequation, setting the register *PATABLE*.

$$\Delta P = P_{transmitted} - P_{received} \tag{4.2}$$

$$FadeMargin = P_{received} - RadioSensitivity \tag{4.3}$$

$$P_{adapted} - \Delta P > RadioSensitivity + FadeMargin \tag{4.4}$$

When a network node sends the ACK packet confirming the successful transmission of a *DATA* packet, it will send it using the same transmission power level stated in the *tx level* field of the received *DATA* packet. In the case of retransmissions, each time all the neighbour nodes have been tried, the transmitter will increase its transmission power level (calculated in function of the desired destination) multiplying it by 2. This increasing policy will be made until the packet was successfully transmitted or a new CONF packet has been received. Therefore, increasing gradually the transmission power in order to attenuate the nodes mobility effects will allow to search actively for the desired neighbour. This fact combined

with the passive refreshing of the neighbours information by listening the packets that travel through the network will allow to progressively approach to an optimum transmission power for a certain neighbour.

4.4 Cloud Computing Platform

The Cloud Computing Platform used was designed and it is maintained by Sensefinity® company. Although stable, this platform is currently under development. It is called *Machinates*® and it was programmed in Java upon the *Amazon Web Services* (AWS) provided by *Amazon*. This Cloud platform is based on a multi-tenant architecture.

The *Machinates*® is essentially based in four main logic layers; service, listeners, business as well as the model layer (see figure 4.23).

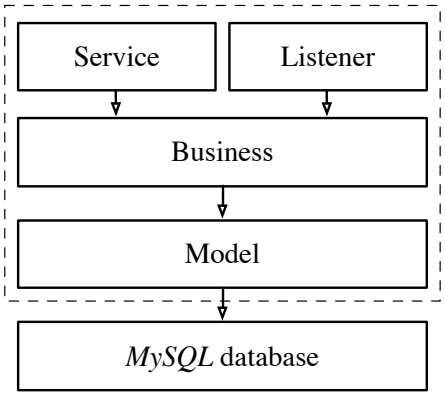


Figure 4.23: *Machinates*® back-end logic structure

The data is stored in a *MySQL* relational database that is directly accessed and managed by the model layer, the lowest layer in the *Machinates*® back-end logic structure. The business layer is the central part of the *Machinates*® and it is responsible to interpret and treat the information, knowing exactly the set of steps in order to accomplish the various system tasks. The service layer corresponds to the entry points for devices to communicate and interact with the system. The listener layer is responsible to periodically execute certain tasks in the system (e.g., in the end of the day calculate the distance that each animal moved during the day).

The *Machinates*® platform only understands the messages that are formatted following the *Columbus*® model. Therefore the WSN nodes have to send the information in *Columbus*® messages. The format of these messages is presented in figure 4.24 and the description of each message field is described in table 4.8. The device's GPS locations will be sent in *Columbus*® position messages and the device's battery information is sent in *Columbus*® measurement messages.

When a data message is delivered to the *Machinates*® back-end, it has to follow a set of steps until it is processed and archived in the database. The possible states that a message can assume in the *Machinates*® are represented in figure 4.25. Each time the *Machinates*® changes the state of a message, a timestamp is saved in the message indicating this state changing in order to maintain a historical view

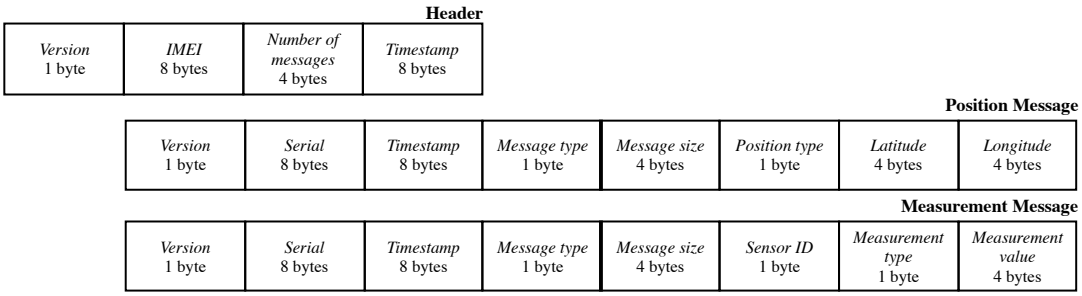


Figure 4.24: *Columbus*® messages format

of the messages.

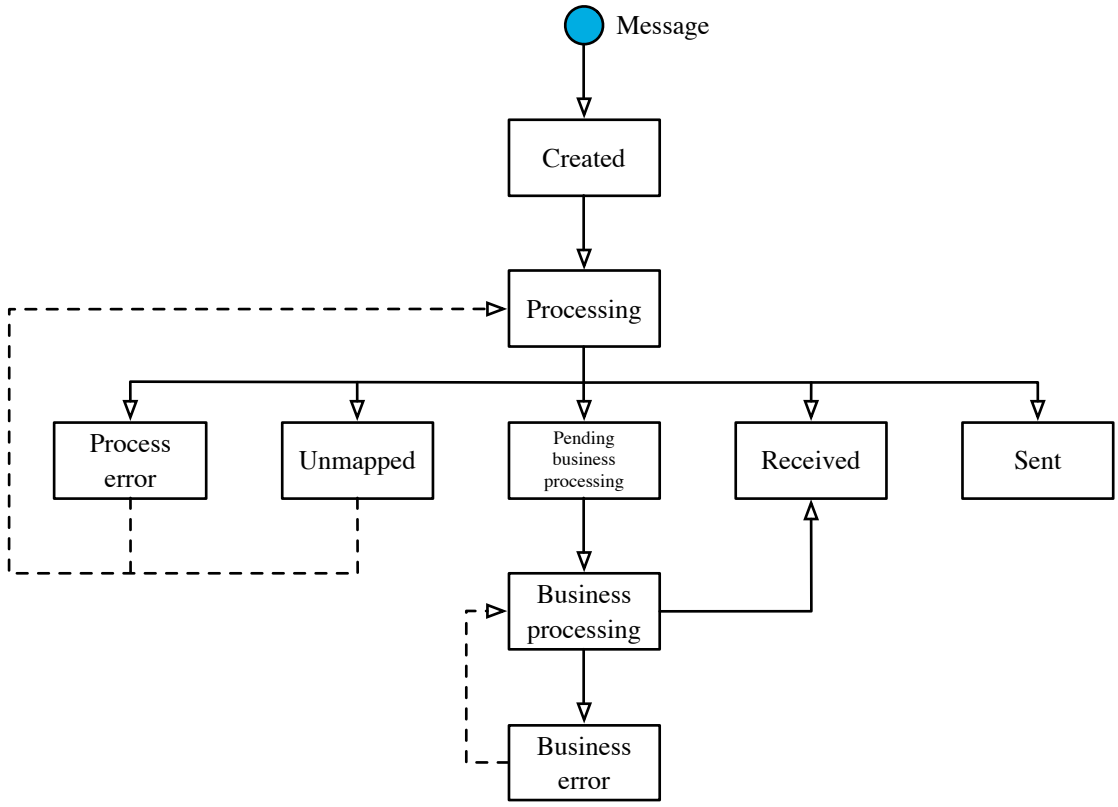


Figure 4.25: *Machinates*® data message's states

Whenever a device sends a data message to the *Machinates*®, it will be marked as *created* and placed in the queue to be later processed. The data message assumes the *processing* state whenever it is pulled by the *Machinates*® in order to be processed. If the message was successfully processed, it will be marked as *received*. If the information is not correctly formatted in the *Columbus*® model and it is not interpretable by the system, it will change its state to *process error*, or it is marked as *unmapped* if the message was originated by a non-listed device. In the case of business specific messages, they will be marked as *pending business processing* to be later processed by the specific business. In this case, the message after being processed may assume the *business error* state in case of some error occurrence or be

Type	Field	Description
Header	*Version*	*Columbus*® message version
	IMEI	IMEI of the device
	Number of messages	Number of sub-messages in the message
	Timestamp	Timestamp of the message
Position	*Version*	*Columbus*® message version
	Serial	Serial of the device
	Timestamp	Timestamp of the sub-message
	Message type	Sub-message type (equals to 5)
	Message size	Size of the sub-message (in bytes)
	Position Type	Indicates the type and format of the location
	Latitude	Location's latitude
	Longitude	Location's longitude
Measurement	*Version*	*Columbus*® message version
	Serial	Serial of the device
	Timestamp	Timestamp of the sub-message
	Message type	Sub-message type (equals to 6)
	Message size	Size of the sub-message (in bytes)
	Sensor ID	Identifies the sensor that collected the information
	Measurement type	Indicates the type of the collected measurement
	Measurement value	Value of the measurement

Table 4.8: Description of the *Columbus*® messages fields

marked as *received* otherwise. In the case of any error occurrence in the message processing, they may be processed again later. If one message is generated by the *Machinates*® back-end to be sent to a specific device, it will assume the *sent* state. In the case of a temporary malfunction of the *Machinates*® that does not allow to store the data messages in the queue to be later processed, the system will store them in the file system until the system recovers.

4.4.1 Smart Cloud Data Processing

Each time a position message is delivered to the *Machinates*® back-end, the geo-fencing algorithm will compare the new received animal's location with the existing virtual fence coordinates. Therefore, it is possible to determine if an animal is inside or outside the virtual fence, and based on that erase or trigger an alarm, respectively. In order to determine if an animal is inside or outside the virtual fence the geo-fencing algorithm is based on the Jordan Curve Theorem [93]. This theorem affirms that it is necessary to draw a straight line from a given point (i.e., the sheep's location) to the outside of the whole drawing (i.e., outside the virtual fence). If the line meets the curve (i.e., the virtual fence) an odd number of times, that point (i.e., the sheep) is on the interior, otherwise if the line meets the curve an even number of times, that point is on the exterior. Observing figure 4.26 it is possible to view an example of the application of the Jordan Curve Theorem. In this figure the line that intersects sheep *a*

meets the virtual fence line 3 times (i.e., an odd number of times), therefore this sheep is on the interior of the virtual fence. The line that intersects sheep b meets the virtual fence line 2 times (i.e., an even number of times), therefore this sheep is on the exterior of the virtual fence. The simplicity given by the Jordan Curve Theorem makes it a very interesting solution for the geo-fencing algorithm.

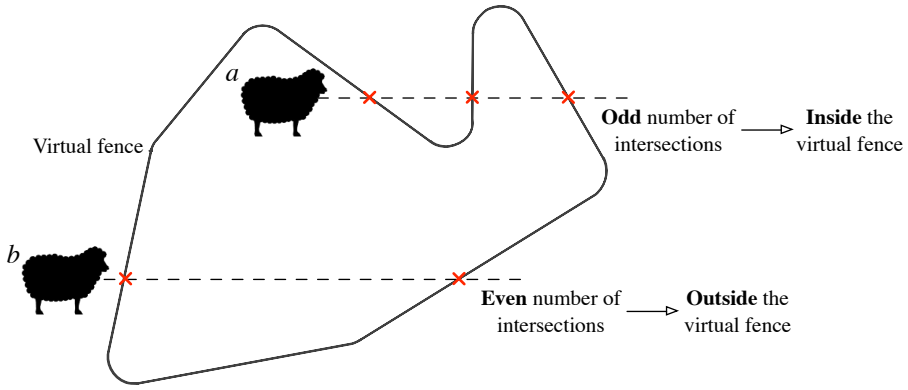


Figure 4.26: Jordan Curve Theorem illustrated

The system may detect variations in the normal bustle of the animals by comparing the total distance travelled by each animal at the end of the day with its historical mean. Therefore there is a process that runs always in the end of each day in order to make these calculations for each sheep of the flock. In order to estimate the total distance travelled by each sheep at the end of the day it is necessary to pick up each location sent by the animal on that day and sum each distance between two points. For instance in figure 4.27 it is illustrated a sheep that reported 4 locations (a, b, c and d). In this case, the total distance travelled by that sheep will be approximated by $d1 + d2 + d3$.

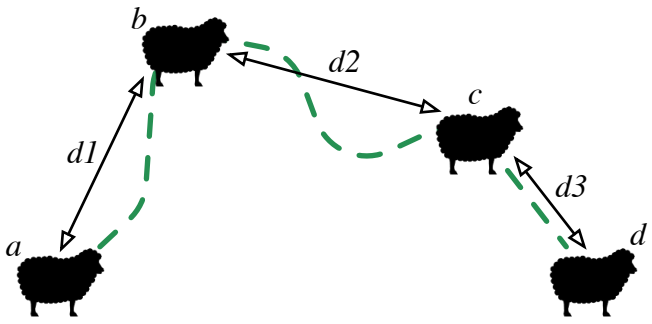


Figure 4.27: Total travelled distance calculation

In order to calculate the distance between two points it was used the Haversine formula [94] (see equation 4.5). In this equation the r corresponds to the Earth radius (i.e., 6371 km), the lat_{i-1} and lon_{i-1} to the latitude and longitude of the first point and the lat_i and lon_i to the latitude and longitude of the second point.

$$d = 2 \times r \times sin^{-1}\left(\sqrt{sin^2\left(\frac{lat_i - lat_{i-1}}{2}\right) + cos\left(lat_i\right) \times cos\left(lat_{i-1}\right) \times sin^2\left(\frac{lon_i - lon_{i-1}}{2}\right)}\right) \quad (4.5)$$

4.5 Web Application

The information collected by the WSN and stored in the *Machinates*® back-end can be consulted by the farmer through a GUI[4] provided by the *Machinates*® system. This GUI is also currently being developed by the Sensefinity® team. This GUI allows the end-user to consult the collected information, query historical data as well as to interact with the system.

The GUI is based in a Model–View–Controller (MVC) architecture and it is decoupled from the *Machinates*® back-end, being installed in the end-used browser and making requests to the *Machinates*® system as needed. It is used JSON in order to communicate with the *Machinates*®, being the actual state of the application not maintained by the system. The GUI uses the *MapQuest* API in order to represent the geographical information in maps.

In figures 4.28 and 4.29 it is illustrated the login page and the dashboard page of the *Machinates*® GUI, respectively.

Figure 4.28: *Machinates*® GUI login page

4.6 Summary

A prototype of the proposed architecture described in the section 3 was implemented. The WSN nodes were implemented around the low power *MSP430* family MCUs as well as the *CC2500* radio for the ad hoc communications inside the WSN infrastructure. The GPS and battery information are periodically sensed and uploaded to the Cloud platform through sink nodes that use the *SIM908* GPRS module. The Cloud platform that was used as the final data back-end was the *Machinates*® from Sensefinity®. This platform will allow to store and process the information collected by the WSN infrastructure. In order

[4] `http://machinates.sensefinity.com` (accessed last time on 19th December 2014)

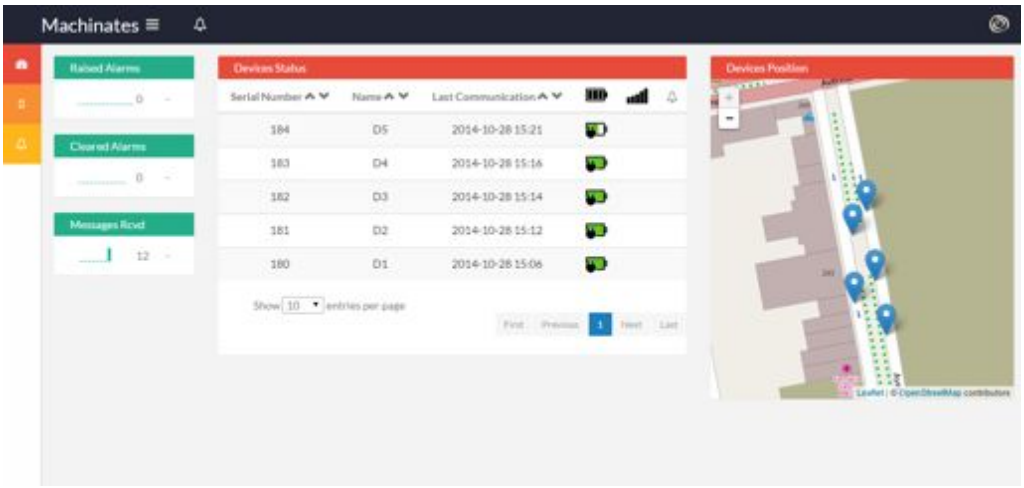


Figure 4.29: *Machinates*® GUI dashboard page

to consult this information as well as to interact with the system, a GUI provided by the *Machinates*® is also available. All the essential components of the system were successfully implemented, therefore allowing for a complete experimental evaluation of the system in a real application scenario.

Chapter 5

Evaluation

This chapter describes experiments carried out in order to evaluate the developed prototype. It describes the testing methodology, the experiments setup, and the obtained experimental results. Through the execution of a set of tests, it is intended to validate the system as well as to identify some adequate future system improvements that will be listed in section 6.1. Each part of the system will be first individually evaluated, and afterwards all system components are integrated together for an end-to-end evaluation, culminating this way with field tests in real application scenario cases.

5.1 System Components Testing

This section describes the set of tests for evaluating independently the WSN infrastructure as well as the Cloud platform. Given the importance of these two components, it is of foremost importance to examine their behaviour and verify some of their technical features.

5.1.1 WSN Infrastructure

Lets first present experimental testing of the WSN infrastructure. It was essentially tested two main aspects of the WSN; the periodic collection of the nodes location and its reporting through the network protocol presented in section 4.3 as well as the *CC2500* radio component.

5.1.1.1 Data Collection (Functional Testing)

As the system is based on the collection and reporting of the geographic location of the WSN nodes, the collection of this type of information and its reporting to a sink node were tested.

Goals The main goals of this experiment were to test:

- The periodical collection of the geographic information of a mobile WSN mesh node;
- The reporting of the collected information to a sink node using the developed network protocol.

Setup In order to execute this experimental test, it was programmed a mesh node that periodically powers on its GPS receiver each 30 seconds, in order to collect its current geographical location. It was also used a sink node that collected the information from the WSN, i.e. in this case from the mesh node. Both of the WSN nodes correspond to mobile nodes that moved side-by-side and separated from each other up to 4 meters. The WSN nodes correspond to the *eZ430-RF2500* kit's nodes (presented in section 4.2.1.1), being the mesh node connected to a battery expansion board and the sink node plugged to a laptop. The mesh node was also equipped with the *Adafruit Ultimate GPS* receiver (presented in section 4.2.4.1). The information reported by the mesh node to the sink node was visualised directly in the laptop, through the serial communication interface. The two nodes used the developed network protocol (presented in section 4.3) in order to communicate with each other.

Results This experiment test ran over 12 minutes and it was collected 24 location samples. During the experiment the WSN nodes had always in a continuous movement, doing a particular circular route. As the sink node and the mesh node were physically separated by a short distance, all the packets were successfully transmitted and so there was no packet loss. The geographic information was obtained in the NMEA 0183 format, thus it was subsequently converted and presented in a map, using for that *Google Maps*. Figure 5.1 illustrates a map with all the locations measured and reported by the mesh node, which corresponds to the actual route.

Figure 5.1: Geographic information collected and reported by a mesh node

5.1.1.2 *CC2500* Radio (Performance Testing)

Ad hoc communications functionality inside the WSN is an important concern for system design. Therefore the *CC2500* transceiver was tested in order to make more sustainable design choices.

Communication's Range The communication range is an important parameter when choosing a radio transceiver. It is important to evaluate in a real experiment how the RSSI varies with the variation of the communication distance.

Goals The main goals of this experiment were to observe:

- The variation of the RSSI according to variations on the physical communication distance between two transceivers;
- The maximum communication range between two transceivers that allow reliable data exchange.

Setup In order to execute this experimental test, it was necessary two *eZ430-RF2500* kit's nodes (presented in section 4.2.1.1), being one programmed as a mesh node connected to a battery expansion board and another being the sink node plugged to a laptop. These two nodes were deployed in an open space with the minimum possible electromagnetic interference and both 1 meter elevated from the ground. In this experiment the sink node was deployed in a fixed position, being the relative position of the mesh node in relation to the sink variated across time. The physical communication distance between this two nodes assumed 50 centimetres as well as 1, 2, 4, 8 and 16 meters. The two nodes used the developed network protocol (presented in section 4.3) in order to communicate with each other. The mesh node was programmed in order to send a network packet each 5 seconds to the sink node using its maximum transmission power level. The packets were received by the sink node and the RSSI information was visualized directly in the laptop, through the serial communication interface.

Results In each position it was considered the first 10 network packets sent by the mesh node, being the RSSI of the packet registered in the sink node. In every tested distance it was verified that there was no packet loss. In figure 5.2 it is presented a graphic illustrating the variation of the RSSI according to the physical communication distances tested. Table 5.1 reports the mean and the standard deviation of the RSSI values for each communication distance tested.

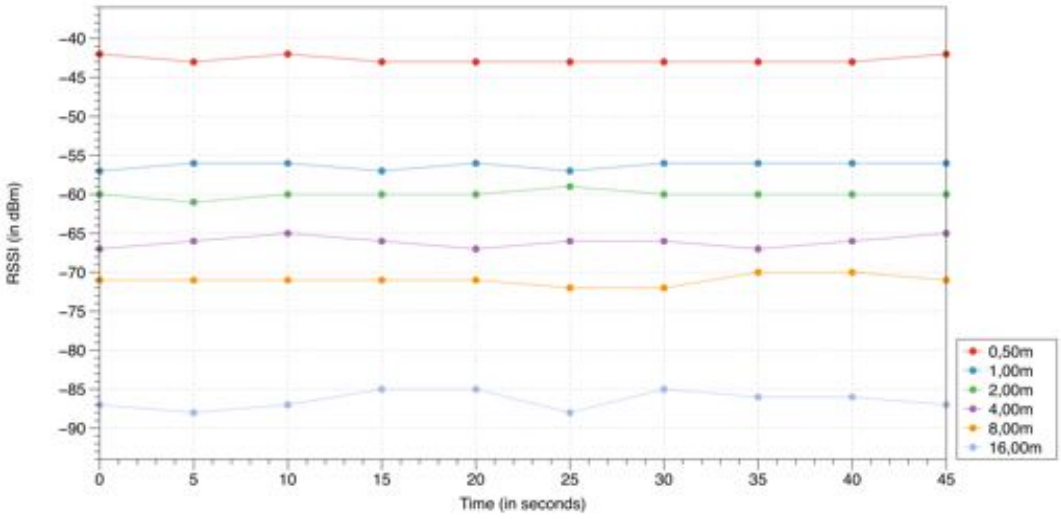


Figure 5.2: RSSI variation according to the communication distance

Observing figure 5.2 the RSSI decreases gradually as the physical communication distance increases, wherein the extremes correspond to the RSSI values for the 0,50m and 16,00m distances. This fact was expected, and it is evident in table 5.1 by analysing the RSSI mean parameter for each tested distance.

Distance (in meters)	0,50	1,00	2,00	4,00	8,00	16,00
Mean Value (in dBm)	-42,70	-56,30	-60,00	-66,10	-71,00	-86,40
Standard Deviation (in dBm)	0,48	0,48	0,47	0,74	0,67	1,17

Table 5.1: Mean and standard deviation of the data represented in figure 5.2

It is possible to observe also that the standard deviation values increase gradually as the communication distance increases. This observation can be justified by the fact that as the distance increases the communications become more unstable, therefore the obtained RSSI values are more susceptible for evidencing larger discrepancies, reaching its top on the 16,00m. It was concluded also that, for optimum transmission conditions, the 16 meters approaches the maximum reliable communication range of the *CC2500* transceivers using its maximum transmission power level and the *CC2500*'s configurations presented in the section 4.2.2. As presented in section 4.1, the sheep graze in cohesive groups, being these 16 meters a good indicator for this system as it is usually sufficient to keep the flock of sheep connected.

Signal Strength Adaptability Considering the advantages presented in section 4.3.1.1 related with the adaptation of the communication signal strength, it is important to verify the adaptation behaviour in a real experimental test and calculate the energy savings that may result from this feature in order to conclude about its effectiveness and real impact on the system.

Goals The main goals of this experiment were to verify:

- How the transceiver adjusts its radio communication signal strength according to variations on the physical communication distance between two transceivers;
- The actual impact in the system concerning the device's energy savings.

Setup The experimental setup used in this test was exactly the same as used in the previous setup scenario where the radio communication's range was evaluated. The sink node was programmed in order to send a *CONF* packet each 5 seconds. The mesh node transmits a network packet to the sink using for that an adapted transmission power calculated based on the information inferred through the reception of the last *CONF* packet (as explained in section 4.3.1.1). Therefore the fact that the mesh node transmits a network packet after receiving a *CONF* packet guarantees that each transmission is independent of each other. The packets were received by the sink node and the transmission power level used by the mesh node was visualized directly in the laptop, through the serial communication interface.

Results Similar to the previous experiment test, the physical communication distance between the sink and the mesh node was varied across time and so assumed 50 centimetres as well as 1, 2, 4, 8 and 16 meters. It was considered the first 10 network packets sent by the mesh node, being the transmission power level used by the mesh node registered in the sink. Based on the information stated in table A.1 it is possible to infer the transmission output power (in dBm) used. In figure 5.3 is presented a graphic that illustrates the output power used by the mesh node in each packet transmission and for a

specific communication distance between the sink and the mesh node. Table 5.2 reports the mean and the standard deviation values for the information presented in figure 5.3.

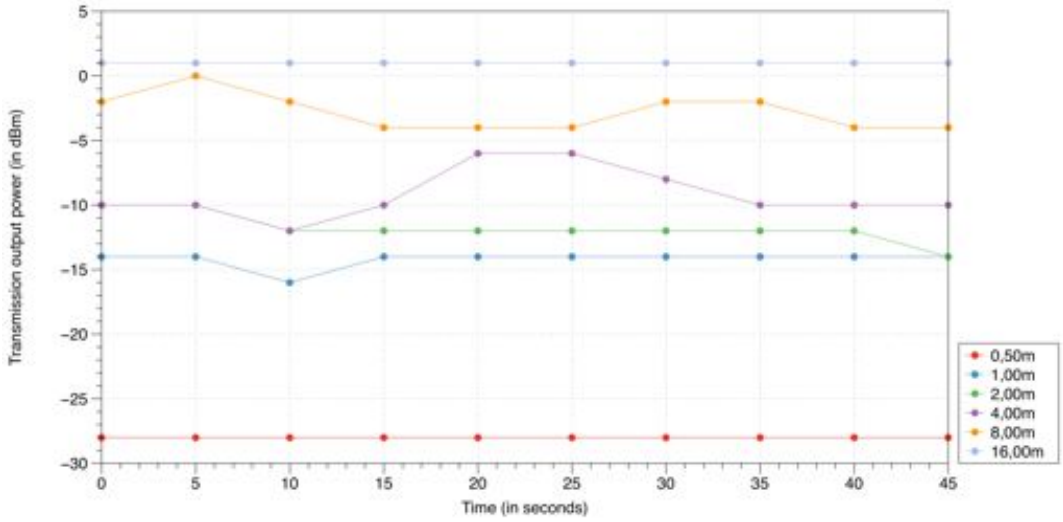


Figure 5.3: Transmission output power adaptation according to the communication distance

Distance (in meters)	0,50	1,00	2,00	4,00	8,00	16,00
Mean Value (in dBm)	-28,00	-14,20	-11,80	-9,20	-2,80	+1,00
Standard Deviation (in dBm)	0	0,63	1,14	1,93	1,40	0

Table 5.2: Mean and standard deviation of the data represented in figure 5.3

Observing figure 5.3 and the transmission output power mean parameter in table 5.2 it is possible to conclude as expected the need to use higher transmission power levels as the distance between the two nodes increases. For 50 centimetres it was observed that it was always used -28,00dBm as output power. In the other extreme for 16 meters (i.e., approaching the maximum communication range for these transceivers) it was necessary to use as expected the maximum transmission power level (i.e., corresponding to +1,00dBm). These facts are represented in table 5.2 whereas the standard deviation parameter assumes 0 in the 0,50 and 16,00 meters. In all the transmissions made, the network packets were transmitted successfully in the first attempt, proving this way that the adopted model for adjusting the transmission power (presented in section 4.3.1.1) gives good guarantees to the system.

Besides decreasing the network interference, transmitting using less power allows to save energy. In figure 5.4 it is possible to compare the current consumption for each tested distance, both using the signal strength adaptation feature, and without it. The current consumption values in the chart were extracted from reference table A.1. For the adaptation on experiment, it was used the current consumption value associated with the necessary transmission output power mean for each tested distance, stated in table 5.2. For the adaptation off experiment it was always used the current consumption associated with the maximum transmission output power level (i.e., corresponding to +1,00dBm). It is possible to infer that the signal strength adaptation feature allows to optimize the packet transmission inside the WSN, saving energy. As expected this energy consumption reduction has more impact as the physical distance between

the two transceivers involved in the communication decreases.

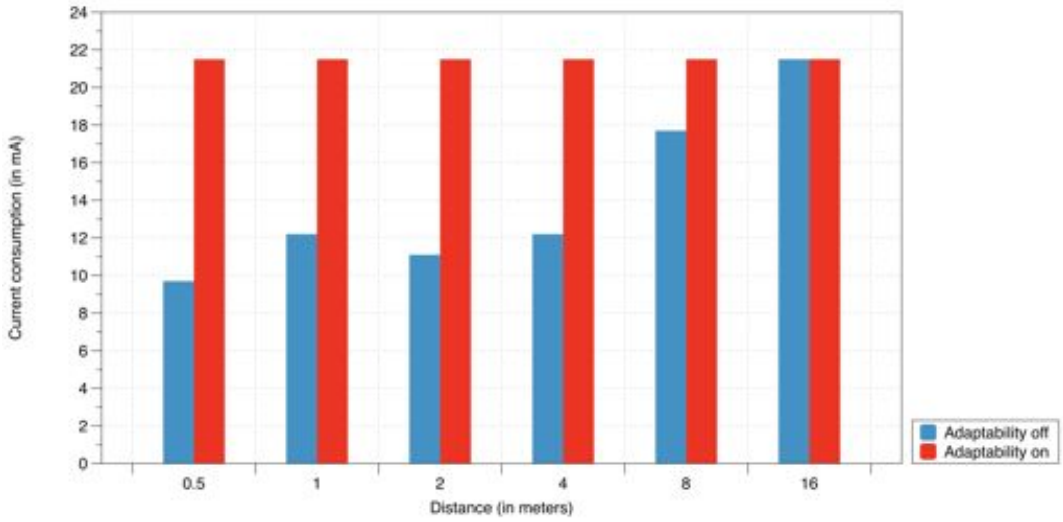


Figure 5.4: Signal strength adaptability impact on the current consumption

5.1.2 Cloud Computing Platform

The *Machinates*® back-end platform was submitted to some load tests in order to test its current operation.

Goals The main goal of this experiment was to observe the behaviour of the Cloud Computing platform with the increase of the number of messages sent to it. It was specifically desired to observe the variation of the processing latency times (i.e., the time interval since a message is delivered to the *Machinates*® until the time it is processed by the platform).

Setup It was used the *SoapUI*[1] software in order to send multiple messages to the *Machinates*®. It was tested five distinct load tests, with 10, 100, 1000, 10000 as well as 100000 messages sent to the *Machinates*®. Each test case was repeated five times, being calculated for each one of them the mean and standard deviation of the processing times of all the messages sent.

Results Figure 5.5 presents a graphical representation for the mean and standard deviation values of the processing times measured, corresponding these values to the mean of the five experiments made for each case. These obtained values are also reported in table 5.3.

Number of uploaded messages	10	100	1000	10000	100000
Mean Value (in seconds)	4,55	14,55	32,87	54,02	161,58
Standard Deviation (in seconds)	1,16	7,34	23,17	47,68	156,30

Table 5.3: Mean and standard deviation of the data represented in figure 5.14

[1] `http://www.soapui.org` (accessed last time on 19th December 2014)

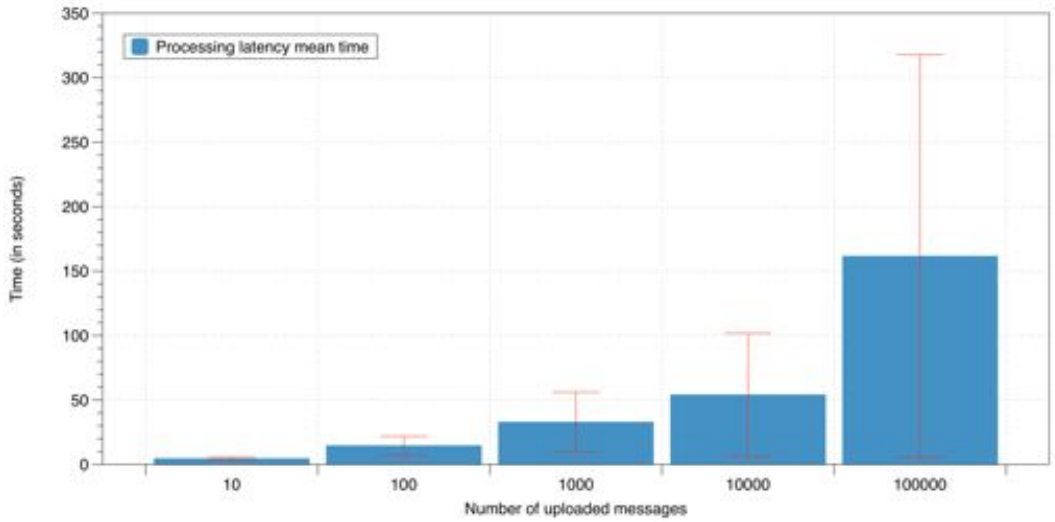


Figure 5.5: Variation of the processing latency mean times according with the number of messages

As expected, the mean processing time is higher as the number of messages sent to the *Machinates*® increases. The variation of the standard deviation values assumes the same behaviour. This behaviour is explained by the fact that as the number of sent messages increases, the number of messages in *Machinates*® processing queue also increases, being the processing time difference between the first and last processed messages very high, thereby drastically increasing the standard deviation values. It is important to clarify that the mean and standard deviation values obtained are related to the processing times of each individual message, which may be a little misleading. Currently the *Machinates*® platform is processing in parallel a block of messages, being the block size dynamic according to the number of messages on the queue. Therefore, during the processing time of a single message, other messages are simultaneously being processed. Notice also that the *Machinates*® platform is still in development, being running in a small AWS instance without auto scaling.

5.2 System-wide Field Testing

The implemented system was tested in real application scenarios. These field evaluation experiments allowed to validate the developed system in a real scenario context as well as to collect real environmental data. It also allowed to collect enriching opinions and other feedback given by the professionals working on this specific branch that could be very useful in future system's improvements.

The collected information during these fields experiments as well as its analysis are presented in the following sections. Whenever appropriate the collected data are presented in the form of a chart or map in order to make its reading clearer to the reader.

5.2.1 *Queijaria Ribeira de Alpreade*'s Experiments

The first system deployment inserted in a concrete field testing case was made with the collaboration of the company *Queijaria Ribeira de Alpreade*, under the scope of the Fundão's *Fab Lab Aldeias do Xisto*.

This company's business is based on the production of regional typical cheese. The innovation in the monitorization of their Livestock assets is seen as a challenging and important aspect.

Setup As mentioned in section 4.1, the system prototype was applied for the monitorization of dairy sheep. A reduced number of nodes was available for these first experimental tests on the field (4 nodes), therefore it was decided to validate this prototype in a small flock of sheep. As there was already a flock of sheep in the farm separated from the main flock corresponding to 8 sheep that were freely grazing in a well defined area, this flock was used as the target for the field tests. In an ideal case all the sheep should have equipped with the system prototype. However, since there was not available enough equipment to equip all the flock, only a subgroup of size four of these sheep were equipped with these devices. Figure 5.6 shows a map corresponding to the area whereas the sheep could be located. Typically, between the sunset and the sunrise the flock of sheep are resting in the open barn signalled on the map, being freely grazing in the field in the remaining of the daytime inside the delimited grazing area. Usually the sheep are grouped in the barn at the beginning and end of the day in order to be milked (i.e., before being set free and after being put back in the barn, respectively). The open barn allowed GPS coverage to the monitorized nodes inside the barn.

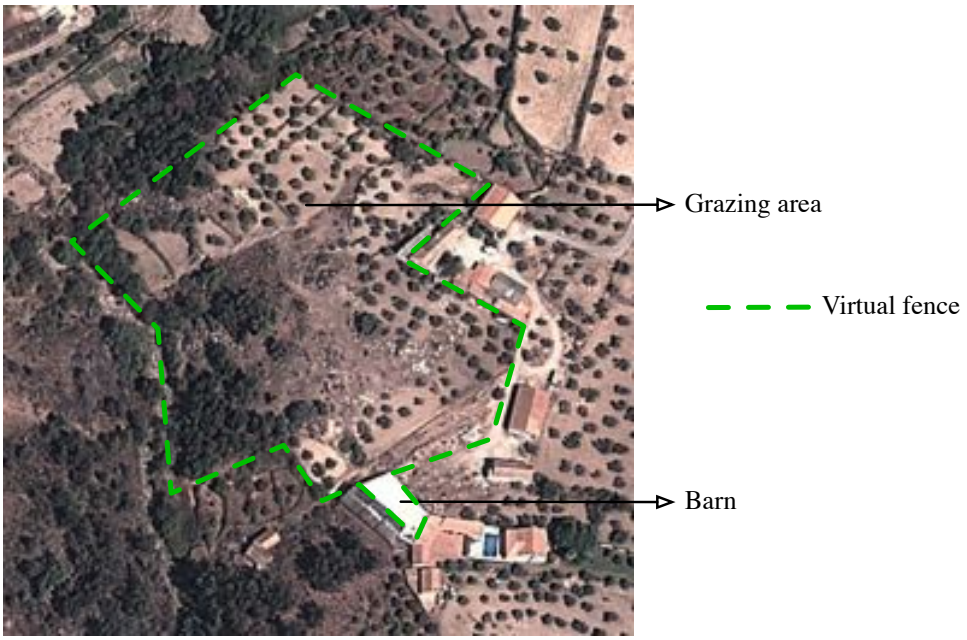


Figure 5.6: Monitorized area in the Portuguese village Zebras

The monitorized sheep were equipped with the ultimate node solution illustrated in figure 4.14. The collected information corresponds to the geographical location of the nodes given by the GPS module as well as the actual state of the their batteries. Each node was programmed to collect its GPS information each 15 minutes (T_{GPS}) and to collect its battery information hourly ($T_{battery}$). The equipment was installed in the sheep's collars as illustrated in figures 5.7 and 5.8.

In these field tests, two of the four nodes corresponded to sink nodes that had capabilities to forward the collected network information to the *Machinates*® using for that the GPRS technology. Two sink

Figure 5.7: Devices in different regional sheep collars

Figure 5.8: Sheep wearing a monitorization collar

nodes were used to add some system redundancy. The remaining two nodes corresponded to mesh nodes that in case of network disconnection had to buffer temporarily the collected information until the connectivity was resumed. The experiments lasted approximately 4 days (i.e., 96 hours), being the devices removed sometimes from the sheep in order to charge the batteries or to be reprogrammed with the purpose of collecting other kind of information.

5.2.1.1 Livestock Mapping

As mentioned before, one of the device's primordial functionalities is to collect its current geographic position and report it to the final data back-end infrastructure. Figure 5.9 shows a screenshot of the *Machinates*® GUI illustrating the overall system's look and feel, as well as the location of each animal on the grazing field.

In figure 5.10 it is represented a heat map (built using the *Google Maps* API) whereas it is marked each geographic position reported by a given sheep during a 24 hours period. Therefore it is possible to clearly identify the areas where the animal remained more active as well as to correlate them with the period of the day in which the information was collected (i.e., if it was collected during the daytime

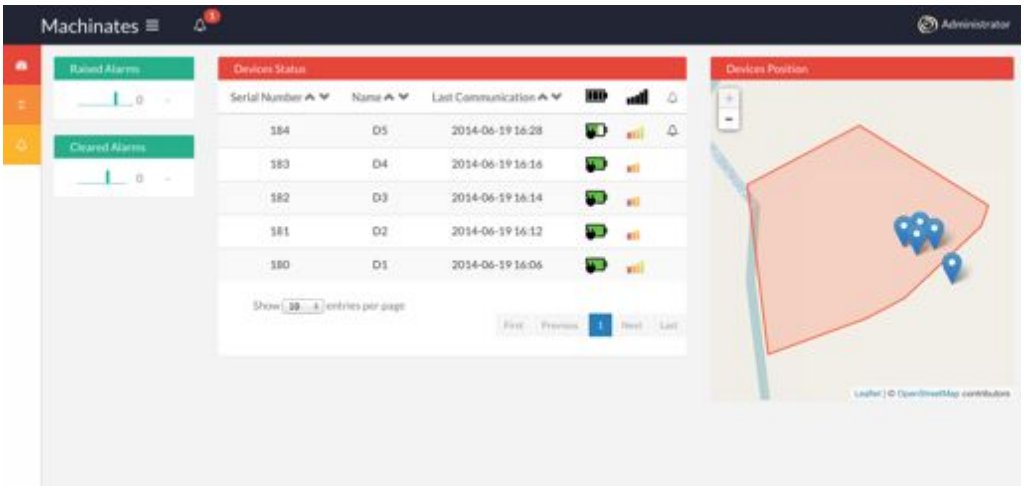


Figure 5.9: Overall vision of the system illustrated on the *Machinates*® GUI

period or the nighttime period).

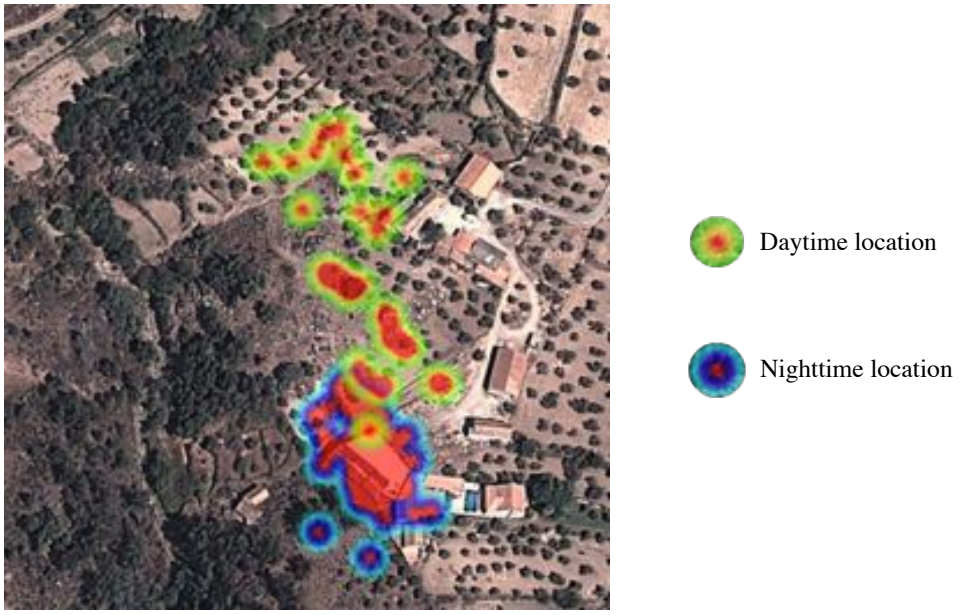


Figure 5.10: Animal's location heat map during 24 hours

It is also possible to observe in figure 5.10 that, as expected, during the daytime period (i.e., the geographical positions collected between 10AM and 20PM) the animal's locations are quite spread along the grazing field. During the nighttime period (i.e., the geographical positions collected between 20PM and 10AM), the geographical area of activity is condensed essentially over the open barn, as it was expected for this specific period of the day when the sheep are resting in the barn. It is also visible in the map some position estimation errors given by the GPS receiver in the order of some meters, that are specially significant in the nighttime positions since during this period there is the barn as reference.

5.2.1.2 Geo-fencing

The geo-fencing concept represents a valuable application that is running upon the developed system. Combining it with the component responsible to the alarms management, it is possible to be notified when an animal crosses a specific virtual boundary and act immediately according to the situation.

In order to verify the correct operation of this feature, it was established a small virtual fence inside the monitorized area, which forced the enabling/disabling of alarms related with the geo-fencing feature. It was verified that every time a sheep position was received with the information that the animal surpassed the virtual boundary, an alarm was triggered. In the opposite situation, every time an animal came back to the allowed grazing area, the triggered alarm was cleared. In figure 5.11 it is possible to view a screenshot of the *Machinates*® GUI illustrating the triggering of an alarm originated by a sheep that crossed the virtual fence, exiting this way the allowed grazing perimeter.

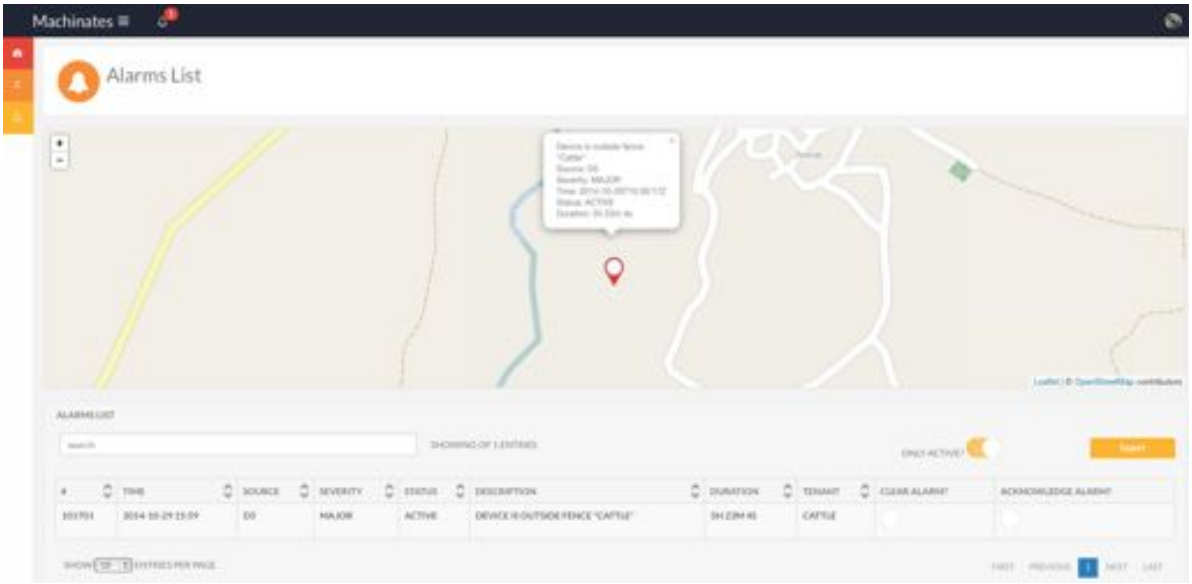


Figure 5.11: Geo-fencing alarm illustrated on the *Machinates*® GUI

5.2.1.3 WSN Infrastructure

As the WSN infrastructure represents a vital part of the system, it also was the target of experimental evaluation in order to extract information related to its behaviour in a real application scenario of animals monitorization. It was tested and verified the connectivity between the WSN nodes along the time, the data latency as well as the battery consumption of the nodes. These topics will be introduced separately hereafter.

Connectivity As previously mentioned in this document, the flocks of sheep form cohesive groups and stay typically together when grazing. In this specific scenario during the nighttime period they are resting at an open barn, confined this way to a reduced area and being the sheep very close to each other (about 1 meter on average). Based on these facts and also considering the conclusions presented in section 5.1.1.2 relatively to the *CC2500*'s communication range, it is expected that the mesh nodes, through multi-hop

communications, could maintain in most of the time connectivity with the sink nodes of the network and through them, with the *Machinates*® infrastructure.

In order to verify this fact, the mesh nodes of the network were programmed to report the amount of time (in seconds) that they had network connectivity (through single-hop or multi-hop) with at least one sink node during 30 minutes. This information was periodically collected during the daytime and the nighttime periods, being represented in figure 5.12. The presented information is related to 5 hours of experiment during daytime and other 5 hours during nighttime, being all data collected by a single mesh node. The network connectivity with the sink nodes was inferred consulting the neighbours table managed by the network protocol presented in section 4.3.

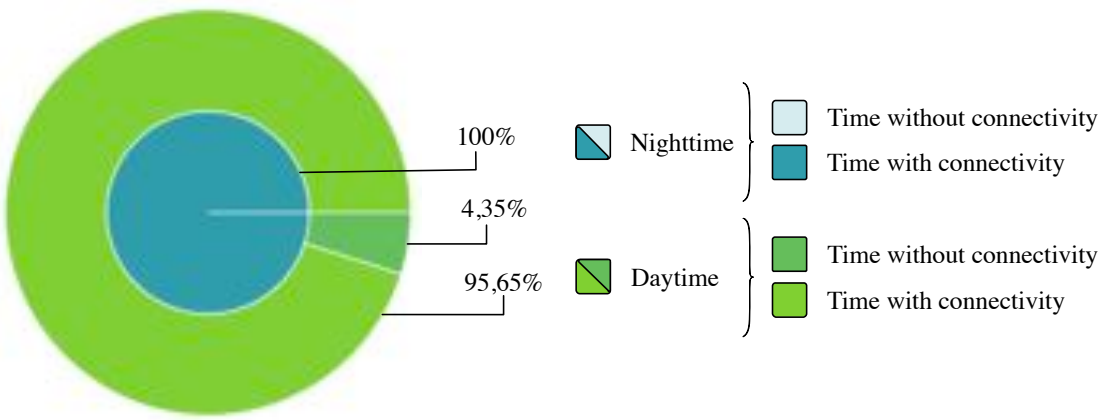


Figure 5.12: Mesh node network connectivity during the daytime and nighttime period

It is possible to notice as expected that during the nighttime period the sheep stay sufficiently together when resting to maintain network connectivity with at least one sink node during the whole time (100%). Looking now for the daytime period, it is possible to observe that the mesh nodes have maintained connectivity with the sink nodes during about 287 minutes out of the 300 minutes of the testing scenario, thus in most of the time (95,65%). As concluded in section 5.1.1.2, the maximum communication range of the *CC2500* transceiver tends to 16 meters. This fact combined with the short distances that sheep usually maintain among themselves during the day and night periods, justify the connectivity time values obtained from this experience. As the sheep are freely grazing/resting in an open field during the daytime (in opposite with the nighttime), it is possible to observe also differences between the network connectivity times during daytime and nighttime, being this value lower as expected during the daytime. The information extracted from this experiment gives a good indicator for the system as the data collected from each network node can flow promptly to the sink node and in consequence to the data back-end with small delays, allowing the system to react more quickly in the existence of an anormal situation that may require special care. Of course, the availability of more WSN nodes on the network will potentially reduce failures on network connectivity as more communication paths become available.

Latency Times Considering latency times collected during the field tests, it is possible to distinguish between the latency time associated with the time interval that the information took to travel since the

time it was collected to the instant that it reaches the *Machinates*® back-end, and the one associated with the time interval that the data took to be processed since it enters in the *Machinates*®.

Figures 5.13 and 5.14 report the variation of the latency times associated with the data packets collected by each of the four nodes of the WSN. In each figure it was represented the latency time for each hour, corresponding this value to the average of the latency times of all the data packets for that given hour. In tables B.1 and B.2 it is reported the mean and the standard deviation of the data illustrated in figure 5.13 and 5.14, respectively.

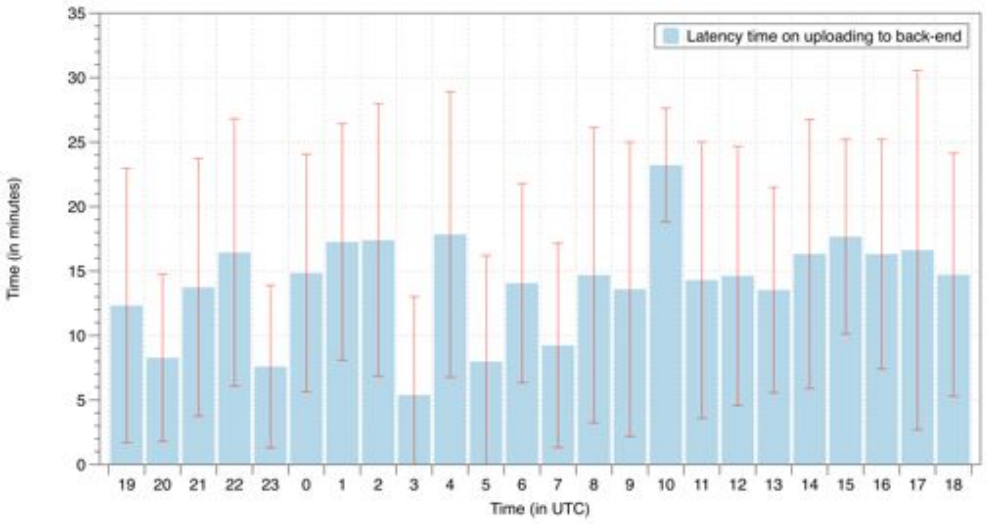


Figure 5.13: Latency times on data uploading to the *Machinates*® back-end

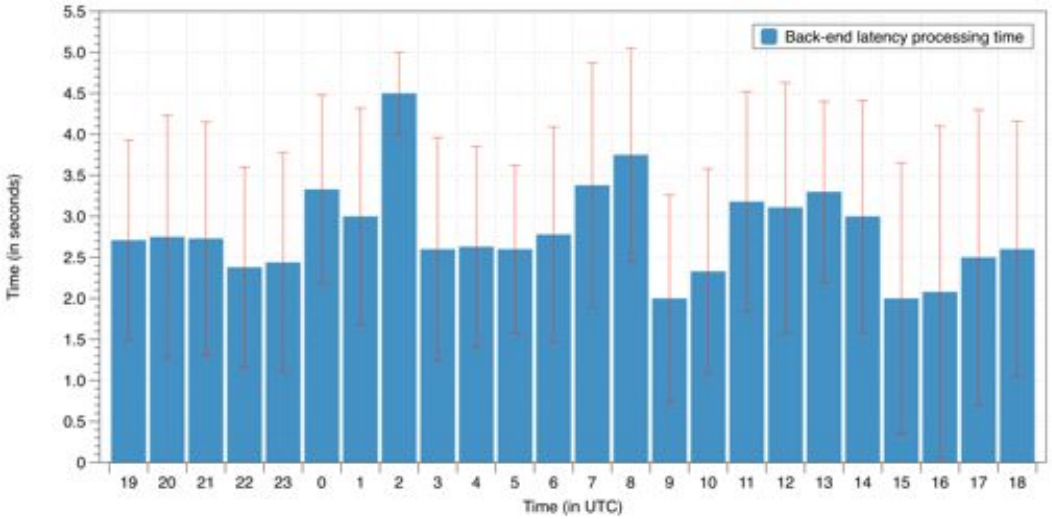


Figure 5.14: *Machinates*® data processing latency times

Before analysing the information illustrated in the figure 5.13, it is important to remind that the sink nodes only upload the information to the *Machinates*® back-end when they have at least 10 data packets (as explained in section 4.2.3), being each WSN node responsible to collect its GPS and battery information each 15 minutes and daily, respectively (see section 4.2.4). Considering these facts, the latency values illustrated in figure 5.13 correspond to the expectations. As the collected data packets had been

built in different timings and buffered in each node, the latency time values to reach the *Machinates*® back-end varies widely from packet to packet, being the standard deviation values stated in table B.1 very high as expected. The maximum latency upload value obtained was 23,22 minutes at 10AM and the minimum corresponded to 5,40 minutes at 3AM. The average for the entire day is 14,09 minutes, which corresponds to a value that it is reasonable and tolerated by this system. If the application requires it, the latency times associated with the data uploading to the *Machinates*® back-end can be easily optimized in the future by decreasing the number of buffered data packets on the sink nodes.

In the figure 5.14 it is possible to observe that the latency processing times of the *Machinates*® back-end never exceeded the 5 seconds, being the average for the entire day 2,82 seconds, which gives good evidence for the system reliable operation.

Power Consumption Although the power consumption was not considered as the most important aspect when the system was designed, it is important to know the actual energy consumption of the WSN nodes.

In figure 5.15 it is illustrated the energy consumption of the monitorized nodes during a period of 24 hours. In the given figure, the monitorized nodes were distinguished between sink and mesh nodes.

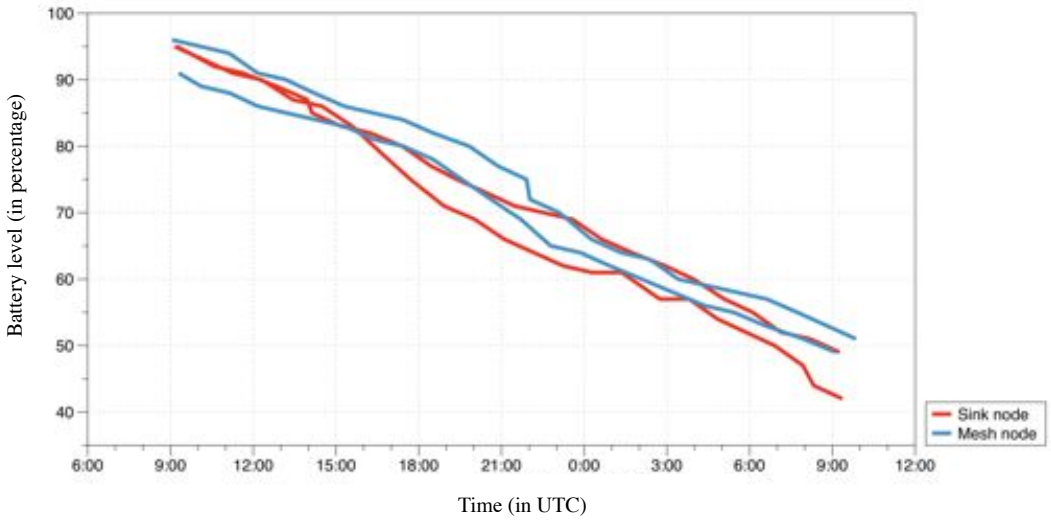


Figure 5.15: Battery consumption over time

It was expected that the energy consumption observed in the sink nodes was higher in relation with the observed in the mesh nodes, as the sink nodes beyond performing the functionalities that the mesh nodes perform (e.g., collecting GPS and battery information, performing tasks related with the network protocol presented in section 4.3, etc.), also have to transmit the data collected by the WSN to the *Machinates*® back-end through GPRS. The absence of a visible energy consumption disparity between these two types of nodes can be justified by the fact that the sink nodes spend only a few seconds transmitting the WSN information to the Cloud platform, thus although this procedure consumes a considerable amount of electrical current (from 80 mA up to 435 mA), it only lasts a few seconds, being the energy consumption of the GPRS module in idle mode (21 mA) considerable lower compared to the

transmitting mode. Furthermore the GPS receiver corresponds to the higher energy consumption module, since the acquisition mode consumes a significant amount of electrical current (77 mA), being the GPS module in this state for a long period of time until it gets a valid GPS fix. The current consumptions of the *SIM908* module can be consulted in the litherature [85].

In the experimental conditions that the WSN nodes were tested, and given its configurations, it was verified that the WSN node's energy autonomy tends to approximately 48 hours. Notice that the evaluated nodes correspond to the first prototype and, as explained in section 4.2.5, are not optimized in relation with the hardware components used (for instance, this solution has two microcontrollers). In the future, this hardware optimization can have impact on the device's energy consumption, reducing it. The WSN node's energy autonomy can be also optimized in the future by simply decreasing data collection frequency, or by increasing the number of batteries in the nodes. Energy harvesting can give here a vital contribution. A very interesting approach could be to equip each node with a small solar panel in order to recharge the batteries. This approach can be specially interesting in sunny environments such as the environment where the system was tested.

5.2.2 Cascais Municipality's Experiments

Some complementar tests were made in order to collect more real information, in an attempt to further improve system evaluation with more subjects (animals). In this context, the Cascais Municipality provided further support, making available their sheep for carrying out further experiments.

Setup Since the primary goal of this project was to collect the maximum possible information about the location of the monitorized animals in order to characterize their behavior, it was decided to collect GPS samples each 3 minutes and maintain the GPS module of the *SIM908* unit always on. The sheep are herding in the natural park of Quinta do Pisão, being the herd composed by 68 sheep. It were only monitorized two sheep. The sheep are set free daily in the morning and are routed to the grazing field located approximately 2km from the barn, being collected again to the barn at the end of the day. The grazing field corresponds to a large dimension space. In figure 5.16 is illustrated the monitorized area.

Results The experiments were done during two days. The devices were working until the batteries were been exhausted. Although different sheep were tracked during different time periods of the day, it were only considered identical monitoring time intervals. For each sheep it was also determined the total travelled distance, applying the formula 4.5 for the collected GPS locations.

In figure 5.17 it is illustrated the variation of the total travelled distance by each monitorized sheep during the experiments done in two days. It is possible to observe that for each hour in the two days the two monitorized sheep have similar travelled distances, indicating they had travelled approximately the same. This fact is typically present for animal groups that maintain cohesive interactions between them, typically staying together. The collected information matches this behaviour as the sheep generally stay together, resulting this way in very similar total travelled distances between each individual sheep. The registered peaks at the 10AM correspond to the period that the sheep were been released to the grazing

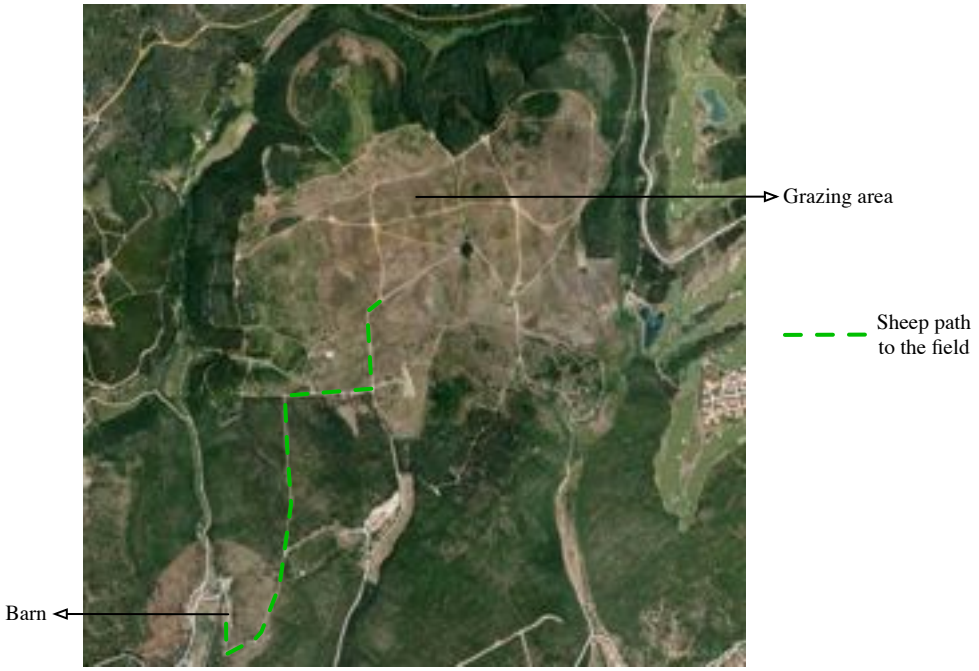


Figure 5.16: Monitorized area in the natural park of Quinta do Pisão

field and travelled approximately 2km to get there. This daily path to the grazing area is illustrated in figure 5.16. The sheep travelled different grazing paths during distinct time periods on each day resulting in large differences in the total travelled distance between the two days.

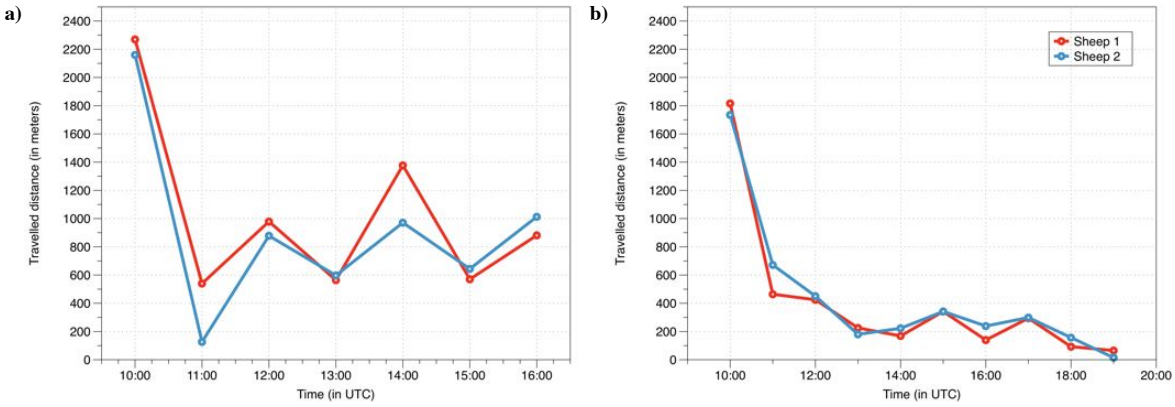


Figure 5.17: Total travelled distance by hour; a) experimental day 1; b) experimental day 2

In figure 5.18 it is illustrated the location variation for the two monitorized sheep during the first experimental day. From the observation of this figure in a) and b) it is possible to observe that the two sheep present similar variations of their latitudes and longitudes along time. It is also possible to observe in c) that these two sheep travelled similar paths along the day.

From the collected information it was intended to distinguish between healthy sheep and thighs sheep or sheep evidencing another type of impediment that makes it impossible to walk normally, showing a pattern outside the usual. However as the thigh sheep that would be the test target was almost recovered, it became impossible to reach any conclusions as this sheep could walk almost normally at the time that

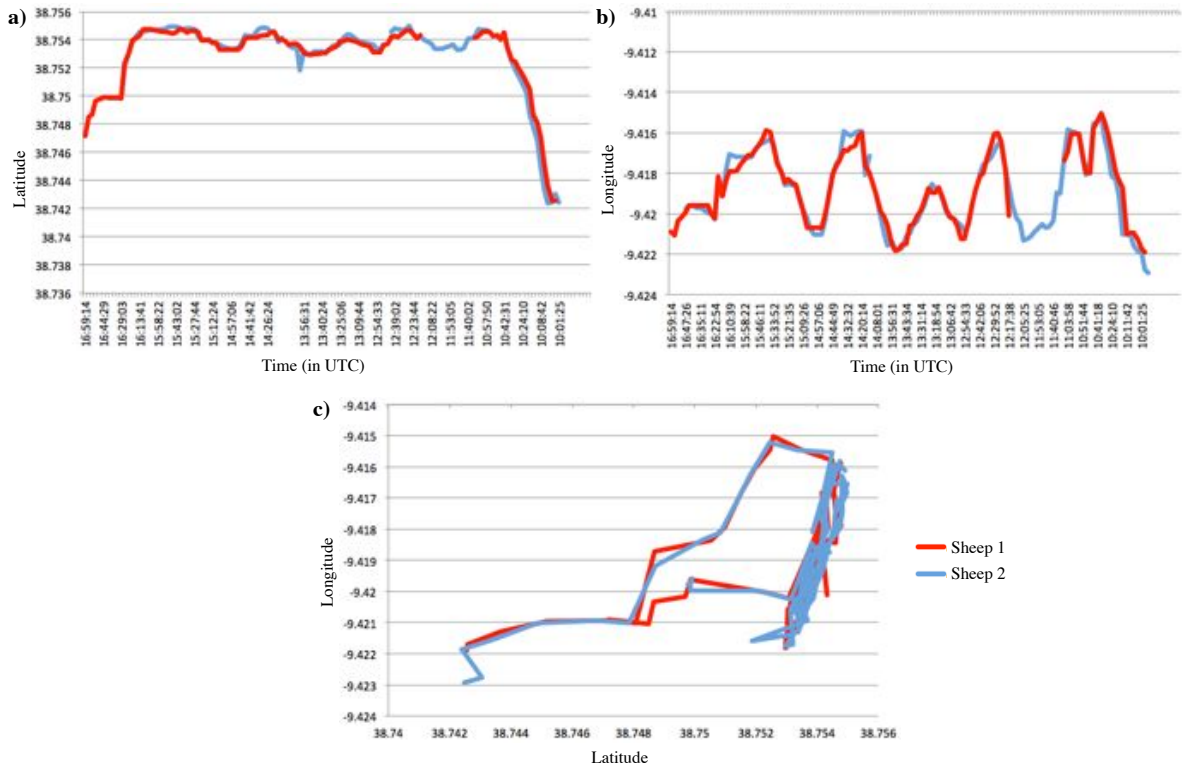


Figure 5.18: Location variation; a) Latitude variation along time; b) Longitude variation along time; c) Latitude and longitude variation

the experiments were performed.

5.2.3 Herdade de Gâmbia's Experiments

This project was also evaluated in the company Herdade de Gâmbia, near Setúbal. The main purpose of these experiments was to infer the capability of the proposed system in detecting and distinguishing between thighs and healthy sheep.

Setup It was decided to collect GPS samples each 3 minutes and maintain the GPS module of the *SIM908* unit always on. The sheep are herding in the Herdade de Gâmbia's pasture area, being the herd composed by 300 sheep. It were only monitorized two sheep, corresponding to a thigh and healthy ones.

Results The experiments were done during two days. The devices were working until the batteries were been exhausted. Although different sheep were tracked during different time periods of the day, it were only considered identical monitoring time intervals.

In each day and for each sheep, it was determined the velocity between two consecutive measured points, through the formula 5.1. In this equation the lat_{i-1} and lon_{i-1} corresponds to the latitude and longitude of the first point, the lat_i and lon_i to the latitude and longitude of the second point, the $timestamp_{i-1}$ the time when the first geo-location point was obtained and the $timestamp_i$ the time when the second geo-location point was obtained. Thereafter, several features were extracted, namely average and standard deviation values for the two experimental days, as well as an entropy measure

(corresponding to the entropy value determined for the maximum scale according to Costa et al. [95]). The results extracted from the two experimental days are presented in table 5.4 and illustrated in figure 5.19.

$$v = \frac{\sqrt{(lat_i - lat_{i-1})^2 + (lon_i - lon_{i-1})^2}}{(timestamp_i - timestamp_{i-1})} \tag{5.1}$$

Day 1	**Average**	**Standard Deviation**	**Entropy**
Normal Sheep	0,21	0,15	1,56
Thigh Sheep	0,18	0,17	1,42
Day 2	**Average**	**Standard Deviation**	**Entropy**
Normal Sheep	0,22	0,16	2,44
Thigh Sheep	0,21	0,24	1,80

Table 5.4: Statistical information collected from the two experimental days

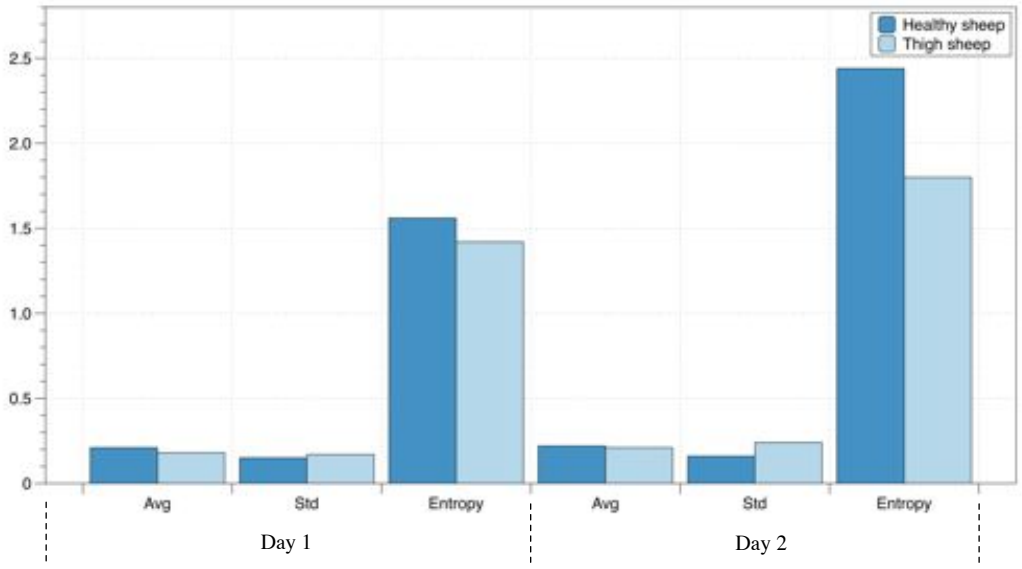


Figure 5.19: Illustration of the statistical information collected from the two experimental days

It can be noticed that the value of entropy is smaller for the thigh sheep during the two days of experiments. This may give some initial hint for a mechanism to identify healthy from non-healthy sheep. Our hypothesis concerning learning animal's health remains to be tested upon the future availability of more training data (just two collars were available for this experimental evaluation). It concerns the variation of the entropy values over several days, namely if for non-healthy individuals the entropy value over several days gets consistently lower than for healthy individuals (as shown in figure 5.19 just for two experimental days). On a neural network, the inputs could be the entropy values of an animal over several days, and the output a value giving the probability of an animal being healthy, or not.

However, given the extremely small dataset available, no meaningful conclusions can be extracted concerning:

- The best features to extract from the data in order to feed a learning mechanism;
- The capability, or not, of identifying healthy animals from non-healthy ones using their patterns of motion.

Concerning the last point, Costa et al. [95] have shown that the fractal property of organisms, such as given by multi-scale entropy measure, gives a measure of an organism healthy, and have shown this strategy applied to Electrocardiography (ECG) signals, as well as motion data. However, on their experiments the sampling frequency is significantly higher than 3 minutes, and windows of analysis contain thousands of points (compared to nearly 100 to 200 data points for 6 to 8 hours collection of sheep location data), and hence is arguable if such approach could bring in the future some benefits, or not, for the problem at hand.

5.3 Summary and Conclusions

The developed WSN system was evaluated through several experiments, in order to verify its actual behaviour and effectiveness whenever employed in real application scenarios. Furthermore this presented evaluation methodology allowed also to identify future system improvements.

In general, the results and observations extracted from the evaluation methodology confirmed that this kind of technological solutions, and specifically this system prototype, has potencial for real-world applicability. The implemented system matched the initial expectations and worked properly in real application scenarios, collecting periodically the information from the monitorized nodes and delivering it to a sink node through the implemented network protocol. Besides the functionality inside the WSN, the collected data were transmitted successfully to the *Machinates*®, where it was finally processed and presented to the end-user.

During the course of the reported experiments it was found that the sheep remain a long time in the same activity zone, evidencing very smooth movements over time. Particularly during the *Queijaria Ribeira de Alpreade*'s experiments, it was found that the area where the system was tested represents an area of poor cellular coverage, thus it was verified that sometimes the equipment was unable to register it on the cellular network and even became disconnected during the data transmission process to the back-end platform. The equipment seemed invisible to the sheep and remained always attached to their collars. The protector equipment case proved its robustness in relation to the application's requirements, protecting it adequately against the shocks and humidity. Also related to this topic and given the experimental conditions, it was found that the equipment tend to slide along the collar of the sheep to its throat. Given this fact, the equipment antennas were pointing to the ground, making difficult to receive a valid GPS fix. These scenario worsened further when sheep lay over the equipment when resting. This problem can be easily resolved in the future, incorporating directly the equipment in the sheep collars.

During the field experiments, some interviews were conducted with some experts on this area, such as farmers, Livestock producers as well as farm managers. From these interviews they were extracted some interesting practical views as well as some facts taken out from the field experience of these professionals:

- The installation of the electronic device inside the animal's rumen through a bolus can be (depending on the animal that is being applied) extremely painful to the animals. For instance, this approach is not advisable to sheep. In this case the installation of the device in the sheep collars is viewed as a good alternative, specially if it is integrated in the collars. In the case of small dimension electronic devices, another good alternative could be the subcutaneous affixation of the device;

- These electronic devices could replace in the future the currently used ear tags (visible in the figure 5.8), as these conventional identifiers get very often detached accidentally from the animal's ear, getting lost in the pasture areas. This fact implies that specialized staff has to be called to the field in order to make a new identification of the animal, thus the proposed system represents a possible solution for this problem;

- The interviewed *Queijaria Ribeira de Alpreade*'s farm managers pointed as a long term goal to extend the monitorization to the entire process of cheese production. Therefore it will allow in the future to correlate information from different specific business stages and infer some important conclusions with the final goal of optimizing all the system and its production. A practical example of this scenario would be to equip the currently used automated milking systems of the farm with sensors and this way correlate the production and quality of the milk produced by a flock of sheep (or individual animals) with their habits and used grazing locations. Comparing all the collected information will allow to readjust certain aspects in order to increase the productivity of the farm. This vision of global monitorization throughout all the production line can be quite interesting in the Livestock area as a way of optimizing the business as well as better understanding the animals and their behaviours.

Chapter 6

Conclusions

The continuous technological evolution and its miniaturization have allowed novel application scenarios and its proliferation to many sectors of our society. Particularly, this crescent technology incorporation on everyday business tasks has the potential to improve the way monitorization is done and at the same time reduce the costs associated to it. This is particularly evident in the Livestock area, where the monitorization of the animals is still often done using human observation.

In this context, the installation of small sensing devices given by the WSNs, combined with high-performance Cloud Computing platforms used to process the collected environmental information could play a vital role in the monitorization of the assets. However the application of these concepts in demanding scenarios may have associated to it significant challenges, such as the lack of standards, the high dynamism of these kind of networks, the very limited resources of the devices as well as its limited energy autonomy.

This dissertation proposes a WSN system based on a Cloud Computing platform specifically designed for the Livestock monitorization area. This monitorization was centered on the periodical reporting of the animals' geographical location, and correspondent extraction of useful information to the farmer, such as variations in the normal bustle of each individual animal, propagation of diseases, thefts, etc. After carrying out interviews with specialized staff, it was possible to make important design decisions.

The WSN is formed by the group of animals to be monitorized, being each device installed in each individual animal. The animal's location is collected using a GPS receiver. The information collected by each animal is propagated in the mobile ad hoc network following a gradient routing protocol strategy. The network nodes have the capability of proactively adapting themselves between mesh or sink nodes, according to the current network topology. Besides collecting information, the sink nodes have the important role of transmitting the WSN information to the Cloud Computing platform through GPRS. The Cloud Computing platform gives support to the WSN by storing and processing its data in order to extract as much information as possible of the current state of each animal as well as of the business in general. In this context the cooperation of the Sensefinity® enterprise was crucial as they have provided the necessary hardware as well as the access to its Cloud Computing resources.

Finally, it was produced the first system functional prototype that was submitted to a rigorous evalua-

tion process. This first prototype was tested in real scenario applications, in particular for the monitorization of free-ranging dairy sheep. The results obtained from these experimental tests were quite positive, giving good indicators of its effectiveness in this kind of monitorization solutions. The implemented system endured the demanding and unpredictable conditions that characterize these environmental scenarios. During these experiments, it was possible to extract very interesting real information as well as enriching feedback from specialized staff, which will certainly be used for future system improvements.

This first system prototype that was used as a proof of concept has demonstrated that it can be effectively employed in this kind of monitorization scenarios, giving automatically valuable information to the farmer that would otherwise be very difficult to extract using traditional monitorization mechanisms.

6.1 Future Work

As future work and in order to improve the implemented system, there are some topics that may be improved or completed, namely:

- Optimizing the hardware solution (e.g. incorporate the low-range radio built-in the *Butterfinger*® board, making it possible to use only one MCU, etc.);
- Explore the Wake-on-Radio (WOR) feature of the *CC2500* in order to save energy, by putting periodically the radio in a sleep mode;
- Adopt an energy harvesting solution in order to prevent to regularly change the batteries. One possible solution is to equip the sheep's collars with a small solar panel;
- Replace the current GPS antenna with an active antenna, in order to improve the reception of the GPS signal;
- Optimize the *Columbus*® messages in order to reduce the size of the messages, therefore optimizing the data transmission to the *Machinates*® platform;
- Allow to change system parameters through the *Machinates*® GUI;
- Implement features that were not completed in this first prototype (e.g., detect and report cuts in the sheep's collars and allow to identify the direct and indirect interactions between the animals);
- Particularly in the *Queijaria Ribeira de Alpreade*'s use-case, extend in the future the monitorization to other components of the business (e.g., milking system) in order to generate more information and combine it, extracting relevant information to the farmer about the current state of the business;
- Continue the experiments done in the company Herdade de Gâmbia with a higher number of monitorized animals in different health conditions for a longer period of time in order to test the solution's effectiveness in detecting and distinguishing between healthy and non-healthy sheep;
- Apply machine learning strategies to collected data in order to detect certain abnormal behavioral patterns that may suggest possible animal diseases.

Appendix A

CC2500 Radio's *PATABLE* Settings

Output Level	**Output Power (dBm)**	***PATABLE* Value**	**Current Consumption (mA)**
1	-30	0x50	9,9
2	-28	0x44	9,7
3	-26	0xC0	10,2
4	-24	0x84	10,1
5	-22	0x81	10,0
6	-20	0x46	10,1
7	-18	0x93	11,7
8	-16	0x55	10,8
9	-14	0x8D	12,2
10	-12	0xC6	11,1
11	-10	0x97	12,2
12	-8	0x6E	14,1
13	-6	0x7F	15,0
14	-4	0xA9	16,2
15	-2	0xBB	17,7
16	0	0xFE	21,2
17	+1	0xFF	21,5

Table A.1: Optimum *PATABLE* settings for the possible output power levels

Appendix B

Queijaria Ribeira de Alpreade's Test

B.1 Data Uploading Latency Times

Time (in UTC)	Mean Value (in minutes)	Standard Deviation (in minutes)
19:00	12,33	10,64
20:00	8,28	6,49
21:00	13,74	9,99
22:00	16,45	10,37
23:00	7,60	6,30
00:00	14,85	9,21
01:00	17,26	9,17
02:00	17,40	10,56
03:00	5,40	7,62
04:00	17,84	11,08
05:00	7,98	8,22
06:00	14,07	7,72
07:00	9,24	7,92
08:00	14,69	11,44
09:00	13,59	11,42
10:00	23,22	4,41
11:00	14,31	10,72
12:00	14,63	10,05
13:00	13,54	7,97
14:00	16,33	10,42
15:00	17,67	7,54
16:00	16,32	8,90
17:00	16,63	13,93
18:00	14,72	9,42

Table B.1: Mean and standard deviation of the data represented in figure 5.13

B.2 *Machinates®* Data Processing Latency Times

Time (in UTC)	Mean Value (in seconds)	Standard Deviation (in seconds)
19:00	2,71	1.22
20:00	2,75	1.48
21:00	2,73	1.42
22:00	2,38	1.22
23:00	2,44	1.34
00:00	3,33	1.15
01:00	3,00	1.32
02:00	4,50	0.50
03:00	2,60	1.36
04:00	2,63	1.22
05:00	2,60	1.02
06:00	2,78	1.31
07:00	3,38	1.49
08:00	3,75	1.30
09:00	2,00	1.26
10:00	2,33	1.25
11:00	3,18	1.34
12:00	3,11	1.52
13:00	3,30	1.10
14:00	3,00	1.41
15:00	2,00	1.65
16:00	2,08	2.02
17:00	2,50	1.80
18:00	2,60	1.56

Table B.2: Mean and standard deviation of the data represented in figure 5.14

Bibliography

[1] M. Upton. "The Role of Livestock in Economic Development and Poverty Reduction". In *Pro-poor Livestock Policy Initiative (PPLPI) Working Paper*, volume 10, February 2004.

[2] E. Tullo, I. Fontana, and M. Guarino. "Precision Livestock Farming: An Overview of Image and Sound Labelling". In *Precision Livestock Farming '13*, pages 30–38, September 2013.

[3] D. Berckmans. "Automatic On-line Monitoring of Animals by Precision Livestock Farming". In *International Society for Animal Hygiene*, pages 27–30, 2004.

[4] K. Kwong, T. Wu, H. Goh, K. Sasloglou, B. Stephen, I. Glover, C. Shen, W. Du, C. Michie, and I. Andonovic. "Practical Considerations for Wireless Sensor Networks in Cattle Monitoring Applications". In *Computers and Electronics in Agriculture*, volume 81, pages 33–44, February 2012.

[5] K. Kwong, T. Wu, H. Goh, K. Sasloglou, B. Stephen, I. Glover, C. Shen, W. Du, C. Michie, and I. Andonovic. "Implementation of Herd Management System with Wireless Sensor Networks". In *Wireless Sensor Systems*, volume 1, issue 2, pages 55–65, June 2011.

[6] W. Edwards, A. Chamra, R. Mayer, and T. Olsen. "Estimated Costs for Livestock Fencing". In *Iowa State University Extension and Outreach*, pages 1–4, February 2012.

[7] F. Reclus and K. Drouard. "Geofencing for Fleet & Freight Management". In *9th International Conference on Intelligent Transport Systems Telecommunications (ITST)*, pages 353–356, October 2009.

[8] D. Swain, G. Bishop-Hurley, and J. Griffiths. "Automatic Cattle Control Systems - Grazing Without Boundaries". In *Farming Ahead*, pages 70–72, June 2009.

[9] P. Sikka, P. Corke, P. Valencia, C. Crossman, D. Swain, and G. Bishop-Hurley. "Wireless Ad Hoc Sensor and Actuator Networks on the Farm". In *Information Processing in Sensor Networks (IPSN)*, pages 492–499, April 2006.

[10] D. Evans. "The Internet of Things: How the Next Evolution of the Internet Is Changing Everything". In *Cisco Internet Business Solutions Group (IBSG)*, April 2011.

[11] L. Tan and N. Wang. "Future Internet: The Internet of Things". In *3rd International Conference on Advanced Computer Theory and Engineering (ICACTE)*, volume 5, pages 376–380, August 2010.

[12] A. Iera, C. Floerkemeier, J. Mitsugi, and G. Morabito. "The Internet of Things". In *IEEE Wireless Communications*, volume 17, issue 6, pages 8–9, December 2010.

[13] L. Atzori, A. Iera, and G. Morabito. "The Internet of Things: A Survey". In *Computer Networks*, volume 54, issue 15, pages 2787–2805, October 2010.

[14] Y. Chen. "Challenges and Opportunities of Internet of Things". In *17th Asia and South Pacific Design Automation Conference (ASP-DAC)*, pages 383–388, January-February 2012.

[15] M. Domingo. "An Overview of the Internet of Things for People with Disabilities". In *Journal of Network and Computer Applications*, volume 35, issue 2, pages 584–596, March 2012.

[16] M. Chorost. "The Networked Pill". In *MIT Technology Review*, March 2008.

[17] M. Domingo. "An Overview of the Internet of Underwater Things". In *Journal of Network and Computer Applications*, volume 35, issue 6, pages 1879–1890, November 2012.

[18] M. Castro, A. Jara, and A. Skarmeta. "An Analysis of M2M Platforms: Challenges and Opportunities for the Internet of Things". In *6th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS)*, pages 757–762, July 2012.

[19] D. Crowe. "Machine-to-Machine Communications". In *Wireless Telecom*, volume Issue 3, pages 28–31, 2007.

[20] V. Galetić, I. Bojić, M. Kušek, G. Ježić, S. Dešić, and D. Huljenić. "Basic Principles of Machine-to-machine Communication and Its Impact on Telecommunications Industry". In *34th International Convention MIPRO*, pages 380–385, May 2011.

[21] G. Wu, S. Talwar, K. Johnsson, N. Himayat, and K. Johnson. "M2M: From Mobile to Embedded Internet". In *IEEE Communications Magazine*, volume 49, issue 4, pages 36–43, April 2011.

[22] K. Chang, A. Soong, M. Tseng, and Z. Xiang. "Global Wireless Machine-to-machine Standardization". In *IEEE Communications Magazine*, volume 15, issue 2, pages 64–69, March-April 2011.

[23] I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. "Wireless Sensor Networks: A Survey". In *Computer Networks*, volume 38, issue 4, pages 393–422, March 2012.

[24] D. Puccinelli and M. Haenggi. "Wireless Sensor Networks: Applications and Challenges of Ubiquitous Sensing". In *IEEE Circuits and Systems Magazine*, volume 5, issue 3, pages 19–31, 2005.

[25] P. Corke, T. Wark, R. Jurdak, W. Hu, P. Valencia, and D. Moore. "Environmental Wireless Sensor Networks". In *Proceedings of the IEEE*, volume 98, issue 11, pages 1903–1917, November 2010.

[26] J. Kahn, R. Katz, and K. Pister. "Emerging Challenges: Mobile Networking for "Smart Dust"". In *Journal of Communications and Networks*, volume 2, issue 3, pages 188–196, 2000.

[27] L. Ruiz, T. Braga, F. Silva, H. Assunção, J. Nogueira, and A. Loureiro. "On the Design of a Self-managed Wireless Sensor Network". In *IEEE Communications Magazine*, volume 43, issue 8, pages 95–102, 2005.

[28] J. Ko, C. Lu, M. Srivastava, J. Stankovic, A. Terzis, and M. Welsh. "Wireless Sensor Networks for Healthcare". In *Proceedings of the IEEE*, volume 98, issue 11, pages 1947–1960, November 2010.

[29] M. Mafuta, M. Zennaro, A. Bagula, G. Ault, H. Gombachika, and T. Chadza. "Successful Deployment of a Wireless Sensor Network for Precision Agriculture in Malawi". In *IEEE 3rd International Conference on Networked Embedded Systems for Every Application (NESEA)*, pages 1–7, 2012.

[30] "The Evolution of Wireless Sensor Networks". In *Silicon Laboratories, Inc. White Papers*, pages 1–5, March 2013.

[31] L. Oliveira and J. Rodrigues. "Wireless Sensor Networks: a Survey on Environmental Monitoring". In *Journal of Communications*, volume 6, issue 2, pages 143–151, April 2011.

[32] T. Arampatzis, J. Lygeros, and S. Manesis. "A Survey of Applications of Wireless Sensors and Wireless Sensor Networks". In *Proceedings of the 13th Mediterranean Conference on Control and Automation*, pages 719–724, June 2005.

[33] R. Szewczyk, A. Mainwaring, J. Polastre, J. Anderson, and D. Culler. "An Analysis of a Large Scale Habitat Monitoring Application". In *Proceedings of the 2nd International Conference on Embedded Networked Sensor Systems (SenSys)*, pages 214–226, 2004.

[34] M. Kochláň, J. Miček, and M. Hyben. "Wireless Sensor Network Energy Harvesting". In *1st International Virtual Conference on Intelligent Transportation Systems (ITS)*, pages 93–97, August 2013.

[35] G. Simon, M. Maróti, Á Lédeczi, G. Balogh, B. Kusy, A. Nádas, G. Pap, J. Sallai, and K. Frampton. "Sensor Network-based Countersniper System". In *2nd International Conference on Embedded Networked Sensor Systems*, pages 1–12, November 2004.

[36] G. Zhao. "Wireless Sensor Networks for Industrial Process Monitoring and Control: A Survey". In *Network Protocols and Algorithms*, volume 3, issue 1, pages 46–63, 2011.

[37] E. Blomqvist. "Security in Sensor Networks". In *Helsinki University of Technology, Control Engineering Laboratory, Report 135*, 2003.

[38] L. Krishnamurthy, R. Adler, P. Buonadonna, J. Chhabra, M. Flanigan, N. Kushalnagar, L. Nachman, and M. Yarvis. "Design and Deployment of Industrial Sensor Networks: Experiences from a Semiconductor Plant and the North Sea". In *Proceedings of the 3rd International Conference on Embedded Networked Sensor Systems (SenSys)*, pages 64–75, November 2005.

[39] C. Baker, K. Armijo, S. Belka, M. Benhabib, V. Bhargava, N. Burkhart, A. Der Minassians, G. Dervisoglu, L. Gutnik, M. Haick, C. Ho, M. Koplow, J. Mangold, S. Robinson, M. Rosa, M. Schwartz, C. Sims, H. Stoffregen, A. Waterbury, E. Leland, T. Pering, and P. Wright. "Wireless Sensor Networks for Home Health Care". In *21st International Conference on Advanced Information Networking and Applications Workshops (AINAW)*, volume 2, pages 832–837, 2007.

[40] A. Barbato, L. Borsani, and A. Capone. "A Wireless Sensor Network Based System for Reducing Home Energy Consumption". In *7th Annual IEEE Communications Society Conference on Sensor Mesh and Ad Hoc Communications and Networks (SECON)*, pages 1–3, 2010.

[41] C. Townsend. "Wireless Sensor Networks: Principles and Applications". In *Sensor Technology Handbook*, pages 575–589, 2005.

[42] A. Shrestha and L. Xing. "A Performance Comparison of Different Topologies for Wireless Sensor Networks". In *IEEE Conference on Technologies for Homeland Security*, pages 280–285, May 2007.

[43] J. Al-Karaki and A. Kamal. "Routing Techniques in Wireless Sensor Networks - A Survey". In *IEEE Wireless Communications*, volume 11, issue 6, pages 6–28, December 2004.

[44] K. Sohraby, D. Minoli, and T. Znati. "Wireless Sensor Networks: Technology, Protocols, and Applications". *Wiley*, 2007.

[45] "IEEE Standard for Local and Metropolitan Area Networks - Part 15.1: Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Wireless Personal Area Networks (WPANs)". In *IEEE Computer Society*, August 2005.

[46] "IEEE Standard for Local and Metropolitan Area Networks - Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs)". In *IEEE Computer Society*, September 2011.

[47] "ZigBee Specification". In *ZigBee Standards Organization*, January 2008.

[48] "IEEE Standard for Local and Metropolitan Area Networks - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications". In *IEEE Computer Society*, August 2005.

[49] P. Baronti, P. Pillai, V. Chook, S. Chessa, A. Gotta, and Y. Hu. "Wireless Sensor Networks - A Survey on the State of the Art and the 802.15.4 and ZigBee Standards". In *Computer Communications*, volume 30, issue 7, pages 1655–1695, May 2007.

[50] E. Alotaibi and B. Mukherjee. "A Survey on Routing Algorithms for Wireless Ad Hoc and Mesh Networks". In *Computer Networks*, volume 56, issue 2, pages 940–965, February 2012.

[51] C. Perkins and P. Bhagwat. "Highly Dynamic Destination-sequenced Distance-vector Routing (DSDV) for Mobile Computers". In *Proceedings of the Conference on Communications Architectures, Protocols and Applications (SIGCOMM)*, pages 234–244, 1994.

[52] P. Jacquet, P. Muhlethaler, T. Clausen, A. Laouiti, A. Qayyum, and L. Viennot. "Optimized Link State Routing Protocol for Ad Hoc Networks". In *Proceedings of the IEEE Multi Topic Conference*, pages 62–68, 2001.

[53] A. Boukerche, B. Turgut, N. Aydin, M. Ahmad, L. Bölöni, and D. Turgut. "Routing Protocols in Ad Hoc Networks: A Survey". In *Computer Networks*, volume 55, issue 13, pages 3032–3080, September 2011.

[54] C. Perkins and M. Royer. “Ad Hoc On-Demand Distance Vector Routing”. In *Workshop on Mobile Computing Systems and Applications (WMCSA)*, pages 90–100, February 1999.

[55] D. Johnson and D. Maltz. “Dynamic Source Routing in Ad Hoc Wireless Networks”. In *Mobile Computing*, volume 353, pages 153–181, 1996.

[56] M. Mauve, J. Widmer, and H. Hartenstein. “A Survey on Position-based Routing in Mobile Ad Hoc Networks”. In *IEEE Network*, volume 15, issue 6, pages 30–39, November-December 2001.

[57] J. Li, J. Jannotti, D. De Couto, D. Karger, and R. Morris. “A Scalable Location Service for Geographic Ad Hoc Routing”. In *Mobile Computing and Networking*, pages 120–130, August 2000.

[58] B. Karp and H. Kung. “GPSR - Greedy Perimeter Stateless Routing for Wireless Networks”. In *Mobile Computing and Networking*, pages 243–254, 2000.

[59] R. Pantoni and D. Brandão. “A Gradient Based Routing Scheme for Street Lighting Wireless Sensor Networks”. In *Journal of Network and Computer Applications*, volume 36, issue 1, pages 77–90, January 2013.

[60] K. Kwong, T. Wu, H. Goh, B. Stephen, M. Gilroy, C. Michie, and I. Andonovic. “Wireless Sensor Networks in Agriculture: Cattle Monitoring for Farming Industries”. In *Progress in Electromagnetics Research Symposium Proceedings*, volume 1-2, pages 1719–1723, 2009.

[61] G. Mao, B. Fidan, and B. Anderson. “Wireless Sensor Network Localization Techniques”. In *Computer Networks*, volume 51, issue 10, pages 2529–2553, July 2007.

[62] A. Pal. “Localization Algorithms in Wireless Sensor Networks: Current Approaches and Future Challenges”. In *Network Protocols and Algorithms*, volume 2, issue 1, 2010.

[63] A. Srinivasan and J. Wu. “A Survey on Secure Localization in Wireless Sensor Networks”. In *Encyclopedia of Wireless and Mobile Communications*, 2008.

[64] T. Alhmiedat and S. Yang. “A Survey: Localization and Tracking Mobile Targets through Wireless Sensors Network”. In *8th Annual Network Symposium*, June 2007.

[65] N. Bulusu, J. Heidemann, and D. Estrin. “GPS-less Low-cost Outdoor Localization for Very Small Devices”. In *IEEE Personal Communications*, volume 7, issue 5, pages 28–34, 2000.

[66] T. He, C. Huang, B. Blum, J. Stankovic, and T. Abdelzaher. “Range-free Localization Schemes for Large Scale Sensor Networks”. In *Proceedings of the 9th Annual International Conference on Mobile Computing and Networking*, pages 81–95, September 2003.

[67] D. Niculescu and B. Nath. “DV Based Positioning in Ad Hoc Networks”. In *Telecommunication Systems*, volume 22, issue 1-4, pages 267–280, January 2003.

[68] T. Wark, C. Crossman, P. Valencia, P. Corke, G. Bishop-Hurley, and D. Swain. “Poster Abstract: A Sensor Network for Compression and Streaming of GPS Trajectory Data”. In *6th ACM conference on Embedded network sensor systems (SenSys)*, pages 439–440, November 2008.

[69] F. Ingelrest, G. Barrenetxea, G. Schaefer, M. Vetterli, O. Couach, and M. Parlange. "SensorScope: Application-specific Sensor Network for Environmental Monitoring". In *Transactions on Sensor Networks (TOSN)*, volume 6, issue 2, February 2010.

[70] W. Seah, Z. Eu, and H. Tan. "Wireless Sensor Networks Powered by Ambient Energy Harvesting (WSN-HEAP) - Survey and Challenges". In *1st International Conference on Wireless Communication, Vehicular Technology, Information Theory and Aerospace & Electronic Systems Technology (Wireless VITAE)*, pages 1–5, May 2009.

[71] P. Mell and T. Grance. "The NIST Definition of Cloud Computing". In *Computer Security*, September 2011.

[72] T. Dillon, C. Wu, and E. Chang. "Cloud Computing: Issues and Challenges". In *Advanced Information Networking and Applications (AINA)*, pages 27–33, April 2010.

[73] S. Patidar, D. Rane, and P. Jain. "A Survey Paper on Cloud Computing". In *Advanced Computing & Communication Technologies (ACCT)*, pages 394–398, January 2012.

[74] Y. Jadeja and K. Modi. "Cloud Computing - Concepts, Architecture and Challenges". In *Computing, Electronics and Electrical Technologies (CEET)*, pages 877–880, March 2012.

[75] R. Liu and I. Wassell. "Opportunities and Challenges of Wireless Sensor Networks Using Cloud Services". In *Internet of Things and Service Platforms (IoTSP)*, pages 27–33, December 2011.

[76] P. Juang, H. Oki, Y. Wang, M. Martonosi, L. Peh, and D. Rubenstein. "Energy-Efficient Computing for Wildlife Tracking: Design Tradeoffs and Early Experiences with ZebraNet". In *10th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS-X)*, pages 96–107, October 2002.

[77] E. Nadimi, R. Jørgensen, V. Blanes-Vidal, and S. Christensen. "Monitoring and Classifying Animal Behavior Using ZigBee-Based Mobile Ad Hoc Wireless Sensor Networks and Artificial Neural Networks". In *Computers and Electronics in Agriculture*, volume 82, pages 44–54, March 2012.

[78] K. Martinez, R. Ong, and J. Hart. "Glacsweb: A Sensor Network for Hostile Environments". In *First Annual IEEE Communications Society Conference on Sensor and Ad Hoc Communications and Networks (SECON)*, pages 81–87, 2004.

[79] Z. Butler, P. Corke, R. Peterson, and D. Rus. "Networked Cows: Virtual Fences for Controlling Cows". In *2nd international conference on Mobile systems, applications, and services*, 2004.

[80] B. Wietrzyk, M. Radenkovic, and I. Kostadinov. "Practical MANETs for Pervasive Cattle Monitoring". In *Seventh International Conference on Networking (ICN)*, pages 14–23, April 2008.

[81] Texas Instruments. "MSP430F22x2 MSP430F22x4 Mixed Signal Microcontroller". August 2012.

[82] Texas Instruments. "eZ430-RF2500 Development Tool User's Guide". April 2009.

[83] Texas Instruments. “MSP430F543xA, MSP430F541xA Mixed Signal Microcontroller”. August 2013.

[84] Texas Instruments. “Low-cost Low-power 2.4GHz RF Transceiver”. In *CC2500*, May 2009.

[85] SIMCom. “SIM908-C Hardware Design”. September 2011.

[86] SIMCom. “SIM908 AT Commands Set”. October 2011.

[87] K. Betke. “The NMEA 0183 Protocol”. May 2000.

[88] SiRF. “NMEA Reference Manual”. December 2007.

[89] Adafruit Industries. “Adafruit Ultimate GPS”. February 2014.

[90] GlobalTop Tech Inc. “GlobalTop MT3339 PMTK Command Packet”. 2012.

[91] K. Srinivasan and P. Levis. “RSSI is Under Appreciated”. In *Proceedings of the Third ACM Workshop on Embedded Networked Sensors*, 2006.

[92] J. Unger. “Deploying License-free Wireless Wide-area Networks”. In *Cisco Press*, February 2003.

[93] F. Ross and W. Ross. “The Jordan Curve Theorem is Non-trivial”. In *Journal of Mathematics and the Arts*, pages 1–4, January 2009.

[94] N. Chopde and M. Nichat. “Landmark Based Shortest Path Detection by Using A* and Haversine Formula”. In *International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)*, volume 1, issue 2, pages 298–302, April 2013.

[95] M. Costa, A. Goldberger, and C. Peng. “Multiscale Entropy Analysis of Biological Signals”. In *Physical Review E*, February 2005.