Universidade de Aveiro
2021

Luís Miguel
Tomé Nóbrega

Tecnologias IoT para Pastoreio e Controlo de Postura Animal

IoT Technologies for Animal Grazing and Posture Control

Universidade de Aveiro
2021

Luís Miguel
Tomé Nóbrega

Tecnologias IoT para Pastoreio e Controlo de Postura Animal

IoT Technologies for Animal Grazing and Posture Control

Tese apresentada às Universidades do Minho, Aveiro e Porto, para cumprimento das requisitos necessários à obtenção do grau de Doutor no âmbito do Programa de Doutoramento em Informática das Universidades do Minho, Aveiro e do Porto (MAP-I), realizada sob a orientação científica do Professor Doutor Pedro Alexandre S. Gonçalves, Professor Adjunto, Escola Superior de Tecnologia e Gestão de Águeda - Universidade de Aveiro e Professor Doutor Paulo Bacelar Reis Pedreiras, Professor Auxiliar, Departamento de Eletrónica, Telecomunicações e Informática - Universidade de Aveiro.

This work is supported by the European Structural Investment 760 Funds (ESIF), through the Operational Competitiveness and 761 Internationalization Programme (COMPETE 2020) [Project Nr. 762 017640 (POCI-01-0145-FEDER-017640)].

This work is funded by FCT/MEC through national funds and when applicable co-funded by FEDER—PT2020 partnership agreement under the project UID/EEA/50008/2019.

This work was also funded by ADVID - Associação para o Desenvolvimento da Viticultura Duriense, mainly through the assigning of the award "Prémio ADVID 2018 - "2º Visconde Vila Maior, Julio Máximo de Oliveira Pimentel".

À Carla e aos meus pais.

o júri / the jury

presidente / president	Doutor Fernando Joaquim Fernandes Tavares Rocha Professor Catedrático, Universidade de Aveiro.
vogais / examiners committee	Doutor Edmundo Heitor Da Silva Monteiro Professor Catedrático, Universidade de Coimbra
	Doutor Gabriel de Sousa Torcato David Professor Associado, Universidade do Porto
	Doutor António José Ribeiro Neves Professor Auxiliar, Universidade de Aveiro
	Doutor Pedro Alexandre de Sousa Gonçalves Professor Adjunto, Universidade de Aveiro
	Doutor Javier Lidiano Silvestre Blanes Senior Lecturer, Universitat Politècnica de València

agradecimentos / acknowledgements

Quem me conhece sabe que aceitar desafios faz parte da minha essência. Regressar à Universidade como doutorando foi o verdadeiro desafio. Regressei à procura da superação, mas também à procura de fazer um reset profissional. Necessitava de um projeto que me obrigasse a dar tudo de mim e, ao mesmo tempo, que me ajudasse a perceber qual o caminho profissional a seguir.

Quando me foi apresentado o projeto SheepIT fiquei desde logo convencido. Por um lado porque tocava áreas onde sempre tive interesse, mas acima de tudo porque tocava numa área que também me acompanha desde os tempos de infância, a agricultura. Tinha consciência dos desafios que me esperavam, não só a nível técnico-tecnológico mas sobretudo no balanço entre os objetivos do projeto e as necessidades de um Doutoramento. Devo admitir que, aliado a todos os outros desafios que ficarão a conhecer durante a leitura deste tese, este foi, provavelmente, o mais complicado de gerir, não só para mim, mas para toda a equipa que me acompanhou.

Foram muitos os que contribuiram para tornar este trabalho possível, e aos quais gostaria de agradecer de uma forma especial. Em primeiro lugar, quero agradecer à equipa que aceitou orientar-me. Dizem que equipa que ganha não se muda, e assim foi. Depois de me terem orientado durante a minha Dissertação de Mestrado, o Professor Doutor Pedro Gonçalves e o Professor Doutor Paulo Pedreiras aceitaram orientar-me mais uma vez, agora no Programa Doutoral MAP-i. Fomos, desde sempre, uma equipa que se complementa, algo fundamental para o sucesso. Por isso, por todo apoio e suporte, pelos conhecimentos e experiência transmitida, e pela amizade e motivação, o meu muito obrigado. Recordo-me, numa das primeiras reuniões, de terem-me dito que seria uma longa e sinuosa caminhada em que muitas vezes saímos fora da estrada em andamento. Lembro-me também de lhes ter transmitido que sou um homem de convições e que não os deixaria ficar mal. Dei o melhor de mim e, como prometido, cheguei à última paragem. Obrigado por me terem acompanhado.

Depois, quero agradecer do fundo do coração, a toda a equipa do SheepIT. Foram várias as pessoas com quem tive o privilégio de trabalhar, mas permitam-me enumerar algumas pela sua relevância. Ao Rui Morais, André Temprilho, António Cardoso, André Tavares e José Pereira, o meu muito obrigado por todos os momentos de trabalho (e não trabalho) que passámos, por todas as discussões relevantes (e não relevantes) com vista à construção da solução e por toda a paciência que tiveram ao longo das muitas maratonas.

Ao restante laboratório de sistemas embutidos do Instituto de Telecomunicações (Bruno Fernandes, João Rufino, Luís Silva e João Almeida), obrigado pelo companheirismo e pelas partilhas de conhecimento. Se não fomos o melhor laboratório que o Instituto de Telecomunicações já teve, ficamos lá muito perto. Queria ainda deixar o meu agredecimento ao grupo ATNOG, especialmente ao Mário Antunes e Daniel Corujo, por todo o suporte e ajuda numa área que me era inicialmente desconhecida. Ao Instituto de Telecomunicações e todos os seus colaboradores, obrigado por tão bem me terem recebido e por terem disponibilizado todas as condições para a realização do meu trabalho.

Deixo ainda uma mensagem especial de agradecimento às quatro entidades que foram fundamentais para a realização deste trabalho. Ao Instituto de Telecomunicações por me ter dado as condições necessárias à realização do trabalho. À Escola Superior Agrária de Viseu, nomeadamente por todo o apoio e ensinamentos a nível do comportamento animal. Muito ficou por fazer, mas muito mais teria ficado sem este suporte. À iFarmTec que nos disponibilizou dados fundamentais para a análise dos resultados dos primeiros protótipos sistema. Por fim, mas não menos importante, um obrigado à Ramos Pinto, no nome do Engenheiro Carlos Peixoto, e à Adega Luís Pato, pelo apoio prestado na análise às suas necessidades a nível da monda de vinha.

Um agradecimento também à Associação para o Desenvolvimento da Viticultura Duriense (ADVID), particularmente pela atribuição do "Prémio ADVID 2018 - "2º Visconde Vila Maior, Julio Máximo de Oliveira Pimentel" a um dos artigos produzidos. Foi um reconhecimento do trabalho realizado por toda a equipa.

Obrigado também à GoContact por me ter recebido de braços abertos, ainda com o meu Doutoramente em curso, e por ter confiado que seria capaz de o terminar e ao mesmo tempo ser competente nas minhas tarefas do dia a dia. Significou muito para mim.

Aos meus amigos e à minha família, especialmente aos meus pais, o meu sincero obrigado. Obrigado pela paciência e apoio incondicional em todos os instantes.

Por fim, o maior obrigado vai para quem escolhi para me acompanhar o resto da vida. Foram anos difíceis, muitas horas fechado no escritório quando, poderiamos estar noutro sítio qualquer, a fazer algo a dois. Não foi fácil, mas agradeço-te por me teres apoiado do início ao fim. Devo-te esta, devo-te tudo. Obrigado Carla Macedo.

Palavras Chave

Monitorização e Controlo de Postura, Internet das Coisas, Redes de Sensores sem Fios, Comunicações, Eficiência Energética, Ovelhas

Resumo

O constante crescimento de ervas infestantes obriga os produtores a manter um processo contínuo de remoção das mesmas com recurso a mecanismos mecânicos e/ou químicos. Entre os mais populares, destacam-se o uso de arados e roçadores no primeiro grupo, e o uso de herbicidas no segundo grupo. No entanto, estes mecanismos são considerados agressivos para as videiras, assim como no segundo caso perigosos para a saúde pública, visto que os químicos podem permanecer no ambiente, contaminando frutos e linhas de água. Adicionalmente, estes processos são caros e exigem mão de obra que escasseia nos dias de hoje, agravado pela necessidade destes processos necessitarem de serem repetidos mais do que uma vez ao longo do ano. O uso de animais, particularmente ovelhas, para controlar o crescimento de infestantes é uma prática ancestral usada em todo o mundo. As ovelhas, enquanto pastam, controlam o crescimento das ervas infestantes, ao mesmo tempo que fertilizam o solo de forma gratuita, ecológica e sustentável. Não obstante, este método foi sendo abandonado visto que os animais também se alimentam da rama, rebentos e frutos da videira, provocando naturais estragos e prejuízos produtivos.

Para mitigar este problema, uma nova solução baseada em tecnologias de Internet das Coisas é proposta no âmbito do projeto SheepIT, cuja espinha dorsal foi construída no âmbito desta tese. O sistema monitoriza as ovelhas enquanto estas pastoreiam nas vinhas, e implementam um mecanismo de controlo de postura que condiciona o seu comportamento de forma a que se alimentem apenas das ervas infestantes. O sistema foi incorporado numa infraestrutura de Internet das Coisas com comunicações sem fios de baixo consumo para recolha de dados e que permite semanas de autonomia, mantendo os dispositivos com um tamanho adequado aos animais.

Neste contexto, a tese suportada neste trabalho defende que *é possível projetar uma sistema baseado em tecnologias de Internet das Coisas, capaz de monitorizar e condicionar a postura de ovelhas, permitindo que estas pastem em vinhas sem comprometer as videiras e as uvas*. A tese é suportada em três pilares fundamentais que se refletem nos principais contributos do trabalho, particularmente: a arquitetura do sistema e respetivo sistema de comunicações; o mecanismo de controlo de postura; e o suporte para implementação de um sistema de localização de baixo custo e baixo consumo energético. A arquitetura é validada em contexto de simulação, e o mecanismo de controlo de postura em contexto de simulação e de experiências em campo. É também demonstrado o funcionamento do sistema e o contributo deste trabalho para a conceção da primeira versão comercial do sistema.

Keywords

Monitoring and Posture Control, Internet of Things, Wireless Sensor Networks, Communications, Energy Efficiency, Sheep

Abstract

The unwanted and adverse weeds that are constantly growing in vineyards, force wine producers to repeatedly remove them through the use of mechanical and chemical methods. These methods include machinery such as plows and brushcutters, and chemicals as herbicides to remove and prevent the growth of weeds both in the inter-row and under-vine areas. Nonetheless, such methods are considered very aggressive for vines, and, in the second case, harmful for the public health, since chemicals may remain in the environment and hence contaminate water lines. Moreover, such processes have to be repeated over the year, making it extremely expensive and toilsome. Using animals, usually ovines, is an ancient practice used around the world. Animals, grazing in vineyards, feed from the unwanted weeds and fertilize the soil, in an inexpensive, ecological and sustainable way. However, sheep may be dangerous to vines since they tend to feed on grapes and on the lower branches of the vines, which causes enormous production losses. To overcome that issue, sheep were traditionally used to weed vineyards only before the beginning of the growth cycle of grapevines, thus still requiring the use of mechanical and/or chemical methods during the remainder of the production cycle.

To mitigate the problems above, a new technological solution was investigated under the scope of the SheepIT project and developed in the scope of this thesis. The system monitors sheep during grazing periods on vineyards and implements a posture control mechanism to instruct them to feed only from the undesired weeds. This mechanism is based on an IoT architecture, being designed to be compact and energy efficient, allowing it to be carried by sheep while attaining an autonomy of weeks.

In this context, the thesis herein sustained states that *it is possible to design an IoT-based system capable of monitoring and conditioning sheep's posture, enabling a safe weeding process in vineyards*. Moreover, we support such thesis in three main pillars that match the main contributions of this work and that are duly explored and validated, namely: the IoT architecture design and required communications, a posture control mechanism and the support for a low-cost and low-power localization mechanism. The system architecture is validated mainly in simulation context while the posture control mechanism is validated both in simulations and field experiments. Furthermore, we demonstrate the feasibility of the system and the contribution of this work towards the first commercial version of the system.

Contents

List of Figures

List of Tables

Glossary

3GPP	3rd Generation Partnership Project
6LoWPAN	IPv6 Low-Power Wireless Networks
6TiSCH	IPv6 over the TSCH
AdaBoost	Adaptive Boosting
AI	Artificial Intelligence
AL	Adaptation Layer
AMQP	Advanced Message Queuing Protocol
ANN	Artificial Neural Network
AoA	Angle of Arrival
API	Application Programming Interface
ASMV	Average Signal Magnitude Vector
AUC	Area Under the Curve
B2B	Beacon-to-Beacon
B2G	Beacon-to-Gateway
BS	Beacon Syncronization Message
BLE	Bluetooth Low Energy
BN	Beacon Notification
C2B	Collar-to-Beacon
CDA	Canonical Discriminent Analysis
CFS	Correlation-based Feature Selection
CN	Collar Notification
C-F	Connectivity + Fusion
COAP	Constrained Application Protocol
COTS	Commercial Off-The-Shelf
CP	Computacional Plataform
CPS	Cyber-Physical Systems
CRUD	Create, Read, Update and Delete
CSMA	Carrier Sense Multiple Access
CSMA/CA	Carrier Sense Multiple Access - Collision Avoidance
DA	Discriminent Analysis
DC	Duty-Cycle
DF	Dominant Frequency
DFT	Discrete Fourier Transform
DM	Data Mining
DODAG	Destination-Oriented Directed Acyclic Graph
DT	Decision Tree
DT-GN	Distance Tracking and joint localization with Gauss-Newton
ICV	Inverse Coefficient of Variation
ICT	Information and Communication Technologies
IETF	Internet Engineering Task Force
IP	Internet Protocol
IPv6	Internet Protocol Version 6
IoT	Internet of Things
IQR	Interquartile Range
ISA	International Society of Automation
ISM	Industrial, Scientific and Medical
JSON	JavaScript Object Notation
ETSI	European Telecommunications Standards Institute
FDA	Flexible Discriminant Analysys
Flexi-TP	Flexible-Schedule-Based TDMA Protocol
FIFO	First-In-First-Out
FN	False Negatives
FP	False Positives
G2B	Gateway-To-Beacon
GPRS	General Packet Radio Service
GPS	Global Positioning System
GSM	Groupe Spécial Mobile or Global System for Mobile Communications
GW	Guarding Window
HMM	Hidden Markov Model
HTTP	Hypertext Transfer Protocol
KDD	Knowledge Discovery from Databases
KF	Kalman Filter
kNN	k-Nearest Neighbour
LDA	Linear Discriminent Analysis
LEFT	Latency and Energy Efficient Flexible TDMA Protocol for Wireless Sensor Networks
LFP	Localization and Fencing Periodicity
LWM2M	Lightweight M2M
LNSM	Log-Normal Shadow Model
LoRa	Long Range
LPWA	Low-Power Wide-Area
LPWAN	Low-Power Wide-Area Networks
LTE	Long-Term Evolution
M2M	Machine-to-Machine

MAC	Medium Access Control
MC	Macro-Cycle
MCC	Matthews Correlation Coefficient
MCR	Mean Crossing Rate
ML	Machine Learning
MO	Medium Occupation
MOM	Message Oriented Middleware
MP	Monitoring Periodicity
MSC	Message Sequence Chart
MTU	Maximum Transmission Unit
MV	Movement Variation
MQTT	Message Queuing Telemetry Transport
NB-IOT	Narrowband-IoT
nC	Nano-Cycle
NFC	Near Field Communication
NN	Neural Networks
OF	Objective Function
OW	Observation Window
PR	Pairing Request/Response
QDA	Quadratic Discriminent Analysis
QoS	Qualite of Service
RB	Rule-Based classifier
RDA	Regularized Discriminant Analysis
RDF	Random Decision Forest
REST	Representational State Transfer
RF	Radio Frequency
RFID	Radio Frequency Identification
RI-EDF	Robust Implicit - Earliest Deadline First
RMS	Root Mean Squarte
ROC	Receiver Operating characteristic Curve
ROLL	Routing Over Low Power and Lossy Networks
RPL	Routing Protocol for Low-Power and Lossy Networks
RSSI	Received Signal Strength Indicator
SDA	Stepwise Discriminant Analysis
SE	Spectral Entropy
SMA	Signal Magnitude Area
SMV	Signal Magnitude Vector
SSL	Secure Sockets Layer
SW	Synchronization Window
SVM	Support Vector Machine
SWaP-C	Size, Weight, Power and Cost
S-MAC	Sensor MAC
TAW	Turn-Around Window
TCP	Transmission Control Protocol
TDMA	Time Division Multiple Access
TDoA	Time Difference Of Arrival
TELCO	Telephone Companies
TIA	Telecommunications Industry Association
TLS	Transport Layer Security
TT	Time-Triggered
ToA	Time of Arrival
TN	True Negatives
TP	True Positives
T-MAC	Timeout MAC
UDP	User Datagram Protocol
UHF	Ultra High Frequency
UKF	Unscented Kalman Filter
WAN	Wide Area Network
WBAN	Wireless Body Area Networks
WirelessHART	Wireless Highway Addressable Remote Transducer Protocol
Wise-MAC	Wireless Sensor MAC
WLAN	Wireless Local Area Networks
WPAN	Wireless Personal Area Networks
WWAN	Wireless Wide Area Networks
WSN	Wireless Sensor Networks
VTW	Variable Traffic-type Window
ZCR	Zero Crossing Rate
Z-MAC	Zebra MAC

CHAPTER 1

Introduction

One of the most critical challenges that present and future generations need to face is how to balance the unending digital and industrial revolutions against the environmental and societal sustainability. Thus, research shall increasingly focus on technological solutions that could contribute to a higher sustainability of critical societal sectors, as industry, transportation or agriculture.
This thesis intends to be a step forward towards the development of an autonomous Internet of Things (IoT)-based system capable of monitoring and conditioning sheep's posture while weeding vineyards. With this in mind, this chapter contextualizes the motivation that sustained the problem definition, presents the thesis statement and enumerates the main contributions. Finally, it ends with the document organization.

1.1 Motivation

The continuous growth of the world population and their demands are triggering important changes in the primary sector. The quantity of goods required to satisfy the current needs is getting increasingly higher, reaching a point where the humanity is consuming more resources than the planet has to offer. Regarding specifically human nutrition, there was a wide adoption of intensive and non-sustainable farming solutions that have a negative impact in the environment. Albeit being utopic to think, at least in the short term, about solutions capable of satisfying the world population nutrition needs without harmful environment effects, it is of utmost importance to at least develop ways to minimize such effects.

Information and Communication Technologies (ICT) are being adopted to increase the efficiency in the production of goods and to reduce the cost, environmental impact and amount of human labour [1], both in the food and livestock sectors. Hence, precision agriculture, e-agriculture and intelligent farming systems arose, supported by tools such as Ubiquitous Computing, Cloud Computing, Satellite Monitoring, Remote Sensing, Context-Aware Computing and IoT [2].

The viticulture sector represents a paradigmatic example where the application of these technologies is being taken seriously by producers. Attaining high quality wines is the main goal of winemakers and can only be achieved through continuous enhancements in the production processes, from the vine's management to the wine production and storage.

Weed control, *i.e.*, the control of the growth of unwanted and undesirable weeds, is a critical practice in vineyard management. These infestant weeds, besides competing with the vines for soil nutrients, water and sunlight, may increase the incidence of diseases due to lack of air circulation, increase the cover for rodents and contribute to a lower harvest efficiency [3]. Hence, producers are forced to repeatedly remove those weeds, both between the vines rows and within the row[1] [4].

Although between rows different methods, such as mowing or shredding, may be used without threatening the vines, the same is not true when applying similar methods to the space between vines. Traditionally these mechanical methods include different agricultural machinery for tillage and mowing, commonly equipped with an automatic vine-skipping mechanism to minimize the risk of damaging vines. However, besides carrying risks to the vines, the process is very time-consuming, since it must be done very carefully and several times through the year, and its efficiency is very limited, since it does not clean all weeds and needs to be repeated. Brushcutters may present a better efficiency and accuracy, but besides being also labour intensive, incur in a even higher risk to the vines [5].

The disadvantages inherent to the mechanical methods have led to the increase of popularity of chemical mechanisms. These methods are cheaper, easier and faster to perform than the mechanical ones and, when correctly applied, may not only minimize the negative impacts in the crop but also be part of the suckering process [2] [3].

Notwithstanding, even when all due care is taken when handling and applying these chemical herbicides, injuries may occur on the grapes due to drift effects, and groundwaters and the fruits may be contaminated [6]. Additionally, the increasing concerns on the effects of these chemical components in the human health are pushing producers to abandon, or at least reduce, this kind of solutions.

All the referred limitations and drawbacks are even more perceived by producers from Douro's Region (Portugal). Here, the high slopes of terrains, together with the typical vines placed in terraces, make both mechanical and chemical mechanisms even more demanding, both in time, cost and labour [7].

Consequently, the viticulture and viniculture sectors are currently in an incessant pursuit for weed control alternatives, less dependent on mechanical methods and free of chemical products - a very trendy and valued aspect in the wine market.

The use of animals to weed vineyards [8], [9], usually sheep, is an ancient practice used around the world. Grazing sheep in vineyards feed from the unwanted weeds and fertilize the soil, in an inexpensive, ecological and sustainable way. However, this solution was progressively abandoned because it could not be used through all the year. Since animals tend to feed from the vines and their fruits, troublesome production losses would be caused if they are not retired before the beginning of the growth cycle of grapevines. Thus, the use of mechanical and/or chemical methods were still required. Moreover, the lack of shepherds, the rise of their wages and the difficulties inherent to the herd management, led to the abandonment of this kind of practice.

Nowadays, however, there is an increasing interest on returning to this old practice. In fact, producers look at this practice as an excellent alternative to reduce the use of the harmful methods described, as well as a tool to face against the expensive and scarce manpower. Nonetheless, shaping this practice to a viable and feasible reality, demands the development of several tools. This unleashed the born of the SheepIT project [10], the cornerstone of this thesis.

[1]A vine is constituted of lines of vines. The weed control must be performed between the rows of the vines but also between and around the vines in the same line.

[2]The suckering process, also known as shoot thinning, consists on the removal of unnecessary buds from the vines.

1.2 Problem statement

Using sheep as a weed control method presents a great potential, both in efficiency and environmental and economical sustainability. However, albeit being an ancient method, nowadays its application strategy must be clearly different. In ancient times, resorting on sheep to weed vineyards required the constant permanence of a shepherd, an occupation in disruption nowadays. Furthermore, even with the presence of a shepherd, noticeable and relevant damages could occur, since it is impossible to monitor and control dozens or hundreds of sheep at the same time.

The SheepIT project intended to take advantage of the wide range of powerful IoT technologies being developed to make viable the use of sheep as a weed control method. Hence, the goal of the project, and thus of this thesis, was to develop an IoT-based system as a tool for allowing the use of sheep for controlling weed growth in vineyards and similar cultures. This system shares many of the common IoT features and requirements, such as:

- there is a need of data exchange between IoT devices;
- the amount of data exchanged is low and size-limited;
- a high autonomy is required, particularly on devices where the replacement or recharge of batteries is arduous or inconvenient;
- the geographical area to be covered can be vast;
- robustness and reliability are required.

Besides these general requirements, the SheepIT solution needs to cope with a set of additional requirements that, when combined with the aforementioned ones, singularize this solution in the state-of-the-art of animal monitoring and IoT platforms for intelligent farming. These requirements are:

- support for an effective posture control mechanism;
- operation environment that can include terrains with irregular slopes and obstacles;
- unforeseeable node mobility (sheep);
- variable node densities due to the flock movement;
- need for real-time localization and virtual fencing support, preferably resorting to Received Signal Strength Indicator (RSSI);
- need to handle frequent system reconfigurations by non-technical personnel.

Considering the needs of the SheepIT project, the foundation of this thesis, three main research questions, intimately related with the technological needs of the solution, were defined:

> **Research question 1:** *How shall the IoT-based platform be designed to meet the defined requirements?*

A tangible component of this thesis is the design and validation of an IoT-based architecture capable of providing a deployable platform for using sheep as a weed control method. This architecture, besides addressing all the defined requirements, shall exploit the identification and composition of system's devices, the means to ensure a seamless interaction between them and mechanisms to make the platform easy to be used by farmers. Although some of the defined requirements are not new in the research community, the combination of all of them carries additional challenges in the definition of the solution's architecture, being discussed in the scope of this thesis.

Research question 2: *Which are the adequate means to ensure a low-power communication mechanism without compromising system requirements?*

Communications are typically responsible for a large portion of energy consumption in any kind of IoT system. Thus, it has important repercussions in system autonomy, which is critical when dealing with devices that may present several constraints on battery recharges or replacements. Therefore, it is important to investigate energy-aware radio technologies and protocols that could comply with such need while guaranteeing the requirements related to the number of devices, timing constraints and localization needs.

Research question 3: *How to grant low-power devices with real-time animal posture control capabilities?*

Monitoring animal behaviour is a quite common activity in livestock industry. Typically those applications support their operation in sensing devices placed on animals that are capable of identifying behaviours (mainly using accelerometry) or locations (mostly using Global Positioning System (GPS)). Regarding the former, most applications employ offline monitoring methods, storing the sensor data in devices' memory for further analysis using powerful computers. Concerning the latter, GPS, that are considerable expensive and energy demanding for IoT applications, are typically used for localization, being the coordinates stored locally or sent via cellular networks to central platforms. However, to the best of our knowledge, deploying an embedded real-time posture control mechanism for weeding sheep in vineyards, is a totally new approach.

1.3 Thesis statement

In light of the research questions above disclosed, the thesis that is sustained in this work states that:

It is possible to design an IoT-based system capable of monitoring and conditioning the sheep's posture, enabling a safe animal weeding process in vineyards. Particularly, the thesis argues that the following key features can be integrated into a single low-power system: embedded real-time animal posture control, data collection and data exchange.

1.4 Main contributions

The main contribution of this thesis is the proposal and validation of an IoT-based architecture that enables the use of sheep as a weed control method in vineyards. As, in the time frame of the thesis development, it is unfeasible of deeply address all the issues involved in the architecture definition, the work focus in three main pillars:

- **The IoT architecture and required communications:** this architecture is the base of the remainder work since its design and implementation not only depend on all system requirements and features, but also because the intended features for the system depend on the successful development of the architecture. In a system where the data exchange is one of the key features, the development of an adequate communication mechanism between devices is crucial. This component of the work is sustained by three main publications:

  [11]. L. Nobrega, P. Pedreiras, P. Goncalves, and S. Silva, "Energy efficient design of a pasture sensor network," in Proceedings of the - *IEEE 5th International Conference on Future Internet of Things and Cloud (FiCloud 2017)*, Prague, Czech Republic, 2017, vol. 2017-January, pp. 91–98.

  This work explores the state-of-the-art of wireless communication solutions and defines the general architecture of the system, particularly concerning the communication protocol within the Wireless Sensor Networks (WSN). Also, an evaluation of the system scalability is presented considering the expected application scenarios.

  [12] Temprilho, L. Nobrega, P. Pedreiras, P. Goncalves, and S. Silva, "M2M Communication stack for intelligent farming," in Proceeding of the *2018 Global Internet of Things Summit*, Bilbao, Spain, 2018.

  This paper is an extension of the theoretical work presented in [11], since the focus is the generalization of the communication stack such that it could be used by similar intelligent farming solutions. Also, a preliminary validation of the implementation of some of its layers is presented, such as the Physical/Medium Access Control (MAC) Layer and the Application Layer.

  [13] Nóbrega, P. Gonçalves, P. Pedreiras, and J. Pereira, "An IoT-based solution for intelligent farming," *Sensors (Switzerland)*, vol. 19, no. 3, 2019.

  Besides reviewing the proposed system's architecture, it extends the IoT communication stack and focuses in the design and deployment of a gateway to address the system requirements. The gateway performance is evaluated together with its feasibility and scalability considering real scenario conditions.

- **The posture control mechanism:** Monitoring typical animal behaviours, such eating, running and moving, is not a novelty in the research domain. Nevertheless, the same is not truth when the goal is real-time monitoring of specific posture behaviours of sheep, particularly the detection of situations where they feed from vines, using low-power devices. Also, the combination between the monitoring and the conditioning mechanisms (named posture control mechanism from now on) were addressed, resulting in three publications:

[14] L. Nobrega, A. Tavares, A. Cardoso, and P. Goncalves, "Animal monitoring based on IoT technologies," in Proceedings of the *2018 IoT Vertical and Topical Summit on Agriculture*, Tuscany, Italy, 2018, pp. 1–5.

This work demonstrates the system monitoring capabilities. It focuses in two main components: the computational platform architecture; and the use of Machine Learning (ML) for handling animal monitoring data, particularly for detecting sheep's posture.

[15] L. Nóbrega, P. Pedreiras, and P. Gonçalves, "SheepIT, an IoT-Based Weed Control System," *Communications in Computer and Information Science*, vol. 953, pp. 131–147, 2019.

This journal article presents the state-of-the-art on animal monitoring solutions and details the posture control mechanism, including the hardware used, the methods used to achieve the posture control mechanism and the validation of such mechanism.

[16] L. Nóbrega, P. Gonçalves, M. Antunes, and D. Corujo, *"Assessing sheep behavior through low-power microcontrollers in smart agriculture scenarios"* Computers. Electronics in Agriculture, vol. 173, 2020.

This work conducted a systematic ML problem solving approach considering state-of-the-art works in animal monitoring but with the states required for the this application. The focus was on low-processing requirements and human readable algorithms that could be easily implemented in low-power microcontrollers. Conclusions from previous works were used to provide an enhanced behaviour monitoring mechanism for sheep.

Besides the central publications presented above, there are additional publications that, despite not being directly addressed in this document, are directly related with the thesis development and hence must be identified, namely:

[17] L. Nóbrega, P. Pedreiras, and P. Gonçalves, "SheepIT - An electronic shepherd for the vineyards," in Proceedings of the 8th International Conference on Information and Communication Technologies in Agriculture, Food & Environment (HAICTA), Chania, Crete, Greece, 2017, vol. 2030, pp. 621–632.

Presented in a focused agriculture conference, this paper aimed at highlighting the innovative aspects of the system. Particularly, it details the state-of-the-art on animal monitoring solutions and the simplified architecture of the system. It also includes the presentation of some of the system's hardware.

[18] L. Nóbrega, P. Gonçalves, P. Pedreiras, R. Morais, and A. Temprilho, "SheepIT: Automated Vineyard Weeding Control System," in Proceedings of the *INFORUM simpósio de informática*, Aveiro, Portugal, 2017.

This communication corresponded to a national dissemination of the work, particularly regarding the innovative and distinctive components of the solution.

[19] A. Cardoso, J. Pereira, L. Nóbrega, P. Gonçalves, P. Pedreiras, and V. Silva, "SheepIT: Activity and Location Monitoring," in Proceedings of the INForum 2018 - Simpósio de Informática, Coimbra, Portugal, 2018, pp. 1–12.

As the system designed shall enable the development of a RSSI-based localization mechanism to track sheep's location, some exploratory experiments were made. The first approach was presented in this paper. The goal was to show the enabling capabilities of the communication scheme to gather the necessary RSSI data to be processed. However, high variability on RSSI measurements were found, as expected, requiring the development of additional mechanisms to minimize such effects.

[20] R. Guedes, P. Pedreiras, L. Nóbrega, and P. Gonçalves, "Towards a low-cost localization system for small ruminants," Comput. Electron. Agric., vol.185, 2021.

This work represents a step forward towards the development of an RSSI-based localization system using the architecture and system developed in the scope of this thesis. It exploits questions such as the use of a calibration process to reduce hardware's variability and implements filtering techniques to reduce RSSI variability. Despite showing the potentiality of such approach, it also shows that there is still relevant work to be done to implement a viable localization system.

1.5 Document Structure

Tackling the research questions and supporting the thesis formulated in this introductory chapter, required to embrace a challenging research journey that is documented as follows:

- **Chapter 2** introduces the basic concepts of IoT, details popular architectures and exploits several communications protocols. It also tackles Machine Learning processes, tools and algorithms. Finally, it introduces localization techniques to enable the design of a system capable of supporting the implementation of a RSSI-based localization mechanism;
- **Chapter 3** presents an overview of related animal monitoring and conditioning works. It focuses in the ones that are closely related with the thesis' application domain;
- **Chapter 4** details the design of the IoT architecture proposed, including rationale, components and protocols;
- **Chapter 5** tackles the implementation of the IoT architecture defined in Chapter 4, with emphasis on the communications features;
- **Chapter 6** exploits the methodological approach towards the development of the posture control mechanism. It also addresses the integration of this mechanism into collars operation;
- **Chapter 7** provides a tool for assessing system scalability and relevant metrics such as power consumption, autonomy and medium occupation;

- **Chapter 8** is devoted to the evaluation and discussion of results, focusing on the results provided during the evaluation of both communication infrastructure and posture control mechanism;
- **Chapter 9** concludes the thesis, summarizes the milestones achieved and draws some ideas about future work.

CHAPTER 2

Background

Designing a system architecture capable of fulfilling the requirements identified in Chapter 1, demands knowledge in different areas, several of which well established in the literature. Thus, we review the most relevant topics used for the conception of the system's architecture. They are the IoT domain, with special emphasis on energy constrained protocols, localization mechanisms and ML problem solving workflow, algorithms and metrics.

2.1 Internet of Things fundamentals

The IoT domain embraces a huge number of applications with different goals, features, requirements and constraints. Consequently, to solve the specific needs of such applications, a wide range of communication models, protocols and standards have been emerging, being either applied on WSN, Machine-to-Machine (M2M) or Cyber-Physical Systems (CPS) [21]. Considering the needs of the agricultural sector, the first part of this section presents a review of the most relevant communication technologies, protocol stacks and standards. Additionally, we evaluate if their use could solve the research questions of this thesis.

2.1.1 IoT/M2M protocols and communication technologies

IoT encompasses an ample number of communication networks, either WSN, M2M, CPS, or a combination of them. From these, M2M communications are seen as the foundation of IoT platforms [22] since they allow the ubiquitous and autonomous communications between devices without human intervention [23]. M2M networks comprise both wired and wireless solutions, but the increasing need of mobility and scalability, together with the inherent high installation costs of wired solutions, have led to an increasing number of M2M wireless options. These can be organized into three groups, as depicted in Figure 2.1. Firstly, the *capillary* solutions [24], where a local network coexists with a local gateway that interacts with Wide Area Networks. Some examples are the Wireless Local Area Networks (WLAN)s as WiFI or HiperLan, and the Short-range Wireless solutions such as Bluetooth, ZigBee and Z-Wave. These solutions are characterized by a data rate that can go from bits per second to megabits per second. Secondly, the popular cellular networks, which have a greater communication range comparing to the capillary solutions, being the 2G, 3G, 4G, 5G and Long-Term Evolution (LTE), some examples. Finally, addressing the specific IoT/M2M communication requirements, particularly power consumption and autonomy, a new set of technologies started to gain relevance, the denominated Low-Power Wide-Area Networks (LPWAN). These technologies combine the wide range of cellular networks with the lower power consumption of *capillary* solutions [25].

Cellular networks and WLANs are not suitable for the purpose of this work due to their high

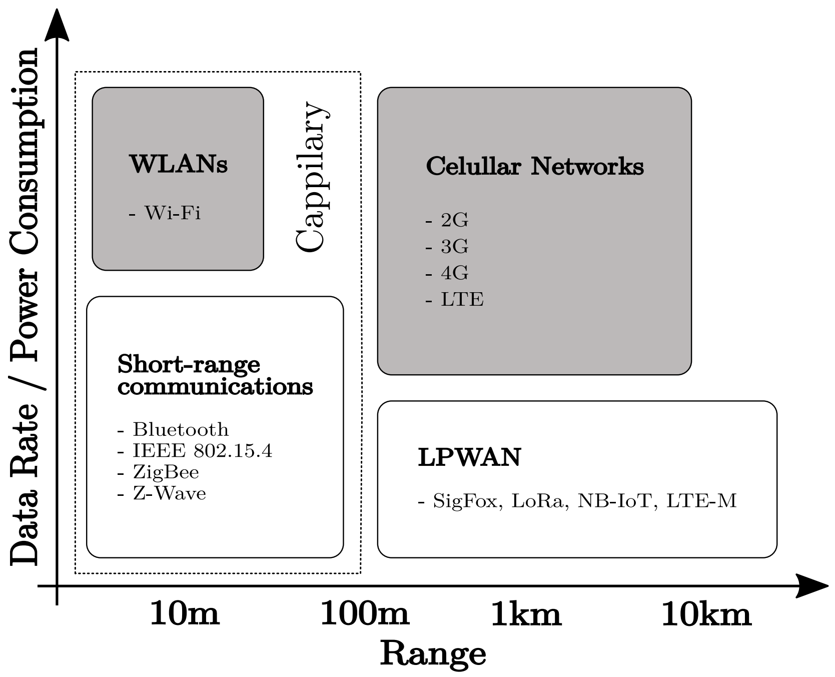


Figure 2.1: Communication technologies in IoT - major groups (based on [25]).

power consumption and price. Thus, our targets were the short-range wireless communications and LPWAN solutions. Among these, some may resort on gateways to interact with other networks while others resort on proprietary protocols.

This heterogeneity of approaches has led to the definition of distinctive IoT protocol stacks models [21] with 3, 4 or 5 layers. The simplest one is based on three layers: *i)* Perception Layer, which is the lowest layer and includes all the functionalities related with physical sensors; *ii)* Network Layer, which is responsible for ensuring the seamless transmission of data gathered on the Perception Layer to the Application Layer; *iii)* and Application Layer that allows the customization of different services according to user requirements and needs. However, this approach is too simplistic and hence a model of 5 layers gained popularity [26]. On this model, besides the Perception and Application Layers, already included in the 3-layer model, three other layers are included, namely: *i)* Transport or Object Abstraction Layer, responsible for receiving data gathered on the Perception Layer and make it available to upper layers; *ii)* Processing, Service Management or Middleware Layer that deals with the huge amount of data that is gathered and with the big heterogeneity of objects within the Perception Layer; and *iii)* Business Layer, that allows the management of the whole IoT system.

Notwithstanding, due to the interoperability and popularity of Internet Protocol (IP), many practical deployments of IoT applications follow a typical 4 layer Transmission Control Protocol (TCP)/IP architecture (Physical and MAC Layer, Network Layer, Transport Layer and Application Layer) with the necessary adaptations, particularly in terms of the protocols that can be used on each layer. For this reason, we decided to follow this approach as it will become clear on Chapter 4. As such, we narrowed the investigation on this approach and explain which protocols are commonly used on each one of those layers, as suggested by Naik [27] and illustrated on Figure 2.2. Furthermore, after reviewing those protocols we also survey two additional approaches. On the one hand, we survey the LPWAN technologies because of their potentiality when applied to IoT/M2M scenarios. On the other hand, we review Non-IP based stacks, which are relevant because there are IoT application domains served by constrained devices, as the one where this thesis is inserted on, with energy, processing and radio bandwidth constraints that are so severe that cannot be handled by the remaining stacks.

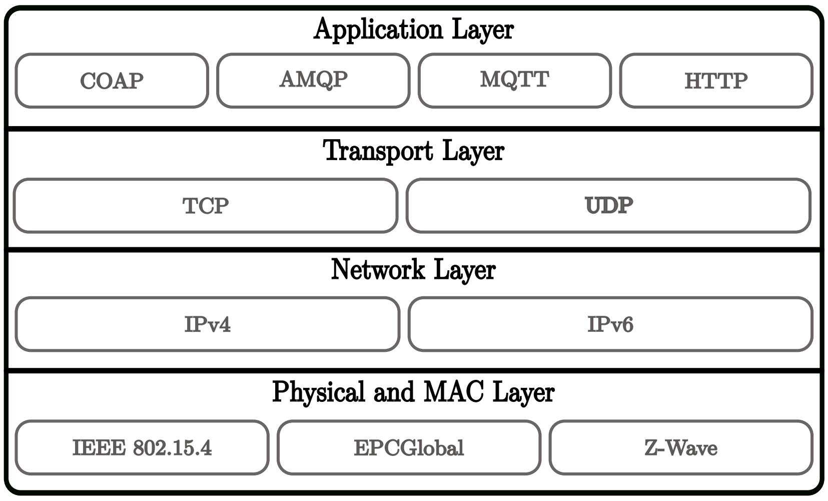


Figure 2.2: Popular Internet of Things standards on a IP-based solution (based on [21]).

2.1.2 Physical and MAC Layer protocols

IEEE 802.15.4 still is one of the most popular standards within the Physical and MAC layers. It was designed with three main requirements in mind: low consumption, low complexity and low cost [28]. Hence, it was adopted by several relevant protocols, such as ZigBee, ISA100 and WirelessHART. It operates mainly in the Industrial, Scientific and Medical (ISM) band of 2.4GHz, although it can also operate in the 868MHz (in Europe) and 915MHz (in the United States of America) bands. It supports data rates up to the 250 kbps and, concerning the MAC Layer, both Carrier Sense Multiple Access - Collision Avoidance (CSMA/CA) and Time Division Multiple Access (TDMA) (in the beacon-enabled mode) are supported. However, the literature reports significant limitations of the latter mode, particularly regarding the network formation [29] and mobility [30]. Besides IEEE 802.15.4, several alternatives emerged at the Physical Layer but, or are not directly inter-operable with IP, requiring intermediate layers or Gateways, as for instance Z-Wave, designed particularly for domotics [31] and EPC-Global for RFID technologies [32]; or are still not adequate for constrained and remote intelligent farming applications, as it happens with LPWAN.

Due to their relevance on constrained networks, several MAC Layer protocols have been proposed along with the referred commercial solutions and that are nowadays part of the background in the area. Generally, MAC protocols can be classified into contention-centered, scheduled-centered or hybrid [33], that are aligned with the use of Carrier Sense Multiple Access (CSMA),TDMA-based techniques or a mix of both. Contention-centered approaches are CSMA-based, and a node transmits only if no activity is detected on the medium. There are several methods to control collisions, usually based on a process of waiting a random amount of time before transmitting and/or retransmitting a packet (back-off). Although allowing simple implementations, for high bandwidth utilizations, CSMA becomes inefficient due to the increasing number of collisions and back-off window durations. The combination of these effects increases the packet loss, degrades the channel utilization and fairness and increases the power consumption [33].

To minimize power consumption in CSMA-based protocols, some solutions support their operation in duty cycles to allow nodes staying in sleep mode during part of the operating cycle. Sensor MAC (S-MAC) [34] uses static duty cycles, while Timeout MAC (T-MAC) [35] uses duty cycles dynamically adjusted according to the existing traffic. However, these protocols still waste time and consequently power in idle listening. To reduce that time, Preamble-based Medium Access mechanisms were proposed [36]. Using this type of approach, nodes that want to transmit, start by transmitting preambles, while the remaining nodes wake up periodically to listen for the medium. If the medium is found to be busy, the nodes wait for data, if not, nodes switch to sleep mode. The preambles can be long as in the case of CSMA with Preamble Sampling [37], short as in the case of X-MAC [38] or short with synchronization, as used in Wireless Sensor MAC (Wise-MAC) [39]. From these, there were multiple variants, adapted to optimize certain behaviours but that are supported on the same principles [40]. These protocols, although more energy-wise comparing to simple CSMA-based MAC protocols, they still incur in excessive overhearing, waste of bandwidth in the case of long preambles and extra overhead in the case of short preambles.

When the traffic is periodic and has known size, TDMA protocols are potentially more efficient, since they allow avoiding idle listening, controlling the bandwidth utilization and guaranteeing fairness for all the communications, in exchange of issues such as higher complexity, lower flexibility and the need for global clock synchronization. The Latency and Energy Efficient Flexible TDMA Protocol for Wireless Sensor Networks (LEFT) protocol [33] is a centralized solution, being the dynamic slots scheduled by a master. BodyMAC [41] was developed for Wireless Body Area Networks (WBAN), for human body sensing. It resorts on a CSMA window for transmitting controlling messages and a TDMA window for transmitting data messages. Beacon messages are used for network synchronization and the time slots are scheduled by a gateway node. Robust Implicit - Earliest Deadline First (RI-EDF) [42] was developed for Real-Time systems where Qualite of Service (QoS) and packet delivery is essential. It mitigates the issues of clock synchronization and bandwidth waste but it doesn't perform well in terms of energy efficiency as well as it assumes that all nodes listen all the traffic.

Zebra MAC (Z-MAC) [43] is an hybrid CSMA/TDMA mechanism that uses CSMA when the traffic load is low, switching to a TDMA-based scheme for higher traffic loads. Flexible-Schedule-Based TDMA Protocol (Flexi-TP) [44], on the other hand, has two phases, an initial setup of the network where CSMA/CA and token ring are used for assigning the time slots, and a gathering data phase, where nodes transmit data in individual time slots. All nodes maintain local information about the other nodes in the network, which enables dynamic changes in real-time.

To support bursty traffic, QueueMAC [45] was proposed, where nodes can ask for more time slots through the use of polling packets. However, these polling packets need a good global synchronization and have a reduced maximum number of hop neighbors. iQueue-MAC [46] allows the assignment of dynamic time slots accordingly to the length of the data queue. The time slots are assigned right upon the detection of data in the queue, which allows the time slots to be quickly assigned using a multi-channel approach. Regardless of their good adaptation to high load traffic applications, this kind of approaches are complex to implement, resulting in significant processing requirements.

In industrial process control, Wireless Highway Addressable Remote Transducer Protocol (WirelessHART) [47] and International Society of Automation (ISA)100 [48] are two of the most widely used protocols. They ensure secure and real-time communications in wireless mesh networks operating in the 2.4GHz ISM band. The referred protocols have a similar operation, using a centralized

TDMA scheme, channel hopping and channel blacklisting. An aggregation of time slots forms a superframe that is periodically repeated over time. The protocol allows defining several superframes, to allow the transmission of messages with different periods. WirelessHART is a full-fledged industrial protocol, that addresses the specific requirements of industrial networks, which are quite different from the required for this thesis. Features like embedded security, multiple channels, multiple topologies and a transport layer that provides reliable connectionless transport services, are not useful for our context, turning this solution unnecessarily complex, expensive and energetically inefficient.

More recently, the working group IPv6 over the TSCH (6TiSCH) introduced a new protocol to the IEEE 802.15.4 protocol's family. The IEEE802.15.4e standard arose aiming at fulfilling three main needs, namely [49]: *i)* overcoming the limitations of previous IEEE802.15.4 MAC Layer, particularly regarding the energy consumption of the router nodes and the susceptibility to suffer from interferences; *ii)* support the requirements of industrial and automation applications; and *iii)* enable the integration with Internet Protocol Version 6 (IPv6) upper layer protocols. To reach a superlative behaviour comparing to previous versions of IEEE802.15.4 MAC layer, the IEEE802.15.4e standard supports its operation in a TDMA approach with the necessary scheduling. Depending on the metric to optimize, three different approaches are proposed. The Low Latency Deterministic Network MAC approach aims at minimizing the latency, the Timeslotted Channel Hopping provides a high reliability to node's communications, while Deterministic and Synchronous Multi-channel Extension is more concerned with QoS requirements. Even though, as ISA100 and WirelessHART, besides the potentiality of IEEE802.15.4e, its main purpose is industrial communications and hence the gains obtained with such complex solution are not worthy for solutions where reliability is not a main requirement.

Among the presented protocols, albeit some present features that are interesting for the context of this thesis, there is no single one that includes all the desired characteristics. Therefore, a combination of several concepts properly articulated to meet the defined requirements was the solution proposed. Particularly, using beacon messages for network synchronization (as BodyMAC and IEEE802.15.4e), using a CSMA approach for the initial setup (as BodyMAC and FlexiTP) and using implicit scheduling (as RI-EDF) is a promising approach. Moreover, a similar architecture to WirelessHART, ISA and IEEE802.15.4e with superframes (slotframes in the case of IEEE802.15.4e) divided into frames and time slots aligns with the defined requirements.

2.1.3 Network Layer protocols

Regarding the Network Layer, IP is the predominant choice, with two coexisting versions, IPv4 and IPv6. The latter one was born due to the exponential increase of the number of addressable devices and consequent exhaustion of available IP addresses. Nevertheless, and despite IPv6 popularity and interoperability, its direct use on IoT devices is not always reasonable. The overhead of such protocols (and inherent processing needs), usually clashes with the constraints associated to IoT devices. Thus, a big effort is being made by the IPv6 Low-Power Wireless Networks (6LoWPAN) working group from Internet Engineering Task Force (IETF), in order to minimize the IPv6 limitations and to make it suitable to be used by IoT devices. As a result of such work, the 6LoWPAN protocol was released [50], defining the specifications of the Network Layer of IP-based Wireless Personal Area Networks (WPAN) based on IEEE 802.15.4 physical and MAC layers. This protocol defines an intermediate layer between the MAC and Network Layers, the Adaptation Layer (AL), that aims at ensuring the interoperability between the referred layers, implementing compression, fragmentation and reassembly mechanisms.

Albeit the advances attained by this AL, the complexity introduced for supporting IPv6 still is too high for technologies that excel by their simplicity, low power consumption and low overhead. Some examples are the Bluetooth Low Energy (BLE) and Near Field Communication (NFC). Hence, the 6lo working group extended the 6LoWPAN protocol to these two technologies, establishing the IPv6-over-foo AL [51]. Pursuing similar goals, Farris et al. [51] proposed a similar solution for integrating Radio Frequency Identification (RFID) systems into IPv6 networks. Notwithstanding, and despite all the efforts, there still exists a significant overhead associated to the implementation of such ALs, which makes them inappropriate for several tightly constrained applications. In these cases, proprietary solutions designed to fulfil the requirements of particular applications still are a common approach.

Still within the Network Layer, the Routing Protocol for Low-Power and Lossy Networks (RPL) [52] is the main option considered when deploying routing on 6LoWPAN solutions. This protocol was specified by the IETF Routing Over Low Power and Lossy Networks (ROLL) Networks Working Group and it consists on a distance vector routing protocol that uses Destination-Oriented Directed Acyclic Graph (DODAG)s to define routes. The construction of the graph is made through an Objective Function (OF) that dynamically defines the routing metrics to be used taking into account network constraints. In addition to the support of different traffic flows, as point-to-point, point-to-multipoint and multipoint-to-point, RPL adapts itself to the network rate and accepts routing metrics as link quality or current battery status of devices, in exchange for a higher computational cost.

2.1.4 Transport Layer protocols

The Transport Layer, also known as Host-to-Host Transport Layer, is directly transposed from IP to the IoT domain. Here, the two most important protocols are the TCP [53] and the User Datagram Protocol (UDP) [54]. TCP is a connection oriented protocol that ensures a reliable data delivery with end-to-end error detection in exchange for a higher overhead and lower data transmission speed compared to UDP. Contrarily, UDP is a connectionless low-overhead protocol that does not ensure any kind of acknowledge nor session control support, privileging the data transmission speed over the data transmission quality. Both protocols are commonly used by different applications and its choice is closely depended with the application requirements.

2.1.5 Application Layer protocols

The Application Layer creates the final level of abstraction that allows the development of different kind of user applications. Also here, the strict computational and energy constraints of IoT devices, prompted the emergence of several lightweight application layer protocols, such as Constrained Application Protocol (COAP), Advanced Message Queuing Protocol (AMQP) and Message Queuing Telemetry Transport (MQTT). Hypertext Transfer Protocol (HTTP) is also used but its genesis is not the same as the previous protocols. However, its popularity and support, brought it also to IoT applications.

COAP [55] is based on a request/response architecture and runs over UDP. As it is based on HTTP methods, it is interoperable with HTTP. For security, it uses the Datagram Transport Layer Security, similar to Transport Layer Security (TLS) in TCP.

AMQP [56] is based on an asynchronous publish/subscribe architecture, runs over TCP and uses TLS/Secure Sockets Layer (SSL) to ensure security. It provides QoS guarantees, specifically: *at most once, at least once and exactly once.*

As AMQP, MQTT [57] also runs over TCP, it is also based on an asynchronous publish/subscribe architecture, it also uses TLS/SSL to ensure security and it also supports three types of QoS: *fire and* *forget, delivered at least once and delivered exactly once.*

HTTP is a popular web messaging protocol based on a request/response architecture [58]. It runs over TCP, uses TLS/SSL for security and by itself it does not define any QoS. Contrary to the aforementioned protocols, HTTP does not define the header and payload sizes, being dependent on the web server implementation.

The choice of the protocol to be used is a daunting task and depends greatly on the kind of application and devices used. Thus, normally, there is not just one possible correct choice [27].

2.1.6 Low-Power Wide-Area Network technologies

An emergent area on low power communications respects the LPWAN technologies. They operate in the license-free bands of the spectrum and can be positioned between the cellular networks and the short-range communications, taking the best of each one. They potentially cover communication ranges similar to a cellular cell, keeping a low power consumption as some of short-range communications technologies. The data rates are also similar to the short-range communication solutions and, despite being smaller than WLANs and cellular networks data rates, are adequate for many IoT solutions. Some examples are Sigfox, Long Range (LoRa), Narrowband-IoT (NB-IOT) and LTE-M.

Sigfox [59] is a centralized, cellular-like, low-throughput wireless communication system. Energy-wise, it is very efficient, but it provides a limited capacity (a few messages per day with up to 12 bytes each), becoming particularly popular for telemetry-like applications. It requires cellular coverage which is based on the deployment of gateways on existing public telecommunications base-stations. Its actual coverage still is relatively small, particularly in rural areas. It does not allow to perform localization and the bandwidth is insufficient for reporting the amount of data required to be transmitted in the scope of the solution proposed in this thesis.

LoRa [60] is a Wireless Wide Area Networks (WWAN) specification designed to allow long range communications for IoT-like applications. It was designed to minimize energy consumption and provides modest bit-rates (300 bps to 50 kbps per channel). It may be used as a private network, but it may also be used to deploy a shared service infrastructure. LoRa provides a reasonable bandwidth and is energy-efficient. It also permits RSSI-based localization, but its performance in this regard is poor, *e.g.*, a practical evaluation in a limited area (110x64m) and with an infrastructure based on 6 anchor nodes, impracticable for farming applications, generates errors of several meters [61]. LoRa also has a proprietary Time of Arrival (ToA)-based localization scheme, although it is closed source and impossible to adapt according to applications needs. Results on its accuracy are scarce, but indicate a poor performance, at least for wide ranges (*e.g.* 100m for stationary nodes using a public location service [62]). Moreover, LoRa prescribes a star topology and constrains the kind of synchronization and communication between nodes and the gateways (*e.g.* for battery-powered devices - class A, it is used an ALOHA-like [63] protocol, without synchronization, and with dedicated receiving slots that may not be needed). Finally, LoRa nodes were considered too expensive to be implemented in each of the animal sensors.

NB-IOT [64] and LTE-M [65], part of 3rd Generation Partnership Project (3GPP) LTE, have been developed for M2M and IoT applications. Despite providing relatively high bandwidth and potentially low consumption, these protocols depend on a public communication infrastructure, which is often unavailable in rural areas. Moreover, the need for paying a fee to the telecom operator by each sensor carrying the transceiver eventually may lead to unbearable exploration costs.

2.1.7 IoT Gateways

The diversity of IoT applications, devices and protocols, creates a natural lack of interoperability. Consequently, to allow the communication of devices that resort on different communication technologies or devices that use different application protocols, an intermediate device is required, typically known as an IoT gateway.

In Zhu *et al.* [66], an IoT gateway that enables the interoperability between Zigbee and General Packet Radio Service (GPRS) protocols is presented, being tailored for applications composed of light, temperature and humidity sensors. A similar work is proposed by Guoqiang *et al.* [67], with an extended number of supported devices and protocols. This Smart IoT gateway allows a multifunctional configuration and supports protocols as Zigbee, RFID and RS485.

Furthermore, considering the popularity of smartphones, a smartphone gateway is proposed in Aloi *et al.* [68]. It supports several protocols like Wi-Fi, ANT+, Bluetooth, NFC, ZigBee and 6LoWPAN. A similar approach is followed in Zachariah *et al.* [69], being presented a smartphone-centric solution on which the main links supported on the connection to sensors are based on BLE and IPv6.

Also in Datta *et al.* [70], a wireless IoT gateway is proposed, mainly supported on Application Programming Interface (API)s composed of Restful [71] web services. The gateway architecture, besides the Representational State Transfer (REST) API, includes dynamic device discovery in order to allow the insertion or removal of devices and a management connection module to handle devices that do not support REST.

Besides these academic research works, several standards that specify the deployment of IoT gateways for supporting M2M/IoT services have been developed. One of the most relevant ones is the OneM2M [72], that has been developed with the participation of important standard development organizations and consortiums (including, for instance, European Telecommunications Standards Institute (ETSI), Telecommunications Industry Association (TIA) and Broadband forum), and powerful companies (e.g. Cisco, Intel, Samsung, Ericsson). OneM2M provides a horizontal platform that supports secure, reliable and efficient [73] operations of multiple IoT/M2M services, particularly resorting on REST APIs. It supports multiple existing application protocols such as HTTP, COAP and MQTT, and communication technologies such as ZigBee, Bluetooth, WiFi and Cellular networks on the sensing domain.

Other turnkey solutions are available, being differentiated by the application protocols and communications technologies supported. Intel [74] provides a gateway that supports protocols such as MQTT, WiFi, Bluetooth, Cellular technologies and Zigbee, but also Serial and USB interfaces. SmartM2M [75] supports HTTP, COAP, Cellular technologies, Zigbee, Bluetooth and Wifi. The Lightweight M2M (LWM2M) [76] supports COAP and 6LoWPAN technologies.

In sum, we can find a wide range of IoT gateways, some specially designed for certain applications, while others try to offer an horizontal integration of a diversity of IoT/M2M protocols and communication technologies. In common, they share the goal of creating an integrated IoT platform that could support multiple applications, protocols and standards. However, none of them fulfils completely the requirements of the solution addressed in the scope of this thesis, mainly in what concerns the local support of application-level services, such as an improved localization mechanism and system management. Even if some of those solutions could be customized, the cost of such integration could be economically unaffordable. Furthermore, and despite the great advantages that a fully integrated gateway could offer, the resources that it takes can be superfluous considering the intelligent farming scenario, where the simplicity, low cost and high autonomy are the key features.

2.2 Localization techniques

Location monitoring plays an important role in a wide range of societal and technological areas. Disaster relief, environment control, tracking people and goods and animal monitoring are just a few examples. In fact, location monitoring, besides allowing to know the location of devices, also enables identifying the location of events of interest, extracting inherent relationships between events or implementing system features such as geographical routing [77].

The intense interest in location services led to the emergence of several solutions aiming at the fulfilment of the requirements of specific applications or domains, with distinctive metrics such as energy efficiency, accuracy, complexity and security [78]. Thus, a localization method is typically application-dependent [79], which means that there is not a single solution capable of handling the requirements of all location applications and domains. Even though, there is a common (or nearly common) structure followed by localization systems. During the following sections, we firstly present localization systems shared structure, before systematizing a established taxonomy and detailing popular techniques used.

2.2.1 Localization phases

A localization system is application-dependent. Hence, several localization systems have been proposed, particularly tuned to satisfy the requirements of a single or group of application scenarios. Even though, typically, three different phases can be identified in such a system [79], [80]:

- **Distance estimation:** also called *ranging*, during this phase the distances (or angles) between nodes are estimated. That information is used by subsequent phases to complete the localization process. As it will be clear in Section 2.2.2, this phase is an intrinsic characteristic of range-based localization systems, being the main techniques used described in Section 2.2.3;
- **Position computation:** the available information about an object location, either in form of distance, angle or connectivity information, may be merged through the use of mathematical and geometrical formulas to establish a more accurate location. The most popular techniques are described in Section 2.2.4;
- **Localization algorithm:** depending on the application scenario, additional mechanisms are typically added to previous phases to create a localization system capable of handling the requirements of a particular application. These include, *e.g.*, filtering capabilities and cooperative algorithms. Several works have been proposed, being presented in the state-of-the-art Chapter (Section 3.1.1).

2.2.2 Localization taxonomy

According to Srinivasan *et al.* [78], localization techniques can be organized into several layers taking into consideration their features (see Figure 2.3). The first, categorizes algorithms into direct approaches, where the absolute localization is acquired either through a manual configuration or a GPS device; and

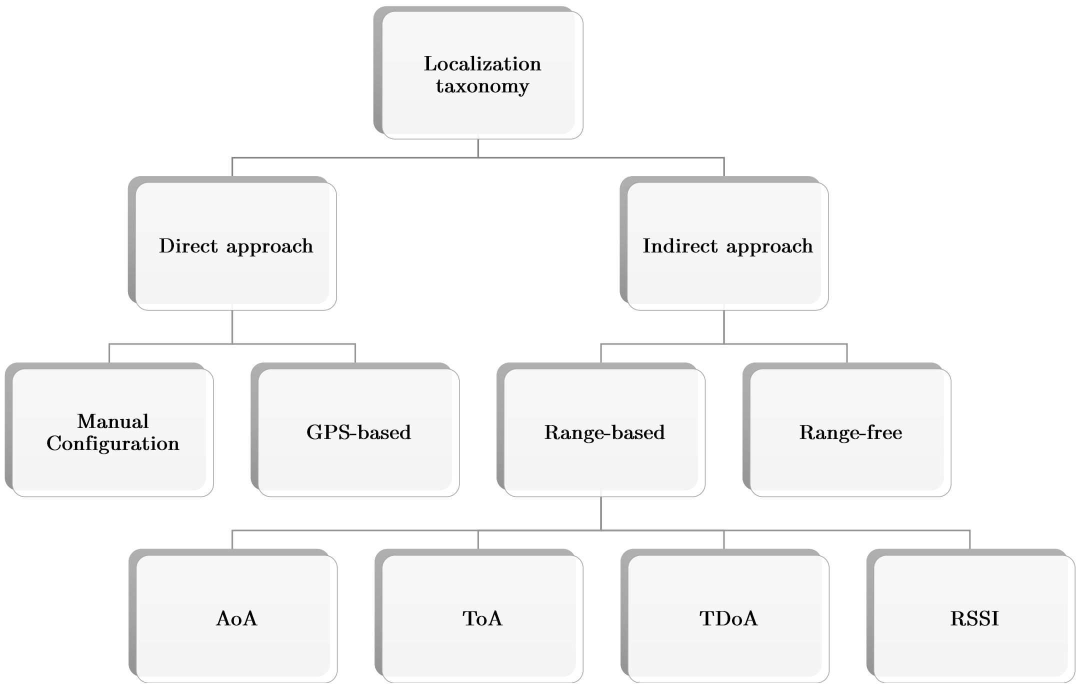


Figure 2.3: Localization techniques taxonomy.

indirect approaches or relative localization algorithms on which blind nodes (with unknown position) are localized relatively to beacon or anchor nodes (that know their position).

Within the indirect approaches group, localization algorithms can be classified into range-based and range-free [81]. Range-free methods do not employ any distance estimation technique, using exclusively connectivity information between nodes to map devices in a certain space. Some examples are the APIT, DV-Hop, Multi-Hop, Centroid and Gradient algorithms [82], very popular in WSNs. Despite being attractive for WSN since they do not require additional hardware and so enable low-cost, low-energy and small size solutions, their accuracy is very limited, hence not suitable for many practical applications [81], such as the one discussed in the scope of this thesis. On the other hand, range-based localization depends on a distance estimation process before employing location computation techniques. It comprises a few methods very popular and widely used in localization services, such as Angle of Arrival (AoA), ToA, Time Difference Of Arrival (TDoA) and RSSI.

2.2.3 Distance estimation/ranging techniques

Range-based algorithms always entail a distance estimation process, typically associated with communication signal features, for instance AoA [83], ToA [84], TDoA [80] or RSSI [85].

Angle of Arrival, as the name suggests, consists on the measurement of the angle settled in a communication between a blind node and a beacon node. More specifically, this method demands the measurement of the communication angle considering a reference direction, commonly known as *orientation* [83]. With angle information is then possible to implement geometric relationships in order to estimate the position of a blind node, which is called ***angulation*** (see Figure 2.4). Considering a two-dimensional plane, at least two beacons are necessary. However, using three beacons is the most popular strategy, known as ***triangulation*** (see section 2.2.4).

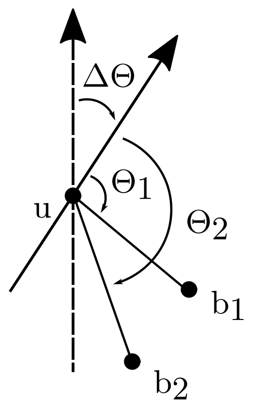
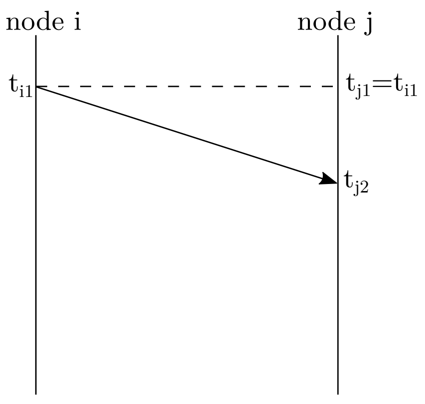


Figure 2.4: AoA technique (based on [83]).

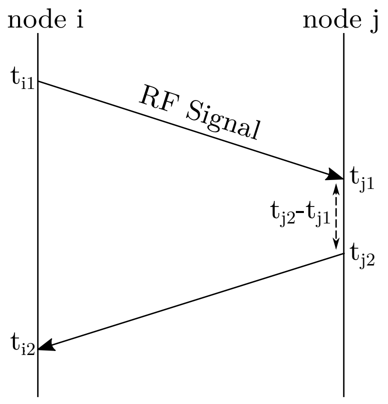
Even if such approach does not requite any type of synchronization mechanism as it happens with ToA or TDoA techniques, AoA presents practical limitations. It is vulnerable to multipath loss, reflection and non-line-of-sigh issues, the accuracy decreases considerable for long distances and, more importantly, it requires costly and complex hardware such as directional antennas or antenna arrays [86].

When the signal-propagation time is used during the distance estimation phase, we are in the presence of a **Time of Arrival** technique, also known as *time of flight*, that, for sake of curiosity, is the technique employed by GPS technology. It resorts on the velocity equation $v = \Delta r/\Delta t$ to estimate the distance r by measuring the time t that a wave with a known velocity v takes to propagate from one point to another. Two different implementation approaches coexist, namely one-way ToA and two-way ToA. The former (Figure 2.5a), only considers the time that a signal takes to travel from a node i to a node j, while the latter considers the propagation time from a node i to a node j, plus the vice and versa, *i.e.* from node j to node i. In the one-way ToA, the distance is given by $r_{ij} = v \times (t_j - t_i)$ but, besides being necessary a timely synchronization between nodes to enable an accurate estimation, node i needs to send the timestamp of its communication to node j. In the two-way ToA (Figure 2.5b), node i transmits to node j at instant t_{i1}, node j receives and processes the message at instant t_{j1} and t_{j2}, respectively, and returns a communication to node j that is received at instant t_{i2}. Thus, the distance is given by $r_{ij} = v \times ((t_{i2} - t_{i1}) - (t_{j2} - t_{j1}))/2$. This method exempts any type of synchronization since each node is responsible for its own calculations, but in contrast, requires a higher overhead since it requires more communication exchanges. Additionally, there is an additional disadvantage shared by both one-way and two-way ToA. Radio frequency wave signal propagation velocity is very high, near to the speed of light. This means that having accurate clocks may not be enough since errors in the order of 10 nano seconds result in errors of 3 meters [81]. To minimize this issue, the solution may be the use of slower signals or the use of TDoA techniques.

Time Difference of Arrival is a distance estimation technique that uses on the different propagation times of different signal waves to calculate the distance between two nodes. There are two ways of doing it. Having multiple beacon nodes transmitting simultaneously (or after a pre-determined time) or transmitting multiple signals from the same node i but with very distinctive propagation speeds (see Figure 2.6). In the former case, the difference time of arrival is taken for each pair transmitter (beacon) - receiver, defining hyperbolas whose interception identifies the location estimation. In the latter, transmitting at the same time signals with very different speeds results on the same signals being received at distinctive times at the receiver j. Hence, a rough estimation is obtained by assuming that



(a) One-way ToA (based on [81]).



(b) Two-way ToA (based on [81]).

Figure 2.5: Time of Arrival operation alternatives.

the arrival time of the faster signal is approximately the same as the departure one. Thus the distance is given by $r_{ij} \approx u \times (t_{j2} - t_{j1})$, being u the propagation velocity of the slower signal. Typically, the slower signals used are ultrasonic signals or audio waves. Thus, although synchronization may be dismissed if using a multiple signal TDoA approach, it requires additional hardware that makes it costly and complex. Moreover, resorting on ultrasonic sensors is fallacious since it brings a relevant constraint, the range.

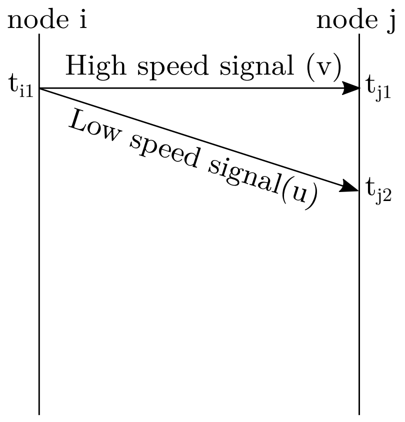


Figure 2.6: TDoA operation (based on [81]).

So far, we presented solutions for distance ranging that require additional, complex and costly hardware, whose use is not viable for applications where cost, size and energy expenditure constraints are fundamental. Therefore, RSSI-based solutions came up as being potentially very attractive since RSSI is measured by almost all commercial Commercial Off-The-Shelf (COTS) radio transceivers [87]. As, theoretically, radio signal power attenuates with the square of the distance, measuring the signal power at the receiver can be used to estimate the distance travelled by the signal. However, in practice, signal propagation is affected by additional internal and external interferences, so three different mathematical models have been proposed [85] to model such interferences:

- **Free Space Model:** assumes perfect propagation conditions, with no obstacles between the transmitter and receiver and it may be used when the transmission distance is higher than the size of the antenna and signal's wavelength. In such cases, the model is given by Equation 2.1:

$$P_r(d) = \frac{P_t G_t G_r \lambda^2}{(4\pi)^2 d^2 L} \tag{2.1}$$

where P_r and P_t are the signal power received in the receiver and transmitter respectively, G_r and G_t are the antennas gains of receiver and transmitter, respectively, λ is the signal wavelength, L is a loss factor and d the distance both transmitter and receiver. In practice, besides the toilsome and erratic measuring processes of both G_r, G_t and L parameters, the propagation assumptions are rarely met in real application scenarios, thus not being suitable for the majority of applications.

- **Two-ray Ground Model:** this model is mainly used in urban scenarios and it is applicable when the distance d between the transmitter and receiver is a few kilometers and the height of the antenna is high (>=50 m). Equation 2.2 describes the referred model, where h_t and h_r represent the highs of transmitter and receiver antennas. Despite being appropriate for urban areas, particularly for cellular networks, such model is not suitable for WSN applications.

$$P_r = P_t G_t G_r \frac{h_t^2 h_r^2}{d^4} \tag{2.2}$$

- **Log-Normal Shadow Model (LNSM):** also known as Log Distance Path Loss, is applicable to more generic applications, either indoor or outdoor since it permits the calibration of its parameters according to the application environment. Equation 2.3 describes such model:

$$P_L = \overline{P_L}(d) + X_\sigma = \overline{P_L}(d_0) + 10\eta \log\left(\frac{d}{d_0}\right) + X_\sigma \tag{2.3}$$

where $\overline{P_L}(d_0)$ é the signal power at a distance d_0, η represents the propagation loss for a particular environment and X_σ is a zero-mean Gaussian variable also related with a particular propagation environment. This means that by empirically calculating environmental-dependent parameters ($\overline{P_L}(d_0)$, d_0, η and X_σ)), it is possible to draw a model adequate for a specific environment.

No matter the model used, no additional hardware is required, making RSSI-based localization approaches very attractive. However, the accuracy is very limited, particularly due to the following phenomena [77]:

- **Fading:** the signal power decreases with the distance, depending on several environmental factors, the so called path loss. Even though the presented models try to minimize such dependency, there are still variations caused by different factors such as constructive/destructive interference, temperature, humidity, antenna orientation and quality, just to cite a few, that are difficult to predict and include in the model. Such variations are named *fading* and possible solutions to minimize them are calibrating the model [86], [88]–[90] and perform measurements in different frequencies [77], [90];
- **Shadowing:** when a signal faces natural or artificial barriers such as people, animals, trees, bushes, walls or furniture, a significant, unforeseeable and frequency-dependent power loss is induced in the signal. As those barriers are unpredictable, dealing with this phenomena is a daunting task. A simple way of minimizing its effects is performing multiple measurements and take, for instance, the measurement's average [77], [88];

- **Hardware asymmetry:** even if belonging to the same model and production batch, there are important inequalities among radio transceivers with a relevant impact in signal's power [91]. Therefore, also for this phenomena, a calibration process is recommended to minimize such effects.

In a nutshell, despite the superior accuracy of AoA, ToA and TDoA comparing to RSSI-based localization techniques, they require additional hardware and in some cases precisely synchronized clocks, making the solution complex and expensive. In contrast, the ubiquitous availability of RSSI measurements in radio transceivers, make RSSI-based methods very attractive, since no additional hardware is required, hence not incurring in additional costs. Notwithstanding, using RSSI requires modelling the signal propagation to cope with several issues, such as path loss, fading and shadowing, whose effects need to be minimized with additional processing techniques such as calibration, multiple measurements, high quality antennas and frequency hoping.

2.2.4 Location Computation

No matter the method used to perform distance estimation, further processing is needed to gauge a more accurate location. This is so since the distance estimation process only allows the detection of potential areas of location and not specific points. Hence, the position or location computation phase typically resorts to geometric information to estimate the location of a node [81]. Some examples are the use of a simple **centroid** technique, **angulation** when there are angle information (triangulation if there is information from three different sources) or **lateration** if the information is distance-based (trilateration when there is data from three different sources). In addition to the referred geometrical approaches, there are also some probabilistic approaches that estimate the position of a node according to a set of probabilities, and some empiric approaches such as fingerprinting technique, where a map is built *a priori* with some known parameters.

2.2.4.1 Lateration, trilateration and multilateration

Lateration is the most popular method for location computing. The basis of its operation is settled in the intersection of circles whose radius is given by a distance estimation process. For computing a single location point in a two dimensional space, at least three circles are required, thus called **trilateration** (*i.e.*, distance information from three pairs beacon-blind node are required). When more than three beacons are used, we are in the presence of a **multilateration** technique. For the sake of simplicity, we continue the description of these techniques considering the trilateration scenario, that is also the most popular.

In a ideal scenario, the three circles intersect in a single point, as depicted in Figure 2.7. In such case, the location of an unknown node (u) can be derived from the system of equations that describe the three circles as described in Equation 2.4:

$$\begin{aligned} d_1^2 &= (x - x_1)^2 + (y - y_1)^2 \\ d_2^2 &= (x - x_2)^2 + (y - y_2)^2 \\ d_3^2 &= (x - x_3)^2 + (y - y_3)^2 \end{aligned} \tag{2.4}$$

where d_i is the distance between beacon b_i and unknown node u and x_i, y_i are the coordinates of beacon nodes. As there is a single intersection point, the three equations can be solved to determine the coordinates (x, y) of the intersection point.

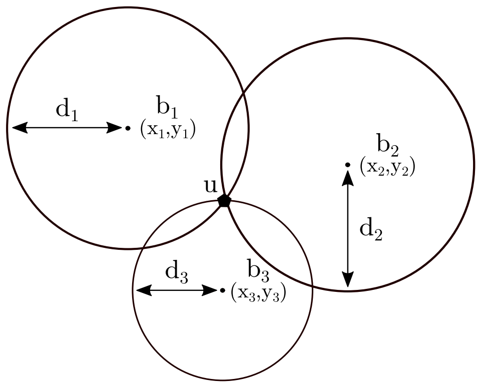


Figure 2.7: Trilateration technique: ideal operation. The three circumferences have a single interception point. *b* stands for beacon and *u* stands for unknown node (based on [92]).

Nevertheless, in practice, and mainly due to the inaccuracy of distance estimation processes, there is not a single interception point, but an overlapping area where the unknown node may be positioned. To overcome this issue, and thus enable the estimation of a position for the blind node within that potential area, two different approaches have been proposed [92], the shortest distance algorithm and the line intersection algorithm.

The shortest distance algorithm representation is illustrated in Figure 2.8. Its rationale is based on the calculation of the three nearest intersection points between the circles built after the distance estimation process. In practice, those points all calculated as follows (the mathematical proofs are out of scope of this thesis, being described in [92]):

1. Selecting two beacons and estimating the distance between each beacon b_i and the blind node u;
2. Drawing the circles that represent all possible positions for node u in relation to the considered beacons, *i.e.*, assuming that we started on beacons b_1 and b_2, and the estimated distances are d_1 and d_2, draw two circles with radius d_1 and d_2;
3. Determining the intersection points (*i.e.* their coordinates) between those two circles (for b_1 and b_2, the intersection points are identified as I and I' in Figure 2.8);
4. From the intersection points obtained, choosing the closest to the beacon not selected in stage 1 (point I in Figure 2.8);
5. Repeating steps 1-4 for the remaining combinations of pairs of beacons;
6. After having the three closest points, assessing their average in order to have an estimation of node u position.

The issues of the shortest distance algorithm are two-fold. Firstly, circles need to intersect, which not always happens due to errors in the distance estimation process. Secondly, the process of determining the nearest three points is not straightforward and it may incur in errors.

Alternatively to the shortest distance algorithm, the same authors proposed the line intersection algorithm (Figure 2.9). Besides resorting on selecting the three nearest intersection points, it uses lines intersections that are built using the intersection points of pairs of circles. More precisely, the process comprises:

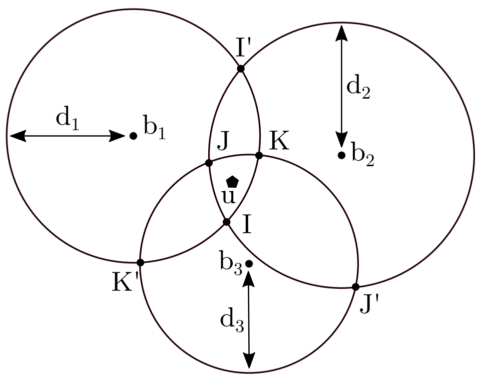


Figure 2.8: Trilateration technique: the shortest distance algorithm (based on [92]).

1. Determining the intersection points (*i.e.* their coordinates) between two circles (for b_1 and b_2, the intersection points are identified as I and I' in Figure 2.9);
2. Determining the line's equation given the two intersection points (line $\overline{I}$ in Figure 2.9);
3. Repeating steps 1-4 for the remaining combinations of pairs of beacons;
4. Determining the interception between the lines obtained from previous steps;

Ideally, the three lines have a single intersection point. However, and again due to distance estimation errors, circles may not intersect, precluding the determination of line's equation and hence making impracticable the use of such algorithms. Henceforth, both algorithms are sensitive to distance estimation errors.

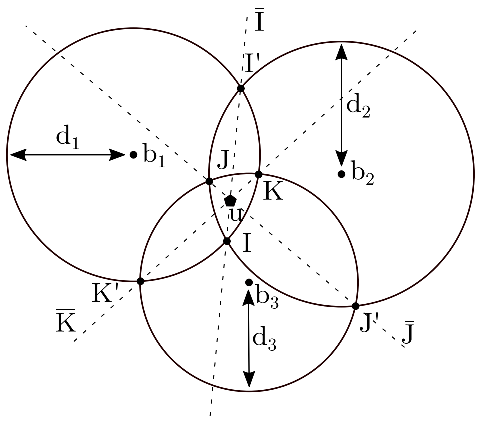


Figure 2.9: Trilateration technique: line intersection algorithm (based on [92]).

2.2.4.2 Angulation and triangulation

Angulation or **triangulation** uses angles information to estimate the location of a blind node. Depending on where the position is computed, namely in the beacons or in the blind node, information from two or three beacons is required [80].

When the computation of the location occurs remotely, *i.e.* within the beacon context, information from two beacons is enough, being applied trigonometric relationships to estimate the location of the blind node. Each beacon measures the angle of the communication link with the blind node, and, using simple trigonometric equations, establishes a line between the beacon and the blind node. Then, the intersection of two lines gives the blind node's position (see leftmost picture of Figure 2.10).

On the other hand, when the location is computed in the blind node side (ee rightmost picture of Figure 2.10), the problem becomes very similar to the trilateration one. In fact, using angle information and knowing beacon's positions, trigonometric laws can be used to convert angle in distance and transform the triangulation problem into a trilateration problem, already described.

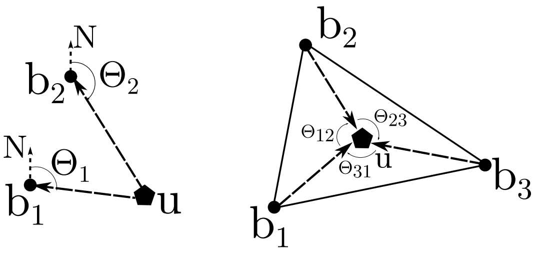


Figure 2.10: Angulation or triangulation technique (based on [80]).

2.2.4.3 Fingerprinting

Wireless **Fingerprinting**, in short, comprises the construction of a database that associates radio frequency measurements, for instance RSSI values, to each possible position in a certain area. This method includes two distinctive phases, namely [93]:

- **Offline training:** during this phase, radio frequency measurements are collected for each possible position of a node. A representative map is built *a priori*, composing a database or radiomap that supports the online localization phase;
- **Online localization:** during the online localization phase, measurements gathered are compared with the stored ones, being chosen the position whose RSSI measurement stored during the offline training better approaches the online measurement. As different methods can be used to select the most suitable location position, several variants of fingerprinting exist. Typical approaches are the use of distance measurements as the Euclidean, Squared Euclidean and Manhattan distances [94], and the k-Nearest Neighbour (kNN) algorithm [93].

The potential of fingerprinting mainly arises from its simplicity and its easier and low-processing online deployment. Furthermore, it avoids modelling propagation issues since a calibration phase is required in each implementation scenario. In fact, the calibration process has both advantages and disadvantages. If, on the one hand, it allows a smooth adaptation to the conditions of the application scenario (hardware, environmental conditions, static obstacles, etc.), thus contributing to an improved localization accuracy, on the other hand, it is a laborious process, impracticable in big area scenarios.

The referred techniques are the cornerstone of common localization approaches. However, complementary algorithms have been investigated to minimize some limitations or drawbacks of both distance and localization estimations. For instance, when using RSSI, several approaches to enhance the modelling function have been proposed to reduce the effects of phenomena such as path-loss, fading and shadowing; or the implementation of filtering techniques as the Kalman Filter and Particle Filter to improve the localization estimation process. These approaches are, as stated, very application-dependent, and thus are explored in more detail in Chapter 3, considering scenarios identical to the one presented in this thesis.

2.3 Machine Learning

ML is nowadays a very trendy topic in the field of Computer Science. In a nutshell, its main purpose is the autonomous retrieval of computational models through the analysis of data. However, when dealing with the ML concept, we are also typically faced with a few additional concepts, such as Knowledge Discovery from Databases (KDD), Data Mining (DM) and Artificial Intelligence (AI), all in the scope of the Data Science field. Therefore, before discussing the most important concepts about ML, its applications and typical strategies for its application, it becomes important to clear up what distinguishes them, how they are related and which are relevant in the scope of this thesis.

The incessant growth in number and power of electronic devices, as well as the massive amount of data generated by those devices, led to a ineluctable need of developing strategies, mechanisms, resources and frameworks to deal with the so called Big Data. Dealing with Big Data requires not only proficient infrastructures to store large amounts of data, but also suitable and powerful instruments to analyze and retrieve relevant information from the human/business perspective. The study and development of these instruments constitutes the Data Science world, that includes a few different but interconnected fields. As illustrated in Figure 2.11, DM and KDD are related with all the remaining fields, making them central within the Data Science scope. However, while KDD encompasses all complex processes for extracting useful knowledge from data, either structured, semi-structured or unstructured, DM is recognized as being a single step within the KDD process [95]. Particularly, applying DM consists in extracting information from previously pre-processed data that can be available on different kinds of databases. Extracting information is mostly related with the retrieval of a set of valuable patterns from data that, in a first instance, are unforeseen from a mere human analysis.

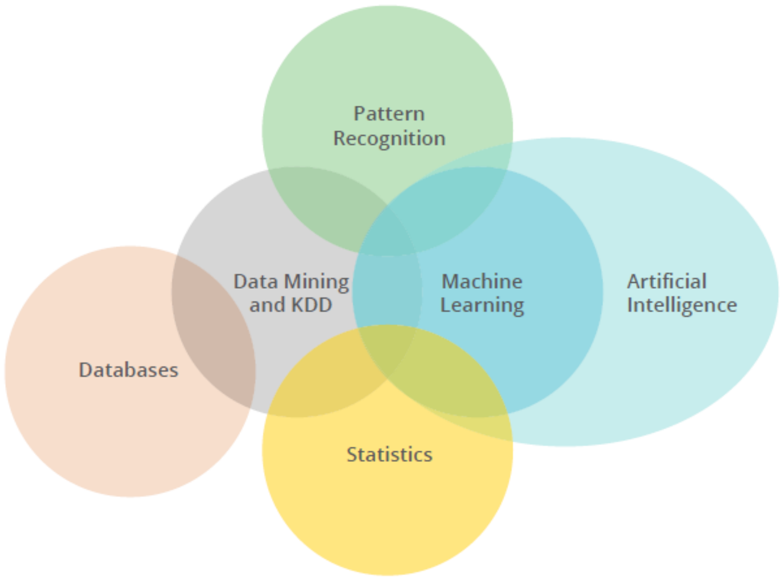


Figure 2.11: Data science fields [96].

On the other hand, the main purpose of the AI field is to endow computers with intelligence similar to the human one. For that, pattern recognition and ML are commonly used. Thus, ML can be contextualized as a tool for both DM and AI fields. Its focus is on extracting information from dynamic systems, creating models that are automatically updated based on past experiences. This adaptability allows machines to be continuously learning when faced with new data, playing a key role, not only in the creation of AI systems, but also in the resolution of KDD processes.

From all the fields depicted in Figure 2.11, there is only one missing reference, namely the statistics one. In fact, there is a strong connection between ML and statistics, since many of ML techniques are based on statistics and optimization methods to find relationships on data. Hence, the techniques and methods available are well-known and established, being discussed in Section 2.3.3. Notwithstanding, the area of ML is not immutable. The application of ML models in different contexts and for different application scenarios is a very active research topic, with a wide range of real applications, for instance: speech recognition; prediction of the rate of cure of patients with different diseases; detection of fraudulent use of credit cards; autonomous vehicles; computers players of chess; or the diagnosis of cancer through the analysis of gene expression, are just some examples that show the wide range of ML applications [97].

2.3.1 Machine Learning taxonomy and workflow

Formally speaking, in any learning task, the aim is to find a model for a function $\hat{f}$ from a set of observations $\{\overrightarrow{x}, y\}$ such that $y = f(\overrightarrow{x})$. Here, $\overrightarrow{x}$ are the independent variables or attributes[1], while y is the dependent variable [98]. This modelling function, obtained in a process commonly denoted as *training*, can be used to predict future events $\overrightarrow{y}$ by applying the function $\hat{f}$ to new registered observations $\overrightarrow{x}$. The nature of the dependent variables together with the type of learning used to model the function $\hat{f}$, leveraged to the definition of a well established taxonomy between ML methods, as depicted in Figure 2.12.

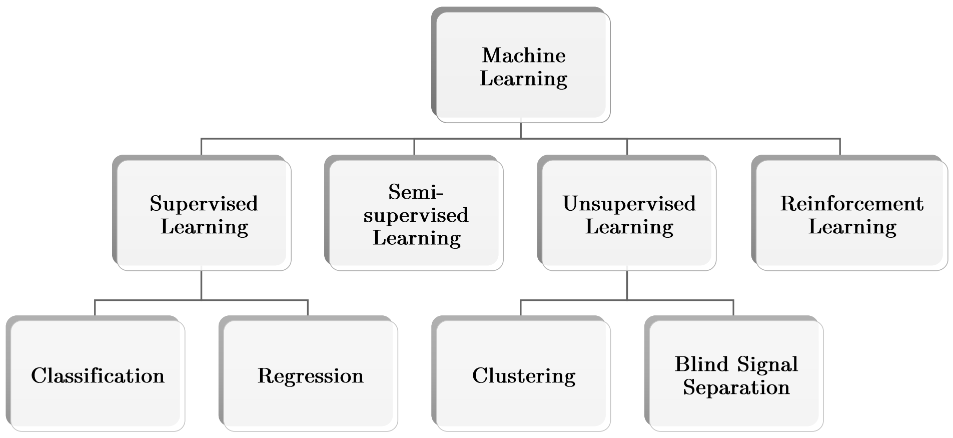


Figure 2.12: Machine Learning taxonomy (based on [99]).

If during the training procedure, the dependent variable is labelled, ***supervised*** or ***predictive learning*** algorithms are used. This means that, during training procedures, values of y are known

[1]In the scope of this thesis we use attributes, features and variables as synonyms in the context of ML applications.

and are used to find the best modelling function $\hat{f}$. In this case, whenever y assumes a value from a finite group of values, the problem is classed *classification.* In contrast, whenever y is continuous, a *regression* problem is encountered. Currently, there is an uncountable number of applications that take advantage of supervised learning. Marketing and advertising companies are continuously extracting information from clients in order to understand and predict the lifetime value of products, analyse the sentiment churn of users and provide recommendations. On the other hand, these methods are also being used by enterprises to optimize resources allocation, detect spikes of demands or predict supplementary needs. In security, they can be used by spam filters, detect malicious emails and links, and also detect frauds. Weather forecast also takes advantage of this kind of learning. In sum, with the increasing availability of data in all sectors of our society, supervised machine learning already plays an important role for helping handling data and extracting valuable information for businesses or daily life tasks.

If the dependent variable y is not defined and labelled, an ***unsupervised*** or ***descriptive learning*** task is encountered. The main goal is to find hidden patterns or data structures that could aid in the representation of the data. This type of learning is typically suited for applications that deal with a great amount of unlabelled and unstructured data such as the produced in social media applications. They mainly focus in applying clustering [100] or blind signal separation [101] algorithms to allow the segmentation of data accordingly to similarities and differences. Some examples are recommendation systems, customer segmentation and big data analysis.

Semi-supervised learning, as the name suggests, combines features from both supervised and unsupervised tasks. In fact, in a semi-supervised learning procedure both labelled and unlabelled data can be found. The labelled data is used to support the model in the identification of the dataset labels. Then, when learning from the unlabelled data, the model can define the boundaries for each one of the labels, and even find additional groups not previously labelled. This type of learning is specially interesting since, on the one hand, labelling manually large amounts of data is toilsome and sometimes unfeasible; and on the other hand, the process of labelling can introduce erratic bias that will be considered by the models. Speech recognition or genetic sequencing are some examples where this strategy is being used.

The last type of learning presented is ***reinforcement learning***. In contrast to the remaining types of learning, these models do not learn with an example dataset, but follow instead an approach of trial and error. It corresponds, thus, to a continuous learning task, since the model uses the current inputs to define the output and, according to the success of that output, the model is reinforced or not. This type of learning is being actively used in several AI applications such as in robotics and self driving cars.

Independently of the ML application and the type of learning used, a general workflow can be defined [96]. This workflow can be seen as a guide for the creation of an iterative methodology for applying ML. This workflow is illustrated in Figure 2.13 on which several steps are highlighted, namely:

1. **Problem discern:** sometimes neglected, this prime step is fundamental for defining the subsequent methods within each one of the ML workflow steps. As an example, it is clear that for defining a suitable strategy for collecting data it is necessary to understand the problem to be solved, which data is theoretically necessary, how it can be gathered, just to cite a few.

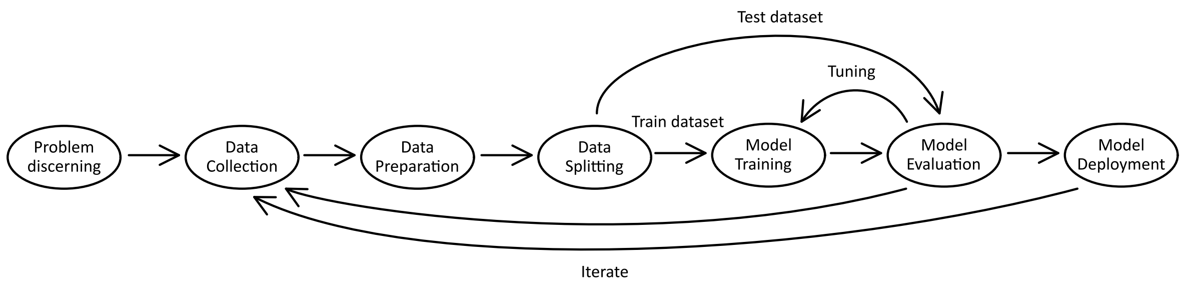


Figure 2.13: ML general workflow (based on [96]).

Frequently, it is necessary to return to this step after the last step of the workflow, commonly because the results are not the expected ones or, at least, do not give answer to the problem;

2. **Data collection:** this step concerns all the procedures related to the data gathering that will feed the model. It may include collecting data directly from proprietary or third party devices/platforms;

3. **Data preparation:** the raw data collected (for example from sensors) is sometimes disorganized and may contain erratic information, such as empty values, missing values or outliers. Consequently, if not properly prepared and cleaned, this data may contribute to the definition of an inaccurate or erratic model. Additionally, the format of the data gathered may preclude its direct use by ML algorithms or may include an excessive number of features, some of them redundant and that require a suitable treatment. The former issue is tackled by feature transformation techniques, while the latter is handled with feature selection techniques. Summing up, this step may include several sophisticated procedures in data cleaning, feature transformation and feature selection, which make it a very demanding and time-consuming process. Due to its relevance, several tools commonly used to handle these issues are detailed in Section 2.3.2;

4. **Data splitting:** this step is particularly related with supervised learning models. When a labelled dataset is available, in order to allow an evaluation of the training model, the dataset is commonly split into two types of datasets, one for training (the training dataset) and one for testing (the testing dataset)[2]. The most common approach is to have an unique split following a pre-defined ratio (90%-10%, 80%-20% and 70%-30% are the most popular), being the training dataset always the one with the greater part. However, in approaches where cross-validation is used, the dataset is divided in several parts (called folds). Then, in an iterative approach, each fold is used as a testing dataset (one per iteration), being the remaining ones used as the training dataset. This means that, for example, when using a 5-fold crossing validation, the training procedure would be composed of 5 iterations, being used one different testing fold per iteration. This process is commonly used to avoid bias in certain parts of the dataset, allowing to test the capability of the model to be generalized to other datasets;

5. **Model training:** during this step, ML learning algorithms are applied. It corresponds to the effective learning phase and, depending on the chosen algorithm and size of the dataset, this phase can last for minutes, hours or even days. The repertory of available algorithms is vast. Thusly, the most relevant for the scope of this thesis are presented and discussed in Section 2.3.3;

[2]In fact, some approaches also split the test dataset into two. One that is used for tuning the model and another that is used exclusively to evaluate the model.

6. **Model evaluation:** evaluating the model is the phase of the process that allows to assess the performance of learning models. When dealing with supervised learning, this process corresponds to apply the trained model to the test dataset and compare the results with the original test dataset. As several algorithms dispose of several configuration parameters, there is typically a tuning iteration. This tuning aims to find the best parameters in order to optimize the performance metrics evaluated. The most popular performance metrics in ML are detailed in Section 2.3.5;

7. **Model deployment**: last but not the least, the model can be deployed in real-use case scenarios. Although being the last step in the workflow, it does not correspond to the end of the process. In fact, any ML process corresponds to a continuous process that need to be continued enhanced, either through the collection of new and richer data or due to new needs of the application. Additionally, models with excellent performance metrics can have unpredictable results in some real-use cases, being thus crucial to evaluate how the model behaves in real-use scenarios and enhance it if necessary.

Having a clear idea of how a ML application workflow is, a more detailed study about several of the stated steps is presented in the following sections. Particularly, the focus is on data preparation, model training and model evaluation steps.

2.3.2 Data Preparation

The raw data collected is rarely ready to be directly used by ML models. For instance it may contain empty fields, outliers, redundant information, features that cannot be used directly by ML algorithms, noisy features or even irrelevant features. The data preparation phase handles these issues by means of three main procedures, namely **data cleansing**, **feature transformation** and **feature selection** (see Figure 2.14). Albeit the most convenient techniques to be used in each one the procedures depend on the data and application, a review of some of the most popular techniques particularly applied to sensor data are detailed hereinafter.

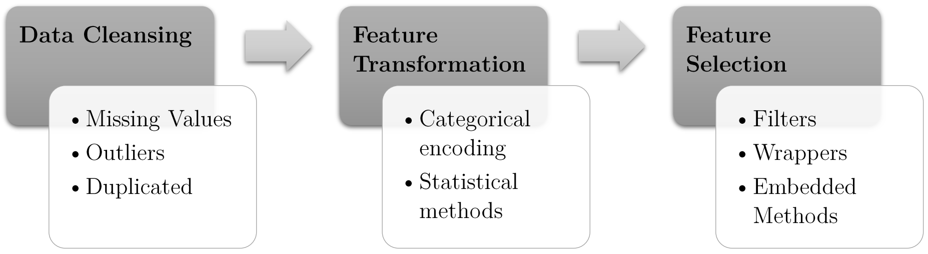


Figure 2.14: Data preparation steps in a ML process.

2.3.2.1 Data Cleansing

Data cleansing or data pre-processing, is generally concerned with erratic data that can be found, generally, in all kinds of datasets. However, as the types and sources of data differ greatly on the application, also the solutions to tackle them do. Among the vast application scenarios, one where it is particularly important to apply data cleansing, are applications where the main source of data are sensors, because sensors may provide a lot of erratic data, either due to noise, malfunctioning operation or external interferences.

A typical issue is the existence of missing or unexpected values that cannot be handled directly by models (*e.g. NAN* - Not A Number, strange symbols). The easiest solution to deal with this issue is to remove such observations[3]. However, as this approach can lead to a lost of information, other strategies can be followed. One of them is to replace them by some statistical value associated with that feature, for instance the mean, the median or the mode [102].

Another typical issue is the existence of outliers, *i.e.*, values that are significantly outside the range registered by the majority of the remaining observations. These values can be justified by noisy source of data, bad measurements or faulty registrations. One possible way to deal with them, is to treat them as missing data. In that context, observations with outliers can be removed or the values replaced with statistic values. Nevertheless, if those outliers are likely to contain valuable information, the best approach is to maintain them unchangeable and use models capable of handling with such issue. An interesting study on outlier detection and handling is presented by Hodge *et al.* [103].

2.3.2.2 Feature Transformation

Commonly, the original features that compose the dataset are not in the appropriate format to allow models to retrieve from them the best benefit. Consequently, after a pre-processing phase, it is common to find a **feature transformation** phase. This phase consists on the construction of a set of new features derived from the original ones. Three big goals are intended: reduce data dimensionality, improve the model performance and speed up the learning phase [104]. One popular approach is to encode variables into a categorical representation. Frequently, some variables are composed of values that cannot be used directly by ML algorithms (*e.g.* categories represented in text), which demands its translation to a numerical representation.

Numerical and statistical manipulation of features is also a common approach when constructing new features. This approach may include the application of a set of simple mathematical operators (*e.g. +, -, **) or the enforcement of more advanced numerical and statistical operations (e.g. minimum, maximum, mean) [104] on the original features. Also, several original features may be combined through the use of different numerical operations to create more advanced features (*e.g.* the Movement Variation (MV) is extracted through the dynamic acceleration components measured by an accelerometer). However, as it happens in the data cleansing phase, the methods used in the feature transformation phase depend greatly on application scenarios. Consequently, the focus is the study of numerical and statistical transformations that are relevant for the scope of this thesis, namely transformations that can be applied to data gathered by inertial sensors such as accelerometers. We focused on the work of Zhang et al. [105], where different types of feature transformation techniques were explored in the context of human activity recognition through the use of inertial sensors. This work is particularly relevant since it was the basis of many of feature transformation techniques used by state-of-the-art works regarding animal monitoring presented in Section 3.1.2.

Table 2.1 summarizes feature transformation methods associated to numerical and statistical manipulation. With their denomination, the notation employed during this document and a brief description are presented. A more detailed description is presented in Appendix A. Mostly of them use the Observation Window (OW) concept that needs to be explained. When big sets of data are collected in a form of streams, it is common to slice those streams into fixed time intervals. Each one of these intervals, is called OW and each OW may contain one or more values. Applying feature

[3]We denote an observation, entry or row to all the information associated to a single event. A dataset is composed of multiple observations which in turn can be composed by several features.

transformation methods to these OW, enables the reduction of the data dimensionality since it is possible to summarize several measurements into a single one, providing benefits in terms of complexity and speed when employing ML algorithms.

Table 2.1: Summary of feature transformation methods: denomination, notation and description.

Feature	**Notation**	**Description**
Minimum	$min(A)$	Minimum value within the observation
Maximum	$max(A)$	Maximum value within the observation
Mean	$\overline{A}$	Arithmetic mean or average value of the observation
Variance	s^2	Sample variance
Standard Deviation	s	Square root of the sample variance
Root Mean Square	A_{RMS}	Quadratic mean - square root of the mean square
Skewness	G_1	Asymmetry of the values distribution
Kurtosis	b_2	Identifies if values distribution is heavy-tailed or not, *i.e.*, if contains a lot of outliers or not
Interquartile Range	IQR	Statistical dispersion, particularly between the 75th and 25th percentiles
Zero Crossing Rate	ZCR	The number of times that the signal changes from negative to positive within the observation
Mean Crossing Rate	MCR	Number of crossings through zero after subtracting each observation value by the observation mean
Energy	E	Squared root of the signal
Signal Magnitude Area	SMA	Magnitude of the signal
Signal Magnitude Vector	SMV	Similar to the SMA, but considering the magnitude of the resultant vector
Movement Variation	MV	Variation of all components of a signal within a observation
Spectral Entropy	SE	Spectral power distribution
Pairwise Correlation	$COR(A, B)$	Correlation between two signals
Dominant Frequency	DF	Frequency with more power after applying Fourier Transformation

These feature transformation techniques, although being statistical and numerical calculations, need to be chosen and implemented manually, accordingly to the data gathered. Thus, it demands a high human effort which is not always desirable. To overcome such issue, some automated strategies have been developed, enabling an automatic construction of new features. An overview of some of those techniques is presented in [104]. Although being a promising approach, it was not considered in the scope of this thesis because they do not allow an easy control over such processes.

2.3.2.3 Feature Selection

Building a wide set of features does not mean better results, quite the contrary. Commonly, a set of features contains redundant features (and thus worthless duplicated information), irrelevant features (that do not contain relevant information for the definition of the model) or noisy features (that may introduce erratic information to the model). Therefore, **feature selection** techniques are very important within the ML workflow.

Feature selection methods can be grouped into three groups, namely, filters, wrappers and embedded methods [104]. **Filters** are applied before training the model and are independent of the model selected. Consequently, they provide a generic evaluation of feature importance, not being biased to any specific model. This aspect can be either an advantage or disadvantage. On the one hand, being generic means that, theoretically, they would behave well no matter the model selected. On the other hand, as the features are not optimized for a specific model, the results of the modelling process may not be optimal. Techniques that rank features are examples of this type of approach.

Wrappers, contrary to filters, resort on learning techniques that seek for the best features within the available dataset. This approach iteratively models a different subset of features, being the models obtained evaluated subsequently. In the end, a rank is built considering the results obtained. This method allows to tune the feature selection procedure in line with the model, although incur in additional computational expenditure since several combinations of features are tested to find the best results.

Finally, **embedded methods** are methods that are embedded in the models. This means that some models have their own feature selection methods, being thus tuned for the respective model.

Besides this popular categorization, an additional one was proposed by Li *et al.* [106], summarized in Figure 2.15. This approach arranges feature selection techniques according to the type of data being handled. The first split is between **static data** and **streaming data**. Static data is data that does not change and whose organization is known, *i.e.* data with a known number of features and size, corresponding to non real-time data. On the other hand, streaming data is real-time data being captured, for instance, by sensors. Thus, it is data being sequentially received without *a prior* knowledge about its size and/or number and size of features. Within this later group, when the number of features is fixed but the size of the observations is unknown, dynamic observations are created, corresponding to **data stream** features. Contrary, when the number of observations is fixed but the size of features is unknown, features are created dynamically, being added or removed as new data arrives (**feature stream**).

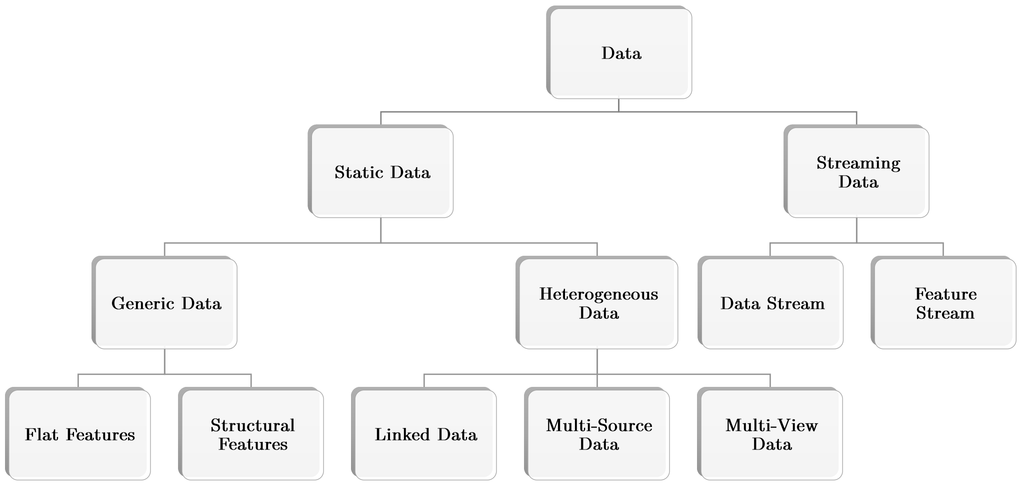


Figure 2.15: Categorization of feature selection algorithms from Data perspective (based on [106]).

Within the static data group, two main distinctions can be identified, namely **generic data** and **heterogeneous data**. Generic data is the most simple type of data, provided by single sources, with no linked data. On the other hand, heterogeneous data deals with more complex data, being considered **linked data** when there are intrinsic relationships within the dataset, **multi-source data** when the data is gathered by different entities or **multi-view data** when the different set of features can be extracted from the same dataset.

Returning to generic data, the most simple type of data, also two big groups are identified. **Flat features** represent data whose features are considered independent, with no intrinsic relation between them. Secondly, **structural features** are features that present some organized relationship between each other, for instance a tree or a graph.

From all the groups mentioned, the traditional feature selection methods used in ML applications are inserted in the flat features. As theses methods are simpler, generic and are deeply used in mostly of ML applications, a further study of some of the most relevant techniques is provided.

Figure 2.16 suggests a taxonomy for the flat features category [106]. This categorization is done according to the technical method used during the feature selection procedure. One common approach it to use a specific metric to measure the feature importance, for instance, the distance, correlation, dependency or the preservation of data similarity. The methods using the latter metric are particularly relevant and are denominated **similarity-based** methods. These kind of methods can be both applied to supervised and unsupervised applications.

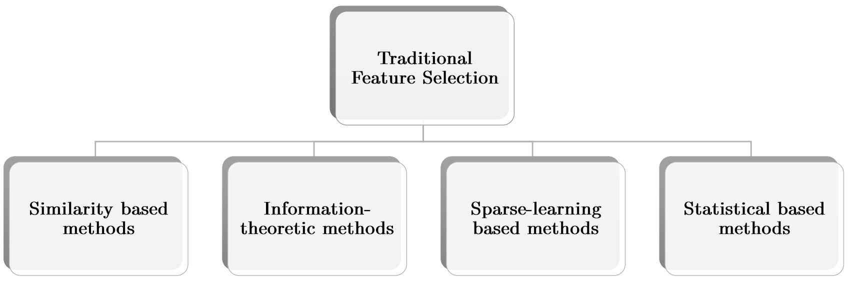

Traditional Feature Selection
Similarity based methods
Information-theoretic methods
Sparse-learning based methods
Statistical based methods

Figure 2.16: Traditional feature selection methods groups [106].

The second group is called **information theoretical-based methods** and it is composed of a wide set of heuristic filtering methods. The metrics used to measure the feature importance are typically based on maximizing the feature relevance and minimizing the feature redundancy. These methods are mainly applied to supervised learning applications with discrete features.

Sparse learning-based methods is a third group whose main characteristic is being embedded in the learning model. These methods are relevant when we want to take the best from the learning model chosen, regardless the inherent bias associated. This means that these feature selection methods can be optimal for specific algorithms but can have a poor performance in others. Particularly, concerning the sparse learning-based methods, their rationale is based on minimizing the fitting errors.

The last group to be presented encompasses **statistical methods**. These methods resort on statistical metrics and are typically applied to features individually, not considering the potential redundancy between them. As the metric used to measure the relevance of features is statistical, the bulk of methods are filters applied to discrete features.

Table 2.2 wraps up some examples of feature selection methods within each one of the four categories presented. Nevertheless, only the methods applied to supervised learning applications are emphasized.

All the presented methods have complex mathematical and statistics concepts behind them that are out of scope of this thesis. This is possible since the increasing advances in the field of DM and

ML have allowed the development of several computer tools that can be easily used by ML users. For instance, several packages and libraries are available both for Python, R, and other programming languages which avoids the exploration and implementation of all these feature selection methods.

Table 2.2: Enumeration of different feature selection methods within the flat features category, also called traditional feature selection (TFS) methods. The enumeration is done considering exclusively supervising learning methods, being presented by category, distinguished among filter and embedded methods and presenting a reference for further details. This summary was done having as basis the work presented in [106].

TFS Category	**Method name**	**Type**	**Reference**
Similarity-based methods	SPEC-extension of laplacian	Filter	[107]
	Fisher Score	Filter	[108]
	Trace Ratio Criterio	Filter	[109]
	ReliefF	Filter	[110]
Information theoretical-based methods	Mutual information maximization (information gain)	Filter	[111]
	Mutual information feature selection	Filter	[112]
	Minimum redundancy maximum relevance	Filter	[83]
	Conditional informax feature extraction	Filter	[113]
	Joint mutual information	Filter	[114]
	Conditional mutual information maximization	Filter	[115]
	Informative fragments	Filter	[116]
	Interaction capping	Filter	[117]
	Double input symmetrical relevance	Filter	[118]
	Fast correlation based filter	Filter	[119]
Sparse learning-based methods	Feature selection with $l_1 - NORM$ regularizer	Embedded	[120]
	Feature selection with $l_{2,1} - NORM$ regularizer	Embedded	[121]
	Efficient and Robust Feature Selection	Embedded	[122]
Statistical methods	T-score	Filter	[123]
	F-score	Filter	[124]
	CHI-square score	Filter	[125]
	Gini index	Filter	[126]
	Correlation-based feature selection	Filter	[127]

2.3.3 Algorithms

Supervised learning uses labelled data, which besides being less complex to implement, , an important feature when dealing with low-power devices. There is a wide set of algorithms that can be used to solve ML tasks. Most of them are deeply related with probabilistic functions on which the problem to be solved is either a parameter optimization or a hypothesis search problem. Hence, ML algorithms that lie under the scope of this category are firstly described. Then a brief description of models that simulate the activity of human neurons (Neural Networks) are described. To the end are left the most understandable models, specially in terms of their outputs, that retrieve from historical observations, general logic hypothesis: Decision Tree (DT) and rule-based classifiers.

2.3.3.1 Discriminant Analysis

Discriminent Analysis (DA) is a simple classifier that uses discriminant functions to calculate boundaries for each class [128]. Different groups of algorithms exist, being Linear Discriminent Analysis (LDA) one of the most popular. When applying LDA, a dimensionality reduction technique seeks for linear boundaries between combination of features that best separate two or more classes. The first version of LDA was designed exclusively for 2-class classification problems [129], being extended to a multi-class classification problem by Rao [130], introducing the Multi-class Linear Discriminant Analysis or Multiple Discriminant Analysis.

Despite having been extended to support also multi-class problems, this type of models is preferably used in two-class classification problems. Considering this scenario, the main idea behind is to find a separation axis that better separates the two supervised classes. This separation axis is calculated according to two simultaneous criteria, namely, maximizing the distance between the means of the two classes and minimizing the variation within each class.

The original LDA model assumes that classes are linearly separable, the data follows a Gaussian distribution and each of the classes has the same variance. However, as these assumptions do not always hold, some extensions to LDA were developed to overcome these limitations, particularly: Quadratic Discriminent Analysis (QDA), on which it is not assumed an identical variance between classes; Flexible Discriminant Analysys (FDA), that allows non-linear combination of features; and Regularized Discriminant Analysis (RDA) that is a generalization of both LDA and QDA, allowing a regularization of the variance according to a configurable factor.

2.3.3.2 Naive Bayes

The Naive Bayes algorithm [131] is a probabilistic model commonly used in classification problems. It is based on the Bayes Theorem that states that the probability of observing the event Y considering that X occurred - named *posterior probability*, is given by $P(Y|X) = \frac{P(X|Y)P(Y)}{P(X)}$, where $P(X|Y)$ is named *likelihood* and represents the probability of observing X given Y, and $P(Y)$ represents the probability of Y without considering X - called *prior probability*. Having this theorem as a basis, the Naive Bayes model assumes that all the features are independent of each other. Thus, if X is a random variable composed of n elements $X = (x_1, ..., x_n)$ and if y is a class, the posterior probability can be given by $P(y|X) = \frac{P(x_1|y)P(x_2|y)...P(x_n|y)P(y)}{P(x_1)P(x_2)...P(x_n)}$. As the denominator is known, we can rewrite the last formula as $P(y|X) \propto P(y)\prod_{i=1}^{n} P(x_i|y)$.

Although this model assumes a condition that is not commonly true in real applications, is very popular in ML applications because it is easy to implement and use (since it is easy to calculate the necessary parameters), as well as it allows to predict the inputs from the outputs and vice and versa.

2.3.3.3 Hidden Markov Model

The Hidden Markov Model (HMM) is based in the Markov chain concept [132]. In a nutshell, a Markov chain is a way of modeling a sequence of events, called states, on which each possible transition is represented by a probability. Furthermore, considering these states, it assumes that the following state only depends on the actual state and not in previous ones. From this description, some similarities with the theorem of Bayes can be found, particularly concerning the coexistence of the conditional probability concept. The hidden concept arises from the fact that often we are interested in capture information about events that are hidden, *i.e.*, that are not directly observed.

Formally speaking, the HMM is supported by two assumptions. The first, already stated, is known as Markov assumption and states that the probability of a state depends exclusively on the previous one, *i.e.*, $P(q_i|q_1...q_{i-1} = P(q_i|q_{i-1})$, where $q_1, q_2...q_N$ are the set of states. The second assumption is called output independence and argues that an output observation o_i is only dependent of the state q_i that produced the output, thus not depending on other states or observations, *i.e.*, $P(o_i|q_1...q_i, ..., q_T, o_1, ..., o_T) = P(o_i|q_i)$, where T is the number of observations.

To clarify the concepts of state and observation, as well as the referred assumptions, an example originally presented by [133] is described thereupon. Imagine that we want to study the global warming history in a specific area but, although there are not available any records about it, there are records about the number of ice creams eaten by a person on all days of the summer and that we can infer the relationship between these two events. The goal it then to estimate the weather state considering the number of ice creams eaten. In this example, the observations are the number of ice creams eaten and the states are the weather states. The weather states are hidden because are not observable in the context of the problem, but are induced by conditions that are observable - the number of ice creams.

There are three main problems to solve in a Markov model:

- **The likelihood problem:** determining $P(O|\lambda)$, where λ represents the set of transition probabilities between states (A) and the set of observation likelihood sequence (B), *i.e.*, the probability of registering an observation o_t given a state q_i, and given an observation sequence O;
- **The Decoding problem:** determining the hidden state sequence $q_1, ...q_N$ (Q) given λ and the observation sequence O;
- **The Learning problem:** determining the best values for A and B given O and Q.

There are several algorithms that can help solving these problems. However, the most popular are the forward algorithm for the likelihood computation, the Viterbi algorithm for decoding and the Baum-Welch or forward-backward algorithm for the learning problem [132].

The features presented make HMM particularly interesting for pattern recognition. Some examples are the detection of biological sequences [134], speech and handwriting recognition [132], and anomaly detection [135]. However, the algorithms typically used to solve the problems involved are computationally expensive and time-consuming, specially with large models. Also, albeit small models can be understandable, the same is not true with large models.

2.3.3.4 Logistic regression model

The models presented so far lie on a class of models called *generative*. This is so because they depend on the conditional probability $P(X|Y)$ to predict the outputs Y. In contrast, *descriminative* models aim at modeling the conditional probability $P(Y|X)$ without considering the joint probability $P(X \cap Y)$. In other words, *discriminative* models use the training data to find the best separating boundaries between classes, using then those boundaries to predict the outputs of an unlabelled dataset. Since these models are less dependent on the probability distributions, they are typically better suited for classification and regression problems.

One of the simplest *discriminative* models is known as the logistic regression model [136]. This model is used in binary classification problems, *i.e.*, problems on which only two outputs are possible. Mathematically speaking, the logistic regression model calculates the separation boundary with a logistic function that can be represented as $\frac{1}{1+exp(-x)} = \frac{e^x}{e^x+1}$. This model is seen as an extension of the linear regression model that performs well in binary regression problems, but not in binary classification ones. Thus, it also assumes a linear boundary between features to predict the output but resorts on a logistic function to create that boundary. Thus, the logistic regression model can be expressed as $log(\frac{P(Y=+1|x)}{1-P(Y=+1|x)})$, where $P(Y=+1) = \frac{1}{1+exp(-(\beta_0+\beta_1 x_1+...+\beta_p x_p))}$.

Albeit being considered an efficient, easy to implement and interpret, and low-processing model, it does not behave well with a large number of features, is vulnerable to overfitting and does not solve non-linear problems. Although the logistic regression model only address binary classification problems, there is an extension of it, called multinominal regression, that allows the classification of multi-class problems. However, its use is not popular and as it follows all the rationale already presented for the case of logistic regression, is not addressed in more detail.

2.3.3.5 Support Vector Machine

Similarly to the logistic regression model, the Support Vector Machine (SVM) model [137] also attempts to model classification or regression problems through separation boundaries that better separate data. In the particular case of the SVM model, these boundaries are hyperplanes whose separation is intended to be maximized. The original model of SVM deals only with linear problems, albeit it can be extended to non-linear problems athwart the use of kernel functions, *i.e.*, functions that transform non-linear problems in linear problems [138].

To understand the basic concepts behind the model, let's consider a simple problem (as detailed in [139]) where the training dataset has inputs in $\mathbb{R}^d$, the output is binary and the problem is linear. Considering this scenario, the SVM model tries to find the hyperplane H that betters splits the two groups of outputs. The hyperplane H can be defined as $f(x) = 0 = \langle w, x \rangle + b$, where $\langle , \rangle$ represents the dot product in $\mathbb{R}^d$. This hyperplane aims at maximizing the distance of all points of data, being the margin of an observation (γ_i) the distance between it and the hyperplane H. Thus, considering an observation i, the margin of the hyperplane H is defined as $\gamma_H = min_{i-1,...,n}\gamma_i$, *i.e.*, it coincides with the nearest observation to the hyperplane H. Typically, many hyperplanes can be used to separate data, thus, the goal of SVM is to encounter the one that maximizes the referred margin, *i.e.* $H^* = \underset{H}{argmax}$ γ_H.

SVM is a very popular model, particularly because it works well when data is clearly separable; it is efficient in high dimensional spaces; and deals well both with structured and semi-structured data. However, it also includes some disadvantages that must be weighted when solving a ML problem. Particularly, it is not suitable for large data sets, specially due to the training time required; it does not perform well with noisy data; it does not allow small calibrations; it requires the use of kernel functions to solve non-linear problems, being sometimes difficult to find the suitable function to use;

and it is also a model whose outputs are difficult to understand and interpret.

2.3.3.6 K-Nearest Neighbour

kNN belongs to the family of the *Nearest Neighbour* algorithms that were originally introduced by Fix *et al.* [140]. Briefly, this type of classifiers uses the concept of proximity (nearest) to classify unlabelled data. The main idea behind this type of algorithms is that data that belong to the same class are likely more similar (*i.e. nearest*) between each other. The classification of unlabelled data is inferred from the classes of the training dataset. Typically the number of neighbours considered to classify a certain entry of the dataset is greater than one, being that the reason for the name kNN, where k represent the number of neighbours considered.

The kNN is both a nonparametric and lazy learning algorithm [141]. It is nonparametric since it is distribution-free, *i.e.*, does not rely on any specific distribution of data, and it is lazy learning because most of the computational effort is drained during the testing phase, being its training phase minimal. Commonly, almost all or even all training data is used during the final classification, which entails a very significant computational effort, especially with large datasets.

Three main steps can be identified when applying kNN [141]. Firstly, the distances between an unlabelled entry x_i and all entries of the training dataset are computed. Typically the Euclidean distance is used, although different ones are also plausible [142]. Secondly, the k nearest neighbours are selected. Finally, the entry x_i is classified considering a voting system among the k nearest neighbours. This voting system can be simply the choice of the most frequent class among all the neighbours or a weighted voting system considering the distances measured.

A small value of k empowers the effects of noisy neighbours. In contrast, a high value of k demands costly computational efforts. These two considerations identify two of the main disadvantages of kNN. Additionally, it has a poor performance in unbalanced datasets and its understandability can be a barrier, specially with a high number of features. On the other side, kNN are simple to implement and have an uncomplicated tuning phase, since we only have to tune the number of k and the distance metric. Another relevant advantage concerns the opportunity to update the model on real-time.

2.3.3.7 Decision Trees, Rule-based Classifiers and Random Forest

DTs [143] constitute a model on which successive tests are made to each feature of the dataset, being defined nodes and branches accordingly to the likelihood of those tests. A DT is composed of: a *root* node, compounded by the feature that better divides the training dataset; a set of *non-leaf* nodes, representing specific and relevant tests on particular features; a set of *branches*, personifying the outcome of those tests; and a set of *leaf nodes*, containing the final classification for the set of tests from the root node until that leaf node (terminal node).

Besides being simpler to implement and requiring low processing, DTs have the particular feature of being easily understandable by humans. Particularly, the output of a DT can be directly transposed to a set of *if's* and *else's*, being the conditions defined by the tests performed on each node.

A common issue felt during the application of this type of modelling corresponds to the overfitting of the training data. This problem occurs when the model is particularly tuned to a specific training data and does not generalize well. Hence, a particular attention must be held on the application of DTs. There are typically two common approaches to avoid overfitting. The first is to simply stop the training before the model fits too much the training data. In practice this means that if the results are close to the perfection, probably we are facing a overfitting issue. The second approach is to prune the tree. A tree with a high number of splits usually contains a set of nodes and branches that were included to match the needs of the training set. Finding the most suitable moment to stop training the model or the most suitable level of pruning is not an exact science and involves, typically, several tests.

Even if it is possible to apply DTs both to continuous and discrete features, they tend to deal better with discrete features. The most popular algorithm that implements this type of models is C4.5 [144], being available on most of ML tools.

Rule-Based classifier (RB) are similar to DTs in the sense that both allow the extraction of a set of conditions from the training dataset. However, although in DTs the rules need to be retrieved from the graph obtained, on rule-based classifiers, the rules are induced directly from the training dataset in a form of *if-then's*. However, while DTs models follow a strategy on which the rules start from finding the general split features and get more specific while descending on the tree, RB models also allow to start on more specific conditions and ascend to more general conditions. Additionally, typically the rules induced from RB models are simpler and fewer, which can be either an advantage or a disadvantage, depending on the application scenario.

Random Forest [145] or Random Decision Forest (RDF) is a ML model that lies on a group of algorithms entitled ensemble learning. Shortly, this type of learning considers the aggregation of several techniques to solve an unique problem. In order to minimize the weak points of individual learning models such as instability or overfitting, this type of learning combines the use of different models, applying different coefficients to each output from each model used. RDF, particularly, combines several DTs, using then a voting system to select the prediction output. Nowadays it is a very popular model, not only because of its prediction accuracy but also because it offers typically a low processing time, even for large datasets. Notwithstanding, even if it is based in DTs that are understandable, the combination of multiple DTs, make its interpretation harder than with simple DT or RB.

Still within the ensemble learning grouping, we highlight the gradient boosting DT implemented for instance by XGBoost package [146]. This approach continuously interacts through multiple prediction attempts, aiming at reducing the error of the model that is calculated using a gradient descent algorithm. It is a powerful algorithm due to its efficiency and scalability.

2.3.4 Neural networks

A different learning approach is presented by Neural Networks (NN)s [147]. Albeit resorting also in mathematics, it tries to emulate the operation of the human nervous system. This mean that the model defines artificial neurons that are connected between each other through weighted connections. These artificial neurons are processing sources and the connections between neurons allow the storage of the knowledge acquired through learning procedures.

Each NN is composed of three types of layers: one input layer that receives inputs; one or more hidden layers where most of the processing occurs; and an output layer where processing outputs are given. The way neurons are connected results in different types of neural networks. A NN is classified as feedforward network, when there is not feedback between layers, *i.e.*, the information is always propagated into the output layer; or as feedback network when there is some sort of feedback between outputs and inputs.

Artificial neurons emulated in Artificial Neural Network (ANN) are named perceptrons, and the most popular model is named Multi-Layer Perceptron. In this model, one or more hidden layers cooperate to acquire knowledge according to inputs received by the input layer. The perceptrons are connected by weighted links, being these weights adjusted by learning procedures. The most popular one is called backpropagation learning [148], on which the error is propagated backwards through the network, triggering the adjustment of the perceptrons' connections weights.

ANNs are seen, for a long time, as a very promising ML and AI mechanism. Particularly, with the incessant computational processing increasing, it is believed that it would be possible to have closer and closer emulations of the human brain activity, which is seen as a potential solution for many problems. In fact, deep learning algorithms [149] are a consequence of that tendency. ANN is a suitable algorithm when there is a need to deal with a dataset with a high number of inputs or with a lot of noise. On the other hand, the predicting models are not human understandable and they have an inherent high computational cost.

2.3.5 Evaluation and performance metrics

Solving a ML problem means following an iterative process on which the model evaluation is vital to the redefinition of the problem solving strategy. Nevertheless, this process is not straightforward, depending on the dataset's characteristics and on the kind of results intended (*e.g.* maximizing the correctness of one specific class, maximizing the global accuracy of the model, etc.). On this section, several evaluation metrics are detailed, specially the ones applied to classification problems.

2.3.5.1 Confusion matrix

The confusion matrix is the most intuitive method to evaluate a ML model. Albeit not being by itself an evaluation measurement, it is the basis of many metrics. A confusion matrix, also called table of confusion, is commonly used in supervised learning problems and besides being possible to use it in multi-classification problems, we consider a two-class example for the sake of simplicity. The skeleton of a 2-class confusion matrix is depicted in Table 2.3. Each cell of the confusion matrix summarizes the number of observations that the model correctly predicts and the observations of cases that the model misses. The following definitions are used:

- **True Positives (TP):** the number of observations on which the model correctly predicts the positive class;
- **False Positives (FP):** counts the number of observations that the model predicts the output as belonging to the positive class, although belonging, in fact, to the negative class [4];
- **True Negatives (TN):** similar to the number of TP but for the negative class, *i.e.*, the number of observations on which the model correctly predicts the negative class;
- **False Negatives (FN):** similar to the number of FP but for the negative class, *i.e.*, the number of observations on which the model predicts the output as belonging to the negative class, although belonging, in fact, to the positive class.

From the definitions presented, it is clear that in an ideal algorithm/model, the number of FP and FN should be zero. Nonetheless, this scenario seldom occurs in real applications. Thus, when solving a

[4]In multi-class problems, this value is composed of the sum of all parcels, namely the number of FP associated to each one of the classes that are not being considered as positive.

ML problem we are mostly interested in minimizing the number of FP and/or FN. Albeit, ideally, both should be minimized, that is not always possible, particularly because there is often a relationship between them, *i.e.*, minimizing the number of FP may increase the number of FN and vice and versa. Hence, the focus on minimizing these parameters is always dependent on the application scenario. Two common examples to depict this situation are the detection of diseases (ill is the positive class) and the classification of an e-mail as *spam* (*spam* is the positive class). In the former example, it is typically better to classify a patient with a disease when in fact he is not ill, *i.e.* it is better to minimize the number of FN instead of the number of FP. In the latter example it is preferable to classify an e-mail as not being *SPAM* although actually being *SPAM*, *i.e.* it is preferred to minimize the number of FP instead of the number of FN because we do not want to loose important e-mails.

Table 2.3: Example of a 2-class confusion matrix.

		True diagnosis	
		Class1 (positive)	Class2 (negative)
Predicted	Class1 (positive)	TP	FP
	Class2 (negative)	FN	TN

2.3.5.2 Binary classification metrics

From the confusion matrix, several evaluating metrics can be extracted. Thereupon are described the metrics commonly used in binary classification problems.

- **Accuracy**: is one of the most used metrics, even if it can lead to misleading conclusions. On the one hand, it is frequently used because it evaluates the ratio between the number of all correct classifications, *i.e.*, TP and TN, and the total number of entries of the dataset (see Equation 2.5). On the other hand, it can prompt misleading conclusions, since in a dataset where a majority class exists (a class that is clearly predominant), the accuracy is greatly influenced by the prediction results of this class. Particularly, if the model predicts the majority class with high accuracy and the remaining ones poorly, the accuracy evaluated is high, although there are classes on which the results are poor. Therefore, this metric shall only be considered in nearly balanced datasets.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + TN} \tag{2.5}$$

- **Precision**: also called Positive Predicted Value, evaluates how many predictions are correct from the total of predicted values of the positive class. Equation 2.6 shows the referred relationship, which may be used, particularly, to understand the impact of the number of FP in predicting capabilities. Thus, if the goal is minimize the number of FP, the precision value shall be close to 100%.

$$Precision = P = \frac{TP}{TP + FP} \tag{2.6}$$

- **Recall**: recall, sensitivity or True Positive Rate, gives a notion about how the model behaves in terms of FN. Hence, the recall calculates the correctness of the positive predicted class considering the total number of real positive cases, *i.e.*, how much the model missed in predicting the positive class. This is represented in Equation 2.7, being its use relevant when the goal is to minimize the number of FN.

$$Recall = R = \frac{TP}{TP + FN} \tag{2.7}$$

- **Specificity**: specificity, selectivity or True Negative Rate is very similar to the recall measurement. However, instead of focusing in the actual positive class, it focuses in the actual negative class. This is reflected in Equation 2.8.

$$Specificity = \frac{TN}{TN + FP} \tag{2.8}$$

- **F1 Score**: in many ML problems, we are not interested in evaluating separately the values of both precision and recall. Instead, we are interested in having a single metric that could give insights about both values. That is the purpose of the F1 Score that is given by the harmonic mean of both precision and recall. A harmonic mean is used rather than a simple arithmetic mean since its behaviour is preferable when the values of precision and recall are very disparate. For instance, given a precision of 1% and a recall of 99%, the arithmetic mean is 50% while the harmonic mean, that is given by Equation 2.9, is 1.98% (close to the value of the precision). Thus, while the 50% does not give any relevant information, a value of 1.98% immediately shows that one of the values is presenting poor results.

$$F1Score = 2\frac{P \times R}{P + R} \tag{2.9}$$

- **Receiver Operating characteristic Curve (ROC) and Area Under the Curve (AUC)**: to facilitate the visualization of model's performance, ROC graphs are commonly used, mainly in binary classification problems. It represents the relationship between true positive rates and the false positive rates [150]. From these ROC graphs, an additional metric, the AUC can be extracted, being always a value between 0 and 1. As high values of true positive rates are intended, in contrast to lower values of false positive rates, high values of AUC are associated to better performances.

2.3.5.3 Multi-class metrics

In the scope of this thesis, resorting on the commonly used binary classification evaluation metrics is not sufficient. In fact, the ML problem to be addressed in the work, corresponds to a multi-class classification problem, being indeed relevant to study evaluation metrics that better fit to this scope. Subsequently, the next topics detail those metrics.

- **Micro-average**: in a multi-class problem, it is not always reasonable to individually evaluate the metrics of each individual class, specially because it would be difficult to analyse and interpret those results. Hence, several strategies may be used, being one of them the micro-average. Basically, the micro-average corresponds to an operation applied to other known metrics as precision, recall or F1 score, being used all classes of the dataset to calculate the performance of the model. In other words, it is a method that considers the individual metrics associated to each class. In the case of the micro-average, three important variants are defined, namely the micro-averaged precision (P_{micro}), the micro-averaged recall (R_{micro}) and the micro-F1 score ($F1_{micro}$) whose formulas are described in Equations 2.10, 2.11 and 2.12, respectively.

$$P_{micro} = \frac{\sum_{i=1}^{N} TP_i}{\sum_{i=1}^{N} TP_i + FP_i}, \tag{2.10}$$

$$R_{micro} = \frac{\sum_{i=1}^{N} TP_i}{\sum_{i=1}^{N} TP_i + FN_i} \tag{2.11}$$

where N is the total number of classes of the model.

$$F1_{micro} = 2\frac{P_{micro} \times R_{micro}}{P_{micro} + R_{micro}} \tag{2.12}$$

A high value of $F1_{micro}$ suggests that the model has a good performance on overall. However, a con of this metric is that it is not suitable to be used in imbalanced datasets because it is not sufficiently sensitive to the performance of individual classes.

- **Macro-average**: like micro-average, the macro-average also offers a way of evaluating the performance of a multi-class model using single values, instead of using one per class. The macro-average is nothing more than an average of individual metrics. Thereby, the macro-averaged precision (P_{macro}), the macro-averaged recall (R_{macro}) and the macro-F1 score ($F1_{macro}$) are given by Equations 2.13, 2.14 and 2.15, respectively.

$$P_{macro} = \frac{1}{N}\sum_{i=1}^{N} \frac{TP_i}{TP_i + FP_i} \tag{2.13}$$

$$R_{micro} = \frac{1}{N}\sum_{i=1}^{N} \frac{TP_i}{TP_i + FN_i} \tag{2.14}$$

$$F1_{macro} = \frac{2 \times P_{macro} \times R_{macro}}{P_{macro} + R_{macro}} \tag{2.15}$$

In contrast to $F1_{micro}$, a high value of $F1_{macro}$ suggests a good performance of individual classes, which means that its use is more suitable for imbalanced datasets. Even though, it can also lead to misleading conclusions since it is a simple arithmetic mean, which means that if a single class has poor results, it will greatly influence the final results, even if, for instance, that class represents a small portion of the dataset.

- **K-category correlation coefficient**: albeit the F1 score, more properly the macro-average F1 score, may have an acceptable behaviour in imbalanced datasets, as it does consider all the four confusion matrix categories, it may lead to misleading conclusions. The Matthews Correlation Coefficient (MCC) [151] tries to overcome that issue by considering in its formula (Equation 2.16) a portion of each class. However, the MCC can only be applied to binary-classification problems. An extension to this approach was presented by Gorodkin [152], which the author entitled of K-category correlation coefficient (R_K) since it extends the concept of the MCC to K classes (see Equation 2.17). Like MCC, the values obtained for the R_K can vary between -1 and +1, being the +1 associated to the highest performance.

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP) \times (TP + FN) \times (TN + FP) \times (TN + FN)}} \tag{2.16}$$

$$R_K = \frac{\sum_{klm} C_{kk}C_{lm} - C_{kl}C_{mk}}{\sqrt{\sum_k \left(\sum_l C_{kl}\right)\left(\sum_{l',k'\neq k} C_{k'l'}\right)}\sqrt{\sum_k \left(\sum_l C_{lk}\right)\left(\sum_{l',k'\neq k} C_{l'k'}\right)}} \tag{2.17}$$

where C is the confusion matrix, C_{kk} is the number of correctly predicted observations concerning class i and C_{kl} are the observations predicted as k but belonging to class l ($l \neq k$).

2.4 Summary and discussion

The solution addressed in the scope of this thesis tackles different areas whose knowledge regarding concepts, techniques, protocols, standards and algorithms are relevant for carrying out the proposed work. They are relevant either because: they can potentially be used to solve needs; they can be a tool to aid addressing specific needs; or because its knowledge is required to understand related topics.

In this chapter, we started by addressing the IoT concept, its typical communication stack and the existing protocols on the different layers of such stack. We focused the analysis in a four layered stack, with a special emphasis on the MAC Layer since its efficiency is crucial for energy constrained applications. Several protocols were presented, either CSMA-based, TDMA-based or a combination of both. From these, a combination of features presented by several existing protocols is the most promising option to achieve the defined requirements. These features include using a TDMA-based scheme as basis, using beacon messages for network synchronization and a CSMA approach for the initial setup. Additionally, it includes using implicit scheduling and a scheme based on a cyclic structure divided in windows and time slots for handling different types of traffic.

Still within the IoT area, we discussed several technologies that could be used. Besides the IEEE 802.15.4, very popular but very limited in terms of range and bandwidth, the potentiality of LPWAN solutions put LoRa, SigFox, NB-IOT and LTE-M as attractive solutions for IoT scenarios. However, they still incur in important limitations for constrained agricultural scenarios as the one addressed in this thesis. Therefore, they are not considered for the solution, but instead, a generic solution based on free ISM bands.

Together with the needs of designing a suitable architecture for energy constrained scenarios, some features require research in order to enable the construction of a supporting architecture. The need for supporting localization services is one of such features and, therefore, localization techniques were surveyed. Typically, three different phases exist. The distance estimation phase computes an estimation of the distance (or angle) between two different nodes, being the source of information for the position computation phase, where geometrical formulas are applied to define a more accurate location. Finally, additional mechanisms may be added to improve the location obtained affter the previous two phases.

GPS-based solutions are the most popular although they still incur in some drawbacks that make them unviable for many applications. GPS cost, power-consumption and size, together with common lost of signal issues, are hurdles particularly relevant for applications with a significant number of nodes. Range-based algorithms arose as alternative and resort on a distance estimation process, typically associated with communication signal features, for instance AoA, ToA, TDoA or RSSI. Despite their superior accuracy when compared to the one provided by RSSI-based methods, AoA, ToA and TDoA require additional hardware and, in some cases, precisely synchronized clocks, making these solutions complex and expensive. In contrast, the widespread availability of RSSI measurements on current radio transceivers makes RSSI-based methods very attractive since no additional hardware is required, hence not incurring on additional costs or energy expenditure. Notwithstanding, using RSSI requires signal propagation modelling, a procedure that entails several challenges, such as dealing with path loss, fading and shadowing phenomena. Thus, several methods to reduce the effects of those phenomena, together with the implementation of filtering techniques as the Kalman and Particle Filters may be proposed.

The other critical feature is the posture control mechanism that includes monitoring sheep behaviour. ML is a popular tool in several areas and applications and can be used with multiple purposes including aiding in the autonomous detection of animal behaviours. As such, we surveyed ML taxonomy, workflow, techniques and algorithms.

Using a ML-based approach means following an iterative approach through a chain that is iterated several times. The first steps concern the identification of the problem, the definition of the strategy to collect data and the collection of data. Afterwards, a very important stage is the data preparation phase. Commonly, the data gathered cannot be used directly by ML methods, either because the format is not suitable, because the data is noisy or because there are too many and redundant features. Consequently, within this stage we identified several data cleansing methods together with several feature transformation and feature selection methods. These methods are deeply related to the type of data collected and to the type of problem to be solved. Hence, the focus were methods that could be applied to data gathered by sensors.

Hereafter, the most popular modelling algorithms were presented. The analysis focused in their processing complexity, ease of implementation, training time and understandability. Considering all the characteristics and the requirements of the application scenario tackled in this thesis, DT, RB and RDF are the most suitable options. Additionally, between this reduced group, DTs allow a higher control and tuning, although they may suffer of overfitting problems that shall be properly monitored during the training and evaluation phases.

The multiple iterations are enabled by the evaluating phase where several performance metrics are calculated. The most basic method is the construction of the confusion matrix. This matrix provides an easy way do visualize the distribution of the observations that were correctly predicted by the trained model. From this matrix, several metrics can be extracted, being the most popular the accuracy, precision, recall, specificity and F1 score. Besides the simplest methods applied to binary classification problems, more advanced methods to deal with multi-class problems were presented, as micro and macro-averages. Finally, metrics suitable to be used with imbalanced datasets were provided. Here, the K-category correlation coefficient is the most promising and efficient metric.

CHAPTER 3

Related work

This chapter presents a literature survey on topics closely related with the scope of this thesis, particularly on animal monitoring and animal conditioning. Despite tackling the ambit of livestock area, the focus is on the ICT components of solutions, particularly on how ICT technologies are being used to help livestock industry to enhance its capacity and productivity through monitoring capabilities.

3.1 Animal monitoring

Monitoring the animal life, namely the inter/intra-species relationships, the interactions with the surrounding environment or merely monitoring animals' location, presents a great relevance to the scientific community. Biologists, zootechnics, livestock breeders, environmental engineers or ecologists, all need to gather data to optimize and improve their work. In that scope, several areas of interest on animal monitoring can be identified, for instance, location monitoring [153], pasture monitoring [154], welfare monitoring [155], reproductive cycle monitoring [99] and behaviour monitoring [156]. As it may be identified through the following sections, these areas are not mutually exclusive, existing a natural overlapping between them.

The goal with this section is to show that, despite the existence of several works that tackle some of the issues addressed in the scope of this thesis, there is not a single solution that solves all the research questions stated in Chapter 1.

3.1.1 Location monitoring

Monitoring animals' location is one of the most relevant and oldest needs of the livestock sector. It is portrayed in the literature with special emphasis both in wild and marine animals but also in domestic ones, such as bovines, ovines and goats. If in the case of wild and marine animals, the justification hangs on the difficulty of observing in loco their movements, essential to study their behaviour, in the case of domestic and livestock animals, the justifications are wider. Besides the importance of knowing the position of each animal for security reasons, such information may also be used as a tool to perceive health conditions, common behaviours and societal interactions, as well as to optimize pasture management [157], altogether contributing for the optimization of farming processes and animal production.

Consequently, over time, several solutions for animal localization were proposed, commercially and academically, both for wild and domestic animals. Direct approaches or absolute localization techniques, where an absolute location (a set of coordinates) is given as an output, are the most used. Within this group, GPS, sometimes combined with accelerometers, stands out. Its use is portrayed in a vast number of works in the literature and is also on the basis of many commercial solutions offering localization services.

Within the commercial group, all products aim at offering a localization service easily accessible via a web interface, either in web browsers or mobile applications. They are mainly distinguished by the animals for which they were designed for, the autonomy of the devices, the technology used to communicate the coordinates and the availability of a virtual fence mechanism. Focusing on solutions applied to livestock animals, products as tNet Smart Agriculture Cow collars (tNet Collars) [158], eShepherd [159], Digitanimal [160] and Nofence [161] excel. In common, they share the use of GPS for determining animals' absolute location, though the way the information is uploaded to backend services differ. For instance, while the eShepherd and tNet Collars use LoRa, NoFence uses Groupe Spécial Mobile or Global System for Mobile Communications (GSM)[1]. Additionally, while solutions as eShepherd and Nofence provide energy harvesting mechanisms, particularly through the use of small integrated solar panels (thus, according to the manufactures, allowing a theoretical infinity autonomy), the solution offered by Digitanimal announces an autonomy of about 1 year[2]. Both eShepherd and tNet's solutions were developed to be used on cattle, while NoFence, despite being initially developed for goats, its use was extended to handle sheep and cattle. Digitanimal presents solutions to all the referred groups. Finally, while eShepherd and NoFence offer a virtual fence mechanism, Digitanimal and tNet's solution only offer monitoring services.

Besides these commercial solutions, several literature works portray the use of devices incorporating GPS to locate cattle [162], [163], white-tailed deers [164], griffons [165], crocodiles [166] and sheep [167]. In this later work, a GPS, together with a jaw and a lying/standing sensor are used to monitor grazing areas of domestic sheep. However, the device used by this solution weights almost 2 kg, needs to be transported in the back of the sheep and has an estimated autonomy of few days. The short autonomy of the device highlights one of the GPS limitations, which concerns the high-energy consumption, that conjugated with its high cost and frequent loss of satellites connection [168], makes it unsuitable for animals' localization, particularly for small to medium size ones.

Several methods have been proposed to mitigate these disadvantages. One common approach is decreasing the duty-cycle of GPS devices, as exemplified in the work of Ruiz-Mirazo *et al.* [169]. However, even if energy consumption can be brought to acceptable levels, GPS-based solutions still are too expensive for large flocks. Hence, an alternative solution is reducing the required number of GPS devices. Maroto et al. [153] propose the use of a combination of GPS and BLE tags, where only the herd leaders carry GPS devices. The non-leader animals are localized by proximity, *i.e.*, through the detection of a successful BLE communication with one of the leaders. Although the authors disclose an autonomy of 365 days for GPS-based devices and 280 days for the BLE-based devices, the duty-cycle is very low, ranging from 30 minutes to 51 minutes between successive tag reads. A similar solution is proposed by Llaria *et al.* [157], although this latter work does not comprise the use of tags in animals not equipped with GPS devices. Also, besides exhibiting a very low duty-cycle (1 measurement per hour) and errors up to 20 m, the system does not even allow the detection of all animals, which is a severe limitation.

Therefore, new alternatives based on algorithms that estimate the relative position between nodes started to arise. Despite their well-known limitations, RSSI-range-based localization mechanisms hold

[1]At the moment of thesis' writing, the technology used by Digitanimal was not available.
[2]There isn't reported information about tNet's solution autonomy.

a tremendous potential, since they enable animal localization resorting only to Radio Frequency (RF) information. This is particularly appealing when the application scenario already includes such RF communications in its architecture. In that scope, Huircan *et al.* [170] propose an RSSI-based localization system for cattle using the ZigBee communication stack. The solution uses RSSI for distance estimation and a Ratiometric Vector Iteration algorithm for location estimation. Measurements are taken every 2 minutes and the obtained errors range from 5.2 m up to 43 m.

Still in this area, the *Electronic Shepherd* [171] was evaluated to monitor the position of a flock in mountains. The architecture of the solution is based on Ultra High Frequency (UHF) modules, carried by sheep, that are capable of communicating with access nodes (gateways), composed by an UHF module, a Wide Area Network (WAN) 802.11 module to transmit data to the Internet and a GPS. Regarding the gateways, two different approaches were tested, namely the use of fixed gateways and the use of mobile gateways. Even thought in the former case the gateways were positioned in places strategically chosen, the range of the UHF modules was too short taking into account the grazing areas involved. Additionally, the 802.11 modules presented a high energy consumption and revealed to be ineffective in sloping areas. Mobile gateways were thus tested assuming that a flock stays together and that a sheep leader exists. That leader was chosen to carry the gateway device, composed off an UHF module, a GPRS and a GPS module. However, this solution experienced some complications, as low autonomy of the gateway, constant communications failures and lack of robustness of the devices.

Also following a range-based approach, Dopico *et al.* [172] designed a solution for monitoring reindeer's location using three types of nodes: secondary nodes that are devices carried by animals, holding a 433 MHz radio transceiver; beacons or primary nodes that are battery-powered mobile nodes composed of a GPS and two radio transceivers (operating in 433 MHz and 166 MHz bands), being carried only by the leaders of the herd; and repeaters, that consist on static nodes operating at the 166 MHz frequency. The system also comprises a central station that, besides owning a 166 MHz radio, includes a GSM interface to communicate with a remote computational platform. In this system, secondary nodes transmit a small identification message whenever energy is available from a kinetic harvesting module. This data is then collected by a primary node, if one is in range of the communication. Similarly, whenever a primary node is in the range of a repeater, the repeater relays all the data gathered to the central station, which in turn transmits them to a server in the Internet. This information is then used to compute reindeer's position using the GPS information and the communication information between the secondary and primary nodes. For the latter type of data, the authors tested several approaches, particularly connectivity + fusion (C-F) information, a regular Kalman Filter (KF), an Unscented Kalman Filter (UKF) and local Distance Tracking and joint localization with Gauss-Newton (DT-GN). They came to the conclusion that DT-GN is the most appropriate solution considering the trade-off between complexity and performance. Also, they conclude that the Connectivity + Fusion (C-F) algorithm provides results similar to the ones presented by Kalman-based solutions (a rough location) but are less complex.

Despite the existence of several similarities, particularly in what concerns to the system architecture, between the approaches presented and the architecture proposed in the scope of this thesis, there are a few aspects that are distinctive. Specifically, in this thesis we are interested in tracking all animals (and not only a few) and with a higher duty-cycle comparing to state-of-the-art solutions. Moreover, regarding sheep's location, we aim at using the existent hardware infrastructure to estimate their location, including the use of filtering approaches combined with behavioural information, instead of the typical filtering approaches that use only acceleration data, which is not yet, to the best of out knowledge, addressed in the literature.

3.1.2 Behaviour monitoring using Machine Learning-based approaches

The emergence of an uncountable number of IoT applications in the agriculture sector, together with the development of powerful tools for dealing with Big Data, have granted the opportunity to retrieve additional and valuable information from agricultural systems. In that scope, ML techniques have been increasingly used in different domains of the agricultural sector such as crop management, water management, soil management and livestock management [173]. In the latter, most of the applications are related to animal monitoring tasks. For instance, welfare monitoring [174], reproductive cycles optimization [175]–[177] and pasture management [178] are just a few examples where ML techniques are being applied to enable the development of new animal monitoring applications, more efficient, accurate, cheaper and less labour intensive.

The target of these applications is vast, going from marine animals such as fur seals [179] and sharks [180], birds [165], wild animals as cheetahs [181], Pumas [182] or Badgers [183], to livestock animals such as cattle [184]–[188], goats [189], horses [190] and sheep [156], [191]–[196]. From this list, cattle monitoring systems stand out, not due to the number or relevance of the works, but instead because of the high commercial interest on the products begot by those animals. An evidence of this is the number of commercial solutions that have already penetrated in the market, as the cases of CowScout [197], Cowlar [198], Cow Alert [199], Cow Manager [200], SCR by Allflex [201], MooMonitor [202] or e-shepherd [159], whose common goal is the activity monitoring of cattle towards an improved efficiency in livestock production, either by improving the lactation, insemination and feeding processes or enabling an earlier disease detection.

The increasing rivalry within the sector is leading to a decrease of the price of this kind of solutions, opening new opportunities in farms where sheep are the main livestock animal. Thus, besides a very few commercial solutions can be found in the market for sheep monitoring [160][3], several relevant research works can be identified. These works differ mainly in how the monitoring device is coupled to sheep (ear tag, collar, leg tag or halter), the number and type of states to be classified, the features used in the model and the sampling rate of the sensors incorporated. A deep analysis on these works is done below, with focus on the applications that target sheep, due to three main reasons: *i)* their application scope (*i.e.* animals) is the same as the one presented in this thesis; *ii)* they use similar techniques to the ones used on cattle monitoring platforms; and *iii)* they are quite recent. As it will be clear through the analysis, the order we present the works is neither related with its date, neither with the ML approach followed, but instead with the number of behaviours considered for classification.

Monitoring and identifying the jaw movements of sheep was the main goal of the work of Giovannetti *et al.* [156] since, accordingly to the authors, such information could be used by livestock producers for improving their grazing systems. Consequently, this work exploits the inherent relation between grazing activities and jaw movements, to build a supervised-based decision system capable of distinguishing three different feeding behaviours: grazing, ruminating and resting.

To gather data, they resort on a halter-placed device (the placement of a halter is illustrated

[3]In fact, the main goal of this solution is the tracking of animals' location although it also includes some behavior monitoring features.

in Figure 3.1) composed of a X-Bee platform containing an accelerometer, a microcontroller and a radio. The accelerometer is the source of data and allows the collection of raw acceleration data with a frequency of 62.5 Hz. However, the microcontroller only sends the first three peaks of accelerations per second and per axis through the radio to a computer, also equipped with a similar radio. During the tests, sheep carrying the devices were video-recorded to enable the classification of the data gathered. This classification consisted on assigning, from the defined feeding behaviours, the prevailing one in each OW, here defined as 1 minute. Also, for the data associated to each one of the frames classified, 12 features were added to summarize the acceleration measurements resulting from that interval. These features were mean, variance and Inverse Coefficient of Variation (ICV) (*i.e.* the mean of the standard deviation), all applied to the three axis acceleration components (x, y and z) and to the acceleration vector's magnitude ($||\mathbf{A}||$).

The feature selection was done using Stepwise Discriminant Analysis (SDA) and its efficiency was tested using Canonical Discriminent Analysis (CDA), an extension of the DA model for multi-class problems. To model the feeding behaviours, the authors used a DA algorithm. The overall accuracy of the model was 93%, considering the seven features selected (the four variances, the $\overline{A}_z$, the ICV_x and the ICV_z). Beyond the three feeding behaviours analysed, grazing presented the best and most consistent results (sensitivity 96%, specificity 97%, precision 95%), while ruminating and resting behaviours presented lower values for all the metrics evaluated.

Following a similar methodology, the work of Decandia *et al.* [193] aimed at discriminating three main sheep behaviours (grazing, ruminating and other behaviours) while studying the effects of the OW sizes in the performance of the predicting system. The main differences between the methodology followed in [156] and [193] are: *i)* the existence of an additional sensor to measure the force applied by the jaw during its movements (a force sensor); *ii)* the addition of three additional features (mean, variance and ICV) related with the data measured by that sensor; *iii)* the calculation of the features for different OW sizes (5 s, 10 s, 30 s, 30 s, 60 s, 120 s, 180 s, 300 s); and *iv)* the establishment of three levels of prevalence within the OW (50-75%, 76-99% and 100%), *i.e.*, the percentage of agreement that should exist between the different classifications within a OW.

The feature selection procedure highlighted the $s^2(||\mathbf{A}||)$ and $\overline{A}_z$ as the most important features

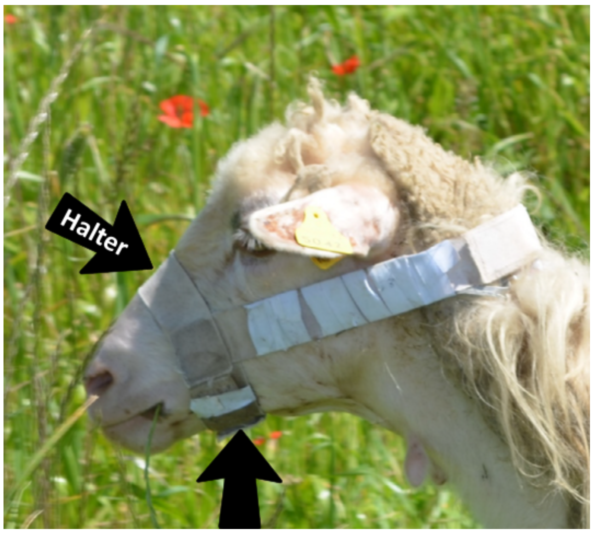


Figure 3.1: Example of an accelerometer placed in a halter [193].

for all the OW's duration and discarded the $\overline{||\mathbf{A}||}$ due to its lower importance. Hence, a total of 14 features were used by a DA model, being the best (global accuracy of 89,7%) and worst performances (global accuracy of 79,3%) achieved with a OW size of 30 s and 300 s, respectively. Additionally, the authors also provide an interesting evaluation of the inconsistencies between the results obtained by similar feature selection methods in different datasets with similar features. Although being unclear which are the exact causes for those inconsistencies, some insights are left about some aspects that can trigger them. The use of different classification criteria, changes in the measurement ranges due to differences in animals' physiognomy or slightly different placement of the devices, are some practical arrangements that may influence the feature selection procedure.

With the aim of developing a device capable of carrying out real-time monitoring of sheep for allowing a precocious detection of lameness situations[4], two complementary strategies were followed. In the work of Walton *et al.* [195], the authors discuss the most suitable choices for the sensor position, sampling frequency and OW size. With the results provided by this study, a subsequent work [196] analyses the importance of different features in this kind of use cases and evaluates the behaviour of several ML algorithms in the classification of three sheep behaviours: lying, standing and walking.

Regarding the experimental set-up, both works resort in a device composed of a microcontroller, a flash memory, a Low-Power Wide-Area (LPWA) radio, although it was not used since the data was stored in the memory of the device, an integrated accelerometer and gyroscope, and a battery. This device was either attached to the ear (ear tag) or to the neck of the animal (collar), and the tests were performed using 6 sheep during 8 days. Each day consisted in a single test, being the data gathered while the device did not run out of battery or memory. To enable the classification of the data gathered, animals carrying the monitoring device were video-recorded during 2 hours per testing day. The videos were analysed and classified considering frames of 1 s independently of the OW chosen. Then, to permit the discretization of the classification label within the OWs, all samples were considered, and the most predominant one was selected[5]. The feature transformation process was similar in both works. Firstly, the magnitude and the rate of change of the resulting magnitude were calculated for both accelerometer and gyroscope measurements. Then, eleven features associated to those measurements were calculated (mean, standard deviation, kurtosis, minimum, maximum, interquartile range, signal area, absolute signal area, number of zero crossings, dominant frequency and spectral entropy), compounding a total of 44 features per sample. From here, the works diverged, and thus are discussed separately.

Following the described procedure, Walton *et al.* [195] tested different sampling rates (8 Hz, 16 Hz and 32 Hz) and OW sizes (3 s,5 s and 7 s), both for ear tag and collar, resulting thus in 18 different tests. In a nutshell, resorting in a RDF, the best overall accuracy (95%) was obtained with a frequency of 32 Hz, for both 5 s and 7 s OW sizes, and for both ear tag and collar. Contrarily, the lowest overall accuracy (89%) was obtained for the combinations ear tag-16 Hz-3 s, and collar-8 Hz-3 s. Even if no relevant differences were identified when using a ear tag and a collar, there are two additional interesting conclusions regarding the OW sizes and the sampling rates. The first is that the choice of the OW

[4]Lameness is a common locomotive dysfunction that can be registered in different kind of animals, including sheep. It is one of the most critical issues felt by livestock producers, causing relevant losses in the sector.

[5]In fact, if all classification samples within the same OW were equal, that label was selected and labelled as *non-mixed.* When not all samples agreed, the most predominant was selected and labelled as *mixed.* However, the authors do not clarify what was their purpose for this labelling procedure. Thus, to simplify our analysis, we simply assume that the predominant classification label was chosen.

duration depends on the goal of the test. Shorter OW are generally better since the data variation is smaller and there is a lower probability of being registered different behaviours in the same OW; longer OW are typically better when complex behaviours are intended to be detected since more data is available. The second regards an expected increase of the performance when increasing the sampling frequency. Notwithstanding, a trade-off between the performance and the power consumption of the devices shall be attained. For instance, in this particular case, an increasing of less than 5% in the overall accuracy (from a frequency of 16 Hz to 32 Hz) implies a reduction of 50% of the device's autonomy.

Fixing the sampling rate at 16 Hz and the OW duration at 7 s, as suggested by Walton *et al.* [195], the work of Mansbridge *et al.* [196] evaluates the effects of the feature selection procedure in the global accuracy using different ML algorithms, namely, RDF, SVM, kNN and Adaptive Boosting (AdaBoost), both for ear tag and collar deployments. The features used were exactly the same used in [195], *i.e.*, a total of 44 features. The RReliefF algorithm was used for ranking those features, resulting in a ordered list of features, both for the ear tag and collar scenarios. Comparing both lists, the dominant frequency and the number of zero crossings of accelerometer measurements are the unique features that present the same ranking, both for the ear tag and collar (first and third places, respectively). The remaining rankings differ, although some tendencies may be observed. For instance, the top three features are related with the accelerometer, which may indicate that the rough classification is done primarily resorting on accelerometer measurements. Contrarily, from the fourth to the fifteenth positions, the majority of the features are related with gyroscope measurements (11/12 for the ear tag and 10/12 for the collar), which may reveal a significant importance of the gyroscope measurements for a more precise tunning in the classification procedures. Concerning the performance of the classification algorithms, RDF clearly outperformed the remaining ones when the number of features is greater than three, and hence are the focus of the authors analysis. Also using RDF, the overall accuracy of the collar use case was higher, independently of the number of features used. However the impact of the number of features used is not linear. In fact, for the collar case, the overall accuracy reaches 87% with a total of 5 features, only 5% less than the best overall accuracy obtained (with a total of 39 features). Thus, for real-time deployments, a trade-off between the number of features and the overall accuracy is stated as very relevant since computing more features requires more computational effort, resulting in higher energy consumptions.

The works presented so far only distinguished three different behaviours, which may be considered insufficient to characterize the natural behaviour of sheep. Hence, the work of Barwick *et al.* [194] aimed at evaluating the effects of three different mounting places of an accelerometer in the classification of four different sheep states, namely, grazing, walking, standing and lying. The three mounting places were an ear tag, a collar and a leg sensor, as illustrated in Figure 3.2. The experimental set-up was composed of devices operating at a frequency of 12 Hz, online tests done in five different sheep and the OW duration considered was 10 s. The data pre-processing consisted in removing all observations whose associated video frames were related with unknown and transitional states as well as the computation of 14 different features - average, maximum and minimum (for each axis) and the movement variation, signal magnitude area, average intensity, entropy and energy. The feature selection was based on the *randomForest* and *varSelRF* libraries of R, being selected the three best ranked features for each type of mounting device. The QDA classification algorithm was then applied to the three use cases, being the results and conclusions slightly different.

The ear tag solution is granted by the authors as being the best solution due to presenting the best accuracy for the grazing, standing and walking behaviours. However, at the same time, it is referred that the lying behaviour could not be evaluated by the ear tag sensor. Consequently, it is not fair to state that this sensor deployment presents, in fact, the best results. For instance, it is expected some classification issues for distinguishing the lying and the standing behaviours as it happens with the collar deployment use case. Despite this clear limitation, the work supports the idea that the sensor deployment has a relevant impact in behaviours accuracies. The ear tag deployment allows the distinction of grazing, standing and walking behaviours with high accuracy (above 93%). The collar deployment reaches high accuracies in detecting grazing, lying and walking behaviours (above 92%), but it has a poor performance in classifying the standing behaviour, since there are movement patterns shared with the grazing and lying behaviours. Finally, the leg deployment presents very high accuracies in detecting the lying and standing behaviours (100%), a relatively high accuracy in detecting the grazing behaviour (91%), but it yields a bad performance in detecting the standing behaviour since is it jumbled with the grazing behaviour.

Albeit being older than remaining works, the work presented by Marais *et al.* [191] extended the sheep behaviour analysis to 5 states, particularly, lying, standing, walking, running and grazing. The data gathering process was responsibility of a device composed of an accelerometer operating at a frequency of 100 Hz, a SD card for storing the data, a microcontroller and a battery. The tests last 3 days and were repeated with 5 different sheep. The OW duration was set to 5.12 s and a greedy selection procedure was selected to rank the 31 different features calculated: mean, standard deviation, variance, skewness, kurtosis, maximum, minimum, energy, frequency-domain entropy and pairwise correlation between the axis - computed for the thee axis; and the Average Signal Magnitude Vector (ASMV).

The LDA and QDA algorithms were chosen to classify the supervised behaviours, being obtained global accuracies of 87% and 90%, respectively. The individual analysis of classifiers' performance

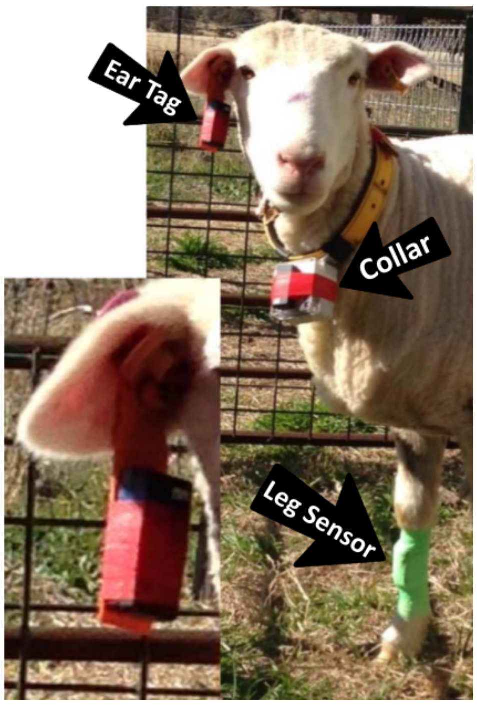


Figure 3.2: Distinction between a collar, an ear tag and a leg sensor [194].

highlighted a higher difficulty in distinguishing the grazing and lying behaviours, with missclassifications around 33%. On the other hand, the running behaviour presented the best accuracies. Contrarily to the works presented before, the classification procedure was not based in the visualization of videos but rather on a manual logging procedure. This precludes a precise and thin classification of the observations, as it is possible with videos. For instance, in this particular work, the human operator only registered the timestamps when the changes in the behaviour were evident, excluding thus, moments of transition and behaviours of short duration, just to cite a few.

Also considering the same 5 states tackled in Marais *et al.* [191], the work presented by Alvarenga *et al.* [192] explored the use of DTs to predict those 5 behaviours. The data was gathered through accelerometers placed under the sheep's jaw, operating at three different frequencies - 5 Hz, 10 Hz and 25 Hz - and considering different OW durations - 3 s, 5 s and 10 s. Nonetheless, within the goals defined for this study, the evaluation of the effects of using different sampling rates was not included. Thus, besides having devices gathering data with different frequencies, all data were merged together. For each OW duration scenario, the mean, standard deviation, minimum and maximum of the following features were computed (composing a set of 44 features): raw values from axis X, Y and Z, Signal Magnitude Area (SMA), Signal Vector Magnitude, MV, energy, entropy, pitch, roll and inclination. Although the wide set of features, feature selection techniques were used to select the 5 most important features for the present dataset. For that, the authors used the *randomForest* and *varSelRF* packages available on R, being the results, *i.e.*, the top rate features, equal for the three study cases: X-axis values, inclination, pitch, MV and Z-axis values.

Applying DT, the overall accuracies varied from 82.9% for OWs of 3 s, to 85.5% for 5 s and to 83.4% for 10 s. However, a deeper analysis on the results obtained allows additional important conclusions. Firstly, and although not tackled in the authors' discussion, the dataset is unbalanced. Particularly, the running and walking behaviours have only a few samples comparing to the remaining states (around 1% and 8% respectively). Then, analysing the different metrics evaluated, the highest precision and specificity for the grazing behaviour was achieved with a 5 s OW while for the running and walking behaviours those metrics were higher for the 10 s OW. Finally, the *Kappa* values also suggested the 5 s OW as the best option for classifying the 5 behaviours.

Wrapping up, from the reported works that use ML algorithm to predict sheep behaviours, the ones that present lower processing demands have limited their analysis to a low number of states. Contrarily, the works that widen their analysis to more states, exploited algorithms that are computationally more demanding and hence not suitable to be employed in low-power microcontrollers. The exception is the work of Alvarenga *et al.* [192] that explored the use of DT for differentiating 5 different states but, as all the other works, it did not distinguished the *Infracting* state, required to allow the detection of sheep's behaviours in vineyards.

Another relevant limitation across all the works regards the classification procedure and the definition of the OW durations. On the one hand, when classifying the gathered data within the context of the OW duration, the prevailing state is commonly taken as the classified state. Although it can be considered the most correct way of doing it, such approach may hide transitional states that are neither considered in the classification or in the modeling. On the other hand, albeit the majority of works suggest that OW of 5 s or 7 s are suitable for detecting sheep behaviour, that durations may not be suitable for real-time detection since the model requires a lot of data.

Also, most of the analysed works employ offline data collection strategies, storing the sensing data gathered by accelerometers in the memory of devices (for instance in micro SD cards), that are afterwards downloaded for further analysis resorting on powerful computers and algorithms. Although a few solutions exhibit some real-time features (*e.g.* transmitting wirelessly the sensor data to computational platform) or present models that could be eventually transposed to constrained devices, none of the works explored the feasibility of such approach.

3.2 Animal conditioning

Animal conditioning is particularly popular for confining animals within pre-defined grazing parcels. Typically, farmers use physical ground-based fences (Figure 3.3), either wood-based or barbed wire-based. However, despite effective, these methods are costly, inflexible and laborious [203]. Their installation is an ardours task, requiring constant maintenance and whenever new parcels are used, all the process needs to be repeated.

Figure 3.3: Example of a ground fence used to contain sheep.

To mitigate traditional fences' issues, electric fences started to be used [204]. Initially, they were used together with physical fences to avoid damages caused by animal aggressive behaviours. Then, electrical fences started to be used as a standalone method for bounding grazing areas. Although still requiring individual placement by humans, the labour involved is drastically smaller. Furthermore, animals tend to recognize the presence of electrified wires, which enables them to avoid such structures. However, electric fences require a electrical source, typically a battery, that needs to be recharged and protected against weather conditions or other external threats. Furthermore, they are not suitable to be used during wet days or in places with dense vegetation, since these conditions may lead to accidents due to electric conduction.

The concept of virtual fences emerged to provide the flexibility not offered by traditional solutions. A virtual fence is composed of an electronic system that allows the definition of fence boundaries [203] in a programmatic way. This fencing scheme does not require any physical barrier [205], being used a stimulus instead (*e.g.* visual, olfactory, sound or electrostatic). As virtual fences do not depend on physical barriers, they have lower installation costs [206] and allow a much easier redefinition of grazing areas, thus contributing for an improved pasture management [205], [207]. Moreover, they allow a better sanitary control since contaminated areas can be easily and quickly isolated.

The first patent of a virtual fence mechanism was published in 1971 and consists in a device coupled to the animal working as a receiver of electromagnetic waves that are generated by a hidden electric cable. The patent was developed for domestic animals and applies sound and electrostatic stimuli. This solution was commercialized with the name Invisible Fence [208] and it is still available in the market. Even if it was conceived for domestic animals, the solution was also tested, with relative success, in goats [209] and cattle [210]. Notwithstanding, the installation process is onerous and requires much effort to install the underground cable. Additionally, it is impractical to use this method in extensive areas as well as in mountainous terrains, areas where animals often graze.

Besides using the electromagnetic field generated by an electric conductor to define the boundaries of the virtual fence, two other methods exist. On the one hand, GPS coordinates may be used and compared with a polygon defined by a group of coordinates [211], [212]. Even if its efficiency is proven [167], [211], [213], the cost and high power consumption dispelled this technology from the virtual fencing solutions. On the other hand, the magnitude of electromagnetic signals may also be used to evaluate the distance of the animal to signal sources [214], [215]. Regarding this latter group, to the best of our knowledge, there aren't works evaluating its effectiveness, which may suggest that there is still a long way towards a viable solution. In fact, while high levels of effectiveness are not reached using highly precise localization mechanism as GPS, it is not worthy evaluating strength signal-based virtual fencing mechanisms, that have lower precision. However, it is necessary to perform research on this area. Therefore, the system described in this thesis, besides not implementing a fencing system, includes on its requirements support for it.

Besides the method for defining the virtual boundary, a virtual fence requires a combination of cues to compel animals to revert their behaviour. These cues can include different sources such as visual, auditory, olfactory, vibration or electrostatic stimulus [216].

Visual cues were tested in bovines [216] and allowed to conclude that animals have the ability to associate cues with specific areas, allowing the control of grazing areas through visual cues. Furthermore, Cibils *et al.* [217] demonstrated that cattle are capable of associating visual cues with electrostatic stimuli. As such, after a learning phase, they are able to revert their behaviour when facing exclusively visual cues, even if cues' location changed. However it is complicated to install efficient sources of visual cues and thus they are not typically used.

The use of audio cues for conditioning animals' behaviour was evaluated by Umstatter *et al.* [218]. Speakers placed with 10 meters of distance between each other were set to emit high frequency sounds in order to disturb cattle. This study proved that audio cues are relevant to implement a virtual fence but not enough to build an effective solution. A similar work, consisting of a auricular speaker, placed individually in each animal, resulted in the publication of a patent named *Ear-A-Round Equipment Platform for Animals* [219]. This device, also tested in cattle, proved to be efficient to contain animals in a virtual fence but only after some training to enable the association between the audio cues and the virtual fence.

Electrostatic stimulation is reported to be the most often used and efficient conditioning solution in the literature [205]. In short, this type of stimulation consists in the emission of high voltage electric discharges with very low currents [207], [209], [220], [221], through a pair of electrodes incorporated in collars that are placed in contact with animal's skin. Within this group of works, *Fay et al.* [209] also showed that the gregarious instinct of goats tend to keep all animals together, thus being effective also in goats that do not carry collars.

However, the combination of different stimuli, for instance, sound, vibrations and electrostatic, to confine cattle in a virtual fence is more consensual in the literature. The work of Bishop *et al.* [222] is an example. A collar producing a 6 kV electrostatic stimulus after audio and vibration cues, proved that the combination of several stimuli contributes considerably for the animals learning process.

The effect of sound and electrostatic stimuli was also evaluated in sheep [223]. The boundaries of the virtual fence mechanism were based in an electromagnetic emitter and some food and social challenges were included, being inserted, respectively, food with higher nutritional value and free animal grazing, in the forbidden areas. Furthermore, a training process evaluation was performed, being compared results of trained and not trained animals. Results showed that trained animals demonstrate a bigger respect to the conditioning process and that the social challenges had bigger impact than the food challenges. Particularly, the work reports several situations where animals not trained won the electrostatic stimulus and joined the group of free sheep.

More recently, and still regarding sheep, Brunberg *et.al* [224], [225] performed two field tests using the NoFence [161] system on sheep. This popular commercial solution for animal localization and virtual fencing, still available on the market, comprises the use of a collar that emits audio cues (frequencies from 2 kHz up to 4.2 kHz during 4 s) followed by an electrostatic stimuli (4 kV, 0.2 s, 0.1 J). A maximum of 5 electrostatic stimuli could be applied. In [224], from a set of 24 sheep, only 9 have successfully learned how to avoid virtual fence boundaries. Furthermore, it reported special challenges when changing the place of the virtual fence. In [225], similar tests, but with sheep accompanied by their ewes, needed to be interrupted since no acceptable results were being obtained. As such, the authors concluded that there are still a lot of challenges to overcome to enable the use of virtual fencing mechanisms.

The results presented by Marini *et al.* [226] when testing a virtual fence based on a dog training collar were more promising, with less than 19% of sheep receiving an electrostatic stimulus after an audio cue. In average, sheep required 8 interactions with the fence to learn the association between the cues. The audio cue comprised an 78-80 dB sound at a frequency of 2.7 kHz, while the electrostatic stimuli last for 20 μs with a current of 42 mA. This study also showed that sheep maintain a certain level of memory, since, after removing the virtual fence, they took 30 minutes to re-enter to the forbidden zone. In a extension to this work, the same authors identified several challenges when applying virtual fences to sheep [227]. Particularly, they state that their ability to get trained to recognize virtual fences it not yet clear. Particularly, they identify challenges on the effectiveness of electrostatic stimuli due to sheep's wool and reported significant differences between sheep.

The effects of this kind of solutions on the animal well-being are ethically relevant and should be carefully addressed. In Lee *et al.* [228], the authors defend that if animals successfully associate audio cues to electrostatic stimuli, it is possible to offer a solution that is ethically similar to electric fences. In fact, there are studies in the literature tackling this issue, being based essentially in health indicators as the weight evolution, the concentration of specific hormones or the heart and breath rates. One example of these studies is a test effectuated in Beagles [229], where three groups of tests were arranged: a group being stimulated when touching a specific object; a group being stimulated when not obeying to instructions given by a trainer; and finally a group being stimulated randomly without any special context. Results revealed a substantial increase of cortisol on the second and third groups, particularly when the timing of stimulation was not adequate. On the first group, it was not identified any meaningful increase on the hormones' concentration. Thereupon, it is vital to enable an association between a stimulus and a specific behaviour to promote the animal's learning process.

The application of electrostatic stimuli causes a small discomfort on the animal, thought without being painful. A collar remotely controlled and capable of generating pulses of 650 V tested in cattle [230], showed that stimuli causes the reaction of animals and associative learning. To evaluate the effects of these stimuli on the well-being of animals, both the variation of the heart rate and the quantity of cortisol and b-endorphin (hormones often associated to animal stress) were measured on two testing groups, one that did not receive any stimulation (the control group) and other receiving 2 to 3 electrostatic pulses. The results revealed an increasing heart rate and concentration of stress-related hormones on animals receiving stimuli comparing to the control group. However, in order to understand if these consequences are exclusively related with electrostatic stimuli, the effects of other common livestock operations were evaluated, as for example the weighing process or the immobilization for treatments. It was shown that well-being metrics are not significantly different comparing situations of stimuli application to common livestock operation. Thus, the authors concluded that electrostatic stimulation does not introduce any special discomfort to animals.

The learning process plays a relevant role in the minimization of the number of electrostatic stimuli. Consequently, the combination of multiple cues facilitates the referred process [222], [231], giving the opportunity to the animal to revert its behaviour before receiving an electrostatic stimulus. Inclusively, it is advocated that this kind of conjugation performs better than using electric fences or shepherds since the learning process allows animals to adapt their behaviour accordingly.

To preserve animal's integrity and well-being, the Electronic Collar Manufacturer's Association was founded and a document created, defining good practices in the implementation of electronic collars [232].

3.3 Summary and discussion

Monitoring sheep's location and, more importantly, their behaviour, is a key element for enabling their use for weed control. Therefore, a literature survey was performed to perceive what is the state-of-the-art on animal monitoring, what techniques are commonly used, in which applications, with what purposes and for which animals.

In fact, animal monitoring is not a new topic, being addressed for many years in difference contexts. The most popular context is the use of GPS devices for tracking animal location. Although cattle is the animal group frequently addressed, namely because of its high commercial value, also other groups are reported, including sheep. These GPS-based solutions differ on optimization mechanisms to minimize GPS issues. For instance, some include accelerometers to improve GPS accuracy or to enable decreasing GPS duty-cycle. Others include energy harvesting mechanisms to improve device's autonomy.

However, and besides their accuracy issues, it is consensual that RF signal strength based localization mechanism would introduce gains on device's cost and autonomy, since no additional hardware would be required on systems where there is already RF hardware to enable communications to occur.

As such, some experiments have been made to assess the feasibility of using RSSI for animal localization. However, there is a space for research on such area, particularly on gregarious animals such as sheep and goats.

As a complement to animal localization, the virtual fencing concept arose. Also here, cattle is the animal group that deserves more attention, although more recently several works came up with some experiments made with sheep. It still is unclear and non-consensual if sheep are capable of respecting virtual fences, particularly due to the difficulties in recognizing the virtual fence boundary and due to the gregarious behaviour of sheep that may clash with the conditioning mechanism. Even so, it is commonly accepted that animals, including sheep, are capable of learning to associate cues with certain penalizations and avoid those penalizations. It is also shown that they are capable of associating stimulus with certain landmarks. However, this training process is not straightforward and depends on several factors such as animal breed, groups' sizes and individual's characteristics.

Besides monitoring animal location, also animal behaviour monitoring is gaining interest. The main purposes are to control health parameters and detect behaviour patterns such that appropriate managing procedures can be taken. In this case, the use o ML techniques is the common approach followed. The processing tasks occur mainly offline in central processing units with processing capabilities that portable sensor-enabled devices do not have. Several behaviours in different animals, including sheep, are addressed in the literature. However, none of them enables the identification of infracting situations in vineyards and that corresponds to actions that may damage the vines or their fruits. Additionally, none of the works tested behaviour monitoring mechanisms running locally on collars or similar devices neither include a mechanism to conditioning sheep's posture when the infracting state is detected.

Table 3.1 summarizes the most relevant solutions surveyed. In summary, there are several solutions that tackle, even if partially, issues that we aim to solve in the scope of this thesis. However, none of them is capable of detecting and conditioning sheep's posture while grazing, *i.e.*, none of them implement a posture control mechanism.

Table 3.1: Summary of the most relevant scientific works and commercial platforms on animal monitoring.

Feature/Solution	**Localization Monitoring**					**Activiy Monitoring**			
	Nofence [161]	**eShepherd** [159]	**Digitanimal** [160]	**Huircan** ***et al.***[170]	**Nadimi** ***et al.*** [233]	**Cowlar** [198]	**CowScout** [197]	**Dutta** ***et al.*** [185]	**Alvarenga** ***et al.*** [192]
Animals	goats	cattle	several	sheep	sheep	cattle	cattle	cattle	sheep
Data gathering	yes	yes	yes	yes	yes	yes	yes	yes	yes
Real-Time Data	yes	yes	yes	yes	yes	yes	yes	no	no
Localization	GPS	GPS	GPS	RSSI	GPS	no	no	no	no
Virtual fence	yes	yes	no	no	no	no	no	no	no
Activity Monitoring	no	no	no	no	yes	yes	yes	yes	yes
Posture control	no	no	no	no	no	no	no	no	no

CHAPTER 4

An IoT-based Architecture for Intelligent Farming

The state-of-the-art analysis revealed the unavailability, on the one hand, of solutions capable of meeting the demands of the SheepIT project, and, on the other hand, of a suitable IoT architecture capable of supporting the accompanying requirements. Therefore, this chapter is devoted to the presentation of the system architecture devised for SheepIT, its components and operation.
We start by defining the functional requirements pursued. These functional requirements are the support of both the overall system architecture and IoT-based communication stack. Moreover, system components are detailed, namely the mobile nodes, called collars, that are carried by sheep and are composed of a set of sensors; static nodes, named beacons, that ensure radio geographical coverage and communications, as well as enable localization services; and information aggregation nodes, the gateways, that besides data aggregation, provide an optional interface with the exterior, namely a computational platform that can be hosted in public or private clouds. A proposal for the computational platform is also presented, despite not being the cornerstone of this thesis. The focus is on the communication stack design, with special emphasis on the MAC Layer due to its importance towards a energy wise system.

4.1 Functional requirements

The main motivation for this thesis was the need of developing an autonomous solution for keeping sheep inside delimited areas of pasture and control their posture while in there. Such system would make possible to use sheep as a cheap and environmental-friendly weed control mechanism, while keeping vines safe. However, developing such a system requires a set of functional requirements that preclude the use of existing solutions. These requirements were defined considering both the goals of the SheepIT project and associated partners knowledge. They are as follows:

- **Posture control:** it represents a key component of the solution. Briefly, it shall enable monitoring and identifying sheep's posture and apply corrective stimuli when undesirable postures are detected. However, this task in not straightforward, on the one hand due to animals dynamism and unpredictability and, on the other hand, due to the limited processing capabilities of the electronic devices to be carried by sheep;
- **Posture detection:** to enable the implementation of a posture control mechanism, it is critical to continuously monitor sheep behaviour. In particular, the neck and head position must be estimated in order to detect if they are feeding from weeds (desired behaviour) or from the

branches or fruits (undesired behaviour - the *Infracting* state). The head position can be estimated by measuring the distance between the neck and the ground, and by measuring the neck inclination. However, reducing the posture detection to a binary classification problem (infracting and non-infracting) has revealed to be insufficient. Henceforth, the posture detection shall be capable of detecting with an overall accuracy of 90% (the state-of-the art mean value) and on real-time the different and most common animal behavioural states, such as *Eating*, *Resting*, *Running* and *Moving*, plus a novel one, the *Infracting* state;

- **Posture Conditioning:** besides being a component of the posture control mechanism, its design shall allow a configuration of several characteristics such that the most suitable combination of stimuli could be arranged by livestock experts or veterinarians. Having the state-of-the-art (Chapter 3) as basis, a combination of different stimuli, particularly audio cues followed by small electrostatic discharges, is the most appropriate solution. Nevertheless, stimulus duration, stimulus sequence, stimulus duty-cycle and start and stop conditions, depend on the animal breed and shall be carefully designed by livestock experts to ensure animals' well-being;
- **Localization and virtual fencing enabler:** the system shall support the implementation of a localization service. In scenarios with a potentially high number of mobile nodes, as the present one, the high cost and energy consumption of GPS devices preclude its use. Hence, the system shall support the implementation of an RSSI-range-based localization mechanism. It is not intended to be a high precision mechanism but shall enable users to have a regular and approximate animal location, as well as have access to a rough tracking along the day. This information can be henceforth analysed, for instance, for identifying pasture preferences, possible illnesses, animal theft, just to cite a few. Furthermore, albeit not addressed in the scope of this thesis, having the support for the implementation of a virtual fence mechanism was defined as a requirement for the system's design since it is a potentially relevant feature in a near future. Project partners pointed out to 6 seconds as the maximum interval time between two consecutive fencing checks (from now on entitled Localization and Fencing Periodicity (LFP)), although it shall be configurable and adjustable according to scenarios needs.
- **Data Collection:** the system shall collect data about each one of the animals (*e.g.* to detect health conditions) as well as information about the herd (*e.g.* to detect predator attacks) for further processing and storage. For this reason, the system shall include an interface to transfer data to a central computer or to a cloud service. New data shall be provided to the user (the farmer) with a certain periodicity, hereinafter referred to as Monitoring Periodicity (MP). This value shall be configurable, although project partners pointed out to 80 seconds as the maximum value;
- **Size, weight and autonomy:** this triple requirement is crucial in every IoT system. On this particular case, sheep are relatively small animals that graze for several hours. Thus, the device to be carried by sheep shall be comfortable, thus constrained in terms of size and weight. As reference, it is desirable to have a collar dimension similar to the Digitanimal [160] and to the noFence [161] collars, since they are important commercial references in the area. Moreover, and since this solution is to be used by farmers, it is important to minimize the number of charging procedures. Therefore, the autonomy shall be at least two weeks (a reference provided by project partners), although an autonomy of 4 months shall be pursued (as suggested in [171] when presenting an electronic shepherd for sheep grazing in mountains);

- **Customization, expandability and scalability:** although being designed considering the particular use case of the SheepIT project, the system shall be able to be used by similar solutions within the farming monitoring area. Likewise, it shall support the integration of new sensors and shall be scalable, allowing a reasonable amount of mobile nodes. Partners defined 1000 sheep as the upper bound size of a herd ($N_{maxCollars}$), although a recent census in Ireland [234] identified the average size of a herd as being around 100 sheep/goats. Furthermore, when presenting the number of herds within certain herd sizes, the maximum group was >500, with a representation of 2.22%, thus below the 1000 sheep. Even though, we maintain the 1000 collars as the maximum number of collars, even if lower values are expected to be found in real use case scenarios;
- **Area coverage:** the area to be covered by the system impacts on the system configuration. Therefore, it shall be considered during system design. In this case, independently of the vineyard size, the partners believe that the system would operate in parcels of a maximum of 5 ha, enabling the implementation of rotational grazing strategies[1];
- **Communications infrastructure:** besides the intrinsic requirements associated to an IoT communication infrastructure (introduced in Chapter 1), this aspect is highly influenced by all the aforementioned functional requirements. It shall enable the communication of hundreds of nodes in a region of a few hectares; it shall collect data periodically and relay it to one or more aggregating devices for further processing; it shall support the implementation of a RSSI-range-based localization mechanism to avoid the use of GPS, which requires periodic transmission of data; it shall support the implementation of a virtual fencing mechanism; it shall use energy-aware radio technologies and protocols since all periodic data transmitted can result in a considerable amount of data; and finally, all these periodic communications require some sort of node-synchronization to allow nodes to minimize the time on which they are turned on, and consequently, minimize the peak and average energy consumption, by means of a proper activity scheduling.

The following sections present the overall system architecture, the communication stack that supports it, and the design of the devices built to meet the requirements.

4.2 Overall system architecture

The main purpose of the system herein proposed is to grant farmers with an intelligent farming solution capable of monitoring and conditioning sheep, allowing them to weed vineyards or similar cultures (*e.g.* orchards) without threatening the cultures. However, besides being specially designed to support this particular use case, the system architecture is designed to support expandability, *i.e.*, to support additional services, for instance, monitoring plants and soil parameters. For the sake of clearness, we empathize the description of system components and their operation in the context of the main application (monitoring and conditioning sheep's posture), although readers should keep in mind the inherent solution expandability.

[1]Rotational grazing corresponds to sectioned grazing parcels that are grazed one at time. While some parcels are being grazed, other are resting, enabling grass and nutrients renewal as well as it allows to avoid health problems due to the accumulation of bacteria and virus [235].

The requirements imposed by the uncertainty of animal's location, their extremely gregarious but unpredictable behaviour, and the effects of the terrain relief in the radio coverage, specially in Douro's region vineyards, led to the design of a new solution, from the IoT devices to be deployed, up to the IoT communication network. Figure 4.1 illustrates the overall system architecture. The system architecture follows a typical IoT architecture, being composed of two main segments conveniently interconnected. The first is a WSN that implements all critical local functions, such as animal behaviour monitoring and conditioning (and additional monitoring services, if required). Additionally, it also enables sensor data gathering that is then forwarded to the user, either through a remote intermediate (the Computacional Plataform (CP)) or through a local access. The second segment concerns a CP, hosted in public or private clouds, that has as functions message brokering, data analysis, data storage and data representation, this latter one through appropriate user interfaces.

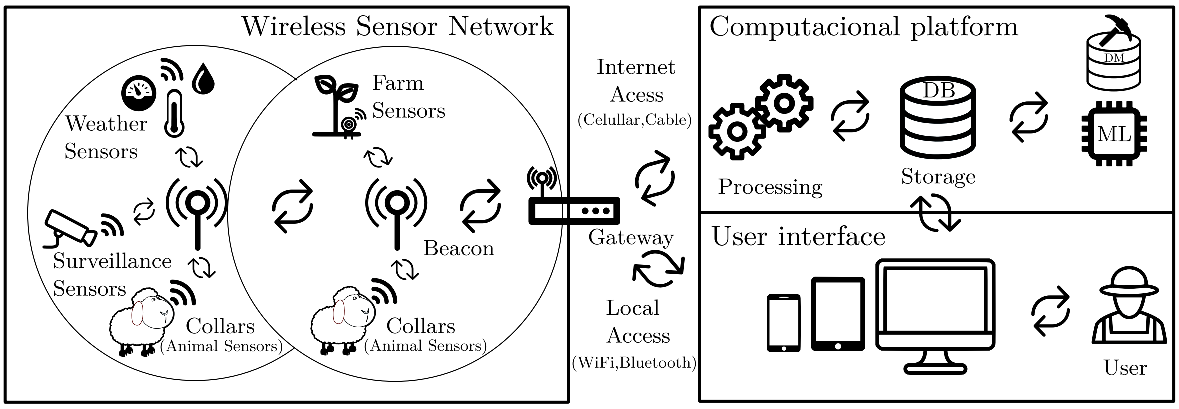


Figure 4.1: Overall system architecture.

Collars are the WSN mobile nodes, being carried by sheep. They contain the sensors and actuators necessary to implement the posture control mechanism: a microprocessor, for carrying out local computations; a battery, to allow portability; and a radio transceiver, to allow communications between collars and the remaining network infrastructure. The sensors, namely an ultrasonic sensor and an accelerometer, are the support of a real-time monitoring mechanism that allows the detection of different sheep behaviours, including their posture. On the other hand, actuators are triggered by a posture control mechanism through the application of appropriate stimuli. The posture control mechanism includes all the logic that grants collars with the capacity to autonomously detect when sheep adopt unwanted postures. A detailed description of collars composition is given in Section 4.3.1.

Grazing areas can be relatively large, exceeding the coverage of a single radio transceiver. Moreover, vineyards, particularity the ones of Douro's region, are typically located in remote locations, with aggressive and irregular steep slopes. Therefore, the communication infrastructure also contemplates **beacon** nodes, which receive data from collars and relay them, eventually through other beacons, to an aggregating gateway. Hence, beacons define the grazing perimeter, and that are necessary to ensure connectivity to enable the collection of real-time sheep's data. Additionally, they serve as landmarks for the RSSI-based localization mechanism and are also responsible for ensuring network synchronization. Beacons' full composition is specified in Section 4.3.2.

The WSN connects to the CP via one or more **gateways**. A gateway can be seen as an extended version of a beacon comprising, besides beacon functionalities, an additional Wide Area Network (WAN) interface (for instance a 3G or 4G interface) and a set of managing and monitoring features. As such, it allows a smooth integration between the non-IP network (the WSN) and the IP-based Internet, and includes local processing capabilities that allow detecting and reporting abnormal situations, either in animal behaviour (*e.g.* high number of infractions) or device operation (*e.g.* low battery); a local management procedure that allows farmers to consult the status and several configuration parameters of the system, without needing to have access to the Internet, ; and a technical management procedure to be used by technicians to perform advanced configurations on communications parameters or perform on-site debug. Further details on the gateway architecture and operation are give in Section 4.3.3.

Although the WSN operates autonomously, without any dependency on the CP, the CP is saw as a valuable additional component, particularly due to its data processing power. Thus, the system architecture contemplates a CP that may be hosted in the cloud or in private servers, and whose main tasks are to aggregate, process, analyse and store stream data, supporting also the implementation of appropriate user interfaces for data visualization. Even if at a first glance there is a superposition of functionalities between the gateway and the CP (both act as aggregating points and include processing capabilities), the goals and tools are disparate. While the gateway focuses in tasks that comprise low processing overhead and short time windows analysis, the CP allows the implementation of powerful DM and ML algorithms, which in turn enable the retrieval of long-term statistics, hidden patterns and tendencies. Relevant examples comprise the detection of animal health conditions, animal lameness, estrus cycles, sheep calving, just to cite a few. Additional details about the CP are given in Section 4.5.

Informally, the operation of the system can be summarized as follows:

- Collars, the mobile nodes carried by sheep, monitor and condition sheep;
- Collars periodically transmit, in a pre-determined and unique time slot, sensed data, both regarding animal behaviour and devices status;
- The transmission periodicity is a system parameter, statically defined *a priori* during system configuration;
- Beacons are static nodes, placed within or in the borders of the grazing parcel, which gather data from collars and retransmit them through the network until reaching the gateway;
- They also grant synchronization among all networking devices, ensuring that no concurrent transmissions occur, *i.e.* avoiding collisions during data transmission;
- Like collars, beacons also have a pre-configured periodic transmissions, enabling the implementation of a RSSI-based localization mechanism;
- Data gathered by collars is retransmitted through one or more beacons until being received by the gateway(s). The gateway(s) processes the data, monitors relevant parameters, triggers alarms if necessary, and prepares data to be sent to the CP;
- The gateway also enables a set of interactions with networking elements, providing local flexibility to the solution;
- The CP receives streams of data from one or more systems, parses and stores it in a structured way;
- DM and ML algorithms can be applied to data, adding extra value to the original gathered data;

- Finally, data is displayed to the user through a web user interface, such that it can be accessed anywhere and anytime, using a browser.

The aforementioned description is a simple overview of the system operation. As one can see, supporting such operation while complying with the requirements bestowed in Section 4.1, requires the design of appropriate devices and the implementation of a communication stack that, due to its alikeness to typical IoT systems, we refer to as an IoT-based communication stack.

4.3 Devices

Within the WSN, several devices may coexist, although in the scope of this thesis we are particularly interested in three type of devices: collars, associated to animal sensing and control; beacons and gateways. Their composition and operation are described in the following sections.

4.3.1 Collars

The collar is a keystone of the solution, particularly due to its functional features. Collars implement the monitoring and conditioning features, supporting the posture control mechanism so that sheep are persuaded to weed with a lower profile, *i.e.*, with a posture that could maintain safe the vines and the fruits. Furthermore, besides implementing these critical features, collars are also the most Size, Weight, Power and Cost (SWaP-C) constrained devices.

Figure 4.2 depicts the collar architecture. Besides the battery and switch modules, whose purpose is straightforward, four main block are identified, particularly:

- **Microcontroller:** it is the "brain" of the device, being responsible for interfacing with sensors, actuators and communications peripherals. It is also responsible for executing the monitoring and conditioning algorithms, as well as the communication scheme;
- **Communications:** besides executing local tasks, collars need to own communication capabilities. As such, they include a communication module that permits receiving and sending data, following a pre-determined communication scheme;
- **Sensors:** include the necessary sensing peripherals that sustain the monitoring mechanism, namely:
  - **Accelerometer:** two-fold information is extracted: *i)* static acceleration measurements that give information about neck's tilt. Through trigonometric manipulation, pitch, roll and yaw angles[2] may be computed [236]; *ii)* dynamic acceleration measurements allow the computation of the intensity of animals movement on each direction[3];
  - **Ultrasounds transducer:** used to measure the distance from sheep's neck to ground, which may be combined with accelerometer data for improving the detection mechanism.

[2]The navigation coordinate system was used as reference. This means that pitch corresponds to rotation around the Y-axis, *i.e.* around the East; roll around the X-axis, *i.e.* the around the North; and yaw around the Z-axis.

[3]The inclusion of a magnetometer device was also evaluated. Notwithstanding, although presenting a theoretical relevance for understanding sheep movement direction, in practice, it leads to very erratic data due to constant interference that causes constant inconsistent values and de-calibrations. That is the result of a wide range of natural and artificial obstacles as well as frequent aggressive movements by sheep.

- **Actuators:** an electromechanical audio transducer, to emit audio signals, and an electrostatic stimulation circuit, to apply electric discharges, are the actuators included in the collar, following existing approaches in the state-of-the-art. Both sensors and actuators are controlled by software, composing the posture control mechanism that is tackled in Chapter 6.

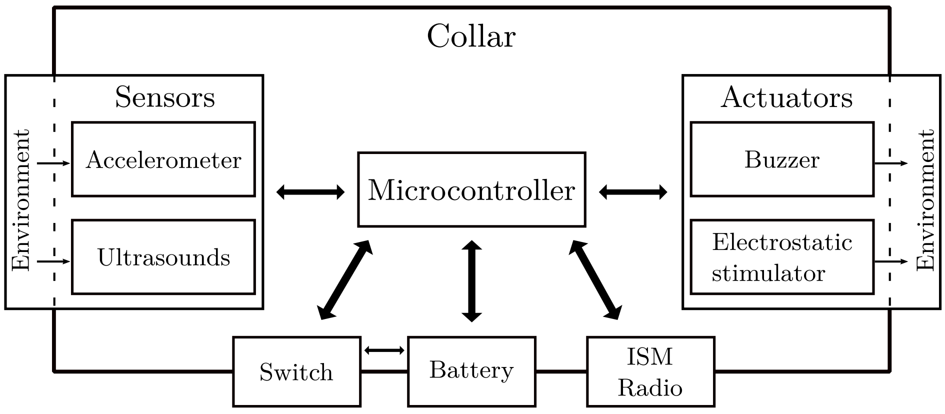


Figure 4.2: Collar internal modules.

4.3.2 Beacons

Beacons are fixed devices that define the grazing perimeter, as it is necessary to ensure connectivity to collars. They also serve as landmarks for enabling a RSSI-based localization system. Thus, when necessary, additional beacons can be disposed on the grazing area to create RF overlapping areas, with the objective of improving the accuracy of RSSI-based localization systems.

This way, it becomes clear that beacons' placement involves a trade-off between several factors, such as the total area to be covered, the topology of the terrain, the number of beacons available for the installation and the accuracy requirements for the localization system. Therefore, beacons placement is for now projected to be performed by experienced personal, although an automatic wizard for system installation is a valid and desired alternative.

Beacons share the same basic composition of a collar except the actuators module that is removed on beacons, and different sensors used on sensors module. This is particularly advantageous since it decreases the effort dedicated to hardware development. As illustrated in Figure 4.3, a beacon also includes a microcontroller, sensors and a communication module. The first controls all internal operation, interfacing with sensors and with the communication module. Regarding sensors, a GPS is included to support the implementation of a RSSI-based localization system.

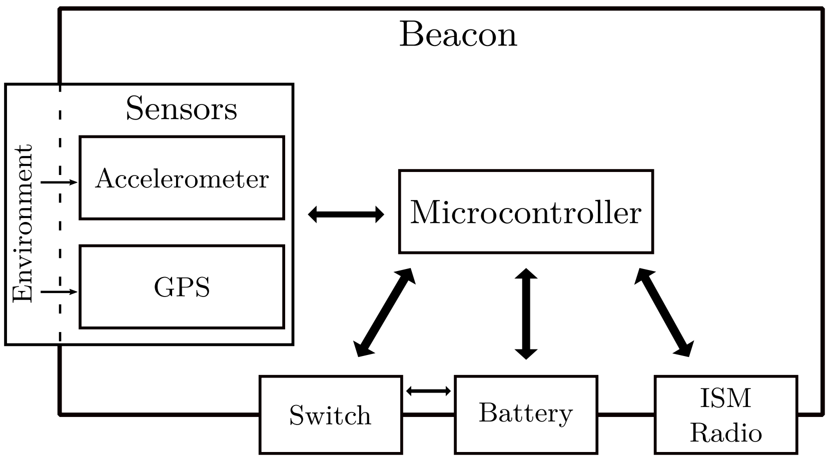


Figure 4.3: Beacon internal modules.

4.3.3 Gateway

A gateway allows WSN data to flow to the CP and vice and versa. In addition to the standard gateway functionality, which is essentially joining networks to allow devices to intercommunicate irrespectively of the network where they belong, the gateway supports additional services, developed to address the specific requirements of farming applications.

The most elemental functionality of the gateway is forwarding data between the WSN, the CP and the user interface. This activity requires minimal processing, consisting in decoding data frames coming from each one of the interfaces, placing the data in a local database and triggering the processes responsible for fetching the data and sending them to the target interface(s).

Many farms are located in remote places and, as such, permanent access to the Internet cannot be considered as granted. Moreover, there are some services that require low response times and, as such, it is essential to deploy them as close to the data sources as possible. Consequently, the gateway supports locally a few critical services, namely:

- **Alarms:** alarms should be raised as quickly as possible in the event of anomalous situations. The gateway collects data from all the animals, so it has a global view that allows it to correlate the received data, thus being a privileged place to detect abnormal situations. For example, if all animals suddenly exhibit a running behaviour, that may indicate that the flock may be under attack of a predator, and thus an alarm should be raised. A sheep that disappears from all the beacons can have escaped, and so an alarm should also be raised;
- **Local management:** to streamline the access to local data, a local web interface is desired. It allows performing on-site administration tasks (*e.g.*, defining the height at which animals are allowed to feed, if the restrictions are active or not, etc.) and consult real-time and historical state data. These operations shall be possible through a WiFi or Bluetooth enabled smartphone or similar mobile device, without requiring access to the CP's web interface;
- **Technical management:** the local web interface shall also enable technical interventions, such as tuning the communication parameters to the specific characteristics of the exploration (kind of terrain, dimension of the flock, etc.), as well as to allow system-level debug.

The number of gateway devices is expected to be small (just one, in most of the cases). Moreover, they are not expected to be moved frequently and are usually deployed in accessible places. Therefore, we assume that gateways are not subject to the same stringent power, size and energy consumption limitations that beacons, and particularly collars, face. Thus, gateways can use more powerful hardware (*e.g.* an embedded PC), both in terms of processing and storage capacity, which is essential to enable the realization of the services mentioned above. As illustrated in Figure 4.4, the gateway comprises Internet and WiFi/Bluetooth interfaces, made available through suitable network interface cards. The gateway also contains a beacon module, interconnected with the embedded PC via a serial interface. The beacon module ensures compliance with the communication scheme of the WSN. Combined, these interfaces abstract the communication services offered to the local services.

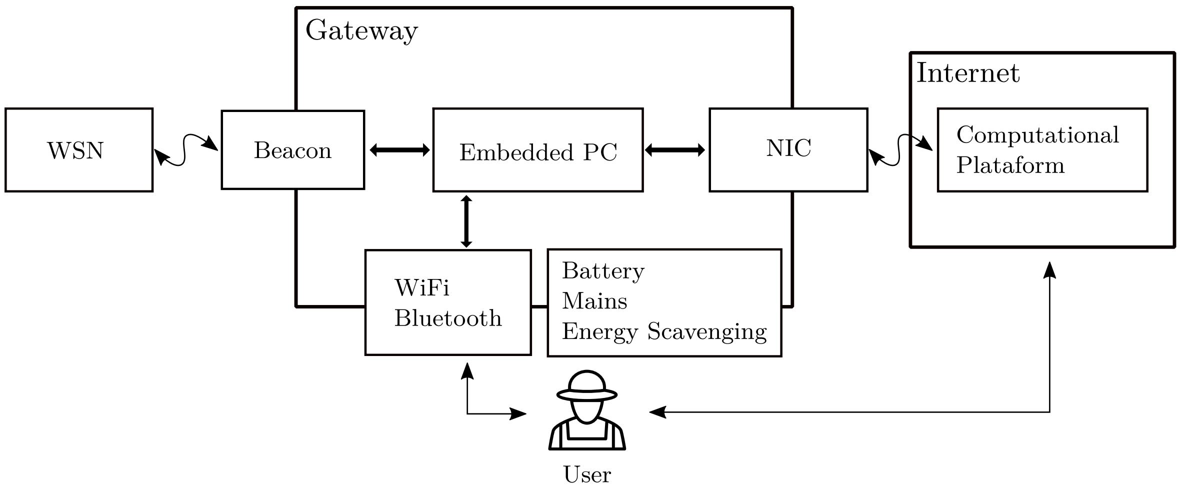


Figure 4.4: Gateway internal modules.

4.4 IoT-based communication stack

Aiming at satisfying the requirements identified in Section 4.1 and supporting the deployment of the system architecture and operation summarized in Section 4.2, the IoT-based communication stack shown in Figure 4.5 was designed. This stack is one of the pillars of this thesis, sustaining the responses to research questions 1 and 2. Despite having the SheepIT application in mind, the stack design was driven by more general goals, aggregating the traffic requirements of several applications related to farming management and supervision, thus justifying the title of the Chapter - *An IoT-based solution for Intelligent Farming.* Our approach follows a four-layer architecture (Physical Layer, MAC Layer, Transport Layer and Application Layer) since it is the one that enables a proper match between the requirements and each one of the layers. In general, there are evident similarities between the proposed stack and a typical IP-based IoT stack (see Section 2.1.1). Nevertheless, the enabling solutions within each of the layers are distinct, thus being tackled and justified one by one during the following sections.

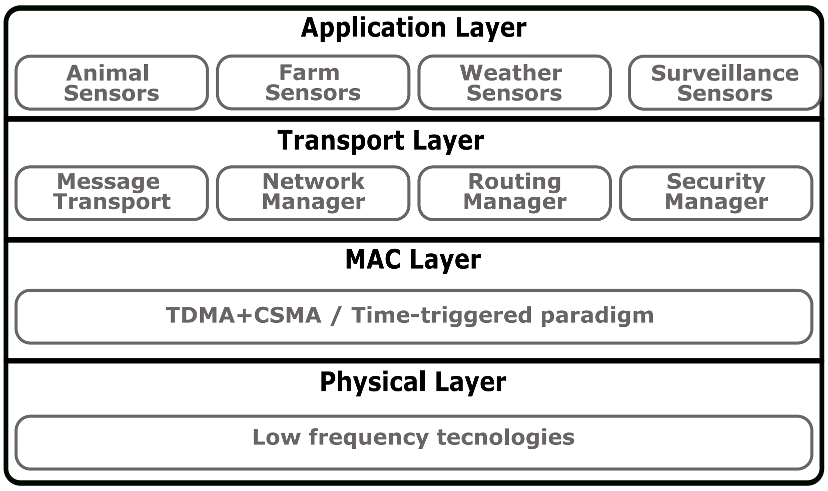


Figure 4.5: IoT communication stack for intelligent farming solutions.

4.4.1 Physical Layer

Wine-growing lands (and cultures similarly planted), mainly in Douro's Region (Portugal) where the terrain is very ruged, with a set of valleys and irregularities carved in the landscape, raise critical challenges in data communication effectiveness, both within the WSN and on data uploading to the CP. In fact, despite being specially relevant on those regions, similar difficulties are shared in regions where the landscape is mainly occupied with farms. Such constraints counterpoint the use of popular communications standards based on higher frequency radio frequency bands (*e.g.* IEEE 802.11, BLE and some implementations of IEEE 802.15.4 and ZigBee). As lower frequencies, thus higher wavelengths, present better propagation performance, they are more suitable for these kind of scenarios.

4.4.2 MAC Layer

When designing a communication stack for systems on which energy-efficiency is of utmost importance, the MAC Layer architecture is vital. As a matter of fact, in IoT systems, the energy expenditure associated to data transmission via radio transceivers represents a substantial quota among the total energy consumption [237]. Consequently, the solution proposed for the MAC Layer is the cornerstone of the entire solution, being also one of the most relevant contributions of this thesis.

Looking at the system requirements and overall architecture, a few devices share the same medium. As it is expected to have a high number of nodes (hundreds of sheep in a few hectares), it is imperative to have an efficient mechanism to control the access to the medium, avoiding undesirable collisions, packets losses, waste of bandwidth and consequently energy consumption. Moreover, the system must accommodate diverse traffic types, with very specific purposes. We already identified some of them, for instance, periodic sensor data sent from collars to beacons, periodic transmissions containing gathered data sent from beacons to beacons and periodic enabling-localization messages from beacons to collars. Nevertheless, the periodic traffic is not enough to support all system needs. There are sparse tasks that demand the support of both periodic and aperiodic communications, for instance when there is a need of adding new collars to the network (pairing).

To deal with this kind of traffic in an energy efficient way, the strategy proposed approaches an hybrid TDMA/CSMA MAC scheme, where the majority of the communications follow a Time-

Triggered (TT) architecture [238], which means that relevant tasks and communications are executed following a predetermined schedule. Hence, this schedule may be defined aiming at optimizing the entire system operation. Particularly, we are mainly interested in maximizing the autonomy of mobile nodes and we do it by minimizing the time that nodes' radios are turned on and by controlling the concurrency among local tasks being executed.

4.4.3 Transport Layer

The Transport Layer includes a set of modules responsible for handling system messages, insertion of new nodes, routing messages and security features. More precisely, four modules were planned, namely: *i)* Message Transport; *ii)* Network Manager; *iii)* Routing Manager; *iv)* Security Manager.

The **Message Transport** module encompasses the needs related with message exchanging, both within the WSN and between the WSN and the CP. In other words, the Message Transport module defines all the messages exchanged within the system and ensures their correct parsing.

The system architecture comprises both a WSN, where a proprietary communication mechanism is applied, and a CP that is based on a typical IP architecture (see Section 4.5 for more details). Thus, message transport entails three different levels of needs, namely:

- **Definition of the messages exchanged within the WSN**: the messages exchanged within the WSN enable not only gathering data to monitor the system and sheep, but are a key-enabler to ensure a proper communication between devices. As within the WSN energy and processing constraints coexist, the messages need to be designed considering such aspects, without compromising system effectiveness;
- **Message fragmentation, reassembly and parsing within the WSN**: the messages within the WSN are transmitted through a radio transceiver with specific features, including a Maximum Transmission Unit (MTU) and maximum transmission baudrate. Hence, besides the message parsing within each one of the devices, packet reassembly is a requisite for some messages;
- **Message translation to an IP-based network**: although the messages exchanged within the WSN follow a proprietary scheme, when delivering them to the CP, it is necessary to convert them to a format such that they could be easily parsed to IP-based components and protocols. That task is performed in the gateway, that is the component that interacts both with constrained devices in the WSN and in the non-constrained components within the CP.

The **Network Manager** aims to enabling the control of the ingress and egress of nodes. It includes:

- **Pairing of devices:** when a device is new on a network, it needs to be recognized and accepted within that network. This process is called pairing and, when successfully established, it allows devices to communicate synchronously in the network within scheduled time-slots;
- **Monitoring and failures detection:** device's operation needs to be monitored in order to prevent, detect and log failures. Therefore, mechanisms of self-monitoring shall be designed and included;

The **Routing** Module has two main purposes, namely, managing the schedule of time slots and managing the active routing scheme on beacons. Both share common objectives, particularly:

- **Minimize the number of transmissions:** in a network with several beacons, it is a waste of resources to propagate data to hops that are further away from the gateway, at least under normal operation conditions. Also, it is not advantageous to transmit firstly data from beacons closest to the gateway and in the end data from the furthest beacons as this would increase the number of transmissions required to deliver all data to the gateway;
- **Minimize power consumption:** optimizing the routing and scheduling schemes allows to reduce average power consumption. Therefore, the routing manager implements routing schemes that aim at decreasing power consumption.

Currently, a static approach associated to the network identification is proposed for the scheduling. As such, during system deployment, beacon distribution is done assuming that they are placed aiming at meeting the aforementioned objectives. However, this does not match the level of self-government intended for the system. Therefore, and considering the needs of the system, the scheduling mechanism shall enable beacons to establish their schedule without requiring a reconfiguration of the entire system. This could be done on a setup phase that would occur whenever a new beacon infrastructure is deployed or restarted. This means that, before starting in normal operation, beacons would establish a schedule that shall be controlled by the gateway. The main deciding rule for scheduling shall be: beacons that are further away from the gateway in terms of number of hops transmit first, although additional rules shall be evaluated.

Regarding routing protocols for WSNs, a wide range of solutions have been proposed in the literature to deal with different network topologies, traffic patterns, metrics to be optimized/minimized, just to cite a few. Even though, we argue that a RPL-like scheme adapted to the constrains of the solution may be a adequate solution. RPL, although designed for communication stacks supported on IPv6, owns features that are aligned with our needs. Particularly, it supports different types of traffic and enables the use of different metrics to establish the active routing. Furthermore, we can easily identify a match between RPL properties and our system properties.

RPL resorts on four main properties, namely, instance ID, DODAG ID, DODAG version number and rank. From these, three of them match parameters expected to exist in the system being presented. DODAG ID identifies isolated networks that share the same gateway, and, therefore, is associated to the field that allows to distinguish clients' properties. The DODAG version includes the routing scheme currently active and it is used to ensure that all beacons are using the same. The rank signalizes the distance, in hops, from the node to the root. Therefore, in this system, it would correspond to the number of hops from the beacon to the gateway. This rank can be calculated in the set up phase through the exchange of successive messages. The preferred parent for each beacon can be calculated through a OF-like that uses beacon location, beacon energy and rank. Both routing version, rank and preferred parent, besides being part of the beacon state, can be transmitted within the periodic messages designed, such that all network are aware of the topology of the beacon infrastructure.

The **Security Manager** is responsible for implementing security measures to protect the system from external threads. There are several security challenges that need to be addressed, although they are out of scope of this thesis. They are mainly related with data confidentiality, data and devices integrity and data availability [239]. Among the most relevant topics, we highlight:

- **Authentication:** it is necessary to ensure that only allowed devices are able to communicate with other authorized devices. Hence, within this topic, besides the pairing process, a specific code, associated with the property where the devices are being used, is incorporated on all messages. This value is used by devices to know if a certain message can be accepted or not;

- **Cloned devices:** cloned devices may be inserted in the network and compromise the operation of neighbouring devices. Therefore, a mechanism to detect or avoid cloned devices is recommended;
- **Integrity:** modified data, both due to malfunctioning devices and external interferences shall be detected, since devices and network operation may be compromised. For instance, if wrong synchronization data is propagated through the network, the integrity of the entire system is compromised, which may entail in significant losses for the user;
- **Interferences**: external interferences, as data integrity, may compromise all system operation. In fact, wireless communications suffer from interference issues, that must be handled conveniently;
- **Malfunctioning devices:** malfunctioning devices shall be detected and removed from the network to avoid threatening situations, including related with sheep safety. For instance, a device whose monitoring and conditioning mechanism do not work properly represents a potential production lost to the farmer.

4.5 Computational Platform

The CP enables WSN data to be processed and stored by more powerful computational elements compared to the ones available within the WSN. Additionally, the use of DM and ML tools supports the creation of additional valuable services to the user, such as the detection of illness and lameness conditions, the monitoring of animal reproductive cycles, the evaluation of preferred grazing areas and timings, the identification of social relationships, to name just a few.

The architecture proposed for the CP is illustrated in Figure 4.6 and it is composed of five different interconnected modules. The Message Oriented Middleware (MOM) broker RabbitMQ [240] is the entry point for data that is sent by one or more gateways. It allows message routing through producers and consumers, being the producers the gateways that send JavaScript Object Notation (JSON) messages to the CP, and the consumer(s) other components, within or outside the CP, interested in handling the data. The broker supports several messaging protocols such as AMQP and MQTT, both based on asynchronous publish/subscribe architectures. Therefore, RabbitMQ works as an intermediary between the Gateway and the remaining platform, managing all the received messages prompted by the Gateway in a First-In-First-Out (FIFO) queue. Also, RabbitMQ allows security mechanisms to be employed, such as SSL/TLS encryption.

Currently, the processing framework Apache Spark [241] is the unique subscriber of RabbitMQ queues. It is the main processing framework and it is responsible for orchestrating all operations of the platform. Among them we highlight JSON data translation, alarms generation, data processing including DM and ML techniques, and data persistence into a database (DB). It combines two main processes namely a stream process that handles real-time traffic; and a batch process that handles non-periodic traffic that is the output of processing tasks performed within the platform. Both processes are able to persist into the DB, feeding it regularly with new data.

Complementarily to this processing framework, a rule management module (Drools) [242] is included in the architecture for allowing the definition of complex event processes and event stream

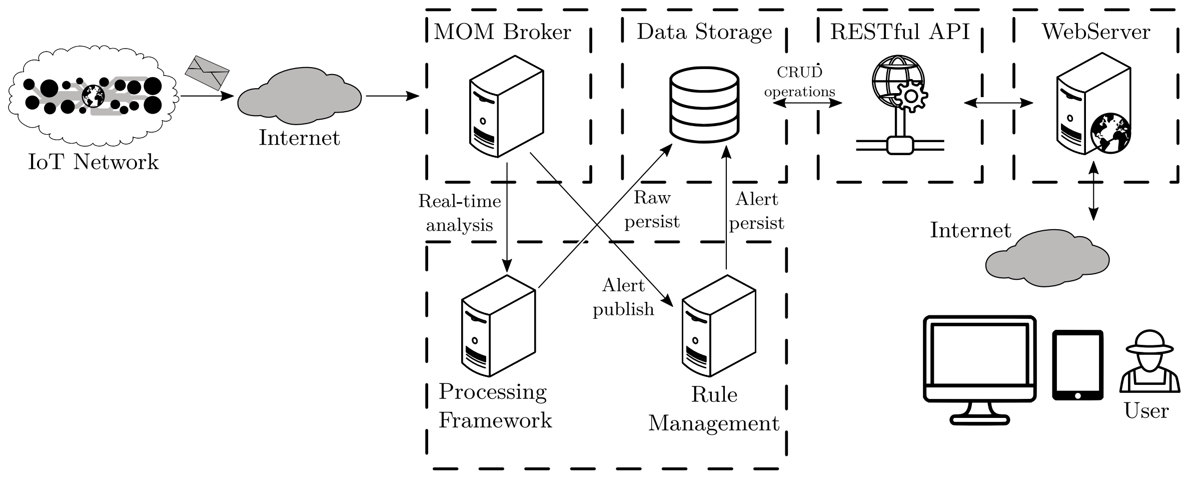


Figure 4.6: Computational Platform architecture.

processes. Briefly, that module allows the generation of predictions, detection of patterns or other relevant relations that may trigger actions as alarms or real time notifications to the user.

A database is also part of the CP. The data gathered directly by collar sensors as well as data extracted through its processing, static information added by the user and other information about system operation, need to be stored conveniently in order to be easily accessed by upper layer applications. As several entities coexist, a relational database model is a more promising approach comparing to a non-relational model. Moreover, besides all the advantages of a NoSQL model, the payload of the messages expected does not justify the use of that approach. There are several available solutions, being PostgreSQL one of the most popular ones, particularly because of its suitability for environments with a large amount of data, besides the security and integrity mechanisms that it presents.

Finally, a REST API for enabling Create, Read, Update and Delete (CRUD) operations is enclosed to facilitate the access to data, either for web development or for facilitating the integration with other web services, for instance for allowing automatic integration of animal information into legal databases for animal registry.

4.6 Summary and discussion

In this chapter we detailed the proposed IoT-based solution architecture for intelligent farming, built considering the needs of the SheepIT use case. We started by reviewing the functional requirements that the solution shall comply with. A few of these functional requirements are shared by any IoT system, although others are specific to the use case disclosed.

Obviously, the functional requirements were the core support for the design of the entire system, including overall system architecture, devices and communication stack. Locally, a WSN is composed of sensors and actuators, beacons and gateways. Different sensors types are acceptable, although the focus of this thesis was on animal sensors, particularly collars armed with inertial sensors and actuators designed to monitor and condition sheep's behaviour. These collars gather data and transmit them to beacons that are placed over grazing areas to collect data and support a RSSI-based localization mechanism. Finally, the gateways allow local data to be transmitted to a remote computational platform.

The communication stack is a pillar of the work and represents one of the main contributions of this thesis. Four main layers were identified, mainly Physical, MAC, Transport and Application layers, although a deeper relevance was given to the MAC Layer due to its criticality for the majority of the requirements defined.

A hybrid MAC scheme, with support for both TDMA and CSMA schemes is the approach proposed. As such, it supports different types of traffic to be transmitted in the same system, offering the best of the two approaches. On the one hand, supports TDMA-based communications, that avoid the waste of unnecessary energy in retransmissions. On other hand, it offers flexibility by supporting CSMA-based communications, particularly relevant for aperiodic transmissions.

Within the Transport Layer, four modules were introduced. The Message Transport module addresses the messages designed to satisfy the needs of the system. The Network Manager manages the ingress and egress of nodes. The Routing Manager handles how messages are routed through the WSN to ensure an efficient delivery process of system messages. It is also responsible for handling the message scheduling, particularly regarding beacon communications. Finally the Security Manager, as the name suggests, handles security issues.

Finally, the computational platform designed to handle all the traffic generated in the WSN was introduced. This computational platform has processing and storage capabilities that are particularly relevant to support the application of DM and ML techniques to extract value from data collected by the system.

CHAPTER 5

IoT-based Communication Stack Implementation

After defining the IoT-based architecture designed to support intelligent farming solutions, including the guiding requirements, the necessary devices and the communication stack, this chapter details the implementation of the IoT-based communication stack. We cover all layers of the stack, and present details regarding their implementation. A careful attention is given to the MAC and Transport Layer, the former due to its criticality on the current consumption of the system, namely on collars, and the latter because it defines the messages to enable communication between devices.

5.1 Physical Layer

The Physical Layer proposed is based on radio transceivers operating preferably at lower frequency ISM bands of 433 MHz or 868 MHz. Within the available standards supporting the 433 MHz and 868 MHz ISM bands, there are a few ones that, due to their features and relevance, must be discussed. Sigfox [59], despite being energy efficient as required for our solution, only allows the transmission of a very modest number of size-limited messages per day. Additionally, it depends on the deployment of private gateways controlled by Telephone Companies (TELCO), consequently, coverage issues are expected, together with the associated high costs.

LoRa [60] technology was designed to support long range communication, particularly for IoT applications. It offers reasonable bit-rates (up to 50 kbps for shorter distances) and it was conceived to be low power. It enables the implementation of RSSI-based or ToA-based localization mechanisms, although known results are poor [62]. Furthermore, the ToA mechanism is proprietary, making it inflexible and unchangeable in order to enable the development of enhancements above it. Supported network topologies are somehow rigid, limiting the implementation of further energy-wise strategies to increase mobile nodes autonomy. All together, coupled with the cost of LoRa nodes, led us to not consider LoRa as an option, at least for being incorporated on mobile nodes (the collars).

Ultimately, a reference to NB-IOT [64] and LTE-M [65] that are being developed by 3GPP LTE. Although being seen as the future of the IoT, it will take some years until they can offer a reasonable coverage since they depend on TELCO infrastructure. Also, the need for paying a fee for each node, may lead to unbearable costs, specially for farming scenarios. Consequently, at the moment, we don't see these protocols as a suitable option.

In sum, despite the recent emergence of some standards with interesting features, they do not enable the development of a solution capable of matching all the requirements. Thusly, our approach comprises the development of a new communication stack, based on a Physical Layer employing commercial radio transceivers operating in the 433 MHz or 868 MHz unlicensed ISM bands.

5.2 MAC Layer

We identified a hybrid MAC scheme and a TT paradigm as the appropriate approaches to base our MAC Layer implementation. We now present and justify its implementation towards the fulfilment of research question 2.

5.2.1 The cyclic-based structure: the Macro and Micro-Cycles

Regarding wireless communications, most of the communications are periodic in nature. Therefore, in order to avoid collisions, packets losses and consequent waste of bandwidth, and to decrease the energy consumption with retransmissions, the permanent-state communications are TDMA-based. This communication scheme is complemented with CSMA-based aperiodic communications, used for events such as device's paring in the network.

Considering these different types of traffic, and additional ones that can be added in the future, we propose a strategy based on temporal multiplexing, using a periodic cyclic-based structure. As depicted in Figure 5.1, a sequence of Micro-Cycles (μCs) compose a Macro-Cycle (MC) that is repeated over time. Each type of μC has a particular purpose and, according to that purpose, a TDMA or CSMA MAC policy.

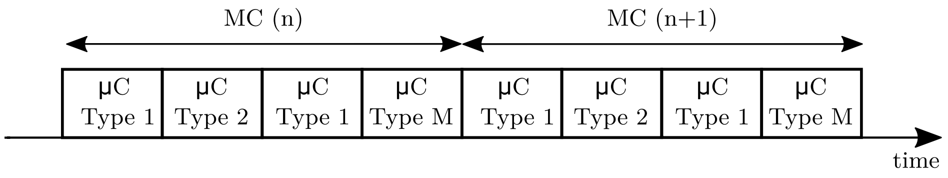


Figure 5.1: Periodic cyclic-based structure proposed for the communication mechanism. The smaller units are called Micro-Cycles (μCs), which are repeated over time composing a Macro-Cycle (MC).

As depicted in Table 5.1, three types of μCs are currently defined. The first is named Pairing Request/Response (PR) μC and is included to allow device pairing. Aiming at the design of a self-configurable network, collars should be able to pair transparently on the system when a new collar needs to be added or when a collar experiences a reset. However, the number, identity and time instant at which these events happen cannot be predicted, so a dynamic pairing process is required (Section 5.3.2). The temporal characteristics of these events suggest the use of a CSMA arbitration scheme. Nodes that are not paired stay in listening mode until hearing the start of a PR μC, where they can ask for pairing. The probabilistic nature of this process may result in several pairing attempts, during which nodes remain awake and with the radio turned on, either to transmit or receive. This consumes a significant amount of energy, but it is important to note that nodes are seldom in this state, thus the impact in the autonomy is limited.

Antagonistically, for μC types on which intervening nodes are already known by the system, *i.e.* already paired in the network, it is desirable to schedule messages following a TDMA scheme. That is the choice for μC types 2 and 3, devoted to Collar-to-Beacon (C2B) and inter-beacon communications

- Beacon-to-Beacon (B2B), respectively. Each node is associated with a dedicated communication slot and communications happen at predefined times. This eliminates collisions and in the case of collars, allows nodes to remain in low power modes, waking up only when strictly necessary, namely a short period in the beginning of each μC, for receiving protocol data, and once during the communication cycle for transmitting the data. This method is particularly useful for C2B μCs where collars transmit data, as it minimizes the energy consumption by collars.

Table 5.1: μCs types currently defined.

μC **Type - Name**	**Purpose**	**MAC Policy**
1 - Pairing Request (PR)	Device's pairing	CSMA
2 - Collar-to-Beacon (C2B)	Collar communications	TDMA
3 - Beacon-to-Beacon (B2B)	Inter-beacon relay	TDMA

Independently of the μC type considered, all respect a common and static structure such as represented in Figure 5.2. They start with a **Synchronization Window (SW)**, during which each beacon sends a synchronization message carrying its identification and information about the current μC type. This information is used by remaining devices to synchronize with the network and allows the collection of RSSI information from multiple devices, crucial for the localization algorithm. During this window, all nodes are awake. Hence, the messages exchanged during this window are very small, aiming at minimizing the total duration of the window. All the messages exchanged are a duty of the message transport module within the Transport Layer and are presented in Section 5.3.

After the SW, there is a window called **Turn-Around Window (TAW)** for processing and where no communications occur. Common processing activities carried in the TAW include processing the messages sent during the SW to identify the current μC, and the preparation of the messages to be sent in the respective time slot. For beacons, the messages received during the SW, in addition to enable network synchronization, allow beacons to identify and maintain information about their neighbours. Furthermore, collars data that is received in previous μCs is processed and stored, such that can be relayed to the gateway. Thus, rather than doing such processing within the assigned communication time slot, all processing is done during the TAW. Such approach allows to decrease the complexity of time slots implementation and it minimizes task concurrency, duration of the time slot and peak energy consumption.

Concerning collars operation, right after processing the messages received during the SW, the posture control mechanism is initialized. That includes reading sensors, processing sensor data and implementing the monitoring and conditioning algorithms (more details about the posture control algorithm is given in Chapter 6). Furthermore, if the current μC type corresponds to a C2B, it creates the message to be sent, minimizing the processing time within collar's time slot.

Finally, within the **Variable Traffic-type Window (VTW)**, the data associated to the μC type is exchanged. If the μC is of Type 1 - PR, aperiodic communications can occur associated to pairing requests and responses. On the other hand, if the μC is of Type 2 - C2B or Type 3 - B2B, the data is sent in scheduled time slots, either by collars or beacons, respectively.

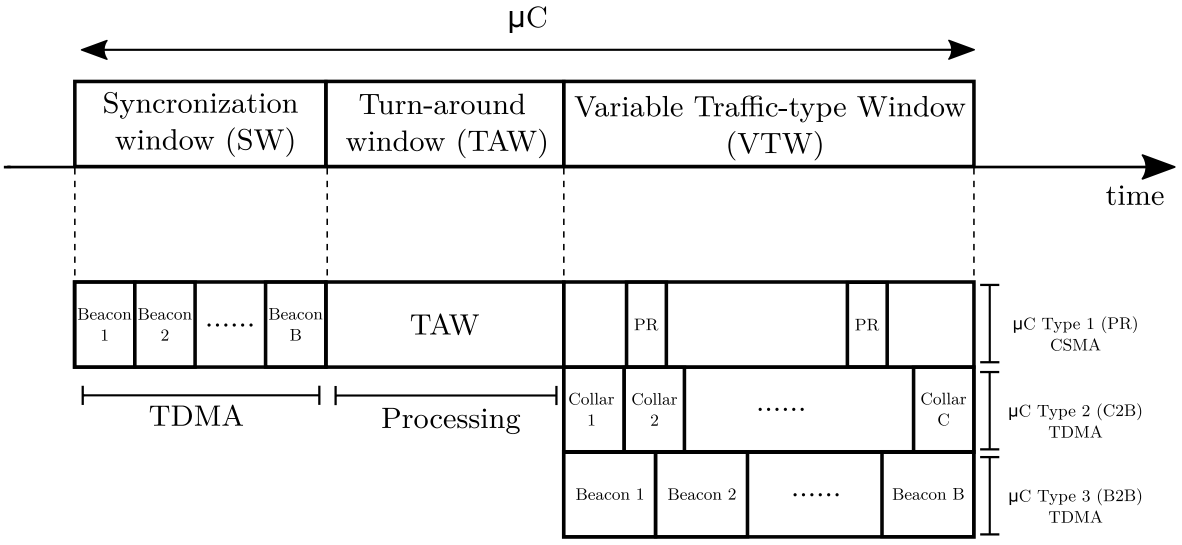


Figure 5.2: μC structure. In the upper part of the figure is illustrated a generic composition of a μC, while in the bottom a detailed representation is given for the three μC types currently defined.

5.2.2 μC minimum duration

The windows that compose a μC depend on several factors, both associated to system and scenario configurations. Within the former group, choices such as the radio transceiver and its characteristics as the baudrate, clock drift and range, sensors duty-cycle and battery capacity, are some examples. Concerning system configurations, the number of sheep to be monitored, the intended grazing area and the periodic reporting needs are the main factors.

Therefore, there are several temporal constraints that shall be deconstructed towards an effective answer of research question 1 and 2, without jeopardizing research question 3. The central one is the μC duration.

The existence of different μC types could suggest the implementation of μCs with different durations. Nevertheless, such approach would result in a more complex synchronization process that is not suited for scenarios with processing-constrained devices, as it is the case of collars (and in some extent to beacons). As such, μCs were designed to have always the same duration, although configurable. The more meaningfully constrain is the minimum μC duration (μC_{min}), that needs to accommodate the temporal requirements of all windows (SW, TAW and VTW), *i.e.*, that needs to be sufficient to encompass the minimum duration for each window, as described in Equation 5.1.

$$\mu C_{min} = SW_{min} + TAW_{min} + VTW_{min} \tag{5.1}$$

As the SW and TAW windows are common to all μC types, the VTW is the differentiating component when calculating μC duration. Among the three types of μC defined, C2B μCs and B2B μCs are associated with periodic communications, with individual time slots for each collar and beacon, respectively. Hence, we assume that the duration calculated for those μC types is sufficient to encompass the aperiodic communications occurring in PR VTW. Thus, the minimum duration of a μC can be defined as the maximum among $\mu C_{C2B_{min}}$ and $\mu C_{B2B_{min}}$, as defined in Equation 5.2. The minimum duration of a μC depends on the minimum duration of each window, and thus, we evaluate each one of them in the following sections.

$$\mu C_{min} = max(\mu C_{C2B_{min}}, \mu C_{B2B_{min}}) = SW_{min} + TAW_{min} + max(VTW_{C2B_{min}}, VTW_{B2B_{min}}) \quad (5.2)$$

5.2.2.1 Time slot composition

Before moving forward to the calculation of the μC minimum duration, it is important to detail the composition of a time slot.

When a node receives a packet, it needs a small amount of time to process and buffer it. As typically this time is not negligible and since it directly affects the time between consecutive transmissions, it needs to be considered in the time slot. We refer to this time as Time Slot Time-to-Receive (TS_{RX}) that, together with the Time Slot Time-to-Transmit (TS_{TX}), composes the time slot duration that we refer as ($TS_{duration}$). This duration depends on different parameters such as the size of the message to be transmitted, the maximum size of the packet allowed for the chosen technology and the baudrate. Hence, different $TS_{duration}$ are settled, according to the carried traffic.

One of the main issues when resorting on a TDMA scheme is the synchronization effectiveness. Clock drift is a common issue whose effect needs to be accounted for. In fact, albeit clock's accuracy and timer's uncertainties have been reduced with technological evolution, there are always slight deviations that may result in malfunctioning systems. For instance, if the TS_{RX} is shrank due to clock drifting, the required time for a receiver node to process a packet is not ensured, which means that if a new packet is transmitted in the meanwhile, it may not be received by the receiver. To overcome this potential issue, a Guarding Window (GW) is added between continuous TSs. This small window allows to absorb detours due to clock drift, avoiding potential overlapping between different communications. The GW value depends on the cock accuracy, being thus hardware-dependent.

5.2.2.2 Synchronization window duration

The SW is the key for network synchronization, the basis for the localization mechanism and it is where beacons announce their presence. That is possible since all beacons transmit small beacon messages called Beacon Syncronization Message (BS). Consequently, the SW duration depends on the number of beacons (N_{Beac}) in the network, on the duration of the time slot associated to a BS (TS_{BS}) and on the duration of the GW, as described by Equation 5.3:

$$SW_{min} = N_{Beac} \times (TS_{BS} + GW) = N_{Beac} \times (TS_{BS_{TX}} + TS_{BS_{RX}} + GW) \quad (5.3)$$

5.2.2.3 Turn-around window duration

The TAW duration is intrinsic to internal processing tasks and thus difficult to estimate. Hence, the duration of the TAW shall be evaluated experimentally and must be tailored for supporting the worst case scenario, *i.e.*, it shall embrace the most time-consuming tasks.

5.2.2.4 Variable Traffic-Type Window duration

As pointed out in Equation 5.2, the VTW minimum duration matches the maximum value among the minimum duration of a C2B and B2B VTWs. The former is given by Equation 5.4, where N_{Coll} is the maximum number of collars within the network and TS_{C2B} is the duration of a C2B time slot.

Similarly, Equation 5.5 gives the B2B VTW minimum duration, where N_{Beac} is the maximum number of beacons and TS_{B2B} stands for the worst-case transmission time of a B2B message. As beacons may have to relay other beacon's data to the gateway, these messages contain both collars data received directly by the beacon, as well as collars data sent by other beacons.

$$VTW_{C2B_{min}} = N_{Coll} \times (TS_{C2B} + GW) = N_{Coll} \times (TS_{C2B_{TX}} + TS_{C2B_{RX}} + GW) \tag{5.4}$$

$$VTW_{B2B_{min}} = N_{Beac} \times (TS_{B2B} + GW) = N_{Beac} \times (TS_{B2B_{TX}} + TS_{B2B_{RX}} + GW) \tag{5.5}$$

Associating equations 5.2, 5.3, 5.4 and 5.5, we get the expression for the minimum duration of a μC (Equation 5.6):

$$\begin{aligned} &\mu C_{min} = \\ &\quad max(N_{Beac} \times (TS_{BS_{TX}} + TS_{BS_{RX}} + GW) + TAW_{min} + N_{Coll} \times (TS_{C2B_{TX}} + TS_{C2B_{RX}} + GW), \\ &\qquad N_{Beac} \times (TS_{BS_{TX}} + TS_{BS_{RX}} + GW) + TAW_{min} + N_{Beac} \times (TS_{B2B_{TX}} + TS_{B2B_{RX}} + GW)) \end{aligned} \tag{5.6}$$

Equation 5.6 applies if we consider a MC with a single C2B μc and a single B2B μC. Nevertheless, that is not always what is materialized, specially in scenarios with a high number of collars or in scenarios where beacons topology requires the re-transmission of packets through several hops to reach the gateway.

Transmitting data from a high number of collars and thus C2B messages, increases significantly the minimum duration of a μC that is bounded by the LFP. Also, it may jeopardize the effectiveness of the localization mechanism. This is so since the minimum duration of a μC constrains the minimum time between SWs, windows where RSSI measurements are taken. Therefore, it may be desirable to have more than one C2B μC to support a high number of collars, without compromising the LFP. Likewise, as the μC duration is the same for all μC-types, the data necessary to be transmitted in a B2B time slot may rendering the transmission of all data within a single B2B μC. Therefore, several B2B μCs may be required to ensure that new data is transmitted to the gateway every MC, which in turn is bounded by the MP.

Consequently, Equation 5.6 can be re-written to support more than one μC of each type. Particularly, defining $N_{\mu C_{C2B}}$ and $N_{\mu C_{B2B}}$ as the number of C2B and B2B μCs, the minimum duration of a μC is given by Equation 5.7:

$$\begin{aligned} &\mu C_{min} = max(\\ &N_{Beac} \times (TS_{BS_{TX}} + TS_{BS_{RX}} + GW) + TAW_{min} + \left\lceil \frac{N_{Coll} \times (TS_{C2B_{TX}} + TS_{C2B_{RX}} + GW)}{N_{\mu C_{C2B}}} \right\rceil, \\ &N_{Beac} \times (TS_{BS_{TX}} + TS_{BS_{RX}} + GW) + TAW_{min} + \left\lceil \frac{N_{Beac} \times (TS_{B2B_{TX}} + TS_{B2B_{RX}} + GW)}{N_{\mu C_{B2B}}} \right\rceil) \end{aligned} \tag{5.7}$$

5.3 Transport Layer

The implementation of the Transport Layer includes, in the scope of this thesis, the message definition within the Message Transport module, the process to support the ingress and egress of devices within the Network Manager and the definition of the routing scheme to support the transmission of B2B messages from several beacons by the Routing Manager. These are the features considered vital to demonstrate the feasibility of the system and support the stated thesis.

5.3.1 Message Transport

Within the Message Transport module, the MSCs and message description of the currently defined messages are detailed. Particularly, we present the messages that are exchanged within all μC-types and the messages that are specific to μC types. Furthermore, and as message fragmentation is expected, namely on B2B messages, we detail such process.

5.3.1.1 MSCs and messages description within the WSN

The definition of the μC and MC structures has already disclosed the existing messages. Below, we expose the MSCs to each μC type. For illustrative purposes and for the sake of clarity, we assume that all nodes listen to each other, albeit that is not the typical real scenario.

The **MSC of a C2B μC** is illustrated in Figure 5.3. At the top, we can identify the messages exchanged during the SW, common to all μCs. They correspond to the broadcast of messages - BS - sent from beacons to all remaining nodes. During this window, all devices are awake and, upon receiving these messages, they are able to identify the starting and type of the new μC, triggering the necessary calculations in order to know if, when and what they should transmit. This allows collars to enter in energy-saving modes right after finishing their tasks.

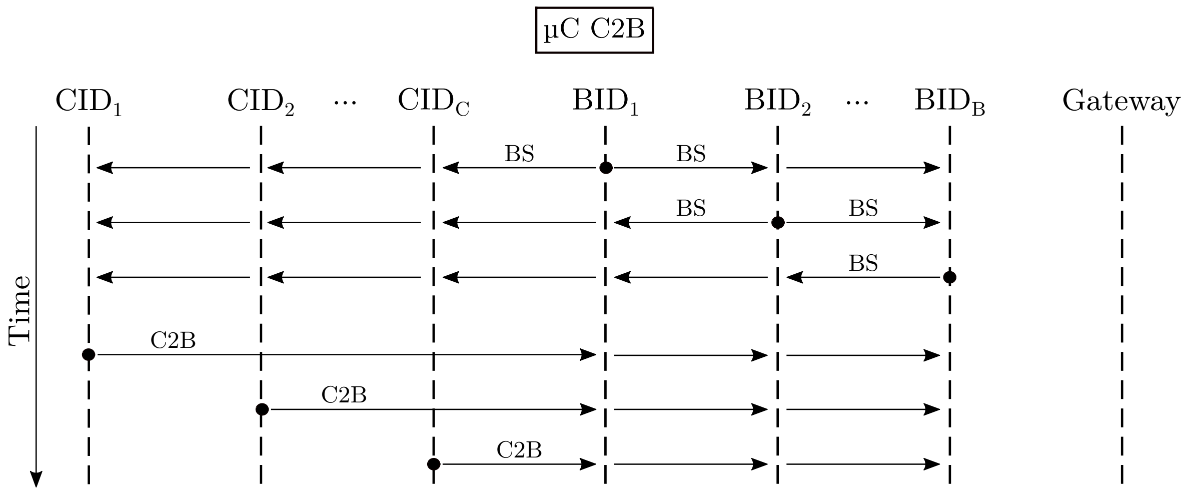


Figure 5.3: MSC of a C2B μC. Dots represent the source of messages, while arrows identify destinations. Firstly, BS are broadcast from beacons to collars, composing the SW window. Then, each collar sends its C2B message to beacons.

Once the SW ends, the messages exchanged depend on the μC type identified within the BS message. In the case of a C2B μC, the messages transmitted are also called C2B (in fact, the name of the μC is associated to the unique messages exchanged within it). They are transmitted, one per collar and per time slot, following the TDMA scheme presented in Section 5.2. Hence, as illustrated in

Figure 5.3, firstly it is transmitted the C2B message from Collar CID_1, then from collar CID_2, until the last collar (CID_C). It is import to note that CID_1 does not mean that it has the ID number 1, but that it is associated with the ID configured to be the first on the system configuration. Those messages are broadcast and received by all beacons in the coverage range of the transmitting collars. During this window, collars are awake only during their transmission, returning to power-saving modes right after finishing their transmission. In contrast, beacons remain awake during the entire window.

Before detailing the MSC of the remaining μC types, we present the structure of the two messages already identified. The BS message structure is described in Table 5.2. Besides including the necessary information to enable the identification of the message type (*Msg_type*), the identification of the network where it can operate (*PropertyID*) and the identification of the beacon sending the message (*ID*), it enables an easy identification of the current μC, including its type and position within the MC (*Type_Counter*). Finally, two additional fields are included with a double purpose. Currently, they are called *Remote_ID* and *Remote_Command*, and enable real-time configurations and actions on collars. Notwithstanding, as they are rarely used, its use can be shared for propagating routing information.

Table 5.2: BS message description.

	Field	**Bytes**	**Description**
Header	Msg_type	1	Identification of the type of message to be transmitted
	PropertyID	2	Unique identifier of the property
	ID	1	Short identifier within the WSN
	SeqNumber	1	Sequence number of this type of message for this device
	Type_Counter	1	Identification of the current μC and μC type
	Remote_ID	2	Collar ID command target | Routing information
	Remote_Command	2	Command code | Routing information
	Total	10	

The fields that compose a C2B message are described in Table 5.3. The first four fields are the same as the ones identified for a BS message, with a single difference on the size of the *ID* field (1 byte in a BS message, 2 bytes in a C2B message). This is justified due to the higher number of collars expected, comparing to the number of beacons. Then, it follows information about device status and operation, namely the battery level and the number of audio cues (*PostureBuzzer*), the number of penalizations and information about the status of the posture control algorithm (*PostureShock*). Although not tackled in the scope of this thesis, the system shall enable the implementation of a virtual fence mechanism. As such, a C2B message has reserved two fields for counting cues and penalizations due to fencing infractions (*FenceBuzzer* and *FenceShock*). For enabling a low-cost localization mechanism, the collar gathers and transmits the best RSSI values (*RCV_RSSI*) calculated when receiving BS messages from its neighbouring beacons and the respective beacons identification (*RCV_RSSI_ID*). The number of values is a system variable, although the default value is set to four since for applying localization techniques such as trilateration, information about at least three beacons is necessary - one additional measurement is added such that it can be used to minimize RSSI measuring error. The remaining fields are reserved for sensing data, particularly data gathered by the accelerometer (static and dynamic accelerations) and by the ultrasounds transducer, as well as the animal state information that is the output of the behaviour algorithm.

Regarding the sensing data fields, two important remarks must be given. The first concerns the animal state information, which was included to support the transmission of sheep state information produced by the monitoring mechanism, *i.e.*, it enables users to track sheep's behaviour history. On the one hand, it enables a real-time analysis of the animal monitoring mechanism. On the other hand, represents a much more understandable information comparing to raw sensors data. The second, is related to the bandwidth occupied by sensors data. A total of 20 bytes from a total of 39 bytes in a C2B message are reserved for transmitting sensor data, which corresponds to more than 50%. Furthermore, as the data transmitted is in a raw format, *i.e.* without any processing, it may not entail relevant information to a common user. Thus, one can see that this message can be refactored in order to transmit only useful information to the user, for instance the animal state behavioural information. However, from the research point of view, the transmission of raw data was of utmost importance, namely because it was the main support to the research tasks required to the design of the posture control mechanism enclosed in Chapter 6.

Table 5.3: C2B message description.

	Field	**Bytes**	**Description**
Header	Msg_type	1	Identification of the type of message to be transmitted
	PropertyID	2	Unique identifier of the property
CollarData	ID	2	Short identifier within the WSN
	SeqNumber	1	Sequence number of this type of message
	Battery	1	Battery information data
	PostureShock	1	Posture shock counter (1 bit for identifying if the posture control mechanism is enable, 7 for counting)
	PostureBuzzer	1	Posture buzzer counter
	FenceShock	1	Reserved: Fence shock counter
	FenceBuzzer	1	Reserved: Fence buzzer counter
	RCV_RSSI	1*4	Best 4 RSSI values measured
	RCV_RSSI_ID	1*4	Beacons IDs associated to RSSI measurements
	Accel	6	Static acceleration data (3-axis)
	D_Accel	6	Dynamic acceleration data (3-axis)
	Ultrasounds_Dist	2	Distance measured by the ultrasounds transducer
	Animal_States	6	Animal state information
	Total	39	

The **MSC of a B2B μC** (Figure 5.4) follows a similar composition to the C2B μC MSC, excepting the approach in the message scheduling within the VTW. After the BS messages, exchanged in the beginning of the μC, B2B messages are transmitted from beacons exclusively to other beacons. Those transmissions also follow a time multiplexing approach but, in this case, the transmission schedule depends on the number collars and beacons configured, on the topology of the network and on the routing scheme defined. Particularly, when the amount of data to be transmitted does not fit in a unique μC, two or more B2B μCs, may be needed to allow data to be relayed to the gateway. Finally, beacons who own a connection with a gateway, transmit all updated data to that gateway within a Beacon-to-Gateway (B2G) message. In Figure 5.4, only one beacon is connected to a gateway, although multiple beacon-gateway independent connections are allowed.

B2B messages encapsulate two types of content, data transmitted by collars and data regarding beacons operation. Data from one collar is enclosed in a Collar Notification (CN) while beacon data is enclosed in one Beacon Notification (BN). Hence, a B2B is composed of a header, plus one or more CNs and one or more BNs. Table 5.4 identifies the fields included in a B2B Header. Besides the fields shared by all the messages (*Msg_type, PropertyID, ID* and *SeqNumber*), it includes information about the battery level and the coordinates of the beacon sending the B2B, and the number of PRs, CNs and BNs encapsulated in the message. This latter information if then used during the reception to know when all data was received.

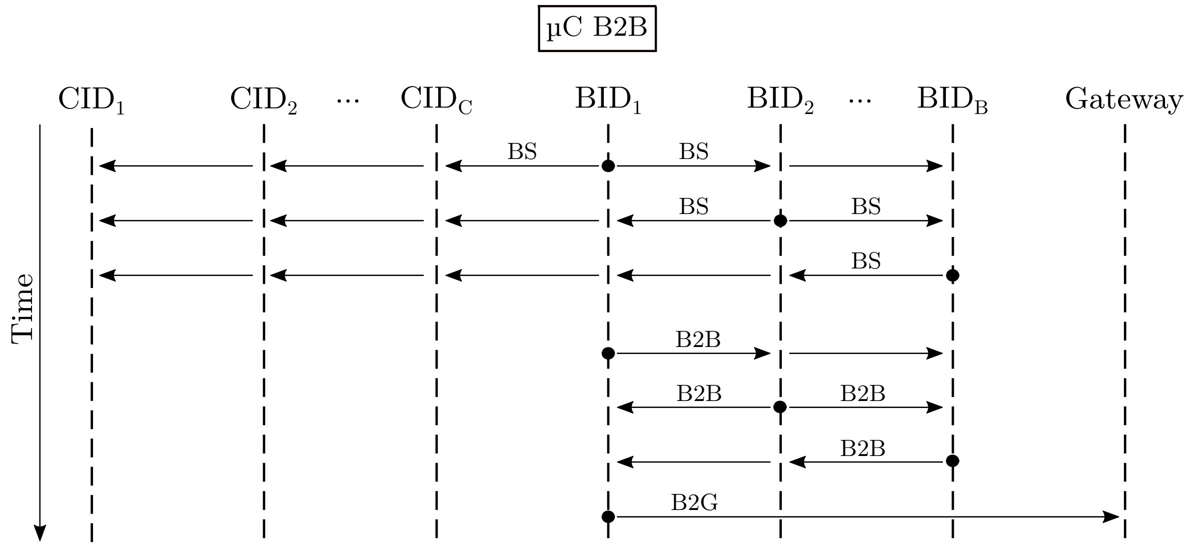


Figure 5.4: MSC of a B2B μC. After the SW window that is identical to all μCs, beacons send B2B messages following a time multiplexing scheme. Here we are assuming that all beacons own a time slot within the same μC. Also, we are assuming a single beacon connected to the gateway, in this case beacon CID_1.

Table 5.4: B2B header description.

	Field	**Bytes**	**Description**
Header	Msg_type	1	Identification of the type of message to be transmitted
	PropertyID	2	Unique identifier of the property
	ID	1	Short identifier within the WSN
	SeqNumber	1	Sequence number of this type of message
	Battery	1	Battery information data
	Coord	8	GPS coordinates
	N_PairingR	2	Pairing requests and responses
	N_CollarNot	2	Number of CN encapsulated
	N_BeaconNot	2	Number of BN encapsulated
	Total	20	

A CN, whose structure is described in Table 5.5, is composed of three different fields. The *ReportingBeacon* identifies the beacon receving the C2B message whose collar data is encapsulated in the CN. As a collar broadcasts its data and two or more beacons may receive it, an agreement must be established *a priori* in order to avoid redundant and duplicate data. Currently, the beacon that receives collar data with the best RSSI (*ReportingBeaconRSSI*) is designated to be the *ReportingBeacon*.

A BN, whose structure is described in Table 5.6, enables beacon information to be propagated through the network. This is vital to allow beacons (and the gateway) to acquire knowledge about all device status (*e.g.* battery, GPS coordinates, messages sequence numbers), which in turn is an input to the routing mechanism. For instance, if a beacon is reporting low battery or if there is not any update on the sequence numbers of the messages, that beacon is probably a point of failure and the routing mechanism needs to take that into consideration.

Table 5.5: CN structure description.

Field	**Bytes**	**Description**
ReportingBeacon	1	Identification of the beacon that reports collars data
ReportingBeaconRSSI	1	RSSI measured when collars data was received
CollarData	36	Collar data
Total	38	

Table 5.6: BN structure description.

Field	**Bytes**	**Description**
ID	1	Identification of the beacon
BS_SeqNumb	1	BS sequence number
B2B_SeqNumb	1	B2B sequence number
Coord	8	GPS coordinates
Battery	1	Battery level
Total	12	

Finally, the **MSC of a PR μC** is depicted in Figure 5.5. When one or more collars (for example $CID_?$ and $CID_{??}$ in Figure 5.5) do not have assigned a network identification, they wait for a PR μC to request it. As these requests are aperiodic, a CSMA-based contention approach is used, which means that collars can send requests at any instant within the VTW of the μC.

The requests are then propagated through the network and are handled by the gateway. Handling, includes assigning a network identifier and back propagate the response to the requester, also within PR messages. As responses are propagated to all beacons, collars may receive duplicate information. Hence, they need to check firstly if any of the responses belong to them and validate if they have not received them yet. If so they accept them, if not, they ignore them (event represented with an X in Figure 5.5). Further details about this process are given in Section 5.3.2, when describing the Network Manager.

Table 5.7 summarizes the structure of a PR message. One can see that a single message was defined, although there are requests (from collars to the gateway) and responses (from the gateway to collars). That is possible since the *ID* field enables that distinction. When collars make requests, they do not possess any ID and this field is sent empty. In this case, remaining nodes know that it is a request from the collar with the serial number encapsulated in the message. In contrast, when the *ID* is filled with a valid value (greater then 0 and below a defined bound), system devices know that it is a response and collars look for the serial number field to check if it belongs to them.

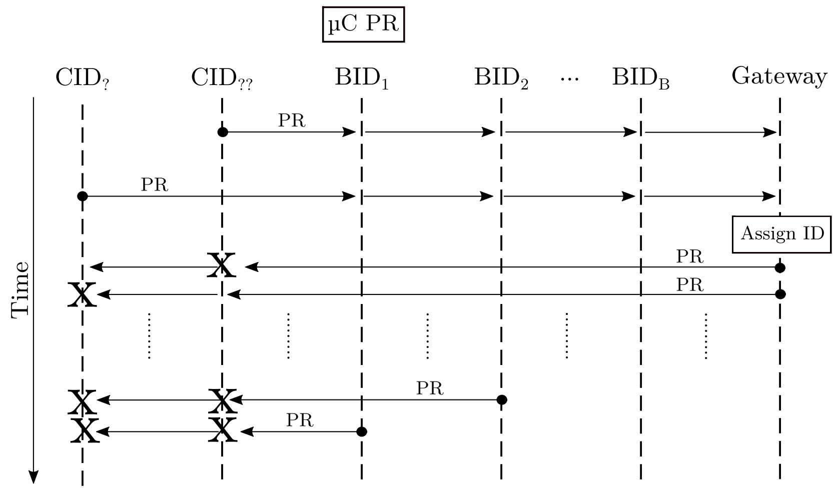


Figure 5.5: MSC of a PR μC. $CID_?$ and $CID_{??}$ are examples of collars not yet paired, which means that they do now have assigned any network identifier. A X means a message ignored.

Table 5.7: PR message description.

Field	**Bytes**	**Description**
ID	2	Field that carries the network identification of a collar
CollarSN	4	Collar serial number
AnimalID	16	Animal identification (animal microchip transponder)
Total	22	

5.3.1.2 Message fragmentation, reassemble and parsing within the WSN

BS, C2B and PR messages do not include any fields to support message fragmentation and reassemble parsing. That is possible as long as the technology and the transceiver chosen to transmit data supports a MTU greater than the biggest message among the referred messages - in this case, 38 bytes of a C2B message.

On the contrary, the length of a B2B can easily scale up to values whose transmission within a single radio package is impracticable. As an indicative example, if a beacon has data from 100 collars to encapsulate in a B2B message, a MTU greater than 3800 bytes would be necessary, which exceeds by far typical MTU sizes of low-power radio transceivers. Hence, B2B messages need to be fragmented by the sink beacon and reassembled in the receiver.

Figure 5.6 illustrates the updated structure of a B2B VTW considering fragmentation needs. Within each B2B slot, no matter to which beacon is assigned, a B2B header plus one or more B2B sub messages coexist, each one with a maximum length equal to the MTU of the technology used. In turn, each B2B sub message is composed of a header and data to be transported, that can be either CNs, BNs and/or PRs (see Table 5.8).

Assuming that all data is encapsulated within a single B2B time slot and that beacons transmit all data once in a MC, the number of B2B sub-messages required to transmit all data (CNs, BNs and PRs) is given by Equation 5.8. As depicted, it depends on the sizes and quantities of CNs, BNs and PRs structures ($sizeof(CN)$, $sizeof(BN)$ and $sizeof(PR)$ and N_{Coll}, N_{Beac} and N_{PR}, respectively)

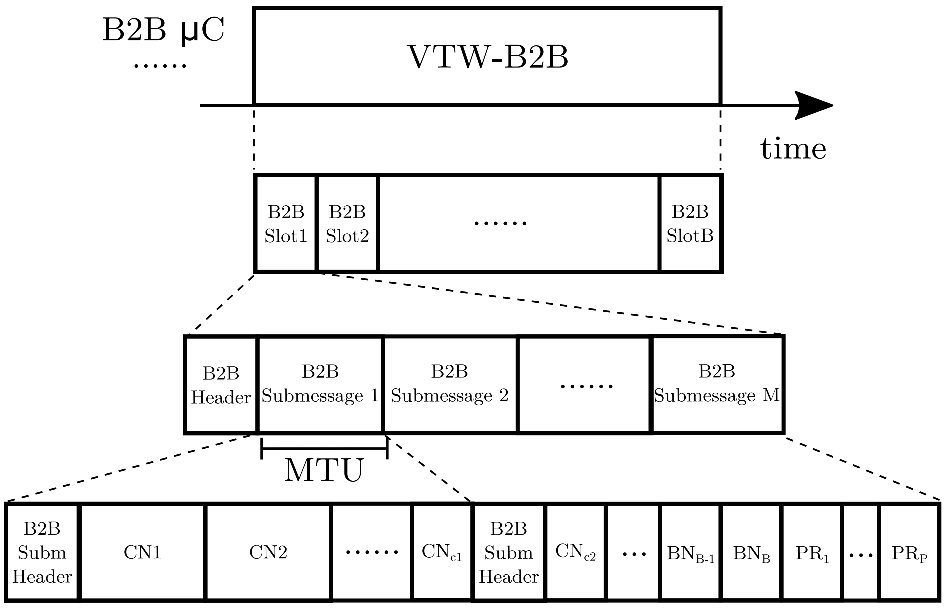


Figure 5.6: B2B message fragmentation within a B2B time slot. CN_{c1} and CN_{c2} represent an example of the messages fragmentation. BN_B and PR_P are the last BN and PR to be transmitted, respectively.

Table 5.8: B2B sub-message description.

	Field	Bytes	Description
Header	Msg_type	1	Identification of the type of message to be transmitted
	PropertyID	2	Unique identifier of the property
	ID	1	Identification of the beacon sending the data
	SeqNumb	1	Sequence number of this type of message
	RoutingInfo	1	Reserved for routing
	Data	MTU-6	Data to be transmitted
	Total	MTU	

to be transmitted in the B2B time slot, and on the MTU and B2B sub-message header sizes.

$$\begin{aligned} N_{B2BSubMessages} &= \frac{Total\, data\, occupied\, by\, CNs,\ BNs\, and\, PRs}{Payload\, of\, the\, MTU} = \\ &= \frac{sizeof(CN) \times N_{Coll} + sizeof(BN) \times N_{Beac} + sizeof(PR) \times N_{PR}}{MTU - sizeof(B2BSubMessageHeader)} \end{aligned} \tag{5.8}$$

The fragmentation process was designed considering two main architectural needs, namely keeping the computational cost low and be a process simple to implement. Therefore, its operation is designed as follows:

- Each beacon owns a single B2B time slot;
- Each B2B time slot starts with the transmission of a B2B header that contains information about the number CNs, BNs and PRs that are encapsulated in the subsequent B2B sub messages;
- Each B2B sub message includes a header that enables receivers to reassemble messages when received;

- Beacons maintain a table with the information of which data is updated and should be transmitted in the respective B2B time slot;
- Only the data updated is transmitted;
- The data is copied such that no empty spaces exist between B2B sub messages;

The aforementioned fragmentation process has a natural impact in the reassembly process, particularly:

- If a B2B header is not successfully received by the receiver, all subsequent packets are discarded. Although this lead to irretrievable data loss, its negative effects can be minimized through an appropriate beacon distribution together with a suitable routing mechanism such that a redundant path is maintained;
- In contrast, if a B2B header is successfully received, all the following B2B sub-messages are stored and reassembled according to the sequence number included within the header of each sub-message;
- In the case of a missing sub-message required to complete some inner structure (a CN, BN or PR), that structure is given as lost.

This fragmentation process impacts in the calculation of the μC minimum duration. We saw that message fragmentation scheme includes the construction of *B2BSubMessages* whose total size is equal to the MTU size of the radio transceiver. Therefore, this logic needs now to be included in Equation 5.7, particularly within the B2B parcel. In that equation we defined $TS_{B2B} = TS_{B2B_{TX}} + TS_{B2B_{RX}}$ as the worst-case transmission time of a B2B. However, as a B2B message may be composed of several *B2BSubMessages*, the worst-case transmission of B2B message needs to be decomposed. Equation 5.9 reflects that, being the minimum duration of a B2B time slot given by the time needed to transmit all required *B2BSubMessages* plus respective GWs, and the time required to transmit a B2B header plus the GW (there is always a B2B header per B2B time slot). $N_{B2BSubMessages}$ is given by 5.8, while $TS_{B2BSubMessage}$ represents the worst-case transmission time of a *B2BSubmessage* which size is equal to the MTU size.

$$TS_{B2B_{min}} = N_{B2BSubMessages} \times (TS_{B2BSubMessage} + GW) + TS_{B2BHeader} + GW \tag{5.9}$$

Combining Equations 5.7 and 5.9, the value for μC_{min} is given by Equation 5.10.

$$\begin{aligned}
&\mu C_{min} = \\
&\quad max(N_{Beac} \times (TS_{BS_{TX}} + TS_{BS_{RX}} + GW) + TAW_{min} + \left\lceil \frac{N_{Coll} \times (TS_{C2B_{TX}} + TS_{C2B_{RX}} + GW)}{N_{\mu C_{C2B}}} \right\rceil, \\
&\qquad N_{Beac} \times (TS_{BS_{TX}} + TS_{BS_{RX}} + GW) + TAW_{min} + \\
&\qquad \left\lceil \frac{N_{Beac} \times (N_{B2BSubMessages} \times (TS_{B2BSubMessage} + GW) + TS_{B2BHeader} + GW)}{N_{\mu C_{B2B}}} \right\rceil)
\end{aligned} \tag{5.10}$$

5.3.2 Network Manager

Figure 5.7 depicts how pairing requests and responses flow through the network when not all devices listen to each other. When a collar does not own a network identifier (the *ID* field is empty), it needs to request one to the gateway. To be able to do it, two requirements must be fulfilled: it must possess a valid serial number; and it must own an animal identifier associated. The former is straightforward, since it is included in the manufacturing process, being only necessary to validate that the device belongs to the associated owner (through the *propertyID* field included in the header of every message). The later is associated to the animal identification, that must follow European regulations [243]. In sum, every animal must have a double identification, an ear tag and a RFID transponder following the standard ISO 11784 [244]. This transponder owns an unique identification of 128 bits that can be accessed by RFID readers complying with the referred standard. That identification corresponds to ours *AnimalID* field, being currently assumed that this association is established *a priori*[1].

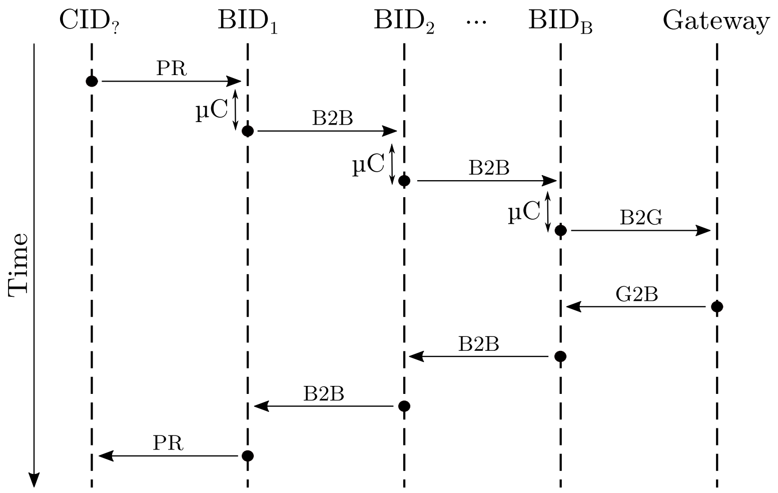


Figure 5.7: PR process description. In this example, $CID_?$ is in the range solely of BID_1, forcing the PR messages to flow through the network. For the sake of clarity, we have omitted the broadcast of PR responses from all beacons except the one that is in the range of $CID_?$. G2B corresponds to a message from the gateway to the beacon containing the response to a received request.

Both serial number and *AnimalID* are encapsulated in a PR request message sent within a PR μC. During the subsequent B2B μCs, the PR message is relayed through the beacons infrastructure towards the gateway, which means that several μCs may be required to finish the pairing process. The gateway controls the process of assigning network identifiers. However, as one ore more gateways may coexist, it is assumed the existence of a master gateway - identified with an unique network identifier - that owns superior privileges among others.

Once the requests are collected in the gateway (the master one), a look up for free identifiers is performed. If available, they are assigned to the collars that have requested them. Assigning means updating the gateway internal table where collars' serial numbers are associated with animal identifiers and with network identifiers. Once it is done, the gateway prepares and sends within a Gateway-To-Beacon (G2B) message the PR response messages to the beacon connected to the gateway. As collars may move during the process, they may become out of the range of the beacon that have

[1]In the future, the same PR μC may be used to set the *AnimalID* within a collar. For that, it is necessary to develop a new device that besides complying with the system communication scheme, shall include a RFID reader for reading the *AnimalID* from the transponder.

received the original PR from the collar. Thus, a beacon handles a PR in two ways: it broadcasts the PR responses in the PR μC and it transmits the PR responses to beacon neighbours in B2B messages. Also, to avoid implementing acknowledgement mechanisms, this later process ends when beacons detect a C2B message whose *ID* is the one encapsulated in the PR message.

The pairing process herein described only applies to collars. As beacons are intended to be part of a set up that shall fulfil predetermined farming requirements (area to cover, topology of the terrain, number of available beacons), it is assumed that beacons are registered *a priori* during system configuration.

5.3.3 Routing Manager

The Routing Manager is responsible for implementing the scheduling and routing schemes that enable beacons to know when and to whom they should transmit.

Regarding the scheduling scheme, currently, B2B timeslots are assigned statically according to beacon's identification (ID). Particularly, the schedule is defined such that beacons with ID 1 transmits first, then transmits the beacon with ID 2, up to last beacon which is defined by the maximum number of beacons. Therefore, to optimize beacon transmission, it is assumed that beacons are placed such that higher IDs are placed near the gateway, while beacons with smaller IDs are placed farther from the gateway. As the closest beacons to the gateway will always need to transmit updated information to the gateway and as we assume that beacon with higher IDs transmit later, we minimize situations on which a beacon transmits data to the gateway without having the most updated information. That occurs, for instance, if most of the collars are around a beacon that is farther from the gateway and hence the information need to flow through the entire network until the beacons that are one hop away from the gateway.

Beacons maintain a list of updated CNs and BNs such that when a beacon receives data, it confirms if it corresponds to new data or if it already owns the most recent one. In the former case, the beacon updates internal tables, if not, it discards the data. When it is its time slot to transmit, it sends the entire table.

Concerning the routing scheme, currently B2B messages are broadcast and processed similarly by all beacons. Therefore, albeit system messages already include information to be used by the routing scheme, in current system implementation beacons accept B2B messages from all remaining beacons, no matter who sent it.

These scheduling and routing schemes were considered at this stage of system development due to its simplicity and low processing demands, which enabled, on the one hand, the validation of system feasibility considering worst-case conditions, and, on the other hand, the development of the preliminary version of the system. However, such approaches are highly inefficient, particularly in what concerns to bandwidth utilization, inter-arrival time of messages in the gateway, energy expenditure and autonomy. Therefore, they are subject to improvements, particularly, following the approaches presented in Section 4.4.3.

5.4 Summary and discussion

This chapter presented the IoT-base communication stack implementation whose architecture was detailed in Chapter 4. The focus were the MAC and Transport layers since, besides influencing greatly the system energy consumption, they are vital to enable the evaluation of the system feasibility.

A cyclic-based structure composed of multiple μCs arranged to form a MC was proposed. To accommodate different types of traffic, both CSMA and TDMA-based schemes are used within a μC structure that is repeated to form a MC. The temporal conditions and constrains associated to this implementation were presented, including the formulas to calculate the different windows that compose a μC.

Regarding the Transport Layer, we defined the messages required to implement the system requirements, particularity for the SheepIT use case, which included messages from collars to beacons (C2B), synchronization messages from beacons to all devices (BS), messages to relay data between beacons (B2B) and messages to enable the pairing of devices. Furthermore, and due to the need of fragmentation of B2B messages, we detailed such process, including the required sub messages.

Finally, we addressed the Network and Routing Managers. The Network Manager defines the methods for the insertion of devices in the system. We detailed the messages exchanged within the PR μC that was designed to handle this kind of aperiodic tasks. We also specify how this paring request enables the association between collars' serial numbers and sheep's identification. The Routing Manager handles beacon schedules and active routing. Currently, simplistic approaches are followed and are worst-case scenarios. As such, although they are not efficient, they enable a first implementation of the system to be evaluated.

The implementation described in this chapter regarding the communication stack allows the identification of several factors that affect the duration of the different windows that compose a μC, and consequently a MC. Some of those factors are hardware-dependent while other are application-dependent. In the former, features as radio transmission baudrate, radio sensitivity, radio range, microcontroller clock drift and operation frequency are some examples. Besides their impact on windows duration, they are not modified (at least significantly) after the choice of the hardware to be used. Conversely, application-dependent factors typically change among use case scenarios and type of applications.

These factors are not independent from each other. For instance, the μC duration, that is the same for all μC types, is inferiorly limited by a minimum duration that is calculated as being the maximum among the minimum duration of a C2B μC and a B2B μC and superiorly limited by the LFP. The duration of a SW, that is common to both μC types, depends on the number of beacons and on size of BS messages. Thus, the impact of the SW duration on the duration of a μC is the same independently of the type of μC and is affected directly by the number of beacons. Concerning the VTW duration, in the case of a C2B μC, it depends on the number of collars with a time slot assigned and on the size of C2B messages. Regarding the B2B μC duration, besides the number of collars, it also depends on the number of beacons, on the MTU of the transceiver and on the routing and scheduling schemes, currently defined considering worst-case conditions.

Furthermore, the duration of a μC has direct repercussion in the duration of a MC since a MC is constituted by several μCs. However, there are two additional dependencies regarding MC duration that must be discussed. The first lies on the minimum composition of a MC. A typical use case scenario will always require three μCs, one C2B, one B2B and one PR (although in testing scenarios the PR μC may be dismissed). The second regards the MP that settles the maximum duration of a MC. Still concerning the MC composition, the number of C2B μCs greatly influences the maximum number collars supported. But the number of B2B μCs also limits the maximum number of collars since it is necessary to ensure that all beacons can transmit all data on the available time. Therefore, these relationships must always be validated to ensure the feasibility of an user scenario.

Lastly, the number of beacons, besides influencing the minimum duration of a μC (and consequently the minimum duration of a MC), depends on the user needs, namely, on the area to be covered. In fact, the number of beacons necessary to cover a certain intended area depends on the range of the radio of the beacon and the range of a beacon depends on several factors, both related with the features of the RF transceiver chosen, the topography of the terrain and the coverage model. Additionally, besides the real coverage range provided by the RF transceiver, we are interested in using a smaller range to allow reduce RSSI variability and improve localization. Consequently, in the scope of this thesis we assume that the coverage range of a RF transceiver ($radiusRadio$) can be reduced by a range factor ($rangeF$) from the maximum radio range ($maxRadiusRadio$). That factor may be adjusted according to the needs of the RSSI-based localization mechanism. We also assume a hexagonal coverage model mainly because multiple cells fit perfectly in the case of a regular hexagon shape and a beacon distribution uniformly distributed.

Therefore, we see that these parameters are not independent, and it evaluation requires a tool for facilitating the evaluation of how the different factors impact on system performance and if there is a feasible combination of factors that allows to answer to all requirements. Consequently, after presenting the posture mechanism in the following chapter, we present a tool to enable the referred evaluation in Chapter 7.

CHAPTER 6

Posture Control Mechanism

The posture control mechanism aims at answering to research question 3. Briefly, the posture control mechanism is composed of an orchestrated interaction between continuous monitoring and conditioning mechanisms. The goal is to continuously monitor the animal behaviour and trigger, when necessary, the suitable conditioning actions. This chapter presents the proposed solution to handle such need. More specifically, we focus on the methodological approach followed to reach the intended goals, being the results presented in Chapter 8.

6.1 Animal behaviour detection

Letting sheep weed vineyards without continuous human supervision while maintaining the vines safe, demands a posture control mechanism that shall continuously monitor sheep's posture and trigger the necessary stimuli when required. Two main components compose the so called **posture control** mechanism. This section focus on the design of the monitoring mechanism, while the conditioning mechanism and its combination with the monitoring mechanism (the posture control) is addressed in Section 6.2.

Monitoring animal's behaviour through electronic devices is not, nowadays, a novelty. Nevertheless, we have already identified a set of requirements that, together, impair using existing mechanisms. Among them, two must be reminded. The first comprises the need of detecting an additional behaviour - the *Infracting* state - not yet tackled in the literature. This state corresponds to behaviours on which sheep are likely to be feeding from branches of vines or fruits and hence, comprises the type of behaviours that the system intends to avoid. The second comprises the need of implementing a real-time mechanism embedded in a low-power microprocessor. Taking this into consideration, the approach followed to solve the problem consisted of taking advantage of ML tools to study the potential features that the posture control mechanism needs to consider.

Building an efficient animal monitoring mechanism raises several difficulties, particularly because it joins two different worlds, the technological and the ethological ones[1]. Thus, albeit the technological component can be somehow controlled through careful development and testing procedures, the same is not true when dealing with animals. The unpredictability and variability of animal behaviour reveals itself to be a tremendous challenge when developing an IoT-based monitoring and conditioning system. Consequently, recognizing our natural knowledge limitations on the ethology area, an iterative problem-solving approach was followed, always resorting on ML tools.

This approach resulted in three different works ([14], [16], [19]). The first two ([14], [19]) are

[1]Ethology is the field which focus is the study of animal behaviour.

the outcome of some exploratory work whose goal was to inspect both preliminary versions of the infrastructure being developed and acquire the necessary experience on dealing with animal monitoring data. The third approach [16] builds up on the knowledge obtained from the initial works, including extensive amounts of data obtained from field trials and state-of-the-art publications. Consequently, it allowed to provide a solution less prone to errors.

6.1.1 *Infracting* and *Not Infracting* states

The first approach towards the development of a posture control mechanism was addressed in [14]. Besides presenting the generic cloud infrastructure for serving the SheepIT solution, the work reported a preliminary version of the posture control mechanism, constructed using ML as a tool.

The absence of previous experience and knowledge on animal monitoring, particularly on sheep monitoring in vineyards, both within the team and in the literature, led to the design of incremental complexity experiments. The opening approach consisted on simplifying animal behaviour to a binary classification problem. The two states considered were the *Infracting* and *Not Infracting* states, where *Infracting* stands for undesired and prohibited behaviours taken by sheep while grazing, *i.e.*, behaviours associated to feeding attempts from the vines or fruits.

Designing and carrying out this experiment rendered two main hurdles. Firstly, at the moment of that work, it was not possible to gather data from a real application scenario, *i.e.*, from a vineyard environment. This has forced the conception of alternative methods to encourage sheep to adopt infracting behaviours, such as the placement of animal food in artificial structures that could simulate vines' height. Secondly, the daunting task associated to the operationalization of this artificial scenario, together with existence of a few prototypes, precluded the test of different sheep. Thus, the data collection phase was reduced to a 3-hours experiment, where a single sheep was released onto a plain field, being its activity recorded on video. At the same time, the collar continuously retrieved time stamped raw sensor data with a frequency of 4 Hz (among the available options, it was a trade-off between data readability and data granularity) and sent it into the IoT network in order to be stored and manually classified[2].

The recorded videos were analysed and all observations were classified into one of the following categories[3]:

- *Resting:* the sheep is standing, with its head facing forward – leftmost image of Figure 6.1;
- *Grazing*, the sheep is eating off the ground, with its head lowered - middle picture of Figure 6.1;
- *Infracting*, the sheep is standing and reaching up for food - rightmost image of Figure 6.1;
- *Walking*, the sheep is moving at low speeds;
- *Running*, the sheep is moving at higher speeds, probably to move away of some potential threats (human presence, unexpected movement, etc.).

[2]In fact, when this experiment took part, the IoT network was only composed of a single beacon directly connected to the gateway. This gateway both stored and sent the raw data to the computational platform to be henceforth processed.

[3]Albeit we were only interested in *Infracting* and *Not Infracting* states, it was decided to consider a wide range of behaviours during the manual classification since it would enable, if required, the reuse of the dataset to perform more exploratory work without the need of reclassify the entire dataset.

As we were only interested in the *Infracting* and *Not Infracting* states, the dataset was re-reclassified such that only the behaviour defined as “standing and reaching up for food” was considered the *Infracting* state. All the remaining cases were reclassified as *Not Infracting*.

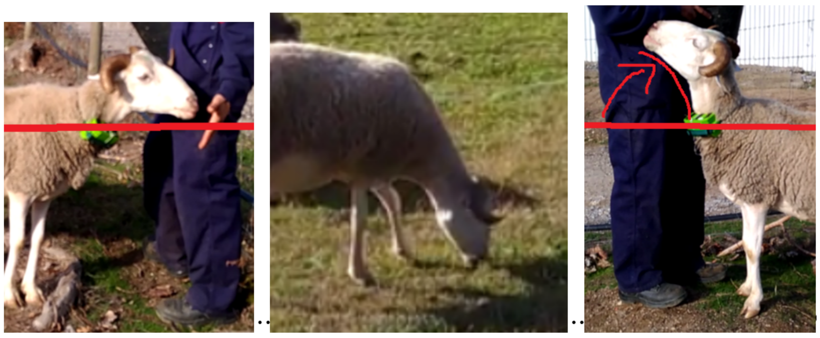


Figure 6.1: Sheep’s behaviour classification: *Resting* (left), *Grazing* (middle) and *Infracting* (standing reaching for food) examples.

Regarding feature transformation and feature selection, no method was trained and tested. All features were considered precisely as received from collars. Such approach was premeditated since the ideal scenario would be minimizing the needs of features transformation within the collar. Thus, a total of 7 features were considered, namely: measured ultrasounds distance and, static and dynamic accelerations on the three axis.

As processing time and complexity were not a concern at that moment, namely because they were not a requirement for this particular work, different ML algorithms were experienced, no matter their computational complexity or processing time. With this analysis we intended to assess with which accuracy sheep’s posture infractions could be detected and which features are relevant to a proper detection. Particularly, we tested some of the most popular algorithms used in classification problems, for instance: RDF, DT using different implementation packages (C50 and *rpart*), XGBoost, NN, SVM and Naïve Bayes. The insights provided by the results of this work could be then used both to improve collars operation as well as obtain more experience with sheep behaviour and data produced by collars while being carried by sheep.

6.1.2 *Infracting*, *Resting* and *Running* states

As it will be clear in Chapter 8, detecting exclusively the *Infracting* and *Not Infracting* states is not sufficient to enable the implementation of a suitable posture control mechanism. Consequently, aiming at minimizing this concern, two additional states were added, the *Resting* and *Running* states. In fact, the experience acquired with the previous work showed that there are behaviours that, besides being classified as *Infracting* due to the position of the neck, are not in fact *Infracting* behaviours. Two noticeable examples are instants when sheep are standing with the head up but without performing any movement, and when sheep are running with the head up but without any chance to feed due to their velocity of movement.

Therefore, a second work [19] established a different approach towards an effective posture monitoring algorithm. Besides the detection of the *Infracting*, *Resting* and *Running* states, this work also considered the need of providing a mechanism that could be integrated in collars. Hence, contrary to the work present in Section 6.1.1, which did not considered any constraints in terms of computational power and complexity when selecting ML algorithms, in this approach, the computational power demands was defined as a requirement. Consequently, DTs were preferred, due its understandably and suitability to be converted into embedded code to be running in low-power microprocessors. Also, some feature transformation was included. Particularly, in addition to the 7 features used in the previous work, two additional features were considered for evaluation, namely the modulus of each dynamic acceleration component, and the magnitude of the dynamic acceleration vector (calculated using Equation A.10, with $N = 1$).

The data collection procedure followed a strategy similar[4] to the one presented in Section 6.1.1. A collar was placed on a sheep pasturing on a plain field and the collar was programmed to retrieve and upload posture-related data at a rate of 4 Hz. Simultaneously, sheep's behaviour was recorded on video to enable a further manual classification using the video recordings and following the same classification rules as the previous work.

From this point on, three binary classification problems were considered to be evaluated and that required different reclassifications:

1. *Infracting* and *Not Infracting*, where *Not Infracting* were all the states unless the *Infracting* state;
2. *Resting* and *Not Resting*, where *Not Resting* were all the states unless the *Resting* state;
3. *Running* and *Not Running*, where *Not Running* were all the states unless the *Running* state;

All the tree binary classifications problems were modeled using DTs and their results combined to construct a merged decision tree. The associated results are presented in Section 8.3.1.2.

6.1.3 *Infracting*, *Resting*, *Grazing*, *Moving* and *Running* states

Previous works unveiled important limitations when resorting to a small number of behaviour states to endorse the requirements of a sheep posture control algorithm. In fact, detecting efficiently the *Infracting* state also demands the detection of other behaviours, such as the *Grazing* or *Eating* states, and the *Resting*, *Moving* and *Running* states. This is supported by the high number of FP and FN regarding the *Infracting* state that were experienced during preliminary experiments (see Sections 8.3.1.1 and 8.3.1.2). The causes for these occurrences are:

- The *Running* state is not enough to minimize the number of FP (regarding the *Infracting* state). In fact, it is common to have sheep moving from on place to another (without being running) with their head up, behaviour that was being identified as *Infracting* state. Thus, detecting the *Moving* state became relevant;
- Models resulted from previous experiments relied in thresholds for a small number of features - two thresholds for the module of dynamic acceleration and one threshold for the pitch angle. Albeit their suitability to be easily implemented in collars since they use a small number of features, they become very sensible to errors that may occur on measurements.

[4]Despite the limitations of this strategy, at the time of this work, there were not available conditions to overpass these limitations. Particularly, there were several constraints with the animal availability that precluded the definition of a data collection procedure with a higher number of animals and with a higher duration.

Consequently, a wider analysis was designed, particularly taking into consideration the use of a higher number of measurements, features and animals. The experiment design followed a strategy employed by most of the works in the area of animal monitoring, although none of existent works explored the detection of the *Infracting* state. The outcome of this work was published in [16] and followed a typical ML-problem solving, as described in Section 2.3.1.

6.1.3.1 The experiment

The experiment design was based in the platform that is being commercialized by iFarmTec [245] for sheep monitoring and control and that was based in the prototype developed in the scope of the SheepIT project. This allowed us to overcome one critical limitation felt in previous works, the lack of data from different sheep in a real scenario deployment. Hence, during the collection phase, data was fetched from a deployed system in an iFarmTec client's livestock farm.

The grazing area was a vineyard in the Bairrada's region (center of Portugal) with around 2.5 ha, with the vine lines separated by around 3 m and with a fence surrounding the whole area, ensuring an adequate and secure grazing. Typically, sheep graze on this parcel throughout all the year and are only taken out for a few weeks during the budburst phase, *i.e.,* in the beginning of spring, to prevent the destruction of the fragile vine buds.

The experiment was conducted over a period where the flock grazed under daily conditions and lasted for 3 days (from the 15th to the 17th of May 2019, hence, well after the budburst phase). Furthermore, each testing day was split into two test windows: morning (from 09:00 to 12:00) and afternoon (from 13:00 to 17:00). During these periods, sheep grazed as usual with the iFarmTec system incorporated, with one different animal being video-recorded per test window to simplify the classification process. To ensure a correct synchronization between the timestamp of the video and the gateway responsible for storing the data, the video recording device and the gateway were temporally synchronized.

6.1.3.2 Applying the ML workflow

Solving a ML-based problem is supported by a common workflow (Section 2.3.1), composed of a several iterative steps or phases. Notwithstanding, within each stage, several strategies, techniques, methods or tools may be used according to the specificities of the problem. Hereinafter, the strategies used in each phase to solve the problem are detailed.

The **collection of data** was done using the monitoring platform described in Chapter 4, despite some required adaptations to allow the collection of data at a higher rate. The configuration of the system deployed on the livestock farm allows the collection of a new trio of dynamic acceleration values (each trio is composed by the x, y and z components of the dynamic acceleration vector) every 20 ms (*i.e.* with a frequency of 50 Hz). The choice of this value was a trade-off between the options offered by the accelerometer being used (that has the following available configurations: 10 Hz, 50 Hz, 100 Hz, 200 Hz, 400 Hz and 800 Hz) and the range of values reported in the state-of-art that go from the 8 Hz to 100 Hz [193] when the target is animal monitoring. Therefore, we choose the second lowest frequency since it is within the range of frequencies reported in the state-of-the-art and because lower frequencies imply lower power consumption. In this work, the option did not fall into the 10 Hz frequency since we were interested in collecting more data without increasing significantly power consumption, but future work shall evaluate this option.

Each new value was not immediately sent through the wireless network, but stored in the internal buffer of the accelerometer until it gets completely filled (the buffer allows the storage of 25 trios). Whenever the buffer is filled, the stored data, that include data concerning dynamic acceleration measurements, a sample of the static acceleration on the three axis (and the respective pitch, roll and yaw angles) and the distance from the sheep's neck to the ground, were sent through the wireless network until reaching the gateway. Here, the data was decoded and stored for further processing.

To enable the implementation of supervised learning techniques, it was necessary to create conditions for classifying the generated data. Hence, all observations were video-recorded using a open source camera application for Android [246] that automatically generates a subtitle file with the associated current timestamp. The resulting files (one per test window) were downloaded from the gateway, the videos associated with each observation were watched and the observations classified following the dictionary described in Table 6.1. Dubious observations or observations that could not be classified (*e.g.* an animal out of sight) were classified with a *X* to be ignored.

Table 6.1: States differentiated during sheep behaviour classification in the last experiment.

State	**Description**
Infracting (I)	Eating from branches above a certain height. For this test, it was defined the height of the irrigation tubes (~50 cm)
Eating (E)	Sheep is eating with its head down. Smooth movements were allowed since typically they seek for grass while moving
Moving (M)	There is a notorious and intended movement from one place to another. Typically, while this happens, sheep is not seeking for food. Trotting was considered moving.
Running (R)	Sheep is running (running away from an obstacle or trying to reach the remaining herd)
Standing (S)	The head is still, *i.e.,* sheep is steady and still
Invalid (X)	Dubious observations or unclassifiable observations

After the manual classification of the dataset, the available original features were:

- *timestamp* - added in the gateway;
- *classification* - added manually during the data classification;
- distance measured using the ultrasounds transducer (*dist.mm)*, the static acceleration on axis x, y and z (*acc_x*, *acc_y* and *acc_y*), the respective angles (*pitch*, *roll* and *yaw*) and the 25 measurements of the dynamic acceleration (on all the 3 axis) - coming directly from the collar.

Two distinct types of features are hence identified: static and dynamic. Static features are single values, *i.e.*, for each timestamp, features take a single value. Dynamic features are a sequence of values, *i.e.*, for each timestamp, features have a sequence of values. This latter kind of features was added to capture the impact of movements and rapid accelerations. Although dynamic features could be used as a whole, there are several benefits in summarize them into single values. Firstly, it enables the control of the number of features. Secondly, it enables the reduction of the size of the resulting model. In this specific scenario, the ML model has to run on a low-power microcontroller, thus low-processing models must be considered.

Taking advantage of the provided state-of-the-art and the provided background in Chapter 2, eight main features were selected for application during the **feature transformation** phase, both to the magnitude of each dynamic acceleration vector and to the rate of change (1st derivative) of that magnitude[5], particularly: minimum, maximum, average, variance, standard variation, Mean Crossing Rate (MCR), Dominant Frequency (DF) and MV. All these features are described in Section 2.3.2.2, as well as the correspondent equations.

Additionally, the unbalanced nature of the dataset due to the sheep natural behaviour and the potential spatial and temporal relationship between states, imposed the seek for additional features that could confirm such relationship. Thus, three additional features were added to the already identified features:

- The previous state (*prevState*);
- The identification if there was a transition from the previous state *(transition)*;
- The number of consecutive and equals states (*nEqualStates*).

In short, the dataset ended up with a total of 27 different features. When a high number of features are available, feature selection techniques shall be applied in order to reduce unappropriated redundancies and to remove noisy or faulty features. Among the different feature selection methods available in the literature, most of them have implementations integrated in several data processing tools and softwares. Hence, the approach followed in this work was based in two main phases:

1. The correlation between features was evaluated such that the most correlated ones could be removed;
2. The algorithms identified in Table 6.2 were used to test feature's importance. These algorithm are available in the *FSelector* package [247] for R.

To endorse the impacts of each technique in a further modelling stage, two different evaluation tasks were considered, namely:

1. **Repercussion in the total accuracy:** for each technique, a DT model was trained and tested, and its total accuracy calculated. At this stage, the data split process consisted of a random split in a proportion of 70% for training and 30% for testing. This simplistic process, allowed a faster identification of the scenarios with the highest potential to be evaluated;
2. **Consistency of the selected attributes between techniques:** the consistency of the results was evaluated among techniques, *i.e.*, besides neglecting techniques which results are dubious, the repeatability of features between techniques was considered.

During the feature selection phase, a random split was used to split data to reduce the complexity of the process. In contrast, during the modelling phase, the **data splitting** consisted of an implementation of a 10-fold cross-validation, maintaining the distributions of the classification labels roughly the same as the existing on the original dataset. The goal with this strategy was to determine if the model being tested would be capable of being generalized to other datasets, minimizing the chances of creating bias points in the dataset. Furthermore, as an unbalanced dataset was expected, upsampling and downsampling techniques were experimented, always after splitting the dataset.

[5]The MV was calculated directly using the dynamic acceleration components (x,y and z axis), being the unique feature that was neither related to the magnitude, neither related to the 1st derivative.

Table 6.2: Feature selection functions selected from the *FSelector* package [247].

Function Name	**Description**
CFS Filter	Resorts on the correlation and entropy measurements
Chi-squared filter	Weights discrete features using a chi-squared test
Consistency-based filter	Measures the consistency of the features
OneR algorithm	Associates rules between features and the class feature
RandomForest filter	Weights features applying the RandomForest algorithm
RReliefF filter	Weights features through the distance between instances

Implementing a continuous monitoring mechanism on a low-power and low-processing microcontroller restrains the kind of ML algorithms that can be used during the **modelling phase**. Thus, the present approach focused entirely in the application of DT algorithms because they allow the extraction of a set of conditions, easily transposed to a set of *if's* and *elses's*, that can be efficiently incorporated in a constrained device, such as the collars of the system being presented. Moreover, they allow a straightforward human interpretation.

6.2 Posture control

Letting sheep weed vineyards without continuous human supervision while maintaining the vines safe, demands a posture control mechanism that shall continuously monitor sheep's posture and trigger the necessary stimuli when required. The monitoring mechanism design was addressed in the previous section, while the conditioning mechanism and its combination with the monitoring mechanism (the posture control) are now discussed. However, before going into details, a relevant clarification about the methodology followed must be given. Conditioning sheep's posture and behaviour requires the use of certain stimuli whose application must be performed carefully. Thus, it must be clear that all methods hereby presented and discussed were supported by scientific works and livestock experts that supervised the stimuli application. Even though, it is out of the scope of this thesis to define and perform experiments on animals to define the final characteristics of the conditioning mechanism. Also it is out of scope determining the effects of the suggested method on animals health and well-being. In fact that tasks were performed within the scope of the SheepIT project by livestock experts and veterinarians that supported the main decisions regarding the conditioning mechanism.

6.2.1 Tools and processes

Whenever an undesired posture is detected through the monitoring mechanism, the conditioning mechanism is invoked to induce the animal to revert its behaviour. This conditioning mechanism comprises two main challenges. On the one hand, the selection and configuration of actions or stimuli that shall compose such mechanism. On the other hand, the definition of how the conditioning mechanism shall interact with the monitoring mechanism.

Regarding the selection of stimuli to be included in the conditioning mechanism, our choices are based in literature studies (see Chapter 3) that point out that a sequence of cues, including warning signals before further penalization, are the most effective. Likewise, the literature reveals that the conjugation of audio as a cue, eventually followed by a small electrostatic discharge, as penalization, is the most effective combination [223].

Consequently, the conditioning mechanism proposed for the collar was also designed to support both type of stimuli. It enables the configuration of the sequence of stimuli as well as their characteristics. Particularly, two parameters were made available, much according to the literature insights but also considering insights acknowledged by livestock experts. For the audio cues, the frequency of the sound wave (associated with the tone of the audio) and the duty cycle (associated with the amplitude of the sound) of the sound wave can be configured. Regarding the electrostatic stimuli, the duration and the duty cycle of the discharge (associated with the power of the discharge) are the configurable parameters.

Notwithstanding, the effectiveness of using cues combined with subsequent penalizations is closely dependent on a training process. This training process corresponds to sheep's cognitive capability of associating a forbidden behaviour to a specific penalization (electrostatic discharge) as well as this penalization to a previous warning signal (the audio cue). When successful, this process allows sheep to revert the behaviour when warned, minimizing or avoiding penalizations. However, the effectiveness of this training process is constrained by the accuracy of the monitoring mechanism and by the correctness of the stimuli application. In fact, sheep have a limited (and variable) cognitive capacity, being crucial to the success of the training process that cues and penalizations are always applied at the right moments. This implies employing a monitoring mechanism with a limited number of FP and FN, as well as it implies the application of stimuli with small latency. The former issue is addressed by the requirements of the monitoring mechanism detailed in the last section, while the later is supported by the collar architecture and by the posture control mechanism state machine described in the following section.

6.2.2 Posture control state machine

Conditioning sheep's posture it is not just about triggering actuators when undesired conditions are detected. In fact, we already pointed out the relation between the cognitive capacity of sheep, the training process and the effectiveness of the posture control mechanism.

Hence, we designed a state machine that implements the posture control mechanism, being presented in Figure 6.2. Periodically, and according to collar's operation, sensors are read and their outputs are given as input to the monitoring mechanism to assess if the animal is incurring on an Infracting state (*INF* signal).

While the animal is not in an Infracting state, the state machine remains on the *Iddle* state. When the animal adopts an infracting posture and it is detected by the monitoring mechanism, the state machine transits to the *Cue* state and starts applying the audio cue (sound sequence). If the animal reverts its behaviour, the state machine returns to the *Iddle* state and the audio cue is stopped. If the animal persists on the incorrect behaviour for a certain amount of time (t_CUE), three transitions may occur. If the next stimulus on the configured sequence (seq_{i+1}) is of type *CUE* or *PEN*, the state machine persists in the same state (*Cue*) or transits to the *Penal* state, respectively. In contrast, if there are no more stimulus in the sequence, it means that all stimuli were applied and the state machine suspends the operation and remains blocked. In the case it transits to the *Penal* state, an electrostatic stimuli starts to be applied. As previously, if the animal reverts its posture, the application of electrostatic stimuli is stopped and the state chart returns to the *Iddle* state. If the animal remains in the *Penal* state for a given duration (t_PEN), but there are still stimulus in the sequence of stimuli, the state machine can either transit to the *Cue* state or remain in the *Penal* state, according to the value of seq_{i+1}. Contrarily, if there are not more stimulus in the sequence ($seq_{i+1} = NULL$), the system suspends all stimuli and remains blocked (*BLK*). This scenario corresponds to undesired situations, that can be either associated to an improper training process or to the existence of refractory sheep[6]. In these cases, the state machine blocks to prevent the unnecessary application of penalizations that would unnecessarily overstress the animal.

As it can be perceived, the stop condition of the state machine behaves as a safety mechanism for the sheep that is configured *a priori* by the human operator. This configuration is reflected in the *!BLK* signal, that together with a *!INF*, unblocks the conditioning mechanism. This feature is considered vital both for the training process and well-being of the animal. In fact, the insights of animal experts in this field defend that after having defined a suitable configuration of stimuli (both sequence, type and characteristics), if the animal does not respond after a sequence of stimuli, probably it will never do it. In this case, besides suspending the mechanism, the most appropriate solution is to rule out that animal because, probably, it is refractory.

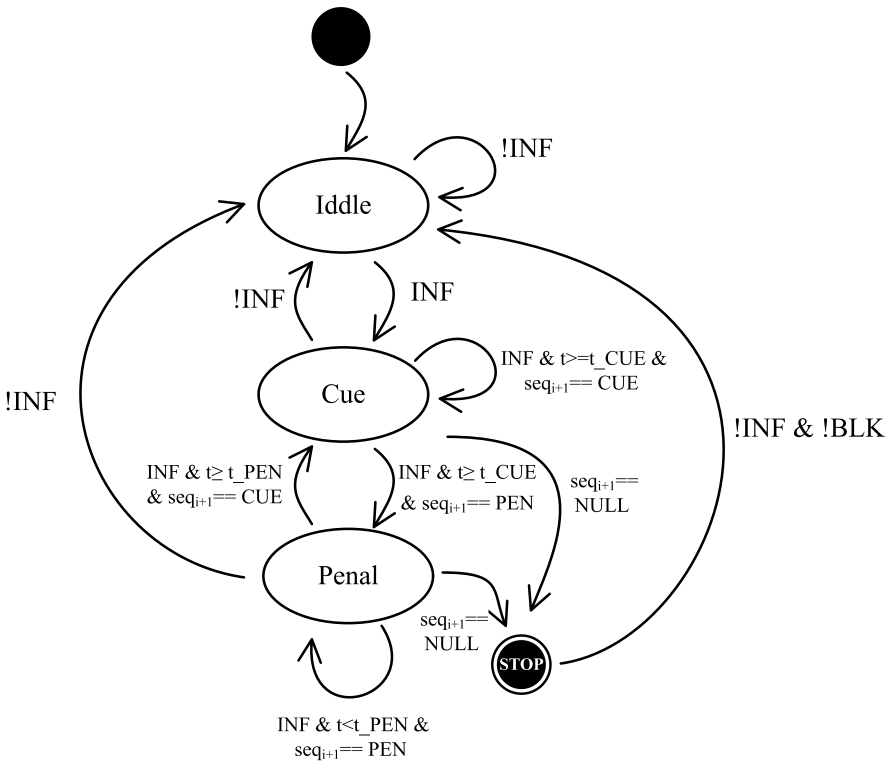

!INF
Iddle
!INF
INF
!INF
Cue
INF & t>=t_CUE & seq_{i+1}== CUE
!INF & !BLK
INF & t≥ t_PEN & seq_{i+1}== CUE
INF & t≥ t_CUE & seq_{i+1}== PEN
seq_{i+1}== NULL
Penal
seq_{i+1}== NULL
STOP
INF & t<t_PEN & seq_{i+1}== PEN

Figure 6.2: Posture control mechanism state machine.

6.2.3 Posture control integration with communications state machine

Last section detailed how the posture control is implemented, particularly how actuators are triggered when the monitoring mechanism detects prohibited behaviours on sheep. However, we did not address yet how such process integrates into collars operation, particularly, how we ensure the correct operation of the posture control mechanism without jeopardizing the communication scheme.

[6]It is common to find animals among different species that either do not have cognitive capacity to get through the training process or do not react when faced to stimuli.

From the communications point of view, a μC is composed of a SW, a TAW and a VTW. Collars internal operation aligns with such structure but by means of two main states, the *Rx state* and the *Active state*, as reveled in Figure 6.3. The *Rx state* corresponds to the SW, where collars are continuously in the receiving state and waiting for data from beacons. The *Active state*, on the other hand, is when collars read sensors and activate actuators, when necessary, following the state machine described in Section 6.2.2. We call to this state, a Nano-Cycle (nC) since it may be repeated several times from the end of the SW up to the end of the μC. As one can see, the TAW that comes right after the SW in the μC structure is omitted in collars internal operation. In fact, although existing from the communications perspective, it does not impact in collar's operation, *i.e.*, there is still a TAW where no communications occur, but collars enter in the active state right after the SW.

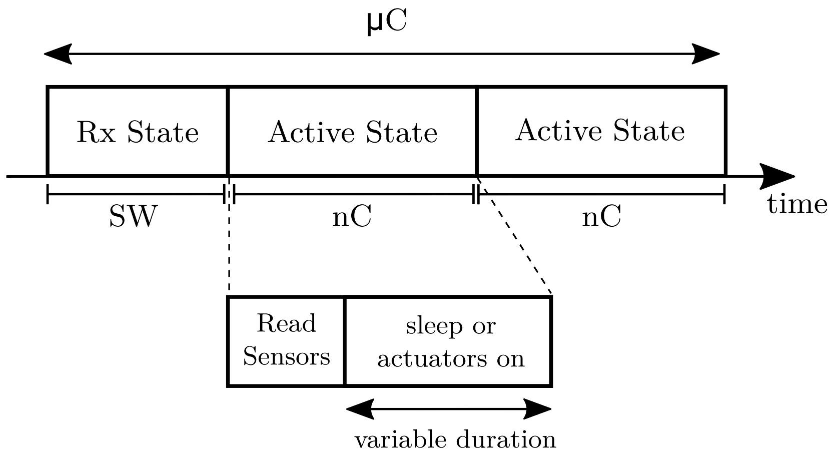


Figure 6.3: Collars internal operation structure within a μC. Also a cyclic structured is used, composed of several nCs where sensors are read and the actuators are triggered according to the posture control mechanism.

A nC corresponds to the collars operation unit. It starts with a window devoted to sensors reading. Albeit sensor measurements are static, the duration of this window is not static since it depends on the distance measured by ultrasounds transceiver. Even though, to simplify calculations, a static and pessimistic value for its duration is considered for this window. The data read from the sensors feeds the posture control mechanism that decides if a stimuli must be set. If so, the first from the sequence is given in the window *sleep or actuators on* identified in Figure 6.2.2. If there is not, the collar enters in the sleep mode until the next nC or μC. Therefore, this window may vary in size, depending on the action being taken and on the stimuli duration. Defining $buzzer_{duration}$ and $electroStim_{duration}$ as the duration of an audio cue and of an electrostatic discharge, the minimum duration for such window ($stimWindow_{duration}$) is given as the maximum among $buzzer_{duration}$ and $electroStim_{duration}$ (Equation 6.1).

$$stimWindow_{duration} = min(buzzer_{duration}, electroStim_{duration}) \tag{6.1}$$

As we assumed a *read sensors* window with a fixed duration, we can define the minimum duration of a nC as depicted in Equation 6.2:

$$nC_{min_{duration}} = readSensors_{duration} + min(buzzer_{duration}, electroStim_{duration}) \tag{6.2}$$

Having the minimum duration for a nC, we are able to calculate the maximum number of nC that fit in a μC as shown in Equation 6.3. We assume the worst case scenario, which corresponds to a C2B μC where a collar needs to send a C2B message.

$$N_{nC_{max}} = \left\lfloor \frac{VTW_{duration} - TS_{TX_{C2B}}}{nC_{min_{duration}}} \right\rfloor \tag{6.3}$$

The use o nCs allows to maintain a TT approach and take advantage of the inherent benefits. Particularly, we can control the timing of each action and ensure periods of sleep that are critical towards the low-power needs. Also, defining a maximum number of nC enables the implementation of a further mechanism to define different levels of reactiveness. For instance, in certain conditions, we may be interested in reducing the frequency of measurements, for instance because sheep is asleep or is keeping a correct behaviour for a long period of time. For that, we can reduce the number of nCs to a value lower than $N_{nC_{max}}$, which means that N_{nC} is always a integer between [0 $N_{nC_{max}}$], where 0 corresponds to the inexistence of measurements and 1 to the maximum number of measurements, *i.e.*, to the maximum number of nCs. The N_{nC} may be defined statically during system configuration or it may be adjusted during system operation according to criteria defined during system configuration.

6.3 Summary and discussion

The posture control mechanism is one of the pillars of this thesis. Besides being the main step to answer to the research question 3, it is fundamental to support our thesis statement.

The posture control mechanism is composed of two orchestrated mechanisms, the behaviour monitoring and conditioning mechanisms. The former was addressed resorting to the potential of ML to detect patterns that are not easily identified by a human analysis. Three iterations were taken, with increasing complexity and completeness. That decision was supported by the need of, on the one hand, acquire know-how on sheep monitoring that initially was very limited, and, on the other hand, maintain the complexity of such mechanism reduced as much as possible.

The output of the monitoring mechanism is then used as an input for the conditioning mechanism. This conditioning mechanism was designed considering two types of stimuli, audio cues and electrostatic discharges. Such cues are triggered according not only to the output of the monitoring mechanism but also according to the sequence of stimuli defined and to the number of stimuli already triggered. The referred sequence of stimuli and some parameters inherent to stimuli can be remotely adjusted, using the network. This approach allows livestock experts to further investigate which would be the most suitable combination for each breed or even for each animal.

Besides exploring both monitoring and conditioning mechanisms, it was necessary to ensure their integration into collars operation without jeopardizing existing communications. Therefore, a TT approach was also followed, maintaining a cyclic structure where sensors are read and stimuli applied in consecutive nCs, whose number may be adjusted according to the needs of the system.

CHAPTER 7

System dimensioning calculator

The implementation of the system's architecture comprises different layers and modules that touch different technological areas. Being unviable to address all topics in the scope of this thesis, the scope was narrowed to the validation of the system's architecture. In particular, we aim to prove that the system design enables to implement a solution that fulfills all requirements.
After specifying the system's architecture and its implementation in previous chapters, we seek now for providing a way of modelling and validating one of the core fmodules of the work, the communications mechanism. Hence, we model the system and develop a simulation tool to enable, on the one hand, the evaluation of the impact of varying several parameters on system performance, and, on the other hand, to offer a way of easily evaluate the feasibility of different use case scenarios. This chapter identifies the requirements of this tool and describes its architecture and implementation. Particularly, we detail the options available on this tool and how they enable the evaluation of system's architecture. Also, the calculations involved are presented. This tool is the base for gathering most of the results presented in Chapter 8.

7.1 Requirements

Evaluating the practical implications of varying all system parameters requires a tool capable of simulating, as close as possible, a deployed system. In fact, it is impractical and unaffordable (both in terms of time and resources) to configure and test in real scenarios, for instance, thousands of collars or dozens of beacons. Moreover, for those whom would be responsible for system selling, it is valuable to have a way of validating scenario conditions and propose scenarios taking into consideration the metrics most valuable for the client. Therefore, a tool capable of modelling and simulating a field deployment was built to give answer to these two-fold type of needs. This tool has the following requirements:

- Enable system modelling according to system's architecture, requirements and implementation detailed in Chapters 4, 5 and 6;
- Allow the calculation of the minimum duration of a μC and the maximum number of collars permitted for different combinations of system parameters;
- Empower the evaluation of different scenarios, by testing different values for system parameters. Particularly, it shall bear two different testing approaches:
  - System parameters are introduced by an user through an input interface. This approach shall allow to evaluate specific configurations, being particularly useful for evaluating scenarios proposed by project partners or users;

  - System parameters are autonomously settled and varied from a set of values previously defined. This approach is particularly relevant for evaluating the behaviour of the system considering different combinations of parameters.

- Enable the validation of a scenario that is composed by a conjugation of different system parameters;
- Calculate different metrics such as medium occupation, energy consumption and autonomy of collars (see Section 7.4) together with system's scalability;
- Permit the analysis of the effects of the number of stimuli on the energy expenditure of collars;
- Support expandability, *i.e.*, allow the integration of new system features without jeopardizing the current version of the simulator.

7.2 Architecture

To fulfil the requirements established in Section 7.1, it was defined the modular architecture depicted in Figure 7.1. The purpose of each module is the following:

- **User Interface:** grants the interface between the user and the simulator. It supports giving inputs and receiving outputs to and from the simulator;
- **Menu:** it is responsible for receiving the necessary inputs. It allows users to select the intended option to be evaluated/validated and collect system parameters necessary to evaluate it;
- **System modelling:** represents the core module of the simulator. According to the option chosen by the user, it models the system. It interacts with other modules of the system, both to gather the necessary information and to provide the necessary answers to the user. For instance, it needs to collect data given by the user through the menu, it needs to load static configurations (from the static configurations module) and it requires data and functionalities from the beacons distribution, messages and energy consumption and autonomy modules;
- **Static configurations**: there are several configurations that are static for a system implementation, for instance radio features such as MTU, baudrate and range, or microcontroller features such as clock drift and CPU speed;
- **Beacons distribution and area calculation:** the number of beacons required to cover a certain area depends on several features. This module, besides storing the configurations regarding beacons distribution (range factor, coverage shape, coverage radius), provides methods such as area and number of necessary beacons calculations;
- **Messages:** this module includes information about the messages currently implemented in the system. All fields that compose each message are identified along with the respective sizes, also provisioning a method for returning messages size;
- **Energy Consumption and Autonomy:** finally, this module is responsible for granting the simulator with capabilities to calculate energy consumption, autonomy and medium occupation.

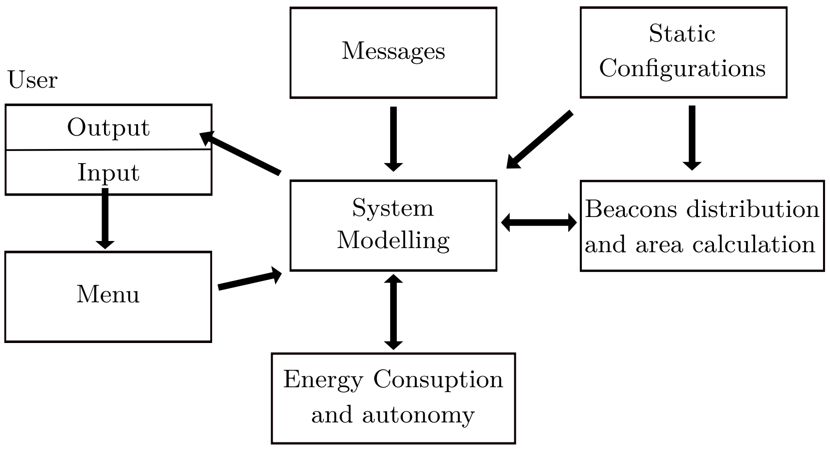

User
Output
Input
Menu
Messages
Static Configurations
System Modelling
Beacons distribution and area calculation
Energy Consuption and autonomy

Figure 7.1: Modular architecture of the simulator.

7.3 Implementation

The implementation of the simulator was performed using Matlab. Figure 7.2 describes the flowchart of the tool.

The first step allows the user to choose between getting system's messages size or evaluate the scalability and associated metrics. If the option is the former, the messages size are returned. If the option is the later, three additional options are prompted to the user. The first regards the evaluation of the minimum duration of a μC, being such process described in Section 7.3.1. The second option permits to evaluate the scalability of the system considering the worst case scenario conditions for an intended area, such as described on Section 7.3.2. Finally, it is possible for an user to validate a particular scenario defined by himself.

7.3.1 Micro-cycle minimum duration

Concerning the evaluation of the μC minimum duration (μC_{min}), Equation 5.10 is used for all calculations. Two options are provided, one where the user introduces the parameters to be evaluated; and another where an autonomous evaluation is performed using different parameter combinations. The parameters requested from the user are identified in Table 7.1 and all must be greater than zero. For the automatic evaluation, the same parameters are required but instead of using values inserted by the user, they are combined from an array of values previously defined. Currently, the contents of this array are defined considering research purposes, particularly to evaluate the system behaviour when configured with different parameters (see Chapter 8).

Table 7.1: Parameters required for the calculation of the μC minimum duration and applied conditions.

Parameter	**Description**	**Condition**
N_{Beac}	Number of beacons considered	> 0
N_{Coll}	Number of collars to support	> 0
$N_{\mu C_{C2B}}$	Number of C2B μC	> 0
$N_{\mu C_{B2B}}$	Number of B2B μC	> 0

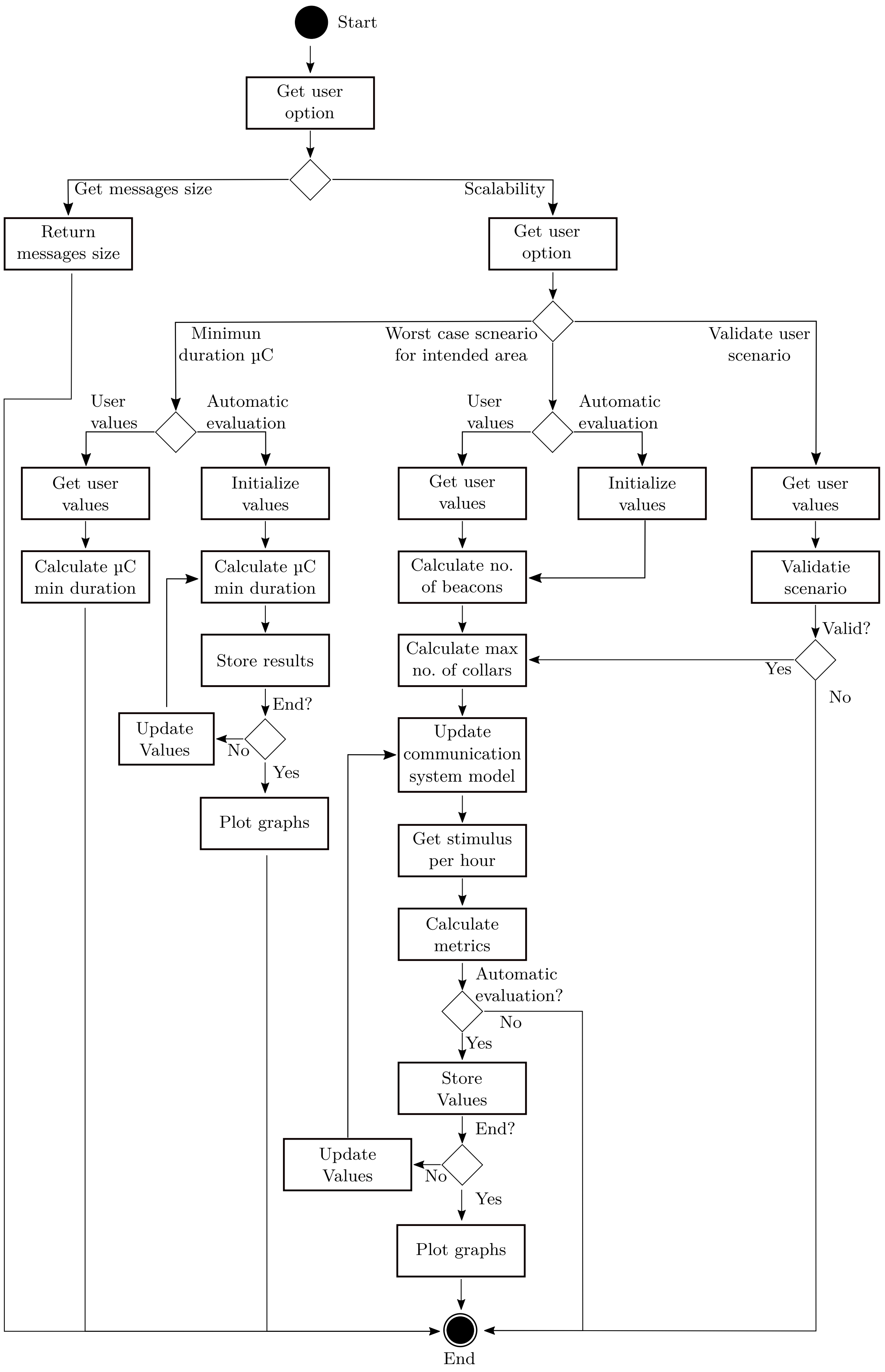


Figure 7.2: Flowchart of simulator's implementation.

7.3.2 Worst-case scenario for intended grazing areas

The rational followed to solve this scenario was as follows:

1. We consider a worst-case scenario on which all beacons transmit once all data to their neighbours, being ensured that beacon communications are scheduled such that new data is propagated to a gateway until the end of the MC;

2. The intended grazing area to be covered by the system is the main input parameter. This value, manually introduced by the user or automatically varied in the case of the automatic evaluation option, enables the calculation of the required number of beacons. Considering a uniform beacon distribution and a hexagonal coverage shape model, the number of beacons (N_{Beac}) is given by dividing the total area to be cover and the area of an hexagon [248]. Equation 7.1 describes that relationship, where $AreaToCover$ is the area in ha and $radiusRadio$ is the range in meters already affected by the range factor ($rangeF$):

$$N_{Beac} = \left\lceil \frac{AreaToCover \times 10000}{\frac{3\sqrt{3} \times radiusRadio^2}{2}} \right\rceil \tag{7.1}$$

3. The MP and the LFP are used as the upper bound limits for the MC and μC duration. Albeit they can also be seen as parameters of the system, they are not expected to vary as the remaining parameters (number of collars, number of beacons, area to graze, etc). Thus, for the option where the user inserts the intended grazing area, the values chosen for those periodicities are 6 s and 80 s, respectively for LFP and MP. Contrarily, within the automatic evaluation option, those parameters are modified from an pre-defined array as for the case of the μC minimum duration evaluation to enable an extensive evaluation of the system's architecture;

4. With the number of beacons and μC and MC maximum durations, an equation with an unique variable (N_{Coll}) can be written as follows:

$$\begin{aligned} MC_{duration} &= N_{\mu C_{PR}} \times \mu C_{duration} + N_{\mu C_{C2B}} \times \mu C_{duration} + N_{\mu C_{B2B}} \times \mu C_{duration} \\ &= LFP + \frac{N_{Coll} \times (TS_{C2B} + GW)}{VTW_{duration}} \times LFP + \\ &+ \frac{N_{B2BSubMessages} \times (TS_{B2BSubMessage} + GW) + TS_{B2BHeader} + GW}{VTW_{duration}} \times LFP, \end{aligned} \tag{7.2}$$

where $N_{\mu C_{PR}}$ was set to 1 (although zero is also possible); $\mu C_{duration}$ is equal to the LFP; $N_{\mu C_{C2B}}$ is given by $\frac{N_{Coll} \times (TS_{C2B} + GW)}{VTW_{duration}}$; $N_{\mu C_{B2B}}$ is given by $\frac{N_{B2BSubMessages} \times (TS_{B2BSubMessage} + GW) + TS_{B2BHeader} + GW}{VTW_{duration}}$; $N_{B2BSubMessages}$ is given by Equation 5.8, neglecting the value of N_{PR}; and $VTW_{duration}$ is given by $LFP - SW_{duration} - TAW_{duration}$. Moreover, TS_{C2B}, $TS_{B2BSubMessage}$, $TS_{B2BHeader}$, GW are known since they are experimentally measured. As N_{Coll} is the unique unknown variable, the equation can be solved. However, the value obtained may not be valid. This is so since additional conditions regarding some equations need to be verified. Particularly, we must ensure that both $N_{B2BSubMessages}$, $N_{\mu C_{C2B}}$ and $N_{\mu C_{B2B}}$ are integers and greater than zero;

5. A validation process is then applied to $N_{B2BSubMessages}$, $N_{\mu C_{C2B}}$ and $N_{\mu C_{B2B}}$ using the value of N_{Coll} calculated from Equation 7.2. This validation process comprises calculating the round up to the nearest integer of the three parameters using N_{Coll} and verifying if the value of

$MC_{duration}$ applying Equation 7.2 is less or equal than MP. If so, N_{Coll} is valid, otherwise it is not and the process is repeated decrementing N_{Coll} by 1. This process is iterated until obtaining a valid output or N_{Coll} reaches zero;

6. When N_{Coll} passes the validation proccess, the respective values of $N_{B2BSubMessages}$, $N_{\mu C_{C2B}}$ and $N_{\mu C_{B2B}}$ are used to update the system model.

Finished the process of calculating the maximum number of collars allowed for an intended grazing area, one additional task is performed. That corresponds to the calculation of some relevant metrics regarding system operation. Their purpose and the process for their calculation are explained in Section 7.4.

This process applies both to the case where the user introduces the area manually and to the case where an automatic evaluation if performed. In this case, the values *AreaToCover*, MP and LFP are chosen from predefined arrays. The values considered for these arrays are addressed in Chapter 8.

7.3.3 Validity of user-defined scenarios

Assessing the validity of an user-defined scenario facilitates the validation of scenarios introduced by the user. This option of the simulator resorts to functions implemented in the context of the addressed features, particularly on the validation and calculation of the number of collars. The main differences are that while in the worst-case scenario for an intended area option, the area is the single input parameter, in the validation of an user scenario, the user is required to fill in the parameters described in Table 7.2.

In the list of parameters, the *AreaToCover* and N_{Beac} appear to be redundant but they are not. The purpose is to provide the opportunity for an user to introduce only an area and automatically know the number of required beacons or to introduce an area and the number of available beacons and check if the number of beacons available is sufficient to cover the inserted area.

Table 7.2: Parameters prompted to the user in the case of the validity of an user scenario.

Parameter	**Description**	**Conditions**
$AreaToCover$	Area to cover (in ha)	> 0
N_{Beac}	Number of beacons considered	> 0 or NULL
N_{Coll}	Number of collars	> 0
$\mu C_{duration}$	Duration of a μC (in ms)	$> TAW_{min} + SW_{min}$
$N_{\mu C_{PR}}$	Number of PR μC	$\{0, 1\}$
$N_{\mu C_{C2B}}$	Number of C2B μC	> 0
$N_{\mu C_{B2B}}$	Number of B2B μC	> 0

The number of collars is also an input parameter, being used to validate the use case scenario. Nevertheless, when the use case is valid, the maximum number of collars is calculated for that scenario. This is useful because besides enabling to validate specific use case scenarios, allows to know if there is any opportunity of expansion, particularly regarding the number of collars. Additionally, for valid use case scenarios, system metrics are calculated.

7.4 Communication evaluation metrics

Energy consumption, autonomy and media utilization are important metrics when evaluating a wireless system of constrained devices. They depend on system features mainly associated with the RF technology (baudrate, clock drift, range, MTU), messages sizes and number of devices. In this section we summarize these relations and provide means to estimate the referred metrics.

7.4.1 Medium Occupation

Medium Occupation (MO) is a metric [249] that allows the evaluation of the effectiveness of medium utilization by wireless devices. In a ideal scenario, considering the maximum number of devices allowed in the network, a high MO percentage means a communication scheme with a good use of bandwidth. However, as it depends on different system configurations (number of collars, number of beacons, duration of μC, number of μCs, beacons distribution, routing mechanism), which in turn depends on the monitoring needs and requirements of the user, its interpretation is not straightforward and must be contextualized within the scenario being evaluated.

Equation 7.3 allows the computation of the MO in a MC interval. The total MO can be computed summing up the individual contributions from all μC MOs. As the amount of data to be transmitted within a PR VTW is on average negligible, its contribution can be ignored. As such, the total MO can be given by the sum of the MO associated to the SWs, TAWs, C2B VTWs and B2B-VTWs. The duration of each one of these windows was discussed in Section 5.2.2, except the term NTS_{B2B} that represents the total number of B2B time slots in use. This value depends on several system configurations, for instance, on the routing scheme and on the total amount of data to be transmitted, that in turn depend on the number of devices and on temporal constraints associated both to the communication scheme and monitoring needs. Equation 7.3 considers the implementation defined in the scope of this thesis. Consequently, the parcel $\sum_1^{N\mu C_{B2B}} NTS_{B2B} \times (TS_{B2B} + GW)$ can be re-written as $N_{B2BSubMessages} \times (TS_{B2BSubMessage} + GW) + TS_{B2BHeader}$.

$$
\begin{aligned}
MO(s) = \sum_1^{N\mu C} MO(\mu C_i) = \sum_1^{N\mu C_{PR}} MO(\mu C_{PR})_j + \sum_1^{N\mu C_{C2B}} MO(\mu C_{C2B})_l + \sum_1^{N\mu C_{B2B}} MO(\mu C_{B2B})_m \approx \\
\approx \sum_1^{N\mu C} MO(SW_i) + \sum_1^{N\mu C_{C2B}} MO(VTW_{C2B})_l + \sum_1^{N\mu C_{B2B}} MO(VTW_{B2B})_m = \\
= \sum_1^{N\mu C} N_{Beac} \times (TS_{BS} + GW + TAW) + (N_{Coll} \times (TS_{C2B} + GW)) + \\
+ \sum_1^{N\mu C_{B2B}} NTS_{B2B} \times (TS_{B2B} + GW) = \\
= \sum_1^{N\mu C} N_{Beac} \times (TS_{BS} + GW + TAW) + (N_{Coll} \times (TS_{C2B} + GW)) + \\
+ \sum_1^{N\mu C_{B2B}} N_{B2BSubMessages} \times (TS_{B2BSubMessage} + GW) + TS_{B2BHeader}
\end{aligned}
\tag{7.3}
$$

For an easier understanding of MO values, they can be given as a percentage as computed by

Equation 7.4, where MO is given by Equation 7.3 and $MC_{duration}$ is the total duration of a MC.

$$MO(\%) = \frac{MO}{MC_{duration}} \times 100 \qquad (7.4)$$

7.4.2 Energy consumption and autonomy

Energy consumption and autonomy are key requirements in any WSN and/or IoT device. The same occurs to this system, particularly for collars, which are the most constrained nodes of the system. Since they are carried by small/medium animals, thus mobile, its size and weight must be bounded to be comfortably carried. Also, operation and maintenance tasks are expected to be carried by farmers and the process of placing and removing collars from sheep is laborious and time-consuming. Thus, the number of battery recharges and/or battery replacements should be minimized.

The main sources of energy consumption in collars are:

- **RF transceiver:** is the main source of power consumption, particularly while transmitting and receiving data, but also in *idle* states. Several transceivers also include one or more energy-saving states, where energy consumption is residual comparing to the remaining states;
- **Microcontroller:** is also responsible for a significant percentage of energy consumption. Despite the existence of an increasing number of low-power microcontrollers, there is a trade-off between low-power and processing capabilities. As RF transceivers, also several microcontrollers include low-power modes;
- **Sensors**: there are several low-power accelerometers, not being, typically, a relevant source of energy consumption comparing to RF transceivers and microcontrollers. Conversely, the ultrasounds transducer, when active, consumes a significant amount of energy, but its duty-cycle, and thus its average consumption, is expected to be small;
- **Actuators:** the impact of actuators' energy consumption in the total energy consumption of the system depends on how many times they are triggered, which is unpredictable. Thus, its impact shall be modelled and evaluated.

For the current simulator implementation, the components and states considered and respective durations for a collar are described in Table 7.3. For the radio component, three states area considered: transmission (Tx), whose duration is equal to the transmission time; reception (Rx), during the SW of all μCs; and idle, in the remaining time. Regarding sensors, the accelerometer is considered to be always on, while the ultrasounds peripheral is active only for certain amount of time ($ultrasounds_{duration}$) every nC. Concerning the actuators, the total duration of a stimulus depends on the number of triggers (N_{buzzer} and $N_{electroStim}$) per MC and on the duration of a single trigger ($buzzer_{duration}$ and $electroStim_{duration}$). For the microcontroller, two states are considered[1]: the running state, during the SW, TAW, reading sensors, transmission of data and while triggering the actuators; and the sleep state, during the remaining time. Finally, also the electrical operation of the system is modelled, through a DC-DC component, that is always on.

The number of audio cues and number of electrostatic stimuli is difficult to model considering a per MC basis. In a system where a low number of stimuli are intended (and even permitted), 1

[1]In fact, there is a third, the idle state that is used in boundary scenarios, particularly when the time in sleep is less than 2ms, however it is negligible.

trigger per MC may be too high. On the other hand, using decimals values is difficult to interpret, particularly to common users. Consequently, the input for the number of stimuli (either audio cues or electrostatic discharges) are given in a per hour basis. These per hour values are then converted by the simulator to number of stimuli per MC.

Table 7.3: Duration of each state component during a MC.

Component	**State**	**Duration**
Radio	Tx	TS_{TX}
	Rx	$N_{\mu C} \times SW_{duration}$
	Idle	$N_{\mu C} \times (\mu C_{duration} - SW_{duration}) - TS_{TX}$
Accelerometer	On	$N_{\mu C} \times \mu C_{duration}$
Ultrasounds	On	$N_{\mu C} \times N_{nC} \times ultrasounds_{duration}$
Buzzer	On	$N_{buzzer} \times buzzer_{duration}$
Electrostatic Stimulator	On	$N_{electroStim} \times electroStim_{duration}$
DC-DC operation	On	$N_{\mu C} \times \mu C_{duration}$
Microcontroller	Running	$SW_{duration} + TAW_{duration} + (N_{nC} \times readSensors_{duration}) \times N_{\mu C} + TS_{TX} + N_{buzzer} \times buzzer_{duration} + N_{electroStim} \times electroStim_{duration}$
	Sleep	$N_{\mu C} \times \mu C_{duration} - \mu C_{running_{duration}}$

As each one of the refereed components operates in different states and with different durations, the Duty-Cycle (DC) for each state and component needs to be calculated. As described in Equation 7.5, the DC associated to a collar component in a certain state ($DC_{compState}$) depends on the time it spends on it ($\Delta t_{compState}$) in a certain timing window. In this case, a MC time window was chosen since it guarantees periodicity.

$$DC_{compState} = \frac{\Delta t_{compState}}{MC} \tag{7.5}$$

With DC values, the average current consumption of the whole system can be calculated as the sum of individual contribution as follows in Equation 7.6.

$$I_{totalAverage}(A) = \sum DC_{compState} \times I_{compState} \tag{7.6}$$

where $I_{compState}$ represents the average current consumption for a component in a certain state.

Finally, considering a battery capacity of Q (mAh) and a battery efficiency of BE, the autonomy (hours) of a collar can be estimated by Equation 7.7.

$$autonomy(h) = \frac{Q}{(I_{totalAverage} * BE)} \tag{7.7}$$

7.5 Summary and discussion

This chapter presented an important tool both to streamline the system evaluation process and to provide a way of testing the feasibility of different scenario conditions. This tool consists on a Matlab application that models system operation and permits some interactions particularly for choosing the intended type of evaluation and introducing system parameters to be considered for such evaluation.

According to the current implementation of the system, the modelling approach and its implementation was explained. Finally, we identified the metrics considered for the system evaluation and that were integrated in the simulator. This simulator is the support of a substantial part of the results presented in Chapter 8 and so enables the assessment of the system architecture proposed.

CHAPTER 8

Results and discussion

In previous chapters the fundamental concepts, mechanisms and tools to tackle the research questions that support our thesis statement were presented and discussed. The IoT architecture design to address all requirements was presented, with special focus on the low-power demands, a tool for facilitating its evaluation was designed and a novel animal posture control mechanism supported on ML techniques was built. Now, we present the results obtained as result of the evaluation of each one of these components and show how they grant the validation of the proposed solution.

8.1 Prototypes and technologies used

Before digging into the results, we introduce not only the prototypes used, but also the technologies used for the implementation of the first versions of the system. Albeit the construction of those prototypes were, to some extent, out of scope of this thesis, they assume a great relevance on showing the feasibility of the system design and consequently on supporting the thesis. Furthermore, as collar's dimensions were a system requirement, it is of utmost importance to show that the system design allows the construction of prototypes matching the minimum requirements.

A collar prototype is illustrated in Figure 8.1. Its dimensions are 7.9 cm x 7.9 cm x 4.1 cm, weighting a total of 190 g, without the strap. Therefore, they are within the required dimensions and weight, being inclusive inferior to the commercial references, particularly to the Dogwatch collar whose dimensions and weight are 10.4 cm x 7.6 cm x 4.8 cm and 265 g, respectively, and to the noFence collar that weights 500 g (dimensions are not provided). Beacon and gateway prototypes are not subject to dimension and weight constraints as collars, even though, their prototypes are depicted in Figure 8.2.

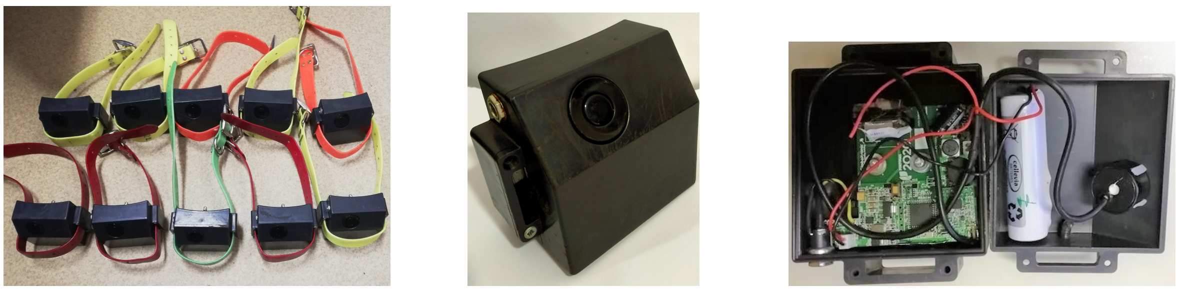
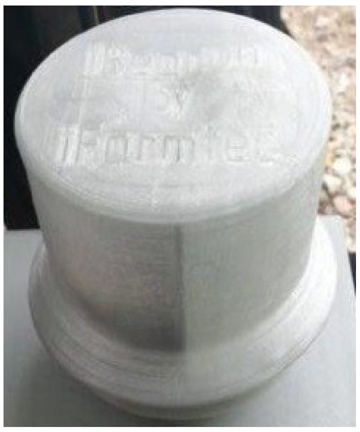

(a) (b) (c)

Figure 8.1: Collar prototype. (a) collars with placing straps included; (b) collar case prototype; (c) inside collar prototype detailed.


(a)

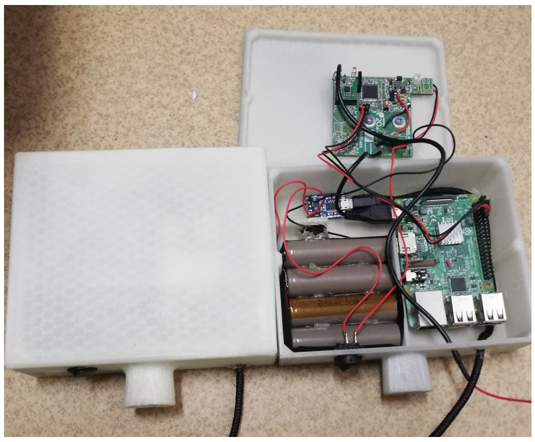

(b)

Figure 8.2: (a) Beacon prototype; (b) Gateway prototype.

In Chapter 4, we identified the composition of all system devices together with their purpose and requirements. We now identify the main technologies used to implement those components as well as their main characteristics.

Collars and beacons share the same printed circuit board, which means that they share most of the components. This allows to minimize both the production and maintenance cost, a critical issue in IoT systems. The main components of the shared electrical circuits are identified in Table 8.1. These components were chosen taking into consideration the system design proposed in the scope of this thesis but also considering practical and production issues that are out of scope of this thesis. Each one of the components owns features that impact on system operation and performance. These features are introduced as they are needed along the chapter.

Table 8.1: Critical components, chosen parts and generic description used to implement collars and beacons.

Component	**Reference**	**Description**
Microcontroller	PIC32MX170F256D [250]	32-bit Microcontroller, with 256 KB Flash and 64 KB SRAM, low-power modes, different communication interfaces (I2C, UART, SPI), Timers and IO ports
Radio	RFM 22B [251]	868 MHz ISM band, output power up to 20 dBm, low-power consumption modes and MTU of 64 B
Accelerometer	LSM303C [252]	Ultra-compact high-performance eCompass, with low-power modes, embedded FIFO and different resolutions
Buzzer	ABT-414-RC [253]	85 dB electromechanical audio traducer, with adjustable output
Ultrasounds transducer	K-14WPP10 [254]	Assembled ultrasonic sensor served by a in-house conditioning circuit

8.2 IoT-based communication stack validation

The IoT-based communication stack comprises four main layers, each one with a specific objective. Albeit all layers are vital to ensure a suitable system operation, some are specially critical for evaluating the feasibility of the solution. Those layers are the Physical Layer, the MAC Layer and the message transport module within the Transport Layer. Consequently, their validation is primordial. The purpose of the following sections goes in the direction of validating the solutions proposed for those layers.

8.2.1 Required parameters

To enable the evaluation of the proposed solution, some parameters need to be known *a priori*. Those parameters are associated to the radio and microcontroller features and configurations, and to the posture control configuration parameters.

The radio used in both collars and beacons is configured to operate at a baudrate of 57600 baud (1 symbol corresponds to 1 bit). The value chosen represents a trade-off between transmission speed and transmission effectiveness and RSSI variability. Albeit being available higher baudrates, the sensitivity of the radio reduces significantly with a relevant impact on the success transmission rate. Furthermore, preliminary experiences with the radio technology have shown that higher baudrates are typically associated to higher RSSI variability, which is undesirable. Still regarding radio operation, its MTU is 64 bytes, thus above the size of a C2B message as desirable. Finally, the typical range of the radio was assumed to be 200 meters [20], although a pessimist *rangeFactor* of 0.5 is considered during all evaluations set. It is not an objective to find the optimal values, but instead use a conservative value in order to validate the feasibility of the solution, even under demanding scenarios. The summary of these parameters is settled in Table 8.2.

Table 8.2: Radio configurations and features.

Radio parameter	**Value**
Baudrate	57600 baud
MTU	64 B
range	200 m
rangeFactor	0.5

Together with radio configurations and features, also the microcontroller influences communication timings. The microcontroller operates at a speed of 20 MHz and includes a 10 ppm accuracy crystal. This is relevant since it is a well-known fact that the efficiency of protocols based on time multiplexing, both in terms of bandwidth utilization and energy consumption, is strongly correlated with the capacity of synchronizing properly the diverse nodes and generating communication events with low *jitter* and skew. Consequently, as previously done for an implementation with different radios and microcontrollers in [11], several experiments were carried out to measure the *jitter* associated with the transmission and timestamping of messages, as well as event generation. Though all *jitter* measurements were below 20 μs, a very conservative value of 1 ms for the GW was considered for building the first prototypes and also during the evaluation.

Still regarding communication timings, all values associated to the time slot definition were measured. These values were measured considering the system prototype developed by iFarmTec and that follow the communication stack designed in the scope of this thesis. Values were measured 5 times considering a normal system operation and the system was restart between measurements. As these values were also measured to be used in a development context, an overestimation of around 5% was considered. This approach allowed to avoid continuous measurements every time a change occurred. Table 8.3 summarizes the measured values and the values considered in the implementation of the system. Regarding TAW's duration, a very conservative was considered. In fact, as different factors may impact on the optimal duration of such window, such as number of collars and beacons to be processed or the GPS readings timeout, a very conservative value was considered.

Table 8.3: System parameters regarding the communication scheme.

System parameter	**Values (ms)**
GW	1.0
$TS_{BS_{RX}}$	0.9
$TS_{BS_{TX}}$	3.7
$TS_{C2B_{RX}}$	5.0
$TS_{C2B_{TX}}$	10.0
$TS_{B2BSubmessage_{RX}}$	2.0
$TS_{B2BSubmessage_{TX}}$	12.0
$TS_{B2BHeader_{RX}}$	2.3
$TS_{B2BHeader_{TX}}$	5.0
TAW	600.0

Table 8.4 compiles the posture control related parameters. These parameters are required to model nCs operation. We identify the duration required to measure all sensors in a worst-case scenario. This worst-case scenario is defined as the time necessary for the ultrasounds module to measure a distance of 1 meter (the expected maximum distance from sheep's neck to the ground) plus the time necessary to gather, store and process the measurements from the accelerometer, including static and dynamic accelerations. Additionally, the number of nCs was set to $N_{nC_{max}}$ (ensuring the maximum allowed number of nCs inside a μC), and buzzer and electrostatic discharge durations were set to 400 ms and 75 ms, respectively.

Table 8.4: Posture control parameters.

Parameter	**Value**
Read Sensors Duration	20 ms
N_{nC}	$N_{nC_{max}}$
Buzzer duration	400 ms
Electrostatic discharge duration	75 ms

Finally, Table 8.5 presents the values required for the estimation of collar's current consumption. When possible, theoretical values were considered, although there are several components on which that was not possible. In those cases, experimental values were used. For that, the respective components were isolated, and the average current consumption measured.

Table 8.5: Collar's components average current consumption. Values identified with a * are measured values.

Components	**Current (mA)**
Radio Tx	85.00
Radio Rx	18.50
Radio Iddle	0.80
Accelerometer	0.39
Ultrasounds*	45.56
Buzzer*	50.00
Eletrostotic stimulator*	638.00
DC-DC operation*	0.50
Microcontroller Running	15.00
Microcontroller Iddle	6.00
Microcontroller Sleep	0.78

8.2.2 Validation of the simulator

To verify the correct operation of the simulator described in Chapter 7, two scenarios were considered for field testing, namely:

- **Scenario 1** - $uC_{duration} = 3.2\,\mathrm{s}, MC_{duration} = 6.4\,\mathrm{s}, N_{Beac} = 10, N_{C2B} = 1, N_{B2B} = 1, N_{PR} = 0$: This was the scenario most frequently used for testing purposes. A smaller μC duration allows to maintain superior supervising accuracy, specially regarding collars internal operation. Furthermore, it allows to get data in the gateway more often, which is crucial during this stage of maturity of the solution;
- **Scenario 2** - $uC_{duration} = 5.0\,\mathrm{s}, MC_{duration} = 45\,\mathrm{s}, N_{Beac} = 16, N_{C2B} = 1, N_{B2B} = 8, N_{PR} = 0$: This scenario was built exclusively to attest the feasibility of the system for a higher number of collars. In practice we were not able to program all collars. The tests were performed by programming the lower and upper bound time slots within the communication infrastructure.

As shown in Table 8.6, regarding the maximum number of collars, the simulator gives a reliable description of the system, since the maximum number of collars given for both scenarios were successfully tested in real scenario conditions. Concerning collars average consumption, variations of 3,38% and 2.27% were observed in scenarios 1 and 2 respectively. We can see that the average current consumption given by the simulator is lower than the measured, which is the result of using theoretical current consumption values regarding individual components of the system. However, the errors are below 5%. Therefore, we can confirm that the simulator correctly models the system behaviour regarding the communication scheme.

Table 8.6: Scenarios used to validate the simulator.

	Simulator		**Validated**	
	N_{Coll}	**Avg. Consumption**	**N_{Coll}**	**Avg. Consumption**
Scenario 1	19	7.11 mA	19	7.35 mA
Scenario 2	207	6.03 mA	207	6.17 mA

8.2.3 μC minimum duration

The calculation of the μC minimum duration is given by Equation 5.10. As it relies on the maximum value among the minimum duration of a C2B and B2B μCs, it is relevant to assess how these parameters impact on the final μC minimum duration according to the defined number of collars (N_{Coll}), beacons (N_{Beac}) and C2B and B2B μCs ($N_{\mu C_{C2B}}$ and $N_{\mu C_{B2B}}$). As such, different values were assessed for each parameter. For the sake of simplicity, all values are taken from the Fibonacci sequence as shown in Table 8.7. The upper bound limits were chosen to cover beyond the expected worst case scenarios.

Table 8.7: Parameters tested on the μC minimum duration assessment.

Parameter	**Values**
N_{Coll}	$\{1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987, 1597\}$
N_{Beac}	$\{1, 2, 3, 5, 8, 13, 21\}$
$N_{\mu C_{C2B}}$	$\{1, 2, 3\}$
$N_{\mu C_{B2B}}$	$\{1, 2, 3, 5, 8, 13\}$

The obtained values were processed and plotted as follows:

1. Results were filtered for a defined $N_{\mu C_{C2B}}$ and $N_{\mu C_{B2B}}$;
2. Data was clustered per N_{Beac};
3. Two data sequences were created and plotted, one associated with the minimum duration of a C2B μC and another with a B2B μC. On the plot, the x-axis represents N_{Coll} and the y-axis represents the μC minimum duration calculated;
4. The process was repeated for all combinations of $N_{\mu C_{C2B}}$, $N_{\mu C_{B2B}}$ and N_{Beac};

An output plot example is depicted in Figure 8.3 (the remaining ones are provided in Appendix B). As expected, the relationship between the μC duration and the number of collars is linear and increases with the number of devices. However, it is clear that there is a set of conditions from which the weight of B2B μC minimum duration prevails over the C2B μC minimum duration.

Analysing the values of Table 8.8, we can see that the turnover in terms of number of beacons (and respective area covered) increases with the $N_{\mu C_{B2B}}$. That is justified due to a higher bandwidth utilization in the B2B VTW comparing to a C2B VTW. In fact, while in the former each collar message is sent in a individual time slot, in the latter, data is fragmented in continuous B2B submessages that are equal to the MTU of the radio (and larger than a C2B message). Therefore, the impact of message headers and GW prevail up to that turnover instant.

Resorting on the same data, we wanted to appraise the μC minimum duration for a N_{Coll} close to the maximum number of animals expected to carry a collar within a grazing vineyard. Hence, we chosen $N_{Coll} = \{610, 987\}$, 987 because it is close to the one thousand devices that was established as the upper bound defined, and 610 because its within the group of > 500 identified in Section 4.1 as being the highest group of animals in an animal census in Ireland. Then, we identified which set of conditions comply with the defined requirements ($MC_{duration} <= 80\,\mathrm{s}$, $\mu C_{duration} <= 6\,\mathrm{s}$ and area covered $>= 5\,\mathrm{ha}$) - see Table 8.9.

For a $N_{Coll} = 987$ only two options comply with the requirements, both needing three C2B μC. With $N_{\mu C_{B2B}} = 5$, a maximum of two beacons can be used (performing a total around 5 ha), with a

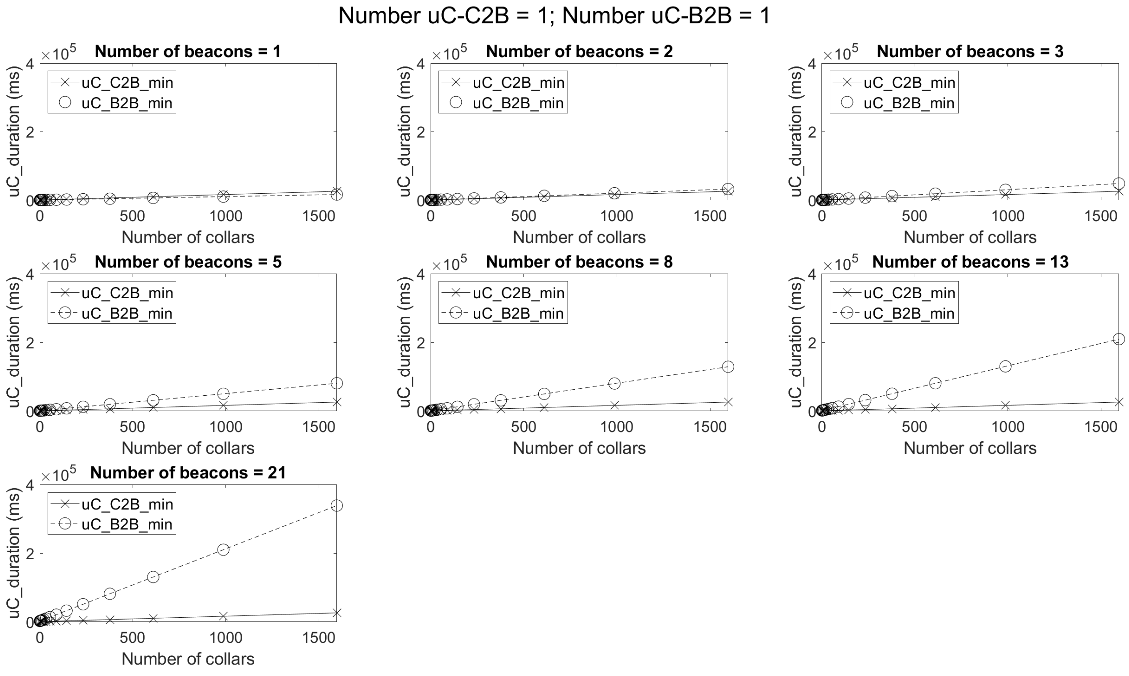


Figure 8.3: μC minimum duration for the combination $N_{\mu C_{C2B}} = 1$; $N_{\mu C_{B2B}} = 1$.

μC duration very close to 6 s and a MC of 53 s. With $N_{\mu C_{B2B}} = 8$, a maximum of three beacons can be used (performing a total around 7 ha), with a μC duration also very close to 6 s and a MC around 71 s. Therefore, although satisfying the requirements (both timings and area to cover), it does not support the addition of more beacons if needed, for example, to increase coverage in shadowing areas.

For $N_{Coll} = 610$, a few additional options are available, with greater flexibility in terms of N_{Beac}. Particularly, the system scale up to 8 beacons, with $N_{\mu C_{C2B}} = 3$ and $N_{\mu C_{B2B}} = 13$. In this case, the associated μC duration is around 4 s while the MC duration is around 75 s.

In sum, considering worst case conditions, a N_{Coll} close to a thousand is supported by the system but it does not scale well regarding the number of beacons. However, N_{Coll} close to a thousand is not likely to happen in real use case scenarios.

Table 8.8: Turnover instants in terms of $*N_{\mu C_{B2B}}$ where $\mu C_{B2B_{min}} > \mu C_{C2B_{min}}$.

$\mathbf{N_{\mu C_{C2B}}}$	$\mathbf{N_{\mu C_{B2B}}}$	$\mathbf{N_{Beac}}$	**Area (ha)**
1	1	2	5.14
1	2	3	7.79
1	3	5	12.99
1	5	8	20.78
1	8	13	33.77
1	13	21	54.55
2	1	-	-
2	2	2	5.54
2	3	3	7.79
2	5	5	12.99
2	8	8	20.78
2	13	13	54.55
3	1	-	-
3	2	2	5.14
3	3	2	5.14
3	5	3	7.99
3	8	5	12.99
3	13	8	20.78

Table 8.9: μC minimum duration for 610 and 984 collars (the closest values to the upper bound value for the maximum number of collars) and the set of parameters that are within the requirements defined (MP <= 80 s, LFP <= 6 s and area covered >= 5 ha). In the case of more than one result, only the one with a higher N_{Beac} is showed.

$\mathbf{N_{Coll}}$	$\mathbf{N_{\mu C_{C2B}}}$	$\mathbf{N_{\mu C_{B2B}}}$	μ**C**$_{\mathbf{duration}}$ **(s)**	**MC**$_{\mathbf{duration}}$ **(s)**	$\mathbf{N_{Beac}}$	**Area (ha)**
987	3	5	5.88	52.88	2	5
987	**3**	**8**	**5.88**	**70.57**	**3**	**7**
610	2	3	5.49	32.95	2	5
610	2	5	5.50	43.97	3	7
610	2	8	5.51	60.59	5	13
610	3	3	4.71	32.95	2	5
610	3	5	4.30	38.72	3	7
610	3	8	4.47	53.61	5	13
610	**3**	**13**	**4.43**	**75.39**	**8**	**21**

8.2.4 Worst-case scenario for intended gazing areas

The previous test, although allowing a quick evaluation about the μC minimum duration for a set of conditions, does not allow to calculate what is, in fact, the maximum number of collars for a set of conditions. Moreover, the previous test does not allow to evaluate how the maximum number of collars evolves for different system periodicities. In this section we fill that gap, assessing different sets of conditions defined by the triple LFP (*i.e.* μC duration), MP (*i.e.* MC duration) and area to be covered (*i.e.* number of beacons). The values used are the ones described in Table 8.10. In this case, the upper bound limits are beyond the expected worst case scenarios in order to give a better idea about the scalability of the system. For each combination of values, the maximum number of collars, the medium occupation and the energy consumption and associated autonomy are calculated using the equations provided in Chapter 7.

Table 8.10: Parameters tested on the worst-case scenario for the intended grazing area assessment.

Parameter	**Values**
$\mu\mathbf{C_{duration}} = \mathbf{LFP}$	$\{1, 2, 4, 6, 10, 15, 20, 30\}s$
$\mathbf{MC_{duration}} = \mathbf{MP}$	$\{30, 45, 60, 80, 100, 125, 150, 180\}s$
Area	$\{1, 2, 3, 4, 5, 6, 7, 8, 10, 13, 21, 34, 55, 89, 144\}ha$

8.2.4.1 Maximum number of collars

For each combination of values of Table 8.10, a subset of graphics as the ones depicted in Figure 8.4 were draw. Fixing the $\mu C_{duration}$, the maximum number of collars was calculated for different areas (the x-axis) and for different $MC_{durations}$. Figure 8.4 illustrates the results for MP = 80 s, the value provided by project partners as the maximum acceptable. The graphics built using the remaining set of combinations are provided in Appendix C. Despite using MC duration values provided in Table 8.10, during the calculations, the final value for the MC duration many not be the same. As a MC is composed of multiple μCs, the MC duration must be a multiple of the μC duration. Therefore, when that did not occur, the corrected value for the of MC duration was also stored.

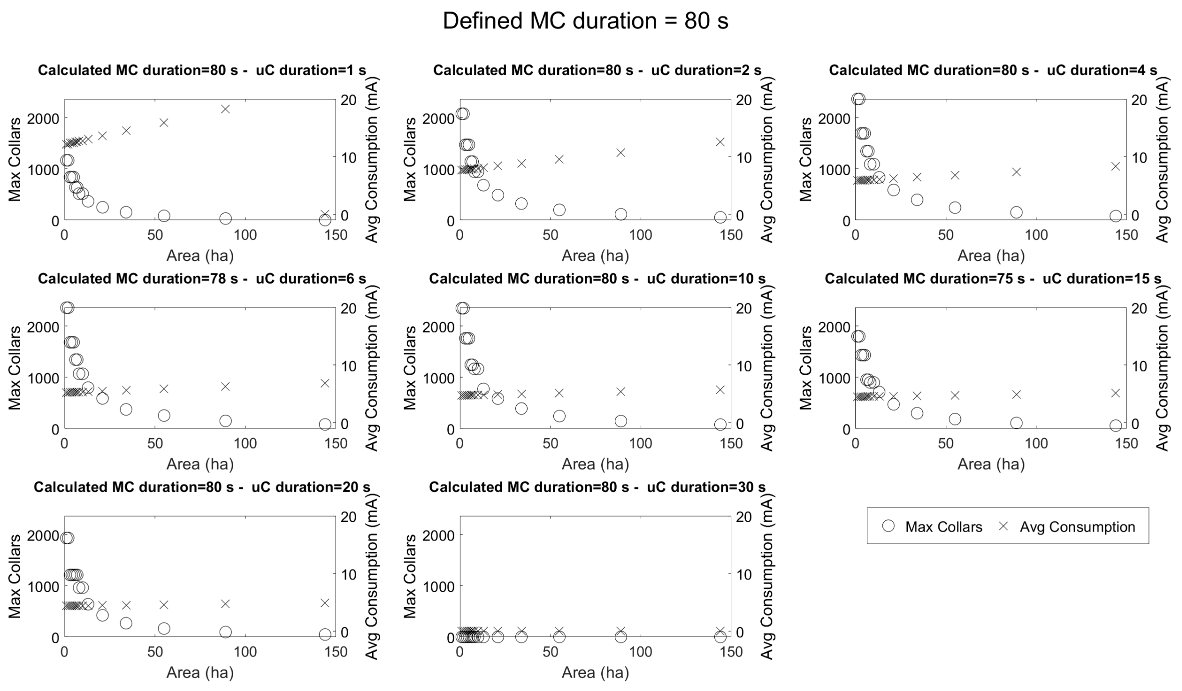


Figure 8.4: Maximum number of collars and average consumption for $MC_{duration}$=80 s. When the points are both close to zero, it means that there isn't a solution and thus the scenario is not feasible (as it happens for $MC_{duration} = 80$ s and $\mu C_{duration} = 30$ s).

The maximum number of collars follows a negative exponential tendency as we increase the area to be covered. Depending on the μC duration, the velocity on which the maximum number of collars tend to zero also vary. Particularly, one can see that for smaller and larger values for the μC duration there is a quicker tendency to zero than for μC durations that are in the middle. This suggests that values around the 4 s, 6 s or 10 s are preferable comparing to the remaining ones, which, in fact, agrees with project partners needs.

For improving values readability, a summary of the more relevant data is provided in Table 8.11. Among all calculations, we show the values for $MC_{duration} = \{60, 80, 100\}$s, $\mu C_{duration} = \{4, 6, 10\}$s and $area = \{1, 5, 21, 144\}$ha, that correspond to project partners requirements values plus the value above and below within the vector of values tested. From the available data, we are able to take the following conclusions:

- The maximum number of collars increases by increasing the MC duration, what is expected since we are increasing the total duration on which data can be transmitted;
- In a few scenarios, fixing the MC duration, but varying the μC duration, results in a maximum number of collars that does not increase continuously. We can observe that, for instance, for $MC_{duration} = 100\,\text{s}$ and $area = 5\,\text{ha}$, for which the maximum number of collars are 2190, 2144 and 2377, for $\mu C_{duration} = 4\,\text{s}, 6\,\text{s}$ and $10\,\text{s}$, respectively. This is the result of data encapsulation within B2B μCs, that provides optimal results depending on the total data transmitted;
- The maximum number of collars among all scenarios is 3039 for $MC_{duration} = 100\,\text{s}$, $\mu C_{duration} = 4\,\text{s}$ and $area = 1\,\text{ha}$, that corresponds to the utilization of an unique beacon;
- The minimum number of collars is 44 collars for the scenario $MC_{duration} = 60\,\text{s}$, $\mu C_{duration} = 10\,\text{s}$ and $area = 144\,\text{ha}$, corresponding to the utilization of 56 beacons;
- The two last scenarios are highly unlikely to occur in reality. The former because there are no herds of that size, or if there are, animals are not kept all together because of managing purposes. Even though, we show that our system supports the coexistence of an high number of devices. For instance, we may be interested in future to include both collars and vineyard sensors in the same communication infrastructure. The later scenario because, typically, grazing plots are kept smaller (smaller than 144 ha) for facilitating grazing rotation periods. However, in situations where the terrain is very irregular, an high number of beacons may be required. In that case, this type of scenario may gain interest. For that cases, user shall look for an optimized combination of system parameters;
- For the scenario with $MC_{duration} = 80\,\text{s}$ and $\mu C_{duration} = 6\,\text{s}$, that matches the maximum periodicities set by project partners, the maximum number of collars for 5 ha is 1682, thus within the requirements, while for 21 ha the maximum number of collars is 587, which is still an acceptable number of animals.

Table 8.11: Maximum number of collars, collars average consumption, autonomy and medium occupation for $MC_{duration} = \{60, 80, 100\}$s, $\mu C_{duration} = \{4, 6, 10\}$s and $area = \{1, 5, 21, 144\}$ha. Besides the value for the MC duration used to calculate the remaining values, also the calculated MC duration for the specific set of parameters is provided. For the sake of readness, the term duration was suppressed for the MCs and μCs.

Defined MC (s)	**μC (s)**	**Calculated MC (s)**	**Area (ha)**	**N_{Coll}**	**Avg.Cons. (mA)**	**Autonomy (h)**	**MO (%)**
60	4	60	1	1694	5.85	480.38	73.66
		60	5	1268	5.90	476.75	76.55
		60	21	418	6.21	452.76	76.09
		60	144	50	8.31	338.41	74.11
	6	60	1	1684	5.25	535.59	73.16
		60	5	1338	5.28	532.57	80.67
		60	21	408	5.49	512.35	73.80
		60	144	51	6.88	408.28	72.97
	10	60	1	1760	4.77	589.81	76.43
		60	5	1172	4.78	587.61	70.61
		60	21	408	4.91	572.65	73.46
		60	144	44	5.75	489.05	63.58
80	4	80	1	2363	5.85	480.72	77.03
		80	5	1691	5.89	477.08	76.54
		80	21	587	6.20	453.06	79.69
		80	144	77	8.30	338.58	77.34
	6	78	1	2357	5.15	545.91	76.79
		78	5	1682	5.18	542.85	76.03
		78	21	587	5.38	522.37	79.25
		78	144	79	6.74	416.76	75.73
	10	80	1	2347	4.76	590.32	76.44
		80	5	1759	4.78	588.12	79.45
		80	21	584	4.90	573.13	78.54
		80	144	76	5.74	489.40	71.59
100	4	100	1	3039	5.84	480.92	79.25
		100	5	2190	5.89	477.28	79.28
		100	21	735	6.20	453.24	79.73
		100	144	105	8.30	338.68	79.30
	6	96	1	3031	5.09	552.30	78.99
		96	5	2144	5.12	549.22	77.52
		96	21	704	5.32	528.57	76.03
		96	144	106	6.66	422.02	77.37
	10	100	1	2934	4.76	590.63	76.43
		100	5	2337	4.78	588.42	84.40
		100	21	720	4.90	573.42	77.36
		100	144	108	5.74	489.61	76.40

8.2.4.2 Energy consumption and autonomy

Figure 8.4 and Table 8.11 also provides data regarding the average consumption of collars for each one of the evaluated scenarios. The purpose was to enable the evaluation of how collars average current consumption evolves in different scenarios.

From Figure 8.4 we see that the average current consumption increases linearly as long as we increase the area to be covered. That is the consequence of increasing the number of beacons on the system, which forces collars to be awake and in the receiving mode more time, therefore, less time in the sleep state. Other conclusion is that the slope of the tendency line decreases as the μC duration increases. That is justified because with the increasing duration of a μC, the percentage of time that a collar spends in the receiving mode decreases. Hence, the impact on current consumption is diluted. In a nutshell, minimizing collars average consumption, implies minimizing the number of beacons and enlarging the μCs duration.

Focusing on the values provided in Table 8.11, particularly on the project partners maximum periodicities ($MC_{duration} = 80\,\text{s}$ and $\mu C_{duration} = 6\,\text{s}$), collars average consumption varies from 5.15 mA up for an area of 1 ha up to 6.74 mA for 144 ha. Applying Equation 7.7 and considering a battery capacity of 2670 mAh and a BE of 95%, autonomies from 545,91 hours down to 416,76 hours are expected for the stated scenarios. That corresponds to approximately 22 and 17 days, thus above the two weeks defined as minimum acceptable, but still away from the desired four months. However, there are two important remarks regarding these values. The first is that we are speaking about 22 days in continuous operations. In practice, the system can (and shall) be stopped during grazing periods, for instance while animals are within the sheepfold. Therefore, considering non grazing periods of 12 hours, the autonomy increases close to more than 40 grazing days. Secondly, as system optimizations are still pursued, there is margin to increase system performance, and consequently collars autonomy.

Nevertheless, this evaluation corresponds to a scenario where no stimuli are applied, which besides being the desirable behaviour, it does not correspond always to the reality. Therefore, it is important to evaluate the impact of stimuli on collars average consumption and autonomy, which is addressed in Section 8.2.4.4.

8.2.4.3 Medium occupation

MO was calculated using Equation 7.3 and the results summarized by the means of two approaches. Table 8.11 shows the MO for particular use cases among all combinations of MC and μC values. One can observe that values vary from 63,58% ($MC_{duration} = 60\,\text{s}$, $\mu C_{duration} = 10\,\text{s}$ and $Area = 144\,\text{ha}$) up to 84,40% ($MC_{duration} = 100\,\text{s}$, $\mu C_{duration} = 10\,\text{s}$ and $Area = 5\,\text{ha}$).

The higher the MO, the better the use of the medium. However, we remember that one PR μC is always contemplated in the MC structure and that the MO associated to its VTW is negligible, thus ignored for this calculation. That means that MO is bounded by this constrain that was an architectural decision.

There is not a clear tendency on the values calculated among all scenarios and within each one of the scenarios, *i.e.*, when fixing the MC and μC durations. However, for the later case, we are able to identify three main behaviours, namely:

- **MO saturates around a certain threshold:** as exemplified in Figure 8.5, in some cases, the MO is approximately the same independently of the area and number of collars considered.

This occurs when $\mu C_{duration} = 1\,\mathrm{s}$ and is justified by a smaller VTW duration, that is turn is a consequence of the value of the TAW that is statically defined for the worst case scenario (600 ms);

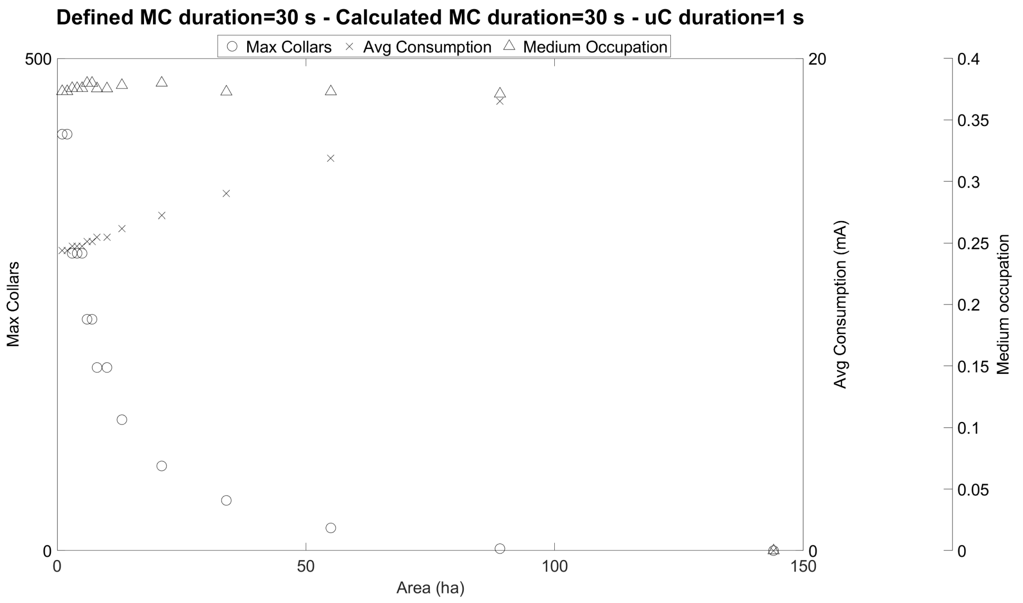


Figure 8.5: Maximum number of collars, average consumption and medium occupation for $MC_{duration} = 30\,\mathrm{s}$ and $\mu C_{duration} = 1\,\mathrm{s}$.

- **MO decreases exponentially:** MO follows the same tendency as the maximum number of collars (see Figure 8.6 for an example). This occurs typically for larger μC durations since the effects of the SW and TAW durations are minimal due to larger VTWs;

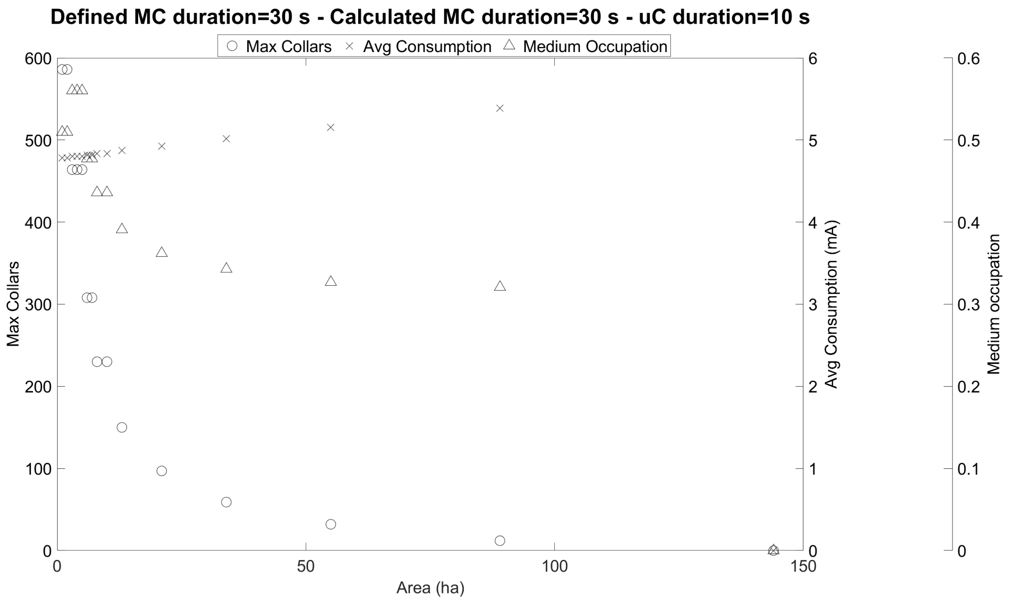


Figure 8.6: Maximum number of collars, average consumption and medium occupation for $MC_{duration} = 30\,\mathrm{s}$ and $\mu C_{duration} = 10\,\mathrm{s}$.

- **MO varies without any tendency:** in most of the scenarios there is not an evident tendency on the MO evolution with the number of collars and area considered. As exemplified in Figure 8.7, the values either increase or decrease depending on the relation between collars and area.

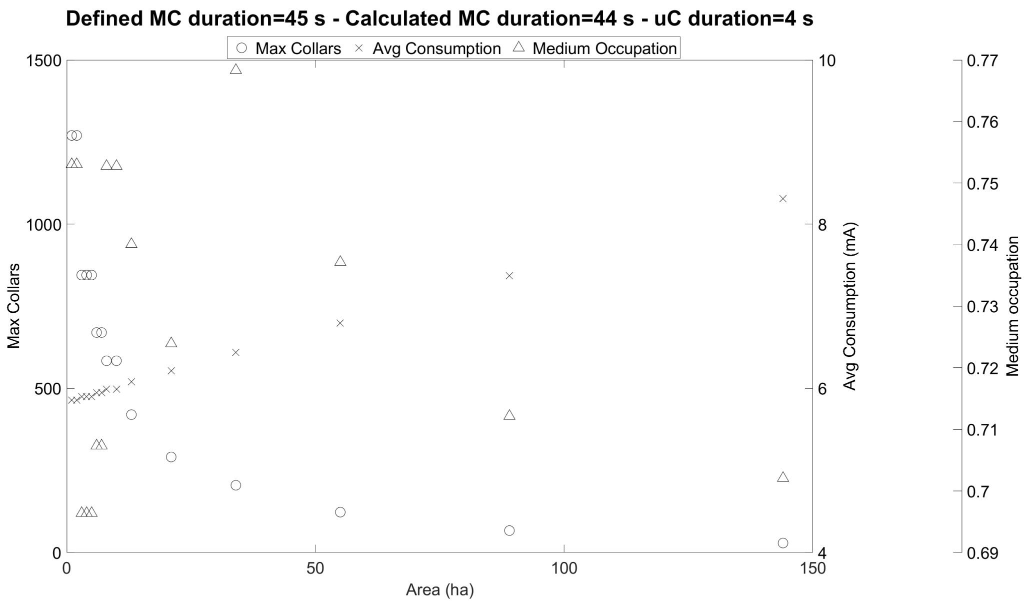


Figure 8.7: Maximum number of collars, average consumption and medium occupation for $MC_{duration} = 45\,\text{s}$ and $\mu C_{duration} = 4\,\text{s}$.

This evaluation confirmed a poor behaviour of the system for smaller μC durations. Furthermore, considering that one PR μC is always included and that a few communications occur on it, the MO for the scenarios expected to be used in practical implementations is in average 77%, which is acceptable.

8.2.4.4 Impact of the number of stimulus in the autonomy of collars

The data provided in Section 8.2.4.2 does not consider the application of any stimuli during collars' operation. However, in practice, that will not happen since one of the main purposes of the system is to condition sheep's posture. Therefore, it becomes relevant to evaluate how the number of stimuli impacts on collars average consumption and autonomy. Therefore, considering the worst-case scenario for an intended area assessment, we fixed the MC duration to 80 s, the μC duration to 6 s and an area to cover of 5 ha as suggested by project partners. Then, different number of buzzer and electrostatic discharges (see Table 8.12) were combined and used to calculate the average consumption of collars and autonomy, using the same approach described in Section 8.2.4.2. A wide range of values were considered to cover different possibilities, from low number of stimuli to a very high number of stimuli.

Table 8.12: Number of audio cues and electrostatic discharges per hour used to evaluate the impact of stimuli on collars average consumption and autonomy.

Number of stimuli per hour assessed
$\{0, 1, 2, 3, 4, 5, 6, 7, 8, 10, 13, 21, 34, 55, 89, 144, 233, 377, 610, 987\}$

Figures 8.8 and 8.9 illustrate the results obtained for collars average current consumption and collars autonomy, respectively. As expected, their behaviour converge. In Figure 8.8, when both values

increase, the average current consumption increases. However, it is observable that the impact of electrostatic discharges is higher comparing to the audio cues. For instance, considering the values in the plane *number of electrostatic discharges-average consumption* and the plane *number of audio cues-average consumption*, the values in the former plane are continuously higher comparing to the later plane.

Contrary, Figure 8.9 shows that the autonomy is higher for lower values of stimuli. It can reach 532 hours of autonomy without stimuli and it decreases up to 109 hours in the case of 987 stimuli per hour of both type of stimuli. However, these upper bound of stimuli are inflated, particularly regarding the number of electrostatic discharges. In Section 6.2.2 we identified a safety system that, according to the configuration provided by user, blocks the application of electrostatic discharges after a defined threshold. Thus, if the user defines a maximum of 3 electrostatic discharge per MC, it means a maximum of 135 electrostatic discharges per hour. In that case the maximum average consumption is 14.30 mA and the autonomy 196.48 hours (around a week of continuous operation) for 987 audio cues.

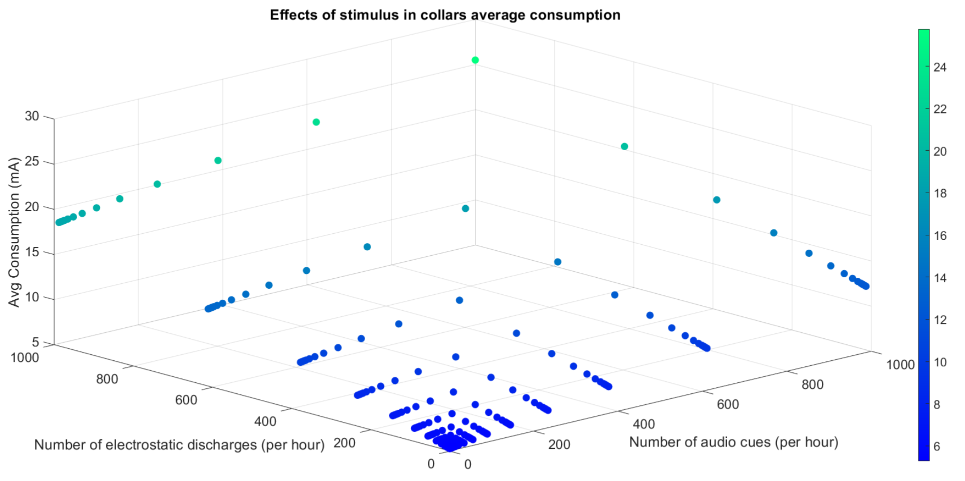


Figure 8.8: Effects of stimulus in collars average current consumption.

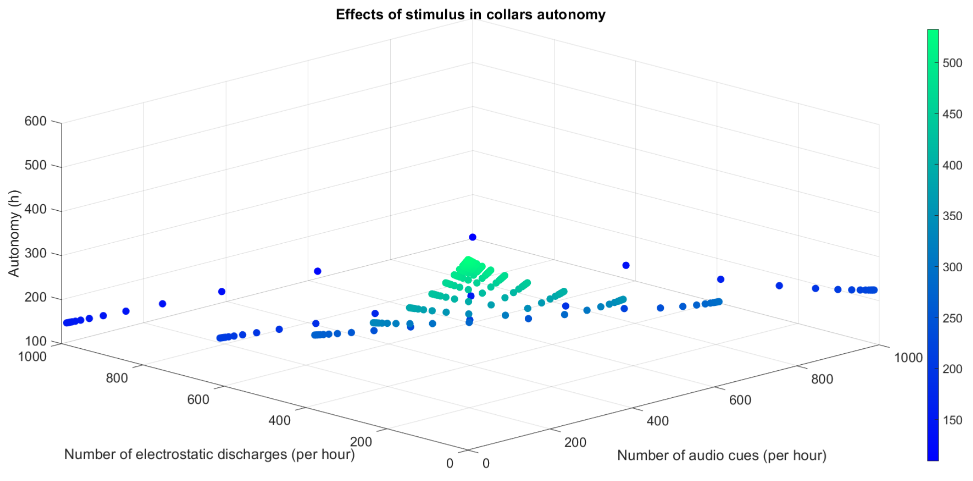


Figure 8.9: Effects of stimulus in collars autonomy.

8.3 Posture Control

The posture control mechanism is an important pillar of this thesis. Its design, detailed in Section 6, comprises two main modules, behaviour monitoring and conditioning, that combine synergies orchestrated by an algorithm that we name posture control algorithm. During this section we present and discuss the results obtained, starting with the animal behaviour detection, before evaluating the operation of the conditioning mechanism.

8.3.1 Animal Behaviour Detection

8.3.1.1 *Infracting* and *Not Infracting*

The data collection phase consisted of a 3-hours experiment, where a single sheep was released onto a plain field, being its activity recorded on video. The data collected was then processed using R. The data preparation phase consisted of simple pre-processing procedures, specially to remove redundant and duplicated data. The resulting dataset consisted of 20555 observations, that were subsequently split randomly in a ratio of 75%-25% into two subsets: a training set (15416 observations) and a test set (5139 observations). The former was used to train the algorithms. The latter was used to test the trained models.

The results were evaluated in terms of the following metrics: *i)* accuracy; *ii)* recall; *iii)* specificity; *iv)* precision; and *v)* ROC and AUC.

Table 8.13 summarizes all the metrics evaluated. As it can be observed, the results do not differ much among all the algorithms, although RDF, DT (using package C50) and XGBoost present the best results in terms of accuracy and AUC. This similarity in terms of results can also be verified in Figure 8.10, on which we can witness that the curves are mostly overlapped.

This exploratory work allowed to infer that it is possible to distinguish the *Infracting* state behaviour of sheep from the remaining ones. However, being a exploratory work, it suffered of several important limitations that needed to be tackled. Firstly, it used a poor dataset, from a single sheep in a single day, which does not allow to extend the conclusions about the model's adaptability to other sheep. Secondly, pruning the animal behaving monitoring to a binary classification problem is not reasonable for a further implementation of a posture control algorithm. This is so because the *Infracting* state revealed itself more complex than initially suspected. After a preliminary implementation of a DT model, we noticed that there are two main situations where identifying more behaviours is of utmost importance. The first comprises the situations when the sheep is standing, with its head up (as occurs in a *Infracting* state), but without eating. The second comprises behaviours when sheep is running with its head up. Both issues required, hence, detection of the *Resting* and *Moving* states.

Table 8.13: ML algorithms evaluation considering the binary classification problem *Infracting* and *Not Infracting*.

	Metric				
Algorithm	**Accuracy**	**Recall**	**Specificity**	**Precision**	**AUC**
RDF	0.9696	0.8267	0.9861	0.8728	0.9870
DT (C50)	0.9693	0.8475	0.9833	0.8539	0.9860
XGBoost	0.9685	0.8267	0.9848	0.8625	0.9880
kNN	0.9622	0.7702	0.9844	0.8503	0.9770
SVM	0.9642	0.7590	0.9879	0.8778	0.9720
DT (rpart)	0.9591	0.8211	0.9750	0.8728	0.9700
Naïve Bayes	0.9527	0.8795	0.9612	0.7230	0.979

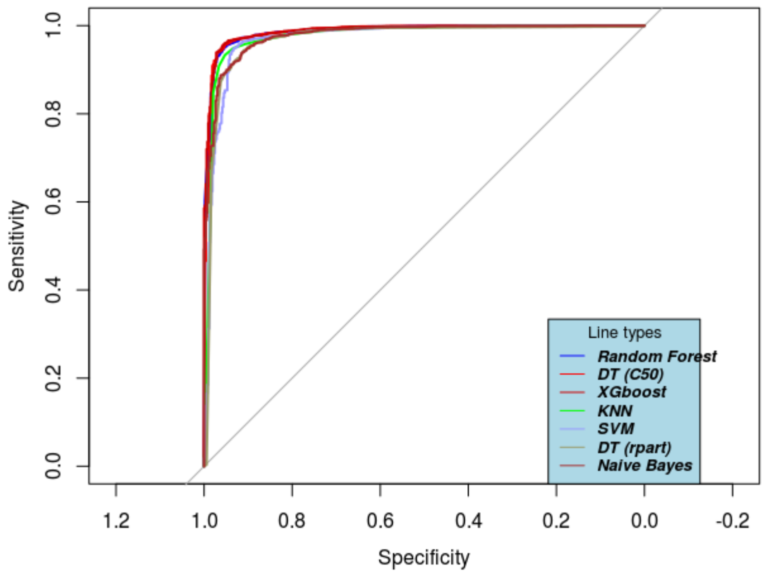


Figure 8.10: ROC Curves for the ML algorithms evaluated.

8.3.1.2 ***Infracting***, ***Resting*** **and** ***Running*** **states**

The method described in Section 6.1.2 was applied in a experiment that last over 3 hours. The data gathered was then processed using R, as it happened with the previous experiment. Some data cleansing techniques were applied, particularly the removal of observations with missing values, the removal of outliers and the removal of observations that were labelled as unclassified, resulting in a dataset with 22626 observations. During the data split phase, the dataset was randomly sorted and divided into two subsets, at a ratio of 25-75%. The larger subset was used for training, and the smaller one for testing.

The *Infracting* and *Not Infracting* states are related with the detection of posture infracting situations. Thus, with exception of the observations classified as *Infracting*, all the remaining observations were reclassified as *Not Infracting*. Considering the sensors incorporated in the collar and the results obtained in the previous works, only two combination of features were used: *i) Pitch angle* (from 3-axis

Static Acceleration) by itself; *ii) Pitch angle* combined with *distance to ground.*

The DT algorithm was used during the modelling phase, being obtained two decision trees, whose accuracies (over 95%) are quite similar (Table 8.14). As no relevant gain is observed when using the combination of *pitch angle* and *distance to ground* measurements, the decision was to use *pitch angle* by itself for detecting animal's posture, and rely on the ultrasound module to compensate effects not predicted in this DT model (*e.g.* terrain slope and animal stature differences).

To handle the resting and running behaviours, two additional sets were derived: *Resting* and *Not Resting* and *Running* and *Not Running.* For both the situations, the DT algorithm was used during the modelling phase, resulting into four decision trees, two for each case. The accuracies obtained for both tested situations concerning the *Resting* behaviour were over 80% (Table 8.14). Although the DT obtained when using the modulus of each dynamic acceleration' component as feature presented a slightly better accuracy, the improvement is not substantial. Better accuracies were not achieved mainly due to difficulties on separating clearly resting and rumination behaviours, even during the classification procedure.

Table 8.14: DTs accuracies for the three studies: *Infracting* vs *Not Infracting*, *Resting* vs *Not Resting* and *Running* vs *Not Running.*

Features	**Posture ACC**	**Resting ACC**	**Running ACC**
Pitch	0.9592	-	-
Pitch + Distance to Ground	0.9595	-	-
Modulus of 3-axis dynamic acceleration component	-	0.8131	0.9662
Modulus of dynamic acceleration vector	-	0.8061	0.9662

Regarding the two *Running* DTs produced, equal results were obtained in terms of accuracy (Table 8.14) - over 96%. This time, the 3-axis dynamic acceleration results only considered one axis to differentiate activity. When testing with each axis separately, the accuracy was the same for the three tests. Thus, the modulus of dynamic acceleration vector was deemed the optimal choice, since it relies on information from the three axis, which make it more robust to errors.

These two later trees allowed ruling out situations where the sheep is *Running* or *Resting*, and therefore not eating. This means that even if its posture is defined as *Infracting* (applying the *Infracting* vs *Not Infracting* DT), if the behaviour is then classified as *Running* or *Resting*, these "false" infractions are discarded. Thus, combining the results obtained for each situation, a merged decision tree, presented in Figure 8.11, was produced. This decision tree differentiates four behaviour states, Resting, Active, Infracting and Running, resulting in two levels of behaviours. Firstly detecting if the posture behaviour corresponds to an infraction or not, and then check if the level of activity corresponds to an infraction situation. This DT, easily transposed to a set of "if's" and "else's", was the first version of the monitoring mechanism implemented in collars.

Despite solving the issues related with the non detection of the *Resting* and *Running* states, there were some issues that still remain. On the one hand, it was also based in a poor dataset, with information about a single sheep. On the other hand, it did not allowed the identification of *Moving* behaviours, particularly when sheep change from grazing areas. Such behaviour also presents (as the

Running state) characteristics that can approach the *Infracting* state, thus a potential point of failure. Hence, the work presented in the following section aimed at evaluating the possibility of integrating a unified mechanism capable of detecting all the behaviours already addressed, namely: Infracting, Resting, Running, Moving and Eating states.

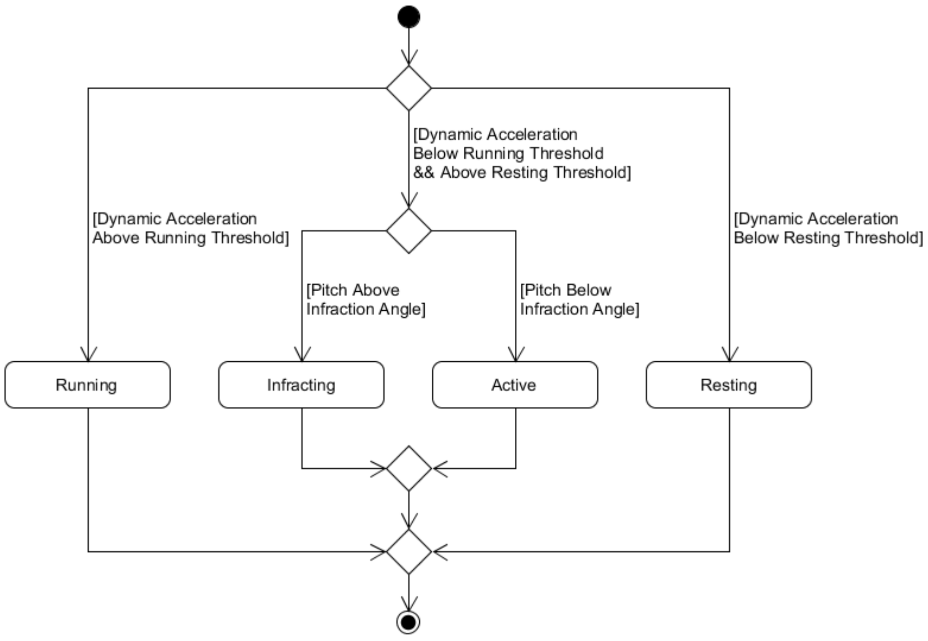

[Dynamic Acceleration Below Running Threshold && Above Resting Threshold]
[Dynamic Acceleration Above Running Threshold]
[Dynamic Acceleration Below Resting Threshold]
[Pitch Above Infraction Angle]
[Pitch Below Infraction Angle]
Running
Infracting
Active
Resting

Figure 8.11: Merged DT.

8.3.1.3 *Infracting*, *Resting*, *Grazing*, *Moving* and *Running* states

The results associated to the experiment described in Section 6.1.3 are now described, from the feature selection procedures up to the evaluation of the results. The summary of the observations can be found in Table 8.15. Most of the dataset observations are of the type *Eating* (70%), which was expected since animals tend to be grazing most of the time. The less representative states are *Infracting* (3%), *Running* (2%) and *Standing* (2%), which was also expected since sheep were free to pasture on a area with a lot of edible weed and without being exposed to external dangers (and thus not having the need to run away). It is relevant to mention that the dataset collection was not performed during a full day, only during morning and afternoon. Summing up, we are in the presence of an unbalanced dataset with a total of 12968 valid experiments, being 1675 observations discarded (approximately 12 %).

Table 8.15: E - *Eating State*; M - *Moving State*, I - *Infracting State*, S - *Standing State*, R - *Running State*, X - Invalid observations.

State	E	M	I	S	R	X	Total
Number of observations	10305	1686	488	260	229	1675	14643

To achieve the desired goal, several tests were made following an iterative approach. As it is impracticable to present and discuss all the iterations, we focused on the more important ones, towards an enhanced monitoring mechanism. In this section we present and discuss the main results of the feature selection stage and the most relevant models trained using ML together with the evaluated metrics.

All **data processing and cleansing** was carried out using RStudio [255], an open source and powerful tool for data science applications. Regarding the data cleansing, the first step was verifying data integrity. For that purpose, all features were summarized and their type, minimum, average and maximum values were checked.

Recalling the dataset's characterization, a total of 27 features were available, 7 directly generated in the collar, 2 added in the gateway, 15 obtained after feature transformation of dynamic acceleration measurements, and 3 additional features added during data processing regarding state's history.

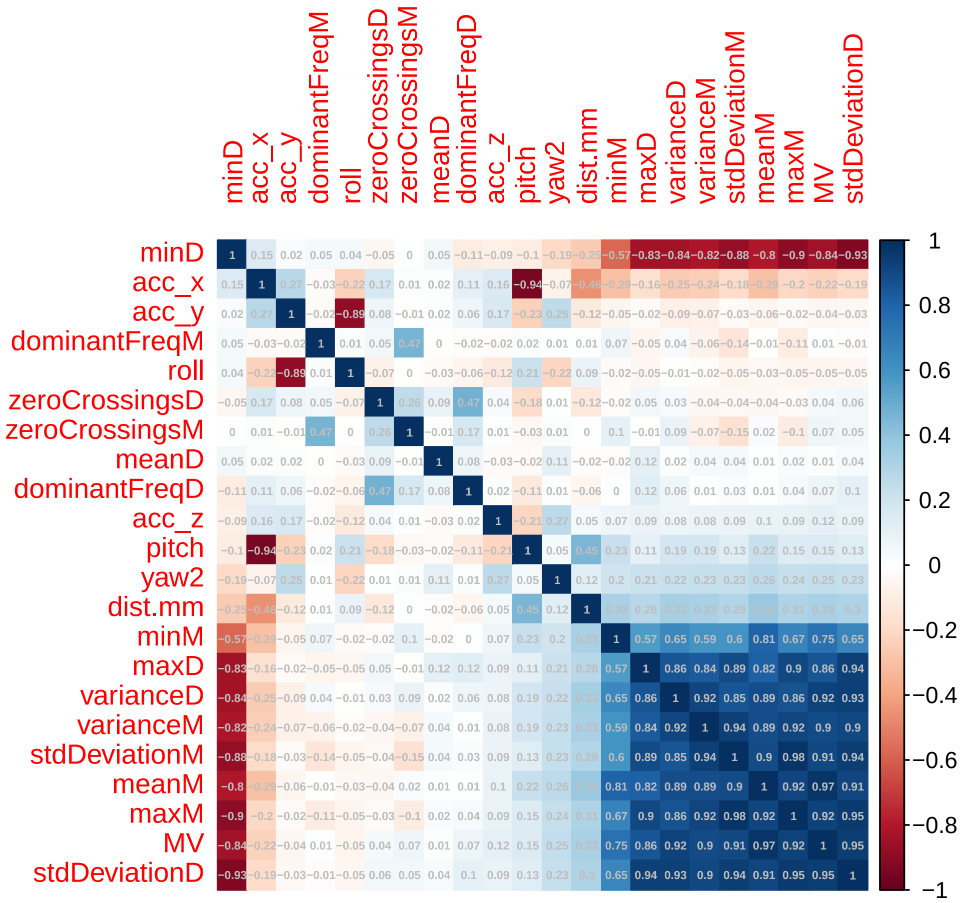


Figure 8.12: Correlation between features obtained after feature transformation.

Concerning the feature selection stage, two tiers of results were achieved, in line with the methodology presented in Section 6.1.3. The former, was the correlation matrix that allowed the identification of redundant features. From the correlation matrix, represented in Figure 8.12, the features whose correlation module scored above 0.9 were selected for isolation. From the most correlated features, an individual analysis was performed, being removed the features that presented a higher number of correlated features (*e.g. stdDeviationD* is highly correlated with *varianceD, maxD, minD, stdDeviationM, meanM* and *maxM*, hence it was removed). This process culminated with the removal of the following features: static acceleration on x-axis (*acc_x*), maximum value of the dynamic acceleration magnitude (*maxM*), standard deviation of the first derivative (*stdDeviationD*), variance of the dynamic acceleration magnitude (*varianceM*) and movement variation (*MV*).

The latter comprised the evaluation of the results' consistency among the selected feature selection techniques described in Section 6.1.3. Hence, the overall accuracy of the models obtained using the different techniques was assessed, with and without considering the use of stateful features.

Table 8.16 summarizes the results obtained without considering the stateful features. Besides the features' ranking regarding each feature selection technique, the global accuracy is also available. The analysis of these results grants the following considerations:

Table 8.16: Feature Selection evaluation without stateful features.

Order/Technique	**CFS**	**Chi Squared**	**oneR**	**Random Forest**	**RReliefF**
1	dist.mm	meanM	dist.mm	pitch	pitch
2	pitch	pitch	pitch	dist.mm	zeroCrossingM
3	-	dist.mm	meaM	meanM	dominantFreqD
4	-	varianceD	varianceD	varianceD	dominantFreqM
5	-	stdDeviationM	stdDeviationM	stdDeviationM	meanM
6	-	minD	minM	minD	zeroCrossingsD
7	-	minM	minD	maxD	acc_z
8	-	maxD	maxD	minM	stdDeviationM
9	-	acc_z	acc_y	acc_z	acc_y
10	-	yaw	acc_z	yaw	varianceD
11	-	acc_y	yaw	acc_y	maxD
12	-	roll	meanD	roll	roll
13	-	dominantFreqD	roll	meanD	yaw
14	-	meanD	zeroCrossingsM	zeroCrossingsD	dist.mm
15	-	zeroCrossingsD	dominantFreqM	zeroCrossingsM	meanD
16	-	dominantFreqM	zeroCrossingsD	dominantFreqM	minD
accuracy	**88.51%**	**90.4%**	**90.46%**	**90.46%**	**90.46%**

- The results obtained using the *consistency-based filter* were automatically discarded since the order of the features obtained was exactly the same as the input order of the features given during the training phase, thus, not entailing valuable information;
- The *dist.mm* and *pitch* features are the most important features for modelling the system. Besides being always in the top four features among all feature selection techniques, it is visible that the Correlation-based Feature Selection (CFS) algorithm only chooses these two features, neglecting all the remaining ones and maintaining an accuracy of 88.51%, only 1.94% below the best case;
- By splitting the features list into four groups (named from now on, tiers of features), it is possible to verify that the *Chi Squared* filter, the *oneR* algorithm and the *Random Forest* filter, present nearly all the same features inside the four tiers, although feature permutations may be found within them. These results show potential consistency between these methods;
- Regarding the *RReliefF* algorithm, a different distribution of features throughout the four tiers can be observed when comparing to the remaining feature selection algorithms, including in the top five features. Here, three of them (*zeroCrosingsM, dominanteFreqD* and *zeroCrossingsD*) were in the end of the list obtained for the other techniques. This situation did not affect the global accuracy, probably because despite being selected by the feature selection techniques they were being disregarded by the model. However, this algorithm presented a higher variability when compared to the remaining ones, *i.e*, when considering different samples from the training dataset, relevant changes were noted in the feature ranking. Thus, considering the referred issue,

together with the strong computational demands required to implement features such as the *dominantFreqD*, led to disregard the use of the *RReliefF* algorithm;

- From the analysis of the ranking list of features obtained using the remaining algorithms (discarding the results from the *RReliefF* algorithm), a high level of agreement was detected in all top ten features. Hence, the accuracy obtained when using the top ten features given by the *oneR* algorithm was assessed to evaluate the impact of reducing the number of features. The result was 90.43%, only 0.03% below the best value obtained using all features. Consequently, hereinafter, only the top ten features selected by the *oneR* algorithm were used.

To evaluate the effects of using the stateful features, the top ten features that resulted from the previous test were taken along with the referred stateful features. The procedure considered was the equivalent to the one applied to select the top ten features, being the results summarized in Table 8.17. Its analysis allows the following conclusions:

- The *prevState* feature becomes the most relevant. In fact, comparing the results obtained using the *CFS* filter (96.27%) with the remaining ones (98.50% and 98.37%), it seems that the model resorts almost exclusively on that feature. Such situation may indicate overfitting, which must be confirmed during the modelling phase;
- The *transition* feature seems to be irrelevant to the model, and thus it will not be used in further tests;
- The *nequalStates* feature is highly ranked when using the *Random Forest* filter but it is poorly ranked when using the *CFS* filter and *Chi Squared* filter. Despite this, it was maintained to build the models;
- Albeit the *prevState* feature predominates, the main features of the previous analysis (*pitch* and *dist.mm*) are still well ranked.

Table 8.17: Feature Selection evaluation with stateful features.

Order/Technique	**CFS**	**Chi Squared**	**oneR**	**Random Forest**
1	prevState	prevState	prevState	prevState
2	-	meanM	dist.mm	nequalStates
3	-	pitch	pitch	pitch
4	-	varianceD	meanM	meanM
5	-	dist.mm	varianceD	stdDeviationM
6	-	stdDeviationM	stdDeviationM	varianceD
7	-	minD	minM	maxD
8	-	minM	minD	dist.mm
9	-	maxD	maxD	minD
10	-	nEqualStates	acc_z	minM
11	-	acc_z	transition	acc_z
12	-	transition	acc_y	acc_y
13	-	meanD	nequalStates	transition
accuracy	**96.27%**	**98.50%**	**98.50%**	**98.37%**

During the **modelling stage**, several models were built, all resorting in DT and considering a 10-fold cross validation approach. The *rpart* package from R was used for all the modelling procedures. The training function has two main tuning parameters, the *complexity parameter (cp)* and the *max depth* parameter. The former, represents how much the relative error is incremented when splitting a node. The later, represents the maximum number of split levels. Thus, during all modelling procedures, both parameters were adjusted, seeking for the best evaluating metrics. Taking into consideration the results of the feature selection stage, four main study cases were considered, particularly:

1. **Case 1:** Using the top ten features obtained after feature selection, without using the state's history;
2. **Case 2:** Using the top ten features plus the *prevState*;
3. **Case 3:** Using the top ten features plus the *prevState* and *nEqualStates*;
4. **Case 4:** Using the top ten features plus the *nEqualStates.*

Two types of tools were used during the **evaluation phase**. Firstly, the construction of confusion matrices to easily visualize and interpret the prediction summary, namely the number of TN, FN, FP and TP. Secondly, the evaluation of some performance metrics commonly used in classification problems in ML, for instance, *accuracy*, *micro* and *macro-averages* of the *F1 score* and the *K-category correlation coefficient*, a multiclass extension of the *Matthews Correlation Coefficient* (see Section 2.3.5). The *accuracy* is used since it is a general and very common metric used in ML problems, found in almost every state-of-art works. However, as the dataset is unbalanced, there are two metrics that assume a higher importance, namely the *macro-average F1 Score* and the *K coefficient.* Both provide a more credible performance of a model in an imbalanced dataset. Their theoretical context and respective equations were presented in Section 2.3.5.

The most promising scenarios, identified through the feature selection phase, were modelled using 10-fold cross validation. This technique has natural consequences on the evaluation phase since, when implementing crossing validation, we need to model and test the same number of times as the number of folds. This means that when testing a scenario that implements a 10-fold crossing validation, it results in 10 confusion matrices and 10 values for the same metric. To summarize the results associated to each scenario, the confusion matrices were summed and the average of the values for each metric were considered.

Case 1: Using the top ten features, without stateful features

The confusion matrix obtained using the top ten features is illustrated in Table 8.18. The most critical situations for the posture control mechanism and whose number is intended to be minimized, are signalled in bold and are underlined. These observations correspond to misclassified observations related to the *Infracting* state, corresponding both to FP and FN. Since both cases have a negative impact, a sum of all these situations is considered when comparing such results. In this case, a total of **366** situations were registered.

Table 8.22 summarizes all the metrics evaluated. The global accuracy obtained for this case was 90.53%, with an average of a total of 25 splits. Analysing the results, we can observe a *micro-average F1 score* value very close to the global accuracy. This means that the model performs well in overall, a behaviour expected due to the high prediction correctness of the *Eating* state, the predominant

Table 8.18: Confusion matrix obtained for case 1.

		Reference					
		E	**I**	**M**	**R**	**S**	**Total**
Predicted	**E**	10023	**43**	355	4	40	**10465**
	I	**46**	313	**123**	**4**	**30**	**516**
	M	158	**109**	1102	84	54	**1507**
	R	3	**0**	57	135	0	**195**
	S	16	**11**	19	0	120	**166**
	Total	**10246**	**476**	**1656**	**227**	**244**	**12849**

one. Contrary, the *macro-average F1 score* presented a lower value, revealing the limitations of the model on correctly classify individual classes. This conclusion is reinforced by the R_K coefficient value obtained, also low.

As the *Eating* state predominates, the model gives a higher importance to the *pitch* angle and to the distance to the ground (*dist.mm)* features. However, for the remaining states, the values for these two features are equivalent or at least very close to each other. Thus, to differentiate those states, the dynamic acceleration features should be vital. However, the use of dynamic acceleration measurements also presented some limitations when applied to this use case. This may be explained, on the one hand, to the very smooth boundaries of those measurements between states, and, on the other hand, to the unbalanced dataset. Consequently, strategies of downsampling and upsampling were tested, but without any relevant improvement.

Additionally, as a high number of misclassifications were detected in observations corresponding to transitions between states and as, theoretically, there are transitions that are more likely to occur than others, the *prevState* feature was added. The objective was to introduce a temporal feature that could help the model to find relationships between states' transitions.

Case 2: Using the top ten features plus the *prevState* feature

As it can be inferred through the analysis of the confusion matrix represented in Table 8.3.1.3, adding the *prevState* feature allowed to decrease the number of FP and FN of the *Infracting* state. The sum of those cases is 88, with a total average of only 5 splits in the DT. Consequently, the evaluation metrics present notorious improvements as it can seen in Table 8.22.

However, after evaluating an example of a DT obtained during the learning stage, unfeasible and unrealistic conditions for practical implementations were detected. Figure 8.13 depicts an example of a pruned DT. The leafs with label (*e.g. prevState = E*) represent decision features, while rectangles represent the predicted output state for a certain branch. Within each rectangle are represented the number of observations that are predicted as the indicated state (for instance "E" in the leftmost rectangle), and the real classification (each column is a state). As it can be seen, conditions that would result in infinite loops are encountered. For instance, we can see that if *prevState* is "E", then the predicted state is also "E", which means that the model predicts the next state as the previous one. Thus, transposing such conditions to a real application, would result in an algorithm getting stuck in the *Eating* state whenever the *prevState* is *Eating.* This undesirable behaviour is explained with the common sheep behaviour, that typically stay in the same state for a considerable amount of time, particularly in the *Eating* state. Hence, considering these features, the problem of minimizing the error is solved by the model by predicting the next state as being the same as the previous one. Therefore, besides promising results, this model does not bring practical advances to the monitoring mechanism.

		Reference					
		E	**I**	**M**	**R**	**S**	**Total**
Prediction	**E**	10119	**23**	111	7	27	**10287**
	I	**2**	423	**29**	**2**	**2**	**458**
	M	102	**27**	1479	23	18	**1649**
	R	2	**1**	23	194	0	**220**
	S	211	**2**	14	1	197	**425**
	Total	**10436**	**476**	**1656**	**227**	**244**	**13039**

Table 8.19: Confusion matrix obtained for the case 2.

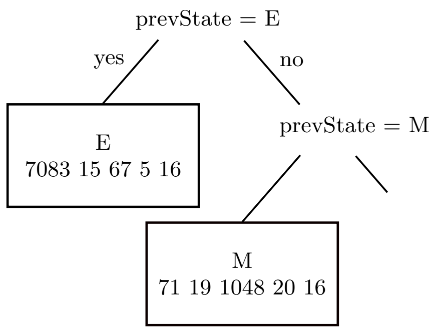


Figure 8.13: Example of a loop extracted from a DT model.

Case 3: Using the top ten features plus the *prevState* and *nEqualStates* features

The previous test exposed a critical limitation of using the *prevState* feature for modelling the sheep behaviour: the creation of dead end conditions in the DT. In other words, transposing such conditions to a real use case scenario would result on a worthless mechanism, since it would get stuck in the same state. Aiming at overcoming this issue, the *nEqualStates* feature was added to the model. The goal was to provide additional information to the model about the transition between states, particularly, information that could avoid dead end conditions.

The obtained confusion matrix is depicted in Table 8.3.1.3. The number of FP decreased again when comparing to the first two tests and the evaluated metrics also conferred better results (Table 8.22). However, and again after analysing an example of a decision tree obtained, a critical limitation was found. As it can be assessed through Figure 8.14, albeit direct infinite loops were avoided, hidden infinite loops were detected. For instance, the conditions for being in the *Eating* state are two: the *prevState* is *Eating* and the *nEqualStates* is greater or equal to 2. This means that if an animal stays in the *Eating* state more than one sample, the monitoring mechanism gets blocked infinitely in such state. This issue occurs not only for the *Eating* state, but also for the remaining states. Downsampling and upsampling techniques were also tested in this use case, being applied both to the classified behaviour states and to the number of transitions (trying to balance the number of transitions). Notwithstanding, no relevant changes or enhancements were detected. Thus, although this approach demonstrates promising results, its implementation is unviable.

		Reference					
		E	**I**	**M**	**R**	**S**	**Total**
Predicted	**E**	10219	**2**	42	5	7	**10275**
	I	**2**	436	**17**	**3**	**9**	**467**
	M	24	**35**	1589	13	34	**1695**
	R	0	**0**	5	206	0	**211**
	S	1	**3**	3	0	194	**201**
	Total	**10246**	**476**	**1656**	**227**	**244**	**12849**

Table 8.20: Confusion matrix obtained for the case 3.

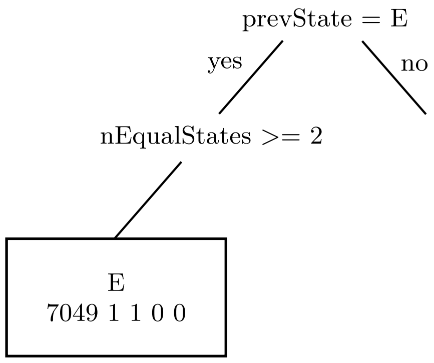


Figure 8.14: Example of a pseudo-loop extracted from a DT model.

Case 4: Using the top ten features plus the *nEqualStates* feature

Albeit the use of the *prevState* feature has revealed itself to be a misstep, at least with the present dataset, the existing improvements from case 2 to case 3 led to the consideration of a fourth use case, joining the top ten features with the *nEqualStates* feature. The obtained confusion matrix is detailed in Table 8.21. Comparing the results obtained with the ones provided for the case 1, a decrease of 20 misclassified cases regarding the *Infracting state* is registered (a decreasing of almost 5%).

Table 8.21: Confusion matrix obtained for the case 4.

		Reference					
		E	**I**	**M**	**R**	**S**	**Total**
Predicted	**E**	10043	**13**	287	16	29	**10388**
	I	**36**	330	**130**	**4**	**32**	**532**
	M	142	**118**	1176	88	58	**1582**
	R	2	**0**	42	119	0	**163**
	S	23	**15**	21	0	125	**184**
	Total	**10246**	**476**	**1656**	**227**	**244**	**12849**

The global accuracy obtained (Table 8.22) was 91.78% with a total average of 25 splits, more than 1% above the global accuracy obtained for case 1. Also, some relevant improvements are observed in the remaining metrics evaluated, including in the *macro-average F1 Score* and R_K coefficient.

Table 8.22: Results of the metrics evaluated for the four cases tested during the modelling phase. Case 4 is next to Case 1 since it is an enhancement of it.

	Case 1	**Case 4**	Case 2	Case 3
Accuracy	0.9100	**0.9178**	0.9660	0.9840
microF	0.9100	**0.9178**	0.9660	0.9840
macroF	0.7031	**0.7086**	0.8947	0.9359
R_K	0.7314	**0.7571**	0.9010	0.9536

Wrapping up, all tests herein presented mainly aimed to provide a structured testing methodology that could contribute to the development of an enhanced monitoring mechanism, more accurate and less prone to errors than the previous implemented ones in the scope of the SheepIT project. For that, the enlargement of behavioural states was vital, without overlooking to the feasibility of the mechanism in a constrained platform, as the collar device.

After pre-processing the data collected from a customer of iFarmTec, the existence of an unbalanced dataset was noticed. This is the result of the natural behaviour of sheep that naturally feed/eat for long periods of time, being the remaining states mainly transient. Albeit upsampling and downsampling techniques have been evaluated without success, the relevant point is that real implementations need to deal with this issue because it is an inherent and continuous condition of the application scenario.

With the goal of minimizing such concern, additional features regarding the state's history were added. The objective was to give, as input, additional information about the refereed natural behaviour of sheep. Although the metrics obtained for these cases were very promising, the results shown to be impracticable in real implementations, particularly when resorting to the *prevState* feature. The appearance of dead end conditions revealed the unfeasibility of using such features. In fact, the number of transitions between states is not relevant comparing to the total number of samples, inducing the model to predict the next state as being equal to the previous one since it results in lower error. This precludes the model to seek for other kind of relationships.

Nonetheless, these results may give good insights for future works. For instance, solutions for forcing the models to break the loops shall be investigated as well as other features that could give better insights to the model about transitions between states. Furthermore, even through in practical use cases, a imbalanced dataset will also be found, bigger and richer datasets would be important to provide a deeper study on this issue.

Taking advantage of the improvements observed from Case 2 to Case 3, a final test using the top ten features plus the feature containing the number of samples on which no changes in the state are registered (*nEqualStates)* was considered. This case not only allowed an improvement of the global accuracy (reaching values similar to the ones obtained in state-of-the-art works) but also a reduction of the total of FP and FN associated to the *Infracting* state. The analysis of the DT obtained did not show any restriction for its implementation in real scenarios, presenting a total of 25 splits.

8.3.2 Posture control mechanism evaluation

Designing a system to be used on animals has several inherent hurdles that hamper the implementation of solid and coherent evaluation methods. All the tests that involved animals were accompanied by animal experts or were the result of data gathered in real use-case scenarios of the commercial product made available. Though, we had access to some of the most relevant but generic results. Considering this, two kinds of feedback are given. Firstly, the evaluation of the training process in a controlled environment, whose goal was to assess if, in fact, sheep are able to associate the audio cues with further penalizations and with undesired behaviours taken (*Infracting* state). Secondly, the evaluation of the posture control mechanism in a real vineyard scenario.

8.3.2.1 Training process

The training process was assessed in a controlled environment, established and performed by livestock experts with the presence of a veterinarian. For this test in particular, livestock experts defined a sequence of stimuli composed of a single combination of a cue followed by a single electrostatic discharge[1]. Also, it is of utmost importance to clarify that this experiment was performed before the definition of the final behaviour monitoring mechanism described in Chapter 6. Thus, the evaluation of the training process herein described focused in evidence of the cognitive capability of sheep to associate the cues with further penalizations and with forbidden behaviours. The relation between the monitoring behaviour mechanism and the effectiveness of the conditioning mechanism was not addressed within the scope of this experiment.

The experience comprised the use of nine sheep with similar size and height, divided into three equal-sized groups. An area of $10\,\mathrm{m}^2$ was enclosed by a fence and three volumes of fay (rich in nutrients) were suspended at a high of 50 cm from the ground, to simulate the vine's high. To ensure the willingness of sheep to feed, they were kept in fast since the night before the test. Additionally, each group experiment lasted for 15 min and manual observations were registered.

During the learning phase, collars were keep at a frequency operation of 5 Hz[2], continuously measuring and immediately triggering the conditioning mechanism, when an infraction was detected.

The first group was used for control. They were not subject to the training phase nor to posture control. As expected, this group of sheep feed from the fay immediately after being released to the area of the experiment.

The two remaining groups were used to evaluate the training process within the scope of the posture control mechanism. The first conclusion taken from the observations was the existence of a leader, *i.e.*, a sheep whose example was followed by others. These leaders are prominent sheep that firstly take decisions, which means that they immediately tried to reach the fay, triggering the conditioning mechanism. As these sheep never had contact before with these stimuli, both received the complete sequence of stimuli, *i.e.*, a warning cue and a penalization, reverting immediately the behaviour after the penalization.

For the first group, the complete sequence was given four times. From this point on, the leader sheep started to instantly revert the infracting behaviour after receiving the warning cue. The remaining sheep behaved similarly, but they did not attempt to feed from the fay as many times as the leader, probably because they were able to perceive the leader's behaviour. After reverting the behaviour, sheep started to look for food on the ground, behaving as it is intended to occur in a vineyard.

A similar behaviour was observed for the second group, except that the leader sheep took six conditioning cycles to start associating the penalization to the warning cue. Remarkably, after this process, this sheep did not try to feed from the fay so regularly as the leader of the first group.

Thought the experiment considered a small number of animals, it clearly showed that it is viable to induce sheep to associate a warning cue to a subsequent penalization and a forbidden behaviour, at least when the actuators are triggered at the right instants. Nevertheless, as the experiment did not evaluate situations where actuators are unduly triggered, it is not possibly to state that applying stimuli at unappropriated instants may compromise the training process, although it is highly likely to be so.

[1]Due to confidentiality issues, the configuration of both stimuli cannot be disclosed.

[2]At the date of this experience, the collar's state machine did not have implemented yet all the features described in Chapter 5. Albeit the μC already designed and implemented, the radio was disabled to reduce the interference in the reading and actuating procedures, which run subsequently every 200 ms.

8.3.2.2 The posture control mechanism in a real use-case

So far, in the scope of this chapter, we presented the tools to address *Research Question 3* that prompts *how can we grant low-power devices with real-time animal posture control capabilities.* These tools are the behaviour monitoring mechanism and the conditioning mechanism, that together compose the posture control mechanism. They were individually addressed in the previous sections, being now their use assessed in a real use-case. This assessment comprises the last step towards the sustainment of this thesis.

The experiment tracked an approach similar to the one presented for the behaviour monitoring mechanism. In other words, we took advantage of data gathered in the iFarmTec's client facilities to perform this evaluation, being aware of the inherent constraints. In fact, all the network parametrizations as well as the posture control configuration were defined by iFarmTec in agreement with the client (see Table 8.23 for detailed information). Thus, this evaluation focuses in the evaluation and discussion of the results more than in justifying the procedures.

Table 8.23: System's configuration in a real use-case experiment. In the stimuli sequence, *b* stands for an audio cue and *s* stands for an electrostatic stimulus.

n_{uC}	$MC_{structure}$	$uC_{duration}$	n_{nC}	$nC_{duration}$	Stimuli Sequence
2	C2B-B2B	3200 ms	4	500 ms	*bbsbss*

The data provided concerned the utilization of the system during three days and eight animals in the same vineyards described in Section 6.1.3. Sheep grazed on the entire parcel approximately from 09:00 am to 17:00 pm, being the posture control algorithm enabled all grazing time, excepting for the first day, on which animals were retracted to the sheepfold during lunch time. An important remark is the fact that, at this moment, all sheep have already been faced with the posture control mechanism, although these tests occurred after a hiatus of approximately 3 weeks, on which sheep did not carry the system.

The evaluation focuses into two different levels. In a first approach, a daily evaluation is considered, with particular account on the first day, on which the effects of the training process are expected to be observed. Then, the evolution of sheep behaviour is weighed, *i.e*, the impact of the posture control mechanism on sheep's behaviour along the three days is taken.

The aforementioned evaluation only considered the data gathered by the IoT system, complemented, in some cases, with some on-site observations. Regarding the data gathered, the analyses targets three main indicators, namely:

- **Number of infractions detected:** when a sheep attempts to feed with a forbidden behaviour, a infraction is considered. This number is normalized to the MC, which means that in one MC, only one increment is considered. This value allows to easily check the evolution of the number of infractions, *i.e.*, the tendency of sheep to adopt undesired behaviours. Also, this value compared with further indicators allow to understand the cognitive capacity of sheep to associate stimuli and how is its reaction to stimuli;
- **Number of penalizations:** number of times that the posture control mechanism enters in the *PEN* state. As it happens with the previous indicator, also this one is normalized to the

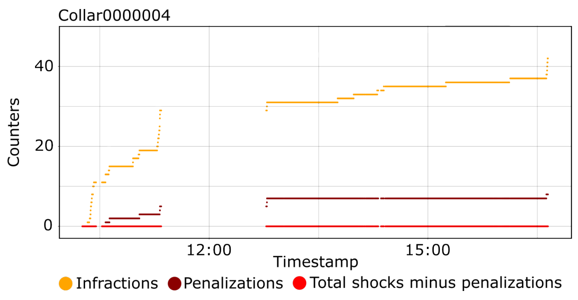
MC. Thus, in one MC, only one increment is allowed. This indicator, together with the number of infractions, allows to derive if sheep respond to audio cues;

- **Difference between the number of electrostatic discharges and the number of penalizations** described in the previous point. This indicator allows to perceive the effectiveness of the first electrostatic discharge.

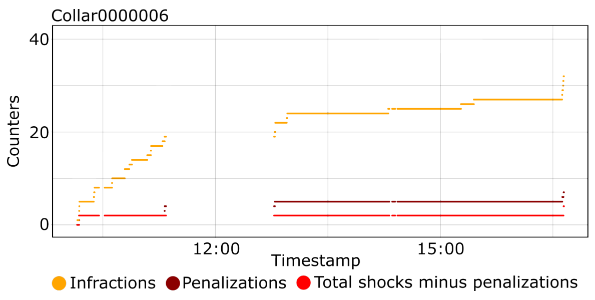
First day evaluation

Evaluating the data gathered during the first day of tests had as main purpose understanding the effectiveness of the training process which is considered fundamental to the success of the mechanism. Although sheep have been already faced with the system, there were a 3 weeks interval since the last use.

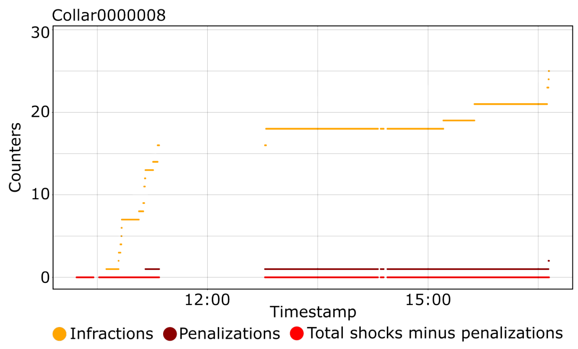
Figure 8.15 illustrates the evolution of relevant indicators for 4 different sheep (Collar 4, 6, 8 and A) with similar, expected and favourable behaviours. In all situations, a sharp growth in the number of detected infractions is observed in the beginning of the day (in the first hours of the morning), together with a small increasing in the number of penalizations. In contrast, in the afternoon period, we observe a stabilization of both indicators, being registered a smaller and less frequent number of infractions as well as an almost null increasing number of penalizations. This behaviour seems to be an evidence of a successful training process among all the sheep.



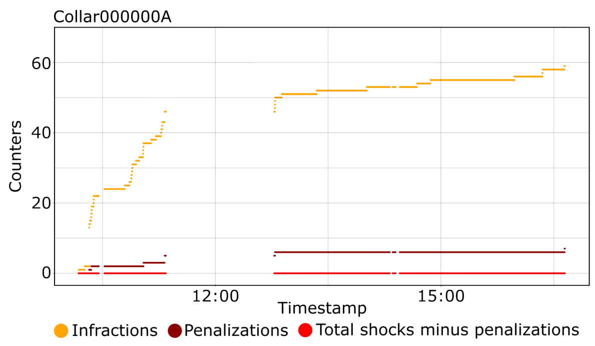
(a)



(b)



(c)



(d)

Figure 8.15: Evolution of posture control indicators of sheep with a favourable behaviour on day 1. (a) Collar 4. (b) Collar 6. (c) Collar 8. (d) Collar A.

Despite the similarities that enabled grouping the referred sheep, there are also some differences that must highlighted. The sheep's behaviour carrying the Collar 8 (Figure 8.15c) approaches what we considered to be an example, particularly because despite the expected higher number of detected infractions, the number of both infractions and penalizations keeps low. In fact, at the end of the day, only 2 penalizations were registered. A possible explanation for this superlative behaviour are two-fold. Firstly, the existence of some sort of memory about the mechanism may had accelerate the training process. Secondly, this particular sheep may present a higher cognitive capacity, allowing it to learn faster than the remaining ones. Concerning the remaining three cases (Figures 8.15a, 8.15b, 8.15d), all present a higher number of infractions and penalizations, even though the number of penalization keeps bounded (<10). Even so, a relevant event may be identified in Figure 8.15b. Beyond the presented cases, is the unique on which the difference between the number of electrostatic discharges and the number of penalizations is greater than zero, which suggests a higher insensitivity to the penalization cues. However, after this event, sheep presented a very favourable behaviour, reacting almost every time to the audio cues.

Though it could be inserted in the group identified as presenting a favourable behaviour, the indicators' evolution of Collar D are evaluated individually in Figure 8.16. Besides the expected high number of infractions in the beginning of the day (particularly during the morning) as it occurs with the previous examples, the same continues to occur during the afternoon. Nevertheless, besides the continuous and high increasing number of infractions, the number of penalizations keeps in a low value. This suggests that sheep successfully learned to associate the audio cues with the further penalizations but do not avoid forbidden behaviours, either because it was not able to associate the stimulus with the forbidden feeding behaviours or simply learned how to avoid the penalizations. Hence, although this sheep have continuous attempts to feed from the vines, the time it spends doing it is very short and insufficient to make visible damage on vines. In short, although this sheep seems to accomplish the training phase, it continuously persists on attempting undesired behaviours but for very short time intervals.

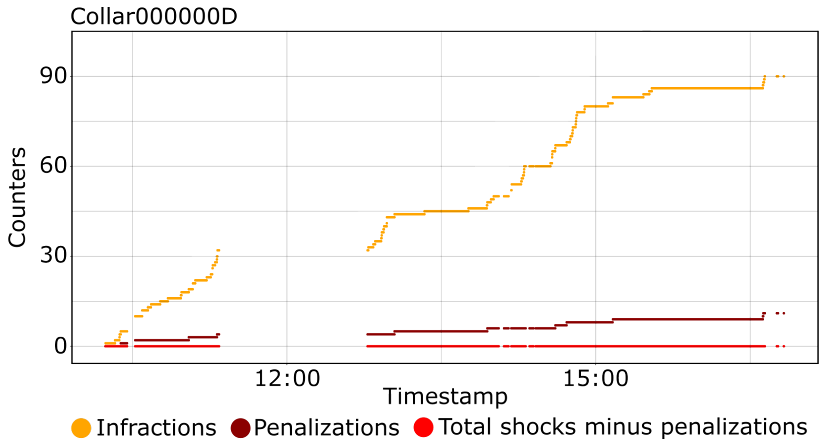


Figure 8.16: Evolution of posture control indicators of a sheep with a favourable but stubborn behaviour on day 1 - Collar D.

In contrast to previous examples, Figure 8.17 typifies one crass example of a refractory sheep. As it can be seen, the indicators continuously rise in a order of magnitude much greater than previous examples. Despite a certain level of mollification in the beginning of the day, probably due to some malfunctioning of the collar (it can be seen a time-interval without data), after around 10:20 a.m. the indicators do not stop increasing. Thus, it seems that this particular sheep does not react to any of the stimulus. Another interesting appointment is that besides the high number of stimulus, it was not possible to observe any kind of discomfort from the sheep. In fact, the first explanation to this event pointed to some malfunctioning of the collar. Notwithstanding, that hypothesis was not confirmed and that sheep was identified as refractory, being its use discouraged.

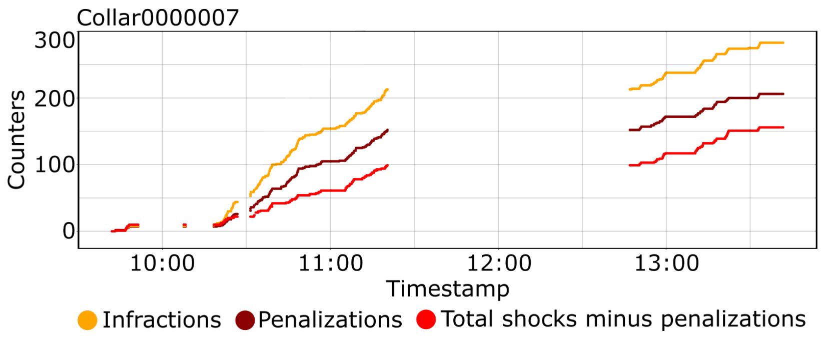
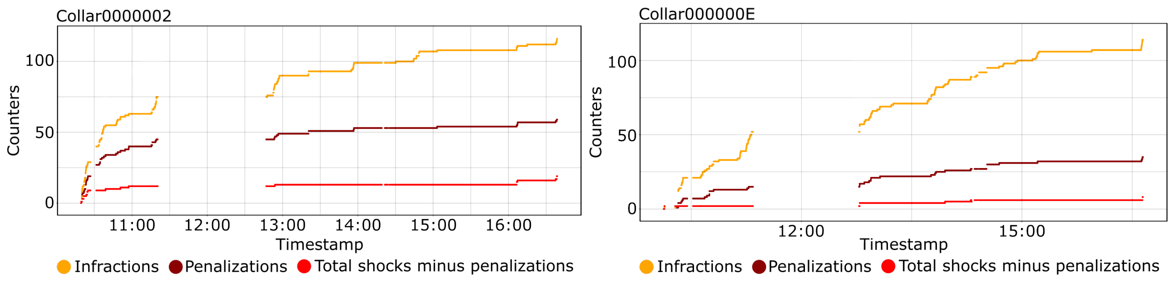


Figure 8.17: Evolution of posture control indicators of a refractory sheep on day 1 - Collar 7.

For the end are left two complex and dubious examples (Figure 8.18), but due to different reasons. Starting from the example of Collar 2 (Figure 8.18a), a massive increasing on all indicators is observed right in the beginning of the day, so that it seems that sheep does not have any kind of reaction. However, approximately after 10:30 am, sheep seems to start stabilizing its behaviour, although there are still sporadic increasings in penalizations' indicator, which may indicate some sort of failure of the training process. At least, during the morning, it seems that this particular sheep did not react to the audio cues (in the majority of the situations where a audio cue was registered, it was followed by penalization) and in several situations there was a need of a second penalization. The example of this sheep becomes even more complex when we analyse the afternoon period. Even if in the beginning of the afternoon (again when the system was initialized) there is a fast increasing of the number of infractions and penalizations, the order of magnitude of this growth is much smaller than the registered during the morning period. Also, we can observe that the infractions are more sporadic and spaced in time as well as the number of penalizations during the afternoon is much smaller than the recorded during the morning (almost 50 during the morning, and only 14 during the afternoon). Furthermore, we notice that the difference between the number of electrostatic discharges and the number of penalizations has a very small variation during the afternoon, which may indicate that sheep started to react to the first penalization. However, if we focus on the absolute values of all indicators, this sheep did not present an expected and favourable behaviour, being necessary to evaluate its behaviour in subsequent days.



(a) (b)

Figure 8.18: Evolution of posture control indicators on day 1. (a) Collar 2. (b) Collar E.

Likewise, also the analysis of Collar E (Figure 8.18b) is tricky and inconclusive. In fact, even if the increase of the indicators is not so intense as the observed for Collar 2, there are multiple instants where there is a relevant increase on such number. Furthermore, contrarily to the case of Collar 2, it is not observable a stabilization on those indicators, which may entail a flop of the training process. However, in contrast to Collar 2 and Collar 7 (associated to the refractory sheep), the difference between the number of electrostatic discharges and the number of penalizations is very small, which theoretically indicates some reaction of sheep to penalization cues. Nevertheless, it must be clarified that this sheep corresponds to the male of the group, presenting an anatomic structure much more robust (i.e. more weighted) than the remaining sheep. Thus, without neglecting the indicators presented, the observations taken suggest that although the male in fact reacts to stimuli, it is a very smooth reaction, giving the impression that it only reacts when it wants and depending on the behaviour of the remaining group.

Multi-day evaluation

Looking towards an evaluation of the repercussions on a multi-day basis of the posture control mechanism, the evolution of the indicators identified in the beginning of Section 8.3.2.2 were again considered. To simplify the comparison with the daily analysis, the same group of collars were maintained.

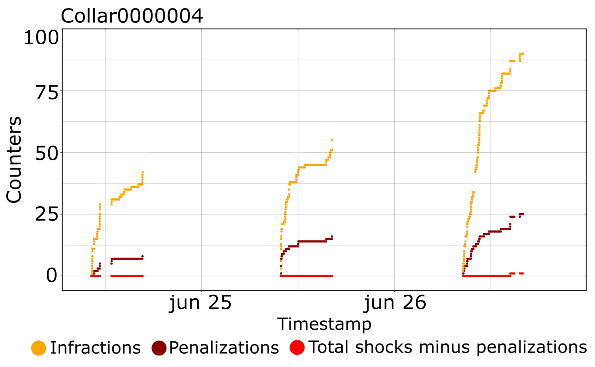
Figure 8.19 depicts the evolution for the first group, *i.e.*, Collar 4, 6, 8 and A. Regarding Collar 4 (Figure 8.19a), the behaviour observed during the second day is quite similar to the observed one during the first day, even if the number of infractions increased. In the third day, a significant increasing in the number of infractions and penalizations is observed. Although the difference between the number of electrostatic discharges and the number of penalizations kept almost null, the number of penalizations reached 25, almost three times more than in the first day. Hence, even if the sheep still keeps on reacting to the posture control mechanism, a deterioration on the behaviour is detected. Similarly, the same is observed in Figure 8.19b for Collar 8, although the number of penalizations and infractions is smaller than for Collar 4.

Figure 8.19c illustrates the evolution for Collar A. As observed, on the second day there was a superior behaviour compared to the first day. In fact, the sheep carrying Collar A was the sheep with the superlative behaviour among all sheep on day 2. Nevertheless, on day three, the behaviour got worse, even if registering an acceptable number of infractions and penalizations.

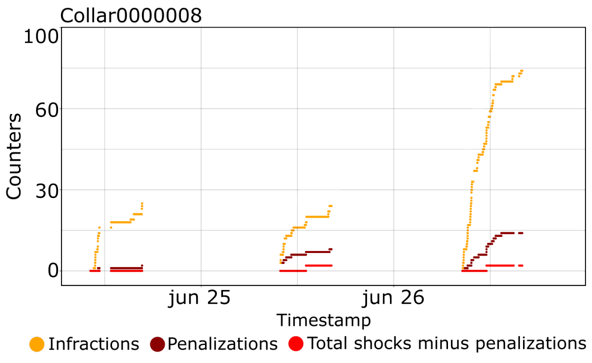
Collar 6 (Figure 8.19d) seems to loose the sensitivity to the posture control mechanism along the three days. Thus, a sheep that presented a favourable behaviour on day one, got into the refractory group on day three (on day two it is not clear yet the behaviour as a refractory one, but that gets clear on day three).

The sheep carrying Collar D (Figure 8.20) has shown on day one a stubborn behaviour, keeping insisting in feeding from vines, despite being observed a desired reaction to the conditioning mechanism. On day two a similar behaviour was observed during the morning, with a high increasing of the number of infraction and penalizations. Nevertheless, the values seem to stabilize during the rest of day. The same does not happen on day three, *i.e.*, also this sheep seems to loose some sensitivity to the stimuli.

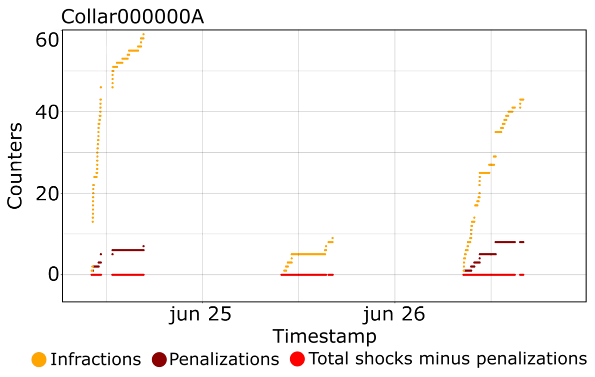
Figure 8.21 illustrates the indicators' evolution of the unique sheep identified clearly as refractory on day one. As observed, though the order of magnitude of all indicators have decreased comparing to day one, the behaviour is yet representative of a refractory sheep, with a high number of penalizations



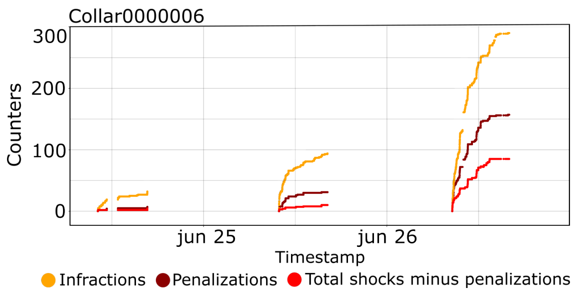
(a)



(b)



(c)



(d)

Figure 8.19: Evolution of posture control indicators regarding (a) Collar 4, (b) Collar 8, (c) Collar A and (d) Collar 6 on a multiday analysis.

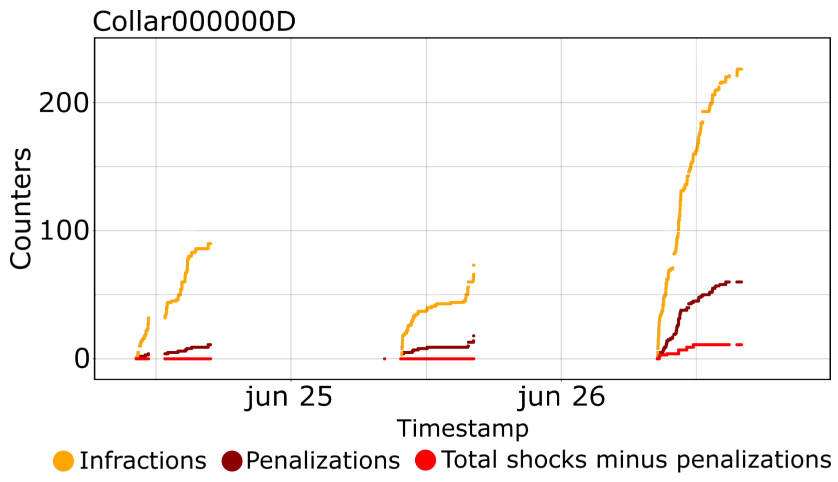


Figure 8.20: Evolution of posture control indicators regarding Collar D on a multi-day analysis.

and, particularly, with the difference between the number of electrostatic discharges and the number of penalizations reaching out high values, particularly on day 3.

Concerning the cases of Collar 2 and Collar E, identified as dubious on the day one evaluation, Figure 8.22 summarize the indicators' evolution for both sheep during the three days. Concerning Collar 2 (Figure 8.22a), contrary to the majority of other cases, it presented an enhanced behaviour

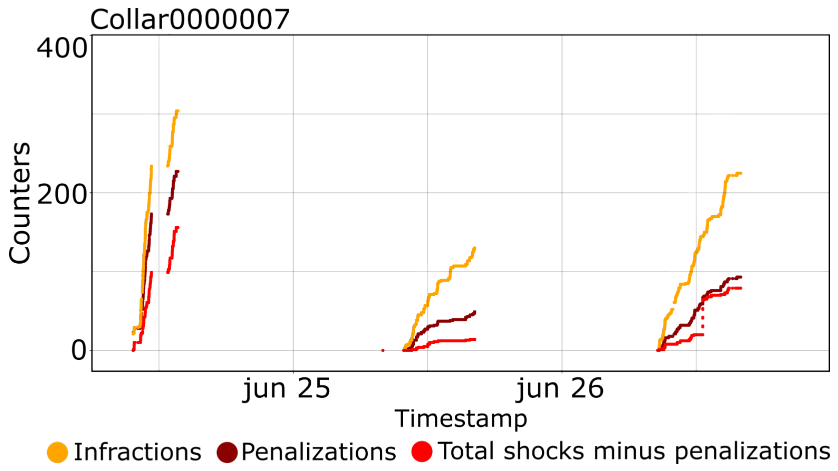
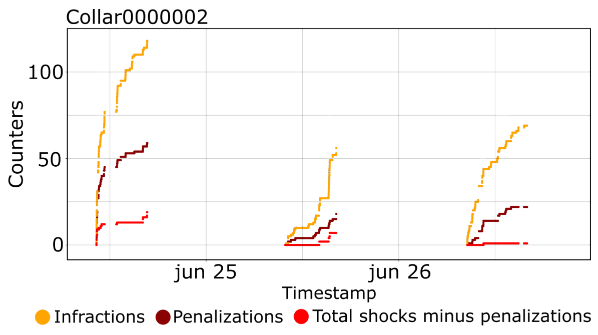


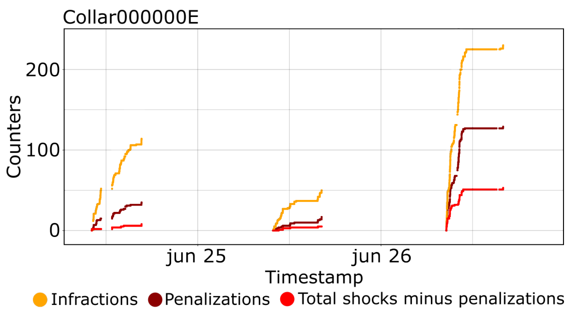
Figure 8.21: Evolution of posture control indicators regarding Collar 7 on a multi-day analysis.

on days two and three. Particularly, we can observe a reduction of more than 50% on all indicators, which is a good indicator. Still, on the end of day two, it is observed a sudden and inexplicable high increase on the indicators related to the electrostatic discharges, something that did not occurred on day 3. Thus, besides the inherent uncertainty it seems that this sheep started to deal better with the mechanism over the time.

In what concerns Collar E, the undesired but expected scenario was confirmed. As registered in Figure 8.22b, despite the favourable behaviour registered on day two, on day three, sheep did not respond to any of the stimuli, becoming totally refractory.



(a)



(b)

Figure 8.22: Evolution of posture control indicators regarding (a) Collar 2 and (b) Collar E on a multiday analysis.

In conclusion, despite the identified limitations of the experiment, there are a bundle of relevant conclusions that may be taken, namely:

- Sheep, indeed, do not have all the same cognitive learning capabilities neither the same sensitivity to stimuli;
- Albeit the posture control mechanism seems to achieve their intents on the beginning of its use, it is not clear if a predictive and static conditioning mechanism is enough to ensure an effective posture control mechanism;

- Some sheep seem to be able to learn throughout time, while other tend to start ignoring the mechanism (or at least seem to know how to deal with it);
- The third day was effectively the worse day among the three days evaluated. There are several reasons that may explain this occurrence, for instance, the increased hunger, the rise of the insensitivity against the stimuli or simply sheep started learning how to deal with the mechanism;
- Another additional conclusion, not directly extracted from the analysis provided, concerns the potential capability of sheep to recognize the pattern of the posture control mechanism and avoid it. In fact, manual observations taken suggest that some sheep seem to learn the timings of the posture control mechanism, feeding from the undesired branches exclusivity during the short period of time while the audio cues are applied;
- Though the technical implementation of the posture control mechanism could be validated in a real-scenario use-case, there is still work to be done particularly regarding the animal understanding field, for instance developing a deep study on how the stimuli configurations and sequences could be dynamically adjusted to become suitable and effective to all sheep.

8.4 Summary and discussion

This chapter aimed at obtaining all results that corroborate all research questions and consequently the thesis proposed.

Firstly, the prototypes and technologies adopted on the prototypes used on the evaluation tasks were presented. These allowed to define all system parameters that depend on implementation choices and that are required for evaluating the system. With the basis for the system evaluation, we started by validating the simulator built by means of two well known scenarios used recurrently in practice. In spite of slight differences between the values given by the simulator and the values measured in real scenarios, they do not jeopardize simulator's correctness.

Using the simulator, several scenarios were tested by varying critical system parameters. Two main scenarios were tested, one whose goal was to assess how the minimum duration of the μC evolves; and a second one to evaluate how the maximum number of collars, collars average consumption and autonomy, and medium occupation evolve for different MC and μC durations.

The achieved results demonstrate that the proposed system is able to fulfil the defined requirements, although for satisfying the need of supporting around one thousand of collars is only possible for a small number of beacons. We also shown that by reducing the maximum number of collars to a more realistic one (around 6 hundreds of collars), the system scale better in terms of number of beacons, offering more flexibility on the beacons distribution.

Notwithstanding, increasing the number of beacons also implies a higher collars' energy consumption since they are forced to be listening to the medium significantly more time. Hence, defining the system parameters requires a trade-off between beacon density (with impact in radio coverage and in the localization mechanism to be developed) and the autonomy desired.

Collars operating without stimuli easily reach the two weeks of continuous operation, corresponding to around 4 grazing weeks. Although satisfying the minimum requirements, collars operation still has room for optimizations seeking for higher autonomies, as desirable.

The second part of the chapter focused on the posture control mechanism in a context of use. Here, the evaluation was based on data and observations provided by external entities specialized in animals. Two evaluations were carried out, namely, the training process evaluation, *i.e.*, the cognitive capability of sheep to associate the stimuli with undesired behaviours and avoid them; and the response of sheep to the posture control mechanism in a real vineyard use-case scenario.

We have verified that, indeed, sheep have a inherent cognitive capacity of associating different types of stimuli with forbidden behaviours. Notwithstanding, although during the training process all sheep have showed reaction to stimuli, the same did not happen during a real vineyard use-case scenario. In fact, this later test showed, as expected, the existence of refractory sheep, *i.e.*, sheep that are not sensible to stimuli. Also, we confirmed that sheep do not respond all the same way, existing sheep with faster learning capabilities than others. Despite the potentiality of the posture control mechanism, the analysis of some indicators gathered during three days of test, showed a decreasing efficiency of the mechanism on the last day. In fact, some observations shown a potential capacity of sheep to adapt themselves to the predictive mechanism which, together with the increasing of the hunger, may have potentiated the increasing of infractions.

Nevertheless, besides the cons registered, the technical component of the system worked as designed, being also prepared to by adjusted by livestock experts, particularly concerning the configuration of the conditioning mechanism.

CHAPTER 9

Conclusions

This last chapter of the thesis concludes the work by summing up the achieved results and main contributions. Additionally, it discusses possible future work in the different areas tackled during this thesis.

The society is facing an incessant digital transformation that is allowing the physical and digital worlds to get closer, namely through the adoption of technological-based solutions. The increasing number of devices, together with their increasing capabilities and miniaturisation, is boosting the rise of new technological solutions in a continuously growing number of societal sectors, including sectors where typically exist some inertia in the adoption of technological solutions.

The agriculture sector is one of such sectors. Within it, weeding control is a process that impacts greatly both in quality production and profit margins, which is leveraging producers to seek for alternatives. Winemakers, particularly the ones that have their vineyards in steep and rough landscapes as it happens in Douro's region have a special interest. Here, using machinery is arduous and using herbicides is increasingly less interesting due to their threatening effects or even due to legislative constrains imposed. Therefore, the use of sheep, as it happened in ancient times, is potentially an alternative solution as far as we are capable of monitor and condition sheep's posture.

This challenge led to the raise of the SheepIT project and of this thesis, whose goal was to develop a technological solution capable of monitoring and conditioning sheep's posture as well as a facilitating animal management platform to enable the use of sheep as a weed control method. The focus of this thesis was to answer to three main research questions associated to the identified challenges. The first concerned the design and implementation of an IoT-based platform to meet the requirements of the solution. Despite the common requirements of an IoT solution, there are particular requirements that are specific to this problem, therefore we started by stating them and how they relate to the proposed architecture. With this definition we proposed an IoT-based architecture for intelligent farming not only adequate for the SheepIT project but also to similar applications. From the WSN devices to the computational platform, all components were defined.

Then, still related with the first research question, but focusing in the communication stack, we detailed the proposed approaches to ensure a low-power communication mechanism, particularly on collars that are the most constrained devices of the solution. A cyclic-based structure composed of different windows associated to different types of traffic (either CSMA or TDMA-based), together with a TT paradigm, was the proposed solution due to its known higher performance when the main goal is to have energy-wise efficient solutions.

From such architecture, we detailed the implementation of the communication stack, namely the layers that most contribute to the energy consumption of collars. We named the different windows of a μC, that is repeated over time to compose a MC. We provide the necessary equations to enable the calculation of the window's duration, which also enabled to identify the parameters that affect both μC and MC durations. Finally, we also defined the messages currently defined and that enable system communication.

To answer to the third research question, the posture control mechanism was presented. This mechanism is divided into two components that interact towards the goal of preventing sheep to weed from the higher branches of the vines. The animal monitoring algorithm is capable of identifying both the *Infracting* state and other common behaviours such as *Eating*, *Moving*, *Running* and *Resting* in real-time and directly within the collar through the implementation of a DT. When *Infracting* states are detected, the conditioning mechanism is triggered, consisting on a sequence of cues.

To validate the mechanisms proposed to answer to the research questions that support the thesis statement, two main processes were used. On the one hand, a tool for modelling the system, enabling the evaluation for different scenarios of critical metrics such as the maximum number of collars allowed, energy consumption and autonomy. A scenario is differentiated by several parameters as intended area to be covered, the LFP and the MP, just to cite the more relevant. On the other hand, data gathered on field trials were considered both to model the animal monitoring algorithm and test and assess the posture control mechanism as a whole.

The results demonstrate that the designed architecture is capable of scaling beyond the one thousand sheep while complying with the monitoring timing requirements if the area to cover is limited to 5 ha. For areas of 21 ha, the maximum number of collars decreases to 587. Concerning the expected autonomy, considering the required maximum periodicities LFP and MP (respectively 80 s and 6 s) and an area coverage of 144 ha, at least 416 h are supported if no stimuli are given. If stimuli are given, this value naturally decreases. However, scenarios on which the number of stimuli is high (particularly electrostatic stimuli) disclose the unavailability of the animal to learn the conditioning mechanism and hence that animal shall not be used.

This was one of the conclusions taken after analyzing data from field trials on which sheep grazed with the proposed system. In fact, despite being shown that sheep are capable of learning to associate audio cues with subsequent electrostatic stimuli and forbidden behaviours, it was also possible do conclude that not all sheep have the same capability of learning and that while some take more time to learn, others never learn, start ignoring or learn how to circumvent the posture control mechanism. Even though, the tests showed the suitability of the proposed solution, although future work on different areas is required.

9.1 Future Work

Despite the promising results achieved, there are several topics on which additional research is necessary aiming at improving existing solutions, either through the investigation of new approaches or through the optimization of existing ones. The most relevant open research lines are:

- **IoT-based architecture**: there are different topics that can be explored in future works, for instance:

- All μC types have the same duration. However, besides simpler, it may not be the optimal option. That becomes particularly relevant for the PR μC that is CSMA-based and where few communications are likely to occur. Therefore it is interesting to evaluate the viability of implementing μCs with different duration without increasing the current consumption of the system;
  - C2B messages currently have redundant and not optimized information. As collars are the most constrained devices, these messages should be shrunk such that only the necessary fields are transmitted;
  - For the TAW window, a upper bound static value was defined. Therefore it is not optimized for all use cases. As such its duration should be modelled considering system's parameters, for instance, number of collars, number of beacons and number of μCs;
  - Albeit a routing scheme was presented, it needs to be implemented and evaluated;
  - Additional security schemes need to be assessed to avoid that external actors may put in risk animals and productions;
  - B2B timeslots schedule is static and defined *a priori*. Automatic scheduling should be evaluated such that no reconfigurations would be required when moving beacon's infrastructure;
  - Assess the impact of increasing the radio *baudrate*, particularly on the communications efficiency, RSSI variability, packet loss and range.

- **RSSI-based localization**: the architecture designed included a requirement associated to the need of supporting RSSI-based localization mechanism. However, the work developed in the scope of this thesis only created the tools for further research. Although some preliminary work have been done and presented in [19], there is a long research journey up to a viable RSSI-based mechanism for herds. Future work includes:
  - Perform a comprehensive survey on the factors that affect RSSI measurements in a agricultural scenario, particularly when ovines graze in vineyards;
  - Implement an autonomous calibration mechanism to accommodate RSSI variations between hardware;
  - Collars implement a monitoring mechanism that allows to know, with a high level of accuracy, the behaviours being taken by sheep. Therefore, such information can be used to improve the accuracy of the localization mechanism;
  - Sheep graze typically close to each other. Therefore, collaborative information can be used also to improve localization accuracy;
- **Posture control:** Besides the good results achieved, there are still open research lines that can be followed regarding animal monitoring, namely:
  - There are still a considerable number of FP and FN regarding the *Infracting* state. Therefore, additional research should be done to seek for features capable of distinguishing both states;
  - Although there is still a *Resting* behaviour associated to moments of low intensity moves, it would be interesting to also identify when sheep are lying down. It is particularly relevant because such information could be used, for instance, to disable the algorithm or to reduce the reactivity index of collars;

- Field trials suggest that some sheep may be capable of learning the timings of the posture control mechanism and, therefore avoid them. Additional trials should be done to attest it and, if confirmed, a certain level of randomness on the measurements should be tested to avoid such issue.
- Long-run trials to guarantee that animal well-being is maintained while using the system and to assess the effectiveness of the posture control mechanism.

References

[1] T. Popović, N. Latinović, A. Pešić, Ž. Zečević, B. Krstajić, and S. Djukanović, “Architecting an IoT-enabled platform for precision agriculture and ecological monitoring: A case study,” *Computers and Electronics in Agriculture*, vol. 140, pp. 255–265, 2017. DOI: 10.1016/j.compag.2017.06.008.

[2] T. Ojha, S. Misra, and N. S. Raghuwanshi, “Wireless sensor networks for agriculture: The state-of-the-art in practice and future challenges,” vol. 118, pp. 66–84, 2015. DOI: 10.1016/j.compag.2015.08.011.

[3] Pergher, Gubiani, and Mainardis, “Field Testing of a Biomass-Fueled Flamer for In-Row Weed Control in the Vineyard,” vol. 9, no. 10, p. 210, 2019. DOI: 10.3390/agriculture9100210.

[4] A. Monteiro and I. Moreira, “Reduced rates of residual and post-emergence herbicides for weed control in vineyards,” *Weed Research*, vol. 44, no. 2, pp. 117–128, 2004. DOI: 10.1111/j.1365-3180.2004.00380.x.

[5] W. T. Lanini, G. T. McGourty, and L. A. Thrupp, “Weed management for organic vineyards,” Tech. Rep., 2011, pp. 69–82. [Online]. Available: http://agroecology.berkeley.edu/resources/weed%20management.pdf.

[6] M. Kennedy and P. Skinkis, “Are Your Weed-control Products Damaging Nearby Vineyards?” Tech. Rep., 2016. [Online]. Available: https://catalog.extension.oregonstate.edu/sites/catalog/files/project/pdf/em9132.pdf.

[7] C. Carlos, “Spraying challenges in the Douro Wine Region of Portugal,” Tech. Rep., 2014, pp. 2–18. [Online]. Available: http://www.advid.pt/imagens/comunicacoes/13993677624772.pdf.

[8] F. Dastgheib and C. Frampton, “Weed management practices in apple orchards and vineyards in the South Island of New Zealand,” *New Zealand Journal of Crop and Horticultural Science*, vol. 28, no. 1, pp. 53–58, 2000. DOI: 10.1080/01140671.2000.9514122.

[9] T. Bekkers, “Weed control options for commercial organic vineyards,” Tech. Rep., 2011, pp. 62–64. [Online]. Available: http://www.tobybekkers.com/uploads/5/4/3/2/5432540/bekkers-julyaug11wvj.pdf.

[10] *SheepIT Project*, Available online: http://www.av.it.pt/sheepit/. (visited on 03/30/2017).

[11] L. Nóbrega, P. Pedreiras, P. Gonçalves, and S. Silva, “Energy efficient design of a pasture sensor network,” in *Proceedings of the IEEE 5th International Conference on Future Internet of Things and Cloud, FiCloud 2017*, Prague, Czech Republic, 2017, pp. 91–98. DOI: 10.1109/FiCloud.2017.36.

[12] A. Temprilho, L. Nóbrega, P. Pedreiras, P. Gonçalves, and S. Silva, “M2M Communication stack for intelligent farming,” in *Proceedings of the Global Internet of Things Summit, GIoTS 2018*, Bilbao, Spain, 2018. DOI: 10.1109/GIOTS.2018.8534560.

[13] L. Nóbrega, P. Gonçalves, P. Pedreiras, and J. Pereira, “An IoT-based solution for intelligent farming,” *Sensors (Switzerland)*, vol. 19, no. 3, p. 603, 2019. DOI: 10.3390/s19030603.

[14] L. Nóbrega, A. Tavares, A. Cardoso, and P. Gonçalves, “Animal monitoring based on IoT technologies,” in *Proceedings of the IoT Vertical and Topical Summit on Agriculture - Tuscany, IOT Tuscany 2018*, Tuscany, Italy, 2018, pp. 1–5. DOI: 10.1109/IOT-TUSCANY.2018.8373045.

[15] L. Nóbrega, P. Pedreiras, and P. Gonçalves, "SheepIT, an IoT-Based Weed Control System," *Communications in Computer and Information Science*, vol. 953, pp. 131–147, 2019. DOI: 10.1007/978-3-030-12998-9-10.

[16] L. Nóbrega, P. Gonçalves, M. Antunes, and D. Corujo, "Assessing sheep behavior through low-power microcontrollers in smart agriculture scenarios," *Computers and Electronics in Agriculture*, vol. 173, p. 105 444, 2020. DOI: 10.1016/j.compag.2020.105444.

[17] L. Nóbrega, P. Pedreiras, and P. Gonçalves, "SheepIT - An electronic shepherd for the vineyards," in *Proceedings of the 8th International Conference on Information and Communication Technologies in Agriculture, Food & Environment (HAICTA)*, vol. 2030, Chania, Crete, Greece, 2017, pp. 621–632.

[18] L. Nóbrega, P. Gonçalves, P. Pedreiras, R. Morais, and A. Temprilho, "SheepIT: Automated Vineyard Weeding Control System," in *Proceedings of the INForum 2017: Simpósio de Informática*, Aveiro, Portugal, 2017.

[19] A. Cardoso, J. Pereira, L. Nóbrega, P. Gonçalves, P. Pedreiras, and V. Silva, "SheepIT: Activity and Location Monitoring," in *Proceedings of the INForum 2018 - Simpósio de Informática*, Coimbra, Portugal, 2018, pp. 1–12.

[20] R. Guedes, P. Pedreiras, L. Nóbrega, and P. Gonçalves, "Towards a low-cost localization system for small ruminants," *Computers and Electronics in Agriculture*, vol. 185, p. 106 172, 2021. DOI: 10.1016/j.compag.2021.106172.

[21] A. Al-Fuqaha, M. Guizani, M. Mohammadi, M. Aledhari, and M. Ayyash, "Internet of things: A survey on enabling technologies, protocols, and applications," *IEEE Communications Surveys and Tutorials*, vol. 17, no. 4, pp. 2347–2376, 2015.

[22] A. Rajandekar and B. Sikdar, "A survey of mac layer issues and protocols for machine-to-machine communications," *IEEE Internet of Things Journal*, vol. 2, no. 2, pp. 175–186, 2015. DOI: 10.1109/JIOT.2015.2394438.

[23] A. Aijaz and A. H. Aghvami, "Cognitive machine-to-machine communications for internet-of-things: A protocol stack perspective," *IEEE Internet of Things Journal*, vol. 2, no. 2, pp. 103–112, 2015. DOI: 10.1109/JIOT.2015.2390775.

[24] N. Accettura, M. R. Palattella, M. Dohler, L. A. Grieco, and G. Boggia, "Standardized power-efficient & internet-enabled communication stack for capillary m2m networks," in *Proceedings of the Wireless Communications and Networking Conference Workshops*, IEEE, Paris, France, 2012.

[25] Vodafone, "Narrowband-IoT: pushing the boundaries of IoT," Tech. Rep., 2017, pp. 1–15. [Online]. Available: https://www.vodacombusiness.co.za/cs/groups/public/documents/document/vodafone_nb%E2%80%93iot_white_paper_fi.pdf.

[26] P. Sethi and S. R. Sarangi, "Internet of Things: Architectures, Protocols, and Applications," vol. 2017, pp. 1–25, 2017. DOI: 10.1155/2017/9324035.

[27] N. Naik, "Choice of effective messaging protocols for iot systems: Mqtt, coap, amqp and http," in *Proceedings of the IEEE International Systems Engineering Symposium*, IEEE, Vienna, Austria, 2017. DOI: 10.1109/ISWCS.2010.5624516.

[28] N. Salman, I. Rasool, and A. H. Kemp, "Overview of the ieee 802.15. 4 standards family for low rate wireless personal area networks," in *Proceedings of the 7th International Symposium on Wireless Communication Systems*, IEEE, York, United Kingdom, 2010. DOI: 10.1109/ISWCS.2010.5624516.

[29] E. Vogli, G. Ribezzo, L. A. Grieco, and G. Boggia, "Fast join and synchronization schema in the ieee 802.15. 4e mac," in *Proceedings of the Wireless Communications and Networking*

Conference Workshops, IEEE, New Orleans, LA, USA, 2015. DOI: 10.1109/ISWCS.2010.5624516.

[30] Y. Al-Nidawi and A. H. Kemp, "Mobility aware framework for timeslotted channel hopping ieee 802.15. 4e sensor networks," *IEEE Sensors Journal*, vol. 15, no. 12, pp. 7112–7125, 2015. DOI: 10.1109/JSEN.2015.2472276.

[31] *Specifications | Z-Wave the Public Standard*, Available online: http://zwavepublic.com/specifications. (visited on 01/25/2018).

[32] EPCglobal Inc, "Specification for RFID Air Interface EPC ™ Radio-Frequency Identity Protocols Class-1 Generation-2 UHF RFID Protocol for Communications at 860 MHz – 960 MHz Version 1.2.0," Tech. Rep., 2006. [Online]. Available: https://www.gs1.org/sites/default/files/docs/epc/uhfc1g2_1_2_0-standard-20080511.pdf.

[33] S. Gajjar, N. Choksi, and M. Sarkar, "LEFT: A Latency and Energy Efficient Flexible TDMA Protocol for Wireless Sensor Networks," *International Journal of Computer Network and Information Security 7.2*, vol. 7, pp. 1–14, 2015. DOI: 10.5815/ijcnis.2015.02.01.

[34] W. Wang, J. Li, Y. Lu, and C. Liu, "Optimization Method of Adaptive Backoff and Duty Cycle for S-MAC Protocol in Wireless Sensor Networks," *IEEE Access*, vol. 9, pp. 15 066–15 073, 2021. DOI: 10.1109/ACCESS.2021.3053328.

[35] B. Jang, J. B. Lim, and M. L. Sichitiu, "An asynchronous scheduled MAC protocol for wireless sensor networks," vol. 57, no. 1, pp. 85–98, 2013. DOI: 10.1016/j.comnet.2012.09.002.

[36] A. Klein, "Preamble-Based Medium Access in Wireless Sensor Networks," in *Wireless Sensor Networks - Technology and Protocols*, InTechOpen, 2012. DOI: 10.5772/48504.

[37] A. El-Hoiydi, "Spatial TDMA and CSMA with preamble sampling for low power ad hoc wireless sensor networks," 2002, pp. 685–692. DOI: 10.1109/ISCC.2002.1021748.

[38] M. Z. Hasan and F. Al-Turjman, "Evaluation of a duty-cycled asynchronous X-MAC protocol for vehicular sensor networks," *Eurasip Journal on Wireless Communications and Networking*, vol. 2017, no. 1, p. 95, 2017. DOI: 10.1186/s13638-017-0882-7.

[39] A. El-Hoiydi and J. D. Decotignie, "WiseMAC: An ultra low power MAC protocol for multi-hop wireless sensor networks," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 3121, no. 5005, pp. 18–31, 2004. DOI: 10.1007/978-3-540-27820-7_4.

[40] C. Cano, B. Bellalta, A. Sfairopoulou, and M. Oliver, "Low energy operation in WSNs: A survey of preamble sampling MAC protocols," *Computer Networks*, vol. 55, no. 15, pp. 3351–3363, 2011. DOI: 10.1016/j.comnet.2011.06.022.

[41] G. F. G. Fang and E. Dutkiewicz, "BodyMAC: Energy efficient TDMA-based MAC protocol for Wireless Body Area Networks," 2009, pp. 1455–1459. DOI: 10.1109/ISCIT.2009.5341045.

[42] T. Crenshaw, S. Hoke, A. Tirumala, and M. Caccamo, "Robust implicit edf: A wireless mac protocol for collaborative real-time systems," *ACM Transactions on Embedded Computing Systems (TECS)*, vol. 6, no. 4, 28–es, 2007. DOI: 10.1145/1274858.1274866.

[43] I. Rhee, A. Warrier, M. Aia, J. Min, and M. Sichitiu, "Z-MAC: A Hybrid MAC Protocol for Wireless Sensor Networks," *IEEE/ACM Transactions on Networking*, vol. 16, no. 3, pp. 511–524, 2008. DOI: 10.1109/TNET.2007.900704.

[44] W. Lee, A. Datta, and R. Cardell-Oliver, "FlexiTP: A Flexible-Schedule-Based TDMA Protocol for Fault-Tolerant and Energy-Efficient Wireless Sensor Networks," *IEEE Transactions on Parallel and Distributed Systems*, vol. 19, no. 6, pp. 851–864, 2008. DOI: 10.1109/TPDS.2007.70774.

[45] S. Zhuo, Y.-Q. Song, Z. Wang, and Z. Wang, “Queue-mac: A queue-length aware hybrid csma/tdma mac protocol for providing dynamic adaptation to traffic and duty-cycle variation in wireless sensor networks,” in *Proceedings of the 9th IEEE International Workshop on Factory Communication Systems*, 2012, pp. 105–114. DOI: 10.1109/WFCS.2012.6242552.

[46] S. Zhuo, Z. Wang, Y. Q. Song, Z. Wang, and L. Almeida, “A Traffic Adaptive Multi-Channel MAC Protocol with Dynamic Slot Allocation for WSNs,” *IEEE Transactions on Mobile Computing*, vol. 15, no. 7, pp. 1600–1613, 2016. DOI: 10.1109/TMC.2015.2473852.

[47] J. Song, S. Han, A. K. Mok, D. Chen, M. Lucas, M. Nixon, and W. Pratt, “WirelessHART: Applying wireless technology in real-time industrial process control,” in *Proceedings of the IEEE Real-Time and Embedded Technology and Applications Symposium, RTAS*, 2008, pp. 377–386. DOI: 10.1109/RTAS.2008.15.

[48] ISA, *ISA Standard - Wireless systems for industrial automation: process control and related applications.* [Online]. Available: https://www.isa.org/products/ansi-isa-100-11a-2011-wireless-systems-for-industr (visited on 03/30/2017).

[49] M. R. Palattella, P. Thubert, X. Vilajosana, T. Watteyne, Q. Wang, and T. Engel, “6tiSCH wireless industrial networks: Determinism meets IPv6,” in *Smart Sensors, Measurement and Instrumentation*, vol. 9, Springer International Publishing, 2014, pp. 111–141. DOI: 10.1007/978-3-319-04223-7_5.

[50] N. Kushalnagar, G. Montenegro, and C. Schumacher, “Ipv6 over low-power wireless personal area networks (6lowpans): Overview, assumptions, problem statement, and goals,” Tech. Rep., 2007. [Online]. Available: https://rfc-editor.org/rfc/rfc4919.txt.

[51] I. Farris, S. Pizzi, M. Merenda, A. Molinaro, R. Carotenuto, and A. Iera, “6lo-rfid: A framework for full integration of smart uhf rfid tags into the internet of things,” *IEEE Network*, vol. 31, no. 5, pp. 66–73, 2017. DOI: 10.1109/MNET.2017.1600269.

[52] T. Winter, P. Thubert, A. Brandt, J. Hui, R. Kelsey, P. Levis, K. Pister, R. Struik, J. P. Vasseur, and R. Alexander, “Rpl: Ipv6 routing protocol for low-power and lossy networks,” Tech. Rep., 2012. [Online]. Available: https://www.net.in.tum.de/fileadmin/TUM/NET/NET-2011-07-1/NET-2011-07-1_09.pdf.

[53] J. Postel, “RFC 793: Transmission Control Protocol,” Tech. Rep., 1981. [Online]. Available: https://datatracker.ietf.org/doc/html/rfc793.

[54] ——, “RFC 768: User Datagram Protocol,” Tech. Rep., 1980. [Online]. Available: https://www.ietf.org/rfc/rfc768.txt#:~:text=This%20User%20Datagram%20Protocol%20(UDP,used%20as%20the%20underlying%20protocol..

[55] Z. Shelby, K. Hartke, and C. Bormann, “The Constrained Application Protocol (CoAP),” Tech. Rep., 2014, p. 112. [Online]. Available: https://www.rfc-editor.org/rfc/pdfrfc/rfc7252.txt.pdf.

[56] M. Arrot and et. al., “Amqp, advanced message queuing protocol - protocol specification,” Tech. Rep., 2006. [Online]. Available: https://www.rabbitmq.com/resources/specs/amqp0-9-1.pdf.

[57] O. Deschambault, A. Gherbi, and C. Légaré, “Efficient implementation of the mqtt protocol for embedded systems,” *Journal of Information Processing Systems*, vol. 13, no. 1, pp. 26–39, 2017. DOI: 10.3745/JIPS.04.0028.

[58] I. Grigorik, “Making the web faster with http 2.0,” *Commun. ACM*, vol. 56, no. 12, pp. 42–49, 2013. DOI: 10.1145/2534706.2534721.

[59] *Sigfox Technology Overview*, Available online: https://www.sigfox.com/en/sigfox-iot-technology-overview. (visited on 01/25/2018).

[60] L. Alliance, Available online: https://www.tuv.com/media/corporate/products_1/electronic_components_and_lasers/TUeV_Rheinland_Overview_LoRa_and_LoRaWANtmp.pdf. (visited on 01/25/2018).

[61] K.-H. Lam, C.-C. Cheung, and W.-C. Lee, "Lora-based localization systems for noisy outdoor environment," in *Proceedings of the IEEE 13th International Conference on Wireless and Mobile Computing, Networking and Communications*, IEEE, Rome, Italy, 2017. DOI: 10.1109/WiMOB.2017.8115843.

[62] *LoRa | Geolocation | Zakelijk KPN Forum*, Available online: https://zakelijkforum.kpn.com/lora-forum-16/lora-geolocation-8555. (visited on 01/18/2019).

[63] F. Baccelli, B. Blaszczyszyn, and P. Muhlethaler, "An aloha protocol for multihop mobile wireless networks," *IEEE Transactions on Information Theory*, vol. 52, no. 2, pp. 421–436, 2006. DOI: 10.1109/TIT.2005.862098.

[64] Y.-P. E. Wang, X. Lin, A. Adhikary, A. Grovlen, Y. Sui, Y. Blankenship, J. Bergman, and H. S. Razaghi, "A Primer on 3GPP Narrowband Internet of Things," *IEEE Communications Magazine*, vol. 55, no. 3, pp. 117–123, 2017. DOI: 10.1109/MCOM.2017.1600510CM.

[65] R. Ratasuk, N. Mangalvedhe, A. Ghosh, and B. Vejlgaard, "Narrowband lte-m system for m2m communication," in *Proceedings of the IEEE 80th Vehicular Technology Conference*, Vancouver, Canada, 2014. DOI: 10.1109/VTCFall.2014.6966070.

[66] Q. Zhu, R. Wang, Q. Chen, Y. Liu, and W. Qin, "Iot gateway: Bridging wireless sensor networks into internet of things," in *Proceedings of the IEEE/IFIP 8th International Conference onEmbedded and Ubiquitous Computing*, IEEE, Hong Kong, China, 2010. DOI: 10.1109/EUC.2010.58.

[67] S. Guoqiang, C. Yanming, Z. Chao, and Z. Yanxu, "Design and implementation of a smart iot gateway," in *Proceedings of the IEEE International Conference on Green Computing and Communications and IEEE Internet of Things and IEEE Cyber, Physical and Social Computing*, IEEE, Beijing, China, 2013. DOI: 10.1109/GreenCom-iThings-CPSCom.2013.130.

[68] G. Aloi, G. Caliciuri, G. Fortino, R. Gravina, P. Pace, W. Russo, and C. Savaglio, "A mobile multi-technology gateway to enable iot interoperability," in *Proceedings of the IEEE First International Conference on Internet-of-Things Design and Implementation*, IEEE, Berlin, Germany, 2016. DOI: 10.1109/IoTDI.2015.29.

[69] T. Zachariah, N. Klugman, B. Campbell, J. Adkins, N. Jackson, and P. Dutta, "The internet of things has a gateway problem," in *Proceedings of the 16th international workshop on mobile computing systems and applications*, ACM, Santa Fe, NM, USA, 2015. DOI: 10.1145/2699343.2699344.

[70] S. K. Datta, C. Bonnet, and N. Nikaein, "An IoT gateway centric architecture to provide novel M2M services," in *Proceedings of the IEEE World Forum on Internet of Things 2014*, Seoul, Korea, 2014, pp. 514–519. DOI: 10.1109/WF-IoT.2014.6803221.

[71] A. Rodriguez, "Restful web services: The basics," Tech. Rep., 2008. [Online]. Available: https://developer.ibm.com/technologies/web-development/articles/ws-restful/.

[72] *OneM2M*, Available online: http://www.onem2m.org/. (visited on 01/16/2019).

[73] A. Al-Fuqaha, A. Khreishah, M. Guizani, A. Rayes, and M. Mohammadi, "Toward better horizontal integration among IoT services," *IEEE Communications Magazine*, vol. 53, no. 9, pp. 72–79, 2015. DOI: 10.1109/MCOM.2015.7263375.

[74] *The Internet of Things (IoT) Starts with Intel Inside®*, Available online: https://www.intel.com/content/www/us/en/internet-of-things/overview.html. (visited on 01/16/2019).

[75] *Smart M2M ToR*, Available online: https://portal.etsi.org/TBSiteMap/SmartM2M/SmartM2MToR.aspx. (visited on 01/16/2019).

[76] *Lightweight M2M (LWM2M) - OMA SpecWorks*, Available online: https://www.omaspecworks.org/what-is-oma-specworks/iot/lightweight-m2m-lwm2m/. (visited on 01/16/2019).

[77] K. Heurtefeux and F. Valois, "Is RSSI a good choice for localization in wireless sensor network?" In *Proceedings of the International Conference on Advanced Information Networking and Applications, AINA*, 2012, pp. 732–739. DOI: 10.1109/AINA.2012.19.

[78] A. Srinivasan and J. Wu, "A Survey on Secure Localization in Wireless Sensor Networks," *Encyclopedia of Wireless and Mobile communications*, vol. 3, no. 11, pp. 1–26, 2007. DOI: 10.1007/s11227-010-0501-4.

[79] B. Mert, L. Min, S. Weiming, and G. Hamada, "Localization in cooperative wireless sensor networks: A review," in *Proceedings of the 13th International Conference on Computer Supported Cooperative Work in Design*, 2009, pp. 438–443. DOI: 10.1109/CSCWD.2009.4968098.

[80] A. Boukerche, H. A. Oliveira, E. F. Nakamura, and A. A. Loureiro, "Localization systems for wireless sensor networks," *IEEE Wireless Communications*, vol. 14, no. 6, pp. 6–12, 2007. DOI: 10.1109/MWC.2007.4407221.

[81] M. Farooq-i-Azam and M. N. Ayyaz, "Location and position estimation in wireless sensor networks," in *Wireless Sensor Networks: Current Status and Future Trends*, 2016, pp. 179–214. DOI: 10.1201/b13092-8.

[82] S. P. Singh and S. C. Sharma, "Range Free Localization Techniques in Wireless Sensor Networks: A Review," vol. 57, pp. 7–16, 2015. DOI: 10.1016/j.procs.2015.07.357.

[83] H. Peng, F. Long, and C. Ding, "Feature selection based on mutual information: Criteria of Max-Dependency, Max-Relevance, and Min-Redundancy," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 8, pp. 1226–1238, 2005. DOI: 10.1109/TPAMI.2005.159.

[84] T. C. Karalar and J. Rabaey, "An RF ToF based ranging implementation for sensor networks," in *Proceedings of the IEEE International Conference on Communications*, vol. 7, 2006, pp. 3347–3352. DOI: 10.1109/ICC.2006.255233.

[85] J. Xu, W. Liu, F. Lang, Y. Zhang, and C. Wang, "Distance Measurement Model Based on RSSI in WSN," *Wireless Sensor Network*, vol. 02, no. 08, pp. 606–611, 2010. DOI: 10.4236/wsn.2010.28072.

[86] J. Wang, R. K. Ghosh, and S. K. Das, "A survey on sensor localization," *Journal of Control Theory and Applications*, vol. 8, no. 1, pp. 2–11, 2010. DOI: 10.1007/s11768-010-9187-7.

[87] P. Abouzar, D. G. Michelson, and M. Hamdi, "RSSI-Based Distributed Self-Localization for Wireless Sensor Networks Used in Precision Agriculture," *IEEE Transactions on Wireless Communications*, vol. 15, no. 10, pp. 6638–6650, 2016. DOI: 10.1109/TWC.2016.2586844.

[88] C. Alippi and G. Vanini, "A RSSI-based and calibrated centralized localization technique for wireless sensor networks," in *Proceedings of the Fourth Annual IEEE International Conference on Pervasive Computing and Communications Workshops, PerCom Workshops*, vol. 2006, 2006, pp. 301–305. DOI: 10.1109/PERCOMW.2006.13.

[89] V. Daiya, J. Ebenezer, S. A. Murty, and B. Raj, "Experimental analysis of RSSI for distance and position estimation," in *Proceedings of the International Conference on Recent Trends in Information Technology, ICRTIT 2011*, 2011, pp. 1093–1098. DOI: 10.1109/ICRTIT.2011.5972367.

[90] S. Hartung, H. Brosenne, and D. Hogrefe, "Practical RSSI long distance measurement evaluation in wireless sensor networks," in *Proceedings of the IEEE Conference on Wireless Sensor*, 2013, pp. 1–6. DOI: 10.1109/ICWISE.2013.6728770.

[91] D. Konings, N. Faulkner, F. Alam, F. Noble, and E. Lai, "Do RSSI values reliably map to RSS in a localization system?" In *Proceedings of the 2nd Workshop on Recent Trends in Telecommunications Research*, 2017. DOI: 10.1109/RTTR.2017.7887867.

[92] S. Pradhan and S. S. Hwang, "Mathematical analysis of line intersection algorithm for TOA trilateration method," in *Proceedings of the joint 7th International Conference on Soft Computing and Intelligent Systems and 15th International Symposium on Advanced Intelligent Systems*, 2014, pp. 1219–1223. DOI: 10.1109/SCIS-ISIS.2014.7044849.

[93] S. Yiu, M. Dashti, H. Claussen, and F. Perez-Cruz, "Wireless RSSI fingerprinting localization," *Signal Processing*, vol. 131, pp. 235–244, Feb. 2017. DOI: 10.1016/j.sigpro.2016.07.005.

[94] B. D. Lakmali and D. Dias, "Database correlation for GSM location in outdoor and indoor environments," in *Proceedings of the 4th International Conference on Information and Automation for Sustainability*, 2008, pp. 42–47. DOI: 10.1109/ICIAFS.2008.4783992.

[95] U. Fayyad, G. Piatetsky-Shapiro, and P. Smyth, "From data mining to knowledge discovery in databases," *AI Magazine*, vol. 17, no. 3, pp. 37–53, 1996. DOI: 10.1609/aimag.v17i3.1230.

[96] Alexsoft, "Machine Learning: Bridging Between Business and Data Science," Tech. Rep., pp. 1–21. [Online]. Available: https://www.altexsoft.com/media/2017/04/Machine-Learning.pdf.

[97] J. Gama, A. C. Lorena, K. Faceli, M. Oliveira, and A. P. d. L. Carvalho, *Extração de conhecimento de dados*. Edições Sílabo, 2015.

[98] T. M. Mitchell, *Machine Learning*. McGraw Hill, 1997.

[99] M. Antunes, "Knowledge extraction from semi-structured sources," Ph.D. dissertation, University of Aveiro, Minho and Porto, 2018. DOI: 10.13140/RG.2.2.25430.70726.

[100] A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: A review," vol. 31, no. 3, pp. 264–323, 1999. DOI: 10.1145/331499.331504.

[101] J. F. Cardoso, "Blind signal separation: Statistical principles," *Proceedings of the IEEE*, vol. 86, no. 10, pp. 2009–2025, 1998. DOI: 10.1109/5.720250.

[102] G. E. Batista and M. C. Monard, "An analysis of four missing data treatment methods for supervised learning," *Applied Artificial Intelligence*, vol. 17, no. 5-6, pp. 519–533, 2003. DOI: 10.1080/713827181.

[103] V. J. Hodge and J. Austin, "A survey of outlier detection methodologies," *Artificial Intelligence Review*, vol. 22, no. 2, pp. 85–126, 2004. DOI: 10.1023/B:AIRE.0000045502.10941.a9.

[104] P. Sondhi, "Feature construction methods: a survey," *Sifaka. Cs. Uiuc. Edu*, vol. 69, pp. 70–71, 2010.

[105] M. Zhang and A. A. Sawchuk, "A feature selection-based framework for human activity recognition using wearable multimodal sensors," in *Proceedings of the 6th International ICST Conference on Body Area Networks*, 2012, pp. 92–98. DOI: 10.4108/icst.bodynets.2011.247018.

[106] J. Li, K. Cheng, S. Wang, F. Morstatter, R. P. Trevino, J. Tang, and H. Liu, "Feature selection: A data perspective," *ACM Computing Surveys*, vol. 50, no. 6, p. 94, 2017. DOI: 10.1145/3136625.

[107] Z. Zhao and H. Liu, “Spectral feature selection for supervised and unsupervised learning,” in *Proceedings of the 24th international conference on Machine learning*, vol. 227, 2007, pp. 1151–1157. DOI: 10.1145/1273496.1273641.

[108] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification.* John Wiley & Sons, 2012.

[109] F. Nie, S. Xiang, Y. Jia, C. Zhang, and S. Yan, “Trace ratio criterion for feature selection,” in *Proceedings of the 23th National Conference on Artificial Intelligence*, vol. 2, 2008, pp. 671–676. DOI: 10.5555/1620163.1620176.

[110] M. Robnik-Šikonja and I. Kononenko, “Theoretical and Empirical Analysis of ReliefF and RReliefF,” *Machine Learning*, vol. 53, no. 1-2, pp. 23–69, 2003. DOI: 10.1023/A:1025667309714.

[111] D. D. Lewis, “Feature selection and feature extraction for text categorization,” in *Proceedings of the workshop on Speech and Natural Language*, A. f. C. Linguistics, Ed., 1992, pp. 212–217. DOI: 10.3115/1075527.1075574.

[112] R. Battiti, “Using Mutual Information for Selecting Features in Supervised Neural Net Learning,” *IEEE Transactions on Neural Networks*, vol. 5, no. 4, pp. 537–550, 1994. DOI: 10.1109/72.298224.

[113] D. Lin and X. Tang, “Conditional infomax learning: An integrated framework for feature extraction and fusion,” in *Proceedings of the Computer Vision – ECCV 2006*, 2006, pp. 68–82. DOI: 10.1007/11744023_6.

[114] H. H. Yang and J. Moody, “Data visualization and feature selection: New algorithms for nongaussian data,” in *Proceedings of the 12th International Conference on Neural Information Processing Systems*, 1999, pp. 687–693.

[115] F. Fleuret, “Fast binary feature selection with conditional mutual information,” *Journal of Machine Learning Research*, vol. 5, pp. 1531–1555, 2004. DOI: 10.5555/1005332.1044711.

[116] M. Vidal-Naquet and S. Ullman, “Object recognition with informative features and linear classification,” in *Proceedings of the IEEE International Conference on Computer Vision*, vol. 1, 2003, pp. 281–288. DOI: 10.1109/iccv.2003.1238356.

[117] A. Jakulin, “Machine Learning Based on Attribute Interactions,” Ph.D. dissertation, Univerza v Ljubljani, 2005, pp. 1–252.

[118] P. E. Meyer and G. Bontempi, “On the use of variable complementarity for feature selection in cancer classification,” in *Proceedings of the EvoWorkshops 2006: Applications of Evolutionary Computing*, 2006, pp. 91–102. DOI: 10.1007/11732242_9.

[119] L. Yu and H. Liu, “Feature Selection for High-Dimensional Data: A Fast Correlation-Based Filter Solution,” in *Proceedings of the 20th International Conference on Machine Learning*, vol. 2, 2003, pp. 856–863, ISBN: 1577351894.

[120] T. Hastie, R. Tibshirani, and M. Wainwright, *Statistical learning with sparsity: The lasso and generalizations.* 2015, pp. 1–337. DOI: 10.1201/b18401.

[121] J. Liu, S. Ji, and J. Ye, “Multi-task feature learning via efficient l2, 1-norm minimization,” in *Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence*, 2009.

[122] F. Nie, H. Huang, X. Cai, and C. Ding, “Efficient and robust feature selection via joint l2,1-norms minimization,” in *Proceedings of the 24th Annual Conference on Neural Information Processing Systems*, 2010.

[123] R. A. Reyment and J. C. Davis, “Statistics and Data Analysis in Geology.,” *Biometrics*, vol. 44, no. 3, p. 918, 1988. DOI: 10.2307/2531613.

[124] S. Wright, “The Interpretation of Population Structure by F-Statistics with Special Regard to Systems of Mating,” *Evolution*, vol. 19, no. 3, p. 395, 1965. DOI: 10.2307/2406450.

[125] H. Liu and R. Setiono, “Chi2: feature selection and discretization of numeric attributes,” in *Proceedings of the International Conference on Tools with Artificial Intelligence*, 1995, pp. 388–391. DOI: 10.1109/tai.1995.479783.

[126] C. W. Gini, “Variability and mutability, contribution to the study of statistical distributions and relations,” *J. American Statistical Association*, vol. 66, pp. 534–544, 1971.

[127] M. Hall and L. a. Smith, “Feature Selection for Machine Learning : Comparing a Correlation-based Filter Approach to the Wrapper CFS : Correlation-based Feature,” 1999. DOI: 10.1.1.50.2192.

[128] A. Tharwat, “Linear vs. quadratic discriminant analysis classifier: a tutorial,” vol. 3, no. 2, p. 145, 2016. DOI: 10.1504/IJAPR.2016.079050.

[129] R. A. Fisher, “the Use of Multiple Measurements in Taxonomic Problems,” *Annals of Eugenics*, vol. 7, no. 2, pp. 179–188, 1936. DOI: 10.1111/j.1469-1809.1936.tb02137.x.

[130] C. R. Rao, “The Utilization of Multiple Measurements in Problems of Biological Classification,” *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 10, no. 2, pp. 159–193, 1948. DOI: 10.1111/j.2517-6161.1948.tb00008.x.

[131] H. Zhang, “The optimality of Naive Bayes,” in *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference*, 2004.

[132] J. H. Jurafsky, Daniel and Martin, *Speech and Language Processing: An introduction to natural language processing.* Prentice-Hall, Inc., 2009.

[133] J. Eisner, “An interactive spreadsheet for teaching the forward-backward algorithm,” in *Proceedings of the Workshop on Effective Tools and Methodologies for Teaching Natural Language Processing and Computational Linguistics*, 2002, pp. 10–18. DOI: 10.3115/1118108.1118110.

[134] A. Krogh, “Chapter 4 An introduction to hidden Markov models for biological sequences,” in *New Comprehensive Biochemistry*, C, vol. 32, Elsevier, 1998, pp. 45–63. DOI: 10.1016/S0167-7306(08)60461-5.

[135] J. Badajena and C. Rout, “Incorporating Hidden Markov Model into Anomaly Detection Technique for Network Intrusion Detection,” *International Journal of Computer Applications*, vol. 53, no. 11, pp. 42–47, 2012. DOI: 10.5120/8469-2395.

[136] J. A. Nelder and R. W. M. Wedderburn, “Generalized Linear Models,” *Journal of the Royal Statistical Society. Series A (General)*, vol. 135, no. 3, p. 370, 1972. DOI: 10.2307/2344614.

[137] C. Cortes and V. Vapnik, “Support-Vector Networks,” *Machine Learning*, vol. 20, no. 3, pp. 273–297, 1995. DOI: 10.1023/A:1022627411411.

[138] J. A. Suykens, “Nonlinear modelling and support vector machines,” in *Proceedings of the 18th IEEE Instrumentation and Measurement Technology Conference*, vol. 1, 2001, pp. 287–294. DOI: 10.1109/imtc.2001.928828.

[139] C. Crisci, B. Ghattas, and G. Perera, *A review of supervised machine learning algorithms and their applications to ecological data*, 2012. DOI: 10.1016/j.ecolmodel.2012.03.001.

[140] E. Fix and J. Hodges, “Discriminatory Analysis, Nonparametric Discrimination: Consistency Properties,” *International Statistical Review / Revue Internationale de Statistique*, no. 3, pp. 238–247, 1951. DOI: 10.2307/1403797.

[141] M. Beckmann, N. F. F. Ebecken, and B. S. L. Pires de Lima, “A KNN Undersampling Approach for Data Balancing,” *Journal of Intelligent Learning Systems and Applications*, vol. 07, no. 04, pp. 104–116, 2015. DOI: 10.4236/jilsa.2015.74010.

[142] S. Boriah, V. Chandola, and V. Kumar, "Similarity measures for categorical data: A comparative evaluation," in *Proceedings of the Society for Industrial and Applied Mathematics - 8th SIAM International Conference on Data Mining 2008*, vol. 1, 2008, pp. 243–254. DOI: 10.1137/1.9781611972788.22.

[143] J. R. Quinlan, "Induction of Decision Trees," *Machine Learning*, vol. 1, no. 1, pp. 81–106, 1986. DOI: 10.1023/A:1022643204877.

[144] J. Quinlan, "C4.5: Programs for Machine Learning," *Morgan Kaufmann Publishers, San Francisco, CA, United States*, vol. 16, pp. 235–240, 1993.

[145] L. Breiman, "Random forests," *Machine Learning*, vol. 45, no. 1, pp. 5–32, 2001. DOI: 10.1023/A:1010933404324.

[146] R. Mitchell, A. Adinets, T. Rao, and E. Frank, "Xgboost: Scalable gpu accelerated learning," 2018. arXiv: 1806.11248 [cs.LG].

[147] S. Haykin, "Neural Networks: A Comprehensive Foundation," *Neural Networks*, vol. 2, no. 2004, p. 41, 2004.

[148] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning internal representations by error propagation," in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition, Vol. 1: Foundations.* MIT Press, 1986, pp. 318–362. DOI: 10.1016/B978-1-4832-1446-7.50010-8.

[149] J. Schmidhuber, *Deep Learning in neural networks: An overview*, 2015. DOI: 10.1016/j.neunet.2014.09.003.

[150] T. Fawcett, "An introduction to ROC analysis," *Pattern Recognition Letters*, vol. 27, no. 8, pp. 861–874, Jun. 2006. DOI: 10.1016/j.patrec.2005.10.010.

[151] D. Chicco, "Ten quick tips for machine learning in computational biology," *BioData Mining*, vol. 10, no. 1, 2017. DOI: 10.1186/s13040-017-0155-3.

[152] J. Gorodkin, "Comparing two K-category assignments by a K-category correlation coefficient," *Computational Biology and Chemistry*, vol. 28, no. 5-6, pp. 367–374, 2004. DOI: 10.1016/j.compbiolchem.2004.09.006.

[153] F. Maroto-Molina, J. Navarro-García, K. Príncipe-Aguirre, I. Gómez-Maqueda, J. E. Guerrero-Ginel, A. Garrido-Varo, and D. C. Pérez-Marín, "A low-cost IOT-based system to monitor the location of a whole herd," *Sensors (Switzerland)*, vol. 19, no. 10, 2019. DOI: 10.3390/s19102298.

[154] D. Gobbett, R. Handcock, A. Zerger, C. Crossman, P. Valencia, T. Wark, and M. Davies, "Prototyping an Operational System with Multiple Sensors for Pasture Monitoring," *Journal of Sensor and Actuator Networks*, vol. 2, no. 3, pp. 388–408, 2013. DOI: 10.3390/jsan2030388.

[155] A. Kumar and G. P. Hancke, "A zigbee-based animal health monitoring system," *IEEE Sensors Journal*, vol. 15, no. 1, pp. 610–617, 2014. DOI: 10.1109/JSEN.2014.2349073.

[156] V. Giovanetti, M. Decandia, G. Molle, M. Acciaro, M. Mameli, A. Cabiddu, R. Cossu, M. G. Serra, C. Manca, S. P. Rassu, and C. Dimauro, "Automatic classification system for grazing, ruminating and resting behaviour of dairy sheep using a tri-axial accelerometer," *Livestock Science*, vol. 196, pp. 42–48, 2017. DOI: 10.1016/j.livsci.2016.12.011.

[157] A. Llaria, G. Terrasson, H. Arregui, and A. Hacala, "Geolocation and monitoring platform for extensive farming in mountain pastures," in *Proceedings of the IEEE International Conference on Industrial Technology*, 2015, pp. 2420–2425. DOI: 10.1109/ICIT.2015.7125454.

[158] Tnet, *Tnet Smart Agriculture*, Available online: https://www.tnet.it/en/iot-for-smart-agriculture. (visited on 06/01/2020).

[159] Agersens, *e-Shepherd*, Available online: https://www.agersens.com/automated-strip-grazing/. (visited on 11/27/2019).

[160] Digitanimal, *GPS animals tracker - Tracking and monitoring livestock*, Available online: https://digitanimal.com/?lang=en/. (visited on 11/27/2019).

[161] Nofence, *nofence.no – GPS-baserte virtuelle gjerder for geiter*, Available online: https://nofence.no/. (visited on 11/28/2018).

[162] S. Kjellqvist, "Determining cattle pasture utilization using GPS-collars," Tech. Rep., 2008, pp. 1–14. [Online]. Available: http://ex-epsilon.slu.se/archive/00002790/01/Determining_Cattle_Pasture_Utilization_Using_GPS-collars.pdf.

[163] L. W. Turner, M. C. Udal, B. T. Larson, and S. A. Shearer, "Monitoring cattle behavior and pasture use with GPS and GIS," *Canadian Journal of Animal Science*, vol. 80, no. 3, pp. 405–413, 2000. DOI: 10.4141/A99-093.

[164] J. L. Bowman, C. O. Kochanny, S. Demarais, and B. D. Leopold, "Evaluation of a GPS collar for white-tailed deer," *Wildlife Society Bulletin*, vol. 28, no. 1, pp. 141–145, 2000. DOI: CitedBy(since1996)46\rExportDate12June2012.

[165] R. Nathan, O. Spiegel, S. Fortmann-Roe, R. Harel, M. Wikelski, and W. M. Getz, "Using tri-axial acceleration data to identify behavioral modes of free-ranging animals: general concepts and tools illustrated for griffon vultures," *Journal of Experimental Biology*, vol. 215, no. 6, pp. 986–996, 2012. DOI: 10.1242/jeb.058602.

[166] J. Hunter, C. Brooking, W. Brimblecombe, R. G. Dwyer, H. A. Campbell, M. E. Watts, and C. E. Franklin, "OzTrack-e-infrastructure to support the management, analysis and sharing of animal tracking data," in *Proceedings of the IEEE 9th International Conference on e-Science, e-Science 2013*, 2013, pp. 140–147. DOI: 10.1109/eScience.2013.38.

[167] S. Rutter, N. Beresford, and G. Roberts, "Use of GPS to identify the grazing areas of hill sheep," *Computers and Electronics in Agriculture*, vol. 17, no. 2, pp. 177–188, 1997. DOI: 10.1016/S0168-1699(96)01303-8.

[168] E. S. Nadimi, H. T. Søgaard, T. Bak, and F. W. Oudshoorn, "ZigBee-based wireless sensor networks for monitoring animal presence and pasture time in a strip of new grass," *Computers and Electronics in Agriculture*, vol. 61, no. 2, pp. 79–87, 2008. DOI: 10.1016/j.compag.2007.09.010.

[169] J. Ruiz-Mirazo, G. J. Bishop-Hurley, and D. L. Swain, "Automated Animal Control: Can Discontinuous Monitoring and Aversive Stimulation Modify Cattle Grazing Behavior?" *Rangeland Ecology and Management*, vol. 64, no. 3, pp. 240–248, 2011. DOI: 10.2111/REM-D-10-00087.1.

[170] J. I. Huircán, C. Muñoz, H. Young, L. Von Dossow, J. Bustos, G. Vivallo, and M. Toneatti, "ZigBee-based wireless sensor network localization for cattle monitoring in grazing fields," *Computers and Electronics in Agriculture*, vol. 74, no. 2, pp. 258–264, 2010. DOI: 10.1016/j.compag.2010.08.014.

[171] B. Thorstensen, T. Syversen, T.-A. Bjornvold, and T. Walseth, "Electronic shepherd - a low-cost, low-bandwidth, wireless network system," in *Proceedings of the 2nd international Conference on Mobile systems, Applications, and Services*, ACM, 2004, pp. 245–255. DOI: http://doi.acm.org/10.1145/990064.990094.

[172] N. I. Dopico, B. Bejar, S. V. Macua, P. Belanovic, and S. Zazo, "Improved animal tracking algorithms using distributed kalman-based filters," in *Proceedings of the 17th European Wireless Conference 2011*, 2011, pp. 631–638.

[173] K. G. Liakos, P. Busato, D. Moshou, S. Pearson, and D. Bochtis, "Machine learning in agriculture: A review," vol. 18, no. 8, 2018. DOI: 10.3390/s18082674.

[174] Á. Ortiz-Pelaez and D. U. Pfeiffer, "Use of data mining techniques to investigate disease risk classification as a proxy for compromised Biosecurity of cattle herds in Wales," *BMC Veterinary Research*, vol. 4, pp. 1–16, 2008. DOI: 10.1186/1746-6148-4-24.

[175] L. Yin, T. Hong, and C. Liu, "Estrus detection in dairy cows from acceleration data using self-learning classification models," *Journal of Computers (Finland)*, vol. 8, no. 10, pp. 2590–2597, 2013. DOI: 10.4304/jcp.8.10.2590-2597.

[176] M. R. Borchers, Y. M. Chang, K. L. Proudfoot, B. A. Wadsworth, A. E. Stone, and J. M. Bewley, "Machine-learning-based calving prediction from activity, lying, and ruminating behaviors in dairy cattle," *Journal of Dairy Science*, vol. 100, no. 7, pp. 5664–5674, 2017. DOI: 10.3168/jds.2016-11526.

[177] G. Miller, M. Mitchell, Z. Barker, K. Giebel, E. Codling, J. Amory, C. Michie, C. Davison, C. Tachtatzis, I. Andonovic, and C.-A. Duthie, "Using animal-mounted sensor technology and machine learning to predict time-to-calving in beef and dairy cows," *Animal*, vol. 14, no. 6, pp. 1304–1312, 2020. DOI: 10.1017/S1751731119003380.

[178] M. L. Williams, N. Mac Parthaláin, P. Brewer, W. P. James, and M. T. Rose, "A novel behavioral model of the pasture-based dairy cow from GPS data using data mining and machine learning techniques," *Journal of Dairy Science*, vol. 99, no. 3, pp. 2063–2075, 2016. DOI: 10.3168/jds.2015-10254.

[179] M. A. Ladds, A. P. Thompson, J. P. Kadar, D. Slip, D. Hocking, and R. Harcourt, "Super machine learning: Improving accuracy and reducing variance of behaviour classification from accelerometry," *Animal Biotelemetry*, vol. 5, no. 1, 2017. DOI: 10.1186/s40317-017-0123-1.

[180] L. R. Brewster, J. J. Dale, T. L. Guttridge, S. H. Gruber, A. C. Hansell, M. Elliott, I. G. Cowx, N. M. Whitney, and A. C. Gleiss, "Development and application of a machine learning algorithm for classification of elasmobranch behaviour from accelerometry data," *Marine Biology*, vol. 165, no. 4, 2018. DOI: 10.1007/s00227-018-3318-y.

[181] S. Grünewälder, F. Broekhuis, D. W. Macdonald, A. M. Wilson, J. W. McNutt, J. Shawe-Taylor, and S. Hailes, "Movement Activity Based Classification of Animal Behaviour with an Application to Data from Cheetah (Acinonyx jubatus)," *PLoS ONE*, vol. 7, no. 11, 2012. DOI: 10.1371/journal.pone.0049120.

[182] Y. Wang, B. Nickel, M. Rutishauser, C. M. Bryce, T. M. Williams, G. Elkaim, and C. C. Wilmers, "Movement, resting, and attack behaviors of wild pumas are revealed by tri-axial accelerometer measurements," *Movement Ecology*, vol. 3, no. 1, 2015. DOI: 10.1186/s40462-015-0030-0.

[183] K. M. McLennan, E. A. Skillings, C. J. B. Rebelo, M. J. Corke, M. A. Pires Moreira, A. J. Morton, and F. Constantino-Casas, "Technical note: Validation of an automatic recording system to assess behavioural activity level in sheep (Ovis aries)," *Small Ruminant Research*, vol. 127, pp. 92–96, 2015. DOI: 10.1016/j.smallrumres.2015.04.002.

[184] M. Schwager, D. M. Anderson, Z. Butler, and D. Rus, "Robust classification of animal tracking data," *Computers and Electronics in Agriculture*, vol. 56, no. 1, pp. 46–59, 2007. DOI: 10.1016/j.compag.2007.01.002.

[185] R. Dutta, D. Smith, R. Rawnsley, G. Bishop-Hurley, J. Hills, G. Timms, and D. Henry, "Dynamic cattle behavioural classification using supervised ensemble classifiers," *Computers and Electronics in Agriculture*, vol. 111, no. 111, pp. 18–28, 2015. DOI: 10.1016/j.compag.2014.12.002.

[186] J. A. Vázquez Diosdado, Z. E. Barker, H. R. Hodges, J. R. Amory, D. P. Croft, N. J. Bell, and E. A. Codling, "Classification of behaviour in housed dairy cows using an accelerometer-based activity monitoring system," *Animal Biotelemetry*, vol. 3, no. 1, pp. 3–15, 2015. DOI: 10.1186/s40317-015-0045-8.

[187] A. Rahman, D. Smith, J. Hills, G. Bishop-Hurley, D. Henry, and R. Rawnsley, “A comparison of autoencoder and statistical features for cattle behaviour classification,” in *Proceedings of the International Joint Conference on Neural Networks*, vol. 2016-October, 2016, pp. 2954–2960. DOI: 10.1109/IJCNN.2016.7727573.

[188] A. Rahman, D. V. Smith, B. Little, A. B. Ingham, P. L. Greenwood, and G. J. Bishop-Hurley, “Cattle behaviour classification from collar, halter, and ear tag sensors,” *Information Processing in Agriculture*, vol. 5, no. 1, pp. 124–133, 2018. DOI: 10.1016/j.inpa.2017.10.001.

[189] K. Sakai, K. Oishi, M. Miwa, H. Kumagai, and H. Hirooka, “Behavior classification of goats using 9-axis multi sensors: The effect of imbalanced datasets on classification performance,” *Computers and Electronics in Agriculture*, vol. 166, p. 105 027, 2019. DOI: 10.1016/j.compag.2019.105027.

[190] D. Gutierrez-Galan, J. P. Dominguez-Morales, E. Cerezuela-Escudero, A. Rios-Navarro, R. Tapiador-Morales, M. Rivas-Perez, M. Dominguez-Morales, A. Jimenez-Fernandez, and A. Linares-Barranco, “Embedded neural network for real-time animal behavior classification,” *Neurocomputing*, vol. 272, pp. 17–26, 2018. DOI: 10.1016/j.neucom.2017.03.090.

[191] J. Marais, S. Le Roux, R. Wolhuter, and T. Niesler, “Automatic classification of sheep behaviour using 3-axis accelerometer data,” in *Proceedings of Pattern Recognition Association of South Africa and AfLaT International Joint Symposium*, Cape Town, RSA, 2014, pp. 1–6.

[192] F. A. Alvarenga, I. Borges, L. Palkovič, J. Rodina, V. H. Oddy, and R. C. Dobos, “Using a three-axis accelerometer to identify and classify sheep behaviour at pasture,” *Applied Animal Behaviour Science*, vol. 181, pp. 91–99, 2016. DOI: 10.1016/j.applanim.2016.05.026.

[193] M. Decandia, V. Giovanetti, G. Molle, M. Acciaro, M. Mameli, A. Cabiddu, R. Cossu, M. G. Serra, C. Manca, S. P. Rassu, and C. Dimauro, “The effect of different time epoch settings on the classification of sheep behaviour using tri-axial accelerometry,” *Computers and Electronics in Agriculture*, vol. 154, pp. 112–119, 2018. DOI: 10.1016/j.compag.2018.09.002.

[194] J. Barwick, D. W. Lamb, R. Dobos, M. Welch, and M. Trotter, “Categorising sheep activity using a tri-axial accelerometer,” *Computers and Electronics in Agriculture*, vol. 145, pp. 289–297, 2018. DOI: 10.1016/j.compag.2018.01.007.

[195] E. Walton, C. Casey, J. Mitsch, J. A. Vázquez-Diosdado, J. Yan, T. Dottorini, K. A. Ellis, A. Winterlich, and J. Kaler, “Evaluation of sampling frequency, window size and sensor position for classification of sheep behaviour,” *Royal Society Open Science*, vol. 5, no. 2, 2018. DOI: 10.1098/rsos.171442.

[196] N. Mansbridge, J. Mitsch, N. Bollard, K. Ellis, G. G. Miguel-Pacheco, T. Dottorini, and J. Kaler, “Feature selection and comparison of machine learning algorithms in classification of grazing and rumination behaviour in sheep,” *Sensors (Switzerland)*, vol. 18, no. 10, p. 3532, 2018. DOI: 10.3390/s18103532.

[197] GEA, *Activity Detection CowScout*, Available online: https://www.gea.com/en/products/activity-detection-cowscout.jsp. (visited on 11/27/2019).

[198] *Cowlar - The smart collar for cows*, Available online: https://cowlar.com/. (visited on 11/27/2019).

[199] IceRobotics, *CowAlert*, Available online: https://www.icerobotics.com/cowalert/. (visited on 11/27/2019).

[200] CowManager, *CowManager Sensor*, Available online: https://www.cowmanager.com/en-us/. (visited on 11/27/2019).

[201] S. Dairy, *Dairy Cow Monitoring and Herd Management Solutions, Precision Dairy Farm Technology*, Available online: https://www.scrdairy.com/. (visited on 11/27/2019).

[202] Dairymaster, *Moomonitor +*, Available online: https://www.dairymaster.com/products/moomonitor/. (visited on 11/27/2019).

[203] D. M. Anderson, "Virtual fencing past, present and future," in *Rangeland Journal*, vol. 29, 2007, pp. 65–78.

[204] S. C. Martin, H. D. McCallum, and F. T. McCallum, *The Wire That Fenced the West*, University of Oklahoma Press, Ed. 1965. DOI: 10.2307/3895504.

[205] C. Umstatter, "The evolution of virtual fences: A review," *Computers and Electronics in Agriculture*, vol. 75, no. 1, pp. 10–22, 2011. DOI: 10.1016/j.compag.2010.10.005.

[206] D. M. Anderson, K. M. Havstad, W. L. Shupe, R. Libeau, J. N. Smith, and L. W. Murray, "Benefits and costs in controlling sheep bonded to cattle without wire fencing," *Small Ruminant Research*, vol. 14, no. 1, pp. 1–8, 1994. DOI: 10.1016/0921-4488(94)90002-7.

[207] S. B. Markus, D. W. Bailey, and D. Jensen, "Comparison of electric fence and a simulated fenceless control system on cattle movements," *Livestock Science*, vol. 170, pp. 203–209, 2014. DOI: 10.1016/j.livsci.2014.10.011.

[208] *Official Site for Invisible Fence Brand Dog Fences - The Invisible Fence® Brand*, Available online: https://www.invisiblefence.com/. (visited on 04/07/2017).

[209] P. K. Fay, V. T. McElligott, and K. M. Havstad, "Containment of free-ranging goats using pulsed-radio-wave-activated shock collars," *Applied Animal Behaviour Science*, vol. 23, no. 1-2, pp. 165–171, 1989. DOI: 10.1016/0168-1591(89)90016-6.

[210] M. Monod, P. Faure, L. Moiroux, and P. Rameau, "Stakeless fencing for mountain pastures," *Journal of Farm Management*, vol. 13, no. 10, pp. 23–30, 2009.

[211] A. R. Rodgers and E. J. G. Lawson, "Field trials of the Lotek GPS collar on moose," in *The Wildlife Society and Biological Resources Division*, 1997, p. 58.

[212] D. M. Anderson and L. W. Murray, "Sheep laterality," *Laterality, Asymmetries of Brain, Behaviour, and Cognition*, vol. 18, no. 2, pp. 179–193, Mar. 2013. DOI: 10.1080/1357650X.2011.647919.

[213] A. Mainwaring, D. Culler, J. Polastre, R. Szewczyk, and J. Anderson, "Wireless Sensor Networks for Habitat Monitoring," in *Proceedings of the 1st ACM International Workshop on Wireless Sensor Networks and Applications*, ACM, 2002, pp. 88–97. DOI: 10.1145/570738.570751.

[214] P. Brose, *US Patent 4,898,120 - Animal training and restraining system*, Available online: https://www.google.com/patents/US4898120, 1990.

[215] H. Aine, *US Patent 5,121,711 - Wireless control of animals*, Available online: https://www.google.com/patents/US5121711, 1992.

[216] L. D. Howery, D. W. Bailey, G. B. Ruyle, and W. J. Renken, "Cattle use visual cues to track food locations," *Applied Animal Behaviour Science*, vol. 67, no. 1-2, pp. 1–14, 2000. DOI: 10.1016/S0168-1591(99)00118-5.

[217] A. F. Cibils, L. D. Howery, and G. B. Ruyle, "Diet and habitat selection by cattle: The relationship between skin- and gut-defense systems," *Applied Animal Behaviour Science*, vol. 88, no. 3-4, pp. 187–208, 2004. DOI: 10.1016/j.applanim.2004.04.001.

[218] C. Umstatter, S. Brocklehurst, D. W. Ross, and M. J. Haskell, "Can the location of cattle be managed using broadcast audio cues?" *Applied Animal Behaviour Science*, vol. 147, no. 1-2, pp. 34–42, 2013. DOI: j.applanim.2013.04.019.

[219] D. Anderson, *US Patent 7,753,007 - Ear-A-Round equipment platform for animals*, Available online: https://www.google.com/patents/US7753007, 2010.

[220] Z. Butler, P. Corke, R. Peterson, and D. Rus, "Virtual fences for controlling cows," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2004, pp. 4429–4436. DOI: 10.1109/robot.2004.1302415.

[221] Z. Butler, P. Corke, R. Peterson, and D. Rus, "From robots to animals: Virtual fences for controlling cattle," in *International Journal of Robotics Research*, vol. 25, 2006, pp. 485–508. DOI: 10.1177/0278364906065375.

[222] G. J. Bishop-Hurley, D. L. Swain, D. M. Anderson, P. Sikka, C. Crossman, and P. Corke, "Virtual fencing applications: Implementing and testing an automated cattle control system," *Computers and Electronics in Agriculture*, vol. 56, no. 1, pp. 14–22, 2007. DOI: 10.1016/j.compag.2006.12.003.

[223] M. Jouven, H. Leroy, A. Ickowicz, and P. Lapeyronie, "Can virtual fences be used to control grazing sheep?" *Rangeland Journal*, vol. 34, no. 1, pp. 111–123, 2012. DOI: 10.1071/RJ11044.

[224] E. I. Brunberg, K. E. Bøe, and K. M. Sørheim, "Testing a new virtual fencing system on sheep," *Acta Agriculturae Scandinavica A: Animal Sciences*, vol. 65, no. 3-4, pp. 168–175, 2015. DOI: 10.1080/09064702.2015.1128478.

[225] E. I. Brunberg, I. K. Bergslid, K. E. Bøe, and K. M. Sørheim, "The ability of ewes with lambs to learn a virtual fencing system," vol. 11, no. 11, pp. 2045–2050, 2017. DOI: 10.1017/S1751731117000891.

[226] D. Marini, R. Llewellyn, S. Belson, and C. Lee, "Controlling within-field sheep movement using virtual fencing," *Animals*, vol. 8, no. 3, 2018. DOI: 10.3390/ani8030031.

[227] D. Marini, M. D. Meuleman, S. Belson, T. B. Rodenburg, R. Llewellyn, and C. Lee, "Developing an ethically acceptable virtual fencing system for sheep," *Animals*, vol. 8, no. 3, 2018. DOI: 10.3390/ani8030033.

[228] C. Lee, J. M. Henshall, T. J. Wark, C. C. Crossman, M. T. Reed, H. G. Brewer, J. O'Grady, and A. D. Fisher, "Associative learning by cattle to enable effective and ethical virtual fences," *Applied Animal Behaviour Science*, vol. 119, no. 1-2, pp. 15–22, 2009. DOI: 10.1016/j.applanim.2009.03.010.

[229] E. Schalke, J. Stichnoth, S. Ott, and R. Jones-Baade, "Clinical signs caused by the use of electric training collars on dogs in everyday life situations," *Applied Animal Behaviour Science*, vol. 105, no. 4, pp. 369–380, 2007. DOI: 10.1016/j.applanim.2006.11.002.

[230] C. Lee, A. D. Fisher, M. T. Reed, and J. M. Henshall, "The effect of low energy electric shock on cortisol, β-endorphin, heart rate and behaviour of cattle," *Applied Animal Behaviour Science*, vol. 113, no. 1-3, pp. 32–42, 2008. DOI: 10.1016/j.applanim.2007.10.002.

[231] C. Lee, K. Prayaga, M. Reed, and J. Henshall, "Methods of training cattle to avoid a location using electrical cues," *Applied Animal Behaviour Science*, vol. 108, no. 3-4, pp. 229–238, 2007. DOI: 10.1016/j.applanim.2006.12.003.

[232] ECMA, "Technical Requirements for Electronic Pet Training and Containment Collars," Tech. Rep., 2008. [Online]. Available: http://ecma.eu.com/wp-content/uploads/2016/10/120411amc-ECMA-Technical-Requirement-6-0-FINAL-APPROVED.pdf.

[233] E. Nadimi, R. Jørgensen, V. Blanes-Vidal, and S. Christensen, "Monitoring and classifying animal behavior using ZigBee-based mobile adhoc Wireless Sensor Networks and Artificial Neural Networks," *Computers and Electronics in Agriculture*, vol. 82, pp. 44–54, 2012. DOI: 10.1016/j.compag.2011.12.008.

[234] DAFM, *National Sheep and Goat Census 2019*, Available online: https://www.agriculture.gov.ie/media/migration/animalhealthwelfare/animalidentificationandmovement/nationalsheepidentificationsystem/NationalSheepandGoatCensuS19050520.pdf. (visited on 10/13/2020).

[235] D. J. Undersander, B. Albert, D. Cosgrove, D. Johnson, and P. Peterson, “Pastures for Profit: A Guide to Rotational Grazing (A3529).,” Tech. Rep., 2002, pp. 1–38. [Online]. Available: https://onpasture.com/wp-content/uploads/2020/04/Pastures-for-profit-A-guide-to-rotational-grazing.pdf.

[236] St, “Using LSM303DLH for a tilt comensated electronic compass,” Tech. Rep. August 2010, 2010, pp. 1–34. [Online]. Available: https://www.pololu.com/file/0J434/LSM303DLH-compass-app-note.pdf.

[237] J. Yan, M. Zhou, and Z. Ding, “Recent Advances in Energy-Efficient Routing Protocols for Wireless Sensor Networks: A Review,” vol. 4, pp. 5673–5686, 2016. DOI: 10.1109/ACCESS.2016.2598719.

[238] H. Kopetz, “Time-triggered model of computation,” in *Proceedings of the 19th Real-Time Systems Symposium*, 1998, pp. 168–177. DOI: 10.1109/REAL.1998.739743.

[239] H. Suo, J. Wan, C. Zou, and J. Liu, “Security in the internet of things: A review,” in *Proceedings - 2012 International Conference on Computer Science and Electronics Engineering, ICCSEE 2012*, vol. 3, 2012, pp. 648–651, ISBN: 9780769546476. DOI: 10.1109/ICCSEE.2012.373.

[240] M. Rostanski, K. Grochla, and A. Seman, “Evaluation of highly available and fault-tolerant middleware clustered architectures using RabbitMQ,” in *Proceedings of Federated Conference on Computer Science and Information Systems*, IEEE, 2014, pp. 879–884. DOI: 10.15439/2014F48.

[241] M. Zaharia, R. S. Xin, P. Wendell, T. Das, M. Armbrust, A. Dave, X. Meng, J. Rosen, S. Venkataraman, and M. J. Franklin, “Apache spark: a unified engine for big data processing,” *Communications of the ACM*, vol. 59, no. 11, pp. 56–65, 2016. DOI: 10.1145/2934664.

[242] M. Proctor, “Drools: A Rule Engine for Complex Event Processing,” in *Proceedings of the 4th international conference on Applications of Graph Transformations with Industrial Relevance*, Springer-Verlag, 2012, pp. 2–2. DOI: 10.1007/978-3-642-34176-2_2.

[243] Commission of the European Communities, “Report from the Commission to the Council on the implementation of electronic identification in sheep and goats,” Tech. Rep., 2007. [Online]. Available: https://eur-lex.europa.eu/legal-content/EN/ALL/?uri=CELEX%3A52007DC0711.

[244] *FDX-B Animal Identification protocol description*, Available online: http://www.priority1design.com.au/fdx-b_animal_identification_protocol.html. (visited on 11/09/2020).

[245] *iFarmtec*, Available online: http://ifarmtec.pt/. (visited on 10/18/2019).

[246] *Open Camera*, Available online: http://opencamera.org.uk/. (visited on 10/18/2019).

[247] *CRAN - Package FSelector*, Available online: https://cran.r-project.org/web/packages/FSelector/index.html. (visited on 10/18/2019).

[248] Varsity Tutors, *How to find the area of a hexagon - High School Math*, Available online: https://www.varsitytutors.com/high_school_math-help/how-to-find-the-area-of-a-hexagon. (visited on 04/03/2021).

[249] M. Portoles-Comeras, C. Ibars, J. Núñez-Martínez, and J. Mangues-Bafalluy, “Characterizing WLAN medium utilization for radio environment maps,” in *Proceedings of the IEEE Vehicular Technology Conference*, 2011, pp. 1–5. DOI: 10.1109/VETECF.2011.6093284.

[250] Microchip, *PIC32MX1XX/2XX 28/36/44-PIN FAMILY Device Pins Program Memory (KB) (1) Data Memory (KB) Remappable Peripherals*, Available online: http://ww1.microchip.com/downloads/en/DeviceDoc/PIC32MX1XX2XX-28-36-44-PIN-DS60001168K.pdf. (visited on 01/23/2021).

[251] Hope Microelectronics Ltd, *Rfm22B/23B ISM TRANSCEIVER MODULE*, Available online: `http://www.hoperf.comhttp://docs-europe.electrocomponents.com/webdocs/0f44/0900766b80f440d6.pdf`. (visited on 01/23/2021).

[252] ST, *LSM303DLHC Ultra-compact high-performance eCompass module: 3D accelerometer and 3D magnetometer Datasheet-production data Features*, Available online: `https://www.st.com/resource/en/datasheet/lsm303c.pdf`. (visited on 01/23/2021).

[253] MulticompPro, *Electromechanical Audio Transducer*, Available online: `http://www.farnell.com/datasheets/2863864.pdf`. (visited on 01/23/2021).

[254] Ekulit, *Assembled ultrasonic sensor K-14WP10*, Available online: `https://www.ekulit.com/wp-content/uploads/datasheets/700670K-14WP10.pdf`. (visited on 01/23/2021).

[255] *RStudio*, Available online: `https://rstudio.com/`. (visited on 10/18/2019).

APPENDIX A

Appendix A: Numerical and Statistical Formulas Commonly Used in Feature Transformation

Minimum and maximum: considering a set of elements A, the minimum (min) correspond to the smallest value within the OW while the maximum correspond to the maximum (max) value within the OW. If A is sorted in a ascending order, $min(\mathbf{A})$ corresponds to the first value and $max(\mathbf{A})$ corresponds to the last value.

Mean: corresponds to the arithmetic mean, also denominated of *average*. Being N the total number of elements in a set $\mathbf{A}$, the mean ($\overline{A}$) is given by:

$$\overline{A} = \frac{1}{N} \sum_{i=1}^{N} A_i \tag{A.1}$$

Variance: the unbiased sample variance[1] s_{N-1}^2 is given by:

$$s_{N-1}^2 = \frac{1}{N-1} \sum_{i=1}^{N} \left(A_i - \overline{A}\right)^2 \tag{A.2}$$

Standard Deviation: the unbiased standard deviation s_{N-1} is given as the square root of the unbiased sample variance s_{N-1}^2:

$$s_{N-1} = \sqrt{\frac{1}{N-1} \sum_{i=1}^{N} \left(A_i - \overline{A}\right)^2} \tag{A.3}$$

Root Mean Square: the Root Mean Squarte (RMS) or quadratic mean is given as the square root of the mean square, *i.e.*;

$$A_{RMS} = \sqrt{\frac{\sum_{i=1}^{N} A_i^2}{N}} \tag{A.4}$$

[1]The sample variance s_N^2 is as biased estimator of the sample variance. In this thesis we consider the unbiased sample variance as preferred.

Skewness: typically, software programs that have this function compute the Fisher-Pearson coefficient of skewness. It is an unbiased version of the Fisher-Pearson coefficient of skewness and is given by:

$$G_1 = \frac{\sqrt{N(N-1)}}{N-2} \frac{\sum_{i=1}^{N} \left(A_i - \overline{A}\right)^3 / N}{s^3} = \frac{\sqrt{N(N-1)}}{N-2} \frac{\frac{1}{N-1} \sum_{i=1}^{N} \left(A_i - \overline{A}\right)^3}{\left(\sqrt{\frac{1}{N} \sum_{i=1}^{N} \left(A_i - \overline{A}\right)^2}\right)^3} \tag{A.5}$$

Kurtosis: the unbiased kurtosis, also called kurtosis excess, is given by:

$$\begin{aligned} b_2 &= \frac{N(N+1)}{(N-1)(N-2)(N-3)} \frac{\sum_{i=1}^{N} \left(A_i - \overline{A}\right)^4}{s^4} - \frac{3(N-1)^2}{(N-2)(N-3)} = \\ &= \frac{N(N+1)}{(N-1)(N-2)(N-3)} \frac{\sum_{i=1}^{N} \left(A_i - \overline{A}\right)^4}{\left(\sqrt{\frac{1}{N-1} \sum_{i=1}^{N} \left(A_i - \overline{A}\right)^2}\right)^4} - \frac{3(N-1)^2}{(N-2)(N-3)} \end{aligned} \tag{A.6}$$

Interquartile Range: the Interquartile Range (IQR) is a statistical measure that allows to infer the dispersion of values. It resorts on the statistical median, and considering $Q1$ and $Q3$ the low and high statistical medians, *i.e.* the 75th percentile and 25th percentile, IQR is given by:

$$IQR = Q3 - Q1 \tag{A.7}$$

Zero Crossing Rate: the Zero Crossing Rate (ZCR) is commonly used in voice detection systems and it measures the signal's rate of change (from positive to negative and vice and versa).

Mean Crossing Rate: the MCR is similar to the ZCR but instead of measuring the crosses through zero of the original signal, it considers the difference between the values within the observation and the mean within the same observation.

Energy: the measurement of the energy is typically applied in signal processing techniques and it is calculated as the squared root of the signal, being commonly normalized by the length of the OW. Hence, the Energy (E) is given by:

$$E = \frac{1}{N} \sum_{i=1}^{N} A_i^2 \tag{A.8}$$

Signal Magnitude Area: the SMA comprises the sum of the magnitude of all the components of a signal. Commonly, SMA is normalized by the length of the OW. Hence, SMA is given by:

$$SMA = \frac{1}{N} \sum_{i=1}^{N} |A_i| \tag{A.9}$$

If, for instance, A_i is composed of three different components, as it happens with the data provided by an accelerometer, the SMA is given as follows:

$$SMA = \frac{1}{N} \left(\sum_{i=1}^{N} |A_x(i)| + |A_y(i)| + |A_z(i)| \right) \tag{A.10}$$

being, A_x,A_y and A_z the three axial components.

Signal Magnitude Vector: it is very similar to the SMA but it is specially used in applications that use accelerometer data. Thus, if A is a vector composed of A_x,A_y and A_z components, the Signal Magnitude Vector (SMV) is given by:

$$SMA = \frac{1}{N}\sum_{i=1}^{N} ||A(i)|| = \frac{1}{N}\sum_{i=1}^{N} \sqrt{A_x(i)^2 + A_y(i)^2 + A_z(i)^2} \tag{A.11}$$

Movement Variation: it measures the cumulative variations within all the components of an accelerometer signal within an observation window, *i.e.*:

$$MV = \frac{1}{N}\left(\sum_{i=1}^{N} |A_x(i+1) - A_x(i)| + |A_y(i+1) - A_y(i)| + |A_z(i+1) - A_z(i)|\right) \tag{A.12}$$

Spectral Entropy: The Spectral Entropy (SE) is based on the Shannon entropy to evaluate the power distribution in the frequency domain. The signal considers the normalized power distribution in the frequency domain as a probability distribution before applying the Shannon entropy formula. In sum, SE is given by:

$$SE = -\sum_{1}^{N} P(i)log_2P(i) \tag{A.13}$$

where $P(i)$ is the normalized power spectral density, *i.e.*:

$$P(i) = \frac{S(i)}{\sum_i S(i)}, \tag{A.14}$$

where $S(i) = |X(m)|^2$ and $X(m)$ is the Discrete Fourier Transform (DFT) of the signal.

Dominant Frequency: Applying DFT it is possible to extract important frequency-domain features. One of them is the DF that represents the frequency with highest energy. The DFT is give by:

$$F_n = \sum_{n=0}^{N-1} f_k e^{-2\pi ink/N} \tag{A.15}$$

APPENDIX B

Appendix B: μC minimum duration

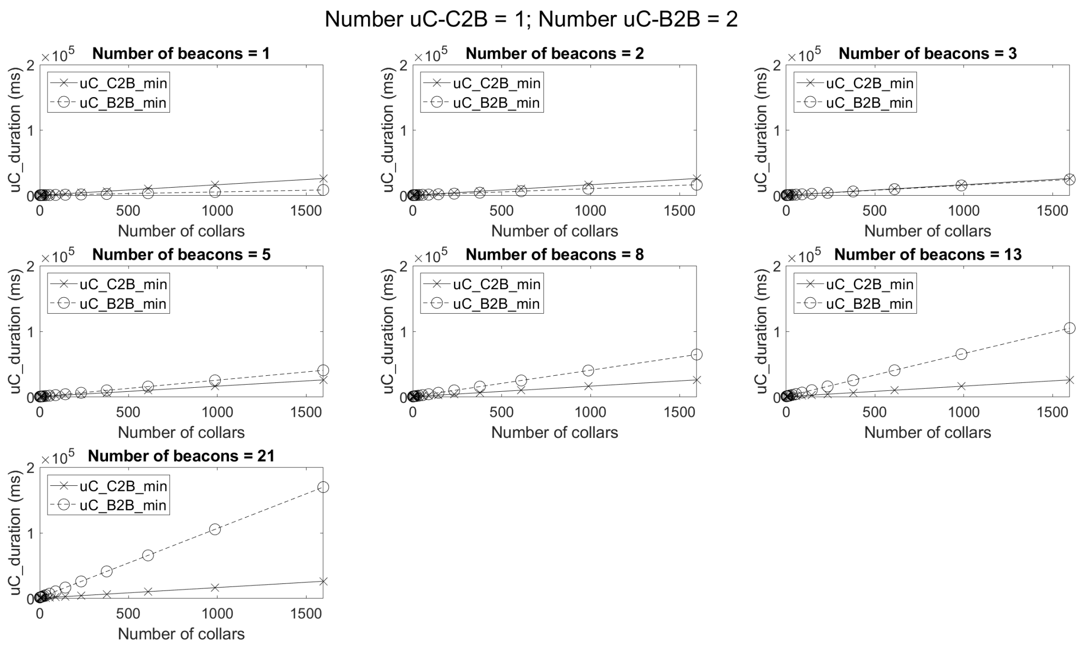


Figure B.1: μC minimum duration for the combination $N_{\mu C_{C2B}} = 1$; $N_{\mu C_{B2B}} = 2$.

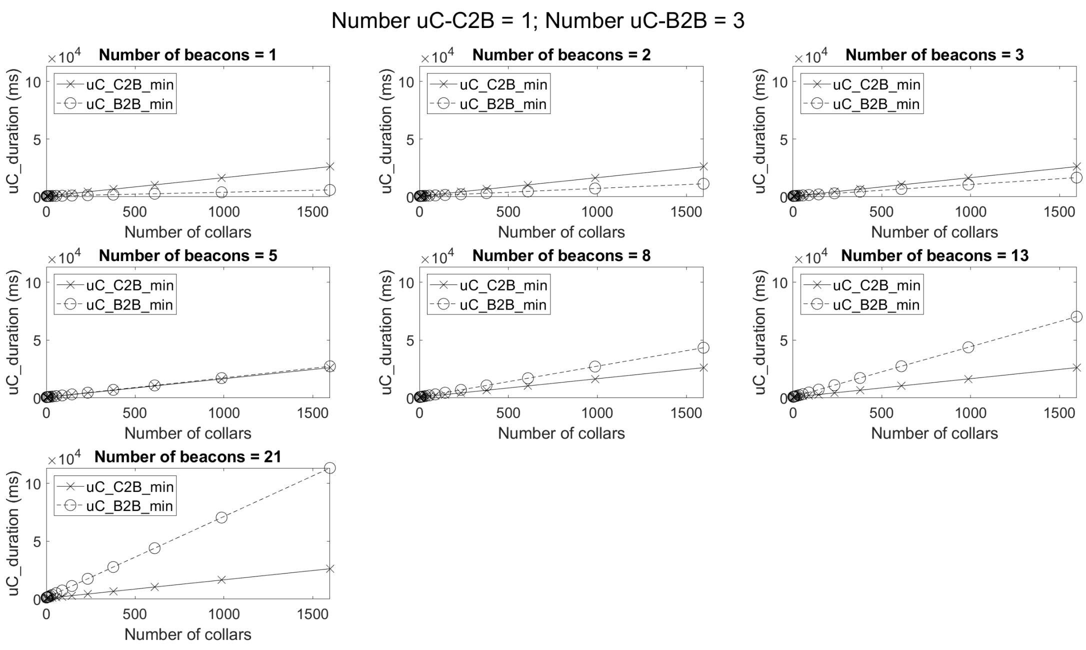


Figure B.2: μC minimum duration for the combination $N_{\mu C_{C2B}} = 1$; $N_{\mu C_{B2B}} = 3$.

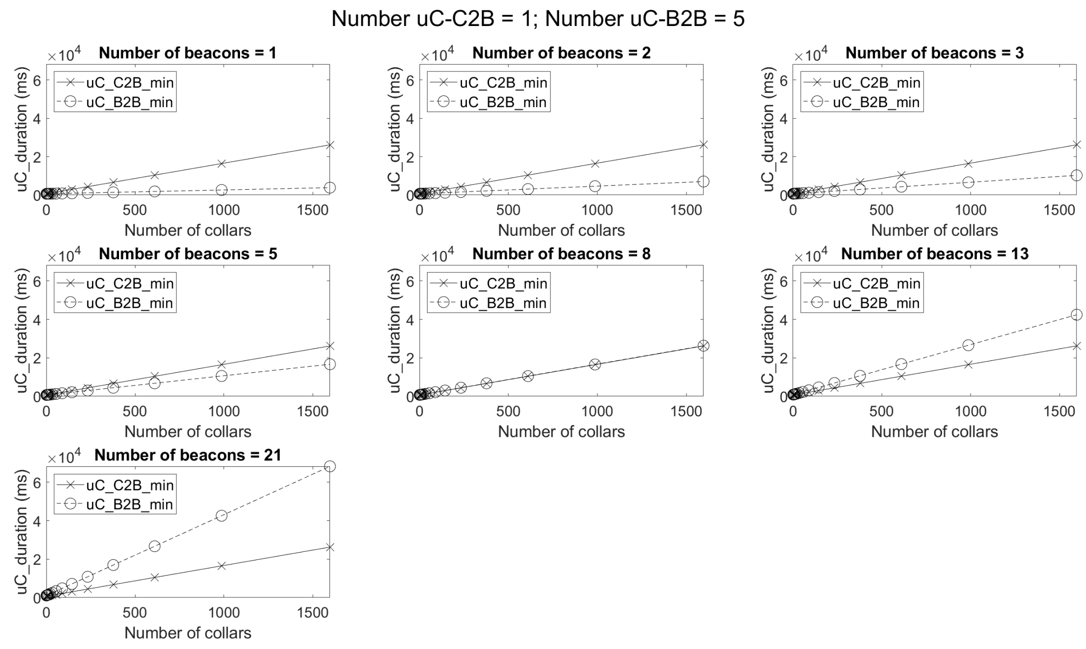


Figure B.3: μC minimum duration for the combination $N_{\mu C_{C2B}} = 1$; $N_{\mu C_{B2B}} = 5$.

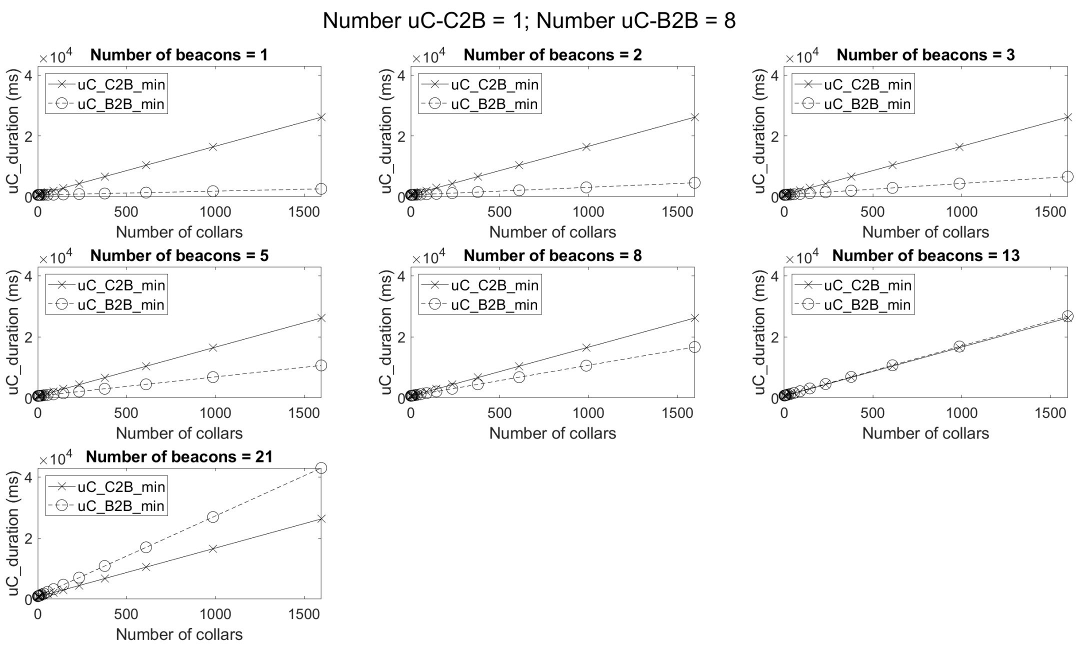


Figure B.4: μC minimum duration for the combination $N_{\mu C_{C2B}} = 1$; $N_{\mu C_{B2B}} = 8$.

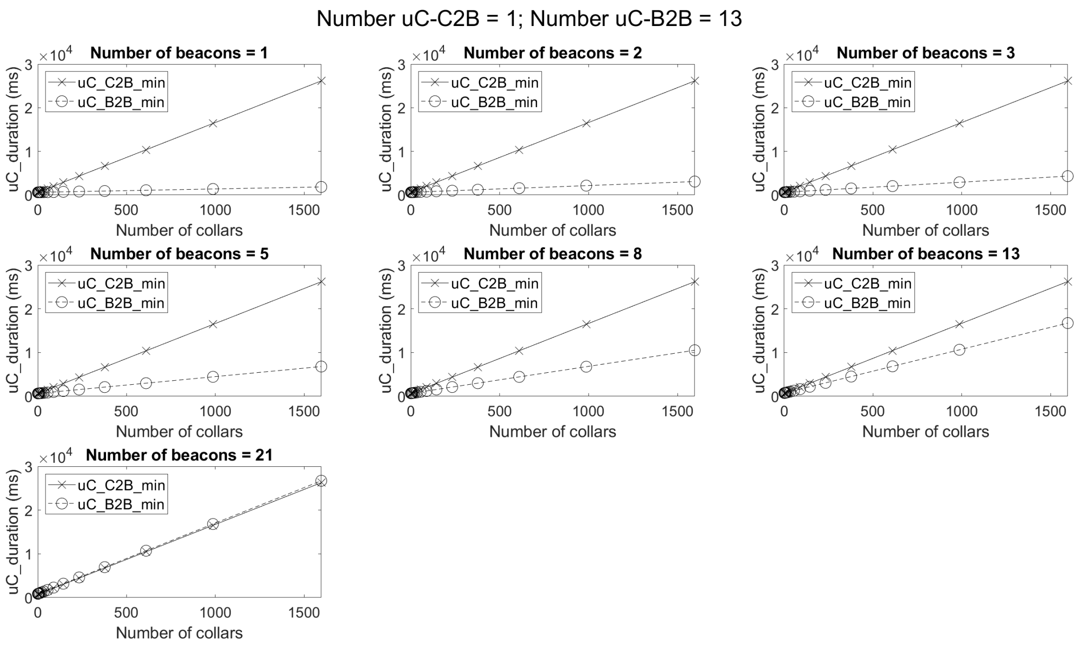


Figure B.5: μC minimum duration for the combination $N_{\mu C_{C2B}} = 1$; $N_{\mu C_{B2B}} = 13$.

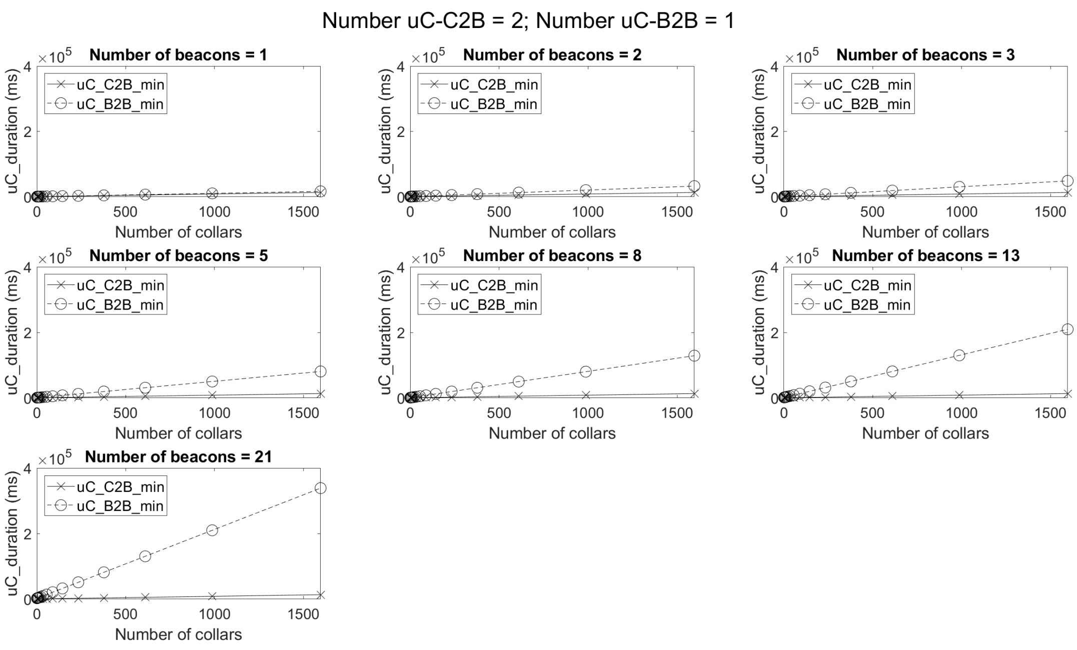


Figure B.6: μC minimum duration for the combination $N_{\mu C_{C2B}} = 2$; $N_{\mu C_{B2B}} = 1$.

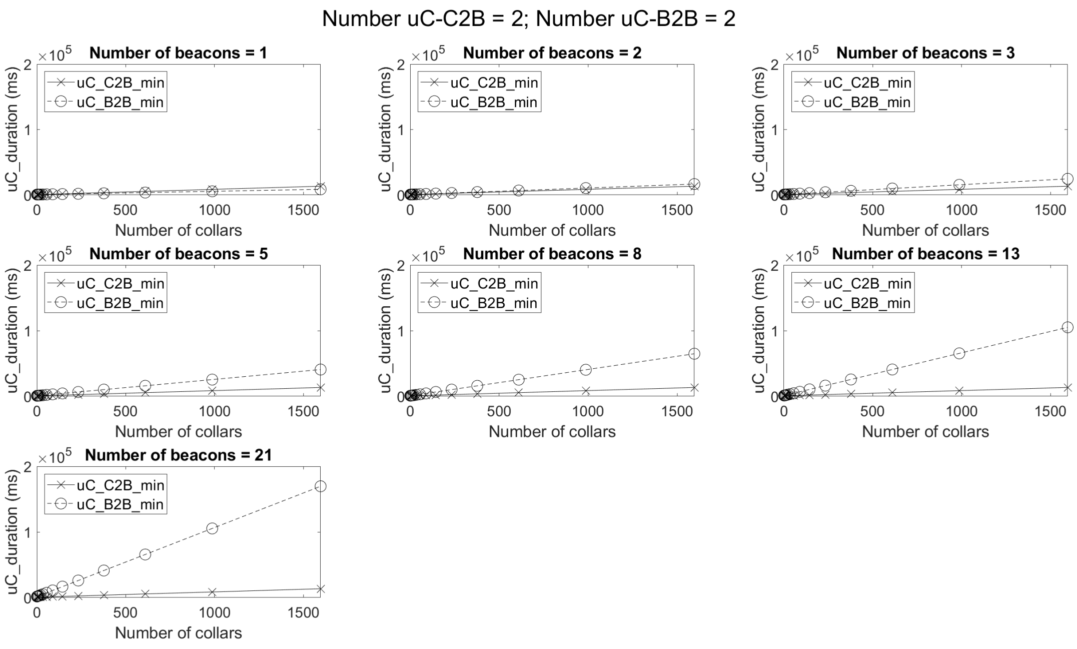


Figure B.7: μC minimum duration for the combination $N_{\mu C_{C2B}} = 2$; $N_{\mu C_{B2B}} = 2$.

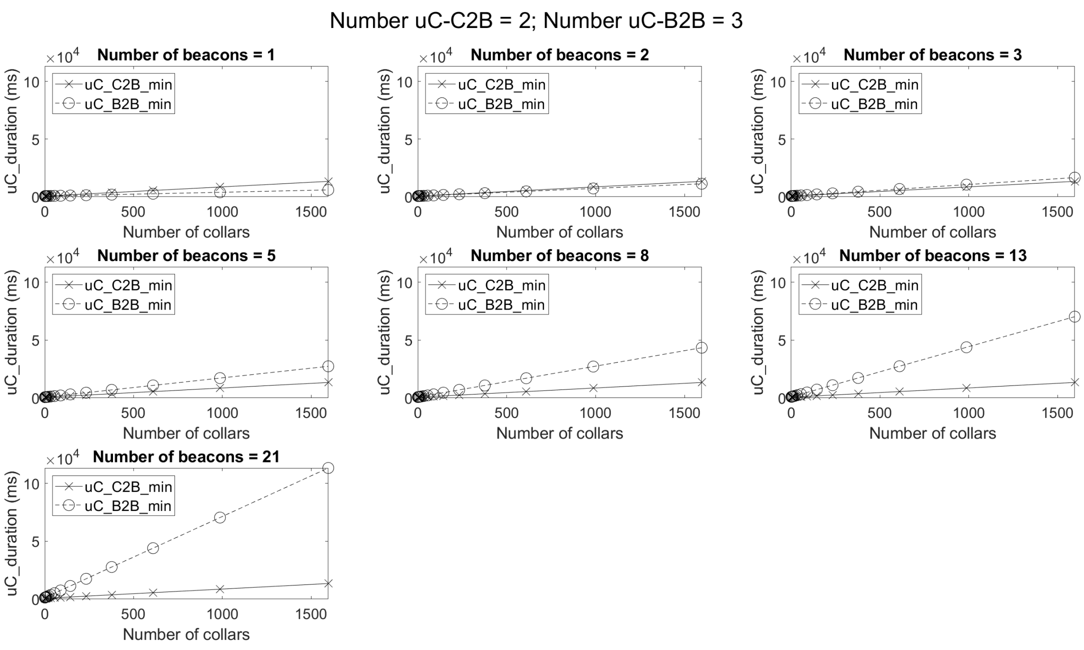


Figure B.8: μC minimum duration for the combination $N_{\mu C_{C2B}} = 2$; $N_{\mu C_{B2B}} = 3$.

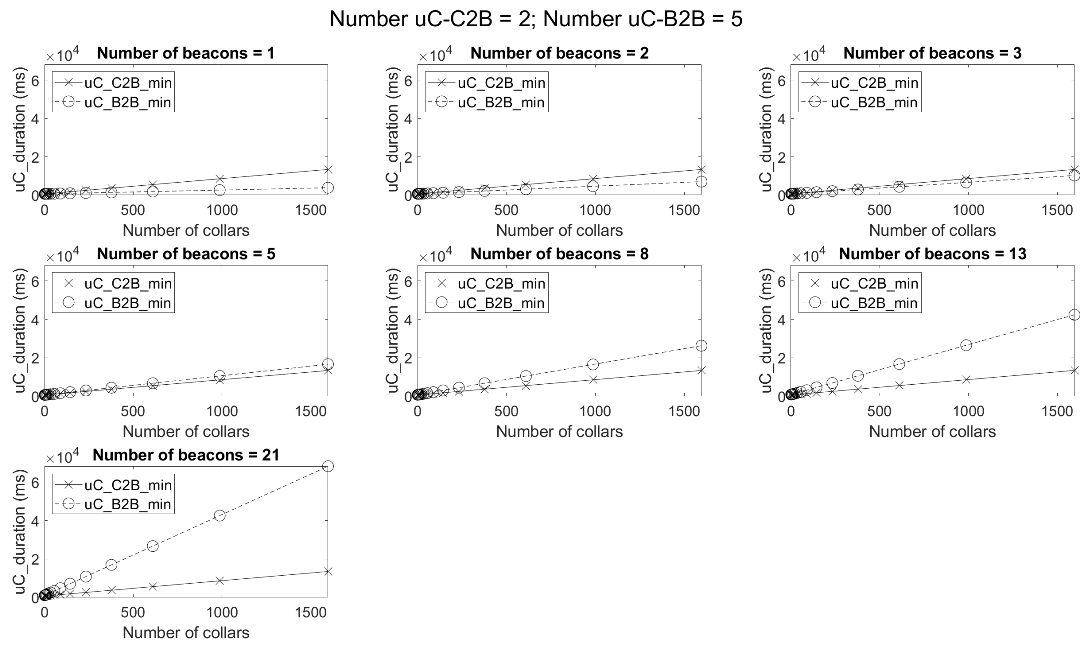


Figure B.9: μC minimum duration for the combination $N_{\mu C_{C2B}} = 2$; $N_{\mu C_{B2B}} = 5$.

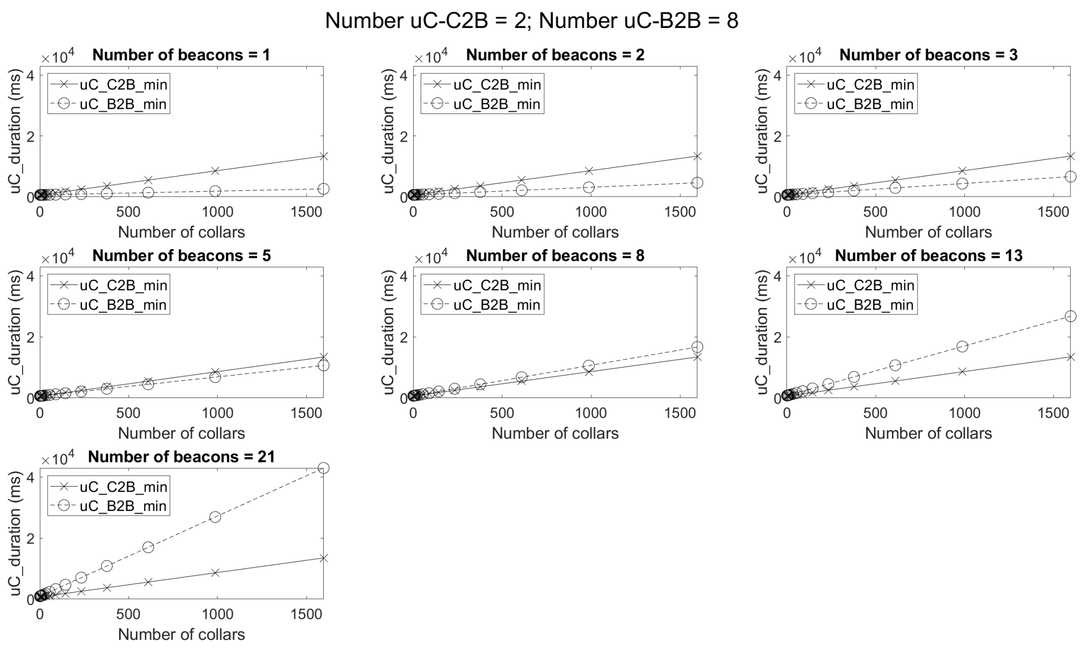


Figure B.10: μC minimum duration for the combination $N_{\mu C_{C2B}} = 2$; $N_{\mu C_{B2B}} = 8$.

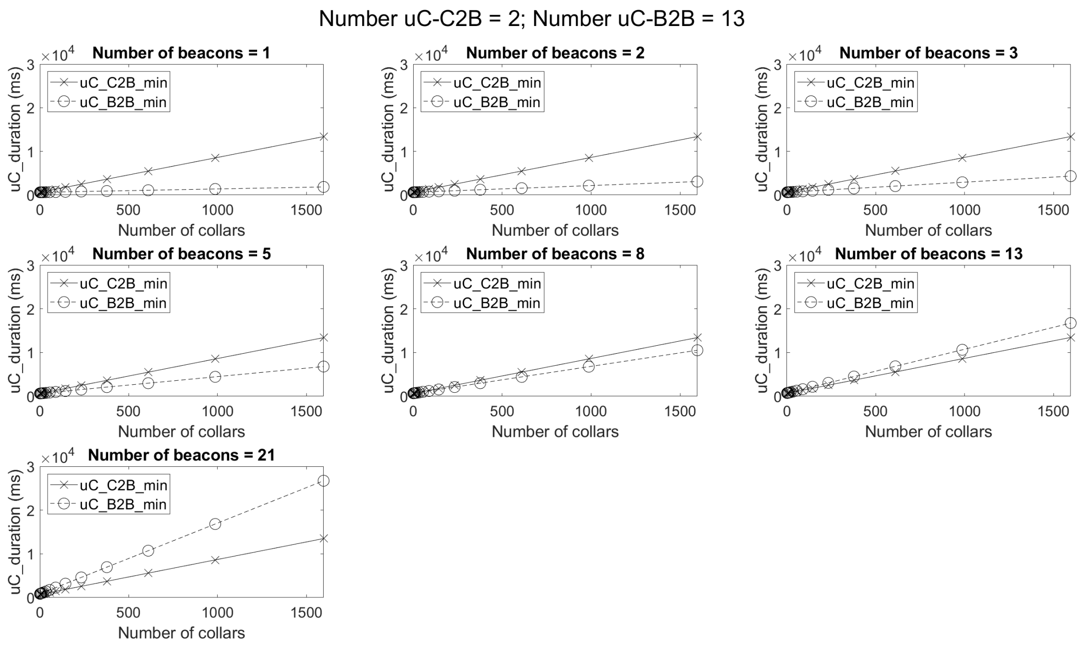


Figure B.11: μC minimum duration for the combination $N_{\mu C_{C2B}} = 2$; $N_{\mu C_{B2B}} = 13$.

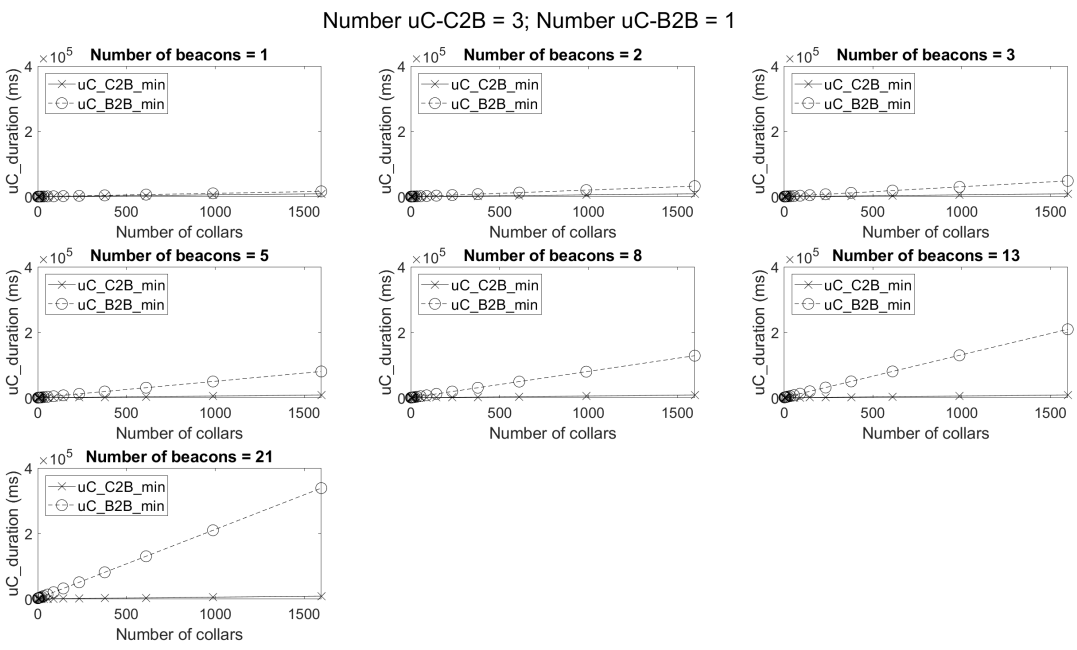


Figure B.12: μC minimum duration for the combination $N_{\mu C_{C2B}} = 3$; $N_{\mu C_{B2B}} = 1$.

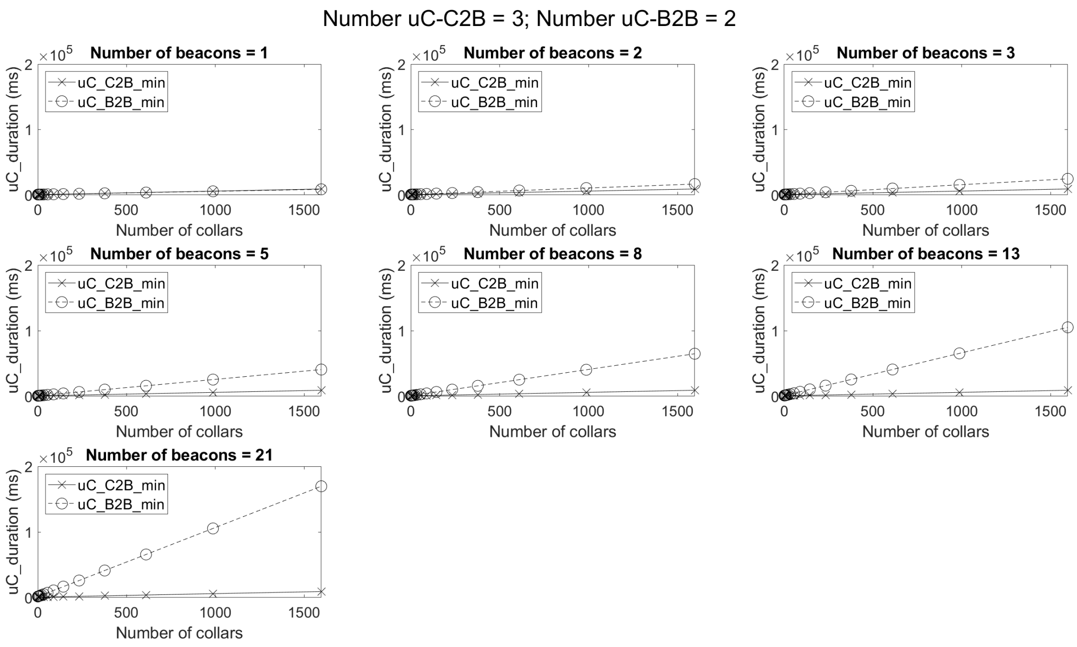


Figure B.13: μC minimum duration for the combination $N_{\mu C_{C2B}} = 3$; $N_{\mu C_{B2B}} = 2$.

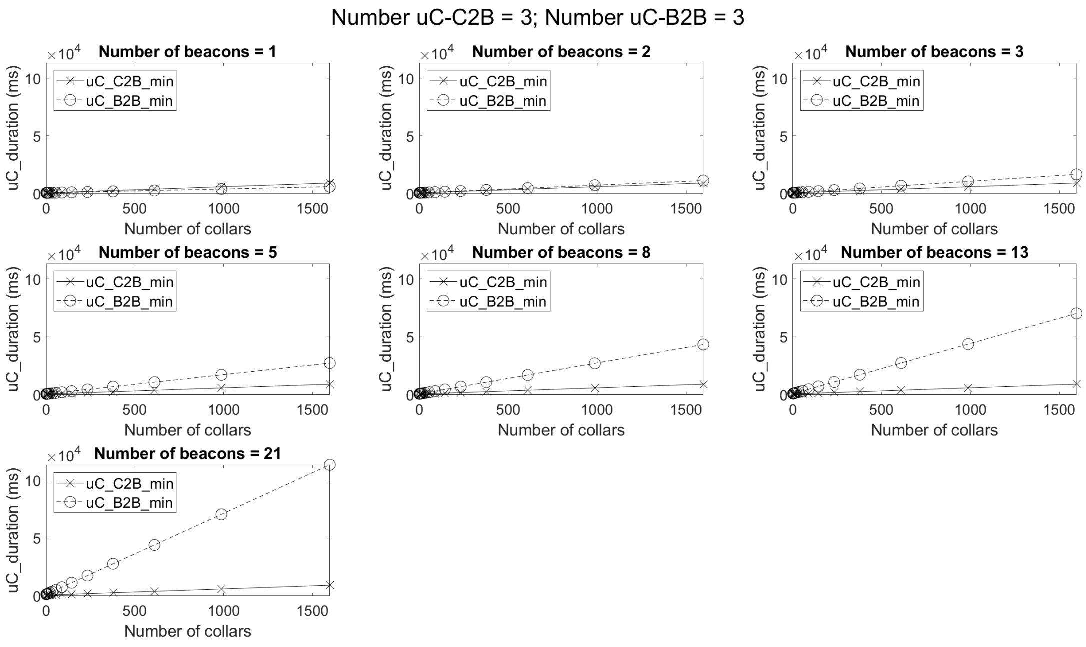


Figure B.14: μC minimum duration for the combination $N_{\mu C_{C2B}} = 3$; $N_{\mu C_{B2B}} = 3$.

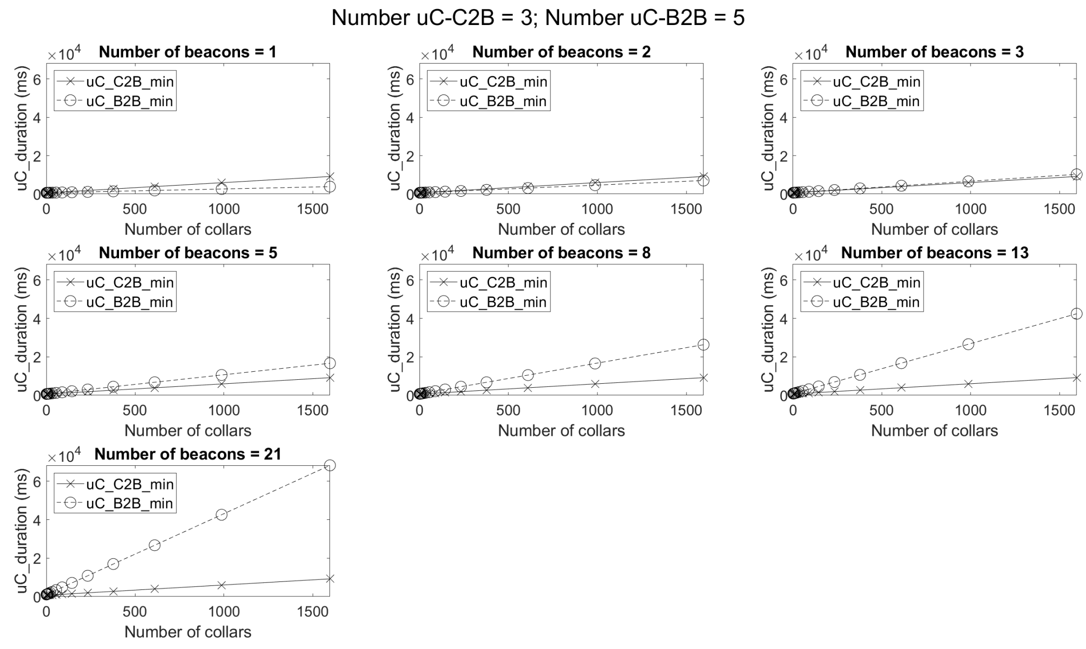


Figure B.15: μC minimum duration for the combination $N_{\mu C_{C2B}} = 3$; $N_{\mu C_{B2B}} = 5$.

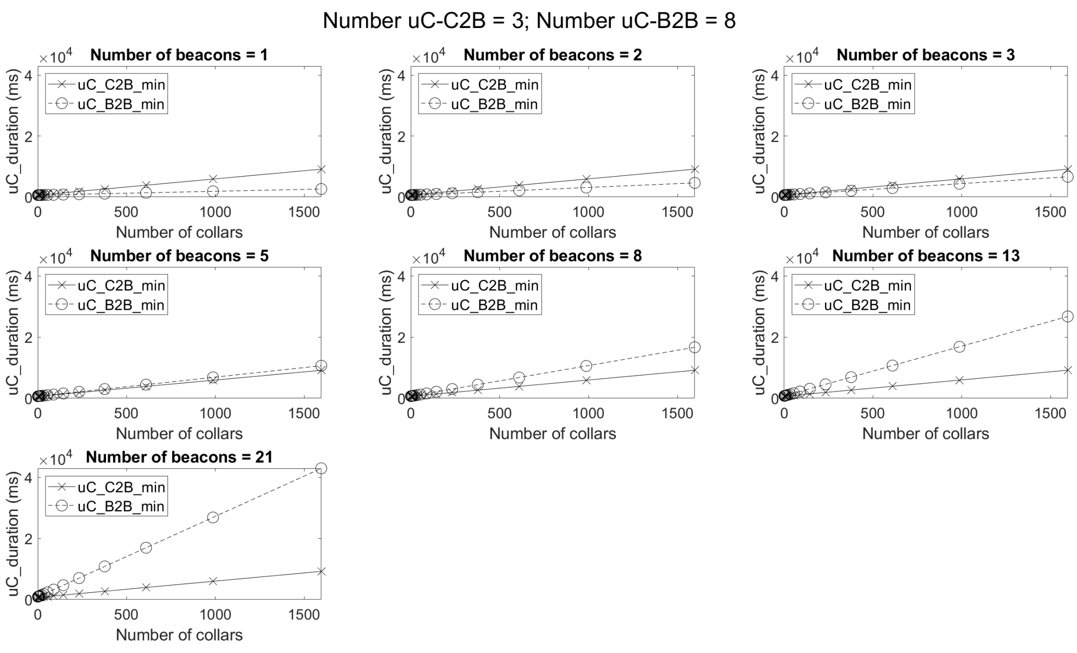


Figure B.16: μC minimum duration for the combination $N_{\mu C_{C2B}} = 3$; $N_{\mu C_{B2B}} = 8$.

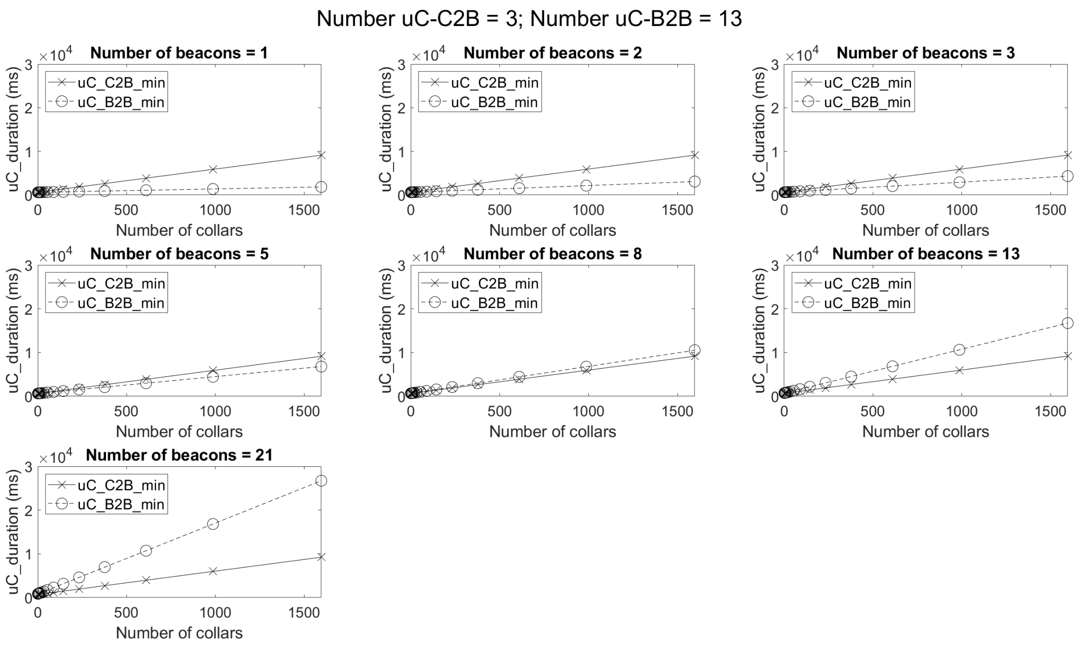


Figure B.17: μC minimum duration for the combination $N_{\mu C_{C2B}} = 3$; $N_{\mu C_{B2B}} = 13$.

APPENDIX C

Appendix C: Maximum number of collars and average consumption considering the worst-case scenario

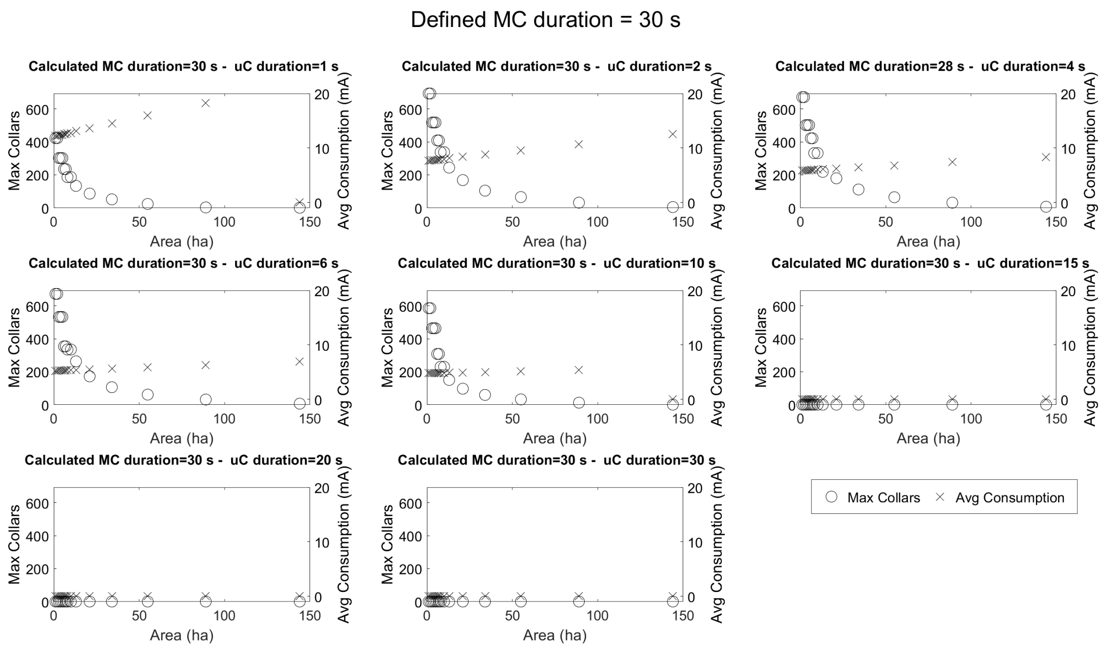


Figure C.1: Maximum number of collars and average consumption considering the worst-case scenario, where $MC_{duration} = 30s$.

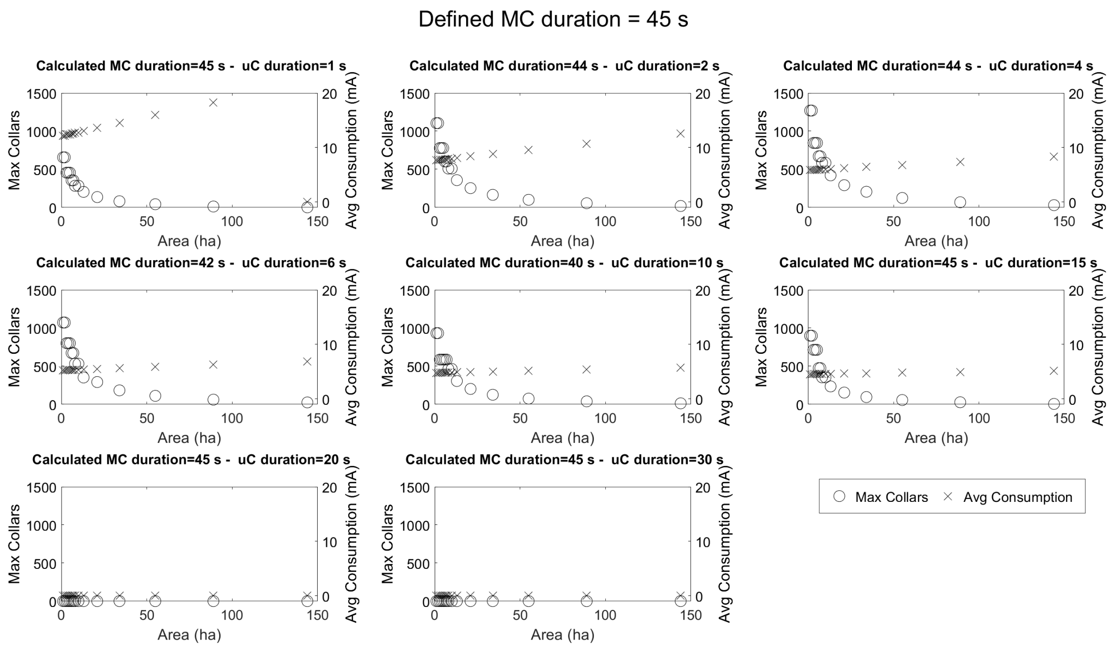


Figure C.2: Maximum number of collars and average consumption considering the worst-case scenario, where $MC_{duration} = 45s$.

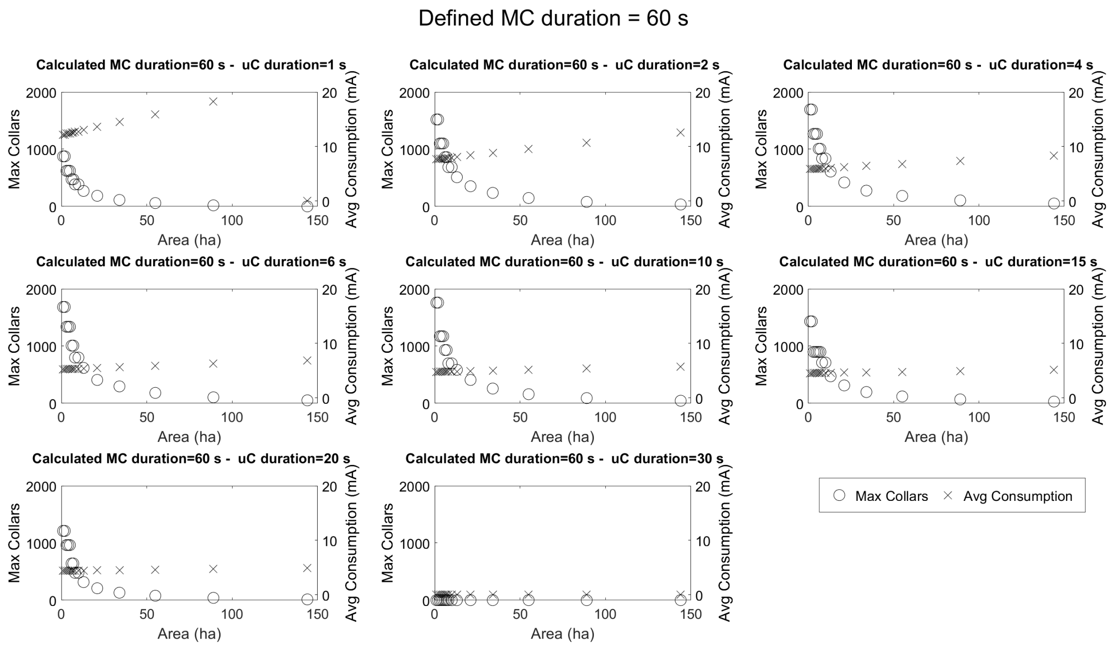


Figure C.3: Maximum number of collars and average consumption considering the worst-case scenario, where $MC_{duration} = 60s$.

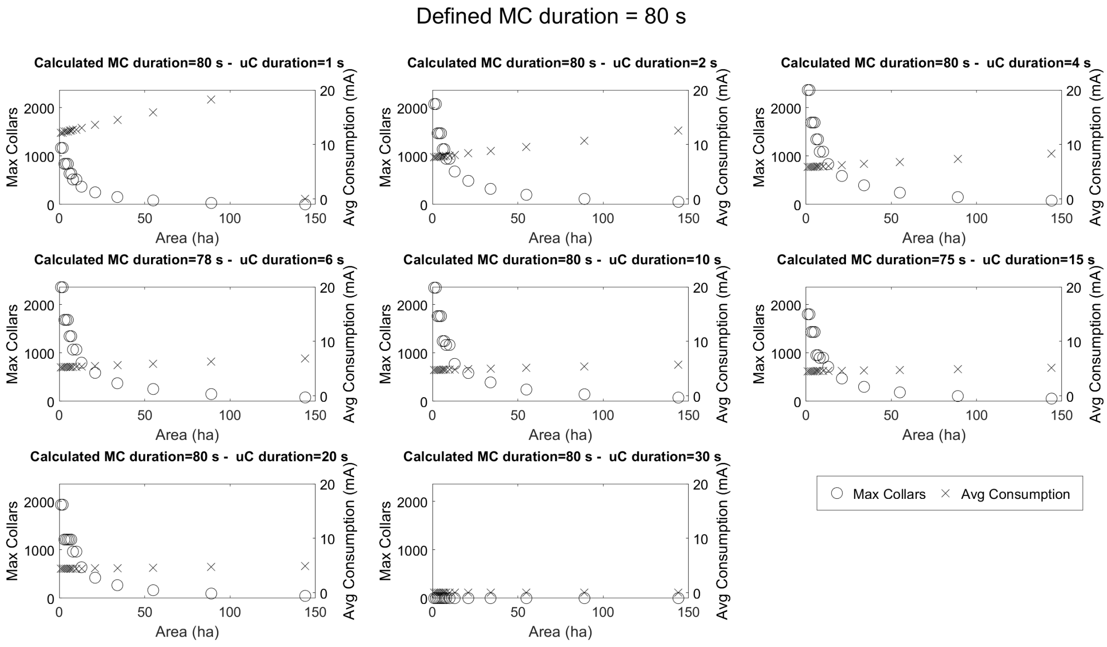


Figure C.4: Maximum number of collars and average consumption considering the worst-case scenario, where $MC_{duration} = 80s$.

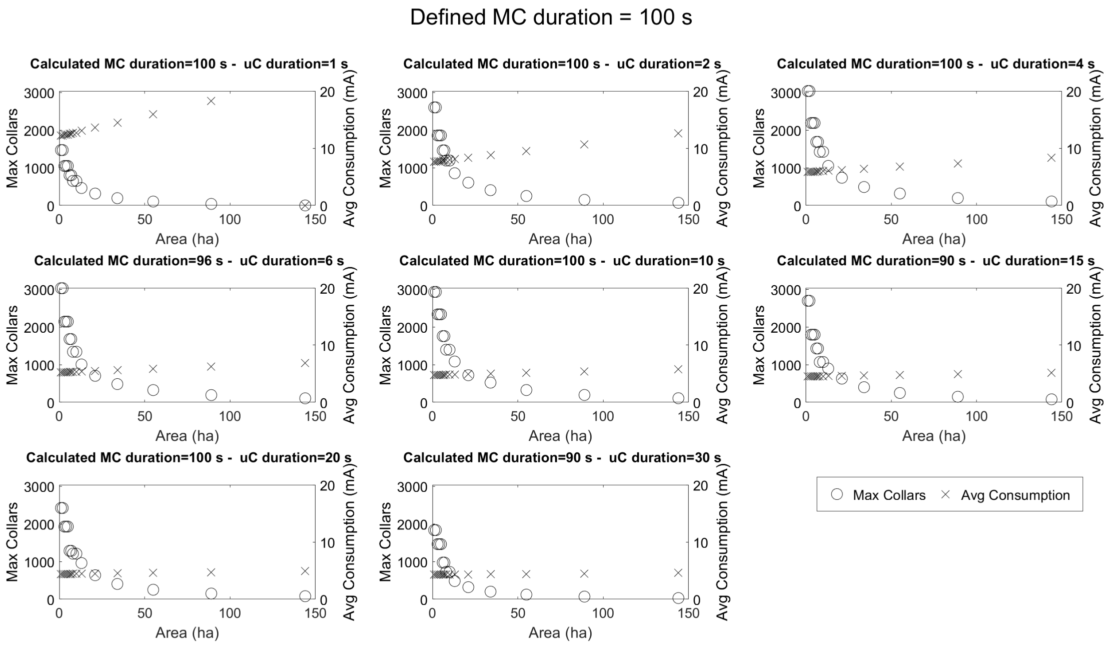


Figure C.5: Maximum number of collars and average consumption considering the worst-case scenario, where $MC_{duration} = 100s$.

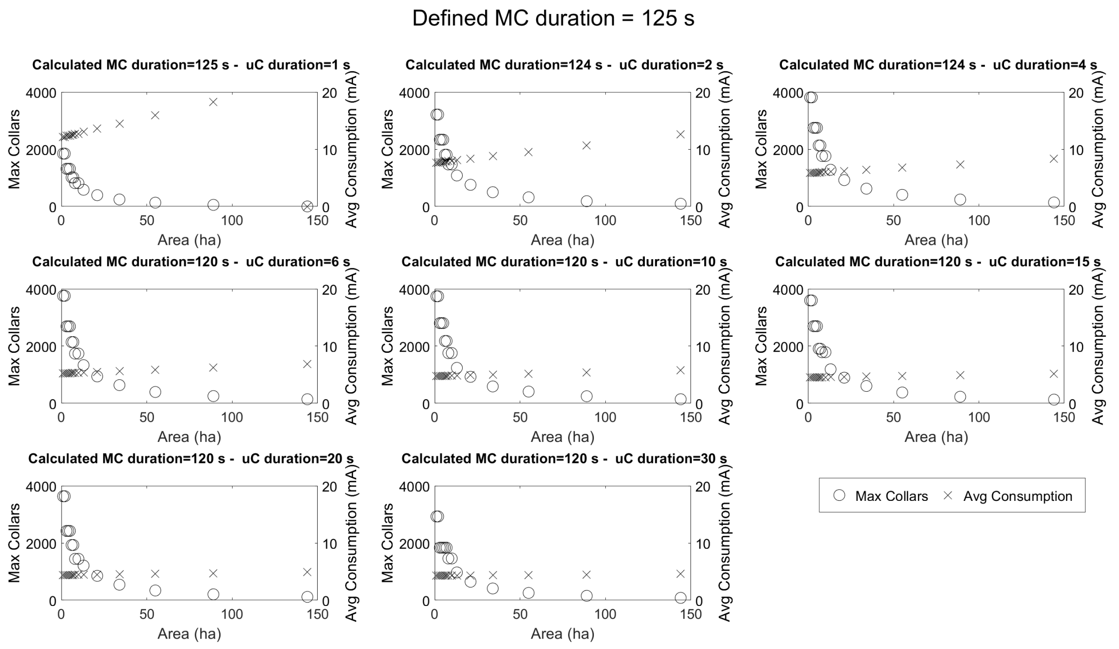


Figure C.6: Maximum number of collars and average consumption considering the worst-case scenario, where $MC_{duration} = 125s$.

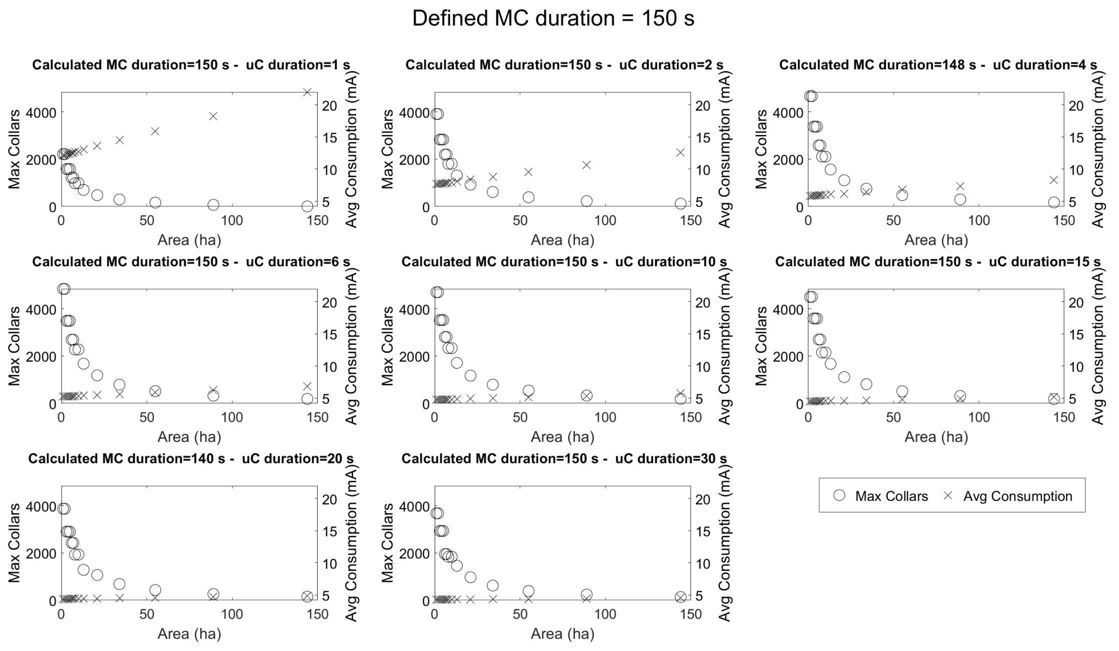


Figure C.7: Maximum number of collars and average consumption considering the worst-case scenario, where $MC_{duration} = 150s$.

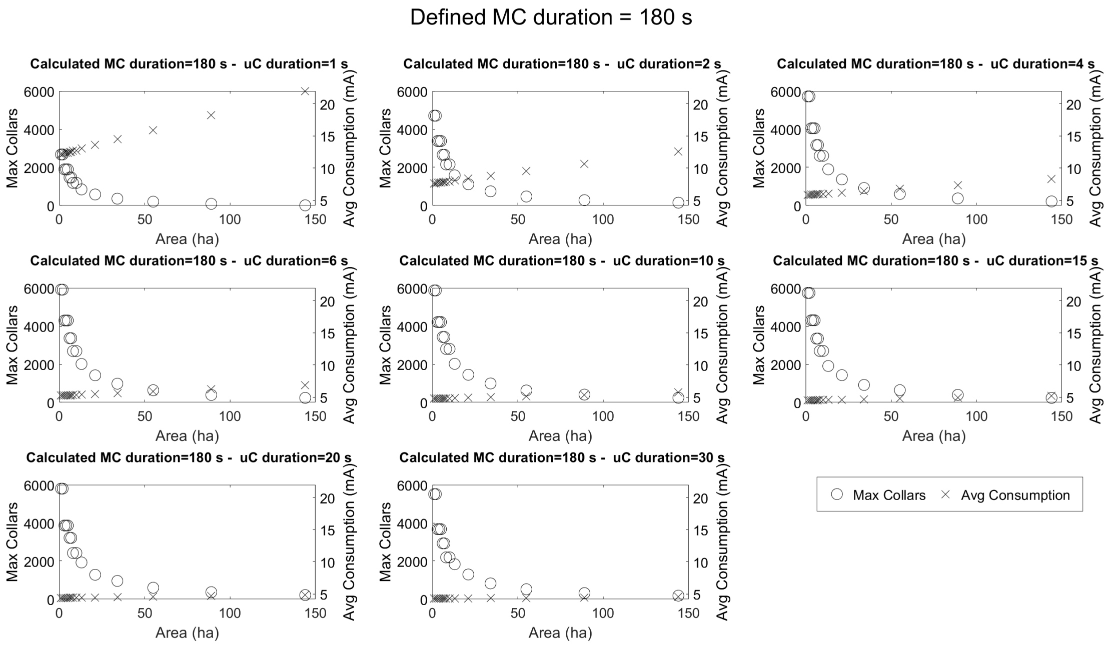


Figure C.8: Maximum number of collars and average consumption considering the worst-case scenario, where $MC_{duration} = 180s$.