ISSN 1688-2806

Universidad de la República
Facultad de Ingeniería

OVIS System: a research platform for online and embedded monitoring of sheep behavior

presented to Facultad de Ingeniería of Universidad de la República by

Varinia Cabrera

in partial fulfillment of the requirements for obtaining the degree of
Magíster en Ingeniería Eléctrica.

Thesis Director
Dr. Julián Oreggioni ..... Universidad de la República

Examiners
Prof. Alicia Fernández ..... Universidad de la República
Prof. Antoni Ivorra ..... Universitat Pompeu Fabra
Dr. Leonardo Steinfeld ..... Universidad de la República

Academic Director
Dr. Julián Oreggioni ..... Universidad de la República

Montevideo
Wednesday 15th April, 2026

OVIS System: a research platform for online and embedded monitoring of sheep behavior, Varinia Cabrera.

ISSN 1688-2806

This thesis was prepared in LaTeX using the iietesis class (v1.1).
It contains a total of 146 pages.
Compiled on Wednesday 15th April, 2026.
http://iie.fing.edu.uy/

Acknowledgments

I would like to acknowledge the contributions of the many people and institutions involved in this work. In the area of embedded software, I thank Andrea Delbuggio (Department of Electronics, IIE, FING, Universidad de la República) for his support and collaboration. The signal processing component of this work was strengthened by the technical insight of Álvaro Gómez (Department of Signals, IIE, FING, Universidad de la República). The development and operation of the server infrastructure was supported by Hernán Cardoso and Martín Pedemonte (Laboratory of Heterogeneous Computing, INCO, FING, Universidad de la República). I am also grateful to Victoria Fernández and Rodolfo Ungerfeld (Department of Veterinary Biosciences, FVET, Universidad de la República) for their expertise in ovine behavior, and to Diego Fraga (DVL Group) for his work on the mechanical design. The assembly of the OVIS devices was carefully carried out by Lucía Sirio. I would also like to thank the members of the thesis evaluation committee—Prof. Alicia Fernández (Universidad de la República), Prof. Antoni Ivorra (Universitat Pompeu Fabra), and Dr. Leonardo Steinfeld (Universidad de la República)—for their time and valuable feedback. Finally, I would like to express my gratitude to my academic advisor, Dr. Julián Oreggioni (Universidad de la República), for his guidance, support, and continuous encouragement throughout this work.

This page has been intentionally left blank.

Abstract

This thesis presents the design, implementation, and evaluation of the OVIS system, a research system for online behavioral monitoring of sheep in extensive livestock systems. The OVIS system includes a custom made collar, referred to as the OVIS device, capable of collecting motion and location data, running an embedded classification algorithm, and transmitting processed information to the OVIS cloud server through Narrowband IoT (NB-IoT). The resulting architecture enables remote behavior analysis without requiring proprietary infrastructure or raw data transmission. In this work, the signal processing pipeline is presented as a proof of concept, intended to illustrate the potential of the developed tool rather than as a finalized, production solution.

The OVIS device features Actinius' Icarus IoT Board, which includes an nRF9160 System in Package (from Nordic Semiconductor) with NB-IoT and Global Navigation Satellite System (GNSS) capabilities. The Icarus IoT Board incorporates the LIS2DH12 three-axis accelerometer from STMicroelectronics and a power management circuit, based on the BQ24074 chip from Texas Instruments, allowing the device to be powered from various sources: three solar panels (totaling 2.8 W), a 2500 mAh Lithium Polymer (LiPo) rechargeable battery, and a micro-B USB port.

The embedded software, referred to as the OVIS application, was built on Zephyr Real Time Operating System (RTOS) and adapted from a Nordic Semiconductor application. It is responsible for periodically acquiring location data, accelerometer readings, battery level, and cellular signal strength. The OVIS application implements online behavior classification using a Random Forest (RF) algorithm, which was trained on a Personal Computer (PC) and translated into C code using Emlearn (a tool to convert trained machine learning models into C code). This enables on-animal inference, meaning that behavior is classified directly on the device worn by the animal. The inference is done by computing signal features over acceleration windows and classifying behaviors on-device, which are then transmitted to be available online. In this context, online classification refers to the ability to transmit raw sensor data for live, server based classification, or to transmit already classified behaviors so that they become available on a computer or server with minimal delay. The OVIS application can transmit both raw and processed data via NB-IoT to the OVIS cloud server for further storage, analysis, and visualization.

Designed with autonomy in mind, it incorporates low power modes. The OVIS application also supports system scalability, enabling future integration of additional sensors and a low transmission mode where only processed data and location data are sent at extended intervals, significantly reducing bandwidth and power consumption. In normal operation, the OVIS device demonstrated over 11 days of autonomy without solar input, with potential for more than 150 days in a low transmission mode.

The OVIS cloud server was built using AWS services, including IoT Core, Lambda, and DynamoDB, to ensure secure, scalable communication between the OVIS devices and the OVIS cloud server. Message Queuing Telemetry Transport (MQTT) over

Transport Layer Security (TLS) ensures secure, low latency transmission of data, which is then processed and made accessible through a responsive web interface built with NextJS 13. The OVIS cloud server supports visualization of battery levels, predicted behaviors, and GNSS locations over selectable time periods, allowing users to monitor animals individually or at flock level.

The OVIS classifier aims to distinguish three distinct locomotion classes: still, walking, and running. The evaluation of the OVIS classifier was done in three stages. First, an embedded version trained using a public dataset and tested with OVIS data (i.e., data collected in the present work by the OVIS device), achieved an overall accuracy of 81%. To improve performance, a second model was trained and tested on the PC using exclusively OVIS data, reaching 84% accuracy. Finally, a third version for binary classification (still vs. movement) was also was trained and tested on the PC with OVIS data, achieving 86% accuracy. Although classification of walking and running remained challenging due to the inherent difficulty in distinguishing dynamic behaviors under natural grazing conditions, the classification algorithm successfully fulfilled its role as a proof of concept.

Thirty OVIS devices were manufactured, and a subset of them was evaluated in laboratory and on-farm scenarios. The mechanical design was developed to ensure durability, proper fit, and animal comfort under extensive grazing conditions. The collar worn enclosure (3D printed in polyethylene terephthalate glycol modified with transparent polycarbonate windows) was iteratively refined to withstand impacts, moisture, and long term outdoor exposure. Adjustable straps and multiple sizes enabled secure attachment across sheep morphologies, while design features such as neutral coloring, USB protection, and antenna placement supported usability and signal performance.

Compared to state of the art and commercial systems, the OVIS system stands out for its modular design, on-animal classification, cloud based architecture, and its possibility to extend autonomy through solar panels. Unlike most alternatives, it relies exclusively on public cellular infrastructure, reducing deployment barriers. Moreover, it supports future enhancements, including additional sensors, extensions to the behavior classification algorithm, and cloud-device collaboration. Despite these strengths, some challenges remain. NB-IoT coverage in rural Uruguay is inconsistent, leading to packet losses exceeding 40% in some areas. Furthermore, GNSS accuracy is limited by signal quality and modem transmission constraints. These issues motivate future exploration of advanced filtering strategies, and differential GNSS correction.

In conclusion, this work provides a foundation for a flexible behavioral monitoring system tailored to the needs of extensive livestock farming. The OVIS system integrates an embedded device equipped with sensors and reliable connectivity, and a user friendly cloud server. This system enables autonomous, scalable monitoring, supporting both scientific investigations and practical flock management. Continued development and optimization of the OVIS system could transform it into a viable long term solution for deployments in agriculture and animal science. These developments include: behavior classifier refinement, cost reduction strategies, enhanced autonomy via solar harvesting or a low transmission mode, improved data logging under poor connectivity conditions (e.g., SD card buffering), and simplified collar management through remote configuration and firmware updates.

Resumen

Esta tesis presenta el diseño, la implementación y la evaluación del sistema OVIS, un sistema de investigación para el monitoreo de comportamiento online de ovinos en sistemas de ganadería extensiva. El sistema OVIS incluye un collar diseñado a medida, denominado dispositivo OVIS, capaz de recolectar datos de movimiento y localización, ejecutar un algoritmo de clasificación embebido y transmitir información procesada al servidor en la nube de OVIS a través de Narrowband IoT (NB-IoT). La arquitectura resultante permite el análisis remoto del comportamiento sin requerir infraestructura propietaria ni la transmisión de datos crudos. En este trabajo, el procesamiento de señales se presenta como una prueba de concepto, destinado a ilustrar el potencial de la herramienta desarrollada más que como una solución final de producción.

El dispositivo OVIS incorpora la placa Icarus IoT Board de Actinius, que incluye un nRF9160 (de Nordic Semiconductor) con capacidades de NB-IoT y Sistema Global de Navegación por Satélite (GNSS). La Icarus IoT Board integra el acelerómetro triaxial LIS2DH12 de STMicroelectronics y un circuito de gestión de energía basado en el chip BQ24074 de Texas Instruments, lo que permite alimentar el dispositivo a partir de diversas fuentes: tres paneles solares (con una potencia total de 2.8 W), una batería recargable de Polímero de Litio (LiPo) de 2500 mAh y un puerto USB micro-B.

El software embebido, denominado aplicación OVIS, fue desarrollado sobre el Sistema Operativo en Tiempo Real (RTOS) Zephyr y adaptado a partir de una aplicación de Nordic Semiconductor. Es responsable de adquirir periódicamente datos de localización, lecturas del acelerómetro, nivel de batería e intensidad de la señal celular. La aplicación OVIS implementa clasificación de comportamiento en línea utilizando un algoritmo de Random Forest (RF), que fue entrenado en una Computadora Personal (PC) y traducido a código C mediante Emlearn (una herramienta para convertir modelos de aprendizaje automático entrenados a código C). Esto habilita la inferencia on-animal, es decir, que el comportamiento se clasifica directamente en el dispositivo que lleva el animal.

La inferencia se realiza mediante el cálculo de características de la señal sobre ventanas de aceleración y la clasificación del comportamiento en el propio dispositivo, cuyos resultados luego se transmiten para estar disponibles online. En este contexto, la clasificación online se refiere a la capacidad de transmitir datos crudos de sensores para su clasificación en vivo basada en el servidor, o de transmitir comportamientos ya clasificados para que estén disponibles en una computadora o un servidor con un retardo mínimo. La aplicación OVIS puede transmitir tanto datos crudos como procesados vía NB-IoT al servidor OVIS para su posterior almacenamiento, análisis y visualización.

Diseñada con la autonomía como objetivo, incorpora modos de bajo consumo. La aplicación OVIS también soporta la escalabilidad del sistema, permitiendo la futura integración de sensores adicionales y un modo de baja transmisión en el que solo se envían datos procesados y datos de localización a intervalos extendidos, reduciendo

significativamente el uso de ancho de banda y el consumo energético. En operación normal, el dispositivo OVIS demostró más de 11 días de autonomía sin aporte solar, con potencial para superar los 150 días en un modo de baja transmisión.

El servidor OVIS fue construido utilizando servicios de AWS, incluyendo IoT Core, Lambda y DynamoDB, para garantizar una comunicación segura y escalable entre los dispositivos OVIS y el servidor OVIS. El protocolo Message Queuing Telemetry Transport (MQTT) sobre Transport Layer Security (TLS) asegura la transmisión segura y de baja latencia de los datos, que luego son procesados y puestos a disposición a través de una interfaz web de tipo responsiva desarrollada con NextJS 13. El servidor OVIS permite la visualización de niveles de batería, comportamientos clasificados y datos de ubicación en períodos de tiempo seleccionables, permitiendo a los usuarios monitorear animales de forma individual o a nivel de majada.

El clasificador OVIS tiene como objetivo distinguir tres clases de locomoción diferentes: quieto, caminando y corriendo. La evaluación del clasificador OVIS se realizó en tres etapas. En primer lugar, una versión embebida entrenada utilizando un conjunto de datos público y evaluada con datos OVIS (es decir, datos recolectados en el presente trabajo por el dispositivo OVIS) alcanzó una precisión global del 81 %. Para mejorar el desempeño, un segundo modelo fue entrenado y evaluado en la PC utilizando exclusivamente datos OVIS, alcanzando una precisión del 84 %. Finalmente, una tercera versión para clasificación binaria (quieto vs. movimiento) también fue entrenada y evaluada en la PC con datos OVIS, alcanzando una precisión del 86 %. Si bien la clasificación de caminar y correr resultó desafiante debido a la dificultad inherente de distinguir comportamientos dinámicos bajo condiciones naturales de pastoreo, el algoritmo de clasificación cumplió exitosamente su rol como prueba de concepto.

Se fabricaron treinta dispositivos OVIS, y un subconjunto de ellos fue evaluado en escenarios de laboratorio y en campo. El diseño mecánico fue desarrollado para garantizar durabilidad, ajuste adecuado y confort animal bajo condiciones de pastoreo extensivo. El encapsulado del collar (impreso en 3D en tereftalato de polietileno modificado con glicol, con ventanas transparentes de policarbonato) fue refinado iterativamente para resistir impactos, humedad y exposición prolongada al aire libre. Correas ajustables y múltiples tamaños permitieron una sujeción segura para distintas morfologías ovinas, mientras que características de diseño como coloración neutra, protección del puerto USB y ubicación de la antena favorecieron la usabilidad y el desempeño de la señal.

En comparación con el estado del arte y con sistemas comerciales, el sistema OVIS se destaca por su diseño modular, la clasificación on-animal, su arquitectura basada en la nube y la posibilidad de extender la autonomía mediante paneles solares. A diferencia de la mayoría de las alternativas, se apoya exclusivamente en infraestructura celular pública, reduciendo las barreras de despliegue. Además, soporta mejoras futuras, incluyendo sensores adicionales, extensiones del algoritmo de clasificación de comportamiento y colaboración de procesamiento nube-dispositivo.

A pesar de estas fortalezas, persisten algunos desafíos. La cobertura NB-IoT en zonas rurales de Uruguay es inconsistente, lo que conduce a pérdidas de paquetes que superan el 40 % en algunas áreas. Asimismo, la precisión GNSS está limitada por la calidad de la señal y las restricciones de transmisión del módem. Estos aspectos motivan la exploración futura de estrategias avanzadas de filtrado y corrección GNSS diferencial.

En conclusión, este trabajo proporciona una base para un sistema flexible de monitoreo de comportamiento ovino adaptado a las necesidades de la ganadería extensiva. El sistema OVIS integra un dispositivo embebido equipado con sensores y conectividad confiable, y un servidor en la nube fácil de usar. Este sistema permite un monitoreo autónomo y escalable, apoyando tanto investigaciones científicas como la gestión

práctica de majadas. El desarrollo y la optimización del sistema OVIS podrían transformarlo en una solución viable a largo plazo para despliegues en agricultura y ciencias animales. Estos desarrollos incluyen: refinamiento del clasificador de comportamiento, estrategias de reducción de costos, mejora de la autonomía mediante cosecha solar o un modo de baja transmisión, registro de datos mejorado bajo condiciones de conectividad deficiente (por ejemplo, almacenamiento en búfer mediante tarjeta SD) y una gestión simplificada de collares mediante configuración remota y actualizaciones de firmware.

Glossary

This chapter provides definitions for certain words and acronyms used throughout the document. The aim is to offer a basic understanding of these terms to aid understanding their use in this manuscript. However, it is not intended to provide an exhaustive explanation. If the reader finds any concepts unclear or incomplete, it is recommended to conduct further research online.

3GPP Third Generation Partnership Project is a global collaboration between telecommunications standardization bodies responsible for developing and maintaining cellular communication standards, including LTE and NB-IoT.

6LoWPAN IPv6 over Low power Wireless Personal Area Networks is a protocol enabling IPv6 communication over low power, low bandwidth IEEE 802.15.4 networks, commonly used in IoT.

accuracy A classification performance metric that measures the proportion of correctly predicted instances among all instances. It is calculated as (where TP is True Positive, TN is True Negative, FP is False Positive and FN is False Negative):

$$\text{accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

ADC Analog to Digital Converter is a circuit that transforms continuous analog signals into discrete digital values so the signal could be processed in a digital system.

AI Artificial Intelligence refers to the area of knowledge involving systems' development capable of performing tasks such as problem solving, learning, decision making, and natural language understanding.

ANOVA F-value Statistical measure that quantifies how well a feature distinguishes between classes by comparing variance between groups.

ANTEL Administración Nacional de Telecomunicaciones, the state owned national cellular network provider in Uruguay.

API Application Programming Interface is a set of defined methods and protocols for software components to communicate and interact.

APN Access Point Name is a gateway between a mobile network and another network (typically the internet), used to configure device network access.

AT Active Timer is a 3GPP timer that specifies how long a device remains reachable for downlink traffic after completing an uplink transmission before entering PSM.

BT Bluetooth is a wireless communication technology for short distances between devices such as IoT systems using low energy radio.

CART Classification and Regression Trees is a ML model. It splits data to obtain a classification or a continuous value outcome (regression). CART recursively divides data and can reduce the size of the decision tree to avoid overfitting.

CAT M1 Category M1 is a cellular IoT technology based on LTE, offering medium data rates, low power consumption, and extended coverage for IoT devices.

CCA Canonical Correlation Analysis is a statistical method used to relate two datasets (e.g. entry data set and the classification groups). This is done finding canonical variable pairs maximizing correlation between the variables.

CDA Canonical Discriminant Analysis is a special case of LDA. CDA aims to maximize class separability by finding canonical variables through linear transformations.

CNN Convolutional Neural Networks are NN for tasks as image processing. Convolutional layers apply filters to the input, extracting patterns. These features are passed into layers to reduce dimensions and to obtain a classification.

CSV Comma Separated Values is a file format used to store tabular data, where each line represents a row and columns are separated by commas.

DA Discriminant Analysis is a statistical classification method that assumes each class follows a Gaussian distribution and seeks to find a combination of features that best separate the classes.

Device tree A data structure used to describe hardware layout to an operating system, often used in embedded systems like Zephyr for hardware configuration.

DNN Deep Neural Networks are a type of NN. An NN is considered deep if it has at least two hidden layers.

DNS Domain Name System translates human readable domain names (e.g., www.example.com) into an IP address used for routing data.

DT Decision Trees is a supervised ML method used for classification. The data is separated into branches based on data characteristics (features) until a classification decision is reached when no more divisions are made.

eDRX Extended Discontinuous Reception is a mechanism defined in LTE and NB-IoT that allows a device to reduce power consumption by extending the intervals at which it listens for downlink paging messages from the network.

ELM Extreme Learning Machine is a ML system that uses a feed forward NN to train data. ELM uses randomly assigned weights in the hidden layer, which are not updated after initialization. This allows the ELM to learn data quickly.

eSIM Embedded SIM is a programmable SIM card embedded into a device, allowing remote provisioning without physical replacement.

F1-score The harmonic mean of precision and sensitivity, used as a balanced performance metric especially for imbalanced datasets. It is calculated as:

$$F1 = 2 \cdot \frac{\text{precision} \cdot \text{sensitivity}}{\text{precision} + \text{sensitivity}}$$

fix Fix is considered a successful determination of position by a GNSS receiver, obtained when sufficient satellite signals meet required quality conditions.

FOTA Firmware Over The Air refers to remotely updating embedded device firmware via wireless communication, allowing bug fixes or upgrades without physical access.

FPU Floating Point Unit is a circuit within a CPU (Central Processing Unit) designed to handle mathematical operations involving floating point numbers.

FSA Forward Stepwise Analysis is a feature selection method. The method starts with no variables and then, it tests each variable selecting the most statistically significant improvement to the classification.

Fselector An R package which provides tools for feature selection.

GNSS Global Navigation Satellite System includes satellite systems like GPS, GLONASS, and Galileo, providing positioning, navigation, and timing services globally.

GPIO General Purpose Input Output are programmable pins on a microcontroller or other hardware devices used to interface with external components like sensors, LEDs, or buttons.

GPS Global Positioning System is a satellite based navigation system providing location and time information.

I2C Inter-Integrated Circuit is a wired master–slave serial communication protocol used for short distance communication between a microcontroller and its peripherals.

ICCID Integrated Circuit Card Identifier is a unique identifier assigned to a SIM card used in mobile networks.

IDE Integrated Development Environment is a software that provides editors, debuggers, build and flash tools to facilitate embedded software development.

IEEE 802.15.4 Standard defining the physical (PHY) and medium access control (MAC) layers for low power, low rate wireless personal area networks.

IMEI International Mobile Equipment Identity is a unique identifier for mobile devices used for device authentication and tracking on cellular networks.

IoT Internet of Things describes a network of devices with sensors, software, and connectivity. The devices are able to collect and communicate data over the internet.

IP Internet Protocol is the set of rules governing how data is sent and received over networks.

IP address Internet Protocol address is a numerical label assigned to each device connected to a network for identification and communication.

IRQ Interrupt Request is a signal sent to a processor to indicate that an event needs immediate attention, commonly used in real time embedded systems.

JSON JavaScript Object Notation is a lightweight data interchange format commonly used in web services and APIs to structure and exchange information.

K Best A univariate feature selection method that ranks all features based on a statistical score and selects the top k best features according to that score.

K-means K-means is a clustering algorithm that divides data into k clusters by minimizing the distance between points within each cluster and their respective centers.

Kappa value Kappa measures classification agreement corrected for chance; 1 indicates perfect, 0 is random.

kernel The core component of an operating system that manages hardware resources and enables communication between hardware and software layers.

KNN k-Nearest Neighbor is an algorithm for classification that assigns a label based on the majority class among the k closest data points.

LDA Linear Discriminant Analysis is a linear classifier that enables to reduce the data dimensions through projecting a dataset onto a lower dimensional space with acceptable class separability.

LiPo Lithium Polymer is a type of rechargeable battery known for its lightweight, and energy density, used in portable electronics and IoT devices.

LoRa Long Range is a low power, long range wireless communication technology for IoT networks, enabling transmission over several kilometers with low data rates.

LoRaWAN Long Range Wide Area Network. A low power, wide area networking protocol built on top of the LoRa physical layer.

LPM Low Power Mode refers to operational states of a device or microcontroller designed to minimize energy consumption.

LSTM Long Short Term Memory is a type of recurrent NN (RNN) designed to capture long term dependencies in sequential data, addressing issues of standard RNN by using memory cells and gates to regulate information flow.

LTE Long Term Evolution is a standard for wireless broadband communication. It offers transmission for mobile devices and IoT systems.

MCC Mobile Country Code is a part of the identifier for a mobile network, specifying the country of the mobile subscriber.

ML Machine Learning is the field that develops algorithms capable of learning patterns from data and making predictions.

MNC Mobile Network Code is used in combination with MCC to uniquely identify a mobile network operator.

MQTT Message Queuing Telemetry Transport is a messaging protocol widely used in IoT applications for communication between devices and servers.

NB-IoT Narrowband IoT is a low power wide area network (LPWAN) technology optimized for IoT applications, offering energy efficient, long range communication and support for massive device connectivity.

NN Neural Network is a ML model consisting of nodes organized in layers. Nodes process input data with an activation function, and pass the output to other nodes. Data flows through an input, multiple hidden, and an output layer.

NoSQL A family of non-relational databases designed for large scale data storage and real time web applications, often used in IoT backends.

on-animal classification Behavior classification performed directly on the animal worn device.

online classification A classification scheme that refers to the ability to transmit raw sensor data for live, server based classification, or to transmit already classified behaviors so that they become available on a computer or server with minimal delay.

PC PC is personal computer.

PCA Principal Component Analysis is a feature selection technique through dimensionality reduction that transforms data into orthogonal components ranked by their ability to reflect data variance.

PCB Printed Circuit Board is a board that connects electronic components using conductive tracks made from copper, on a non-conductive base.

PCBA Printed Circuit Board Assembled refers to a PCB with electronic components already mounted.

PCO Protocol Configuration Options are negotiated parameters exchanged between a device and network during attachment, such as IP type, DNS, and authentication.

PDN Packet Data Network refers to a network providing IP connectivity between a mobile device and external data services like the internet.

PDOP Position Dilution of Precision is a value that reflects the geometrical quality of satellite positions. A lower PDOP indicates better positioning accuracy.

PLF Precision Livestock Farming is related to innovation in farming techniques, using technologies for data gathering and processing. An aim for PLF is to automate the animal monitoring while optimizing health, welfare, and productivity of livestock.

PLMN Public Land Mobile Network is a network established and operated by a mobile service provider, identified by MCC and MNC.

PMC Power Management Circuit is circuit responsible for managing power usage in electronic systems.

PME Pattern Matching Engine is a tool designed to identify patterns within datasets. This document refers a particular PME with 128 nodes capable of performing models like KNN.

precision A classification performance metric defined as the ratio of true positive predictions to the total predicted positives. It is calculated as (where TP is True Positive and FP is False Positive):

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

PSM Power Saving Mode is a cellular IoT feature that allows a device to enter deep sleep states while remaining registered to the network, minimizing power consumption.

QDA Quadratic Discriminant Analysis is a classification technique that extends LDA by allowing for quadratic decision boundaries, increasing flexibility for more complex data where linear limits are not enough.

RA Relief Algorithm is a feature selection technique that evaluates features based on value differences in nearest neighbor pairs. It decreases a feature's score for differences in same class pairs and increases it for differences in different class pairs.

RF Random Forest is a supervised ML method used for classification. The technique uses multiple DT and combines their outputs for a final result.

ROC Receiver Operating Characteristic is a plot of the true positive rate (sensitivity) in function of the false positive rate (specificity) for different threshold value of a parameter.

RRC Radio Resource Control is a signaling protocol defined by 3GPP that manages the allocation, maintenance, and release of radio resources in cellular networks.

RSRP Reference Signal Received Power is a LTE metric indicating the average received signal strength from a cell tower.

RTC Real Time Clock is a low power clock used in embedded systems to keep track of current time, even when the main system is off.

RTOS A Real Time Operating System is an operating system designed to guarantee timing deadlines by providing deterministic task scheduling and predictable response times. It is widely used in embedded systems and IoT applications.

SD SD card is a non-volatile flash memory card used for external data storage.

SDA Stepwise Discriminant Analysis is a feature selection method that iteratively includes or excludes variables based on their statistical significance for improving classification performance.

sensitivity Also known as recall, it is a metric that measures the proportion of actual positives that were correctly identified. It is calculated as (where TP is True Positive and FN is False Negative):

$$\text{sensitivity} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

SFS Sequential Forward Selection is a feature selection technique. At each step, it selects the feature that most improves the model's performance according to a criterion (e.g., accuracy, error rate).

SIM Subscriber Identity Module is a chip that securely stores user identity and encryption keys used to authenticate devices to mobile networks.

SMOTE Synthetic Minority Over sampling Technique is a resampling technique that generates artificial samples for the minority class in imbalanced datasets.

specificity A classification metric that measures the proportion of actual negatives correctly identified as such. It is calculated as (where TN is True Negative and FP is False Positive):

$$\text{specificity} = \frac{\text{TN}}{\text{TN} + \text{FP}}$$

SPI Serial Peripheral Interface is a wired synchronous serial communication protocol used for high speed communication between a microcontroller and its peripherals.

SSR Static Site Rendering is a web development technique where pages are prerendered at build time, enhancing load speed and reducing server load.

SVM Support Vector Machine is a supervised learning model. This model consist in finding the hyperplane which best separates data into classes, focusing on maximizing the margin between the closest data points of opposite classes.

TAU Tracking Area Update timer defines the maximum time a device may remain in PSM without sending an update to maintain its registration.

TCP Transmission Control Protocol is a connection oriented communication protocol. TCP is design to ensure reliable, ordered, and error checked data transmission between devices in a network.

TLS Transport Layer Security is a cryptographic protocol that ensures secure communication over a network, widely used in MQTT and HTTPS.

TTFF Time To First Fix is the time a GNSS receiver takes to acquire satellite signals and calculate its initial position.

UART Universal Asynchronous Receiver-Transmitter is a hardware communication protocol that enables asynchronous serial communication between devices.

UDP User Datagram Protocol is a connectionless communication protocol designed for low latency and high speed data transmission. It is often used in applications where performance is more important than reliability.

USCI Universal Serial Communication Interface is a module used in microcontrollers to support various communication protocols such as UART, SPI, and I2C.

XBee XBee is a brand of wireless communication modules widely used in IoT applications, supporting protocols like ZigBee.

ZigBee Zigbee is a wireless communication protocol designed for low power, low data rate, and short range communication, frequently used in IoT applications.

This page has been intentionally left blank.

Contents

Acknowledgments **i**

Abstract **iii**

Resumen **v**

1 Introduction **1**
1.1 Problem description 1
1.2 State of the art 3
1.2.1 Behavior classifiers developed with commercial data loggers . . 4
1.2.2 Custom made behavior classifiers 8
1.2.3 Commercial solutions 10
1.3 Research context and thesis goals 11
1.3.1 Precedent prototype solution 11
1.3.2 General objective and thesis scope 12
1.3.3 Contributions within the OVIS project 12
1.4 OVIS system description 13
1.5 Manuscript structure 14

2 Hardware **17**
2.1 Communication technology 17
2.1.1 IEEE 802.15.4 18
2.1.2 LoRaWAN 18
2.1.3 CAT M1 18
2.1.4 NB-IoT 19
2.2 Sensor types 21
2.2.1 Accelerometer 22
2.2.2 Gyroscope 22
2.2.3 Thermometer 22
2.2.4 GNSS 23
2.3 Power supply 23
2.3.1 Battery 23
2.3.2 Solar panels 24
2.4 Microcontroller 24

3 Embedded software **27**
3.1 Zephyr RTOS 27
3.1.1 General characteristics 27
3.1.2 Schedule: threads and events 29
3.2 OVIS application 30

3.2.1 Asset Tracker v2 . . . 30
3.2.2 Modules . . . 32

4 Signal processing **39**
4.1 RF selection . . . 39
4.2 Classifier on the PC . . . 40
4.2.1 External database description . . . 40
4.2.2 Behaviors to classify . . . 40
4.2.3 Dataset limitations . . . 41
4.2.4 Feature computation . . . 42
4.2.5 Feature selection . . . 43
4.2.6 Classification algorithm . . . 43
4.2.7 Results . . . 43
4.3 Classifier on the OVIS device . . . 44
4.3.1 C implementation verification . . . 44
4.3.2 Embedded software implementation . . . 44

5 OVIS cloud server **47**
5.1 Backend component . . . 48
5.1.1 Communications . . . 48
5.2 Database component . . . 50
5.3 User interface . . . 50
5.3.1 Login and home screen . . . 50
5.3.2 Map tab . . . 51
5.3.3 Other tabs . . . 53

6 Mechanical design **57**
6.1 Requirements . . . 57
6.1.1 Functionality and usability . . . 57
6.1.2 Durability . . . 58
6.1.3 Weight . . . 58
6.1.4 Color selection . . . 58
6.1.5 Collar adjustment . . . 59
6.1.6 Enclosure sizes . . . 59
6.2 Tests . . . 59
6.2.1 Hermeticity test . . . 60
6.2.2 First stage field tests . . . 60
6.2.3 Second stage field tests . . . 62
6.2.4 Third stage field tests . . . 65
6.3 Proposed design . . . 66
6.3.1 Sizes . . . 67
6.3.2 Quantity . . . 67
6.3.3 Technology and materials . . . 67
6.3.4 Assembly process . . . 67

7 Testing **69**
7.1 Autonomy . . . 69
7.1.1 PSM: low power NB-IoT configuration . . . 69
7.1.2 Autonomy measurement . . . 70
7.1.3 Consumption profile characterization . . . 70
7.1.4 Battery discharge characterization . . . 72
7.1.5 Estimated effective battery capacity . . . 73
7.1.6 Low transmission operating mode . . . 74

7.2 Other electronics characteristics . . . 75
7.2.1 Accuracy of the GNSS . . . 75
7.2.2 Performance of the communication OVIS device to OVIS cloud server . . . 77
7.3 Classification algorithm performance . . . 79
7.3.1 Behavior characterization through live observations . . . 80
7.3.2 Classification results and analysis . . . 81

8 Conclusion **83**
8.1 Main outcomes of this work . . . 83
8.1.1 State of the art comparison . . . 84
8.2 Technical limitations . . . 84
8.3 Future work . . . 84

A Parturition related behavior classifiers **87**

B Key OVIS modules description **91**
B.1 Application module . . . 91
B.1.1 Messages arrival: event subscription . . . 92
B.1.2 Messages handlers . . . 93
B.2 Data module . . . 93
B.2.1 Messages arrival: event subscription . . . 95
B.2.2 Messages handlers . . . 95
B.3 Sensor module . . . 97
B.3.1 Messages arrival: event subscription . . . 97
B.3.2 Messages handlers . . . 98
B.3.3 Accelerometer configuration and management . . . 98

C OVIS Device Assembly **101**

D Consumption model and autonomy extrapolation **105**
D.1 Simplified consumption model . . . 105
D.1.1 Identification of active and inactive periods . . . 105
D.1.2 Variables required to define the model . . . 106
D.1.3 Model 1: characterization based on period and duty cycle . . . 107
D.1.4 Model 2: fixed 50 s period and average inactive current . . . 108
D.1.5 Model selection . . . 108
D.2 Extrapolation from normal mode to low transmission mode . . . 109
D.2.1 Data volume calculation . . . 109
D.2.2 Cycle calculation for low transmission mode . . . 110
D.2.3 Estimated consumption and autonomy projection . . . 110

References **113**

Table list **118**

Figure list **121**

This page has been intentionally left blank.

Chapter 1

Introduction

1.1 Problem description

Livestock health and productivity are key concerns in agricultural systems, particularly under extensive livestock conditions such as those in Uruguay. The Uruguayan extensive system is characterized by animals grazing natural open field pastures in large paddocks, often with sheep and cattle grazing together. In such conditions, animals live freely for several weeks or even months, without infrastructure or human resources to monitor them [1].

The development of wearable devices and biosensors using Internet of Things (IoT) technologies is crucial for Precision Livestock Farming (PLF), offering tools that improve both animal welfare and production outcomes. Wearable devices are physical systems worn by an animal that can record, process, and transmit data, and may also perform specific actions such as triggering alarms, activating actuators, or interacting with external systems. Biosensors are a type of sensor that detect biological or chemical variables and generate an electrical signal in response. In many cases, wearable devices may incorporate biosensors or other sensors such as accelerometers, gyroscopes, thermometers, or Global Navigation Satellite Systems (GNSS) modules. These instruments support the detection of movement and behavior, stress levels, disease, and metabolic activity. As such, they can provide real time insights for early detection of health or management issues, supporting data driven decision making [2].

Artificial Intelligence (AI) and Machine Learning (ML) techniques are being explored to aid these technologies, transforming raw sensor readings into actionable knowledge. A typical PLF system may include primary sensors (e.g., accelerometers, temperature sensors) that capture physical magnitudes. These are then processed to estimate higher level features (e.g., rumination, restlessness) that can be used to monitor the animal condition. However, interpreting this data remains a challenge, especially in extensive systems where real time decisions are needed and infrastructure is limited.

In livestock, major pathologies include parasites, infectious diseases, foot affections such as foot rot, and reproductive conditions such as pregnancy ketosis, abortion, and mastitis [3]. Relevant physiological or behavioral parameters typically measured in PLF include motion, temperature, feeding activity, heart rate, and position. For instance, sudden decreases in grazing time or rumination, or a change in movement patterns, may signal issues like fever or lameness. Additionally, respiration and heart rate can support early detection of diseases such as pneumonia or milk fever [2]. Reproductive efficiency is another key driver of productivity. Technologies such as artificial

insemination aim to improve reproductive performance, making it essential to monitor estrous cycles and detect estrus (heat), the period when female bovines are sexually receptive [2]. These events can be inferred through changes in behavior patterns and activity levels, highlighting again the relevance of sensor monitoring.

In 2022, there were 6.2 million sheep in Uruguay [4]. One of the main limitations for sector development is reproductive efficiency, with high lamb mortality as a central problem [5–7]. Mortality is particularly severe during winter, which coincides with gestation and reduced pasture availability. This undernutrition during placenta and fetal growth results in lightweight lambs and low colostrum production. Weather conditions (temperature, rainfall) fluctuate widely year to year, complicating management planning. Furthermore, bonding between ewe and lamb must occur within a critical two hour window after birth. In many cases, especially for primiparous ewes and those with twins, this bond is not formed, increasing the risk of abandonment. Predation is another major factor. Producers report that foxes, feral dogs, wild boars, and caranchoes contribute significantly to neonatal losses, especially in the first days of life [8]. Overall, about 30% of lambs die after birth [5, 9], most within the first 72 hours [5], amounting to an estimated one million lambs lost in 2021 in Uruguay.

In this context, having online access to behavioral data from ewes could enable the implementation of strategies to improve reproductive efficiency, with a particular focus on reducing neonatal lamb mortality. Relevant indicators include time allocation to different activities, grazing budget, flock cohesion, reactions to handling (shearing, mating), and unusual behaviors. These metrics could allow researchers and producers to explore targeted interventions, such as providing farrowing pens, administering nutritional supplements, or offering assistance during lambing. In addition, signs of collective restlessness may be used to detect the presence of predators.

Despite these needs, there is still a lack of wearable and IoT based monitoring systems adapted to extensive livestock farming. To be viable in production settings, these solutions must be low cost, low maintenance, and comfortable for the animal. It should be considered that the market value of an individual animal (typically 60–100 USD per lamb), and the total cost of the system (devices, servers, communication infrastructure) must be proportionally small relative to flock value. Beyond affordability, the value provided by the system must justify its adoption (its added value must exceed its cost).

Maintenance requirements must also be minimal. In extensive systems, bringing animals in to change batteries or service devices is not always feasible and could negatively affect animal welfare. A system that requires frequent physical intervention defeats the purpose of long term remote monitoring. Devices must also avoid influencing animal behavior. If a collar causes discomfort or stress, the animal may try to remove it, potentially injuring itself and generating biased data. These conditions differ from those of short term research experiments, which have more human and material resources, and are typically conducted in controlled environments.

Online systems that can monitor and predict key behaviors such as lambing, estrus, or predator induced stress could have a major impact on both productivity and animal welfare. While this long term goal is ambitious, it provides a useful direction for developing research platforms. In practice, translating such information into farming decisions requires robust sensing systems, reliable data acquisition, and further validation.

In this context, the problem addressed in this thesis focuses on first steps toward that broader vision: the development of a system capable of acquiring, processing, and transmitting behavioral and positional sensor data with sufficient autonomy and without requiring frequent maintenance. Moreover, the signal processing explored here should be understood as a proof of concept, intended to demonstrate the potential

of the proposed architecture and to establish a solid foundation for future work on behavior detection and prediction.

1.2 State of the art

Analyzing the state of the art reveals that existing on-animal monitoring systems for sheep fall broadly into two categories: commercial data loggers and custom-made research devices. Commercial devices are relatively few and often provide limited flexibility, with their analysis summarized in Section 1.2.3. In contrast, custom-made research systems constitute the majority of published work and offer deeper insight into the design choices and constraints that shape current development. For this reason, the detailed classification presented in this chapter focuses primarily on these research systems, examining their sensing capacities, data handling strategies, and processing approaches. Additionally, systems specifically designed for lambing or parturition monitoring (an especially important problem in Uruguay due to the high impact of lamb survival on productivity) are reviewed separately in the Appendix. These systems represent a natural next step for future research building upon the platform developed in this thesis. A comprehensive review of on-animal systems that monitor sheep behavior is presented in [10], with another review focusing specifically on sheep behavior studied using accelerometer data in [11]. Understanding the variables that characterize these systems is essential for evaluating their functionality and applicability.

This work focuses primarily on systems that use accelerometers and, in some cases, GNSS sensors. This selection responds to practical constraints observed in extensive livestock farming, where accelerometers provide a balance between information value and energy efficiency, and GNSS offers location data crucial for behavioral interpretation in open environments. Although some studies explore the use of other sensors, such as gyroscopes, magnetometers, or temperature sensors, these are typically less common. The selection criteria also reflect the objective of designing a scalable, low maintenance solution suitable for production settings.

Several technological and methodological dimensions define the landscape of wearable monitoring systems. These include the system's aim (e.g., health monitoring, reproductive efficiency, behavior mapping), device location and harness design, physical characteristics like weight and size, and the use of microcontrollers. Memory availability (whether internal to the microcontroller or via external modules such as SD cards) is another key factor, especially when raw data is logged rather than transmitted. Furthermore, system autonomy should be considered. Other technical aspects include the type of communication used (e.g., radio frequency or cellular), whether the system transmits raw or processed data, and the types and sampling rates of sensors employed.

Another dimension of these systems is how they process and interpret the collected data. At a basic level, sensor generate continuous raw signals that must be translated into higher level information about the animal's behavior. This involves steps such as signal segmentation, feature extraction, and classification. These signal processing tasks are essential to convert raw measurements into meaningful behaviors like grazing, resting, or walking. Traditionally, these methods have been highly heuristic, which are often sensitive to noise and systems that rely on handcrafted rules and features tailored to specific setups. However, recent advances in ML techniques offer more flexible and robust alternatives. ML approaches can automatically identify patterns in sensor data and have shown great promise in improving sensor data processing.

Finally, two attributes distinguish system capabilities: whether classification is performed directly on the device (on-animal classification), and whether information

is made available online (online classification) [1].

1.2.1 Behavior classifiers developed with commercial data loggers

Most of the studied systems use commercial data logger devices included in the equipment worn by the animal. A major advantage of these systems is their typically low weight and small size (usually a few grams), which facilitates their use in live animals without significantly affecting behavior. However, they often lack the ability to transmit raw data, as information is stored in external memory (an SD card) for later retrieval. Additionally, sensor settings are generally not configurable because they are controlled by proprietary software, and behavioral classification is performed off-animal, typically on a Personal Computer (PC) or server after data is downloaded. These limitations increase maintenance needs, since a researcher or user must be present and physically access the animal to retrieve data or service the device. This is not always feasible in extensive livestock contexts. Furthermore, because data processing occurs after retrieval, these systems cannot provide producers with timely information to support management decisions. The following paragraphs describe specific solutions and highlight their main characteristics. Table 1.1 summarizes the most relevant articles under this description.

[12] presents a collar type device designed to discriminate between active and inactive behaviors in sheep. Although the study considers only two behavioral categories, it is notable for its simple design and reasonable performance. The sensor weighs 7.8 g and measures $3.5 \times 2.0 \times 3.5$ cm^3. The device includes an accelerometer, a memory of 128 kB, and a reported autonomy of 22.5 to 45 days. The experiment was carried out with nine multiparous Texel ewes. The two behavioral categories were defined by combining six observed behaviors into either active (grazing, standing, standing/ruminating, walking) or inactive (lying, lying/ruminating) groups. A single feature related to signal energy was calculated over a 25 s window to classify activity. The system achieved an overall accuracy of 80% for the active class and 74.6% for the inactive class. These results highlight the practical advantage of collapsing the behaviors into two broad categories to improve classification performance in applied settings.

[13] develops a wearable device integrated into a halter. The system includes an onboard data logger, a three-axis accelerometer, and a Lithium Polymer (LiPo) battery. Data are downloaded via Bluetooth (BT) once a day. The accelerometer was configured to operate at sampling frequencies of 5 Hz, 10 Hz, or 25 Hz, and experiments were conducted on 10 South African Merino sheep in a 30 square meter paddock. Features were computed using time windows of 3 s, 5 s, and 10 s. For each window size, the most relevant features were selected using a Random Forest (RF) algorithm. The top features included the mean values of the X-axis, Z-axis, inclination, pitch, and movement variation. A Decision Tree (DT) classifier was then trained to distinguish five behaviors: grazing, lying, running, standing, and walking. For the 5 s window, the system achieved an accuracy of 85%, with additional reported performance metrics: precision of 86.2%, sensitivity of 85.3%, specificity of 96.3%, and a Kappa value of 0.80. This solution is notable as one of the first complete sheep behavior classifiers with solid results across multiple metrics.

[14] designs a system for gait and posture discrimination based on a three-axis accelerometer mounted on the hind leg of the sheep. The study was conducted with 13 mature Pramenka sheep (10 ewes and 3 rams), clinically healthy and free from locomo-

[1] In this work, on-animal classification refers to behavior classification performed directly on the animal worn device. Whereas, online classification refers to the ability to transmit raw sensor data for live, server based classification, or to transmit already classified behaviors so that they become available on a computer or server with minimal delay.

tor disorders. The device includes an accelerometer, a 64 kB memory, and a reported autonomy of 7 days at a 100 Hz acquisition rate. The device weighs 18 g and measures $5.8 \times 3.3 \times 2.3$ cm^3. Two classifiers were developed, one for gait (walking, trotting, galloping) and another for posture (standing, lying), using Discriminant Analysis (DA) and Canonical Correlation Analysis (CCA) for classification, and Forward Stepwise Analysis (FSA) for feature selection. Both classifiers used a 3 s window. Gait data were acquired at 33 Hz, and posture data at 5 Hz. The gait classifier achieved a total accuracy of 89.6%. Most misclassifications occurred between trotting and the other two gaits; walking and galloping, which were never confused with each other. The posture classifier, based on RFAC values[2], achieved 99%. This solution demonstrates strong performance using relatively simple statistical methods and is notable for its clarity in distinguishing both posture and locomotion behaviors in sheep.

[15] focuses on evaluating different classification techniques for sheep and goat behavior recognition system. The dataset includes 2 Texel sheep and 4 Dutch white goats. The sensing device is located on the neck and acquires accelerometer data at 200 Hz, along with gyroscope and magnetometer measurements. Data can either be stored on an SD card or transmitted raw via radio frequency. Although autonomy is not explicitly stated, the collar was used in experiments that lasted at least one day. The study compares several classifiers and concludes that a Deep Neural Network (DNN) performs best when classifying five behaviors, grazing, standing, walking, lying, and ruminating, achieving 95% overall accuracy. The precision, sensitivity, and F1-score are also reported per class, with consistent performance across behaviors (average precision 94%, sensitivity 95%, F1-score 94%). Feature selection is performed using the Relief Algorithm (RA), identifying three key features. This work is also notable for providing a public dataset containing synchronized timestamped sensor readings (accelerometer, gyroscope, and magnetometer on all three axes) alongside behavior annotations, temperature, and pressure. This dataset served as the training base for the first algorithm developed in this present work.

[16] proposes an ear tag device for general behavior classification. This location is considered great for extensive livestock contexts, as the sensor could potentially be integrated into an identification tag, commonly used in Uruguayan production systems. The sensor includes a three-axis accelerometer, weighs 17.7 g, and measures $5.0 \times 2.5 \times 1.2$cm^3. Accelerometer data is sampled at 12 Hz, with an internal memory of 8 GB. The device reports an autonomy of one day at a sampling rate of 25 Hz. The study used 5 Merino crossed with Poll Dorset ewes and classified three behaviors: grazing, standing, and walking, using Quadratic Discriminant Analysis (QDA), with features selected with RF from a 1 s time window. QDA achieved an overall accuracy of 97.3% in the best model. Class precision ranged from 91.9% (walking) to 98.2% (grazing), sensitivity from 96.8% (walking) to 98.6% (standing), and F1-scores from 93.4% to 98.4%.

[17] presents a general behavior classifier using an ear tag device. The system attempts to classify activities using three different ethogram[3] categories. The first ethogram includes four specific behaviors: lying, standing, walking, and grazing. The second ethogram groups behaviors into two classes, active and inactive, while the third separates posture into upright and prostrate categories. This separation enables classification at different levels of abstraction. The experiments were conducted using 16 mature Merino ewes. The device contains a three-axis accelerometer, weighs 11 g, and measures $2.3 \times 3.25 \times 0.76$ cm^3. The accelerometer sampling rate is 12.5 Hz, with

[2] RFAC (Ratio of Forces Around the Center) is a feature based on accelerometer data used to distinguish posture.

[3] An ethogram is a catalog of defined animal behaviors used for systematic observation and classification.

internal memory of 512 MB and a manufacturer reported autonomy of 30 days.

The first ethogram uses a Support Vector Machine (SVM) with 19 features, no feature selection, and a 10 s window, achieving 76.9% overall accuracy. For this model, precision ranged from 71.9% (walking) to 82.5% (standing), and sensitivity from 66.3% to 83.6%. The second ethogram uses a Classification and Regression Tree (CART) algorithm, also with 19 features and no feature selection, and a window size of 30 s, achieving 98.1% accuracy. Finally, the third ethogram applies Linear Discriminant Analysis (LDA) using three features selected by RF over a 30 s window, achieving 90.6% accuracy.

[18] focuses on the effects of calculating features with a moving window to classify four behaviors: grazing, standing, walking, and lying. Similarly, [19] explores the use of different window sizes simultaneously that is, classifying behavior using features extracted from windows of various durations. Studies analyzing the effect of window size on classification performance are extremely valuable, as they provide practical tools for selecting an appropriate window length for a given application.

The experiment in [18] used 5 Merino crossed with Poll Dorset ewes and evaluated data from sensors deployed on the ear, leg, and collar. The accelerometer device and classification model (QDA with RF feature selection) were the same as the ones used in [16]. The classifier was tested using moving windows of 3, 5, and 10 seconds. The overall performance did not significantly differ across window sizes, but the 10 s window showed slightly higher classification rates for certain behaviors.

Prediction accuracies for the ear worn sensor ranged from 86% to 95%, for the collar worn sensor from 67% to 88%, and for the leg worn sensor from 48% to 94%, depending on which behavior and which sheep were analyzed. The precision, sensitivity, and class accuracy were reported, highlighting notable between-animal variation, particularly for standing and grazing behaviors. The study underscores the importance of sensor positioning for robust classification in online livestock monitoring systems.

[19] develops a collar device located on the sheep's neck. The device includes a 4 GB accelerometer data logger with a weight of 19 g and a volume of $4.6 \times 3.3 \times 1.5$ cm^3. The accelerometer works at 30 Hz, and the device reports an autonomy of 25 days. The experiment was conducted with 17 Merino sheep over two days. The study classifies four behaviors (grazing, ruminating, walking, and standing) and explores the effect of using features calculated with multiple window sizes (2 s, 10 s, and 15 s) simultaneously. The RF classifier was found to outperform SVM and LDA. With only the top three features selected by RF, the system achieved a 94.6% overall accuracy. Whereas, with nine features, the model reached 98.6% accuracy and behavior metrics exceeded 97% in precision, sensitivity, and F1-score. Walking class presented the greatest challenge to detect.

[20] proposes a system to classify four behaviors: grazing, walking, scratching, and inactivity (defined as standing and resting). The device is placed on the animal's neck and includes an 8 MB accelerometer data logger. It weighs 8 g, measures $3.6 \times 2.7 \times 1.0$ cm^3, and integrates an nRF52 SoC microcontroller from Nordic Semiconductor, although BT communication is not utilized. The accelerometer collects data at 12.5 Hz. Experiments were conducted with 8 Hebridean ewes in a 1500 m^2 paddock over two days.

17 features were extracted using a 5 s sliding window and then reduced to nine using Principal Component Analysis (PCA) and correlation filtering. A RF classifier achieved an overall accuracy of 99.43% and a Kappa value of 98.66%. Class sensitivity ranged from 98.26% to 99.87%, specificity from 99.60% to 99.92%, and F1-scores from 91.53% to 99.90%. The results indicate that even closely related behaviors, such as grazing and walking, can be robustly separated using this setup.

[21] develops a general behavior classifier using the same device as in [19], mounted

on a neck collar. The experiment was conducted with 6 Perendale ewe lambs for classifier training, and applied to 10 additional lambs for field validation. Raw accelerometer data was recorded at 30 Hz and aggregated in 5 s windows. A RF algorithm was used to classify three behaviors (grazing, resting, and walking), where resting was defined as a combination of standing and lying. No crafted features were used (i.e, classification relied on raw data only). The classifier achieved an overall accuracy of 89.6% and out-of-bag misclassification[4] rate of 10.4%. Reported class performance metrics ranged from 72% to 96%, grazing showing high sensitivity (94%), resting with high precision and specificity (96%), and walking with the lowest scores overall. These results indicate that the system performs best at detecting resting and grazing, while walking remains the most challenging behavior to classify.

[22] uses the same neck mounted device as [20], incorporating a Nordic Semiconductor nRF52 SoC and an 8 MB memory. The accelerometer samples at 12.5 Hz, but the BT capability of the SoC is not used. The study involved 12 sheep (Texel, Welsh Mountain, and Lleyn breeds) and classifies three behaviors: grazing, activity, and inactivity (including standing and resting). A Convolutional Neural Network (CNN) classifier was trained using 52 features calculated with a 5 s window, without a feature selection algorithm. The model achieved a general accuracy of 96.59%, with 94.95% sensitivity, 96.74% specificity, and a macro F1-score of 95.42%. Although this approach is designed for deferred processing, computational demands of using a CNN and 52 features make it unsuitable for direct implementation on embedded systems with limited resources.

[23] aims to classify both postures and behaviors using a neck mounted device weighing 100 g. The device includes a 512 MB data logger and a three-axis accelerometer sampling at 50 Hz. A RF algorithm is used both for classification and feature selection. Three features, computed using a 6 s window, were selected to train the model. Postures classified were standing and lying, with an accuracy of 83.7%. Behaviors classified included grazing, ruminating, walking, and inactivity, with an overall accuracy of 70.9%. The precision and sensitivity ranged from 67% to 88% depending on the class. This study is characterized for its extensive livestock approach, conducting experiments in a commercial flock of over 100 animals.

[24] implements a system using two devices to classify nine complex behaviors: sitting, standing/grazing, standing, standing/ruminating, sitting/ruminating, walking, standing/walking/grazing, walking/grazing, and standing/walking. This classification reflects the reality that behaviors often overlap (sheep may graze while walking or standing) making it difficult to separate them. The authors hypothesize that combined behaviors such as grazing/walking and grazing/standing generate distinguishable sensor signals. The system uses a mouth mounted device identical to those in [19] and [21], and an ear mounted device as in [17]. 7 mature Merino ewes were used in the study. A RF classifier was applied without feature selection, using features extracted over a 10 s window. Data augmentation was performed using the Synthetic Minority Over sampling Technique (SMOTE) algorithm. The system achieved a 72.4% overall accuracy, with average sensitivity of 74.7% and a macro F1-score of 66.3%.

Summary

Systems based on commercial data logger generally stand out for their compact design (typically weighing less than 100 g and occupying under 25 cm^3) making them suitable for wearable applications. While some devices achieve autonomies longer than 20 days, this is usually limited to systems that store data locally and require manual

[4] Out-of-bag misclassification is an internal estimate of the prediction error in RF models, calculated during training without needing a separate test set.

retrieval. In contrast, those that support online data transmission tend to have shorter operating times (often no more than a day or two). Most studies classify between 2 and 4 behaviors, with only a few addressing more complex classifications (i.e., 9 behaviors). Reported accuracies are generally above 70%, and frequently exceed 90% when fewer behaviors are involved. Overall, these systems reveal a clear trade-off between operational simplicity, battery life, and classification complexity, and typically rely on off-animal and deferred data processing.

1.2.2 Custom made behavior classifiers

Custom made behavior classifiers are an alternative to systems based on commercial data loggers. Custom made behavior classifiers are developed with greater flexibility, enabling on-animal classification and online classification. This reduces the need for frequent device retrieval and lowers maintenance demands, which are relevant in extensive livestock systems. Custom built solutions also allow more control over sensor configuration and data handling. The following paragraphs present several custom made systems and Table 1.2 outlines some key systems.

[25] designs a general behavior classifier with a collar device weighing 281 g and measuring $14.6 \times 8 \times 6.5$ cm^3. The device features an MSP430FR5739 from Texas Instruments, an accelerometer sampling at 100 Hz, and 2 GB of external memory. Although autonomy is not explicitly reported, the collar is functional for at least one day based on experiment duration. Five behaviors (lying, standing, walking, running, and grazing) are classified using QDA. Ten features, calculated with a 5.12 s window, are selected using Sequential Forward Selection (SFS). Classification was performed with deferred processing using data collected from five Dohne Merino sheep over three days. The best classifier achieved 89.7% overall accuracy.

[26] uses the same device as [25] and adds new functionalities: radio frequency communication and GNSS acquisition. Also, the same group of 5 Dohne Merino sheep used in [25] was used in [26]. Moreover, LDA is employed to classify the same five behaviors. Twelve features are calculated with a 5.12 s window and selected using SFS. This study is notable for implementing on-animal classification and online classification (online transmission of the behavioral state rather than the raw data). The device produces updates every 5.3 s. The system achieved 82.4% on-animal classification accuracy. Class performance metrics show that lying down, standing, and running are classified with high precision, while grazing presents the greatest challenge, with 38.5% of grazing instances misclassified as lying.

[27] focuses on classifying eating behaviors with a device located in the animal's mouth. The system includes a three-axis accelerometer with a sampling frequency of 62.5 Hz. This system logs data to an SD card, and transmits raw data online via XBee. Although autonomy is not explicitly reported, the device was operational for at least one day based on the experiment's duration. The classifier distinguishes between grazing, ruminating, and resting using Canonical Discriminant Analysis (CDA), and Stepwise Discriminant Analysis (SDA), which selects seven features calculated with a 60 s window. The model achieved a general accuracy of 93%.

[28] investigates the effects of sensor frequency, position, and window size on behavior classification. The study develops a custom device weighing 4 g and measuring $3.16 \times 3.5 \times 0.9$ cm^3, featuring an Intel Quark SE C1000 microcontroller with a 384 MB non-volatile memory. Although the device supports radio frequency communication, this functionality is not used in the experiments. Accelerometer data is sampled at 32 Hz, and the device is positioned on the neck or ear. Autonomy is not explicitly stated, but the system functioned for at least one day based on experiment duration. The study involved eight sheep and classified three behaviors, lying, standing, and

walking, using a RF algorithm. No feature selection algorithm was performed, 44 features were extracted using a 7 s window (this window size showed the best results). The system achieved 95% overall accuracy. This work is notable for analyzing how window size, sampling rate, and device location affect classification, and for developing a compact embedded platform that includes a Pattern Matching Engine (PME), which facilitates on-device classification.

[29] focuses on selecting algorithms and features for a general behavior classifier. The device is the same as in [28] and is worn around an animal's neck or ear. The accelerometer is sampled at 16 Hz. 6 sheep of Texel cross (3), Suffolk cross (1), and Mule (2) breeds were used. Three behaviors (grazing, ruminating, and other) are classified using RF. The algorithm uses 39 features, which were selected with RA and calculated with a 7 s window. The system achieves 92% accuracy using collar data. For grazing, precision, sensitivity, F1-score, and specificity were 96%, 93%, 95%, and 98%; for ruminating, 92%, 87%, 89%, and 97%; and for other, 89%, 95%, 92%, and 91%. Collar placement slightly outperformed ear data, with both showing robust performance.

[30] uses the same device as [28] and [29], with added radio frequency communication. The study reports an autonomy of 2.4 days. The accelerometer is sampled at 16 Hz, and data is collected from 6 adult ewes (Texel, Suffolk, and Mule crosses). Three behaviors (lying, standing, walking) are classified using K-means and KNN algorithms, based on 20 features extracted from 7 s windows. The system achieves 85.18% accuracy, with on-animal classification and online classification (online transmission of results, but excluding raw data). Reported metrics for the best configuration include macro precision of 86%, sensitivity of 85%, and F1-score of 85%.

[31] develops a grazing behavior classifier with a collar device located around the animal's neck. As was noted for [12], although only two behaviors are classified, this work stands out due to its simple design and excellent results. The device uses an ARM Cortex-M0 and includes an SD memory card. While autonomy is not explicitly reported, the collar is functional for at least one day based on the experiment duration. Grazing and non-grazing are distinguished using LDA with 55 features and no feature selection. Features are calculated with a 10 s window. The system achieves a 98.65% accuracy. Reported metrics at this window size include 98.52% sensitivity, 98.92% precision, 98.79% specificity, and 98.3% F1-score for the grazing class.

[32] designs a general behavior classifier with a collar device located around the animal's neck. This study is notable for its on-animal classification and online classification (online result transmission, as well as raw data transmission). The device features a Texas Instruments CC1110 SoC microcontroller, uses radio frequency for communication, and has 32 kB of memory. The accelerometer is sampled at 16 Hz. Five behaviors are classified using a DT algorithm, and 11 features are selected using the Fselector R package. The study reports an autonomy of 134 days[5]. The study is also notable for the long autonomy it reports, making it the closest antecedent to the type of system targeted in this work. Performance metrics include an accuracy of 91.78%, a micro-average[6] F1-score of 91.78%, a macro-average F1-score of 70.86%, and a K-category correlation coefficient[7] of 0.7571.

[33] studies the effect of sensor position on behavior classification, tested on 3 adult Sarda dairy ewes. The device stores data on an SD card and samples acceleration at

[5]This autonomy is based on a different system prototype presented by the same researchers.

[6]Micro-average gives more weight to frequent classes; macro-average treats all classes equally.

[7]The K-category correlation coefficient is a generalization of the Kappa value for class classification, measuring agreement between predicted and actual labels beyond chance. Values close to 1 indicate strong agreement.

62.5 Hz. Although autonomy is not explicitly reported, the collar is functional for at least one day based on experiment duration. CDA is used to classify grazing, ruminating, and other behaviors with 12 features and no feature selection, computed using 300 s windows. Performance metrics reported for the collar position at 300 s epoch include, for grazing: 96.9% precision, 98.6% sensitivity, 93.9% specificity, and 96.9% F1-score; for ruminating: 81.8%, 75.%, 98.9%, and 70.6% respectively; and for other activities: 81.0%, 77.1%, 94.3%, and 79.0% respectively. Overall accuracy at this configuration reaches 90% with a Kappa value of 0.8.

[34] develops a behavior classification system with a device located around the animal's neck. The system stores data on an SD card and samples acceleration at 20 Hz. Five behaviors (lying, standing, walking, running, and grazing) are classified using an Extreme Learning Machine (ELM) algorithm with 48 features and no feature selection. Features are computed using a 3 s window. The study was conducted on 20 sheep of Kazakh breed. The best model achieved an accuracy of 84.8%. Class performance ranged from 76.5% to 94.6% in precision, and from 74.3% to 94.6% in sensitivity, with an overall macro F1-score of 83.3%.

Summary

Custom made systems are generally bulkier than systems that use commercial data loggers, with weights often exceeding 100 g and volumes greater than 25 cm^3, although some recent solutions achieve more compact designs. Although autonomies range from one day to over 100 days, most systems provide limited operating time, reflecting designs that are not yet oriented toward real production environments. Many devices include embedded microcontrollers (e.g., ARM Cortex or MSP430 families), and several implement online classification, offering functionalities not present in most commercial loggers. The number of classified behaviors typically ranges from 2 to 5. Reported classification performance is generally high, with most systems achieving over 85% accuracy. Although results are strong, studies that implement on-animal classification tend to report slightly lower accuracies than those relying on server based processing, reflecting the computational and energy constraints of embedded devices. These systems demonstrate the potential for developing tailored, low maintenance solutions that support data driven decisions in extensive livestock systems, though they involve greater design and deployment complexity. Also, reported results suggest that systems performing on-animal classification tend to exhibit slightly lower accuracies than those relying on server based processing, probably due to the computational and energy constraints inherent to embedded devices.

1.2.3 Commercial solutions

The transition from experimental prototypes to deployable solutions has led to commercial livestock monitoring systems. Although primarily developed for cattle, some of these platforms have been successfully applied to sheep, especially in extensive grazing environments.

Nofence, [35], a Norwegian company founded in 2011, offers a solar powered Global Positioning System (GPS) collar for cattle, goats, and sheep. The collar enables virtual fencing via audio warnings and mild electrical pulses, online tracking, and boundary alerts through a mobile app. The system includes a three-axis accelerometer, GPS receiver, and cellular connectivity Long Term Evolution (LTE) and BT.

SheepIT (presented in [32]) is a commercial IoT solution targeted at vineyards that integrates posture detection and behavioral conditioning using accelerometers and ultrasound sensors. The collar monitors posture online and emits stimuli when sheep

attempt to eat where they should not. Pilot trials with industrial scale deployment demonstrated classification accuracy above 91% and no adverse animal welfare effects.

Other commercial collars, such as those from **Digitanimal**, [36], offer GPS tracking and basic motion alerts but typically lack advanced behavior classification beyond activity level detection.

Summary

Overall, commercial solutions are mostly limited to boundary breaches, posture alerts, or simple activity metrics. This gap presents an opportunity to incorporate behavior analytics into solutions that can operate effectively in extensive livestock management settings.

1.3 Research context and thesis goals

Based on previous experiences [37], [38], an interdisciplinary research group was conformed at the Universidad de la República in 2020, led by Julián Oreggioni (Facultad de Ingeniería) and Rodolfo Ungerfeld (Facultad de Veterinaria). The group aims to contribute solutions to research on sheep behavior and welfare under extensive production conditions through the development of information and communication technology based systems. In a first stage, an initial system prototype was developed (see Section 1.3.1), yielding promising results [39]. In a second stage, within the framework of the project "Sistema electrónico para la caracterización del comportamiento de ovino" (Comisión Sectorial de Investigación Científica, Universidad de la República, Research and Development project), the undergraduate thesis in Computer Engineering by Hernán Cardoso [40], as well as the present master's thesis, were developed and funded. In this context, the OVIS system, which is presented throughout the present document, was conceived.

1.3.1 Precedent prototype solution

Before presenting the OVIS system, it is also essential to acknowledge a direct technical antecedent: a precedent prototype developed in 2021 as part of an Electrical Engineering Final Project (from Instituto de Ingeniería Eléctrica, Facultad de Ingeniería, Universidad de la República). This earlier device, established foundational design concepts and provided practical insights that informed the system reported in this manuscript.

This Project titled *Sistema para la caracterización de la dinámica espacial en ovinos, DEO* (System to characterize the spatial dynamics of sheep) involved the design, manufacture, and testing of a prototype collar device for monitoring sheep behavior and was later presented in [39].

The project included a preliminary evaluation of key technologies, such as wireless communication and energy management. The device integrated a Texas Instruments MSP-EXP432P401R microcontroller, a Bosch Sensortec BMI160 three-axis accelerometer, and a Quectel BG96 Narrowband-IoT (NB-IoT) modem with GNSS capabilities. It supported two operating modes: validation mode (VM), used for validating activity classification algorithms, and research mode (RM), designed for long term animal trials. In VM, accelerometer data, behavioral status (running, walking, standing, or head down), and GPS location were transmitted to the Central System every 20 seconds, resulting in a battery life of 51 hours. In RM, only status and location were sent every 2 minutes or more, extending battery life to over 10 days.

The system performed on-animal classification and online classification using a LDA algorithm, with preliminary tests showing 88% accuracy in classifying behavioral states. However, several limitations were identified: (i) design issues in the power subsystem, (ii) high energy consumption, acceptable for veterinary trials (2 to 10 days autonomy) but inadequate for livestock production, (iii) a fragile enclosure, and (iv) limited Central System and user interface functionality. Furthermore, in late 2021, Texas Instruments discontinued the MSP-EXP432P401R microcontroller, prompting the need to redesign the hardware base for future development. The present manuscript describes a research platform representing the next generation of DEO initial prototype.

1.3.2 General objective and thesis scope

The general objective of this thesis is to design, implement, and evaluate a research platform for long-term monitoring of sheep behavior, capable of performing on-animal processing and providing online behavioral data, while operating under the energy constraints of extensive livestock systems.

The scope of this thesis is defined by the design and validation of the OVIS system as an integrated research platform, composed of a non-invasive wearable collar and a cloud based server. The system should perform on-animal behavior classification and provide periodic online access to processed data, reducing communication requirements and supporting scalability.

Energy efficiency should be a central design criterion, guiding hardware and software decisions to enable long-term operation under realistic field conditions. OVIS system should be conceived as a research platform rather than a commercial solution, aimed at supporting experimentation and validation of embedded sensing, on-device processing, and connectivity strategies for animal behavior monitoring.

1.3.3 Contributions within the OVIS project

The development of OVIS system has been progressively documented in previous articles and has involved the coordinated work of a multidisciplinary team including electronics, embedded systems, mechanical design, signal processing, computing, and animal science[8]. An initial version of OVIS system focused on the device fabrication

[8]Early research on behavioral sensing and prototype experimentation was carried out by Victoria Campiotti, Nicolás Finozzi, and Juan Irazoqui (Instituto de Ingeniería Eléctrica, Facultad de Ingeniería, Universidad de la República), whose contributions established the initial foundations for on-animal behavioral monitoring (see Section 1.3.1).

The first end-to-end stage of the OVIS system engineering development reported in this thesis (including hardware design, electronic integration, embedded software, signal processing, cloud server, enclosure development, testing, field trials and system implementation) was led by Varinia Cabrera under the supervision of Julián Oreggioni, and is currently being further developed and maintained by Rocío Cabral, also under his supervision.

Embedded software was mainly developed by Varinia Cabrera under Julián Oreggioni supervision. This development also benefited from the participation of Andrea Delbuggio (all four contributors are affiliated with the Departamento de Electrónica, Instituto de Ingeniería Eléctrica, Facultad de Ingeniería, Universidad de la República).

Signal processing development, coding, and analysis was performed by Varinia Cabrera under the guidance of Álvaro Gómez (Departamento de Señales, Instituto de Ingeniería Eléctrica, Facultad de Ingeniería, Universidad de la República).

Cloud infrastructure, backend services, and the web interface were developed by Hernán Cardoso under the supervision of Julián Oreggioni and Martín Pedemonte (Instituto de Computación, Facultad de Ingeniería, Universidad de la República), within the framework of his undergraduate Computer Engineer thesis. In addition, Varinia Cabrera contributed to the

was presented at the EMBC (IEEE Engineering in Medicine and Biology Society) conference in 2023, [41]. Further improvements and preliminary applications were explored in the 2023 CAFE (IEEE Conference On AgriFood Electronics) article [42], while a more comprehensive version was detailed in [43] for 2024 TAFE (IEEE Transactions on AgriFood Electronics). This thesis builds on these contributions, extending the OVIS system's capabilities.

1.4 OVIS system description

The system (see Fig. 1.1) consists of a wearable collar device capable of collecting movement and location data, and a cloud server that stores the data and provides a web based user interface accessible from a PC, cellphone, or similar device. It uses the Message Queuing Telemetry Transport (MQTT) protocol and NB-IoT communication.

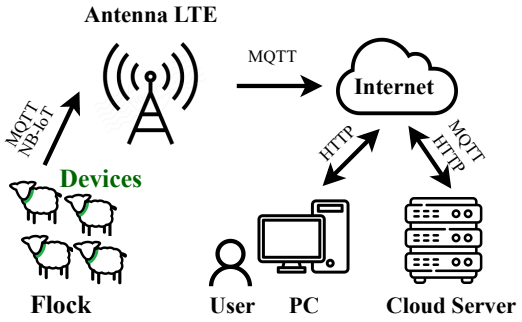


Figure 1.1: Functional diagram of the system. Figure adapted from [43].

The device sends encoded accelerometer data collected at 25 Hz, behavioral states computed every 2.4 s, battery level, and LTE signal strength to the cloud server every 50 s. When enabled, it also includes location data acquired every 10 s to 30 s. All reporting intervals are configurable and can be reduced to meet specific application needs. The system supports experiments with animals and is capable of handling large volumes of data, including collection, processing, and online transmission.

In practice, the use of the OVIS device's GNSS capabilities is conditional and depends on the requirements of each experiment. As quantified in Section 7.2.1, the accuracy of the GNSS data is generally low for the device's intended use, which makes its practical usefulness application dependent. Furthermore, enabling periodic location data has a significant impact on energy consumption (see Section 7.1.2). For this reason, the system supports operation both with and without GNSS, and whenever

specification and testing of the cloud server components.

The design requirements of the OVIS device enclosure were led by Varinia Cabrera and defined by the entire OVIS research team (engineers and veterinarians) in collaboration with the DVL group company, with guidance from its Director, Diego Fraga. The mechanical design and prototype fabrication were carried out by DVL. Enclosure testing was led by Varinia Cabrera and conducted jointly with the entire OVIS research team.

Field trials, animal handling, and expertise on sheep behavior were led by Prof. Rodolfo Ungerfeld (Facultad de Veterinaria, Universidad de la República), ensuring the scientific and practical relevance of the system. This part of the work involved the direct participation of Varinia Cabrera and Julián Oreggioni, and, on specific occasions, other members of the OVIS research team.

GNSS is used, this is explicitly stated in the document. This flexibility is essential for balancing autonomy and data needs.

The system is designed for scalability. Currently, data processing occurs locally on each collar, enabling online behavior classification. This functionality could be extended to include additional on-device processing using new sensors or the same one. Moreover, integrating data from multiple collars with external information sources (e.g., weather or satellite imagery) would enable advanced server based processing.

The platform facilitates engineering research, such as on-device sheep behavior classification, while also serving as a novel tool for animal production studies.

1.5 Manuscript structure

This document reports the steps undertaken for the conception, design, implementation, fabrication, and testing of the OVIS system. The remainder of the thesis is organized as follows. Chapter 2 presents the hardware of the OVIS device, including communication technologies, sensing components, power supply options, and the selected microcontroller. Chapter 3 details the embedded software, describing the Zephyr Real Time Operating System (RTOS), the application structure, and the custom modules that implement data acquisition, on-device processing, and communication. Chapter 4 focuses on the signal processing, including external database preparation, feature computation and selection, and the behavioral classifier. Chapter 5 describes the OVIS cloud server, outlining its overall architecture and the main components that support data management, communication, and user interaction. Chapter 6 presents the mechanical design of the collar enclosure, including functional requirements, tests, and the proposed final design. Chapter 7 reports the testing and performance evaluation of the system, including electronics behavior, GNSS accuracy, communication accuracy, autonomy for the OVIS device and classifier performance. Finally, Chapter 8 provides concluding remarks and outlines future work building upon the OVIS system.

	[13]	[15]	[17]	[18]	[19]	[21]	[22]	[23]	[24]
General									
Application	GBC[I]	GBC	GBC	Window effect	Window effect	GBC	GBC	Posture and GBC	GBC
Custom made	No	No	No	No	No	No	No	No	No
Location	Mouth	Neck	Ear	Ear	Neck	Neck	Neck	Neck	Mouth and Ear
Online data	Yes	Yes	No	No	No	No	No	No	No
Weight (g)	8	-	11	17.7	19	19	8	100	19 | 11
Size (cm^3)	3.0×2.0×0.7	-	2.3x3.25x0.76	5.0×2.5×1.2	4.6x3.3x1.5	4.6x3.3x1.5	3.6x2.7x1.0	-	4.6x3.3x1.5 | 2.3x3.25x0.76
Technologies									
Microcontroller	ARM processor	-	-	-	-	-	nRF52 SOC	-	-
Communication	BT	radio frequency	No	No	Not used	Not used	No	No	Not used | No
Accel. rate (Hz)	5, 10, 25	200	12.5	12	30	30	12.5	50	30 | 25
GNSS rate	No	No	No	No	No	No	No	No	No
Memory	32 MB	SD	512 MB	8 GB	4 GB	4 GB	8 MB	512 MB	4 GB | 512 MB
Raw data trans.	Yes	Yes	No	No	No	No	No	No	No
Autonomy (days)	>1	>1	30@12.5Hz	1@25Hz	25	25	-	-	25 | 30@12.5Hz
Classification									
#Behaviors	5	5	4 | 2 | 2	4	4	3	3	2 | 4	9
Class. algorithm	DT	DNN	SVM | CART | LDA	QDA	RF	RF	CNN	RF	RF with SMOTE
Feature selection	RF	RA	No | No |RF	RF	RF	No	No		RF No
#Features	5	3	19 | 19 | 3	3	3	raw data	52	3	-
Window size (s)	5	-	10 | 30 | 30	10	2, 10 and 15	5	2	6	10
Accuracy (%)	85.5	95	76.9 | 98.1 | 90.6	89 - 93	94.6	89.6	96.59	83.7 | 70.9	72.4
On-animal class.	No	No	No	No	No	No	No	No	No

I: GBC = General Behavior Classifier.

Table 1.1: Behavior Classifiers with Commercial Data Loggers.

	[26]	[31]	[30]	[32]	[33]
General					
Application	GBC[I]	Grazing behaviors classifier	GBC	GBC	Sensor position effect
Custom made	Yes	Yes	Yes	Yes	Yes
Location	Neck	Neck	Ear	Neck	Neck
Online data	Yes	No	Yes	Yes	No
Weight (g)	281	-	4	-	-
Size (cm^3)	14.6×8×6.5	-	3.16×3.5×0.9	-	-
Technologies					
Microcontroller	MSP430FR5739	ARM Cortex-MO	Intel® Quark™ SE C1000	Texas CC1110 SoC	-
Communication	radio frequency	No	radio frequency	radio frequency	Not used
Accel. rate (Hz)	100	20	16	50	62.5
GNSS rate	0 - configurable timer	No	No	No	No
Memory	2 GB	any SD	384 kB	32 kB	SD
Raw data trans.	No	No	No	Yes	No
Autonomy (days)	>1	>1	2.4	134	>1
Classification					
#Behaviors	5	2	3	5	3
Class. algorithm	LDA	LDA	K-means and KNN	DT	CDA, DA
Feature selection	SFS	No	No	Fselector CRAN	No
#Features	12	55	20	11	12
Window size (s)	5.12	10	7	-	300
Accuracy (%)	82.4	98.65	85.18	91.78	90
On-animal class.	Yes	No	Yes	Yes	No

I: GBC = General Behavior Classifier.

Table 1.2: Custom made behavior classifiers.

Chapter 2

Hardware

A hardware diagram of the device can be seen in Fig. 2.1. The device features Actinius' Icarus IoT Board, which includes Nordic Semiconductor's nRF9160 System-in-Package (SiP) that integrates a NB-IoT modem and a GNSS module. The nRF9160 is equipped with an ARM Cortex-M33 processor, which includes 1 MB of flash and 256 kB of RAM. The board incorporates the LIS2DH12 three-axis accelerometer from STMicroelectronics. The Power Management Circuit (PMC), based on the BQ24074 chip from Texas Instruments, allows the device to be powered from various sources: three solar panels (totaling 2.8 W), a 2500 mAh LiPo battery, and a micro-B USB port.

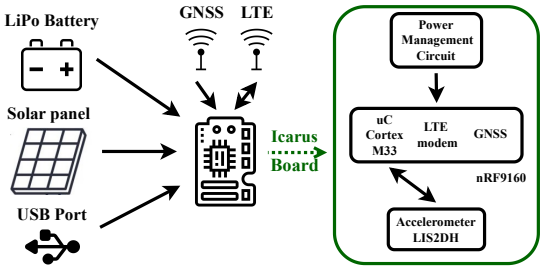


Figure 2.1: Hardware diagram of the device. Figure taken from [43].

The previous work Sistema para la caracterización de la dinámica espacial en ovinos (DEO), presented in [39] and introduced in Section 1.3.1, served as the starting point for defining the technologies and hardware platforms used in the present system. However, [39] employed the Texas Instruments MSP-EXP432P401R, and as stated in Section 1.3.1, this microcontroller was discontinued by the manufacturer. This required rethinking which technologies to implement. The idea behind these decisions is explained in the following sections.

2.1 Communication technology

When considering the communication technology for the OVIS system, it is useful to acknowledge the different roles that wireless connectivity may play in a livestock monitoring scenario. Although short range communication between collars could be

imagined as part of certain configurations, the needs of this project extend beyond the immediate vicinity of the herd. In practice, the system should ensure that data is reliably transmitted to a remote server for storage, visualization, and processing.

Several options were evaluated, including NB-IoT, Category M1 (CAT M1), Long Range Wide Area Network (LoRaWAN), and protocols based on IEEE 802.15.4, such as 6LoWPAN and ZigBee. While this list does not exhaustively cover all reviewed technologies, it includes those most relevant. Table 2.1 summarizes key characteristics of each technology, providing a comparative perspective across factors such as topology, data rate, coverage, and energy consumption.

2.1.1 IEEE 802.15.4

Protocols based on IEEE 802.15.4, such as ZigBee and 6LoWPAN, are well suited for local or indoor environments (typically within tens to hundreds of meters), where devices are in close proximity and can relay data through multiple nodes [44]. This network structure prioritizes connectivity: if one node fails, others can reroute the data, ensuring reliable communication across the network.

However, these protocols are less compatible with the OVIS system, which must operate over large outdoor areas (hundreds of meters to kilometers) and aims to minimize additional infrastructure. IEEE 802.15.4 networks require dedicated gateways and are typically operated by the system owner, increasing deployment complexity and cost [45].

In terms of quality of service, packet delivery may degrade as the network scales, particularly under high node mobility or radio interference. IEEE 802.15.4 supports data rates up to approximately 250 kbps, as specified in the standard, which limits its suitability for applications requiring frequent or high throughput updates. These networks remain valuable in scenarios where cellular or long range coverage are unavailable, but they are not ideal for extensive systems with high update frequency or large scale deployments.

2.1.2 LoRaWAN

LoRaWAN is ideal for applications that require low power consumption and long range communication across sparse areas, such as rural monitoring systems [46]. Operating on unlicensed frequencies, LoRaWAN allows for private deployment without telecommunications costs, providing flexibility and full control over the network (advantageous in areas lacking cellular coverage). LoRaWAN devices are well suited for battery powered use cases, with low standby consumption enabling extended autonomy in sparse communication scenarios.

However, deploying a LoRaWAN network requires private infrastructure, including gateways and servers, which can make it costly and complex to install and maintain in large, remote areas, such as those needed for extensive livestock farming. While LoRaWAN's range is considerable, coverage gaps may appear without a sufficient number of gateways. Its low data rate (between 0.3 and 27 kbps depending on configuration), is suited for small packets of information at infrequent intervals, and is less compatible with applications that require frequent or high throughput updates [46]. Overall, the total cost of deployment and ongoing maintenance must be carefully considered for large scale systems.

2.1.3 CAT M1

CAT M1 is a cellular communication technology designed for IoT applications that require higher data rates and mobility support, including Voice-over-LTE (VoLTE).

It operates over existing LTE infrastructure, avoiding the need for custom gateways or proprietary servers, which simplifies deployment and reduces maintenance requirements. Also, total infrastructure cost is relatively low due to reuse of cellular networks.

Its data rate can reach approximately 1 Mbps, with low communication latency (around 10–15 ms), enabling applications that demand frequent data exchange. Its continuous connection as devices move across cellular zones makes CAT M1 particularly useful for mobile applications like vehicle tracking. However, the mobility capabilities of CAT M1 far exceed the requirements of livestock monitoring, where animals move slowly and are often stationary within the context of this application.

Coverage is generally good in both urban and rural environments, though slightly more limited than as NB-IoT in remote areas. CAT M1 supports power saving features like Power Saving Mode (PSM) and Extended Discontinuous Reception (eDRX), but overall energy consumption remains moderate. Compared to NB-IoT, it is less optimized for low power operation, which can reduce battery life in long term.

CAT M1 is best suited for connected applications that involve mobility, larger payloads, or regular data transmissions across wide areas.

2.1.4 NB-IoT

NB-IoT is a 3GPP standardized cellular communication technology introduced as part of the LTE family. Although it shares the same core infrastructure as CAT M1, it adopts a much narrower bandwidth and a simplified connectivity model tailored for low power, wide area IoT applications. NB-IoT allows devices to activate and release radio resources only when needed, enabling long periods of operation in low power modes while maintaining reliable access to the network.

NB-IoT provides extensive and robust coverage as part of the existing cellular infrastructure, eliminating the need for proprietary gateways, local antennas, or custom servers. Devices connect directly to the operator's network, shifting responsibility for connectivity to the service provider and reducing deployment and maintenance costs. In Uruguay, Administración Nacional de Telecomunicaciones (ANTEL) offers nationwide NB-IoT service at a cost of USD 0.45 (excluding taxes) per device[1] Figure 2.2 shows the current NB-IoT coverage map.

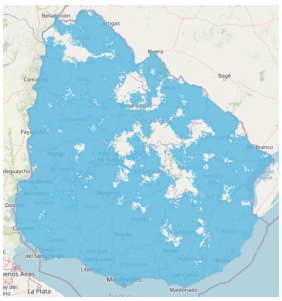


Figure 2.2: NB-IoT coverage provided by ANTEL in Uruguay (2025).

In terms of performance, NB-IoT supports low power modes. Typical data rates range from 20 to 250 kbps, which is sufficient for periodic transmission of small payloads. Latency is moderate (1.5–10 s), making it suitable for applications that do not require real time response. NB-IoT is optimized for large outdoor environments

[1] ANTEL IoT plan for NB-IoT devices, 2025 pricing.

and challenging reception conditions, which is essential for livestock monitoring in extensive agricultural areas, as in the OVIS system.

In summary, NB-IoT provides a cost effective, low power, and highly scalable communication solution that aligns with the OVIS system requirements for long term autonomy, minimal infrastructure, and reliable wide-area connectivity. Since NB-IoT was selected as the communication technology for the OVIS system, understanding its internal behavior becomes important and so, NB-IoT RRC state machine is described below.

NB-IoT state machine

The NB-IoT device's power consumption is largely determined by the Radio Resource Control (RRC) state machine. Figure 2.3 illustrates the main RRC states. In RRC Connected, the modem maintains an active radio link for uplink and downlink data transfer. When no traffic occurs and the RRC Inactivity Timer expires, the network releases radio resources and the modem moves to RRC Idle. In this state, the device remains registered but no longer holds a dedicated channel.

Within RRC Idle, the modem alternates between Idle, Paging, and PSM. In Idle, the device periodically wakes up using eDRX to synchronize or receive data, temporarily entering Paging. These wake-ups occur while the Active Timer (AT) is running. Once it expires, the modem enters PSM (if enabled), where it remains fully inactive. The device only exits PSM when a scheduled timer (TAU) expires or when an uplink transmission is required.

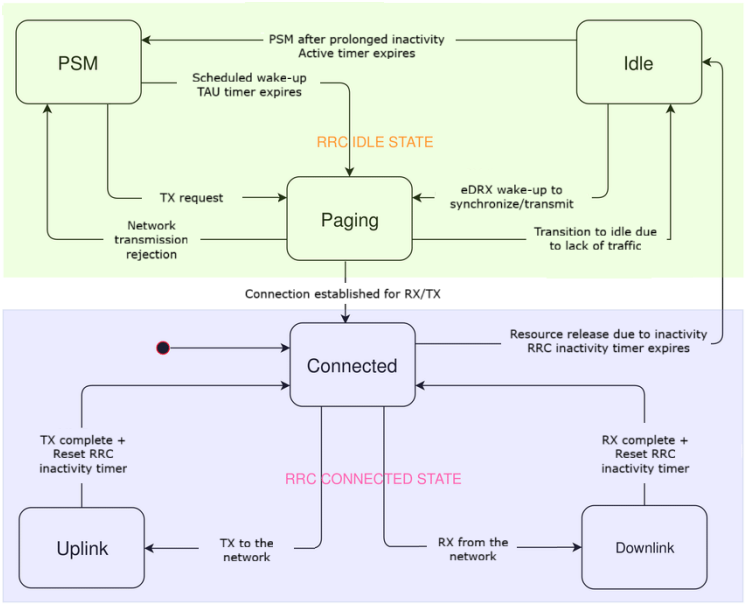


Figure 2.3: RRC state diagram in NB-IoT, showing the transitions between Connected, Uplink, Downlink, Idle, Paging, and PSM.

Transport and Application Layer

Once NB-IoT was selected, the transport and application layers had to be defined. NB-IoT supports standard transport protocols, including the Transmission Control

Protocol (TCP) and the User Datagram Protocol (UDP).

TCP is a connection oriented protocol that ensures reliable data delivery through acknowledgments, retransmissions, and ordered packet handling. In theory, these features make TCP attractive for systems where missing or corrupted data may compromise processing or interpretation, which motivated its selection for OVIS system.

By contrast, UDP provides a lighter weight, connectionless alternative that avoids buffering requirements but does not guarantee delivery. While this lack of reliability can be problematic, it reduces complexity, energy consumption, and memory buildup when connectivity is unstable.

It must be considered that achieving end-to-end reliability over NB-IoT has practical implications. In rural deployments with intermittent connectivity, TCP may accumulate unsent data in local buffers until the connection recovers. In a resource constrained device such as the OVIS decive, memory limitations restrict how much data can be stored. As a result, TCP's reliability guarantees are difficult to sustain in practice without significantly increasing memory capacity. This suggests that although TCP was adopted in the current implementation (supported by the reliability mechanisms outlined above) exploring a UDP based approach remains promising for future iterations.

With the transport layer explicitly set to TCP in the present design, MQTT was selected as the application layer. MQTT is widely adopted in IoT systems, lightweight, and supported by readily available libraries on both the microcontroller and server sides, which facilitated integration into the OVIS system.

General Characteristic	**ZigBee**	**6LoWPAN**	**LoRaWAN**	**CAT M1**	**NB-IoT**
Topologies supported	Mesh/Star	Mesh/Star	Star	Star (Cellular)	Star (Cellular)
Max data rate per terminal	250 kbps	250 kbps	0.3–27 kbps	Up to 1 Mbps	60 kbps down, 50 kbps up
Range/Coverage	∼100 m in., ∼1 km out.	∼100 m in., ∼1 km out.	2–5 km urban, 10–15 km rural	∼5 km urban	Hard to reach indoor, rural, wide areas
Deployment model	Private	Private	Private / Public network	Public cellular network	Public cellular network

Table 2.1: Comparison of communication technologies. Adapted from Finnegan et al. [47]. In=Indoor, out=outdoor.

2.2 Sensor types

Designing an animal monitoring system means the device has to gather information to characterize the animal's behavior. When data collection involves biological signals, as in this case, sensors must not alter the environment, influence the animal's behavior, or cause harm to the animal. These requirements translate into a need for sensors that are both accurate and simple, with size and weight as key considerations.

Based on [10], a comprehensive review of on-animal systems that monitor sheep behavior, the most common sensors incorporated in sheep monitoring systems are accelerometers, gyroscopes, thermometers, and location sensors.

2.2.1 Accelerometer

Three-axis accelerometers are the most common sensor in this type of system, as they could potentially provide information about movement patterns and activity levels of the animals.

Commercial accelerometers include standard interfaces for communicating collected data to a microcontroller, such as Serial Peripheral Interface (SPI) or Inter Integrated Circuit (I2C) protocol.

Moreover, most accelerometers offer a wide range of acquisition frequencies and bandwidth. This frequency must be carefully chosen based on the signal being measured, as it must respect the Nyquist–Shannon sampling theorem. Other variables to consider when choosing an accelerometer include the full-scale range and bit resolution. These features must be adequate to properly capture the desired signals.

Accelerometers come in small integrated circuits, meaning they do not add significant weight or size to this application.

The prior work in [39] successfully used an accelerometer for classifying sheep behavior, so it made sense to continue this approach and use an accelerometer as the main sensor for gathering information.

The LIS2DH12 three-axis accelerometer from STMicroelectronics was included with the chosen board (see Section 2.4). The accelerometer's main characteristics were sufficient for this application: the LIS2DH12 has a selectable full scale up to $\pm$16g, acquisition frequencies from 1 Hz to 5.3 kHz, a 12-bit resolution Analog to Digital Converter (ADC), and supports multiple operating modes. In Low Power Mode (LPM), it consumes 34 µA at 100 Hz, while in normal and high resolution modes, power consumption increases to 200 µA and 220 µA, respectively, at the same sampling rate. These characteristics make the LIS2DH12 an appropriate choice for this device.

2.2.2 Gyroscope

Gyroscopes are widely used to track rotational movements, which are useful for understanding behaviors such as head tilts or specific feeding actions. As gyroscopes capture angular velocity across three axes, they provide data complementary to accelerometers.

Like accelerometers, gyroscopes are compact and lightweight, typically housed in small integrated circuits, making them suitable for wearable applications on animals.

Power consumption is an important consideration with gyroscopes. These sensor typically measure angular velocity using vibrating elements and detect changes in vibration due to rotation. This process requires energy to keep the vibrating element oscillating, increasing power consumption. Moreover, gyroscopes often require signal processing to interpret the rotational data, including filtering and compensating for drift.

Although gyroscopes are often used in animal behavior studies to track orientation and motion changes, they were not included in this design to minimize power consumption and simplify data processing. While gyroscopes could provide additional detail, the accelerometer alone was considered sufficient for detecting movement patterns in this application.

2.2.3 Thermometer

Animal and ambient temperature are variables commonly incorporated into animal monitoring systems to track health indicators. Climate variables can also drastically affect animal behavior.

Measuring the sheep's body temperature would require a subcutaneous sensor. Although implantation is not a complex procedure, subcutaneous devices cannot guaran-

tee a stable position and may migrate within the animal's body, which raises sanitary and commercial concerns [48]. Since the OVIS system is designed to be fully non-invasive and collar worn device, subcutaneous temperature monitoring was discarded.

Measuring ambient temperature is simpler. A temperature sensor could be added to the device, or the microcontroller's temperature sensor could be used. Additionally, the accelerometer LIS2DH12 includes an embedded temperature sensor. While several methods exist to monitor ambient temperature, a decision was made to focus on movement based behavior analysis for simplicity. However, climate monitoring should be a future step for this system.

2.2.4 GNSS

GNSS sensors are essential for tracking location and movement over large areas. Location data can be extremely valuable in an animal monitoring system, as the whereabouts of a sheep could indicate threats (e.g., predators, theft). Moreover, sheep are social animals, so data on flock behavior or an individual sheep's behavior relative to its flock could be useful.

However, GNSS can significantly impact battery life, so sampling rates and data collection intervals must be carefully chosen to balance location tracking and power efficiency. Regarding size, GNSS modules can be implemented as integrated circuits, but location sensors need an external antenna, which must also be considered.

The GNSS module used in this system was chosen because it came integrated with the selected microcontroller and board (see Section 2.4). However, the module's main characteristics were considered sufficient for this application. This module is a GPS L1 C/A, this means that uses the primary frequency for civilian GPS signals, 1575.42 MHz (L1), and modulates the signals with the Coarse/Acquisition (C/A) Code. These signals are transmitted by a constellation of satellites orbiting the Earth, and the system determines its position by calculating the time delay between the moment each satellite's signal is sent and the moment it is received. This GNSS module reportedly consumes about 25 mW during active positioning, which is low compared to standalone GNSS modules that can draw over 100 mW.

The GNSS cold start time (initial Time To First Fix -TTFF- from a powered-off state) is about 30 s to 40 s, while warm start times (after brief power-downs) are around 5 s to 10 s. By managing duty cycles to use warm starts, the system can save power while still achieving rapid position fixes when needed.

2.3 Power supply

When designing a system for extensive livestock farming, considerations around power consumption, battery type and life, autonomy, and energy harvesting are crucial. Since the device is expected to operate for long periods without manual intervention, the goal was to achieve virtually unlimited autonomy. Based on the estimated power consumption of all components, this led to the decision to use rechargeable batteries in combination with an energy harvesting solution. Among the available alternatives, only solar panels were considered viable for this context, based on [49].

2.3.1 Battery

The device's battery was selected after determining the board (see Section 2.4). The Icarus IoT board includes a PMC, allowing the device to be powered from multiple sources: a micro-B USB port, an external voltage input, and a LiPo battery. The battery chemistry is predetermined, as the charger chip embedded in the Icarus IoT

Board is configured specifically for this battery type. The board also specifies that the battery voltage should range between 3.2 V and 4.2 V, with a nominal voltage of 3.7 V.

Then, the battery must be a LiPo with a nominal voltage of 3.7 V. Beyond this, the selection involved balancing capacity and size. The chosen battery provides a capacity of 2500 mAh, with dimensions of approximately 7.8x50x60 mm.

2.3.2 Solar panels

To extend the device's autonomy, solar energy harvesting was incorporated into the design. The quantity and size of the solar panels were selected with careful consideration of the device's mechanical design (see Chapter 6).

The device is designed to function as a collar surrounding the sheep's neck. While a circular or oval shape sounds ideal, a flat surface was more appropriate for mounting solar panels. Therefore, the collar was designed with three exterior flat sections, each holding a solar panel (see Figure 6.9). The lateral sections were angled to maximize solar exposure while maintaining a shape that adjusts to the sheep's neck.

When selecting the solar panels, the most important balance was between size and power output. Larger panels generate more power, which translates to extended autonomy for the device. Since maximizing autonomy was a priority and also that the panel size did not interfere with animal welfare, the overall collar dimensions were determined taking into account the solar panel size.

Two IXOLAR SM531K10TF panels were selected for the lateral sides, and a smaller IXOLAR SM261K10TF panel was chosen for the front side. The front panel had to be smaller to allow key components, such as antennas, an unobstructed view of the open sky, minimizing interference. The lateral panels have a size of 89x65x1.2 mm, a maximum power of 1141 mW, with 5.58 V as typical voltage and 204.5 mA as typical current. Whereas the front panel presents a size of 67x45x1.2 mm, a maximum power of 570.8 mW, with 5.58 V as typical voltage and 102.3 mA as typical current. It is important all panels work at the same voltage, so they can be connected in a parallel configuration.

2.4 Microcontroller

To decide which microcontroller would replace the MSP-EXP432P401R, a comparison was made between several companies and devices. The main goal was to find a microcontroller with similar or better capabilities than the MSP-EXP432P401R. Additionally, the new microcontroller needed to be able to drive several sensors and enable wireless communication. The final decision came down to the STM32L433RC from STMicroelectronics and the nRF9160 from Nordic Semiconductor. Table 2.2 shows the main characteristics of the three microcontrollers. It is important to note that the nRF9160 works as a SiP, meaning the system includes two microcontrollers: one dedicated to managing wireless communication and location sensor acquisition, and the other, reported in Table 2.2, dedicated to the application itself.

The nRF9160 requires a higher power supply than the STM32L433RC but uses an ARM Cortex-M33, which offers better performance than the ARM Cortex-M4F. STM32L433RC achieves a maximum frequency of 80 MHz, while nRF9160 has a higher frequency than the MSP432P401R at 64 MHz. Considering that 48 MHz was sufficient for the previous application, the maximum frequency of the nRF9160 should suffice.

In terms of memory, the nRF9160 quadruples both the flash and RAM capacities, with 1 MB and 256 kB respectively, compared to the STM32L433RC and

General caracteristic	**MSP432P401r (Texas Instrument)**	**STM32L433RC (ST microelectronics)**	**nRF9160 (Nordic Semi Conductor)**
Power supply (V)	1.62 - 3.7	1.71 - 3.6	3.0 - 5.5
Microcontroller architecture	ARM Cortex M4F	ARM Cortex M4F	ARM Cortex M33
Word size (bits)	32	32	32
Max. frequency	48	80	64
Flash	256 kB	256 kB	1 MB
RAM (kB)	64	64	256
FPU	Yes	Yes	Yes
RTOS	Yes	Yes	Yes
Libraries	MSP432 Driver Library	STM32CubeL4	nRF Connect SDK
Power consumption			
Run Mode (mA)	1.62@48 MHz	3.2@80 MHz	6.6@64 MHz
LPMs (µA) (RAM retain, RTC on)	8.9	1.4	2.7
Wakeup Time from LPM (µs)	6	5	30 (LTE wakeup)
Peripherals			
Timers	4x16 bit, 1x32 bit	5x16 bit/32 bit, 1x16 bit low power	5x32 bit
RTC	32 bit	32 bit	24 bit
SPI	4xeUSCI_B modules, up to 16 Mbps	3xSPI interfaces, up to 32 Mbps	4xSPI interfaces, up to 8 Mbps
I/O (GPIO)	84	51	32
ADC	14-bit, 24 channels, up to 1 Msps	12-bit, 16 channels, up to 5 Msps	12-bit, 8 channels, up to 200 ksps

Table 2.2: Microcontrollers comparison.

MSP432P401R. This is an important distinction, considering the amount of sensor data to be stored and the expected on-device processing.

LPMs are difficult to compare since each manufacturer reports them under different conditions. Table 2.2 aims to align the most comparable low power figures available in the respective datasheets. Based on these values, the nRF9160 and the STM32L433RC exhibit similar low power consumption, and the data do not support an order-of-magnitude difference between them. It is important to consider, however, that the nRF9160 integrates two microcontrollers and an CAT M1 and NB-IoT modem, features that inherently increase power consumption during cellular communication events.

The peripheral capabilities of the STM32L433RC and MSP432P401R are slightly different, but in both cases are considered sufficient for the needs of the current project.

It is important to mention that the decision between these two devices was made in 2021, during the global chip shortage, which led to long wait times. The STM32L433RC has a development board with a size that did not allow it to be included in a sheep collar. This required designing and manufacturing a custom Printed Circuit Board Assembly (PCBA), and lead times were uncertain, potentially affecting the overall project deadlines. On the other hand, the nRF9160 was available in Actinius' Icarus IoT Board. Actinius is a different company from Nordic Semiconductor, which introduced additional overhead during development, as it required understanding potential incompatibilities and implementation differences between Actinius' hardware and Nordic Semiconductor's reference designs.

However, this board comes with an integrated three-axis accelerometer, the LIS2DH from STMicroelectronics. Also, the Icarus IoT Board has a PMC based on the BQ24074 chip from Texas Instruments, allowing the device to be powered from various sources, including solar panels, USB, and a LiPo battery. Furthermore, the board supports wireless communication through NB-IoT LTE and CAT M1 LTE, for this low power, wide area connectivity system. Taking all of these arguments into account, the decision was made to select the Icarus IoT Board from Actinius with the nRF9160.

This page has been intentionally left blank.

Chapter 3

Embedded software

This solution[1] was developed using Zephyr RTOS and C programming language. Zephyr RTOS is open source and developed specifically for devices with constrained resources, such as low power IoT embedded applications. Zephyr RTOS is Nordic's recommended RTOS for the nRF9160. This means Zephyr RTOS has Nordic's support, forums and documentation, including drivers and sample applications. The drivers and sample applications are part of Nordic's library for their devices (including nRF9160), the nRF Connect SDK. This library was widely used in the present system.

nRF Connect SDK has its extension for Visual Studio Code (VSCode). The extension allows using VSCode as an Integrated Development Environment (IDE) for the Icarus IoT Board. Developing, debugging, building and flashing code into the board are some of the capabilities VSCode and nRF Connect SDK enable.

3.1 Zephyr RTOS

Zephyr is an RTOS, supporting various hardware platforms including ARM Cortex-M series, which is the processor used in the nRF9160, as well as several communication protocols like TCP, MQTT, and NB-IoT. It should also be noted that the RTOS is preemptive, meaning that the system can give precedence to higher priority tasks. Zephyr supports Firmware Over the Air (FOTA) updates, meaning devices can be updated remotely[2].

3.1.1 General characteristics

Zephyr RTOS has an event driven architecture, highly modular and with a hierarchical structure. This gives a separation between the RTOS elements and the specific application. On an architectural level Zephyr RTOS code can be divided into three layers [50]. In each layer, a module can use modules from the same and its lower

[1] Part of the embedded software implementation was carried out with the support of Andrea Delbuggio. Andrea programmed embedded software functions (battery level measurement, base64 encoding, among others) and played a crucial role in the final phase of debugging, testing, and integrating the embedded software as a whole. Except for the contributions detailed before, it should be assumed that the remaining work in this chapter was led by Varinia Cabrera.

[2] This characteristic was not applied in the present system, however, it allows a lower maintenance, facilitating large scale deployments. This is appropriate for an extensive livestock farming motoring system, and should be explored in the future.

leveled layer. Figure 3.1 shows a diagram with the three layers. The diagram does not attempt to be exhaustive and only shows some example modules as reference for each layer.

The kernel layer manages threads, scheduling, power management, memory and more.

The subsystem layer works as a support for the kernel and has functionalities such as network management, drivers and system functionalities as debugging. The drivers are designed to interface with peripherals (e.g., GPIO, UART, SPI) while maintaining standardized Zephyr RTOS interface, ensuring consistence regardless of the hardware.

The application layer has the custom project and involves the modules managing sensors, communication protocols and data processing. The main module (**main.c**) is the entry point for the application and for additional modules with a specific functionality.

Configuration files work together to define the system's behavior and hardware setup. The **prj.conf** file specifies the application level settings. Meanwhile, development boards have their own configuration files, provided as **.overlay** files, which modify the default Device tree files (**.dts**). Device tree files define hardware configuration, how peripherals are connected and initialized. At build time, all these configurations are merged into the Kconfig system, unifying application settings, board specific configurations, and hardware definitions.

Zephyr's build system is structured by CMake (**CMakeLists.txt**). The build unites a collection of core libraries with kernel functionality, subsystems, device drivers and application code. If external libraries are needed, they can be integrated.

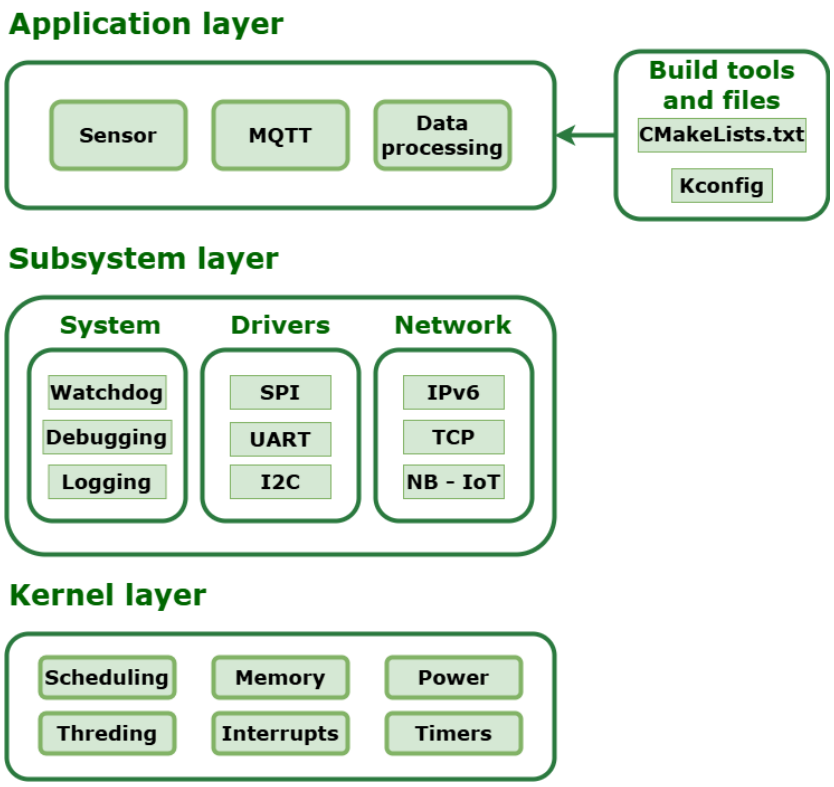


Figure 3.1: Layer diagram for Zephyr RTOS. Figure inspired from [50].

3.1.2 Schedule: threads and events

An application developed on Zephyr RTOS organizes its execution using a combination of initializations, threads and events. Initialization functions are executed during boot and can be divided in pre-kernel, kernel initialization and application initialization.

For asynchronous occurrences, Zephyr uses events. They represent notifications that something has happened, such as a hardware interrupt, a timer expiration, or a software triggered occurrence. Whereas, threads are designed to perform specific tasks independently. Each thread has an assigned priority. Moreover, threads can wait for specific events and be scheduled for execution when those events occur.

Figure 3.2: Thread state machine for Zephyr RTOS. Diagram inspired from [51].

Figure 3.2 presents the different states a thread can be in: New, Ready, Waiting, Suspended, and Terminated. When a thread starts, it transitions from New to Ready, awaiting execution. The scheduler dispatches the highest-priority Ready thread, transitioning it from the Ready state to the Running state. Furthermore, because the scheduler is preemptive, a Running thread can be halted and moved back to the Ready state.

When a thread needs to wait for an event or a group of events to occur, the system uses synchronization and notification mechanisms (such as an asynchronous action that triggers an Interrupt Request -IRQ-, the release of a semaphore, or a message queue becoming non-empty) to transition the thread from the Running state to the Waiting state. Once the awaited event occurs, these same mechanisms can notify the scheduler that the thread can resume execution, causing a transition from Waiting back to the Ready state.

Once a thread is started it typically executes forever. However, a thread can be prevented from executing for an indefinite period of time if it becomes Suspended. A thread cannot be scheduled until another thread removes the suspension. Moreover, a thread may asynchronously end its execution by aborting (Terminated thread). The kernel automatically aborts a thread if the thread triggers a fatal error condition.

Event queues are used to manage multiple events. An event queue stores messages, allowing threads to process them based on a defined order of priority.

3.2 OVIS application

The OVIS embedded software (OVIS application) must provide a solution to the following problem. The device must be able to acquire several types of data. One is GNSS data, and another is accelerometer data with a specific sensor acquisition frequency. Additionally, the battery level and cellular signal strength must also be acquired. The device must obtain this data periodically at a defined time interval.

Furthermore, the device must serve as an animal behavior classifier. To achieve this aim the device has to process and classify the gathered information. In particular, it must calculate specific features with a defined window size for the acceleration data. These features serve as input data for an algorithm implemented on the device (for implementation details and algorithm selection see the Signal processing chapter).

Moreover, the device must be able to transmit both raw and processed data via NB-IoT. So the data can be stored, further processed, and displayed at the OVIS cloud server. Also, the OVIS application should be designed to maximize the device's autonomy. This includes implementing low power modes at all levels of the system: sensors, microcontroller, NB-IoT communication, and PMC.

Finally, as mentioned before, the system scalability should characterize it. The embedded software design should allow: further on-animal and server based processing, the addition of biopotential signals sensors and/or environmental sensors (e.g., thermometer).

3.2.1 Asset Tracker v2

The OVIS application was based on the Asset Tracker v2 application from Nordic Semiconductor [52]. The similarities between the operation of Asset Tracker v2 and the functionalities required by the OVIS application encouraged the use of the Nordic Semiconductor example as a starting point for the OVIS application.

The Asset Tracker v2 solution is designed to track vehicles that can travel long distances at high speeds (compared to the movement speed of a sheep), such as tractors and trucks. The application samples and transmits sensor data to a connected cloud service using LTE (CAT M1 or NB-IoT). The application focuses on the power saving features of the nRF9160 chip. It is worth mentioning that AWS (Amazon Web Services) IoT Core is supported by the application as a cloud service (the AWS preference is analyzed in the OVIS cloud server chapter).

The Asset Tracker v2 has two operating modes: active and passive. In active mode, it transmits location data, environmental data, battery status, and LTE signal information at regular intervals through LTE to a cloud service, regardless of movement. In passive mode, it transmits the mentioned data when movement is detected by the accelerometer and if no movement is detected, the data is transmitted at significantly larger intervals.

Rather than implementing a motion detector (as in passive mode) to determine when to send data, the OVIS application should transmit multiple types of data at regular intervals, as in active mode. However, active mode lacks the accelerometer data collection present in passive mode. Therefore, the OVIS application architecture and backbone are inspired by the active mode, while integrating the accelerometer functionality from the passive mode. Additionally, other functionalities were incorporated to meet the requirements of the OVIS application. Main changes from Asset Tracker v2 application are described below.

Accelerometer integration

The accelerometer in the Icarus IoT board, the LIS2DH from STMicroelectronics, differs from the ADXL362 used in Nordic's Asset Tracker v2 application. As a result, a different driver was required for the LIS2DH accelerometer. In addition, the software was adapted to acquire acceleration data at 25 Hz and stream it to the OVIS cloud server. This posed a challenge due to the high data rate required for continuous accelerometer streaming.

Accelerometer data coded with base64

Because of the large volume of data generated, the accelerometer readings were encoded using base64 before being sent to the OVIS cloud server. This approach was chosen for several reasons: it was simple to implement, requiring minimal additional processing. Also, it was the most transparent solution for Asset Tracker v2, maintaining compatibility and minimizing the need for changes in the application logic. Most importantly, it was the most transparent solution for the server, allowing easy parsing and storage of the incoming data without complex decoding logic. Being that said, encoding accelerometer data in base64 is not the most efficient method. Therefore, more efficient alternatives should be explored for future versions.

Each acceleration sample consists of three values (one value for each accelerometer axis). As the sensor is configured with a $\pm$ 4000 mg scale, the acceleration take values from -4000 mg to 4000 mg. For this reason, five characters (four digits and one sign character) are used by the method applied in Asset Tracker v2 for sending data to the cloud, representing the acceleration samples in a human readable format. If each acceleration value is represented with five characters, it gives a total of 5 x 3 = 15 characters per sample.

Considering each acceleration value is expressed by the accelerometer with two bytes, the minimum number required to represent one acceleration value is 16 bits, this means 65536 possible values to represent ($2^{16} = 65536$). It is important to clarify that, although the LIS2DH features a 12-bit ADC in high resolution mode, the sensor does not output packed 12-bit values. Instead, acceleration samples are always delivered as left justified 16-bit two's complement words [53].

To reduce the original Asset Tracker v2 data representation, the acceleration values are coded in base64 according to the RFC4648 standard. Base64 encodes converting data into a text representation with a 64 character alphabet. In other words, with a 3 byte word a total of 262144 values could be represented ($64^3 = 262144$)[3], being this size enough for one acceleration value representation.

Using a 3 character word does not fully comply with base64 coding since the technique works with 4 characters words, a padding character (=) should be added but since the acceleration data has a fixed width and 3 characters are always expected, no padding is needed in this case.

So, this code gives a total of 3 x 3 = 9 characters per sample. This technique reduces the number of bytes needed from 15 to 9 bytes, which is 60 % of the original payload.

Algorithm in the microcontroller

One of the main contributions of this work is the implementation of an animal behavior classifier, whose algorithm details and selection are discussed in the Signal processing chapter).

[3] If a 2 byte word was used, not every possible value could be represented since $64^2 = 4096 < 65536 = 2^{16}$.

GNSS and LTE implementation

Due to RAM constraints in nRF9160, the maximum manageable throughput is approximately 15 kB per packet. This limitation defines a fixed packet transmission interval of 50 seconds. Most of this payload is occupied by raw acceleration data: at an acquisition frequency of 25 Hz, and with three axes, a total of 75 samples per second are recorded. Each axis sample is coded in 3 bytes, resulting in a total of 225 bytes per second. Over a 50 second period, this results in 11250 bytes per packet. Additional data, such as processed values, modem information, battery level, timestamps, and GNSS data are also included, but their contribution to the packet size is minor in comparison[4]. The transmission of this packet takes the modem approximately 20 seconds.

Regarding transmission and acquisition times, the nRF9160 has a clear limitation: the GNSS cannot obtain a location while the modem is transmitting. Therefore, the timing of transmissions to the OVIS cloud server must take into account the GNSS configuration.

While the modem is not transmitting, during the 30 s window between packets, up to three location fixes can be acquired (e.g., at 10, 20, and 30 seconds) and included in the packet sent to the OVIS cloud server every 50 seconds.

If the device were to transmit information at extended intervals, the 20 second transmission limitation imposed by the raw accelerometer data becomes irrelevant. In that case, location data could be obtained without major restrictions on acquisition frequency. However, it is important to consider the following restriction: location data is acquired once at startup, after which the GNSS module is deactivated. It is then periodically reactivated. The GNSS requires at least 30 consecutive seconds to obtain a satellite fix when acquiring a position for the first time, or when satellite data expires typically every 2 to 3 hours. In such cases, the LTE modem must remain idle for at least 30 seconds.

Other modifications

Optimizations were made to minimize RAM usage and expand data storage availability. For example, initially, acceleration data had a separate static buffer in each module where it was needed. Since three modules used accelerometer data, three static buffers were required. Instead, the OVIS application stores accelerometer data in a single shared static buffer, accessible to the relevant modules.

Furthermore, the collection of battery level was modified to work with the Icarus IoT Board's PMC. Also, the retrieval of environment data was disable since environmental sensors are not present in Icarus IoT Board.

3.2.2 Modules

OVIS application software architecture is modular, featuring an event manager for software modules communication. Modules can submit and subscribe to events. In Figure 3.3 OVIS's modules can be seen, distinguishing modules with and without threads. Six modules are implemented for the software custom functionalities (Application, Data, Sensor, Cloud, Modem and GNSS) and two modules (Utils and Debug) provide assistance to the others, helping with monitoring, administration and debugging capabilities.

The modules with threads receive messages that might take a while to process. Such messages are not suitable to be processed directly in the event handler. Modules

[4]The amount of data transmitted in a 50 s interval is quantified Appendix D.

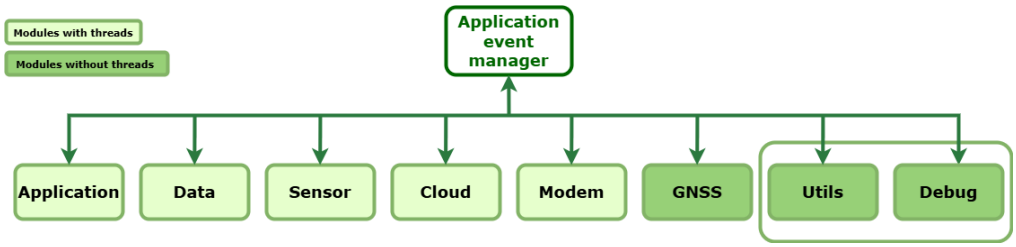


Figure 3.3: Module diagram for OVIS embedded software. Figure inspired from [52].

without thread process events in the context of system work queue in the Application Event Manager. So, their workloads must be light and non-blocking. The different work flows between modules with and without threads can be seen in Figure 3.4. In a module with thread the event manager transforms the event into a message and the message is queued. The dedicated thread will act on the message depending its state. Whereas, in a module without thread the event is converted into a message and is processed directly.

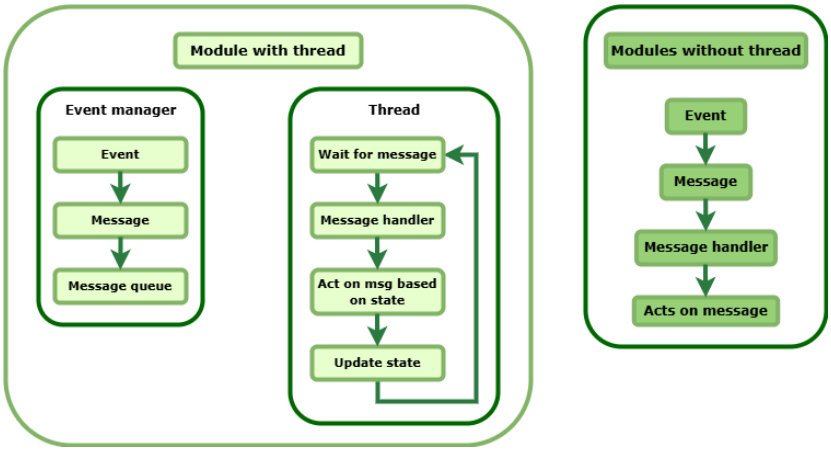


Figure 3.4: Module work flow with and without thread. Figure inspired from [52].

In the following paragraphs, a brief description of each module is provided to clarify their main functionalities. The most important modules for the OVIS application (i.e., application module, data module and sensor module) are discussed in Appendix Key OVIS modules description with an augmented and more technical explanation.

Application module

The application module (**main.c**) is the core of OVIS application. The module oversees initialization processes and coordinates periodic data requests from other modules. This module is central to managing system operations, running the primary thread responsible for executing OVIS application.

Said data includes static and dynamic modem data (see their definition in the modem module description), battery voltage data, accelerometer raw data, accelerometer processed data and location data.

The application module requests this information from the data module, which centralizes data collection from other modules. Then, with this collected information, the system is able to transmit to the OVIS cloud server through the cloud module.

These requests occur periodically and are commanded with a timer on the application module. This timer function is to trigger an event to retrieve data. The data request actually occurs when function **data_get()** submits an event, APP_EVT_DATA_GET[5], signaling other modules to report the data. The timer duration is set to 50 seconds, as the OVIS application transmits data to the OVIS cloud server every 50 seconds.

Data module

The data module gathers all sampled data from other modules (determined by the application module and defined in the application module description) and stores it in circular static buffer structures, centralizing data handling. Therefore, the data module manages data encoding, accelerometer data processing for transmission, and monitors the status of data requests made by the application module. Moreover, this module has dedicated thread, which was started by the application module.

When acceleration data acquisition is ready the data processing is done here, in the data module. Accelerometer data processing involves calculating specific features from raw accelerometer data and performing a classification and behavior prediction using the implemented algorithm and the extracted features[6].

If the device is connected to OVIS cloud server, the data module prepares different type of data to be sent to OVIS cloud server. The encoding process focuses on two elements: format data with JavaScript Object Notation (JSON) to be readable by an AWS server (as is OVIS cloud server) and code accelerometer data with base64 to reduce its volume. To achieve these objectives several auxiliary functions are used from **aws_iot_codec.c**, **jason_common.c** and **common.c** libraries. In particular the **common.c** library is modified to code the acceleration data. When data is coded without error event DATA_EVT_DATA_SEND is submitted. Said event is monitored by the cloud module and this is the module that actually sends the gathered information to the OVIS cloud server.

Beyond the JSON structure, the body of the message itself is formatted as plain text composed of identifier–value pairs separated by newline characters (see 3.1).

Listing 3.1: Illustrative example of how an MQTT message is received by the OVIS cloud server.

```
rsrp:-100
ts_roam:1762971998858
x:/yo/zI/zA/yw/y4/yg/zA/zQ/zI/...
y:/Zo/ZY/Zg/ZQ/...
z:/P4/QI/QI/Pg/...
b:4060
ts_bat:1762971998785
e:1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1,1
```

Among the transmitted fields are the signal quality indicator (`rsrp`), timestamps (`ts_roam`, `ts_bat`), battery voltage (`b`), event markers (`e`), and the base64 encoded acceleration series for the three axes (`x`, `y`, `z`). Whereas, the remaining fields are transmitted as integers or ASCII formatted lists. The resulting packet size depends on the number of samples within the feature calculation window used by the behavior

[5]Events names in OVIS application start with the abbreviation of the module submitting the event, followed by EVT and then words that characterize the event itself.

[6]Due to its importance this implementation is explained in the Signal processing chapter.

classifier. With the configuration used in the OVIS device the message length is approximately 11.3 kB (see Appendix D for details). The following snippet shows a simplified example of the format in which the OVIS cloud server receives MQTT messages sent by the device:

Additionally, the data module sets configurations for the whole OVIS application which are stored in flash memory, including settings such as the data transmission period to the OVIS cloud server and the GNSS sampling interval. The application is designed to receive configuration updates through a reboot[7].

Sensor module

The sensor module is design to interact with hardware elements. In the present OVIS application manages accelerometer and battery data, nevertheless other sensors could be added (e.g., biopotential signals sensors and/or environmental sensors). This module has a dedicated thread, analogous to the one in data module, with a similar execution function.

This module ensures the LIS2DH accelerometer is properly configured and initialized at boot time, then sets up and enables interrupts from the accelerometer. The communication with LIS2DH is set up through I2C and the accelerometer is configured with a full-scale range $\pm$ 4 g with the high resolution mode to obtain 12-bit acceleration measurements. Then, a dedicated thread is created to wait for, enable, and handle interrupts[8]. After thread creation, data-ready interrupts are set by configuring the General Purpose Input Output (GPIO) pin as an input to receive the interrupt signal and register an interrupt callback function for when the interrupt actually occurs. Finally, accelerometer measurements are enabled and a data output rate or acquisition frequency of 25 Hz is set.

Sensor data is retrieved for the three axes X, Y and Z. When accelerometer data is ready, the raw data is scaled adjusting the values to fit the configured range ($\pm$ 4 g) and the data module is notified through an event, containing the scaled acceleration data for each axis.

When the application module request information, the sensor module retrieves battery voltage information using an Actinius provided library. The library configures, initializes and reads the nRF9160 ADC peripheral. After battery voltage is read, the data module is informed battery data is ready.

Cloud module

The cloud module is responsible for establishing and maintaining the connection to AWS IoT Core, since OVIS cloud server is hosted in AWS.

This module uses a generic Application Programming Interface (API, the cloud wrapper API) to interface with AWS IoT Core library through an associated integration layer. This generic API allows OVIS application to manage cloud communication through functions such as connect, send, and disconnect, hiding the functionality that is specific to AWS IoT Core library implementation.

The module uses MQTT over Transport Layer Security (TLS) and TCP to communicate with AWS using a specific MQTT topic. Library **aws_iot_integration.c**

[7] The code for receiving configuration updates on; every connection to the server and when the device sends information to the server, is already implemented and could be tested in the future.

[8] This means that the LIS2DH driver will use a dedicated thread (in addition to the sensor module thread) to handle sensor interrupts instead of relying on the system work queue. This ensures that interrupts are handled as quickly as possible, reducing the likelihood of losing acceleration data.

was modified to create the topic $< imei > /datos\_ovis$ and send all the information to OVIS cloud server through this topic.

The cloud module implements connection awareness by maintaining an internal state based on events from the modem module and notifications from the cloud wrapper API. If the module is disconnected, it will try to reconnect while the LTE connection is still valid. The number of reconnection attempts can be configured with Kconfig options. If the maximum number of retries is reached, the module emits an event (CLOUD_EVT_ERROR), causing OVIS application to reboot.

To connect with AWS IoT Core, some mandatory Kconfig options must be configured and the required certificates must be provisioned to the modem using the specified Kconfig options mentioned before. For more information on how to set up a connection and provision certificates for the modem, see Section Connecting to AWS IoT Core.

Modem module

The modem module communicates with the modem to control the LTE connection and retrieve information from the modem about the device and LTE network. To achieve this the module uses a library called LTE Link Controller, this library allows receiving modem updates on connection status and network information.

By default, it attempts to establish a LTE connection upon initialization, continuing until a network is found. If the device connects to a new LTE cell, the module ensures that the cell ID and the tracking area code are provided. The module can also handle disconnections. Connection parameters (e.g., access technology and PSM timers), can be configured via Kconfig options, for more information on how to set up a connection NB-IoT LTE see Connecting to the cellular network.

The data module requests information from the modem module, which retrieves both static and dynamic modem data using the Modem information library[9]. Static modem data does not change frequently and is therefore collected only once per OVIS application boot. This information includes the Integrated Circuit Card Identifier (ICCID), which uniquely identifies the Subscriber Identity Module (SIM) card, and the International Mobile Equipment Identity (IMEI), a unique identifier of the modem hardware. In addition, static modem data comprises the firmware version, application version, and board version running on the modem.

In contrast, dynamic modem data varies based on network conditions. It includes the area code and cell ID, which identify the current LTE cell and the area where the device is connected. Additionally, it provides the Mobile Country Code (MCC) and Mobile Network Code (MNC) that define the mobile network operator, as well as the IP address assigned to the device. The Access Point Name (APN), which specifies the gateway for network connectivity, and the LTE frequency band in use are also part of this data. Moreover, dynamic modem data indicates whether the device operates in CAT M1 or NB-IoT mode and includes the Reference Signal Received Power (RSRP), which measures signal strength.

GNSS module

The GNSS module, as its name indicates, manages the GNSS functionality of the OVIS application and is used to acquire location data. It communicates with the nRF9160 modem using the GNSS interface[10], which is responsible for configuring the GNSS module and retrieving location data.

[9] Modem information library can be used to obtain specific data from a modem.
[10] GNSS interface is a library used to control the GNSS component.

A satellite search begins when the module receives an event APP_EVT_DATA_GET from the application module requesting certain data types to be sampled. Reporting location data is considered of great value, however, not sampling GNSS data has a direct effect on power consumption[11]. Thus, OVIS application allows disabling location data requests.

The module is designed to operate in single fix mode, meaning it will search for a satellite fix for a defined time interval and then stop, either successfully acquiring a fix or timing out. The GNSS module receives configuration from the data module to acquire location data every 50 seconds, as mentioned in the data module description.

When a satellite fix is successfully obtained, the module sends a GNSS_EVT_DATA_READY event. If the attempt times out, it sends a GNSS_EVT_TIMEOUT event.

Unlike other modules, the GNSS module does not employ a dedicated thread for processing messages at application level. This is because these inter-module messages are considered lightweight and can be handled efficiently by the Application Event Manager without requiring additional thread scheduling overhead.

In contrast, events originating from the GNSS interface follow a different path. Notifications from the GNSS hardware are triggered within interrupt context, where performing any time consuming or blocking operation is unsafe. To address this situation, the GNSS module internally creates a small dedicated thread whose sole purpose is to defer these hardware driven events out of the interrupt context and process them safely at thread level.

Utility module

The utility module provides the functionality to administer and monitor mechanisms in the application architecture. One mechanism is the shutdown process in cases of irrecoverable errors, ensuring a graceful shutdown of the application. During initialization, each module registers to support graceful shutdown.

The shutdown sequence begins when a module detects a critical error and sends an event (e.g., CLOUD_EVT_ERROR). Then, the utility module starts a reboot timer defined by CONFIG_REBOOT_TIMEOUT (a Kconfig option). Moreover, the module requests a shutdown with the event (UTIL_EVT_SHUTDOWN_REQUEST) to other modules. Once a module completes its shutdown tasks, it declares it is ready for shutdown with an event (e.g., CLOUD_EVT_SHUTDOWN_READY). When all modules have confirmed a possible shutdown, the utility module reboots the application at an earlier point than the time set initially by the reboot timer. If the reboot timer expires before all modules report readiness, the utility module triggers an immediate reboot to avoid leaving the system in an undefined shutdown state.

Another mechanism is the watchdog timer, which monitors the system work queue. The timeout is configured using the CONFIG_WATCHDOG_APPLICATION_TIMEOUT_SEC Kconfig option. In this application, the default Asset Tracker v2 watchdog is used; the utility module feeds it regularly, and if the system work queue becomes blocked and fails to feed the watchdog within the configured timeout, a hardware reboot is automatically triggered.

Debug module

This module boost the application's debugging capabilities by subscribing to all system events and integrating the Memfault tool through the nRF Connect SDK. This tool collects device data, like core dumps, error traces, metrics, and logs. The module tracks

[11] For more information on OVIS device's autonomy see the Testing chapter.

metrics, such as LTE and stack performance, and defines custom metrics for GNSS operation (e.g., time to fix, search timeout duration, number of satellites tracked). The module also implements a software watchdog to trigger assertions before hardware timeouts, this allows core dump collections before a reboot.

Chapter 4

Signal processing

This chapter outlines the design, development, and deployment of a signal processing algorithm specifically aimed at enabling on-animal classification of sheep behavior using the OVIS device. This embedded system is constrained in terms of computational power, memory, and energy resources, which presents significant challenges for executing data intensive tasks like behavior classification. To address these challenges, the implemented signal processing technique had to strike a careful balance between computational simplicity and algorithmic robustness to ensure reliable performance in farm environments. Implementing such capabilities directly on the OVIS device reduces the dependency on continuous cloud connectivity, lowers data transmission costs, and facilitates faster, autonomous decision making in the field.

The primary goal of this implementation was to evaluate the feasibility of performing on-animal classification and online classification, enabling online inference with minimal delay. Rather than optimizing for maximal classification accuracy, this approach focused on achieving reliable inference under the practical constraints of the embedded system.

A classifier was trained and validated on a PC using a publicly available labeled dataset of sheep motion, and then implemented in the OVIS device. The classifier computes statistical features from a sliding window of acceleration data and outputs behavior predictions in situ. Preliminary validation tests were carried out under representative field conditions, enabling the evaluation of classifier performance in a farm setting with varying terrains, animal postures, and possible collar rotations.

4.1 RF selection

Before implementing the on-device behavior classification, it is important to clarify the underlying ML workflow. The approach used in this work is supervised learning, meaning that the algorithm learns from labeled examples of animal behavior. First, a labeled dataset must be used, where each segment of sensor data is associated with a known behavior. These labeled data are then used to train a model on a PC, where computational resources are abundant. Once the model parameters are learned and determined, the final classifier is exported and executed directly on the OVIS device, which performs prediction using only the features calculated on-animal.

With this workflow in mind, an extensive literature review was conducted to assess existing approaches in animal behavior monitoring. This review, described in Section State of the art from the Introduction chapter, revealed that RF classifiers were com-

monly employed across multiple studies, including [19], [20], [21], [23], and [24]. Their RF adoption is primarily due to its robustness, good generalization capabilities, and relatively low computational cost. These works demonstrated that wearable systems with embedded inertial sensors can successfully classify a range of behaviors in ruminants, including sheep and goats.

In particular, [23] showcased the use of RF models to distinguish between activity levels and postures in ewes and lambs within commercial farming conditions. The study emphasized the necessity of robust feature selection techniques to mitigate the effects of noise and inter-animal variability.

Based on these findings, and aligned with the computational constraints of the OVIS system, the RF technique was selected. RF models consist of multiple DT operating in parallel. Each tree is trained on a random subset of the total available features and independently predicts a class (sheep behavior). The final prediction is determined by selecting the most frequent class prediction among all the trees. This method is particularly suitable for embedded environments where the goal is to obtain reliable predictions with a limited set of features.

4.2 Classifier on the PC

The development and implementation of an end-to-end algorithm for sheep behavior classification on a PC environment served as a precursor to its deployment on the OVIS device. This phase included several stages: studying an external dataset, preprocessing and filtering the data, extracting features, applying feature selection techniques, and optimizing classifier hyperparameters. This process was iteratively refined with focus on balancing accuracy, computational load, and ease of implementation on embedded systems.

All the necessary processing steps were implemented using Python. The scikit-learn library was used, because it offers flexible and a well documented tools for model training, validation, and feature construction. This environment allowed fast prototyping with systematic exploration of parameter combinations and dataset partitioning schemes.

4.2.1 External database description

To train and validate the classifier, the dataset published in [15] was selected. This dataset includes high resolution motion data collected using collar worn sensors comprising a three-axis accelerometer and a gyroscope. The experiment involved 4 goats and 2 sheep and was conducted in a semi-controlled environment to simulate real animal behavior. Data was sampled at a frequency of 200 Hz to capture motion patterns associated with various behaviors.

The original annotations included behavioral states such as lying down, standing, walking, trotting, running, grazing, trembling, scratching, and fighting. However, due to insufficient representation and imbalance among some of these classes, particularly for sheep-specific behaviors like trembling and lying down, certain labels were excluded from further analysis. Five behavior states were retained for sheep: standing, walking, trotting, running, and grazing. These labels represent the starting point for the classification task and are refined in the following subsection.

4.2.2 Behaviors to classify

Sheep behaviors are not mutually exclusive, particularly when locomotion and foraging occur simultaneously. For instance, an animal may graze while moving slowly or re-

main stationary, which introduces ambiguity when labels are treated as discrete classes. To reduce this overlap and simplify the classification problem, grazing was removed from the targets because it overlaps with both static and low intensity movement. Similarly, trotting and running were merged into a single high intensity locomotion category, as their motion patterns are difficult to distinguish reliably.

After these adjustments, the original five labels were consolidated into three locomotion focused classes: still, walking, and running. The standing label was grouped under still to better capture all stationary postures. This reduced label set provides a cleaner behavioral separation and tries to mitigates confusion between similar states.

4.2.3 Dataset limitations

Although the dataset published in [15] provides high resolution motion recordings, its effective statistical support for behavior classification is limited. First, only two sheep (S1 and S2) are available, which inherently restricts the variability in body size, collar fit, and individual locomotion patterns.

A second limitation is the strong imbalance across behavioral labels in the raw dataset. Before any preprocessing, and considering the original five related classes (standing, walking, running, trotting, and grazing), the distribution of samples is highly imbalanced. Table 4.1 shows the number of samples per behavior at the original sampling frequency of 200 Hz. Grazing dominates the dataset, whereas classes such as running or trotting are comparatively rare. This imbalance complicates classifier training, as models tend to overfit the majority class.

Table 4.1: Original sample counts per behavior (200 Hz, no label merging).

Label	S1 samples	S2 samples
grazing	1 238 034	835 304
running	118 613	94 816
standing	650 014	1 165 536
trotting	126 040	110 409
walking	251 593	370 273

Moreover, the recordings had to be downsampled from 200 Hz to 25 Hz (by retaining one out of every eight samples) to match the sampling characteristics of the OVIS device. After applying all these operations, the resulting class distribution is shown in Table 4.2. Despite the reduction in class imbalance, the overall dataset remains relatively small.

Table 4.2: Sample counts after removing certain labels, merging trotting and running, and downsampling to 25 Hz.

Label	S1 samples	S2 samples
standing	81 252	145 692
walking	31 450	46 283
running	30 582	25 655

In summary, although the external dataset is useful for initial prototyping, the combination of limited sample diversity (only two individuals), strong class imbalance in the raw recordings, and the reduced number of samples after downsampling imposes important constraints on classifier generalization. These limitations must be considered when interpreting the results of the ML models trained in this work.

4.2.4 Feature computation

For effective behavior classification, it is important to extract statistical features from raw sensor data. A sliding window approach was adopted to segment time series acceleration data into overlapping segments. Specifically, a window of 125 samples (equivalent to a 5 s window with samples acquired at 25 Hz) was chosen for feature computation.

A 52% overlap (60 sample step size) was used between successive windows to ensure smoother transitions. The features values enable the inference of a sheep's state every 2.4 s.

From each window, nine features were computed for each of the three acceleration axes and for the acceleration vector norm, resulting in a total of 36 features. Additionally, the raw values from each axis and the vector norm were included as four extra variables, yielding a final feature vector of size 40. The extracted features were: mean, minimum, standard deviation, skewness, kurtosis, entropy, zero crossing count, crest factor, and 25th percentile (see Table 4.3 for feature definition). The features were selected based on their relevance in the reviewed studies, while also being computationally simple enough for embedded system implementation.

Table 4.3: Classifier features description.

Feature	**Definition**
Mean	$\mu = \frac{1}{N}\sum_{i=1}^{N} x_i$
Standard deviation	$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$
Minimum value	Minimum value in N samples
Zero crossing	Number of zero crossings in N samples
Crest factor	Maximum value in N samples divided by RMS[I]
25% percentile	Value below which 25% of samples are found
Entropy	$-\sum_{i=1}^{N} p(x_i) log(p(x_i))$ [II]
Kurtosis	$(\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^4)/\sigma^4$
Skewness	$(\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^3)/\sigma^3$

[I] Root Mean Square.

[II] $p(x_i) = \frac{|X_k|}{\sum_{k=1}^{N}|X_k|}$, $X_k = \sum_{n=0}^{N-1} x_n e^{-i2\pi kn/N}$, $k = 0, .., N-1$

4.2.5 Feature selection

Feature selection plays a role in reducing model complexity and improving classification performance. In this work, the K Best algorithm from scikit-learn was employed to select the most informative features. This method ranks features according to their relevance and selects the top k candidates. In this implementation, feature relevance was computed using scikit-learn's default score function for K Best, the ANOVA F-value ($f\_classif$), which quantifies the statistical relationship between each feature and the target.

The selection process was conducted using motion data from the two individual sheep (S1 and S2) in the dataset. The five selected features were: minimum value and standard deviation of the acceleration vector norm, standard deviation and zero crossing count in the Z axis, and standard deviation in the Y axis. These features demonstrated high variance between classes and contributed to classifier performance. Limiting the feature set to five inputs also allowed for more efficient implementation on the OVIS device.

4.2.6 Classification algorithm

With the feature set defined, the classification model was trained using data from sheep S2 as the training source, while sheep S1 was reserved exclusively for testing to ensure subject independent evaluation. A grid search technique, combined with 10-fold cross validation, was employed to identify optimal hyperparameters for the training of the RF model. The parameters varied were: max_depth (the maximum number of splits each tree is allowed to make) and $n\_estimators$ (the number of trees used in the forest). Notably, the $max\_features$ parameter was kept at its default value ('$sqrt$'), meaning that at each split the classifier considers a number of features equal to the square root of the total number of available features, in line with [23]. In the final stage of tuning, tree depths were evaluated from 1 to 9, and the number of trees ranged from 1 to 21. The best model utilized a depth of 6 and 10 trees.

4.2.7 Results

This specific RF achieved a general 98 % accuracy[1]. Figure 4.1 displays the confusion matrix for this RF trained and tested on the PC with the external database from [15].

The trained RF model demonstrated high precision and sensitivity across the three target classes. The majority class, standing achieved excellent metrics, with sensitivity and precision values at 99.2%. Walking was predicted with 99.6% sensitivity but exhibited a slight drop in precision to 95.4%, due to occasional confusion with the running class. The running class presented the greatest challenge, with 3.7% of instances misclassified as walking, yielding a sensitivity of 95.5%.

Despite these minor misclassifications, the model's overall performance was robust. Balanced F1-scores across all classes confirmed the model's generalization capacity. These findings validated both the effectiveness of the selected features and the suitability of the selected hyperparameters.

[1] In the present manuscript most of the tests performed are described in Testing chapter. However it was found pertinent for the results of RF on the PC trained and tested with external database from [15] to be explained here.

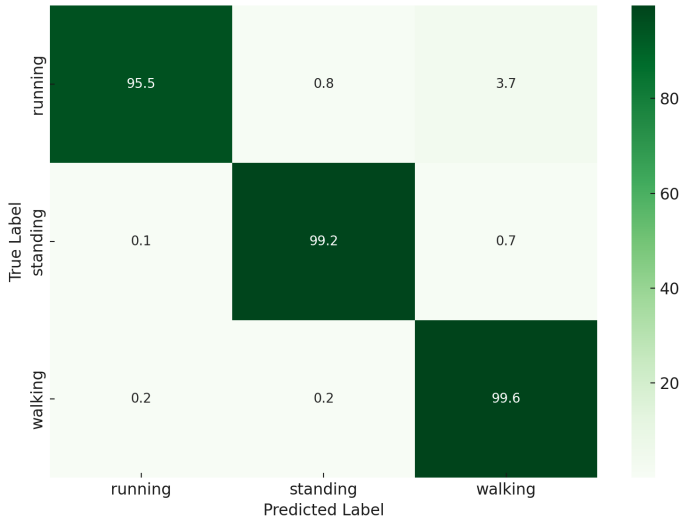


Figure 4.1: Confusion matrix for the RF trained and tested on the PC with external database from [15].

4.3 Classifier on the OVIS device

To integrate the PC trained RF, presented in Section 4.2, into the microcontroller, *Embedded Learn* (*Emlearn*) tool from [54] was employed. *Emlearn* is an open source tool that facilitates the conversion of ML models, such as RF, into C code suitable for execution in resource constrained environments. It provides both the model conversion logic and a lightweight runtime library (**eml_trees.h**), enabling on-animal inference with minimal overhead.

The PC trained RF model, which was validated using external data, was exported from Python using *Emlearn*'s utility scripts. This translation process generated a static C representation of the trees, preserving the model structure and thresholds. The classifier on the OVIS device is referred as the OVIS classifier. This embedded algorithm approach is critical to reduce the volume of data transmitted wirelessly and to extend battery life by avoiding continuous raw data streaming.

4.3.1 C implementation verification

Although the RF model implemented in C numerical output (see Figure 4.2) closely matches the Python version, slight variations in arithmetic or rounding can cause discrepancies in the resulting confusion matrix. Nevertheless, the overall classification accuracy remained consistent, demonstrating the robustness of the *Emlearn* generated implementation.

4.3.2 Embedded software implementation

The embedded software responsible for executing the classifier integrates several steps. Once a new accelerometer sample is acquired, it is scaled from mg to m/s^2 and then inserted into a circular buffer that stores recent samples. This buffer maintains a rolling window of 125 data points, which corresponds to a 5 s window based on the 25 Hz sampling rate established earlier.

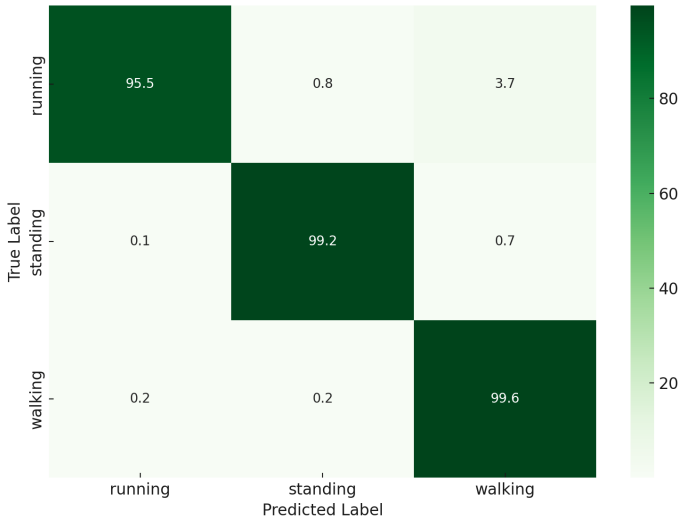


Figure 4.2: Confusion matrix for RF model implemented in C.

To allow overlapping window analysis, the embedded software distinguishes between the first complete window and subsequent ones. After initialization, inference is performed every 60 new samples, resulting in approximately 52% overlap. This decision aligns with the PC implementation and ensures that temporal resolution is preserved.

Upon collecting a valid window, the system computes the five selected features from Subsection Feature selection. These features are passed to the OVIS classifier, which produces a classification label corresponding to one of the three locomotion states: still, walking, or running[2].

The classification result is stored locally in a buffer and marked with a timestamp. The associated raw acceleration data that contributed to the classification is also flagged for optional future transmission. This strategy ensures traceability between raw data and inferred behavior and supports deferred analysis at server level.

This architecture enables on-animal classification and online classification while preserving the ability to analyze or refine performance using raw historical data.

[2]The performance of the OVIS classifier is considered part of the end-to-end test of the OVIS system and is described in Testing chapter.

This page has been intentionally left blank.

Chapter 5

OVIS cloud server

The design and implementation of the OVIS cloud server exceeds the scope of the present thesis. This solution was designed and developed by research partners[1]. Nevertheless, since it is a custom software solution for the OVIS system, the present work includes the specifications and testing for the OVIS cloud server. Additionally, this work covered the correct communication with the cellular network (i.e., NB-IoT and LTE), and specifically with the OVIS cloud server itself (through MQTT over TLS and TCP).

The complete software solution architecture is shown in Figure 5.1. It comprises three main components: the back-end, the front-end, and the database. The back-end includes the MQTT Broker and the Computing resource responsible for processing the messages received from the OVIS devices. The front-end is implemented as a Website, and it serves the OVIS system as the OVIS user interface. This user interface enable users to visualize, monitor and track the collected information online, as well as to access historical data when needed. The database serves as persistent storage, organizing and maintaining all received information in a structured format to support both the OVIS user interface and potential data analysis processes. The entire solution is deployed within the AWS ecosystem as the OVIS cloud server.

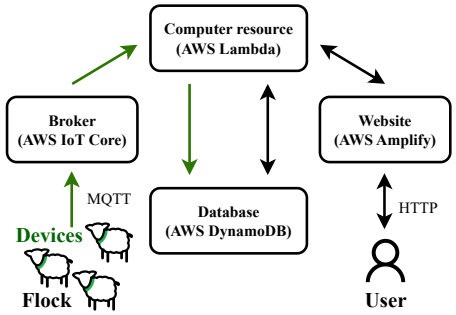


Figure 5.1: Architecture of the OVIS cloud server solution.

[1]The work was carried out as Hernán Cardoso's undergraduate thesis in Computer Engineering, which has been presented in [40], under the supervision of Martín Pedemonte and Julián Oreggioni. Except for the contributions detailed before, it should be assumed that the remaining work in this chapter was led by Varinia Cabrera.

5.1 Backend component

AWS IoT Core enables secure connection of multiple devices to the OVIS cloud server and efficiently routes received messages to other AWS services for further processing. When a device publishes a message to a specific MQTT topic, it is intercepted by the AWS IoT Core message broker. Further information on communication between OVIS devices and AWS IoT Core can be read in Communications.

Using an AWS feature called Rules, these messages are automatically forwarded to AWS Lambda functions without the need for intermediate infrastructure. When a message is received by AWS IoT Core from an OVIS device, AWS Lambda automatically triggers a function to handle it. This function runs inside a temporary computing environment based on Node.js (a platform used to execute JavaScript code). AWS Lambda creates a new instance of this environment for each incoming message, allowing the function to process data independently. As a result, even if many messages are received simultaneously, AWS Lambda can handle them in parallel by launching multiple environments. This execution model enables the OVIS system to process a high volume of incoming messages, ensuring scalability and reducing the risk of performance bottlenecks.

5.1.1 Communications

For OVIS devices to communicate with OVIS cloud server, they must have successful communication with the cellular network and with AWS IoT Core message broker itself. The following subsections explain the particularities of these communications.

Connecting to the cellular network

The nRF9160 and the Icarus IoT Board support both LTE and NB-IoT technologies. These technologies are also supported by Nordic's Asset Tracker v2 example and by ANTEL. Despite this, establishing communication between an OVIS device and the cellular network involved several challenges.

To ensure reliable communication between the OVIS device and the cellular network, it is necessary to apply several custom configurations in the Kconfig options of the OVIS application[2].

The Icarus IoT Board comes with an Embedded SIM (eSIM) which provides LTE and NB-IoT and can be activated by registering the Icarus IoT Board on an Actinius platform. After a free plan for three months, a paid plan can be purchased. However, this feature is only functional in certain selected service areas in the European Union and United States. Therefore, external SIMs are required for OVIS device communication. The Icarus IoT Board comes with an external nano SIM holder. The need for an external SIM meant obtaining LTE and NB-IoT service from ANTEL. The external SIM was activated through a Kconfig option.

Acquiring cellular service from ANTEL involved other, non-trivial configurations, for example, the need for legacy Protocol Configuration Options (PCO) support. PCO is a set of parameters exchanged between the device and the network when a Packet Data Network (PDN) connection is established. These parameters help configure IP addresses or Domain Name System (DNS) servers. Legacy PCO support refers to enabling compatibility with older or non-standard implementations of these configuration options. ANTEL's IoT network still relies on legacy PCO formats instead of newer standardized methods. Legacy PCO support was enabled through a Kconfig

[2]For more information about Kconfig options see General characteristics from Zephyr RTOS section.

option. Also, the APN was explicitly set to ANTEL's IoT service using a specific Kconfig option.

To improve connection stability and avoid network scanning delays, the Public Land Mobile Network (PLMN) and LTE frequency bands were locked to search immediately for ANTEL's IoT service (bands 3 and 26) using the Kconfig options.

Moreover, other configurations were recommended by Actinius. For example, OVIS devices were configured to operate exclusively in NB-IoT mode with GPS support and to prioritize NB-IoT networks. Also, dual-stack IP was configured.

These tailored settings were essential to adapt the OVIS application to the local network conditions and ensure reliable end-to-end communication between the OVIS device and the OVIS cloud server.

Connecting to AWS IoT Core

OVIS devices connect to AWS IoT Core over a secure TLS and TCP connection using the MQTT transmission protocol. The secure communication to AWS IoT Core involves both authentication and authorization mechanisms. Authentication verifies a device's identity; in this case, MQTT authentication uses mutual TLS with X.509 certificates.

X.509 is a standard that defines the structure of public-key certificates commonly used in internet security protocols such as TLS. Each certificate includes a public key, identity attributes, and a digital signature generated by a trusted Certificate Authority (CA). This signature is what allows a device to verify that the certificate (and therefore the entity presenting it) is legitimate. These elements enable both ends of the connection to prove their identities during the TLS handshake.

In a mutual TLS setup, authentication works in both directions. The device authenticates the server by checking that the server certificate was issued by a trusted CA whose certificate is stored locally on the device. Conversely, AWS authenticates the device by validating the device's certificate when it is presented during the handshake.

Conceptually, the CA certificate enables the device to confirm who the server is, while the device certificate (formed by a public–private key pair) allows AWS to confirm who the device is. The public key can be shared freely and is uploaded to AWS IoT Core as part of the device identity. The private key, however, remains securely stored on the device and never leaves it. Only the private key can decrypt data or sign messages generated using the corresponding public key, making it the critical element that proves the device's authenticity.

To generate the necessary certificates (CA certificate and public–private key pair) for a single OVIS device, a *Thing* (endpoint) can be created in AWS IoT Core. Certificates can then be generated and associated with that *Thing*, and subsequently configured on the OVIS device. The Cellular Monitor App within the nRF Connect SDK from Nordic Semiconductor is used to provision the certificates onto the device. Additionally, the embedded software configuration of the OVIS device must include the *Thing* name and the broker's IP address.

Authorization defines what the OVIS device is allowed to do once authenticated. AWS IoT Core policies are created to assign permissions, typically to a *Things Group* (a collection of devices that can share common permissions).

In the case of the OVIS application, specific configurations must be declared in the embedded software to establish secure MQTT communication with AWS IoT Core. These include the security tag, the broker host name, and the client ID. The security tag indicates the identifier used to access the correct credentials stored in the device, while the broker host name and client ID ensure that each device can be identified by the MQTT Broker. These are configured using Kconfig options.

All these configurations are essential for establishing secure and reliable communication between each OVIS device and AWS IoT Core.

5.2 Database component

AWS DynamoDB was chosen to host OVIS database mainly to ensure scalability. AWS DynamoDB is a fully managed NoSQL database service that uses a key value model rather than a traditional tabular structure. Moreover, the design enables the system to adapt dynamically to changing data formats without requiring manual modifications or database migrations. These features are useful when incoming messages from devices can vary in structure and may not always contain the same set of fields.

Furthermore, AWS DynamoDB is optimized for high write throughput. This aligns with the needs of the OVIS application, since it prioritizes storing frequent incoming messages from multiple devices over complex querying.

The described auto scaling capabilities of AWS DynamoDB ensure that the database can automatically adjust capacity in response to varying loads, maintaining consistent performance even as the number of devices or the volume of messages increases.

5.3 User interface

OVIS cloud server also hosts a web based application (implemented using AWS Amplify) that enables users to interact with the data stored in the back-end. This website was developed using NextJS 13, a modern React framework that supports Server-Side Rendering (SSR). SSR plays an important role in improving performance and compatibility. By generating HTML on the server before sending it to the client, SSR ensures faster initial page loads and reduces the computational burden on the user's device. This is important for the OVIS system, where users rely on a wide range of devices, including older or resource constrained ones. The following paragraphs describe and illustrate, through figures, the different parts of the OVIS user interface.

5.3.1 Login and home screen

At first glance, the website displays the login page, shown in Figure 5.2. On the left side, a pair (username and password) can be entered to access the rest of the user interface.

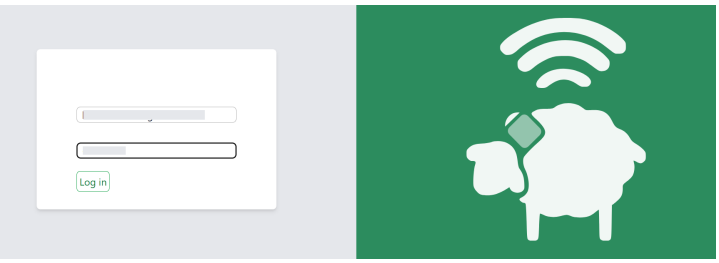


Figure 5.2: Login screen.

Once inside, the home screen is presented (Figure 5.3). It is worth mentioning that most of the OVIS user interface is in Spanish, as it was tested in Uruguay with

Uruguayan researchers[3]. The home screen shows a navigation list on the left side (number 1 in Figure 5.3). This list can be hidden by clicking the button with horizontal bars in the top left corner (number 2 in Figure 5.3). This screen also displays a small summary, indicating how many OVIS devices are enabled, have reported GNSS data, and have predicted behavior in the last day (number 3 in Figure 5.3). Additionally, a brief user guide explaining some interface functions is available (number 4 in Figure 5.3).

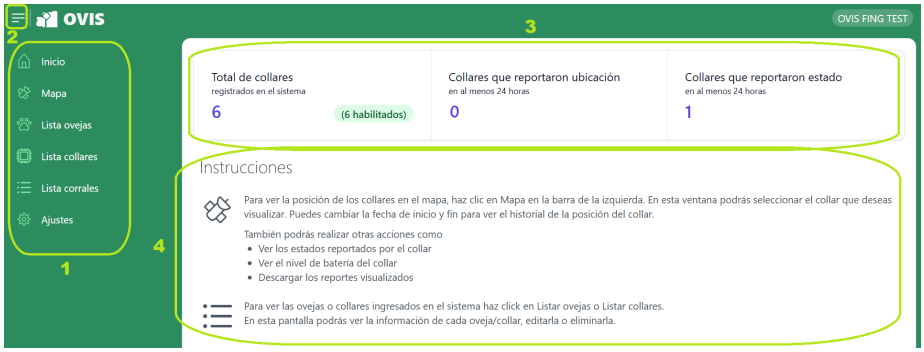


Figure 5.3: Home screen.

5.3.2 Map tab

By pressing the button labeled 'Mapa' (Map in English, number 1 in Figure 5.4), a list of all enabled OVIS devices is shown (number 2 in Figure 5.4). In this list, only active OVIS devices appear in green, displaying information such as battery level (in mAh), RSRP (in dBm), time of the last data report to the OVIS cloud server, and time of the last known location.

When an active OVIS device is selected (e.g., 'Collar B11' in Figure 5.4), it is possible to navigate through different types of data using another menu (number 3 in Figure 5.4). This menu allows users to view 'Mapa' (location data), 'Batería' (battery level), 'Actividad' (behavior data), 'Collar' (collar information), and 'Resumen' (data summary).

All data shown through this menu is intended to be viewed over a specific time period, which is displayed in blue letters below the menu (number 3 in Figure 5.4). This time window can be changed by clicking the blue text, which opens a pop-up window. The pop-up window is shown in Figure 5.5 and is designed to select the desired time window.

Location data

At the time the pictures were taken, the active OVIS device had location data requests disabled (Figure 5.4). However, this feature is shown in Figure 5.6, which displays an earlier version of the OVIS user interface.

Battery level

Pressing the 'Batería' (Battery) button displays the battery level evolution over time (Figure 5.7).

[3]It is worth mentioning that the language can be changed and reprogrammed if necessary.

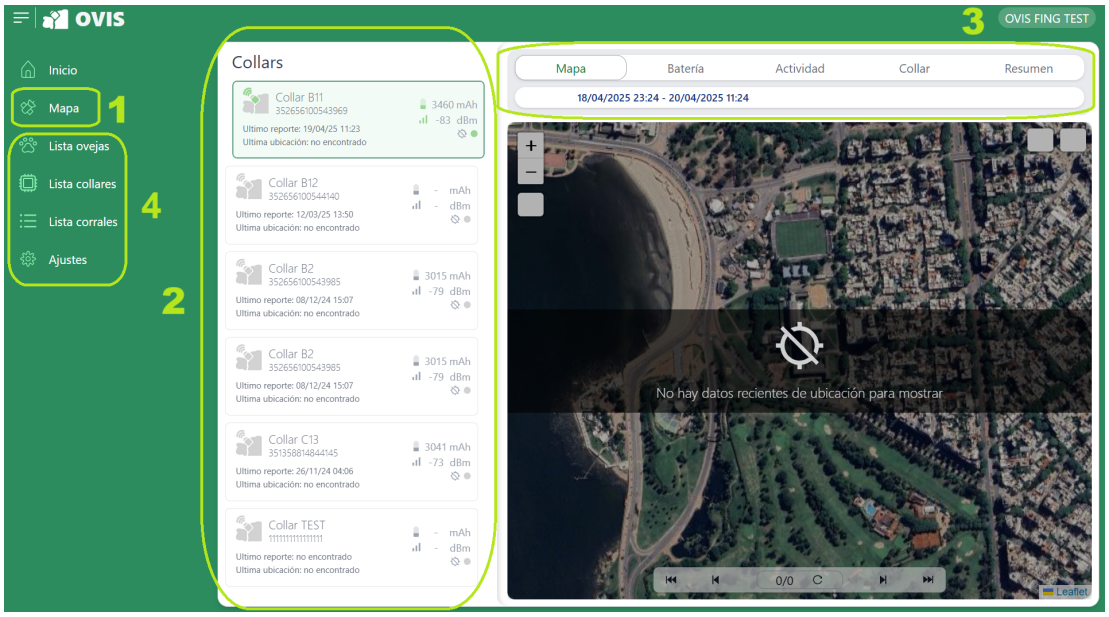


Figure 5.4: Location map.

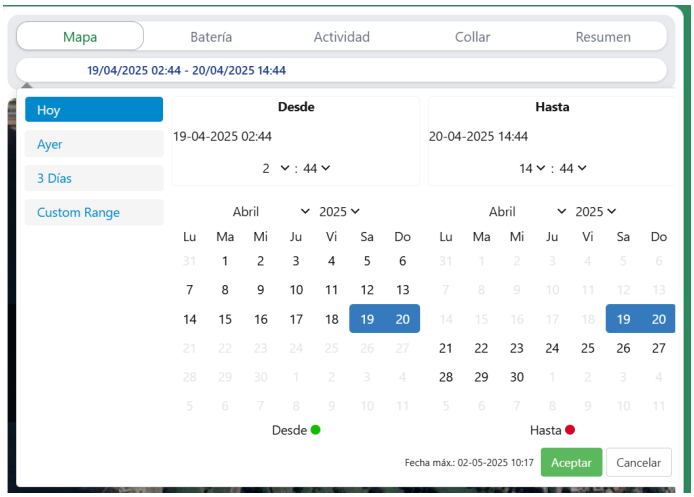


Figure 5.5: Pop-up menu to select time window.

Behavior information

Another useful dataset can be viewed under the 'Actividad' (Activity) tab. This section shows the behavior predicted by the active OVIS device over a given time period. Figure 5.8 shows an animal standing (this behavior was predicted while the OVIS device was resting on a table for an extended period). This tab also includes a pie chart illustrating the distribution of different predicted behaviors, as an example of possible information that can be displayed.

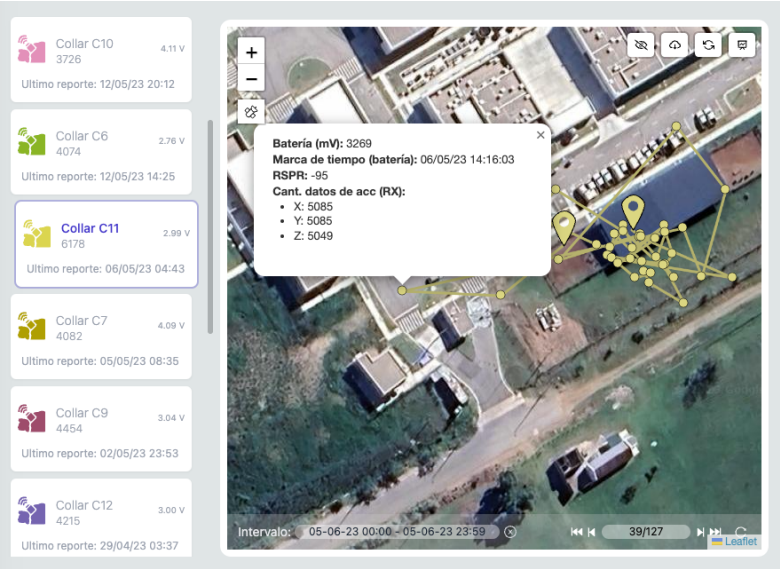


Figure 5.6: User interface showing sheep location data. Figure taken from [42].

Collar information and summary

The 'Collar' tab provides general information (Figure 5.9) about the device, including its name, IMEI, and the animal currently wearing it. This is extremely useful since a single OVIS device may be worn by different sheep at different times. Identifying the animal currently associated with the device helps researchers accurately interpret the collected data.

The 'Resumen' (summary) tab provides a report summary from the selected OVIS device over a specified period of time. For example, in Figure 5.10, the B11 OVIS device reported: 825 battery level records, 0 location data records (since the feature was disabled), 828 acceleration records, and 828 predicted behavior records.

This tab also allows users to export the data generated by the selected OVIS device over a specified period in Comma Separated Values (CSV) format. Users can filter to download only battery levels, location data, or predicted behavior over time. Exported data may optionally include raw acceleration data.

5.3.3 Other tabs

In addition to the 'Inicio' (home screen) and 'Mapa'(map) tabs, other options in the list displayed on the left side (number 4 in Figure 5.3) include: 'Lista ovejas' (sheep list), 'Lista collares' (collars list), 'Lista corrales' (pen list), and 'Ajustes' (settings).

The sheep, collars, and pens lists are used to track which OVIS devices are assigned to which animals and pens. As mentioned earlier, it is important for researchers to know which animal a particular dataset corresponds to, and which pen the animal belongs to. These tabs facilitate the registration and tracking of the device-animal-pen association. As these interfaces are very simple figures are omitted.

The settings tab is also basic (figure omitted), and allows users to change the name of the farm or establishment. Similar to the pie chart in Figure 5.8, this section was included as an example of possible elements that might be included in the user interface.

Figure 5.7: Battery evolution over time.

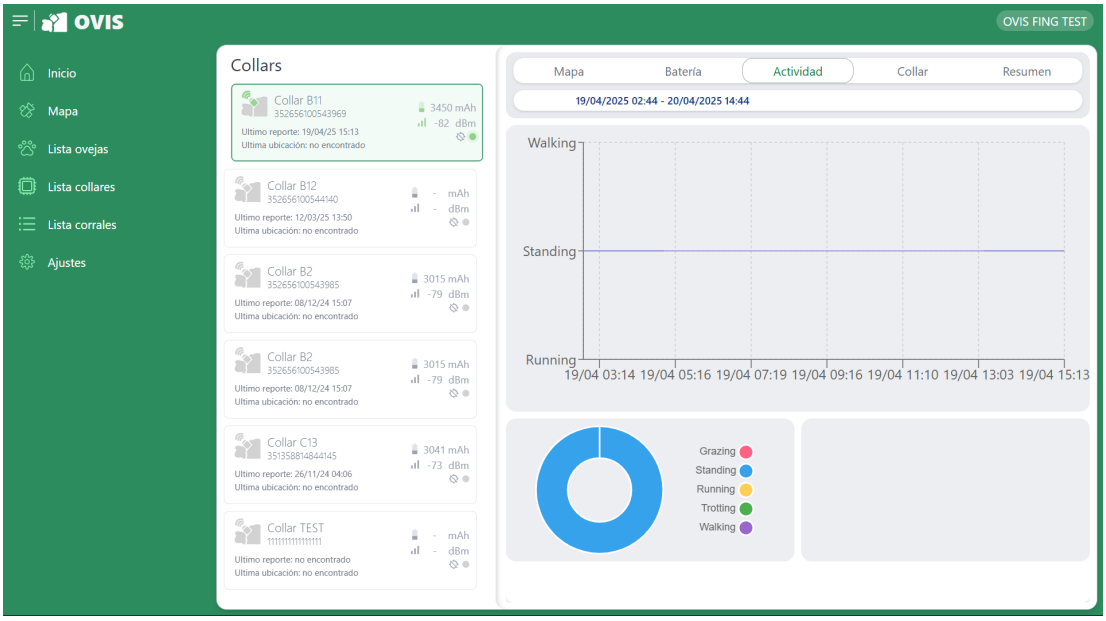


Figure 5.8: Possible sheep behavior analysis.

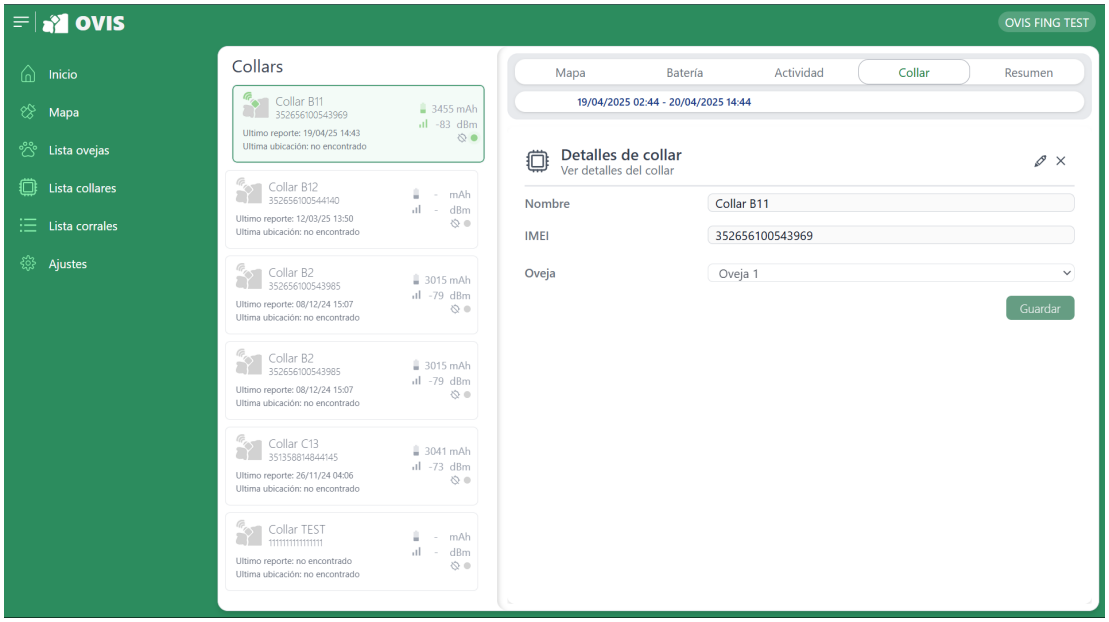


Figure 5.9: Collar information displayed.

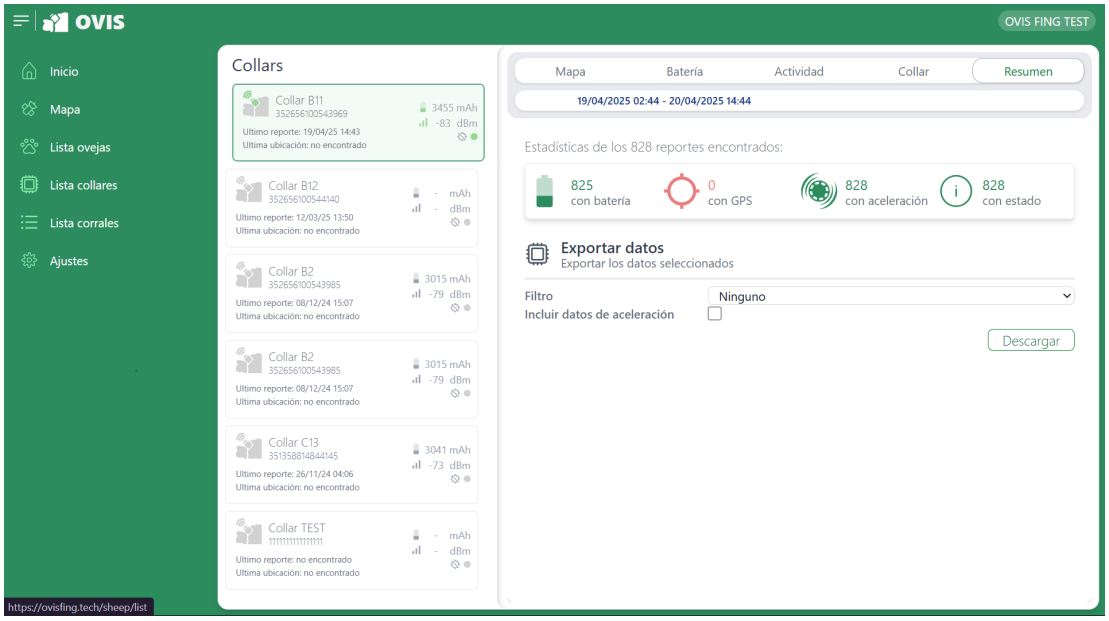


Figure 5.10: Data summary and export possibilities for an OVIS device.

This page has been intentionally left blank.

Chapter 6

Mechanical design

This chapter outlines the process of designing, fabricating, assembling, and testing the OVIS device's enclosure[1]. The next sections describe the requirements definition, mechanical testing[2] and proposed design.

6.1 Requirements

The mechanical design of the OVIS device's enclosure was guided by a set of functional, structural, and animal welfare requirements. Table 6.1 summarizes the high level requirements that the enclosure must satisfy. The remainder of this section provides a detailed explanation of each requirement, including the underlying motivations and design considerations.

6.1.1 Functionality and usability

A main design requirement is that the enclosure must securely contain, protect, and allow the correct functioning of all components selected in the hardware design (see Chapter 2). This includes the solar panels, battery, Icarus IoT Board, GNSS antenna, and cellular antenna. The mechanical housing must therefore provide mechanical protection, environmental sealing, and appropriate placement and orientation of each component so that electrical performance, charging capability, and signal reception are not compromised.

To facilitate charging the battery and programming of the Icarus IoT Board, a USB port was included in the design. This required the creation of a side lid or USB cover. This cover protects the USB port from environmental factors, while still providing easy access. This cover is shown in Figure 6.9.

[1] The design requirements were defined by OVIS research team (engineers and veterinarians) in collaboration with the DVL group company, with guidance from its Director, Diego Fraga. The mechanical design and prototype fabrication stage of the OVIS device's enclosure were carried out by DVL. The main fabrication stage was performed by the company VegaDS (Joaquín Vega). The solar panel soldering and component assembling on the enclosure were performed by Lucía Sirio. The testing of the OVIS device's enclosure was carried out by the OVIS research team, DVL, and Lucía Sirio. The entire process was supported by the OVIS research team and its input as a client. Except for the contributions detailed before, it should be assumed that the remaining work in this chapter was led by Varinia Cabrera.

[2] In the present manuscript most of the tests performed are described in Testing chapter, however it was found pertinent for the mechanical testing to be explained here.

ID	Requirement	Description
R1	Functionality and usability	Must contain, protect, and allow correct functioning of electrical components.
R2	Durability	Must withstand impacts, rain, dust, and long term use in farm environments.
R3	Weight	Total enclosure + collar weight $< 500\,\mathrm{g}$ for animal welfare.
R4	Color selection	Avoid attracting predators and overheating while maintaining visibility.
R5	Collar adjustment	Collar must remain stable on the neck (no sliding or rotation) while adapting to wool and neck-size changes.
R6	Enclosure sizes	Multiple sizes needed to accommodate different morphologies (breed, sex, age).

Table 6.1: Summary of mechanical design requirements for the OVIS device's enclosure.

The antennas used by the OVIS device (i.e., the GNSS antenna and the cellular antenna) must be positioned upwards towards the sky, ensuring optimal signal reception. Therefore, the antennas were placed without solar panel obstruction to avoid interference. In addition, the orientation of the accelerometer is also important. For accurate behavior classification, the accelerometer must remain fixed relative to the sheep, ensuring that the measured axes consistently reflect the animal's movements. Any rotation or instability in the enclosure would distort the acceleration signals and degrade classification performance.

6.1.2 Durability

The OVIS device's enclosure had to be resistant enough to withstand adverse weather conditions. It also had to endure impacts resulting from interactions with the animals. This was crucial to ensure the longevity under typical conditions found in a farming environment. The design took into account environmental factors, including exposure to sunlight, rain and dust.

6.1.3 Weight

Regarding animal welfare, the device design must consider its weight, volume, and color. The primary goal for the case's weight was to ensure it did not exceed 500 grams, which was considered the upper limit by OVIS's veterinary researchers to prevent discomfort for the animals. This weight constraint ensured that the device could be worn by sheep without affecting their mobility or overall well-being.

6.1.4 Color selection

Color played an important role in the design process, particularly for visibility purposes. In the initial stages, it was decided to use fluorescent or bright colors to ensure the device would be easily visible in the field, especially when the animals were grazing

in open areas. However, it soon became apparent that these bright colors could attract predators, posing a threat to the sheep safety. As a result, the design team opted for neutral colors, such as white or light gray, which would not attract predators, while still being easily visible in a variety of environmental conditions. The use of black was avoided because it has a tendency to absorb heat, which could be detrimental to the sheep, particularly in warmer climates.

6.1.5 Collar adjustment

The collar must remain firmly positioned in the sheep's neck and should not slide downwards, as such movement would alter both antenna orientation and accelerometer alignment. However, a significant challenge in the design process was accommodating variations in neck size due to wool growth and shearing, as well as the presence of lanolin fat on the sheep's fur. These factors could affect the fit of the collar, making it difficult to ensure a secure and comfortable fit for all animals. To address these, an adjustable collar system was incorporated, allowing for different neck sizes to be accommodated. Several buckle types were evaluated and can be seen in Figure 6.1, the first prototype had the black classic buckle.

Figure 6.1: Primary evaluation for buckles to use in the adjustable collar system.

6.1.6 Enclosure sizes

Variations in the sheep's morphology, including age, sex, and breed, needed to be considered. So, multiple sizes of enclosures were designed. This approach ensured that the device would fit securely on sheep of various sizes, preventing the collar from rotating or becoming uncomfortable.

6.2 Tests

Several tests were conducted throughout the OVIS device's enclosure development process to evaluate the mechanical performance of the enclosure and collar system before and under controlled conditions[3]. These tests aimed to validate the requirements defined in Section 6.1. Each test is therefore associated with one or more requirement. The design was iterated until reaching a proper enclosure, meeting the requirements explained before.

The tests are presented in chronological order to reflect the iterative nature of the design process. This approach helps illustrate how each test informed the next,

[3]All animal experiments conducted within the framework of this thesis, which involved the recording of sheep behavior and spatial dynamics, did not involve any stressful or painful handling. All procedures were approved by the the Ethics Committee (CHEA) of Facultad de Veterinaria, Universidad de la República under Protocol CHEA 1030/20, "Validación de dispositivos electrónicos para el estudio del comportamiento animal" (from March 2020 to February 2025).

allowing for continuous improvements and refinements to meet the final functional and operational requirements.

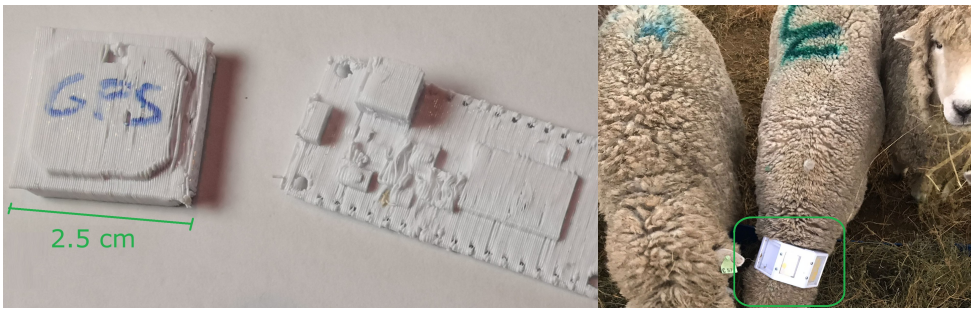
6.2.1 Hermeticity test

A hermeticity test was performed during the OVIS Device Assembly (which is explained in Appendix OVIS Device Assembly). The hermeticity test involves exposing the OVIS device's enclosure directly to a constant flow of water, as shown in Figure C.2. This test is done to confirm the electronics and inside of the enclosure are protected when exposed to liquids. By doing so, the test directly verifies compliance with the durability requirement (R2), ensuring resistance to environmental conditions, and with the functionality and usability requirement (R1), which demands that internal components remain operational and safeguarded under typical farm conditions.

6.2.2 First stage field tests

In the initial stage, the tests were exploratory to assess the new design. The goal was to characterize which elements of the design were acceptable and which needed to be improved for iterative redesign.

At first, there was no knowledge of how the animals would react to wearing the OVIS device's enclosure. Moreover, if the enclosure could endure impacts caused by the animals. Furthermore, despite the fact the enclosure already passed an hermeticity test, adverse weather conditions could make the enclosure's inside vulnerable to water and dirt, making possible to damage the OVIS device's electronics. For these reasons a mock collar was found helpful to be used in the tests (i.e., plastic enclosure with internal weights, no electronics). Figure 6.2a shows the non-functional internal weights.



(a) Fake collar components. (b) Mock collar on sheep.

Figure 6.2: Mock collar used in initial tests without risking the OVIS device's electronics.

Mock collar initial test

A detailed summary of this initial mock collar test is provided in Table 6.2. It was performed with a mock collar on a female Corridale sheep at FVET's[4] research farm[5] to evaluate the mechanical attachment and comfort. Figure 6.2b shows the mock collar being worn by a sheep. FVET team monitored the fit, with periodic checks planned every 8–12 hours. Then, an analog test was performed on a male sheep the following day.

[4]FVET is the Facultad de Veterinaria, Universidad de la República.
[5]All the test conducted at FVET's research farm is done in Corridale sheep.

These tests confirmed the basic suitability of the collar structure, meaning the collar could be securely attached to the sheep without causing visible discomfort. Furthermore, the collar showed no movement relative to the animal. This is crucial because the solar panels and antennas must face the open sky, and the accelerometer must remain fixed (relative to the sheep) for accurate measurements.

Duration	**Collar**	**Participants**	**Animals**	**Location**	**Goal**
2 days	Mock	DVL, IIE, FVET	1F, 1M	FVET research farm	General fit and comfort
Related requirements					
R1, R5					
Results					
Basic suitability confirmed.					

Table 6.2: Mock collar initial test. DVL: design team; IIE: electrical engineering team; FVET: veterinarian team; F = female, M = male.

First OVIS device overall test

Immediately after the first test a decision was made to install two OVIS devices (i.e., with electronics) on the same female and male sheep (for more information on the electronics results see the Testing chapter). The main outcomes of this test are summarized in Table 6.3. The collar on the female sheep performed successfully, staying on and without collar movement relative to the animal. Figure 6.3 displays a photo compilation of the female sheep wearing the OVIS device.

Whereas, the collar on the male sheep fell off and was found detached from the animal. Male sheep are more confrontational, this meant more impacts between animals, showing the need for a more robust enclosure. Furthermore, male sheep are bigger, this raised the issue about the need of different collar sizes.

Duration	**Collar**	**Participants**	**Animals**	**Location**	**Goal**
1 day	w/electronics	DVL, IIE, FVET	1F, 1M	FVET research farm	Overall test
Related requirements					
R1, R2, R5					
Results					
Female collar stayed on while male collar fell off.					

Table 6.3: First OVIS device overall test.

Figure 6.3: Photo compilation of the first OVIS device being worn by a female sheep.

6.2.3 Second stage field tests

The second stage was characterized by an iterative redesign process, with tests conducted over a longer period than in the first stage. Furthermore, every second stage test marked problems or risks in the original design to be corrected.

General robustness and attachment evaluation

After the overall functionality test, it was decided to leave the OVIS device on the female sheep for an extended period (i.e., two weeks). During this period, the system operated as expected: the device continued transmitting data normally, and its performance could be monitored through the OVIS cloud server. While these two weeks passed, a retest was conducted using a mock collar on the male sheep to adjust the collar's fit without causing visible discomfort. The main findings of this evaluation are summarized in Table 6.4.

After the two week period, the OVIS device worn by the female sheep stopped reconnecting to the network, indicating a loss of communication. This event prompted an inspection of the OVIS device worn by the female sheep and on the mock collar worn by the male sheep.

Visual analysis confirmed that the mock collar had a broken buckle (see Figure 6.4a). This result raised the issue of changing the buckle used for the OVIS device enclosure. The buckle was changed for a brand new and the mock collar was installed again on the male sheep. Based on the buckle problem, a new market research was made by DVL, resulting in finding the buckle shown in Figure 6.4b. This buckle is the one used in the final version of OVIS device enclosure.

Duration	**Collar**	**Participants**	**Animals**	**Location**	**Goal**
2 weeks	Mock	DVL, IIE, FVET	1M	FVET research farm	Durability and attachment
Related requirements					
R2, R5					
Results					
Mock collar was detached and buckle was broken.					

Table 6.4: Attachment evaluation.

(a) Detached mock collar from male sheep.

(b) Buckle of final OVIS device enclosure.

Figure 6.4: Broken mock collar buckle (left) and new buckle used in the final enclosure (right).

Moreover, analysis done on the OVIS device installed on the female sheep reveled rain intrusion in the OVIS device's interior. The main observations from test are summarized in Table 6.5. This result was fairly unfortunate, however, the electronics were not damaged. The main hypothesis focused on the USB lid being wrongly screwed.

Thus, the USB lid was screwed correctly and the lid was also taped just in case. It is important to mention that heavy rain was expected in the following days.

Duration	Collar	Participants	Animals	Location	Goal
2 weeks	w/electronics	DVL, IIE, FVET	1F	FVET research farm	Extended period robustness
Related requirements					
R1, R2					
Results					
Rain intrusion in the OVIS device's interior.					

Table 6.5: Robustness and durability evaluation.

Hermeticity, robustness and attachment evaluation

In a follow-up test, the OVIS device remained attached to the female sheep during rain, showing no water ingress. The main observations from this evaluation are summarized in Table 6.6. Figure 6.5 shows a photo compilation, the sheep is seen fairly wet and in the first picture (left to right) rain drops can be seen on the OVIS device. However, rain drops were confirmed to be on the outside of the OVIS device since there was visual confirmation that solar panels in the inside were dry (rest of the pictures in Figure 6.5). Extended testing for several weeks revealed the female collar resisted weather conditions.

Figure 6.5: Post rain test confirming OVIS device resistance.

Duration	Collar	Participants	Animals	Location	Goal
3–4 weeks	w/electronics	DVL, IIE, FVET	1F	FVET research farm	Hermeticity over extended rain period
Related requirements					
R1, R2					
Results					
No water intrusion in the OVIS device's interior.					

Table 6.6: Hermeticity evaluation.

It was confirmed that the male's collar failed early because the sheep managed to remove it, even though the buckle itself did not break. The main findings of this test are summarized in Table 6.7. This indicated a fastening and adjustment issue rather than a structural failure, highlighting the need to improve the attachment robustness.

Size and wool influence

Building on the previous failed test on the male sheep (where the animal removed the collar due to a fastening and adjustment issue), an improvement was implemented

Duration	Collar	Participants	Animals	Location	Goal
Short term (early failure)	Mock	DVL, IIE, FVET	1M	FVET research farm	Durability and attachment
Related requirements					
R2, R5					
Results					
Collar failed early without breaking the buckle.					

Table 6.7: Robustness and attachment.

to reinforce both the buckle and the adjustment system. However, a concern related to sheep size remained unresolved. To address this, a test was conducted on a small, sheared male sheep to evaluate the influence of body size and wool. The main outcomes of this evaluation are summarized in Table 6.8. In this test, the collar initially broke due to a plastic component failure, but after replacement, the fit remained stable.

Duration	Collar	Participants	Animals	Location	Goal
Short term	Mock	DVL, IIE, FVET	1M	FVET research farm	Evaluate influence of size/wool
Related requirements					
R2, R5, R6					
Results					
Successful testing confirmed no movement.					

Table 6.8: Size and wool influence.

Different sizes test

Focusing on collar size and morphology, short series tests were carried out over several months with different collar sizes to assess fit across various morphologies. The main outcomes of these tests are summarized in Table 6.9. Adjustments improved performance in diverse animals. Tests took place at FVET and included enclosure redesigns to create the proposed sizes explain in Enclosure sizes. A common problem during these months was collar rotation and loose attachment, this is shown in Figure 6.6.

Figure 6.6: Field observations where clear OVIS device rotation can be seen.

Duration	Collar	Participants	Animals	Location	Goal
Several months	Mock and w/electronics	DVL, IIE, FVET	Several animals	FVET research farm	Different size design
Related requirements					
R1, R5, R6					
Results					
Three sizes were created.					

Table 6.9: Different sizes test.

6.2.4 Third stage field tests

The third stage aimed to validate the final design, ensuring that the common problems found in the first and second stages were fully resolved. The third stage meant different animals breeds, different facilities and several collars installed at the same time during an extended period of time.

Útica Farm trial

Seven collars were installed at Útica farm on East Friesian sheep (see Figure 6.7). A summary of this design validation test is presented in Table 6.10. After an adaptation period, some collars rotated due to looseness or accidental button presses during feeding. After adjustments, the collars performed successfully for two weeks without further rotation. Animals were kept in pens at night and pastured during the day[6].

Figure 6.7: Compilation pictures from Útica farm trial.

Duration	**Collar**	**Participants**	**Animals**	**Location**	**Goal**
2–3 weeks	w/electronics	DVL, IIE	Several animals	Útica farm	Final design validation
Related requirements					
R1, R2, R5, R6					
Results					
Successful design validation, some improvement suggestions for future versions.					

Table 6.10: Útica test summary.

FVET final trial

Collars were installed on five animals (see Figure 6.8) to be removed after 2–4 weeks of continuous use to evaluate long term mechanical wear and general attachment integrity. The main outcomes of this final trial are summarized in Table 6.11. No structural damage was observed, validating the design under prolonged use.

Duration	**Collar**	**Participants**	**Animals**	**Location**	**Goal**
2–4 weeks	w/electronics	DVL, IIE, FVET	Five animals	FVET research farm	Final design validation
Related requirements					
R1, R2, R5, R6					
Results					
Successful design validation.					

Table 6.11: Final test summary.

[6]Notably, user feedback from Ana María García contributed to design improvements.

Figure 6.8: Compilation pictures from final FVET trial.

6.3 Proposed design

Figure 6.9 presents the proposed design for OVIS device's enclosure and its different parts. OVIS device's enclosure was designed with two main objectives: ensuring proper functioning and addressing the well-being of the animals.

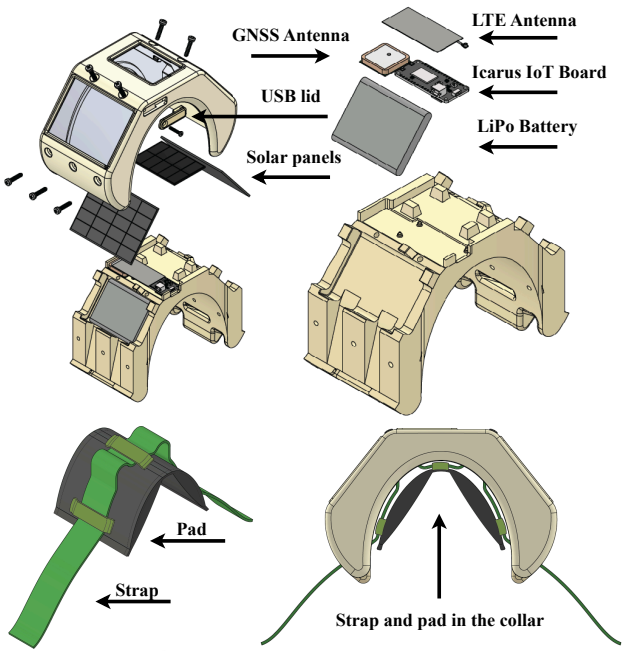


Figure 6.9: OVIS device's enclosure: case with assembly details (left, up), bottom case with internal components (right, up), strap and pad (left, down), strap and pad in the complete collar (right, down). Figure taken from [42].

6.3.1 Sizes

The following variations were implemented: A small enclosure with a filled pad for smaller animals, a small enclosure with an empty pad for medium sized animals and large enclosure with an empty pad for larger animals.

These different sized enclosures allowed for proper collar fixation, ensuring that the device remained in place, regardless of animal's size. This solution was crucial for ensuring the stability of the system across various sheep breeds and sizes.

6.3.2 Quantity

The OVIS project aimed to produce a total of 30 devices, successfully developed in two rounds: short series and "long" series. First 5 OVIS devices, as a first prototype, were created and used to perform preliminary tests and review the design, ensuring that any necessary improvements or adjustments could be made before proceeding any further. 25 OVIS devices were produced in the second round. These devices were part of the "long" series phase, which aimed to finalize the design and confirm that the device met all functional and operational requirements.

6.3.3 Technology and materials

Because of the quantity needed, the Fused Deposition Modeling (FDM) 3D printing technology was chosen, with the addition of transparent polycarbonate windows in the solar panel areas. Production was carried out using polyethylene terephthalate glycol-modified (PETG) filament, chosen for its high mechanical resistance. This was complemented by an internal cellular structure, which allowed weight reduction without compromising strength, achieving a weight of 480 grams.

6.3.4 Assembly process

A proper assembly process ensures a reliable design and directly impacts functionality. In Appendix OVIS Device Assembly a general summary of OVIS device assembly process is presented: it outlines the components, materials, procedures, and considerations.

This page has been intentionally left blank.

Chapter 7

Testing

This chapter presents the experimental end-to-end evaluation of the OVIS system, integrating sensing, local processing, wireless communication, and animal deployment. First, the OVIS device autonomy is characterized including a detailed analysis. Other electronics characteristic are reported: the positioning GNSS is assessed under different conditions, also the reliability of OVIS device–to–OVIS cloud server communication is studied across rural, suburban, and urban scenarios. Finally, the performance of the embedded behavior classification algorithm is evaluated using annotated field data from sheep. These tests were performed iteratively in multiple settings: at Facultad de Ingeniería, Universidad de la República and Facultad de Veterinaria's Migues experimental field without animals for controlled verification scenarios. Moreover, tests were performed with animals at Facultad de Veterinaria, Universidad de la República and at Útica farm on Corriedale and East Friesian sheep.

7.1 Autonomy

7.1.1 PSM: low power NB-IoT configuration

Before presenting the autonomy results of the OVIS device, it is important to clarify that all measurements were performed with the device operating over NB-IoT with PSM enabled. Figure 7.1 illustrates the general logic of PSM. After completing a transmission, the modem transitions to a low power state while retaining its registration context with the network.

Since the OVIS system does not require downlink traffic, eDRX is disabled and the AT is set to zero[1], allowing the modem to enter PSM immediately after each transmission.

The expiration of the TAU timer defines the maximum duration the device may remain in PSM without reporting to the network. For the OVIS device, the periodic TAU is configured to 320 hours (approximately 13 days), the maximum value allowed by 3GPP. This choice minimizes unnecessary signaling and is supported by ANTEL. Moreover, the periodic OVIS application transmissions naturally refresh the network registration.

[1]During the first autonomy measurement, the AT was not zero; it used the default value from Asset Tracker v2.

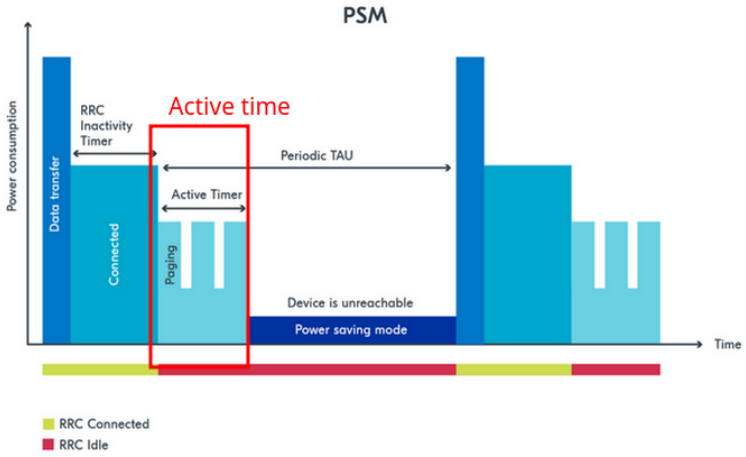


Figure 7.1: General schematic of PSM operation. Adapted from [55].

7.1.2 Autonomy measurement

Figure 7.2 shows a battery discharge cycle reported by three different versions of OVIS device during the experiments on animals. Table 7.1 summarizes main settings of these versions. In version 1 (v1, red line), the solar panels were disconnected, the accelerometer data rate is 25 Hz, and switched off the GNSS. These changes almost double the autonomy compared to the previous published version (blue line) [42]. Preliminary results indicate that GNSS is almost exclusively responsible for the increase. The version 2 (v2, yellow line) shows that reducing the AT to zero increases the autonomy by 20 %, and running the RF behavior classifier has negligible impact. Then, the OVIS device autonomy exceeds 267 hours (11 days), characterized by streaming raw and processed accelerometer data.

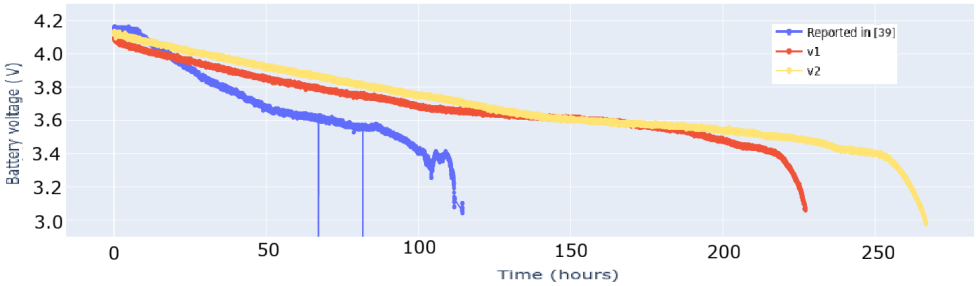


Figure 7.2: Discharge of the OVIS device's for different settings (see Table 7.1)

7.1.3 Consumption profile characterization

Based on the autonomy results presented in the previous section, it becomes interesting to explore the OVIS device's energy consumption in greater detail in order to

Table 7.1: Autonomy of the OVIS device for different settings

OVIS device version	Accel Rate (Hz)	Payload (kB)	GNSS	AT (s)	RF	Solar panel	Autonomy (hs)
v0 [42]	100	45.2	yes	20	no	yes	114
v1	25	11.3	no	20	no	no	227
v2 [43]	25	11.3	no	0	yes	no	267

Note: The payload is quantified in Appendix D.

better understand its operating lifetime. To this end, an extended experimental characterization of the device's current profile was performed. The resulting consumption profile could serve as the foundation for refining the operating modes of the OVIS system and for proposing an optimized configuration suitable for field deployment.

The consumption profile of the OVIS device was measured using the Otii Arc Pro, configured as both power supply and acquisition tool. The device operated at 3.7 V, and current, voltage, and power were sampled at high resolution throughout the experiment. LTE signal quality remained stable (average RSRP of –75.6 dBm), and GNSS was disabled.

Over a total duration of 59 927 s (approximately 16.6 hours), the device exhibited an average current of 12.0 mA, with peaks up to 314 mA. The total energy consumption reached 738 mWh. Given a 50 s transmission cycle, 1 198 uplinks were expected; the OVIS cloud server received 1 155 messages, corresponding to a loss of about 3.6%.

Expected behavior

Figure 7.3 shows a representative portion of the consumption profile under expected operating conditions. The pattern is highly periodic, consisting of short high current transmission peaks followed by low power intervals during PSM. In this segment considered normal or expected behavior, the average current is 11.2 mA, with a maximum of 271 mA, consistent with the overall measurement.

Observed anomalies

Two types of deviations from the expected pattern were identified. These segments represent a small fraction of the total measurement but highlight situations where the OVIS device showed additional consumption values.

The first anomaly (Fig. 7.4a) shows prolonged periods of connection attempts followed by irregular transmission peaks, with an average current of 18.9 mA and peaks up to 304 mA. This pattern is consistent with temporary connectivity issues that force the device to repeatedly attempt network attachment or transmission. Whereas, the second anomaly (Fig. 7.4b) exhibits peak values similar to the normal profile (maximum 274 mA, average 12.3 mA), but with an altered temporal sequence. These events are compatible with possible retransmissions, which introduce irregular spikes without significantly affecting the overall average consumption. Overall, the anomalies appear to originate from two mechanisms: persistent connectivity difficulties and brief recovery retransmissions.

Autonomy estimation based in current profile

Using the measured average current of 12.0 mA, a first estimation of autonomy for the OVIS device can be obtained by combining this value with the nominal 2500 mAh battery capacity. This yields an approximate operating lifetime of 208 hours (∼8 days

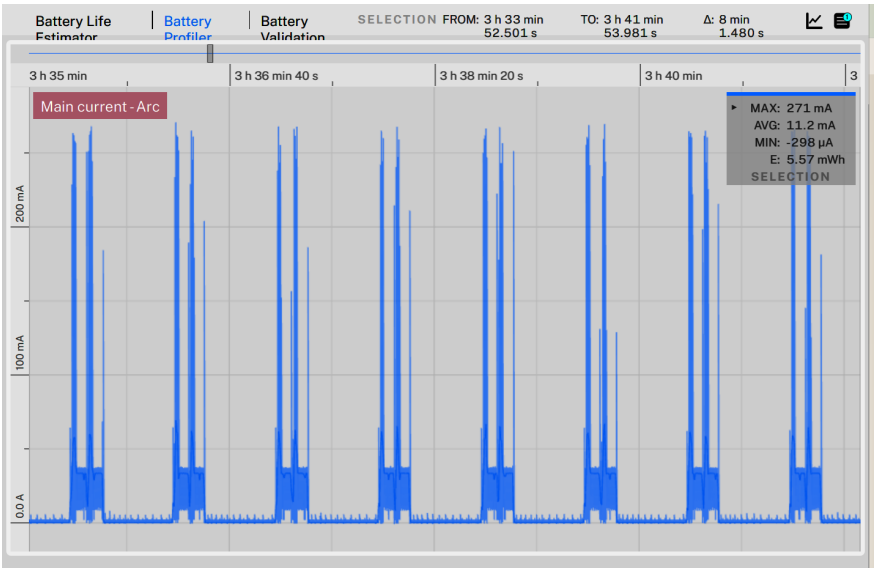
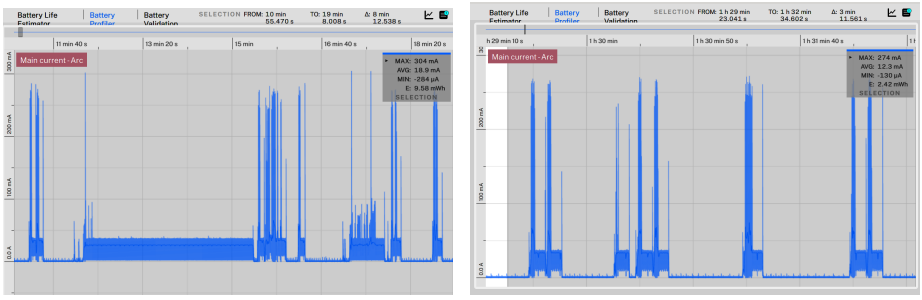


Figure 7.3: Typical consumption profile of the OVIS device, showing the characteristic 50 s NB-IoT transmission cycle with low power intervals in PSM.



(a) Low power intervals followed by irregular high-current peaks.

(b) Irregular peaks possibly caused by short retransmission events.

Figure 7.4: Examples of anomalous segments in the consumption profile.

and 16 hours). Notably, this estimate is lower than the autonomy measured experimentally (267 hours, approximately 11 days), suggesting that the effective capacity of the battery may exceed its nominal datasheet value. This observation motivates a more detailed characterization of the battery's actual usable capacity.

7.1.4 Battery discharge characterization

To further understand the OVIS device's operating lifetime, it is useful to evaluate the battery under controlled and repeatable conditions. A full discharge test driven by an idealized consumption profile provides a stable baseline for autonomy estimation against which real performance can later be compared.

The battery discharge experiment was carried out using the Battery Profiler module of the Otii Arc Pro, which applies a user defined dynamic load to the battery while accurately measuring voltage, current, and accumulated energy. The consumption

model (described in Appendix D) used in this test was derived from the Consumption profile characterization presented earlier.

A simplified two phase periodic model was selected for the test, consisting of: an active phase with a current of 35.552 mA lasting 16.12 s, followed by an inactive phase of 753 µA lasting 33.88 s, resulting in a total period of 50 s.

The discharge began with the battery fully charged at 4.15 V. A cutoff voltage of 3.0 V was configured, corresponding to the minimum operating voltage of the OVIS device. Once the battery reached this value, the instrument measured the open-circuit voltage (OCV). Because the OCV settles to a slightly higher value than the cutoff under no load conditions, it is reasonable to use an OCV threshold of 3.2 V to confirm that the battery has indeed reached the minimum operating level. Once the battery reached this condition, the discharge process was terminated, preventing the battery from entering a region where the OVIS device exhibits unstable behavior.

Autonomy estimation based on battery discharge

The test lasted 11 days, 16 hours, 1 minute, and 12 seconds (approximately 280 hours), representing the achievable autonomy of the OVIS device when operating under a consumption pattern equivalent to the modeled profile. This result provides a direct estimate of the OVIS device's battery life under nominal and repeatable conditions. Moreover, it is consistent with autonomy measured experimentally (267 hours, approximately 11 days), taken together, it reinforces the idea that the effective usable capacity of the battery may exceed its datasheet specification.

A summary of the three autonomy values for the OVIS device is presented in Table 7.2.

Table 7.2: Comparison between experimental and estimated autonomies of the OVIS device.

Autonomy type	**Duration**	**Estimation method**
Experimental	267 h ≈ 11 d 3 h	-
Estimation	280 h 1 min 12 s ≈ 11 d 16 h 1 min 12 s	Battery discharge
Estimation	208 h ≈ 8 d 16 h	Nominal battery capacity

7.1.5 Estimated effective battery capacity

To estimate the effective battery capacity the autonomy measured experimentally and its corresponding average current can be used. Furthermore, the battery discharge characterization and its corresponding average current can also be used.

Using the autonomy measured experimentally of 267 hours together with the average current of 11.74 mA[2], the resulting capacity is approximately 3135 mAh. Likewise, the battery discharge characterization test lasted 280 hours with an average current of 12 mA, yielding an estimated capacity of about 3360 mAh. These values, summarized in Table 7.3, substantially exceed the nominal 2500 mAh specified by the manufacturer. The estimate derived from the earlier measurement is 25.4% higher, while the modeled discharge suggests an increase of 34.4%.

[2]This average current corresponds to a prior estimation made at the time of the autonomy measurement experiment, based on the mean consumption over 20 transmission cycles.

Table 7.3: Estimated effective battery capacity based on measurements and modeled profiles.

Data source	Discharge (hours)	Average current (mA)	Capacity (mAh)	Difference from nominal (%)
Manufacturer	–	–	2500	–
Estimation 1	267	11.74	3135	25.4
Estimation 2	280	12.00	3360	34.4

7.1.6 Low transmission operating mode

The autonomy achieved for the OVIS device (approximately 11 days) highlights the potential for further extending the OVIS device's lifetime by reducing the volume and frequency of transmitted data. In a real deployment, sending raw accelerometer samples would not useful for veterinarians or livestock managers, whose interest lies in the processed behavioral states rather than the underlying sensor traces. This motivates the definition of an alternative operating mode in which only high level information is transmitted at a low rate.

In such a configuration, the device would continue sampling the accelerometer at 25 Hz and computing behavioral states at 2.4 s intervals, but instead of sending the raw data, it would store the predicted states locally and transmit them only every few hours together with the battery level and LTE signal quality. This strategy drastically reduces the transmitted data volume and lowers the transmission duty cycle, allowing a substantial improvement in battery autonomy while keeping the information relevant for field use.

To estimate the autonomy achievable under this low transmission configuration, the energy characterization obtained for the 50 s transmission cycle can be reused. Since the OVIS device already stores locally the amount of data generated during a 50 s cycle without exceeding its memory constraints, it would be feasible to accumulate an equivalent volume of information over a longer period before transmitting. Under this assumption, the approach is to accumulate, over several hours, an amount of data equivalent to what is transmitted in a typical 50 s cycle, and send it in a single uplink event with an active current pulse analogous to the one already characterized.

This allows extrapolating the expected autonomy using only two elements: (1) the charge consumed in a representative 50 s cycle, and (2) the transmission interval selected for the low transmission mode. In practical terms, estimating the autonomy of the low transmission mode simply requires comparing the charge consumed in a typical 50 s cycle with the charge that would be consumed when that same cycle is executed only once every several hours.

Autonomy estimation for the low transmission mode

To estimate the autonomy achievable under a low transmission configuration, the characterization obtained for the 50 s cycle was reused and scaled to the longer reporting interval. The steps involved in this extrapolation are conceptually simple, and the full derivation is provided in Appendix D; however, the key intermediate results are outlined here to contextualize the final values.

First, the amount of data transmitted in a 50 s interval is quantified in ($\sim$11.3 kB). This volume fits comfortably within the OVIS device's memory constraints, which means that the same amount of information can be accumulated locally over several hours without storage modifications. Second, by dividing this 50 s data volume by the size of a behavior state data entry, it is possible to determine how long it would take to accumulate an equivalent number of states if only processed states are stored.

This leads to an accumulation period of approximately 3 hours and 46 minutes, corresponding to about 271 consecutive 50 s cycles. Third, once the accumulation interval is known, the energy used in that interval (3 hours and 46 minutes) can be calculated in two ways: assuming the device continues transmitting every 50 s (normal operating mode), which scales the measured charge per cycle by a factor of 271; and assuming the device transmits only once at the end of the 3 h 46 min interval (low transmission operating mode), using a single active transmission pulse comparable to the one already characterized.

When the extrapolation is performed using the experimental OVIS device autonomy measurement of 267 hours together with the average current estimate of 11.74 mA (derived from a 20 cycle characterization), the resulting comparison between normal operation and the low transmission operating mode indicates that the latter requires approximately 7.4% of the charge consumed by the former. This leads to a projected autonomy of about 150 days.

In contrast, using the battery discharge as autonomy estimation for OVIS device of 280 hours together with the consumption characterization based on 1198 transmission cycles (an average current of 12 mA) results in a lower relative consumption of 6.6% between normal operation and the low transmission operating mode. Under these conditions, the low transmission operating mode could achieve a projected autonomy of 176 days.

These values, obtained under different assumptions and levels of characterization detail, are summarized in Table 7.4. These results illustrate how reducing the transmission frequency from one message every 50 s to one every several hours can extend the autonomy of the OVIS device from roughly 11 days to between 150 and 176 days.

Table 7.4: Projected autonomy of the low transmission operating mode.

Scenario	**Normal mode autonomy (hours)**	**Relative consumption (%)**	**Low trans. mode autonomy (days)**
Experimiental	267	7.4	150
Modeled	280	6.6	176

7.2 Other electronics characteristics

Besides the autonomy evaluation, the OVIS device was subjected to a comprehensive evaluation of its electronic subsystems, including embedded sensing, local processing, and wireless transmission capabilities. This included verification of the device's end-to-end operational functionality: on-device data acquisition, behavior classification, online NB-IoT transmission to the OVIS cloud server, and subsequent visualization on the OVIS user interface.

7.2.1 Accuracy of the GNSS

To evaluate the positional accuracy of the GNSS data from OVIS's device, a series of static tests were conducted under controlled conditions at Facultad de Ingeniería, Universidad de la República. These tests aimed to quantify the magnitude of localization error when the device remains motionless, thereby eliminating potential effects related to animal movement or signal obstruction. In addition to the controlled environment measurements, GNSS performance was also evaluated at Facultad de Veterinaria, Universidad de la República in a semi-controlled environment with animals.

All the measurements were conducted using Nordic Semiconductor's official GNSS sample for the nRF9160 SiP.

Indoor tests

Indoor tests were initially conducted as a convenient way to familiarize with the GNSS acquisition process and to establish a worst case reference scenario. Since sheep in extensive grazing systems are rarely located indoors (except occasionally when sheltered) these measurements are not representative of typical operational conditions. Two indoor test are reported here.

In the initial test, the device was placed at a fixed location and its reported positions were logged over time. The results revealed a mean error of 14.4 m, with a standard deviation of 13.9 m. This level of inaccuracy is notable when requiring precise behavioral mapping or verification of fence compliance. Environmental factors and satellite geometry contribute to this variability.

In the second indoor test (also with the device motionless inside an office), a TTFF of approximately 2000 seconds (∼33 minutes) was observed. This long acquisition time was not seen systematically; instead, it reflects a worst case event under severely attenuated reception conditions. According to the datasheet [56] and GNSS application notes [57], a typical TTFF under ideal conditions should be significantly shorter than the values found in OVIS system.

Excluding the extended TTFF period, the logging window corresponded to 11,552 expected samples at a frequency of 1 Hz. Of these, 10,261 samples were successfully recorded, indicating a post-fix loss of 1,291 samples (approximately 11.18%), likely due to intermittent signal obstructions inherent to an indoor environment.

To interpret positional dispersion, heatmaps and coordinate plots were generated from all recorded data from the second test (Fig. 7.5). When filtering for Position Dilution of Precision (PDOP <2, a commonly recommended threshold [58]), spatial clustering improved significantly (Fig. 7.6a). A low PDOP value indicates that the satellites used in the fix are well distributed in the sky, resulting in better geometric conditions for accurate positioning. Since PDOP depends directly on both the number of visible satellites and their relative geometry, a similar improvement was obtained when filtering by fixes that used a high number of satellites (Fig. 7.6b).

Outdoor tests

Complementary static open sky tests were conducted to validate indoor results, with measured spreads consistent with an average deviation of 11.1 m, illustrated in Fig. 7.7.

In addition to the controlled outdoor measurements, GNSS performance was also evaluated during a semi-controlled outdoor environment with animals. As shown in Fig. 7.8, the sheep were continuously located outdoors in an open paddock, however, the reported positions are highly dispersed, with many points falling inside nearby buildings or outside the actual fenced area. This behavior mirrors the deviation observed in static open sky tests and further confirms that, even under unobstructed outdoor conditions, the positional accuracy of the OVIS device remains insufficient for applications requiring precise spatial tracking or reliable geo-fencing.

Results analysis

Other developers using the nRF9160 have reported similar limitations in GNSS performance. Users have observed prolonged TTFF exceeding 10 minutes even under favorable sky conditions [59], and cases where no fix was obtained despite strong satellite signals [60]. Furthermore, discrepancies between reported and actual accuracy values have ranged from 13 m to 98 m [61].

A key limitation arises from the nRF9160's multiplexed architecture, where GNSS and LTE operations cannot occur concurrently. Disabling LTE has reportedly im-

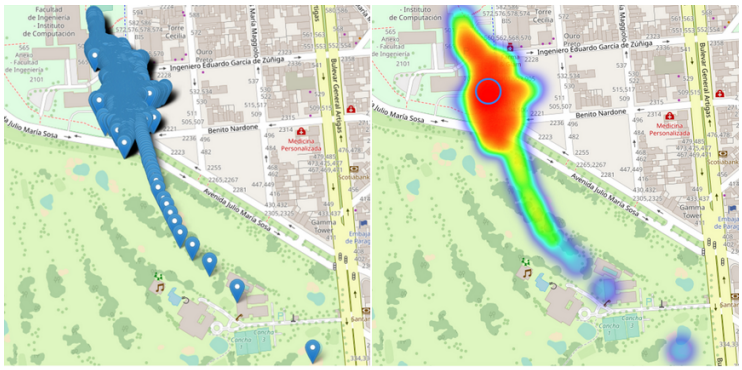


Figure 7.5: GNSS trace using all collected data. The dispersion illustrates positional variability under static indoor conditions.

(a) GNSS positions filtered by PDOP < 2. A low dilution of precision reflects favorable satellite geometry, resulting in more consistent position estimates.

(b) GNSS positions filtered by fixes using 9 satellites. A larger number of satellites generally improves geometric conditions, which enhances location consistency.

Figure 7.6: Filtered GNSS data under two conditions: low PDOP (left) and high satellite count (right). Both metrics are closely related, as satellite geometry and availability jointly determine positioning accuracy.

proved GNSS acquisition times [62], suggesting potential radio frequency interference or scheduling constraints.

To mitigate these challenges, a preliminary analysis showed that applying outlier rejection techniques significantly improved the consistency of reported positions.

7.2.2 Performance of the communication OVIS device to OVIS cloud server

The performance and reliability of device-to-cloud communication were evaluated using multiple NB-IoT deployments under varying environmental and network condi-

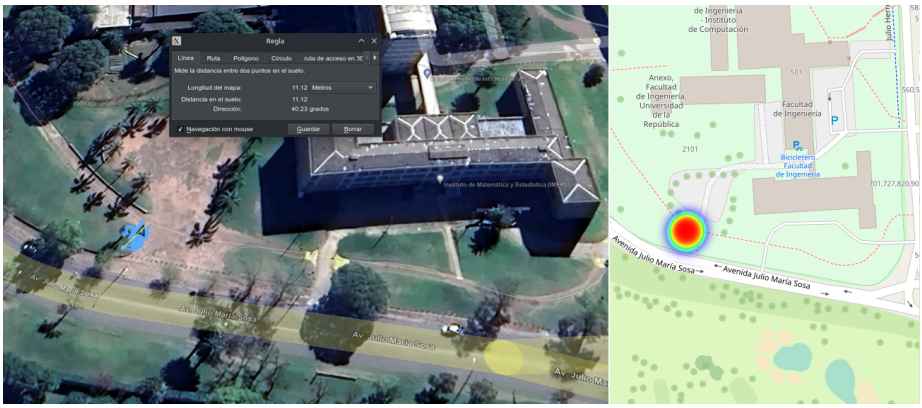


Figure 7.7: Outdoor test under open sky conditions. A positional deviation of 11.1 m was observed despite static antenna placement.

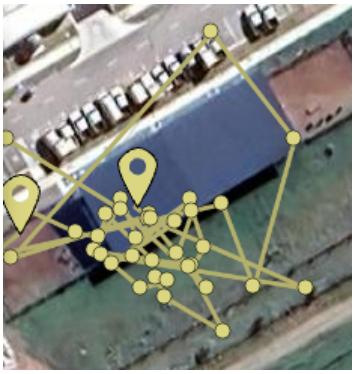


Figure 7.8: GNSS positions recorded during an outdoor animal test. Despite the sheep remaining within the fenced paddock, the reported locations are scattered across nearby buildings and areas outside the enclosure.

tions. Table 7.5 summarizes the results from these tests, which included both OVIS device versions from [42] and upgraded version from [43] operating simultaneously. The main metric of interest was the percentage of data packets successfully received by the OVIS cloud server, which reflects the end-to-end transmission reliability.

The tests were performed across rural (Facultad de Veterinaria's Migues experimental field), suburban (Facultad de Veterinaria, Universidad de la República), and urban (Facultad de Ingeniería, Universidad de la República) locations in Uruguay using the ANTEL's NB-IoT infrastructure. The RSRP was logged when available, as an indicator of radio signal strength. According to 3GPP standards and field experience [63], RSRP values above −80 dBm are considered excellent, −80 dBm to −90 dBm are good, −90 dBm to −100 dBm are fair, and values below −100 dBm indicate poor signal quality.

In rural settings, packet delivery varied significantly depending on signal quality. For instance, under poor signal conditions (RSRP = −135 dBm), packet success was limited to 54.7%. When signal improved to −105 dBm, the delivery rate rose to 83.3%. In contrast, urban deployments achieved high reliability even at higher data volumes: for a payload of 45.2 kB per packet, the delivery success remained above 95% with

RSRP values of −95 dBm and −75 dBm.

Suburban tests further demonstrated the stability of NB-IoT transmission under moderate payloads and varied data rates. In two cases where RSRP was not available, delivery rates remained above 95%, confirming that infrastructure in suburban areas was more consistent. Additionally, testing with four devices from [43] transmitting concurrently (each with 11.3 kB payloads) showed 95% delivery success under good signal conditions (RSRP >−95 dBm), validating the scalability of the setup under parallel device operation.

These tests are part of an ongoing collaboration with ANTEL and serve as a diagnostic tool for identifying weak coverage zones. While urban and suburban performance is satisfactory, rural NB-IoT coverage remains a challenge, as evidenced by lower delivery rates under weak signal conditions. In this context, it is relevant to note that some mechanisms typically associated with enhanced reliability, such as redundant transmissions do not fundamentally solve the underlying coverage limitations. Redundant transmissions refer to sending multiple copies of the same payload to increase the probability that at least one is delivered; however, this approach increases energy consumption and uplink traffic, and is difficult to sustain under the strict duty cycle and battery constraints of the OVIS device.

A key aspect that emerged from these tests is the interaction between weak coverage and the choice of transport protocol. The system uses TCP under the assumption that its connection oriented nature would prevent packet loss. However, maintaining reliability through TCP requires buffering all unsent data until the connection is restored, which implicitly assumes generous memory availability. Since the OVIS device operates with limited memory and must handle potentially long periods without connectivity in rural deployments, this design choice restricts its ability to recover from extended outages. Under these conditions, TCP's retransmission and control mechanisms can become counterproductive, leading to backlog accumulation and degraded system responsiveness. This result suggests that lightweight, connectionless approaches such as UDP may provide a more suitable trade-off between robustness and resource constraints in highly intermittent rural NB-IoT environments.

Table 7.5: Device - cloud server communication tests.

Location of the facility	RSRP (dbm)	Data rate (packets/min)	Payload (bytes)	Received packets	Device reported in
rural	-135	12.0	300	54.7 %	[42]
rural	-105	12.0	300	83.3 %	[42]
urban	-95	1.2	45.2k	95.8 %	[42]
urban	-75	1.2	45.2k	96.0 %	[42]
suburban	N/A	3.0	300	95.9 %	[42]
suburban	N/A	0.5	300	99.6 %	[42]
suburban	> -95	1.2	11.3k	95.0 %	[43]

7.3 Classification algorithm performance

The OVIS classifier effectiveness was evaluated through a series of field validation tests conducted between November 2023 and January 2024, in collaboration with IIE, INCO, and FVET [3]. The primary objective was to assess the algorithm's ability to

[3]IIE is the Instituto de Ingeniería Eléctrica, INCO is the Instituto de Computación, both from Facultad de Ingeniería and FVET is the Facultad de Veterinaria. All three belong to

detect locomotion based sheep behaviors using online accelerometer data collected from the OVIS device. Throughout three distinct data collection sessions, field deployments captured behavioral data from live animals equipped with an OVIS device[4].

7.3.1 Behavior characterization through live observations

Accurately assessing the performance of the OVIS classifier requires establishing a reliable ground truth for the animals' behavior. This was achieved through a process of live behavioral observations, supported by synchronized video recordings and manual annotations. The OVIS device enabled online monitoring of the sheep's behavior, but validating the predicted states required comparing them against human labeled observations.

Behavioral labeling was conducted using an ethogram divided into three independent and mutually exclusive categories: locomotion, posture, and activity (Table 7.6). This structure, adapted from [23], allowed for the disambiguation of overlapping behaviors (e.g., walking while grazing). However, in this stage, only the locomotion category was used, as it aligned with the trained OVIS classifier.

Table 7.6: Ethogram used for labeling data.

Category 1 Locomotion	Category 2 Posture	Category 3 Activity
Still	Standing	Grazing
Walking	Lying	Ruminating
Running	Moving	Drinking
		Other

The live observations took place across several controlled trials, each lasting two to three hours. Groups of five adult ewes were selected, and collars were worn by two or three animals per trial. This setup was designed to avoid altering group dynamics or interfering with natural behavior. Live behavior was annotated through synchronized video recordings and in-person observations conducted in controlled paddock environments.

Video data were collected using a GoPro Hero 3+ camera positioned to capture full animal movement. Synchronization between video and accelerometer data was achieved by inducing a pronounced, isolated motion in front of the camera following a period of inactivity. This event created a recognizable signal in both the video and sensor data, facilitating temporal alignment.

In parallel, live annotations were conducted by one trained observer per collared sheep. Observers recorded behavioral labels at one minute intervals over a two hour session, with rotation every 20 minutes to maintain attention quality. Video annotations were then reviewed and cross referenced with live annotations. The resulting combined annotations provided a robust foundation for training and validating the OVIS classifier.

Universidad de la República.

[4]All animal experiments conducted within the framework of this thesis, which involved the recording of sheep behavior and spatial dynamics, did not involve any stressful or painful handling. All procedures were approved by the the Ethics Committee (CHEA) of Facultad de Veterinaria, Universidad de la República under Protocol CHEA 1030/20, "Validación de dispositivos electrónicos para el estudio del comportamiento animal" (from March 2020 to February 2025).

7.3.2 Classification results and analysis

The OVIS classifier implemented on the OVIS device featured 10 trees and predicted three locomotion states. Evaluation results indicated an overall accuracy of 81%. To contextualize this reported 81% accuracy, it is important to highlight the size and distribution of the annotated dataset used for evaluating the embedded classifier. A total of 296,230 samples were analyzed across three sheep. The first sheep contributed 26,376 annotated samples, while the second and third contributed 134,927 each. These values reflect the full set of synchronized accelerometer–annotation pairs used to compute the performance metrics reported in Figure 7.9. The standing class exhibited the best performance, with 88% precision, 90% sensitivity, and an F1-score of 89%. In contrast, performance on walking and running was suboptimal, running class showed a 82% precision, 56% sensitivity, and a corresponding F1-score of 66%. Meanwhile, the walking class showed notably lower values, with 48% precision, 47% sensitivity, and an F1-score of 48%.

This outcome reflects significant confusion between dynamic activities, especially walking and running, which often overlap in natural grazing environments. Given the high confusion between these two behavioral states, the problem was redefined into a binary classification task with still and movement as classes, achieving a small improvement in the movement class F1-score to 52%. This binary formulation reduces label fragmentation and mitigates misclassification between walking and running, since both behaviors share similar acceleration patterns and often transition smoothly in grazing contexts. However, the improvement in the movement class F1-score is modest (52%) because the underlying source of confusion (high variability within movement patterns and limited annotated samples for dynamic behaviors) remains present. Collapsing the classes simplifies the decision boundary but does not fully resolve the intrinsic overlap of motion signatures.

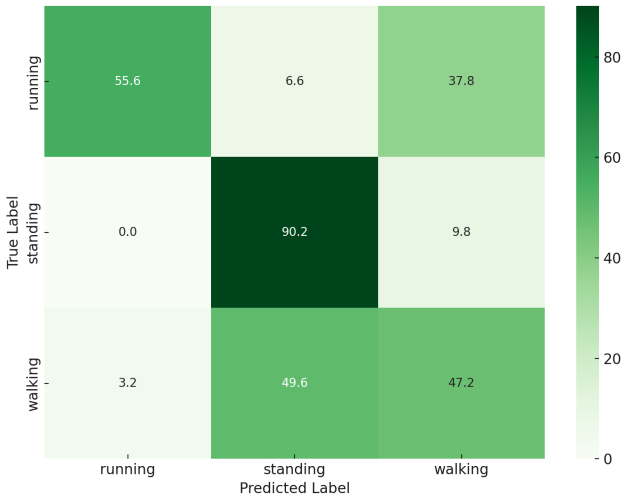


Figure 7.9: Confusion matrix for RF trained with the external database from [15] on the PC and tested with dataset gathered in the present study.

To further enhance performance, a new RF algorithm was later trained and tested on the PC using only the dataset collected in the present work (OVIS data), independently from [15]. The OVIS data used for this server based analysis consisted of four labeled subsets obtained from individual sheep, with a total of 283,394 samples.

The distribution of samples across individuals is summarized in Table 7.7. For this evaluation, data from Sheep 1 was used as the test set, while the remaining three individuals were used for training and validation. This dataset was used exclusively for a server based classifier optimization in both the three state and two state formulations, enabling a more representative evaluation of the algorithm under the natural grazing conditions captured in this study. With three states and three trees, selected through 10-fold cross-validation, the classifier achieved 84% overall accuracy. The still class continued to show excellent results (87% precision, 97% sensitivity, 92% F1-score). The running class improved moderately (61% F1-score), while walking remained a challenge (33% F1-score).

Table 7.7: Sample distribution of the OVIS data used for training and testing the second and third PC classifiers.

Sheep	**samples**
Sheep 1	62,878
Sheep 2	25,179
Sheep 3	105,574
Sheep 4	89,763
Total	**283,394**

A third RF classifier was trained and tested using OVIS data on the PC with two states (still, movement) using 10 trees. This model achieved an overall accuracy of 86%, with still achieving 87% precision, 98% sensitivity, and 91% F1-score. Although 'movement' improved in precision (80%) compared to the embedded model (47%), its sensitivity dropped to 37%. This reduction reflects a shift toward more conservative decision making: the classifier strongly prioritizes avoiding false positives, meaning that when it predicts movement, it is very likely to be correct, but it fails to detect many true movement episodes. This behavior also reveals that the model tends to classify ambiguous or high intensity movement patterns as stillness. Thus, although the classifier provides highly reliable predictions for the still class, its ability to correctly identify movement remains limited.

Finally, these results illustrate a clear bias toward the majority class (still), a common effect in imbalanced training scenarios. This raises the question of whether balancing strategies (such as downsampling the majority class) might have helped mitigate this bias during training. In this work, such balancing was not applied, but it remains a relevant alternative for future iterations of the model.

The results underscore the reliability of the algorithm in detecting static behavior, while highlighting the inherent difficulty in distinguishing dynamic behaviors such as walking and running under natural grazing conditions.

Chapter 8

Conclusion

8.1 Main outcomes of this work

This work presented the design, manufacture, implementation, and validation of an end-to-end research platform aimed for online monitoring sheep behavior in extensive livestock systems. In summary, this work lays the foundations for a robust and extensible sheep monitoring platform suited for research. The integration of embedded processing, long range wireless communication, and online cloud analysis makes it a promising tool for advancing animal welfare, behavioral science, and PLF. Considering further improvements, the OVIS system may evolve into a solution with autonomous operation in large scale, low intervention grazing systems.

The proposed system integrates a custom and wearable electronic collar and a cloud server. The collar is equipped with an Icarus IoT Board from Actinius, this board has a three-axis accelerometer (LIS2DH from STelectronics) and GNSS sensor. The board is capable of long range wireless transmission over the LTE, NB-IoT protocol and local data processing. The collar also has solar panels (totaling 2.8 W), and a 2500 mAh Lithium Polymer battery. On the cloud server side, an infrastructure based on Amazon Web Services was deployed to receive, decode, store, and visualize the data received from the collar online.

The embedded software was developed using Zephyr RTOS and was based on a modular and driven with events architecture. The application acquires accelerometer data at 25 Hz, battery level, and LTE signal strength every 50 s. When enabled, it also acquires location data every 10 s to 30 s. The accelerometer samples are encoded to reduce the transmitted data. Moreover, the application allows on-animal processing of sensor signals, that translates in classification of behavioral states computed every 2.4 s directly on the device.

The collar sends all gathered data to the cloud server every 50 s. In all cases, communication is handled by the integrated LTE modem in the nRF9160 System-in-Package, using MQTT over TLS to ensure security and reliability. The cloud architecture includes AWS IoT Core, Lambda functions (Node.js), DynamoDB storage, and a user web interface built in NextJS 13.

The system facilitates data processing, both collar and server side. Regarding behavior classification, an RF algorithm was embedded in the microcontroller and evaluated in live experiments. The algorithm distinguishes between three locomotion states (still, walking, and running) and achieved an overall accuracy of 81%, with notably high precision and sensitivity for the still class. However, the performance for walking and running was more limited, primarily due to class imbalance: those states

had approximately 60 % fewer samples than still class. To address this, a second RF model was trained on the PC using only the dataset collected in this work. This model, achieved an overall accuracy of 84 % for the three class case, and 86 % for a binary classifier distinguishing between still and movement. These results confirm that the quality and balance of the dataset are crucial for classification performance, and they motivate the generation of more annotated data.

Thirty collars were manufactured. The entire system was developed through an interdisciplinary collaboration between Engineering, Veterinary, and Industrial Design teams, and it has been validated through field deployments and controlled tests in research facilities.

One of the key outcomes of this work is the demonstration of a fully operational, monitoring collar with high autonomy. In normal mode, where both raw and on-device processed data are transmitted, the collar achieved more than 267 hour of uninterrupted operation. This corresponds to over 11 days of autonomy without solar energy harvesting. Based on current consumption measurements, autonomy could reach more than 150 days when operating in a low transmission mode. In this mode, the device would sample the accelerometer at 25 Hz, compute behavior states every 2.4 s, and transmit only the accumulated behavior states together with battery level and LTE signal metrics approximately every four hours. These results indicate that the system already meets the requirements for short and medium term research trials, and has the potential to support long term deployments after further energy optimizations.

8.1.1 State of the art comparison

A comparative analysis with other systems reported in the literature (see Table 8.1) highlights several distinguishing features of the OVIS system. While previous works often relied on proprietary communication systems, custom servers, or deferred processing analysis, the OVIS system offers: online cloud connectivity through public infrastructure, using NB-IoT technology, on-animal classification capabilities, enabling behavioral analysis without the need to transmit raw data. Also, the cloud infrastructure is customizable with a user friendly interface. Furthermore, the design is modular, facilitating future extensions (e.g., new sensors). Promising results indicate that autonomy can be extended from 11 to over 150 days, without relying on solar panels.

8.2 Technical limitations

Despite these achievements, several technical challenges remain. GNSS performance in suburban conditions revealed significant dispersion under low quality signal conditions. Furthermore, NB-IoT coverage in Uruguay is heterogeneous: while urban areas achieve over 95 % packet success rate, some rural zones exhibit packet loss exceeding 40 % due to low RSRP and signal quality indicators.

8.3 Future work

To move towards a robust application and support adoption in extensive systems, several lines of future work have been identified.

Improving classification performance remains a central goal. This involves increasing the number of detectable behavioral states (such as grazing, lying, and ruminating) and refining model architecture, including adjustments in feature selection, segmentation strategy, and model parameters to improve discrimination between behavioral

	[64]	[26]	[30]	[32]	This work
Application	Parturition	GBC[I]	GBC	GBC	GBC
Communication	No	propietary	propietary	propietary	NB-IoT
Online classification	No	Yes	Yes	Yes	Yes
On-animal classification	No	Yes	Yes	Yes	Yes
Raw data transmission	No	No	No	Yes	Yes
Device					
Custom-made device	No	Yes	Yes	Yes	Yes
Device location	Neck and Ear	Neck	Ear	Neck	Neck
Autonomy (days)	>15	>1	2.4	134	11 / 150+
Accelerometer sample rate (Hz)	12.5	100	16	50	25
GNSS sample period (min)	2 - 3	0 - configurable timer	No	No	0.2 - 240
Memory	64 MB	2 GB	384 kB	32 kB	1 MB
Weight (g)	N/A	281	N/A	N/A	480
Size (cm^3)	N/A	14.6×8×6.5	N/A	N/A	19.2x10.2x12.0
Microcontroller / SoC	N/A	MSP430FR5739	Quark SE C1000	CC1110	nRF9160
Classification					
Number of behaviors	1 (Partuition)	5	3	5	3
Accuracy (%)	91	82.4	85.2	91.8	81
Classification algorithm	SVM	LDA	K-means & KNN	DT	RF
Feature selection algorithm	ROC curve analysis	SFS	No	Fselector CRAN	K Best
Number of features	4	12	20	11	5

I: GBC = General Behavior Classifier.

Table 8.1: State-of-the-art comparison

states while maintaining computational feasibility for on-animal classification. Moreover, it would be valuable to develop a larger and more diverse behavioral dataset. The classifiers trained in this thesis relied on limited data, so building a public, high quality dataset covering several animals, breeds, and sensor configurations would greatly support progress in this research area.

Furthermore, extending the sensing capabilities of the collar could be explored to capture physiological variables from the animal. The integration of sensors for temperature, heart rate, or respiration (preferably using wireless and low power technologies) could enable more comprehensive animal monitoring. In addition, GNSS performance should be enhanced. Applying differential correction techniques, along with filtering strategies based on PDOP metrics, could significantly improve location accuracy.

Energy autonomy should also be increased by leveraging the solar energy harvesting provided by the solar panels. Moreover, the low transmission operating mode proposed in this thesis should be implemented and evaluated. This mode is intended for users who do not require raw data and has the potential to achieve autonomy on the order of hundreds of days, making it highly suitable for long term deployments in extensive production environments.

Improving the reliability of data logging under poor signal conditions is also necessary to ensure data integrity (e.g., adding an SD card to the OVIS device for buffering).

Reducing the complexity of collar management and maintenance is another important aspect. Enabling remote configuration of parameters and FOTA updates from the OVIS cloud server would significantly simplify field operations. Additionally, cost reduction strategies remain crucial for large scale deployment.

Finally, future work should explore on-animal and cloud server processing collaboration. This includes developing algorithms that fuse multiple sources of data, such as weather conditions, paddock configurations, or inter animal dynamics, to improve behavioral insight and decision making.

This page has been intentionally left blank.

Appendix A

Parturition related behavior classifiers

Accurate detection of parturition events in livestock is critical for improving animal welfare and optimizing management practices. Several studies have explored the use of wearable sensors and ML algorithms to predict and classify behaviors associated with parturition. These works focus on identifying behavioral changes and extracting features from sensor data to anticipate birth events. The following paragraphs review parturition related behavior classifiers, and Table A.1 summarizes the studies and their respective approaches.

[65] aims to detect parturition events, predicting the lamb's birth date with 84% accuracy using a two-stage detection algorithm[1] and one feature derived from 4 hour windows. The custom collar device, placed around the neck, weighs 4 g and measures 5.6 × 3.9 × 1.5 cm^3, with 8 GB of memory and autonomy exceeding 17 days. Accelerometer data is sampled at 10 Hz. The system was tested on 76 multiparous Merino ewes. Although the best performing method reached a Mean Absolute Error[2] of 5.33 hours for parturition onset estimation, this resolution was not sufficient to determine the exact start of labor. However, it was adequate to determine the correct birth date in 84% of the animals. This solution is notable for being the first parturition predictor tested in extensive conditions with promising results.

[66] investigates behavioral changes surrounding parturition and confirms their presence. Meanwhile, [67] goes further, suggesting that sensor derived metrics could directly predict lambing events. Both studies use the same ear tag device as in [17], a three-axis accelerometer sampled at 12.5 Hz and tested on 13 mature Debouillet ewes. Although no predictive model for parturition is implemented, each article examines parturition related patterns. [66] analyzes five behaviors using 10 s and 30 s windows, summarized per hour and day.

While, [67] applies a RF classifier to predict seven mutually exclusive behaviors, achieving 66.7% accuracy for seven behaviors and 87.2% for the binary classification of active vs. inactive behaviors, using five features computed from 60 s windows. Reported performance per class shows good results for feeding (76% precision, 52%

[1] The two-stage algorithm first detects a significant drop in activity using a rolling mean of the 90th percentile of acceleration magnitude, then identifies the lowest activity point near that drop as the predicted time of parturition.

[2] Mean Absolute Error (MAE) quantifies the average absolute difference between predicted and actual values; lower values indicate more accurate predictions.

Appendix A. Parturition related behavior classifiers

sensitivity), lying (84%, 78%), and standing (56%, 62%), while behaviors like licking lamb, contractions, and walking had lower metrics.

[64] predicts parturition events within a 6 hour window with 91% accuracy. The system uses two devices: the first, with an accelerometer, is the same as in [17], [66], and [67]; the second, worn on the neck, includes a location sensor, weighs 37 g, and has a volume of $4.6 \times 4.15 \times 1.4$ cm^3. Location data are collected every 3 minutes. The device has 64 MB of memory and an autonomy of over 15 days. [64] uses SVM for classification and applies ROC curve analysis to select four features calculated with 10 s and 30 s windows and summarized hourly. The study involved 18 twin bearing Merino ewes. This solution is notable for incorporating both accelerometer and location data, and for requiring social features such as proximity to peers based on sensor data.

[68] examines behavior changes during parturition using a mouth type device weighing 291 g, equipped with a three-axis accelerometer sampled at 30 Hz, 4 GB of memory, and over 25 days of autonomy. The study involved two trials with 32 and 165 cross Merino ewes with Border Leicester and East Friesian. Five behaviors (licking, grazing, idling, ruminating, walking) were classified using an SVM algorithm with 30 features calculated from 5 s windows and aggregated hourly. The model achieved concordance rates between predicted and observed behavior ranging from 81% (licking) to 96% (idling). Additionally, a deep learning model was trained to predict lambing time, achieving 90% confidence intervals for predictions up to 10 days before parturition.

[69] classifies six behaviors, ruminating, grazing, walking, idle, labour, and licking, using a halter mounted accelerometer sampled at 30 Hz. The same device was employed in [19], [21], and [24]. A total of 130 Merino ewes were used: 109 lambing ewes and 21 non-pregnant ewes. The classification was performed using a Long Short Term Memory (LSTM) model with 115 features extracted through a CNN from raw data using multiple window sizes (5 s, 10 s, 20 s, 30 s, and 60 s). The LSTM model achieved 87.6% accuracy and 78% macro F1-score. When isolating only licking and labour detection reached 84.8% accuracy and 84% macro F1-score, with licking sensitivity at 90% and labour sensitivity improving from 56% to 93% after fine-tuning.

Summary

Parturition related behavior classifiers are fewer in number but show potential. Most systems build upon devices previously validated for general behavior classification, often combining accelerometer data with social or location based features. Reported accuracies typically exceed 80%. Feature extraction commonly uses long time windows (up to several minutes or hours), and aggregation is often done hourly. While several approaches confirm the presence of behavioral changes before lambing, only a few offer prediction capabilities. These results suggest a path for future work in precision parturition monitoring, especially in extensive systems where timely intervention is critical.

	[65]	[66]	[67]	[64]	[68]
General					
Application	P^I detection	Changes in P	P detection	P detection	Changes in P
Custom made	No	No	No	No	No
Location	Neck	Ear	Ear	Neck and Ear	Mouth
Online data	No	No	No	No	No
Weight (g)	25.5	11	11	37 | 11	291
Size (cm^3)	5.6x3.9x1.5	2.3x3.25x0.76	2.3x3.25x0.76	4.6x4.15x1.4 | 2.3x3.25x0.76	4.6x3.3x1.5
Technologies					
Microcontroller	-	-	-	-	-
Communication	No	No	No	No	Not used
Accel. rate (Hz)	10	12.5	12.5	12.5	30
GNSS rate	No	No	No	3 min | 2 min	No
Memory	8 GB	512 MB	512 MB	64 MB | 512 MB	4 GB
Raw data trans.	No	No	No	No	No
Autonomy (days)	>17	30@12.5Hz	30@12.5Hz	>15 | 30@12.5Hz	25
Classification					
#Behaviors	1	5	1	1	5
Used algorithm	2 - stage	-	-	SVM	SVM
Feature selection	No	No	RF	ROC curve analysis	No
#Features	1 feature	-	5 features	4 features	30 features
Window size	4 h	10s and 30s summarized /h,/d	60s averaged/h	10s and 30s summarized/h	5s aggregated /h,/d
Accuracy (%)	84	-	-	91	90-96 | 81 licking
Conclusion	Event is predicted	Behavior changes	Sensor features could predict birth	Event is predicted	Events are detected

I: P = Parturition.

Table A.1: Parturition related behavior classifiers.

This page has been intentionally left blank.

Appendix B

Key OVIS modules description

This appendix provides a detailed overview of the most important embedded software modules used in the OVIS application. These include the application module, which orchestrates the system's behavior. The data module, responsible for centralizing data handing: storage, encoding and processing for transmission. And the sensor module, which interfaces with hardware elements, managing accelerometer and battery data. This appendix offers an expanded and more technical explanation of each module's functionality, structure, and interactions within the system.

B.1 Application module

The application module (**main.c**) is the core of OVIS application[1]. The module handles all initializations for proper functionality and periodic data requests to other modules (see which data is requested).

The application module has the main function with a loop running the application general state machine with three states: STATE_INIT the initial state of the module, STATE_RUNNING where the module sends out sample requests periodically. STATE_SHUTDOWN the module has been shut down after receiving a request from the utility module. The main function initializes the Application Event Manager and starts the rest of the modules (an APP_EVT_START event is triggered to let other modules know the application module has started). After initialization, the loop controls the state machine. The main function code segment is shown.

```
void main(void){
  /* Application Event Manager initialization and start of other modules */
  while (true) {
    module_get_next_msg(&self, &msg);
    switch (state) {
    case STATE_INIT:
      on_state_init(&msg);
      break;
    case STATE_RUNNING:
      on_state_running(&msg);
      break;
    case STATE_SHUTDOWN:
```

[1]The code organization and layout used in the application module are replicated across other modules. As this is the first module, the following paragraphs will provide a deeper explanation with code segments and fragments that are generally applicable to the entire software. In other modules, the explanations will be more concise.

```
      /* The shutdown state has no transition.*/
      break;
    default:
      LOG_WRN("Unknown application state");
      break;
    }
    on_all_events(&msg);
  }
}
```

B.1.1 Messages arrival: event subscription

The loop waits K_FOREVER for a new message in **module_get_next_msg(...)** and then organizes the new message in the module messages queue. K_FOREVER is infinite timeout delay[2]. Messages are generated from the module's own events or from other modules' events. To be aware of these events, the application module subscribes to them, as shown in the subscription code below.

```
APP_EVENT_LISTENER(MODULE, app_event_handler);
APP_EVENT_SUBSCRIBE_EARLY(MODULE, cloud_module_event);
APP_EVENT_SUBSCRIBE(MODULE, app_module_event);
APP_EVENT_SUBSCRIBE(MODULE, data_module_event);
APP_EVENT_SUBSCRIBE(MODULE, util_module_event);
APP_EVENT_SUBSCRIBE_FINAL(MODULE, sensor_module_event);
APP_EVENT_SUBSCRIBE_FINAL(MODULE, modem_module_event);
```

For a module to receive events managed by the Application Event Manager, the module has to be registered as a listener and then can subscribe to different event types. In this case, the application is registered as a listener with APP_EVENT_LISTENER and **app_event_handler**.

Moreover, the application subscribes with different priorities to several event types. APP_EVENT_SUBSCRIBE_EARLY is used to get first priority notifications before other listeners, APP_EVENT_SUBSCRIBE is for standard notification and APP_EVENT_SUBSCRIBE_FINAL is for last priority notifications. There is no defined order in which subscribers of the same priority are notified. The application module subscribes to events from cloud, data, utility, sensor, modem modules and its own events.

The Application Event Manager handler (**app_event_handler**) is called to put event data into messages and adds them to the application message queue. If an error happens when queuing a message, the error is registered and sent. The Application Event Manager handler code segment is shown.

```
static bool app_event_handler(const struct app_event_header *aeh)
{
  /* Variable initialization*/
  /* The function also does the following (for a cloud module event) in
  case of application, data, sensor, utility and modem modules events. The
  cloud module case is shown as an example*/
  if (is_cloud_module_event(aeh)) {
    struct cloud_module_event *evt = cast_cloud_module_event(aeh);
    msg.module.cloud = *evt;
    enqueue_msg = true;
  }
  /*...*/
  if (enqueue_msg) {
    int err = module_enqueue_msg(&self, &msg);
```

[2]kernel functions and tools names start with a letter K.

```
    if (err) {
      LOG_ERR("Message could not be enqueued");
      SEND_ERROR(app, APP_EVT_ERROR, err);
    }
  }
  return false;
}
```

After **module_get_next_msg(...)** receives a message, depending on the loop state, the loop acts on the message accordingly.

B.1.2 Messages handlers

Every state has its message handler function. The function **on_state_init(...)** (corresponding to STATE_INIT) main responsibility is to start the timer that triggers the information request to the data module for then be able to transmit the information to the OVIS cloud server.

This timer is given the name **data_sample_timer** and a periodic timer duration (obtained from the data module, as explained in the data module description). The timer duration is set to be 50 seconds, as the OVIS application transmits every 50 seconds to the OVIS cloud server. When **data_sample_timer** has a timeout **data_sample_timer_handler(...)** is called. This function send an event APP_EVT_DATA_GET_ALL as to request all information[3].

After the data timer has started, the function **on_state_init(...)** proceeds to change state to running. The **on_state_running(...)** function corresponding to STATE_RUNNING code is shown.

```
/* Message handler for STATE_RUNNING. */
static void on_state_running(struct app_msg_data *msg)
{
  if (IS_EVENT(msg, cloud, CLOUD_EVT_CONNECTED)) {
    data_get();
  }
  if (IS_EVENT(msg, app, APP_EVT_DATA_GET_ALL)) {
    data_get();
  }
}
```

The function **on_state_running(...)** calls **data_get()** if there is a cloud connection event or if there has to be a request to gather data from the data module. **data_get()** creates a list of all data to be asked from other modules (which data is detailed in the application module short description). Also, **data_get()** submits an event APP_EVT_DATA_GET to signal other modules to report the data.

Regardless the state of the application module there is a message handler that is called for all states. This is function **on_all_events(...)**, which mainly process the utility module request to change the application module to STATE_SHUTDOWN. The timer **data_sample_timer** is stopped and the state is changed.

B.2 Data module

This module is of most importance for OVIS application as it centralizes data sampled by other modules and stores it into circular buffers. Also, it keeps track of data requested by the application module.

[3] app_module_event.h header file contains a list of all the events sent by the module.

This module has dedicated thread, which was started by the application module. The kernel tool K_THREAD_DEFINE is used to define and initialize a static thread. The thread is given a name associated with the data module, a stack size, an associated function **module_thread_fn()** for execution and a thread priority, in this case is the lowest (K_LOWEST_APPLICATION_THREAD_PRIO). The **module_thread_fn()** code segment is shown, it can be seen that a typical thread structure is similar to the application module main function.

```
static void module_thread_fn(void){
  /* Variable initialization */
  self.thread_id = k_current_get();
  err = module_start(&self);
  /*Possible error registration */
  state_set(STATE_CLOUD_DISCONNECTED);
  /*...*/
  err = setup();
  /*Possible error registration */
  while (true) {
    module_get_next_msg(&self, &msg);
    switch (state) {
    case STATE_CLOUD_DISCONNECTED:
      on_cloud_state_disconnected(&msg);
      break;
    case STATE_CLOUD_CONNECTED:
      on_cloud_state_connected(&msg);
      break;
    case STATE_SHUTDOWN:
      /* The shutdown state has no transition. */
      break;
    default:
      LOG_WRN("Unknown sub state.");
      break;
    }
    on_all_states(&msg);
  }
}
```

The thread starts with the thread initialization through **k_current_get()** and **module_start(...)**. These functions identify and start the current thread. If an error happens when starting, the error is as the one detailed in the application module description.

After this, the module state (or thread sub-state[4].) is set with **state_set(...)** in STATE_CLOUD_DISCONNECTED. The other possible states are: STATE_CLOUD_CONNECTED and STATE_SHUTDOWN.

Before entering the loop, the thread function calls **setup()**. The data module sets configurations for the whole OVIS application which are stored in flash memory, including settings such as the data transmission period to the OVIS cloud server and the GNSS sampling interval. If an error happens when setting configurations, the error is registered and sent as is the one detailed in the application module description. The application is designed to receive configuration updates through a reboot[5].

[4]The thread from a module already has a state as explain in figure 3.2. Nevertheless, the module states could be seen as sub-states of a thread in Running state.

[5]The code for receiving configuration updates on: every connection to the server and whenever the device sends information to the server is already implemented and could be tested in the future.

B.2.1 Messages arrival: event subscription

Following an identical logic to the subscriptions done in the application module, the data module subscribes with early priority to a cloud, modem, GNSS and sensor module events since critical data would be arriving from those modules. Also, the data module subscribes with standard priority to application and utility module events and its own events. The Application Event Manager handler is analog to the one defined in the application module. After receiving a message, depending on the loop state, the loop acts on the message accordingly.

B.2.2 Messages handlers

on_cloud_state_disconnected(...) just changes the module state with **state_set(...)** if a CLOUD_EVT_CONNECTED event happens.

While, **on_cloud_state_connected(...)** prepares different type of data to be sent to OVIS cloud server. If an APP_EVT_DATA_GET_ALL event happens (the application module is requesting all information), **data_encode()** function is called. The code segment for this function is shown.

```
static void data_encode(void)
{
  /* Variable initialization */
  if (!date_time_is_valid()) {
    return;
  }
    /*...*/
    /*Function parameters are the encoded information array, buffer, its
    size and head for every data type,*/
    err = cloud_codec_encode_batch_data(&codec,
    gnss_buf,...,accel_state_buf,
    ARRAY_SIZE(gnss_buf),...,ARRAY_SIZE(accel_state_buf),
    head_gnss_buf,...,head_accel_state_buf);
        switch (err) {
        case 0:
            LOG_DBG("Data encoded successfully");
            data_send(DATA_EVT_DATA_SEND, &codec);
            break;
        /*Cases different than zero generate different error
        registrations*/
        }
}
```

This function first checks if the timestamp is not valid time and if so cloud publication will be aborted. Data will be stored in its buffer regardless, as circular buffers are used. Moreover, if the connection is lost, the device will continue sampling and storing data in the buffers, then empty them in batches upon reconnection.

Secondly, the function will proceed with coding data through **cloud_codec_encode_batch_data(...)**. The encoding process focuses on two elements: format data with JSON format to be readable for an AWS server (as is OVIS cloud server) and code accelerometer data with base64 as explained in Accelerometer data coded with base64 to reduce its volume. To achieve these objectives several auxiliary functions are used from **aws_iot_codec.c**, **jason_common.c** and **common.c** libraries and particularly the **common.c** library is modified to code the acceleration data. If the encoding process goes wrong an error is registered.

Finally, **data_encode()** will submit a DATA_EVT_DATA_SEND when data is coded without error. Said event is monitored by the cloud module and this is the module that actually sends the gathered information to the OVIS cloud server.

Another event that **on_cloud_state_connected(...)** monitors is CLOUD_EVT_CONFIG_EMPTY. This happens when the OVIS cloud server has not received the device configuration stored in flash memory. Then, **config_send()** is called and an analog process as the one described earlier for encoding data is followed. When configuration is coded, a DATA_EVT_CONFIG_SEND event is submitted and the event is monitored by the cloud module.

Furthermore, **on_cloud_state_connected(...)** monitors if a CLOUD_EVT_DISCONNECTED event happens and then the module state changes with **state_set(...)** to STATE_CLOUD_DISCONNECTED.

Regardless the state of the data module there is a message handler that is called for all states. This is function **on_all_states(...)**. This handler manages all data gathering. To achieve this, the handler has to monitor two events types for each data type (modem static and dynamic, GNSS, accelerometer and battery data).

One event type is for when the data will not be ready for this transmission (e.g., MODEM_EVT_MODEM_DYNAMIC_DATA_NOT_READY), this event type is triggered if an error was registered by the module that provides the information or a timeout occurred in said module and the data was not retrieved.

Whereas, the other event type is for when data is ready for this transmission (e.g., MODEM_EVT_MODEM_DYNAMIC_DATA_READY). As an example of this kind of event, the code segment where **on_all_states(...)** function reacts to modem dynamic data being ready is shown.

```
static void on_all_states(struct data_msg_data *msg){
/*...*/
    if (IS_EVENT(msg, modem, MODEM_EVT_MODEM_DYNAMIC_DATA_READY)){
        /* Data such as area code, cell ID, MCC and MNC coming from the
    message data are stored locally here. As an example, RSRP                storage
    is shown.*/
        struct cloud_data_modem_dynamic new_modem_data = {
          .rsrp = msg->module.modem.data.modem_dynamic.rsrp,
          /*...*/
          /* Message provides different flags indicating if the information
          is new or not.*/
          .rsrp_fresh = msg->module.modem.data.modem_dynamic.rsrp_fresh,
          /*...*/
          /*This flag indicates if the information is ready to encode.*/
          .queued = true
        };
        /*...*/
        /* For string data types (APN, IP) information is stored locally
        with strcpy.*/
        strcpy(new_modem_data.ip, msg->module.modem.data.modem_dynamic.
        ip_address);
        /*...*/
        /* Finally all modem dynamic data is storaged in the circular
        static buffer.*/
        cloud_codec_populate_modem_dynamic_buffer(
                modem_dyn_buf,
                &new_modem_data,
                &head_modem_dyn_buf,
                ARRAY_SIZE(modem_dyn_buf));

        requested_data_status_set(APP_DATA_MODEM_DYNAMIC);
      }
/*...*/
}
```

When a data type is ready, the general structure is as it follows. First of all, non-string data coming from the event message is stored in data module local structure (e.g., new_modem_data). Moreover, the event message provides different flags indicating if the information is new or not, the flag content is also stored in the same local structure. Then, function **strcpy(...)** is used to copy the string data received in the event message to the local structure (e.g., new_modem_data). Finally, functions like **cloud_codec_populate_modem_dynamic_buffer(...)** are used to fill the circular static buffers that store the information.

Regardless the event type (if the data type is ready or not), the data retrieval status is registered with function **requested_data_status_set(...)**.

It is relevant to mention that when acceleration data is ready the data processing is done here, in the data module (feature calculation, algorithm classification and behavior prediction), due to its importance this code is explained in Signal processing chapter.

The function **on_all_states(...)** also processes the utility module request to change the data module to STATE_SHUTDOWN. Moreover, if an APP_EVT_START happens the data module triggers a DATA_EVT_CONFIG_INIT to inform other modules of the device configuration.

B.3 Sensor module

The sensor module is designed to handle sensors. The current OVIS application manages accelerometer and battery data. This module has a dedicated thread, analogous to the one presented in the (data module description), with a similar execution function and the lowest system priority. The module thread operates in three sub-states[6]: STATE_INIT, STATE_RUNNING, and STATE_SHUTDOWN.

The thread is initialized, identified, and started in the same way as shown in the data module description. Before entering the loop, the thread function calls **setup()**, which calls **ext_sensors_init(...)**, with two main tasks. First, it assigns a handler (**ext_sensor_handler**) to **ext_sensors.c**, meaning that when an event is triggered by **ext_sensors.c**, the system will automatically call **ext_sensor_handler**.

Second, **ext_sensors_init(...)** verifies that the LIS2DH accelerometer was properly configured and initialized by the system at boot time. This is achieved by calling **device_is_ready(...)**, the LIS2DH configuration is obtained from the Device tree.

The accelerometer is configured with a full-scale range ± 4 g, a data output rate or acquisition frequency of 25 Hz, the high resolution mode to obtain 12-bit acceleration measurements and its own thread[7]. The system configures and initializes the accelerometer using **lis2dh.c** and **lis2dh_trigger.c**, the accelerometer libraries for Zephyr RTOS. Due to its importance, this process is explained in Accelerometer configuration and management.

B.3.1 Messages arrival: event subscription

Following an identical logic to the subscriptions done in the application module, the sensor module subscribes with standard priority to application, data and utility module events. The Application Event Manager handler is analog to the one defined in the

[6]The thread from a module already has a state as explain in figure 3.2. Nevertheless, the module states could be seen as sub-states of a thread in Running state.

[7]This means that the LIS2DH driver will use a dedicated thread to handle sensor interrupts instead of relying on the system work queue. This ensures that interrupts are handled as quickly as possible, reducing the likelihood of losing acceleration data.

application module. Moreover, after receiving a message, depending on the loop state, the loop acts on the message accordingly.

B.3.2 Messages handlers

The function **on_state_init(...)** (corresponding to STATE_INIT) main responsibility is to set up and enable the interrupts from the accelerometer. When an accelerometer interrupt happens, the information is retrieved from the accelerometer. All of this is also done using the accelerometer libraries for Zephyr RTOS. Due to its importance, this process is explained in Accelerometer configuration and management.

When data is retrieved from the accelerometer an event EXT_SENSOR_EVT_ACCELEROMETER_ACT_TRIGGER from **ext_sensors.c** is sent. This event is handled by **ext_sensor_handler**, which sends an event SENSOR_EVT_MOVEMENT_ACTIVITY_DETECTED to inform the data module that accelerometer data is ready.

The handler **on_state_running(...)** monitors if event APP_EVT_DATA_GET_ALL happens and the application module is requesting information. When this event happens, only function **adc_battery_data_get(...)** is called, since the accelerometer data is retrieved through the mentioned interrupts. Function **adc_battery_data_get(...)** executes **ext_sensors_battery_voltage_get(...)**, this function is implemented in the **adc_battery_voltage.c** library provided by Actinius. The library obtains the battery voltage from the LiPo battery configuring, initializing and reading the nRF9160 ADC peripheral. After the battery voltage is read, **adc_battery_data_get(...)** sends an event SENSOR_EVT_ADC_BATTERY_DATA_READY to inform the data module that battery information is ready.

Regardless the state of the sensor module, handler **on_all_events(...)**, processes the utility module request to go to STATE_SHUTDOWN.

B.3.3 Accelerometer configuration and management

Boot time initialization

At boot time, the LIS2DH accelerometer is initialized using by a Zephyr macro. The sensor is initialized with a specific priority after kernel services are set up. Also, it is determined where the accelerometer configuration and actual data will be stored. Sensor functions are assigned for execution-time operations. Moreover, **lis2dh_init(...)** is assigned as the initialization function for LIS2DH, executed at boot time.

The **lis2dh_init(...)** function sets up the communication with LIS2DH through I2C and configures the accelerometer's full-scale range (± 4 g) and enables high-resolution mode (to obtain 12-bit acceleration measurements). After this, interrupts are configured when **lis2dh_init(...)** calls **lis2dh_init_interrupt(...)**.

The **lis2dh_init_interrupt(...)** function has two main objectives. First, it creates a dedicated Zephyr thread (**lis2dh_thread(...)**) to wait for, enable, and handle interrupts. The thread starts immediately after creation, because K_NO_WAIT is used as the delay parameter. A semaphore is initialized with a count of 0, ensuring the thread runs only when an interrupt occurs or the interrupt must be enabled.

Second, it configures data-ready interrupts (INT1) by configuring the GPIO pin as an input to receive the interrupt signal and registers an interrupt callback function (**lis2dh_gpio_int1_callback(...)** for when the interrupt actually occurs. After **lis2dh_init_interrupt(...)** is executed, **lis2dh_init(...)** enables accelerometer measurements and sets the data output rate or acquisition frequency (25 Hz).

Execution-Time Operations

Thread Execution

The **lis2dh_thread(...)** waits indefinitely for the semaphore to be given. This only happens when an accelerometer interrupt occurs or when interrupts need to be enabled. Once the semaphore is given, the thread takes the semaphore and calls **lis2dh_thread_cb(...)**, which handles or enables the interrupt accordingly.

Interrupt Setup and Handling

The function **on_state_init(...)** starts the accelerometer by calling **accelerometer_callback_set**, which calls **ext_sensors_accelerometer_trigger_callback_set(...)**. This calls **lis2dh_trigger_set(...)**, which internally calls **lis2dh_trigger_drdy_set(...)**.

The function **lis2dh_trigger_drdy_set(...)** assigns the high-level interrupt handler **accelerometer_trigger_handler(...)** (not to be confused with **lis2dh_gpio_int1_callback(...)**, which is a low-level handler). It also sets a flag (START_TRIG_INT1) to signal that interrupts should be enabled, and signals the thread using the semaphore to enable the interrupt.

Once the semaphore is given, **lis2dh_thread_cb(...)** determines whether it needs to enable or handle the interrupt based on these flags: START_TRIG_INT1, which indicates that the interrupt process should start, and TRIGGED_INT1, which indicates that an interrupt has already occurred and needs processing.

Enabling the Interrupt

If the START_TRIG_INT1 flag is set, **lis2dh_thread_cb(...)** calls **lis2dh_start_trigger_int1(...)**, which temporarily disables data sampling and calls **setup_int1(...)** to enable the GPIO interrupt associated with INT1, re-enables data sampling, and enables INT1 in the accelerometer.

Handling the Interrupt

When an accelerometer interrupt occurs, **lis2dh_gpio_int1_callback(...)** is called. This function sets the TRIGGED_INT1 flag to indicate that the interrupt has occurred, disables interrupts using **setup_int1(...)** while processing, and notifies the thread using the semaphore to handle the interrupt.

Since TRIGGED_INT1 is now set, **lis2dh_thread_cb(...)** calls the interrupt handler (**accelerometer_trigger_handler(...)**) to process the data and re-enables the interrupt using **setup_int1(...)**.

Data management and Event Generation

accelerometer_trigger_handler(...) function performs three key actions. First, it fetches the raw accelerometer data using **lis2dh_sample_fetch(...)**. Second, it extracts and converts X, Y, and Z acceleration values using **lis2dh_channel_get(...)**. In this function, the raw data is scaled using **lis2dh_convert(...)**, which adjusts the values to fit the configured range (± 4 g). Finally, **accelerometer_trigger_handler(...)** sends an event (EXT_SENSOR_EVT_ACCELEROMETER_ACT_TRIGGER) containing the converted acceleration data for each axis.

This page has been intentionally left blank.

Appendix C

OVIS Device Assembly

This appendix provides a set of preliminary notes regarding the OVIS device assembly. It is not intended to serve as a step-by-step or complete assembly guide, nor does it reflect the full or current manufacturing process. Instead, it offers a very high-level overview of key stages (such as the hermeticity test) and general considerations that may help understand how certain parts of the assembly were approached.

Components and Materials

The following are the main components required for the assembly of the OVIS device:

- **Icarus IoT Board**
- **SIM Card**
- **GNSS Antenna**
- **Cellular Antenna**
- **LiPo Battery**
- **Solar Panels**: Two large panels for the sides and one smaller panel for the top.
- **Plastic Parts**: Includes the top and bottom parts of the container and side lid (USB cover).
- **Adjustable collar system**: Includes strap, pad and buckle.
- **Screws and O-rings**: Used for securing the top and bottom parts.
- **Silicone Cold Sealant** (e.g., Sikasil C – Transparent)
- **Cable Materials**: Including black and red thermoformed cables, tin solder, and double sided 3M tape.
- **Acrylic Panels**: To protect solar panels from damage.
- **Pressure Hooks and Straps**: For securing the device to the animal.
- **Other Materials**: Poxipol, fine sandpaper, and various tools.

Assembly Steps

1. Charge the battery.
2. Sand the SIM card to ensure it fits easily into the holder.
3. Check the soldering of the GNSS antenna.

4. (Optional) Program Icarus IoT Board, if needed.
5. Assemble the solar panel interconnections.
6. Solder the solar panel cables to the Icarus IoT Board.
7. Attach the acrylic panels to the top cover to protect the solar panels. See Figure C.1.

Figure C.1: Acrylic panels placement in the top cover to protect the solar panels.

8. Perform the hermeticity test. See Figure C.2.

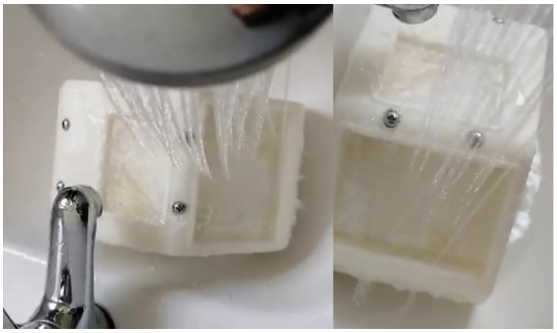


Figure C.2: OVIS device enclosure being directly exposed to a constant flow of water.

9. Put the SIM card into Icarus IoT Board.
10. Install the Icarus IoT Board into the enclosure with a drop of silicone for fixation.
11. Connect the battery to the Icarus IoT Board and secure it inside the enclosure with double sided tape.
12. Place a drop of cold silicone on the battery cable where Figure C.3 indicates in red.
13. Attach the GNSS antenna and cellular antenna to the Icarus IoT Board.

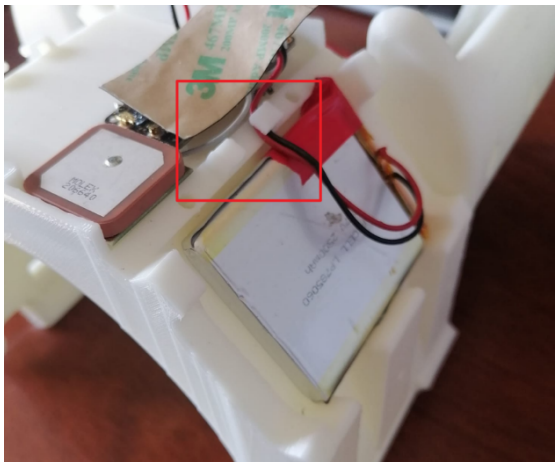


Figure C.3: Component placing inside OVIS device enclosure.

14. Secure the solar panels to the container, ensuring they are pressed against the acrylic panels.
15. Close the enclosure by attaching the top cover to the base with screws.
16. Install the USB cover and screw it into place.
17. Secure the strap and pad for comfort and stability.

Basic functionality testing

1. Verify that the device connects to the server and transmits data.
2. Check that the solar panels are functioning and charging the battery as expected.

This page has been intentionally left blank.

Appendix D

Consumption model and autonomy extrapolation

D.1 Simplified consumption model

The construction of a simplified consumption model has two main objectives. First, it aims to provide a controlled representation of the OVIS device's current profile that can be used to carry out reproducible battery discharge tests, free from circumstantial anomalies, while preserving the energy characteristics observed in real measurements. Second, the model makes it possible to understand the device's consumption behavior in a clearer and more general way, offering a compact description that retains the main magnitudes without needing to refer to the full measured trace.

All design decisions for this creating this simplified consumption model were done by analyzing the current trace obtained in the consumption profile characterization for the OVIS device (see Consumption profile characterization).

D.1.1 Identification of active and inactive periods

As a first step, the current trace must be segmented into active intervals (corresponding to transmissions or intensive processing) and inactive intervals (where the device remains idle and is in PSM). This separation allows the consumption to be divided into well defined cycles[1] and enables the derivation of a simplified model that can later be used in the battery discharge.

The identification of active and inactive periods begins with the application of a current threshold. A value of 10 mA was chosen as the minimum current above which the device is considered to be effectively active. Each sample is then classified as active if the current exceeds this threshold, or inactive otherwise. This classification makes possible the detection of activity episodes by delimiting their start and end instants, that is, the intervals during which the consumption remains clearly above the idle level.

However, the measured signal exhibits certain characteristics that require additional processing. On the one hand, small gaps of low current are observed between consecutive high consumption bursts, which should not be interpreted as inactive

[1] In this context, a *cycle* is understood as the sequence consisting of one active phase followed by its corresponding inactive phase.

periods. These brief drops below the threshold are due to the dynamics of the transmissions themselves and do not imply that the device has returned to an inactive state. Figure D.1 shows that, although the current falls below the 10 mA threshold, these instants still belong to the same activity block and should be merged to avoid splitting the active period. To address this, all active intervals separated by less than 1.7 s are merged into a single block; this value is defined as *max_gap*.

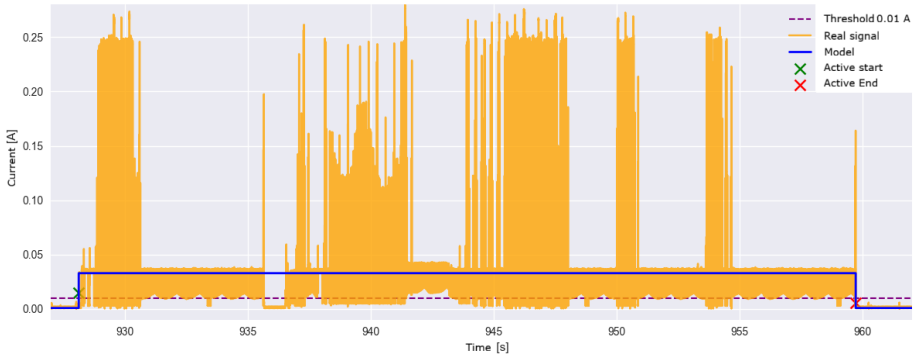


Figure D.1: Example of a signal with transient dips below the 10 mA threshold. These intervals must be considered part of the same active period.

On the other hand, very short peaks also appear, which may be due to noise or isolated fluctuations but do not represent a true activity period. To discard them, a minimum duration criterion of 0.01 s (denoted as *min_duration*) was applied so that only active intervals of importance are retained.

These parameters (the 10 mA threshold, the maximum separation time of 1.7 s between active intervals, and the minimum duration of 0.01 s) were determined experimentally by analyzing different sections of the current trace and testing the criteria on representative fragments of the full consumption profile.

The final result is a set of intervals that reflect the significant activity phases of the OVIS device, while everything outside them is interpreted as inactive periods.

D.1.2 Variables required to define the model

Once active and inactive periods have been identified, two fundamental quantities can be computed: the average current in the active state and the average current in the inactive state. These values characterize the typical consumption level in each phase, but they are not sufficient on their own to define a consumption profile model. It is also necessary to introduce temporal parameters that determine the structure of a typical cycle: three values are required, the total duration of a cycle (T), the active time within that cycle (t_{act}), and the inactive time within that cycle (t_{inact}). These parameters are related by:

$$T = t_{\text{act}} + t_{\text{inact}}.$$

Thus, to characterize the model, four variables must be defined: two currents (average active and inactive currents) and two of the three characteristic times (period, active time, and inactive time), knowing that the unspecified time is obtained from the equation above. Figure D.2 summarizes these concepts graphically, showing an idealized consumption profile in the form of a rectangular waveform. The scheme shows the average current values in the active and inactive states, as well as the characteristic times that define the period of a cycle.

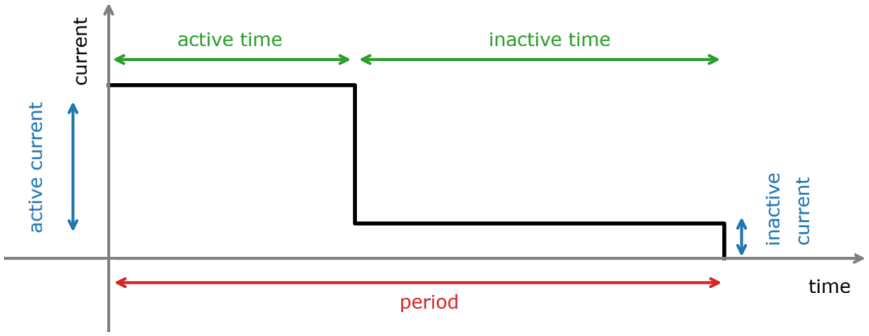


Figure D.2: Ideal consumption profile, showing the active and inactive phases, the average current in each phase, and the duration of the phases and the total period.

The way in which the temporal parameters are defined is what differentiates the models considered. Both start from the same average currents but adopt different criteria for establishing the temporal structure of the cycle. It is important to emphasize that, regardless of the chosen definition, a validity condition is imposed: the model must preserve the total charge measured during the experiment. In this way, even though the fine detail of instantaneous peaks is lost, the representation preserves the energy coherence required for use in the battery discharge characterization.

D.1.3 Model 1: characterization based on period and duty cycle

Starting from the total duration of the measurement and the number of identified cycles, an average period is obtained, which is taken as the **cycle period** in the model. The number of cycles is obtained by counting how many times the signal transitions from an active state to an inactive state.

Next, the total time during which the device was active is measured, and with the total duration of the measurement, the *duty cycle* (the fraction of the total time corresponding to active intervals) can be computed. Using this value and the cycle period, the **active and inactive times** of the cycle in the model are directly defined.

Once the times are known, the charge consumed in each state (active and inactive) is estimated. To do so, the current data must be integrated over the corresponding intervals.[2] In the case of active intervals, only samples exceeding the current threshold are selected. Each selected current value is multiplied by the sampling time step, and the sum of all these products yields the total active charge consumed over the entire measurement. The same procedure is applied to the inactive intervals: samples that remain below the threshold are considered, and by applying the same method, the total inactive charge is obtained.

Finally, the total charge of the measurement is obtained by combining both contributions. Having the total active and inactive charge and the number of cycles, the average charge corresponding to the active phase and to the inactive phase of a typical cycle can be computed.

Each of these average charges is then distributed over the corresponding active or inactive time of the cycle. Thus, dividing the average active charge by the active time of the cycle yields a value representing the **active current of the cycle**. Simi-

[2]Since the measured signal is composed of discrete samples, this integration is approximated by summing all current values in each state, multiplied by the sampling interval.

larly, dividing the average inactive charge by the inactive time of the cycle yields the **inactive current of the cycle**.

With these values, the ideal profile of Model 1 is fully defined. This scheme aims to preserve the temporal structure of the real behavior, summarized in the period and the duty cycle, and maintains energy coherence by reproducing, on average, the same charge measured during the experiment. In this way, Model 1 offers a simplified representation that remains faithful to the main consumption magnitudes.

D.1.4 Model 2: fixed 50 s period and average inactive current

In this model, the **cycle period** is fixed from the outset at 50 s. This choice reflects the expected behavior of the OVIS device under nominal operation (without anomalies) and acts as a temporal backbone on which the entire model is built.

The **average inactive current** is obtained as the mean of all current samples belonging to intervals classified as inactive. This reference current ensures that the model accurately captures the idle consumption level.

Next, the number of cycles is determined by dividing the total recorded time by the fixed period of 50 s. Using this value, the average inactive charge per cycle is calculated by dividing the total inactive charge[3] by the number of cycles.

With the average inactive charge and the average inactive current, the **average inactive time** of each cycle can be obtained. From this value, and keeping the period fixed at 50 s, the **average active time** is obtained directly as the difference between the period and the inactive time.

The next step is to compute the average active charge per cycle, by dividing the total active charge[4] by the number of cycles. The **average active current** is then obtained as the ratio between the average active charge and the average active time.

In this way, the model has both reference currents (active and inactive) and the associated durations of each phase within a 50 s cycle. As in Model 1, the total active and inactive charges are directly calculated from the measured signal by integrating the currents over their respective intervals. This ensures that the sum of both matches the total measured charge. From these quantities, and with the period fixed at 50 s, the model profile is constructed. Thus, even with a different temporal definition, Model 2 preserves the energy coherence of the measurement and guarantees that the total consumed charge is not altered. At the same time, Model 2 reflects the expected operating pattern of the device under normal conditions.

D.1.5 Model selection

Table D.1 summarizes the main differences between the two models proposed to represent the OVIS device's consumption profile.

Model 2 was chosen because it more faithfully reflects the expected behavior of the OVIS device, by fixing the cycle period at 50 s, as occurs under conditions. In contrast, Model 1 defines the period based on the global average of the measurement and can be affected by anomalies. Model 2 explicitly preserves the expected temporal structure of the device and guarantees energy coherence by maintaining the same total charge observed in the measurement. The final parameters of the selected model are: active current 35.552 mA, active time 16.12 s, inactive current 0.753 mA, and inactive time 33.88 s.

[3]The total inactive charge was already computed in Model 1 as the sum of all inactive currents multiplied by the sampling period.

[4]The total active charge was already computed in Model 1 as the sum of all active currents multiplied by the sampling period.

Aspect	Model 1	Model 2
Cycle period	Average of measured cycles	Fixed at 50 s
Computation of $T_{\text{inact}}/T_{\text{act}}$	Using period and duty cycle	T_{inact} from inactive charge and current; $T_{\text{act}} = 50$ s – T_{inact}
Computation of I_{act}	From active charge and T_{act}	From active charge and T_{act}
Computation of I_{inact}	From inactive charge and T_{inact}	Average of inactive samples
Energy coherence	Total charge preserved	Total charge preserved
Advantage	Reflects the actual measured structure	Represents the nominal behavior of the device
Limitation	Sensitive to anomalies or variations	Does not reflect real fluctuations

Table D.1: Comparison between the two consumption models: I_{inact} inactive current, I_{act} active current, T_{inact} inactive time, T_{act} active time.

D.2 Extrapolation from normal mode to low transmission mode

D.2.1 Data volume calculation

The first step is to compute the number of bytes transmitted in a 50 s period in normal mode, considering the expected behavior of the OVIS device. This calculation includes the bytes associated with raw accelerometer data, the bytes corresponding to the behavioral states generated in that interval, and the bytes used to report battery level and LTE signal strength.

Acceleration data

The device acquires acceleration at a sampling rate of 25 Hz, with three axes and a format of three characters (bytes) per value. In addition, the axis labels (x:, y:, z:) and three newline characters add 9 extra bytes per 50 s cycle:

$$3 \text{ characters/value} \times 3 \text{ values/sample} \times 25 \text{ samples/s} \times 50 \text{ s} + 3 \text{ bytes} = 11259 \text{ bytes}.$$

State data

Behavioral states are generated from non-overlapping temporal windows of 60 accelerometer samples, that is, one state is obtained for every 60 samples. Each state occupies 2 bytes (the value itself and a comma separator), plus 2 bytes for the label (e:) and 1 byte for a newline character per cycle. The total volume of state data is therefore:

$$\left(\frac{25 \text{ samples/s} \times 50 \text{ s}}{60 \text{ samples/state}}\right) \times 2 \text{ bytes/state} + 3 \text{ bytes} \approx 45 \text{ bytes}.$$

Signal and battery data

The signal level is reported with four characters, one for the sign and up to three for the value (for example, -101). To this, 5 bytes for the label (rsrp:) and 1 newline byte are added, resulting in a total of 10 bytes. Similarly, the battery level is reported in millivolts with four characters (for example, 4169), plus 2 bytes for the label (b:) and 1 newline byte, giving a total of 7 bytes.

In conclusion, the data accumulated in a 50 s period are:

$$11259 \text{ bytes} + 45 \text{ bytes} + 10 \text{ bytes} + 7 \text{ bytes} = 11321 \text{ bytes}.$$

D.2.2 Cycle calculation for low transmission mode

Out of the total 11321 bytes sent in 50 s, 11304 bytes are needed to transmit the raw acceleration data and the generated states. If this data volume were used exclusively for state transmission, and considering 2 bytes per label (`e:`) and 1 newline byte per state, the maximum number of states would be:

$$\frac{11301 \text{ bytes}}{2 \text{ bytes/state}} = 5650.5 \quad \Rightarrow \quad 5650 \text{ states}.$$

Each state is generated from 60 accelerometer samples, sampled at 25 Hz. Therefore, the total time needed to acquire the data for 5650 states is:

$$t = \frac{5650 \times 60}{25} = 13560 \text{ s} \approx 3 \text{ h } 46 \text{ min}.$$

This time corresponds to approximately 3 h 46 min of continuous acquisition, that is, about 271 periods of 50 s.

D.2.3 Estimated consumption and autonomy projection

The extrapolation from normal mode to low transmission mode is carried out in terms of charge consumption, since battery autonomy depends on the total charge drawn rather than on instantaneous current values. For this reason, instead of comparing average currents, the charge associated with each normal mode cycle is computed and used as a reference to estimate the consumption under the low transmission strategy.

To compare both transmission strategies, the current profiles for each case are represented in Figures D.3 and D.4. In normal mode, the device sends data every $T = 50\,\text{s}$. The resulting consumption profile can be described as a periodic rectangular waveform, with an active phase of duration t_A and current I_A, followed by an inactive phase of duration t_I with current I_I. In low transmission mode, the device stores the samples locally and transmits only the behavioral states, together with battery level and signal strength, every nT, with $n = 271$, which corresponds to an interval of 3 h 46 min. In this scenario, the waveform shows a single transmission pulse with charge equivalent to $I_A t_A$ and an inactive consumption of I_I during almost the entire interval, with charge equivalent to $I_I \times (nt_I + (n-1)t_A)$.

To compare the consumption of both modes, the total charge consumed over the same time interval is considered. The experimental profile in normal mode was modeled with the following values: $I_A = 35.552\,\text{mA}$, $t_A = 16.12\,\text{s}$, $I_I = 0.753\,\text{mA}$, and $t_I = 33.88\,\text{s}$. Therefore, the charge per cycle in this mode is:

$$Q_{NM} = I_A t_A + I_I t_I = 35.552 \times 16.12 + 0.753 \times 33.88 = 0.16628 \text{ mAh}.$$

Since in low transmission mode the transmission occurs every $n \approx 271$ periods of 50 s, the normal mode charge must be scaled to the same interval:

$$Q_{NM}^{(n)} = nQ_{NM} = 271 \times 0.16628 \text{ mAh} = 45.06 \text{ mAh}.$$

Meanwhile, in low transmission mode, during an interval nT there is a single active pulse and the rest is inactive consumption:

$$Q_{LTM} = I_A t_A + I_I\big(nt_I + (n-1)t_A\big) = 35.552 \times 16.12 + 0.753\,\big(271 \times 33.88 + 270 \times 16.12\big)$$

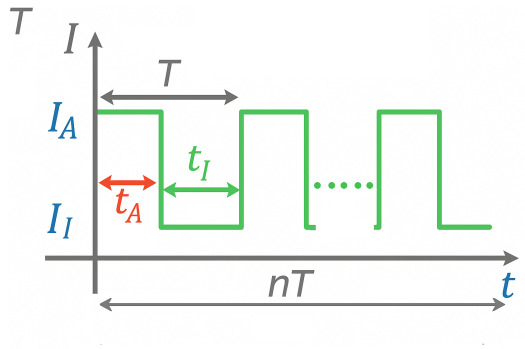


Figure D.3: Current profile corresponding to normal mode. Each period $T = 50\,\text{s}$ includes the transmission of acceleration data, computed states, battery level, and signal strength.

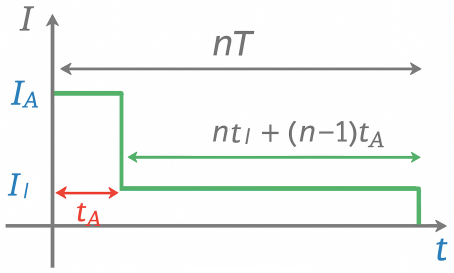


Figure D.4: Current profile corresponding to low transmission mode. The device stores samples locally and transmits only the states together with battery level and signal strength every nT, with $n \approx 271$.

$$Q_{LTM} = 2.99 \text{ mAh}.$$

Therefore, over the same interval $nT \approx 3\,\text{h}\,46\,\text{min}$ (with $n = 271$ periods),

$$\frac{Q_{LTM}}{Q_{NM}^{(n)}} \approx \frac{2.99}{45.06} = 0.066 \quad (6.6\%).$$

Given that the autonomy estimate for normal mode from the modeled battery discharge was 280 hours, and considering that the estimated consumption in low transmission mode corresponds to 6.6% of that in normal mode, the projected autonomy in low transmission mode is:

$$\text{Autonomy}_{LTM} = \frac{280}{0.066} \approx 4217\,\text{h} \approx 176 \text{ days} \approx 5.9 \text{ months}.$$

This value shows the substantial autonomy increase that can be achieved in low transmission mode by drastically reducing the data transmission frequency.

This page has been intentionally left blank.

Bibliography

[1] R. Cardellino, "La producción ovina en uruguay," 2015, last accessed 8 August 2025. [Online]. Available: https://web.archive.org/web/20160421115706/http://dohnetresarboles.com.uy/newsletters/bob/agronomia2015.pdf

[2] M. Alipio and M. L. Villena, "Intelligent wearable devices and biosensors for monitoring cattle health conditions: A review and classification," *Smart Health*, vol. 27, p. 100369, 2023.

[3] J. B. Morlan and A. Casaretto, "Principales patologias en los actuales sistemas de produccion ovina del uruguay. una puesta al día." *XL Jornadas Uruguayas de Buiatría*, pp. 19–30, 2012.

[4] Ministerio de Ganadería Agricultura y Pesca (MGAP), "Anuario estadístico agropecuario (DIEA) 2023," 2023, last accessed 8 August 2025. [Online]. Available: https://descargas.mgap.gub.uy/DIEA/Anuarios/Anuario2023/ANUARIO2023WEB.pdf

[5] J. J. Mari, "Pérdidas perinatales en corderos (in spanish)," *1eras. Jornadas Ovinas Veterinarias*, pp. 1–12, 1979.

[6] R. Nowak and P. Poindron, "From birth to colostrum: early steps leading to lamb survival," *Reproduction Nutrition Development*, vol. 46, no. 4, pp. 431–446, 2006.

[7] R. Nowak, R. H. Porter, F. Levy, P. Orgeur, and B. Schaal, "Role of mother–young interactions in the survival of offspring in domestic mammals," *Reviews of Reproduction*, vol. 5, no. 3, pp. 153–163, 2000.

[8] N. Zambra, J. Piaggio, and R. Ungerfeld, "Encuesta sobre predación ovina en Uruguay. [Resumen]," in *Actas del 6º Congreso de la Asociación Uruguaya de Producción Animal*, 2018.

[9] J. Olivera-Muzante, "¿es posible mejorar la supervivencia de corderos en nuestros sistemas ovinos?" pp. 15–17, 2015, last accessed 8 August 2025. [Online]. Available: https://www.produccion-animal.com.ar/produccion_ovina/produccion_ovina/253-cangue_olivera.pdf

[10] E. S. Fogarty, D. L. Swain, G. Cronin, and M. Trotter, "Autonomous on-animal sensors in sheep research: A systematic review," *Computers and Electronics in Agriculture*, vol. 150, pp. 245–256, 2018.

[11] L. Riaboff, L. Shalloo, A. F. Smeaton, S. Couvreur, A. Madouasse, and M. T. Keane, "Predicting livestock behaviour using accelerometers: A systematic review of processing techniques for ruminant behaviour prediction from raw accelerometer data," *Computers and Electronics in Agriculture*, vol. 192, p. 106610, 2022.

[12] K. M. McLennan, E. A. Skillings, C. J. 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.

[13] F. Alvarenga, I. Borges, L. Palkovič, J. Rodina, H. Oddy, and R. Dobos, "Using a three-axis accelerometer to identify and classify sheep behaviour at pasture," *Applied Animal Behaviour Science*, 2016.

[14] M. Radeski and V. Ilieski, "Gait and posture discrimination in sheep using a tri-axial accelerometer," *Animal*, vol. 11, no. 7, pp. 1249–1257, 2017.

[15] J. W. Kamminga, H. C. Bisby, D. V. Le, N. Meratnia, and P. J. Havinga, "Generic online animal activity recognition on collar tags," in *ACM International Symposium on Wearable Computers*, 2017.

[16] 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.

[17] E. S. Fogarty, D. L. Swain, G. M. Cronin, L. E. Moraes, and M. Trotter, "Behaviour classification of extensively grazed sheep using machine learning," *Computers and Electronics in Agriculture*, vol. 169, p. 105175, 2020.

[18] J. Barwick, D. W. Lamb, R. Dobos, M. Welch, D. Schneider, and M. Trotter, "Identifying sheep activity from tri-axial acceleration signals using a moving window classification model," *Remote Sensing*, vol. 12, no. 4, p. 646, 2020.

[19] S. Hu, A. Ingham, S. Schmoelzl, J. McNally, B. Little, D. Smith, G. Bishop-Hurley, Y.-G. Wang, and Y. Li, "Inclusion of features derived from a mixture of time window sizes improved classification accuracy of machine learning algorithms for sheep grazing behaviours," *Computers and Electronics in Agriculture*, vol. 179, p. 105857, 2020.

[20] N. Kleanthous, A. Hussain, W. Khan, J. Sneddon, and A. Mason, "Feature extraction and random forest to identify sheep behavior from accelerometer data," in *Intelligent Computing Methodologies: 16th International Conference (ICIC)*. Springer, 2020.

[21] S. J. Ikurior, N. Marquetoux, S. T. Leu, R. A. Corner-Thomas, I. Scott, and W. E. Pomroy, "What are sheep doing? tri-axial accelerometer sensor data identify the diel activity pattern of ewe lambs on pasture," *Sensors*, vol. 21, no. 20, 2021.

[22] N. Kleanthous, A. Hussain, W. Khan, J. Sneddon, and P. Liatsis, "Deep transfer learning in sheep activity recognition using accelerometer data," *Expert Systems with Applications*, vol. 207, p. 117925, 2022.

[23] E. Price, J. Langford, T. W. Fawcett, A. J. Wilson, and D. P. Croft, "Classifying the posture and activity of ewes and lambs using accelerometers and machine learning on a commercial flock," *Applied Animal Behaviour Science*, vol. 251, p. 105630, 2022.

[24] K. Turner, A. Thompson, I. Harris, M. Ferguson, and F. Sohel, "Deep learning based classification of sheep behaviour from accelerometer data with imbalance," *Information Processing in Agriculture*, 2022.

[25] J. Marais, R. Wolhuter, T. Niesler, and S. Le Roux, "Automatic classification of sheep behaviour using 3-axis accelerometer data," 2014.

[26] S. P. Le Roux, J. Marias, R. Wolhuter, and T. Niesler, "Animal-borne behaviour classification for sheep (dohne merino) and rhinoceros (ceratotherium simum and diceros bicornis)," *Animal Biotelemetry*, vol. 5, pp. 1–13, 2017.

[27] V. Giovanetti, M. Decandia, G. Molle, M. Acciaro, M. Mameli, A. Cabiddu, R. Cossu, M. Serra, C. Manca, S. 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.

[28] E. Walton, C. Casey, J. Mitsch, J. Vázquez-Diosdado, J. Yan, T. Dottorini, K. 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.

[29] 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*, vol. 18, no. 10, 2018.

[30] J. A. Vázquez-Diosdado, V. Paul, K. A. Ellis, D. Coates, R. Loomba, and J. Kaler, "A combined offline and online algorithm for real-time and long-term classification of sheep behaviour: Novel approach for precision livestock farming," *Sensors*, vol. 19, no. 14, p. 3201, 2019.

[31] L. Guo, M. Welch, R. Dobos, P. Kwan, and W. Wang, "Comparison of grazing behaviour of sheep on pasture with different sward surface heights using an inertial measurement unit sensor," *Computers and Electronics in Agriculture*, vol. 150, pp. 394–401, 2018.

[32] 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. 105444, 2020.

[33] M. Decandia, S. Rassu, V. Psiroukis, I. Hadjigeorgiou, S. Fountas, G. Molle, M. Acciaro, A. Cabiddu, M. Mameli, C. Dimauro, and V. Giovanetti, "Evaluation of proper sensor position for classification of sheep behaviour through accelerometers," *Small Ruminant Research*, vol. 201, p. 106445, 2021.

[34] Z. Jin, L. Guo, H. Shu, J. Qi, Y. Li, B. Xu, W. Zhang, K. Wang, and W. Wang, "Behavior classification and analysis of grazing sheep on pasture with different sward surface heights using machine learning," *Animals*, vol. 12, no. 14, 2022.

[35] Nofence AS, "Nofence virtual grazing," 2024, last accessed 13 June 2025. [Online]. Available: https://www.nofence.no

[36] Digitanimal, "Digitanimal: GPS livestock monitoring," 2024, last accessed 13 June 2025. [Online]. Available: https://digitanimal.com

[37] N. Acosta, N. Barreto, P. Caitano, R. Marichal, M. Pedemonte, and J. Oreggioni, "Research platform for cattle virtual fences," in *Proceedings of the IEEE International Conference on Industrial Technology (ICIT)*, 2020, pp. 797–802.

[38] P. Castro-Lisboa, J. Oreggioni, and E. Machado, "System and device for monitoring the reproductive activity of animals," 2020, US Patent 10,575,501.

[39] V. Campiotti, N. Finozzi, J. Irazoqui, V. Cabrera, R. Ungerfeld, and J. Oreggioni, "Wearable device to monitor sheep behavior," *IEEE Embedded Systems Letters*, vol. 15, no. 2, pp. 89–92, 2023.

[40] H. Cardoso, "Herramientas de software para monitoreo de ganado ovino," 12 2025, supervisores: Martín Pedemonte, Julián Oreggioni.

[41] V. Cabrera, A. Delbuggio, H. Cardoso, A. Gómez, M. Pedemonte, R. Ungerfeld, and J. Oreggioni, "Design and fabrication of a sheep activity monitor platform," in *IEEE Engineering in Medicine and Biology Society (EMBC)*, Sydney, Australia, 2023.

[42] V. Cabrera, A. Delbuggio, H. Cardoso, D. Fraga, A. Gómez, M. Pedemonte, R. Ungerfeld, and J. Oreggioni, "Research platform to study sheep behavior," in *IEEE Conference on AgriFood Electronics (CAFE)*, 2023.

[43] ——, "Harnessing technology for livestock research: An online sheep behavior monitoring system," *IEEE Transactions on AgriFood Electronics*, pp. 1–8, 2024.

[44] A. Kurniawan, *Practical Contiki-NG: Programming for Wireless Sensor Networks*. Apress, 2018.

[45] A. L. Colina, A. Vives, A. Bagula, M. Zennaro, and E. Pietrosemoli, *IoT in Five Days*, rev 1.1 ed., 2016, last accessed 8 August 2025. [Online]. Available: https://github.com/marcozennaro/IPv6-WSN-book/releases/

[46] A. Augustin, J. Yi, T. Clausen, and D. Townsley, "A study of lora: Long range & low power networks for the internet of things," *Sensors*, vol. 16, no. 9, p. 1466, 2016.

[47] J. Finnegan and S. Brown, "A comparative survey of lpwa networking," *arXiv preprint arXiv:1802.04222*, 2018, last accessed 8 August 2025. [Online]. Available: https://arxiv.org/abs/1802.04222

[48] "Bases del desafío de trazabilidad animal." [Online]. Available: https://www.anii.org.uy/upcms/files/llamados/documentos/bases-desafio-trazabilidad-animal-vf-12.05.2025-1-.pdf

[49] N. Barreto, J. Oreggioni, and L. Steinfeld, "Energy harvesting and storage solutions for low-power iot devices in livestock industry," in *2024 IEEE 15th Latin America Symposium on Circuits and Systems (LASCAS)*, 2024.

[50] J. Vidmar, E. Eigbe, O. Oosterlee, and S. Sharma, "Delft Students On Software Architecture. Zephyr Project," 2019, delft University of Technology. Last accessed 8 August 2025. [Online]. Available: https://se.ewi.tudelft.nl/desosa2019/chapters/zephyr/#ModuleOrganization

[51] Zephyr Project members and individual contributors, "Zephyr Project documentation," 2024, last accessed 8 August 2025. [Online]. Available: https://docs.zephyrproject.org/latest/kernel/services/threads/index.html

[52] Nordic Semiconductor, "Asset Tracker v2 documentation," 2022, last accessed 8 August 2025. [Online]. Available: https://docs.nordicsemi.com/bundle/ncs-2.1.0/page/nrf/applications/asset_tracker_v2/README.html

[53] *LIS2DH12: MEMS digital output motion sensor, ultra-low-power high-performance 3-axis femto accelerometer*, STMicroelectronics, 2017, last accessed 25 November 2025. [Online]. Available: https://www.st.com/resource/en/datasheet/lis2dh12.pdf

[54] J. Nordby, M. Cooke, and A. Horvath, "Emlearn: Machine Learning inference engine for microcontrollers and embedded devices," 2019, last accessed 8 August 2025. [Online]. Available: https://doi.org/10.5281/zenodo.2589394

[55] N. Semiconductor, "Maximizing battery lifetime in cellular iot: an analysis of edrx, psm and as-rai," 2020.

[56] Nordic Semiconductor, "nRF9160 Product Specification v1.2," 2021, last accessed 8 August 2025. [Online]. Available: https://infocenter.nordicsemi.com/pdf/nRF9160_PS_v1.2.pdf

[57] NovAtel, "GPS Position Accuracy Measures, Application Note APN-029," 2020, last accessed 8 August 2025. [Online]. Available: https://www.gnss.ca/app_notes/APN-029_GPS_Position_Accuracy_Measures_Application_Note.html

[58] M. Tao and Tersus GNSS, "What is DOP in GNSS?" 2024, last accessed 8 August 2025. [Online]. Available: https://www.tersus-gnss.com/tech_blog/what-is-dop-in-gnss

[59] Nordic DevZone, "nRF9160 GNSS long time to fix with healthy RF electronics," 2021, last accessed 8 August 2025. [Online]. Available: https://devzone.nordicsemi.com/f/nordic-q-a/71656/nrf9160-gps-long-time-to-fix-with-healthy-rf-electronics/299904

[60] ——, "nRF9160 - no GPS fix, good C/N0 with 12 satellites," 2023, last accessed 8 August 2025. [Online]. Available: https://devzone.nordicsemi.com/f/nordic-q-a/107014/nrf9160---no-gps-fix-good-c-n0-with-12-satellites

[61] ——, "GPS accuracy discrepancy: printed vs actual co-ordinates," 2024, last accessed 8 August 2025. [Online]. Available: https://devzone.nordicsemi.com/f/nordic-q-a/117856/gps-accuracy-discrepancy-printed-vs-actual-coordinates-on-google-maps

[62] Circuit Dojo Community, "LTE/GPS on the nRF9160," 2022, last accessed 8 August 2025. [Online]. Available: https://community.circuitdojo.com/d/655-lte-gps-on-the-nrf9160

[63] M. Sebastián, E. Gonzalo, S. Andrés, and S. Leonardo, "Open-source cellular iot technologies coverage data collection system for precision agriculture," in *2024 IEEE 15th Latin America Symposium on Circuits and Systems (LASCAS)*, 2024, pp. 1–5.

[64] E. S. Fogarty, D. L. Swain, G. M. Cronin, L. E. Moraes, D. W. Bailey, and M. Trotter, "Developing a simulated online model that integrates gnss, accelerometer and weather data to detect parturition events in grazing sheep: a machine learning approach," *Animals*, vol. 11, no. 2, p. 303, 2021.

[65] D. Smith, J. McNally, B. Little, A. Ingham, and S. Schmoelzl, "Automatic detection of parturition in pregnant ewes using a three-axis accelerometer," *Computers and Electronics in Agriculture*, vol. 173, p. 105392, 2020.

[66] E. Fogarty, D. Swain, G. Cronin, L. Moraes, and M. Trotter, "Can accelerometer ear tags identify behavioural changes in sheep associated with parturition?" *Animal Reproduction Science*, vol. 216, p. 106345, 2020.

[67] S. C. Gurule, C. T. Tobin, D. W. Bailey, and J. A. Hernandez Gifford, "Evaluation of the tri-axial accelerometer to identify and predict parturition-related activities of debouillet ewes in an intensive setting," *Applied Animal Behaviour Science*, vol. 237, p. 105296, 2021.

[68] R. Sohi, F. Almasi, H. Nguyen, A. Carroll, J. Trompf, M. Weerasinghe, A. Bervan, B. I. Godoy, A. Ahmed, M. J. Stear, A. Desai, and M. Jois, "Determination of ewe behaviour around lambing time and prediction of parturition 7 days prior to lambing by tri-axial accelerometer sensors in an extensive farming system," *Animal Production Science*, vol. 62, pp. 1729–1738, 2022.

[69] K. E. Turner, F. Sohel, I. Harris, M. Ferguson, and A. Thompson, "Lambing event detection using deep learning from accelerometer data," *Computers and Electronics in Agriculture*, vol. 208, p. 107787, 2023.

This page has been intentionally left blank.

List of Tables

1.1 Behavior Classifiers with Commercial Data Loggers. 15
1.2 Custom made behavior classifiers. 16

2.1 Comparison of communication technologies. Adapted from Finnegan et al. [47]. In=Indoor, out=outdoor. 21
2.2 Microcontrollers comparison. 25

4.1 Original sample counts per behavior (200 Hz, no label merging). . . . 41
4.2 Sample counts after removing certain labels, merging trotting and running, and downsampling to 25 Hz. 41
4.3 Classifier features description. 42

6.1 Summary of mechanical design requirements for the OVIS device's enclosure. 58
6.2 Mock collar initial test. DVL: design team; IIE: electrical engineering team; FVET: veterinarian team; F = female, M = male. 61
6.3 First OVIS device overall test. 61
6.4 Attachment evaluation. 62
6.5 Robustness and durability evaluation. 63
6.6 Hermeticity evaluation. 63
6.7 Robustness and attachment. 64
6.8 Size and wool influence. 64
6.9 Different sizes test. 64
6.10 Útica test summary. 65
6.11 Final test summary. 65

7.1 Autonomy of the OVIS device for different settings 71
7.2 Comparison between experimental and estimated autonomies of the OVIS device. 73
7.3 Estimated effective battery capacity based on measurements and modeled profiles. 74
7.4 Projected autonomy of the low transmission operating mode. 75
7.5 Device - cloud server communication tests. 79
7.6 Ethogram used for labeling data. 80
7.7 Sample distribution of the OVIS data used for training and testing the second and third PC classifiers. 82

8.1 State-of-the-art comparison 85

A.1 Parturition related behavior classifiers. 89

List of Tables

D.1 Comparison between the two consumption models: I_{inact} inactive current, I_{act} active current, T_{inact} inactive time, T_{act} active time. 109

List of Figures

1.1 Functional diagram of the system. Figure adapted from [43]. 13

2.1 Hardware diagram of the device. Figure taken from [43]. 17
2.2 NB-IoT coverage provided by ANTEL in Uruguay (2025). 19
2.3 RRC state diagram in NB-IoT, showing the transitions between Connected, Uplink, Downlink, Idle, Paging, and PSM. 20

3.1 Layer diagram for Zephyr RTOS. Figure inspired from [50]. 28
3.2 Thread state machine for Zephyr RTOS. Diagram inspired from [51]. . 29
3.3 Module diagram for OVIS embedded software. Figure inspired from [52]. 33
3.4 Module work flow with and without thread. Figure inspired from [52]. 33

4.1 Confusion matrix for the RF trained and tested on the PC with external database from [15]. 44
4.2 Confusion matrix for RF model implemented in C. 45

5.1 Architecture of the OVIS cloud server solution. 47
5.2 Login screen. 50
5.3 Home screen. 51
5.4 Location map. 52
5.5 Pop-up menu to select time window. 52
5.6 User interface showing sheep location data. Figure taken from [42]. . . 53
5.7 Battery evolution over time. 54
5.8 Possible sheep behavior analysis. 54
5.9 Collar information displayed. 55
5.10 Data summary and export possibilities for an OVIS device. 55

6.1 Primary evaluation for buckles to use in the adjustable collar system. . 59
6.2 Mock collar used in initial tests without risking the OVIS device's electronics. 60
6.3 Photo compilation of the first OVIS device being worn by a female sheep. 61
6.4 Broken mock collar buckle (left) and new buckle used in the final enclosure (right). 62
6.5 Post rain test confirming OVIS device resistance. 63
6.6 Field observations where clear OVIS device rotation can be seen. . . . 64
6.7 Compilation pictures from Útica farm trial. 65
6.8 Compilation pictures from final FVET trial. 66
6.9 OVIS device's enclosure: case with assembly details (left, up), bottom case with internal components (right, up), strap and pad (left, down), strap and pad in the complete collar (right, down). Figure taken from [42]. 66

7.1 General schematic of PSM operation. Adapted from [55]. 70
7.2 Discharge of the OVIS device's for different settings (see Table 7.1) . . 70
7.3 Typical consumption profile of the OVIS device, showing the characteristic 50 s NB-IoT transmission cycle with low power intervals in PSM. 72
7.4 Examples of anomalous segments in the consumption profile. 72
7.5 GNSS trace using all collected data. The dispersion illustrates positional variability under static indoor conditions. 77
7.6 Filtered GNSS data under two conditions: low PDOP (left) and high satellite count (right). Both metrics are closely related, as satellite geometry and availability jointly determine positioning accuracy. . . . 77
7.7 Outdoor test under open sky conditions. A positional deviation of 11.1 m was observed despite static antenna placement. 78
7.8 GNSS positions recorded during an outdoor animal test. Despite the sheep remaining within the fenced paddock, the reported locations are scattered across nearby buildings and areas outside the enclosure. . . . 78
7.9 Confusion matrix for RF trained with the external database from [15] on the PC and tested with dataset gathered in the present study. . . . 81

C.1 Acrylic panels placement in the top cover to protect the solar panels. . 102
C.2 OVIS device enclosure being directly exposed to a constant flow of water. 102
C.3 Component placing inside OVIS device enclosure. 103

D.1 Example of a signal with transient dips below the 10 mA threshold. These intervals must be considered part of the same active period. . . 106
D.2 Ideal consumption profile, showing the active and inactive phases, the average current in each phase, and the duration of the phases and the total period. 107
D.3 Current profile corresponding to normal mode. Each period $T = 50\,\mathrm{s}$ includes the transmission of acceleration data, computed states, battery level, and signal strength. 111
D.4 Current profile corresponding to low transmission mode. The device stores samples locally and transmits only the states together with battery level and signal strength every nT, with $n \approx 271$. 111

This is the last page.
Compiled on Wednesday 15th April, 2026.
http://iie.fing.edu.uy/