**ESCUELA TÉCNICA SUPERIOR DE INGENIEROS INDUSTRIALES Y DE TELECOMUNICACIÓN**

**UNIVERSIDAD DE CANTABRIA**

***Trabajo Fin de Grado***

# Técnicas de sensado y análisis del comportamiento del ganado en explotaciones extensivas

## (Sensing and analysis techniques of livestock behavior in extensive farming)

**Para acceder al Título de**

***Graduado en Ingeniería de Tecnologías de Telecomunicación***

**Autor: Marina Cordovilla Serrano**

**Septiembre – 2022**

# GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

## CALIFICACIÓN DEL TRABAJO FIN DE GRADO

**Realizado por:** Marina Cordovilla Serrano
**Director del TFG:** Pablo Pedro Sánchez Espeso
**Título:** "Técnicas de sensado y análisis del comportamiento del ganado en explotaciones extensivas"
**Title:** "Sensing and analysis techniques of livestock behavior in extensive farming"

**Presentado a examen el día:** 22 de septiembre de 2022

para acceder al Título de

# GRADUADO EN INGENIERÍA DE TECNOLOGÍAS DE TELECOMUNICACIÓN

Composición del Tribunal:
Presidente (Apellidos, Nombre): Sánchez Espeso, Pablo Pedro
Secretario (Apellidos, Nombre): Fernández Solorzano, Víctor
Vocal (Apellidos, Nombre): Domingo Gracia, Marta

Este Tribunal ha resuelto otorgar la calificación de: .....

Fdo.: El Presidente

Fdo.: El Secretario

Fdo.: El Vocal

Fdo.: El Director del TFG
(sólo si es distinto del Secretario)

Vº Bº del Subdirector

Trabajo Fin de Grado N.º
(a asignar por Secretaría)

# Agradecimientos

En primer lugar, agradecer a mis padres el apoyo que me han dado durante la carrera, aunque a veces no entendiesen lo que les contaba. Pero sobre todo a mi madre, por estar siempre pendiente de que todo fuese bien.

También dar las gracias a mis amigos, compañeros de mención y a las personas que han ido uniéndose durante estos años, por animarme a mirar siempre hacia delante.

Gracias a CIFA y a Predictia por el soporte y ayuda prestada. Gracias a Raúl, investigador de CIFA, y a los pastores de La Jerrizuela, por implicarse tanto con nosotros, dedicarnos su tiempo y enseñarnos todo lo necesario a la hora de tratar con los animales.

Por último, agradecer a mi tutor del TFG, Pablo Sánchez, por darme la oportunidad de escoger este tema tan interesante cuando le dije que no quería quedarme sentada detrás de un ordenador y haberme ayudado a solucionar cualquier problema que me surgía durante el transcurso del proyecto.

# Resumen

El objetivo principal de este Trabajo de Fin de Grado es el desarrollo de técnicas de inteligencia artificial para el reconocimiento de la actividad animal. El estudio abarca desde el desarrollo del software del sensor, hasta el entrenamiento de árboles de decisión para identificar las acciones que realiza el animal, pasando por todo el proceso de captura de datos, procesado de la señal y anotación para el aprendizaje. El proceso de anotación ha sido complejo, ya que ha sido necesario grabar en video las acciones que realiza el animal para, posteriormente, anotar los datos de los sensores. La sincronización del video y de los datos de los sensores ha sido uno de los retos afrontados, así como la caracterización del comportamiento de los sensores.

Además de entrenar técnicas de inteligencia artificial, se ha desarrollado un sensor que ha sido mejorado con la experiencia adquirida durante las tomas de datos en campo. Este proceso ha dado lugar a 3 versiones del sensor, con diferente software embebido, gestión externa y tipo de montaje en el animal.

Los resultados muestran que las técnicas de aprendizaje utilizadas proporcionan resultados similares a trabajos publicados anteriormente. Con los datos capturados con el sensor desarrollado se obtienen resultados menos brillantes, como consecuencia del tipo de terreno en el que se están monitorizando los animales. Estos resultados ponen de manifiesto la dificultad para generalizar resultados en este campo y la necesidad, por tanto, de seguir trabajando en él.

# Abstract

The main purpose of this Final Degree Project is the development of Artificial Intelligence techniques for the animal activity recognition. The study ranges from the development of the sensor software to the training of decision trees to identify the actions conducted by the animal, going through the entire process of data capture, signal processing and annotation for learning. The annotation process has been complex, since it has been necessary to videotape the actions conducted by the animal to later record the data from the sensors. The synchronization of the video and the data from the sensors has been one of the challenges faced, as well as the characterization of the behavior of the sensors.

In addition to training artificial intelligence techniques, a sensor that has been improved with the acquired experience during field data collection has been developed. This process has led to three versions of the sensor, with different embedded software, external management, and type of mounting on the animal.

The results show that the learning techniques used provide related results to previously published works. Less brilliant results are obtained with the data captured with the developed sensor, because of the type of terrain in which the animals are being monitored. These results highlight the difficulty in generalizing results in this field and the need, therefore, to continue working on it.

# Índice general

# Índice de figuras

# Índice de tablas

# Lista de acrónimos

| | |
|---|---|
| AAR | *Animal Activity Recognition* |
| BORIS | *Behavioral Observation Research Interactive Software* |
| BSP | *Board support package* |
| CIFA | Centro de Investigación y Formación Agraria |
| DT | *Decision Tree* |
| DSP | *Digital Signal Processor* |
| FPU | *Floating-point Unit* |
| GPS | *Global Positioning Systems* |
| HAL | *Hardware Abstraction Layer* |
| IA | Inteligencia Artificial |
| IMU | *Inertial Measurement Units* |
| k-NN | *k-Nearest Neighbors* |
| MLC | *Machine Learning Core* |
| PLF | *Precision Livestock Farming* |
| RTC | *Real Time Clock* |
| RTOS | *Real Time Operating System* |
| SVM | *Support Vector Machines* |
| TIC | Tecnologías de la Información y la Comunicación |

# Capítulo 1. Introducción

## 1.1. Motivación

La ganadería extensiva es un tipo de producción ganadera en el que se aprovechan con eficacia los recursos del territorio donde se aplica. Algunos aspectos clave de este tipo de ganadería son la elección de razas autóctonas, la libre movilidad del ganado frente a la estabulación, el bienestar animal, así como la conservación y mejora del paisaje, la generación de alimentos de gran calidad y la conservación de los suelos y pastos [1].

En los últimos años, la ganadería de precisión (PLF, *Precision Livestock Farming*) ha sido uno de los sectores que más innovaciones ha incorporado al desarrollo de tecnologías orientadas a la mejora de la eficiencia en explotaciones ganaderas y de la calidad de los productos de origen animal. El objetivo de la ganadería de precisión es la gestión individual del animal, mediante la monitorización continua y en tiempo real de su salud, bienestar, capacidad de producción y reproducción e impacto en el entorno [2]. Dicha tecnología implica la introducción de dispositivos que incluyen Tecnologías de la Información y la Comunicación (TIC), como es el caso de Sistemas de Posicionamiento Global (GPS, *Global Positioning Systems*) y Unidades de Medición Inercial (IMU, *Inertial Measurement Units*), entre otros muchos. Estos sensores permiten la monitorización constante y la evaluación del comportamiento del ganado, así como el análisis de su actividad, facilitando el trabajo del ganadero al tiempo que mejoran las condiciones de vida del ganado, con un seguimiento individualizado de cada animal [3], [4].

En esta línea se están desarrollando tecnologías de reconocimiento de la actividad animal (AAR, *Animal Activity Recognition*) cuyo objetivo es la identificación de la actividad de un individuo o rebaño mediante el análisis de datos de video o sensores. Esta tecnología se está popularizando, debido a su utilidad en numerosos campos como la salud, la seguridad y la monitorización remota tanto de animales domésticos (mascotas) como de ganado de producción, tanto en explotaciones intensivas como extensivas [5].

La implantación de procesos basados en ganadería de precisión tiene varias fases:

- Captura y digitalización de información de comportamiento animal mediante sensores, sin perder de vista el importante papel que juegan las observaciones del ganadero, el cual es crucial para el buen funcionamiento de la explotación.
- Procesado y análisis de los datos obtenidos en la primera fase.
- Definición de estrategias que permitan extraer información relevante para el ganadero a partir del análisis de los datos capturados.
- Puesta en marcha de las medidas oportunas para optimizar el proceso de producción ganadera y bienestar animal.

## 1.2. Objetivos del proyecto

El objetivo de este Trabajo de Fin de Grado es estudiar y desarrollar sensores y técnicas de inteligencia artificial que permitan monitorizar y analizar el comportamiento del ganado en explotaciones extensivas. Esta información puede ser de gran utilidad a la hora de evaluar el rendimiento ganadero y mejorar el bienestar animal.

En particular, se estudiará el caso de rebaños de cabras y ovejas que hacen uso de pastos de montaña de forma extensiva. El trabajo se ha realizado en colaboración con el Centro de Investigación y Formación Agraria de Cantabria (CIFA), lo que ha permitido realizar medidas reales con los sensores desarrollados en el proyecto en una finca del Gobierno de Cantabria (La Jerrizuela en Los Corrales de Buelna) con rebaños de cabras y ovejas.

A lo largo del proyecto se abordan los siguientes puntos:

- Desarrollo de sensores para la captura del comportamiento animal, prestando especial atención al desarrollo de la parte software. Tanto el software como algunos aspectos del hardware del sensor se han modificado gracias a la experiencia adquirida durante con su uso en entornos reales.
- Captura de datos reales en campo con dichos sensores y tratamiento previo al análisis de estos. Los sensores se han colocado tanto en cabras como en ovejas.
- Análisis de los datos obtenidos empleando técnicas de Inteligencia Artificial. Estas técnicas se han entrenado con datos de los sensores y permiten detectar de forma automática del comportamiento animal.

## 1.3. Estructura del documento

El Trabajo Fin de Grado se divide en 8 capítulos, siendo esta introducción el primero de ellos.

En el Capítulo 2 se presenta un estudio de trabajos anteriores sobre los temas a tratar en el proyecto.

Seguidamente, en el Capítulo 3, se describe el hardware empleado, así como el diseño de una caja auxiliar para el sensor. En el Capítulo 4, se describe el software desarrollado a lo largo del proyecto y sus distintas versiones.

En el Capítulo 5 se describe el proceso de calibración del sensor y se explican los pasos llevados a cabo para la toma de datos en campo.

A continuación, en el Capítulo 6, se explica cuál ha sido el módulo de Inteligencia Artificial empleado, así como su proceso de configuración. En el Capítulo 7, se describe el conjunto de datos y la manera de procesar los datos y entrenar al algoritmo de Inteligencia Artificial.

Finalmente, en el Capítulo 8, se presentan las conclusiones y líneas futuras que pueden ofrecer los sensores desarrollados en este proyecto.

# Capítulo 2. Estado del arte

Durante los últimos años se ha producido un enorme avance, tanto en las tecnologías de implementación de sistemas electrónicos y sensores como en el desarrollo de técnicas de inteligencia artificial para el análisis de actividad humana y animal. Estos avances han supuesto un incremento considerable en el interés por el desarrollo de tecnologías que permitan el análisis de la actividad del ganado.

Para poder analizar el comportamiento animal es necesario disponer de dispositivos que capturen los datos. Dicha información será posteriormente procesada y analizada [6]. En este capítulo se estudiarán brevemente técnicas de captura y procesado de la información con algoritmos de inteligencia artificial.

## 2.1. Captura de datos

Para analizar el comportamiento animal se han utilizado principalmente 3 tipos de sensores:

- Sensores de movimiento, que determinan la aceleración o velocidad angular de la actividad del animal.
- Módulos GPS, que proporcionan las coordenadas en las cuales se encuentra el elemento monitorizado.
- Sensores fisiológicos, que miden parámetros físicos como la temperatura del individuo.

Otros tipos de sensores, como cámaras o balanzas, se utilizan en un número reducido de trabajos. Tanto a nivel de investigación como comercial, los sensores de movimiento son los dispositivos más utilizados para capturar información sobre el comportamiento animal. Por esta razón, se realiza a continuación un estudio de los parámetros que afectan a estos sensores.

### *Sensores de movimiento*

La mayoría de los sensores de movimiento empleados para el análisis de la actividad animal incluyen únicamente acelerómetros. Estos sensores miden la aceleración de los movimientos del animal en tres ejes ortogonales (X, Y, Z), proporcionando 3 valores de salida. Además de la aceleración del movimiento, los sensores también miden la gravedad (vector g). La popularidad de estos dispositivos nace de su bajo coste y consumo, así como de los buenos resultados que normalmente proporcionan para el reconocimiento de la actividad humana y animal. Como limitaciones de estos dispositivos se pueden citar el ruido y la dificultad para separar la aceleración de la gravedad de otras fuentes de aceleración. Por ello, cada vez más estudios combinan el acelerómetro con otros sensores, como giroscopios y magnetómetros, lo que facilita determinar la orientación y posición del sensor y del animal. Un giróscopo es un sensor que mide la velocidad angular en los 3 ejes del acelerómetro, lo que permite utilizar técnicas para reducir el ruido de este (por ejemplo, filtros Kalman) y/o determinar la posición del sensor y el animal. Un magnetómetro es un sensor que determina la posición del campo magnético terrestre (brújula digital). Normalmente se utiliza para mejorar la determinación de la orientación

del animal y reducir el impacto del vector gravedad en las medidas. En este trabajo se han capturado los datos de los 3 sensores (acelerómetro, giroscopio y magnetómetro), aunque solo se ha utilizado el acelerómetro para el análisis.

### *Frecuencia de muestreo del sensor*

La mayoría de los estudios emplean frecuencias de muestreo entorno a los 10 Hz, ya que si emplean frecuencias menores se ha observado un decremento en la calidad de los resultados. Sin embargo, en algunos casos se han empleado frecuencias de muestreo por encima de los 20 Hz (hasta 100Hz), con objeto de mejorar la efectividad del análisis [6], [7]. La frecuencia de muestreo del sensor es clave para determinar el consumo del sistema de captura y, por ello, la duración de la batería. Como los sensores capturan información de forma autónoma, interfiriendo lo menos posible con la actividad del individuo, la reducción del consumo es un aspecto clave del diseño. Por ello es necesario reducir la frecuencia de muestreo lo máximo posible, para de esta forma poder alargar la vida de la batería y la operación del dispositivo.

### *Posición del sensor en el animal*

La posición del sensor en el animal es un punto muy importante para el análisis posterior y viene determinado por el objetivo del estudio. Si lo que se desea es realizar estudios relacionados con la alimentación, la posición óptima será en el cuello del animal. Si por el contrario se desea realizar un estudio sobre los movimientos que realiza el animal cuando camina (por ejemplo, determinar si cojea), la posición óptima será en las patas [6]–[8]. Para estudios de la actividad del ganado que incluyan varias situaciones diferentes, como movimiento, rumia, rascado o alimentación, la mayoría de los trabajos asumen que los sensores están colocados en el cuello del animal.

## 2.2. Procesado de datos

Para procesar los datos recogidos, la mayor parte de los trabajos agrupa o comprime la información en ventanas temporales. Para ello, la serie temporal es dividida en segmentos de tamaño *w*, cuyo valor depende de la acción o movimiento del animal que se está estudiando. Además, las ventanas de datos pueden solaparse (*overlapping*), de forma que una ventana comparte parte de sus datos con la ventana anterior y la siguiente [9].

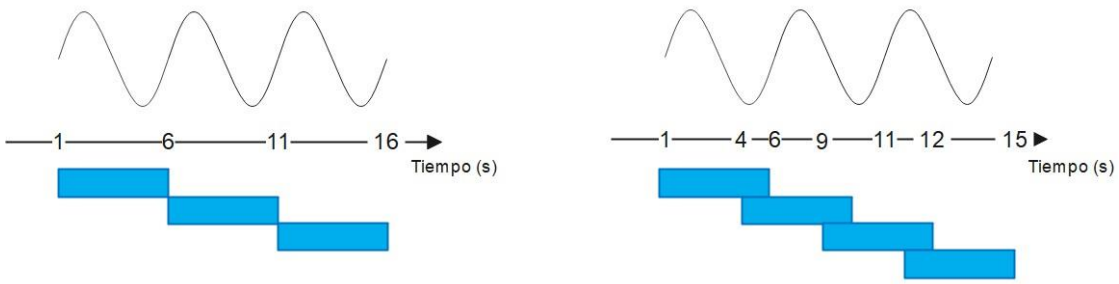


*Ilustración 1. Ventanas temporales sin solapamiento (izq.) y con solapamiento (dcha.)*

A partir de los datos de cada ventana se extraen una serie de características o *features*. Las *features* se extraen típicamente en el dominio del tiempo o/y la frecuencia e incluyen parámetros asociados a los datos de la ventana como la media o la mediana, así como parámetros de dispersión como la varianza, la desviación estándar, el mínimo, el máximo, etc. El objetivo es reducir a un número pequeño de valores significativo el conjunto de datos de la ventana (típicamente las muestras obtenidas del acelerómetro durante un tiempo que varía entre 1 y 10 segundos). Estos valores significativos asociados a cada ventana son los que utilizan las técnicas de inteligencia artificial para extraer información. El objetivo del procesado de datos es, por lo tanto, reducir el número de valores a analizar minimizando el ruido y la correlación entre datos [8], [10].

## 2.3. Técnicas de Inteligencia Artificial

### *Aprendizaje Automático o Machine Learning*

*Machine Learning* o Aprendizaje Automático es una disciplina de Inteligencia Artificial cuyo objetivo es desarrollar aplicaciones de ordenador que puedan aprender de los datos. Para poder aplicar técnicas de Aprendizaje Automático a problemas diversos, se han desarrollado una serie de librerías y entornos de programación que facilitan su uso. Una de las librerías más populares es *scikit-learn*, un entorno *Python* que permite entrenar y utilizar varias técnicas de Aprendizaje Automático de forma sencilla y directa [11]. En *Machine Learning* existen dos tipos de algoritmos, dependiendo del tipo de aprendizaje:

- **Algoritmos de aprendizaje supervisado**
  Estas técnicas precisan de un conjunto de datos en los cuales previamente se han identificado las caracteriza que deben ser aprendidas por el programa. Este conjunto de datos anotados (o *data-set*) se utiliza para guiar el proceso de entrenamiento y la validación de los resultados de este. La gran mayoría de los algoritmos de aprendizaje supervisado forman parte del entorno *scikit-learn*. Se incluyen en esta categoría modelos lineales generalizados, como la regresión lineal, las máquinas de vectores de soporte (SVM), los árboles de decisión y los métodos bayesianos [11].

- **Algoritmos de aprendizaje no supervisado**
  A diferencia del caso anterior, estas técnicas no utilizan un conjunto de datos anotados. Por ello, en lugar de adquirir el conocimiento a partir de casos predefinidos, estos algoritmos extraen información de las similitudes que existen entre los elementos que procesan, agrupándolos en grupos con características similares. Al igual que en el caso de los algoritmos de aprendizaje supervisado, *scikit-learn* cuenta con la implementación eficiente de multitud de técnicas para aprendizaje no supervisado, como son la agrupación, el análisis factorial, el análisis de componentes principales y las redes neuronales no supervisadas [11].

A continuación, se resumen algunas de las técnicas de aprendizaje automático empleadas habitualmente para reconocimiento de la actividad:

- **Árboles de decisión o *Decision Trees* (DT)**
  Son algoritmos de aprendizaje supervisado que se utilizan típicamente en tareas de clasificación y regresión. Las técnicas de clasificación determinan la clase a la que pertenece el elemento que se analiza mientras que las de regresión generan valores continuos asociados al mismo. Los árboles de decisión son modelos predictivos que dividen el espacio de decisiones en regiones no solapadas, de forma que cada observación que es asignada a una región predice el valor de respuesta de esta. Básicamente se emplea una estrategia de divide y vencerás para alcanzar una conclusión (clasificación) a parir de los datos de entrada [11], [12]. Los árboles están formados por nodos, los cuales plantean una disyuntiva con diferentes opciones. Existen tres tipos de nodos:
    - *Nodo raíz*: representa la población completa o muestra.
    - *Nodos de decisión*. Después de la primera división, estos nodos dividen la muestra en función de los valores que tomen las diferentes variables de esta.
    - *Nodos terminales*: indican el final de la muestra y se corresponden con las clases o categorías a identificar.

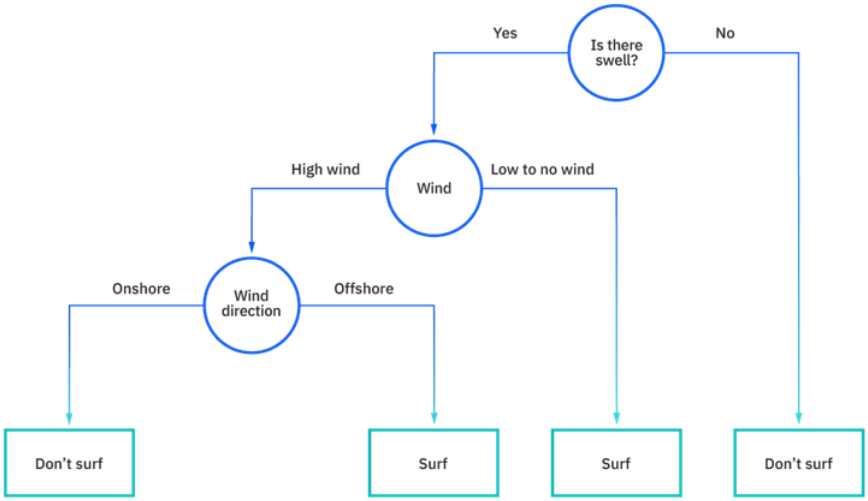


*Ilustración 2. Estructura de un árbol de decisión y ejemplo de este*

Los árboles de decisión son, por lo general, algoritmos sencillos de implementar eficientemente, y resuelven con buenos resultados actividades diversas, como saber si el animal está tumbado, de pie, andando, etc. En la clasificación de actividades con DT normalmente se utilizan los datos capturados por acelerómetros colocados en el cuerpo del animal.

- ***Random Forest***
  Los modelos de *Random Forest* utilizan un conjunto de árboles de decisión, donde cada árbol utiliza una parte de los datos de entrenamiento (*bagging)*. La respuesta de los árboles de decisión es combinada utilizando técnicas como voto suave (soft-voting). Esta estructura permite reducir la varianza de las predicciones a través de la combinación de los resultados de varios clasificadores, cada uno de ellos entrenado con diferentes subconjuntos tomados de la misma población. Una vez que los árboles llegan a los nodos terminales, se combinan las predicciones, obteniendo la predicción final del modelo mediante la función media, por ejemplo. El principal problema de los árboles de decisión es que tienden a sobre-

ajustar (*overfit*) los datos de entrenamiento, lo que permite generar muy buenos resultados a costa de una pobre generalización del algoritmo a situaciones que no han aparecido directamente durante el entrenamiento [11].

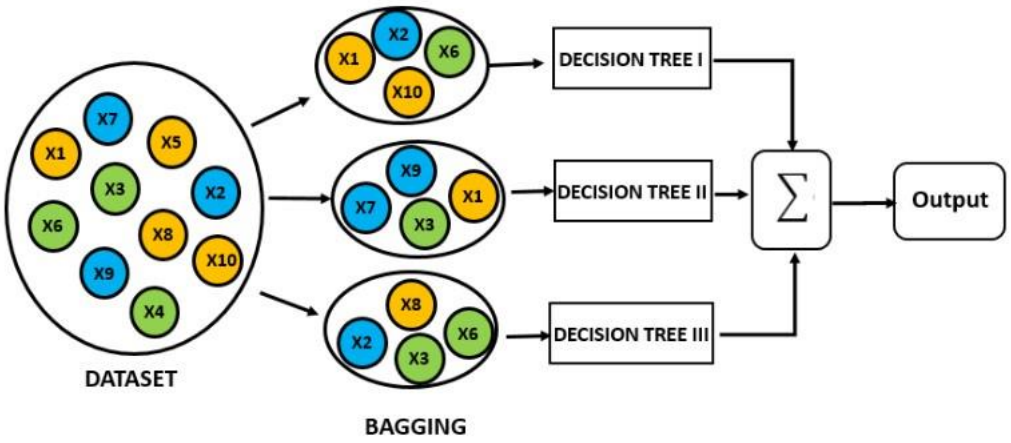


*Ilustración 3. Estructura del modelo de Random Forest*

Al igual que los árboles de decisión, este tipo de modelo se emplea para detectar las distintas actividades que realizan los animales.

- **Algoritmos de *boosting***

  Al igual que el algoritmo *Random Forest*, este tipo de técnicas combina algoritmos más simples. En este caso, los modelos se entrenan secuencialmente. A las instancias de entrenamiento más difíciles de predecir, es decir, las que producen el mayor error, se les asigna un mayor peso, para que los siguientes modelos le den más importancia. Cada modelo se enfoca en aprender de los datos, pero también en especializarse en las instancias difíciles que han complicado a los submodelos anteriores, de forma que un modelo que genera excelentes predicciones tendrá más influencia sobre la decisión final [11].

  El objetivo de las técnicas de *Random Forest* es reducir el sesgo en vez de la varianza.

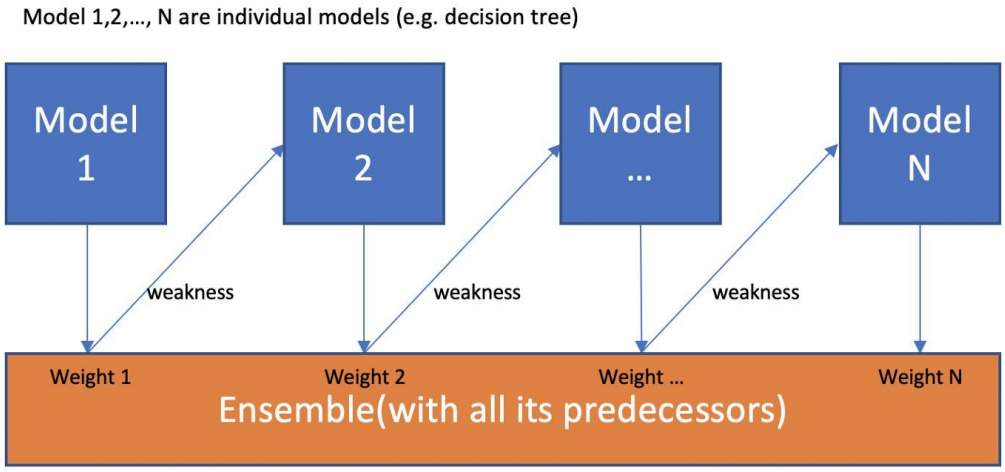


*Ilustración 4. Estructura del modelo de boosting*

- **Vecinos más cercanos o k-NN (*k-nearest neighbors*)**
  El algoritmo de k vecinos más cercanos, también conocido como KNN o k-NN, es un clasificador de aprendizaje supervisado, que utiliza la proximidad para hacer clasificaciones o predicciones sobre la agrupación de un punto de datos individual, basándose en la búsqueda de los elementos más similares al que se quiere predecir dentro de un conjunto de datos. Para medir esa similitud entre los elementos, se usan diferentes medidas de la distancia como por ejemplo la distancia euclidiana, la distancia Manhattan o la distancia Minkowski entre otras.

  El valor de k en el algoritmo define cuántos vecinos se verificarán para determinar la clasificación de un elemento específico. En función de la medida de distancia elegida, se ordenan los elementos y se seleccionan los k más cercanos.
  Una vez que se han obtenido los valores de cada uno de los k seleccionados, se aplica una media estadística para obtener la predicción [11].

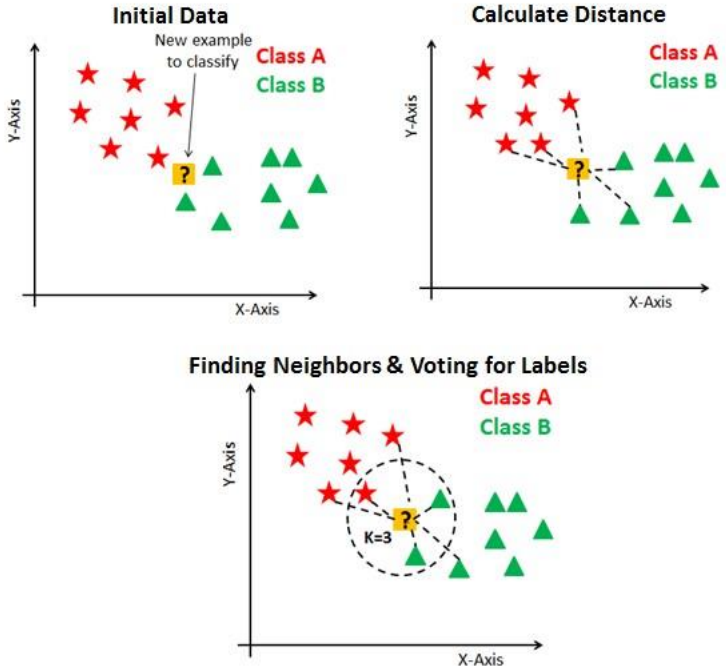


*Ilustración 5. Ejemplo del modelo de k-NN*

- **Máquinas de vectores de soporte o *Support Vector Machines* (SVM)**
  Dado un conjunto de puntos, en el que cada uno de ellos pertenece a una de las dos posibles categorías, un algoritmo basado en SVM construye un modelo capaz de predecir si un punto nuevo (cuya categoría se desconoce) pertenece a una categoría o a la otra.
  En este método, los datos de entrada son vistos como un vector p-dimensional (una lista ordenada de p números). La SVM busca un hiperplano que separe de forma óptima los puntos de una clase de la de otra. En el concepto de "separación óptima" es donde reside la característica fundamental de las SVM: este tipo de algoritmos buscan el hiperplano que tenga la máxima distancia con los puntos que estén más cerca de él mismo. Por esta razón, a veces se les conoce a las SVM como clasificadores de margen máximo. De esta forma, los puntos del vector que

son etiquetados con una categoría estarán a un lado del hiperplano y los casos que se encuentren en la otra categoría estarán al otro lado [11].

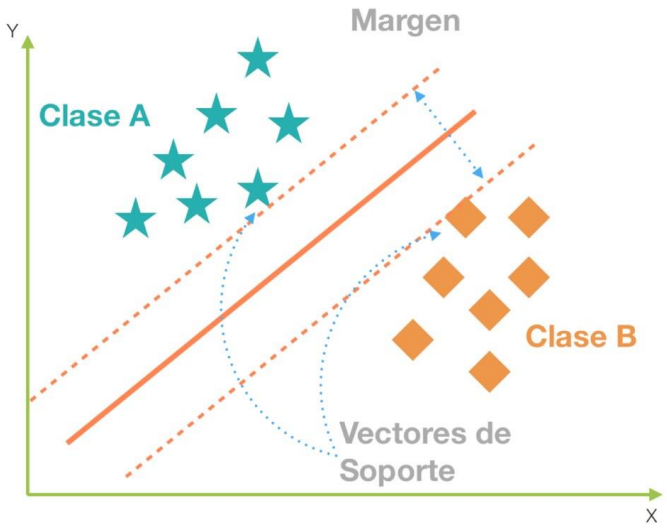


*Ilustración 6. Ejemplo de modelo de SVM*

## *Entrenamiento*

Los modelos anteriormente presentados se entrenan con un conjunto de datos anotados o *dataset*. Dicho conjunto será dividido en dos subconjuntos que serán utilizados en tareas diferentes: una parte amplia (entre el 70% y 80% de los datos) será utilizado como datos de entrenamiento mientras que el resto de las muestras se utilizarán en la validación. La división entre el conjunto de datos de entrenamiento y validación es un aspecto clave para optimizar el proceso de entrenamiento y puede producir situaciones de sobreajuste (*overfit*). Para evitar este problema se han propuesto diversas técnicas para realizar la partición del conjunto de datos anotados. Dichas técnicas son:

- ***Random Split***
  Esta división implica una partición completamente aleatoria de los datos. Por lo tanto, una vez que los datos se cargan y se procesan, se mezclan y se dividen en subconjuntos de entrenamiento. Debido a la completa aleatorización, se mezclan los datos de todas las grabaciones y por lo tanto los datos de todos los animales y tiempos. Por ello, este tipo de aproximación pierde capacidad de generalización [11].

- **Validación cruzada**
  Para evitar desperdiciar demasiados datos de entrenamiento en conjuntos de validación, una de las técnicas más comúnmente usadas es la validación cruzada: el conjunto de entrenamiento se divide en subconjuntos complementarios y cada modelo se entrena con una combinación diferente de estos subconjuntos y se valida con las partes restantes.

  En este proyecto, los datos de entrenamiento serán todos los datos correspondientes a cada animal y día, a excepción de uno de los conjuntos, que se usará para validar el modelo entrenado. Además, al realizar una validación

cruzada, todas las grabaciones pasarán a formar parte del conjunto de datos de entrenamiento [11].

# Capítulo 3. Hardware del sistema utilizado para monitorizar animales

Como regla general, el hardware de los sensores empleados para reconocimiento de la actividad animal utiliza el esquema de la figura adjunta:

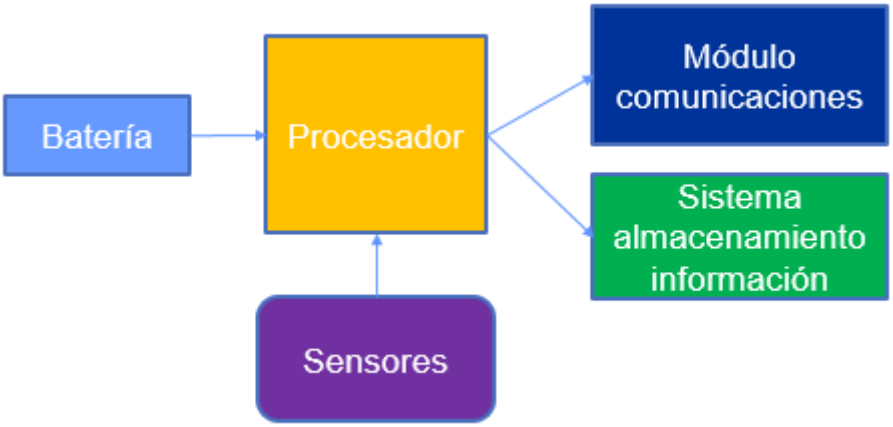


*Ilustración 7. Esquema del hardware de los sensores*

El sistema se constituye alrededor de un microcontrolador de bajo consumo, que ejecuta el software embebido. Dicho sensor es alimentado por una batería y recibe datos de los sensores del módulo. Además, es capaz de guardar información en un sistema de almacenamiento interno. Periódicamente, el módulo la transmite a la nube la información capturada utilizando técnicas de corto alcance (como BLE) o de telefonía móvil (red 4G, etc.).

## 3.1. Plataforma hardware seleccionada

En primer lugar, se realiza un estudio de los componentes hardware que se encuentran disponibles en el mercado para poder implementar el sensor. Tras dicho análisis, se decide utilizar el módulo comercial SensorTile.box (STEVAL-MKSBOXV1) de *STMicroelectronics*, ya que es el que mejor se ajusta a los requerimientos del sistema [13].

*Ilustración 8. SensorTile.box. Placa y caja del sensor*

Algunas de las ventajas más relevantes para el proyecto del módulo comercial SensorTile.box son:

- Posee los tres sensores necesarios para el desarrollo del proyecto: acelerómetro, giroscopio y magnetómetro.
- El módulo proporciona ejemplos de software embebido que facilitan su programación. Dicho software es de código abierto y modificable por el usuario, lo cual es imprescindible si existe la necesidad de cambiar las características de funcionamiento de los diferentes sensores.
- El módulo integra adicionalmente la función de *data logger* y la capacidad de transmisión de datos de forma inalámbrica.
- Por último, es un dispositivo con programación multiplataforma ya que el entorno de desarrollo está basado en Eclipse y puede ejecutarse en Windows, Linux y macOS. En cambio, otras plataformas similares están limitadas a ser programadas desde Windows.

## *Características de la placa empleada*

El módulo SensorTile.box consta de varios sensores de alta precisión, equivalentes a una IMU (*Inertial Measurement Unit*) de 9 ejes:

- Sensor de temperatura digital (STTS751)
- IMU de 6 ejes (acelerómetro y giroscopio) que cuenta con hardware específico para implementar algoritmos de inteligencia artificial (LSM6DSOX)
- Acelerómetros de 3 ejes con mayores rangos de aceleración que el acelerómetro citado anteriormente (LIS2DW12 y LIS3DHH)
- Magnetómetro de 3 ejes, el cual combinado con los acelerómetros y giroscopios anteriores forman una IMU de 9 ejes (LIS2MDL)
- Altímetro / sensor de presión (LPS22HH)
- Micrófono / sensor de audio (MP23ABS1)
- Sensor de humedad (HTS221)

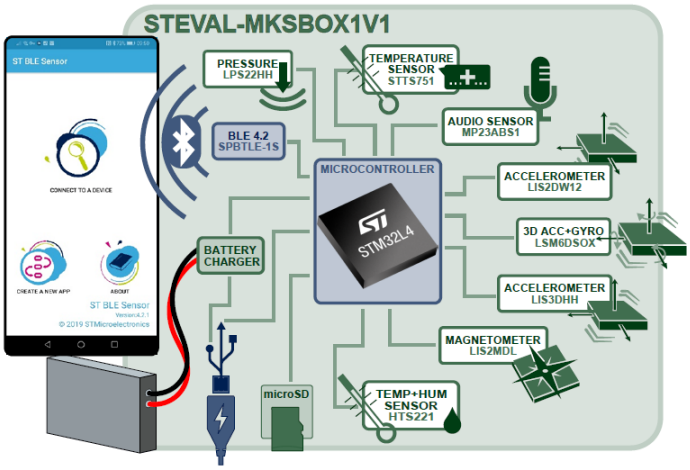


*Ilustración 9. Diagrama de bloques de SensorTile.box*

El módulo incluye un microcontrolador ARM Cortex-M4 (STM32L4R9) de potencia ultra baja con procesador de señales digitales (DSP, *Digital Signal Processor*) y unidad de punto flotante (FPU, *Floating-point Unit*). Además, integra componentes que proporcionan comunicaciones mediante conectividad *Bluetooth Smart v4.2* (BlueNRG-M2) e interfaz USB, una tarjeta de almacenamiento de datos tipo microSD y alimentación por batería de Ion-Litio. Dicha batería proporciona una autonomía de aproximadamente 7 horas, conforme a pruebas realizadas de vida útil de la batería) [13].

El peso del módulo es reducido, lo que minimiza el impacto del sensor en el comportamiento del animal en campo.

## 3.2. Diseño de la caja auxiliar

Durante las pruebas con el sensor se detectó la necesidad de desarrollar una caja que permitiese sujetar de manera más segura el sensor al collar del animal y, además, redujese la cantidad de ruido captado por el micrófono del sensor. En micrófono se utiliza principalmente para detectar cuando el animal mastica y/o muerde la comida. A continuación, se puede observar cuál fue el primer montaje del sensor en los collares:

*Ilustración 10. Primera versión del montaje del collar*

Para solucionar los problemas de aislamiento acústico del sensor, se diseñó una caja de plástico con aislamiento en las paredes, con la intención de minimizar la captura de sonido ajeno al animal. La caja atenúa los ruidos externos por encima de 6dB. Otra de las funciones de la caja es la de facilitar la colocación del sensor en el collar, gracias a dos asas laterales.

Además de la caja, se ha desarrollado un amplificador acústico que facilita la captación de sonido en una única dirección. Dicho amplificador posee una membrana con un diámetro mayor que el del tubo sonoro que comunica con el micrófono, lo que permite una ganancia de aproximadamente 3dB.

A continuación, se muestra el diseño de la caja y el amplificador:


*Ilustración 11. Caja con mejoras*

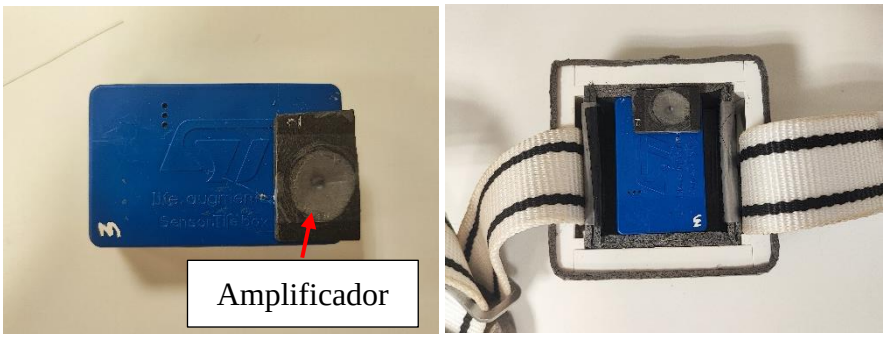


*Ilustración 12. Amplificador acústico*

# Capítulo 4. Software desarrollado en el proyecto

Como se ha mencionado en las características del hardware, el módulo incluye un microcontrolador de bajo consumo en el cual se ejecuta el software del sensor. Además, se dispone de una interfaz de programación y depuración del software embebido, que permite desarrollar código complejo utilizando el entorno de desarrollo abierto STM32 (STM32 ODE), que incluye un paquete de funciones de inteligencia artificial (IA) .

A lo largo del proyecto, se desarrollan cuatro softwares distintos, cada uno con una función específica. Además, se emplea un software comercial para configurar módulos internos de los sensores para inteligencia artificial:

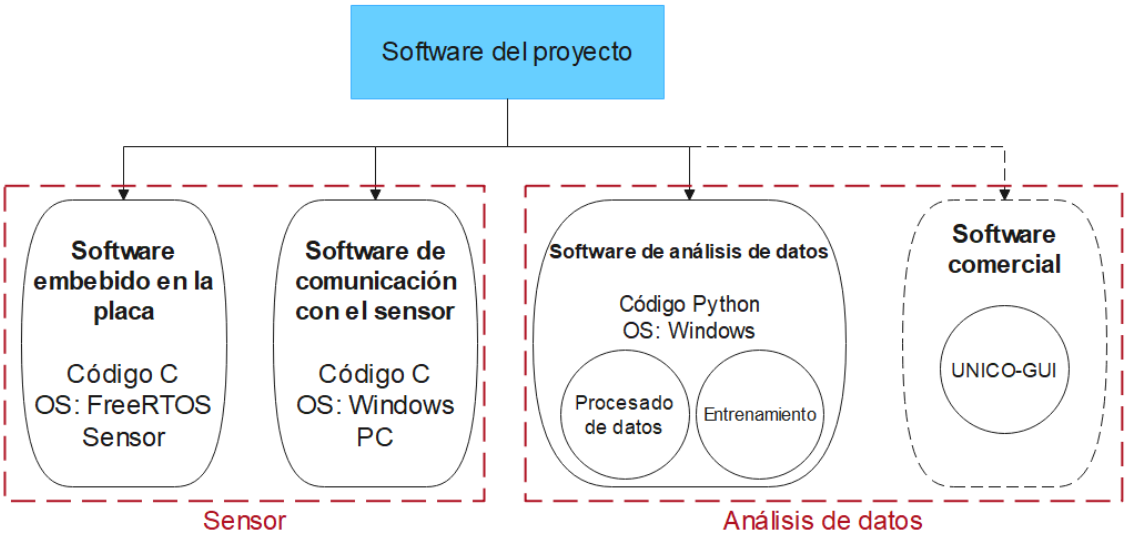


*Ilustración 13. Esquema del software empleado en el proyecto*

En este capítulo se va a presentar exclusivamente el software que gestiona el sensor, es decir, el software embebido en la placa y el software de comunicación con el sensor. El software dedicado al análisis de datos se explica en capítulos posteriores.

## 4.1. Sistema operativo

La mayoría de los sistemas operativos permiten ejecutar varios hilos de ejecución al mismo tiempo, es decir, son sistemas multitarea. Como cada núcleo de un procesador solo puede ejecutar un único hilo en un momento determinado, es el planificador del sistema el encargado de decidir qué hilo ejecutar y cuándo.

El planificador en un sistema operativo en tiempo real (RTOS, *Real Time Operating System*) está diseñado para proporcionar un patrón de ejecución predecible, lo que es muy interesante para sistemas embebidos ya que suelen tener requisitos de tiempo real. Un requisito de tiempo real es aquel que especifica que el sistema embebido debe responder a un determinado evento dentro de un tiempo estrictamente definido [14].

En el caso de este proyecto, el tipo de sistema operativo empleado en el módulo de sensores es *FreeRTOS*, un tipo de RTOS diseñado para ser lo suficientemente pequeño

como para ejecutarse en un microcontrolador, aunque su uso no se limita exclusivamente a aplicaciones de microcontroladores.

Por otro lado, el ordenador desde el que se desarrollan los distintos códigos de software cuenta con sistema operativo Windows. Por ello, se emplean programas de desarrollo que puedan ejecutar desde dicho sistema operativo como son Eclipse y el entorno de programación propio del fabricante del sensor, *STM32CubeIDE* el cual está basado en Eclipse.

## 4.2. Descripción del software del sensor

A continuación, se describen las diferentes versiones del código que se han ido generando a lo largo del proyecto:

### *Versión 1. Transmisión de datos en tiempo real*

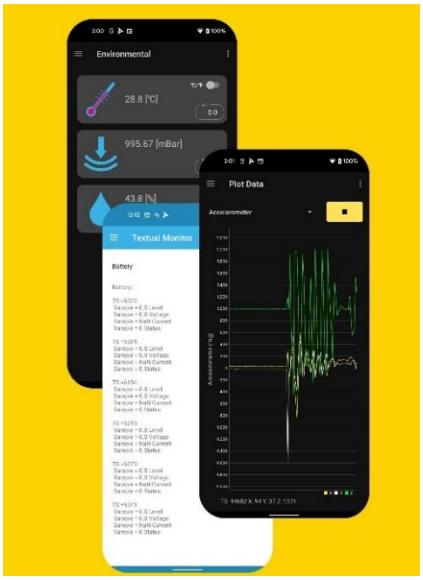


*Ilustración 14. Aplicación ST BLE Sensor*

En el estado del arte se observó que en algunos proyectos se han empleado sensores que transmiten los datos capturados en tiempo real a un dispositivo móvil. En estos casos, los animales se encuentran en recintos cerrados, por lo que las distancias entre los sensores y los equipos de almacenamiento de datos eran conocidas.

En la primera versión del software, se recogen los datos de los sensores de movimiento (acelerómetro, giroscopio y magnetómetro) y se transmiten mediante la interfaz de Bluetooth a la aplicación móvil *ST BLE Sensor*.

Esta primera versión del software fue evaluada en el laboratorio, donde desde un primer momento se producían pérdidas de señal entre los dispositivos tras varios minutos de funcionamiento y, por lo tanto, la pérdida de datos, invalidando de esta forma las mediciones. Tras llevar a cabo varias pruebas para intentar resolver el problema, se llegó a la conclusión de que el fallo se encontraba en la comunicación entre el módulo de Bluetooth y el dispositivo móvil, ya que los problemas surgían cuando el dispositivo móvil no interactuaba continuamente con el sensor o se mantenía en stand-by. Es por esta razón, por la que se decidió desarrollar una segunda versión del software del sistema, dejando de lado el uso del módulo de Bluetooth.

### *Versión 2. Almacenamiento de datos en la memoria SD*

En la segunda versión del software, se decide almacenar los datos del sensor en la tarjeta SD. Posteriormente, tras extraer la tarjeta del sensor, los datos son procesados para una correcta interpretación.

En esta versión se comenzaron a almacenar los datos recogidos por el micrófono para poder ampliar el tipo de datos disponibles. Es en este momento cuando se detectan

numerosos problemas, como la captura de sonidos que no provienen del animal que porta el sensor (ruido de otros animales o de los propios pastores e investigadores) o pérdida de la intensidad de la señal debido al roce del sensor con el pelaje o lana del animal, así como el sonido de los cencerros. Para intentar solucionar este problema, se diseña una caja cuyo diseño se ha explicado en apartados anteriores.

Una de las razones por las que se decide grabar el sonido de los animales es porque se observa que algunos mordiscos son audibles. Tras procesar los datos, algunos mordiscos se pueden ver como líneas transversales en el espectrograma.

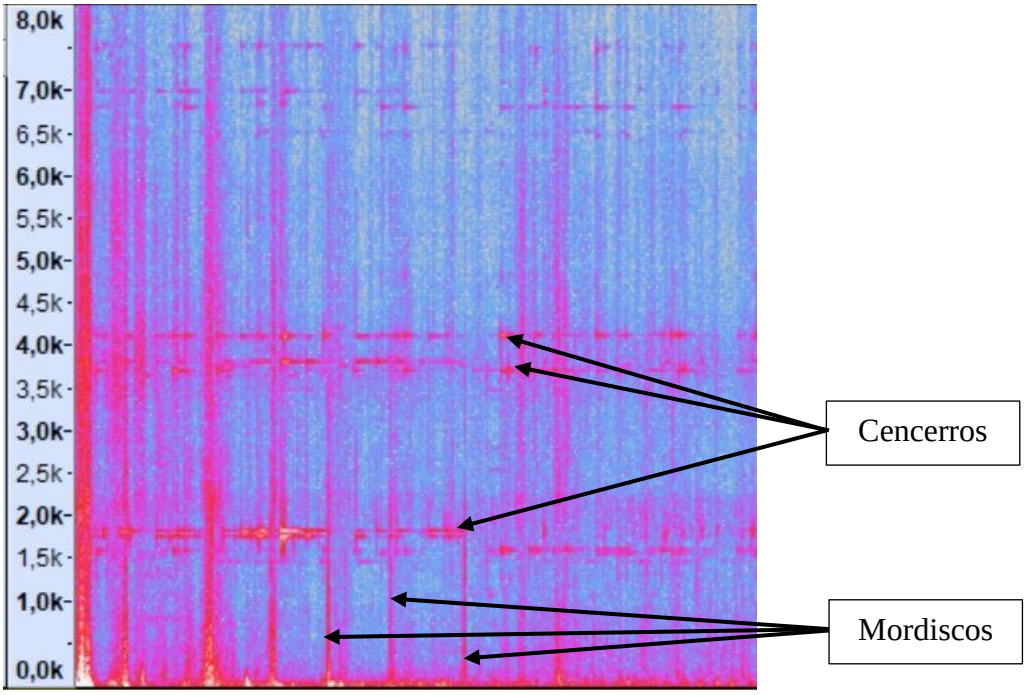


*Ilustración 15. Espectrograma de la señal de audio*

Otro problema encontrado es la continua necesidad de desmontar el sensor de los collares, así como tener que desatornillar la caja protectora del sensor cada vez que se quiere extraer la tarjeta SD para observar los datos obtenidos. Por esta razón, se crea una nueva versión del software, la cual permite el control del sensor y el acceso a los datos a través de la interfaz USB.

## *Versión 3. Optimización de la captura y sincronización de los datos*

En la última versión del software, se ha puesto el foco en la mejora de la captura de datos de los sensores y en la optimización de la sincronización. El proceso de captura de datos para anotación es complejo. Por una parte, tenemos un sensor colocado en el animal. Por otra, tenemos que saber que está haciendo el animal para anotar los datos de entrenamiento/validación. Para ello es necesario grabar un video en el que se observen las actividades del animal.
Uno de los mayores problemas encontrados es la sincronización entre los datos capturados por el módulo y la secuencia de video que permite identificar los distintos comportamientos del animal. Si el reloj del sensor y de la cámara no están sincronizados,

no es posible establecer una correspondencia entre los datos del sensor y los comportamientos observados en los vídeos. Por esta razón, en esta revisión del software se ha prestado especial atención a la sincronización del sensor con una fuente externa, en este caso el reloj del ordenador, lo que permite sincronizar los vídeos con precisión de segundos. A continuación, se describe la última versión del software desarrollado.

## *Software de comunicación entre el PC y el sensor*

Para poder llevar a cabo la sincronización del sensor con el ordenador se ha desarrollado un software en el PC, que permite el control del sensor y el acceso a sus datos a través de la interfaz USB.

```
Found port: COM5.
Found port: COM11.
Found port: COM13.
Found port: COM3.
Found port: COM12.
Found port: COM6.
Found port: COM10.
Found port: COM4.
Found port: COM8.  VID=483 PID=5740
Found 9 ports.

Selected port COM8
Connected!

Sensor ID:MEMS shield demo,101,7.1.0,0.0.0,MKSBOX1V1

Menu:
        0 - Disconnect
        1 - Ping
        2 - Sensor ID
        3 - Get Status
        4 - Pulse button
        5 - Get time/date
        6 - Set time/date
Option?1
Ping ok!
```

*Ilustración 16. Software de control y sincronización del sensor mediante USB*

Como se puede observar en la captura, una vez que se ejecuta el programa, se informa de los puertos COM. A continuación, se detecta el puerto en el que se encuentra el sensor y se informa del identificador ( ID) asignado al módulo.
Además, se ha desarrollado un menú con distintas opciones, mediante las cuales se puede interactuar con el sensor y así solicitar que realice distintas tareas. Como se puede observar en la *Ilustración 16*, se puede comprobar que la conexión funciona correctamente.
Las distintas opciones con las que cuenta el programa son:

- 0: Desconectar el sensor.
- 1: Realizar un test de la conexión.
- 2: Obtener el identificador (ID) del sensor.
- 3: Obtener el estado del sensor. Es 0 si el sensor no está tomando datos y 1 si se encuentra en funcionamiento.
- 4: Pulsar el botón para comenzar la toma de datos o en caso de estar tomándolos, parar.
- 5: Obtener la hora y la fecha que tiene el sensor en ese momento.

- 6: Poner en hora y fecha el sensor, según la hora del ordenador.

```
Menu:
        0 - Disconnect
        1 - Ping
        2 - Sensor ID
        3 - Get Status
        4 - Pulse button
        5 - Get time/date
        6 - Set time/date
Option?5
Sensor Time:  0:14: 6    1- 1-2000 Monday

Menu:
        0 - Disconnect
        1 - Ping
        2 - Sensor ID
        3 - Get Status
        4 - Pulse button
        5 - Get time/date
        6 - Set time/date
Option?6
PC Time: 11:17:51 8-9-2022 Thursday
Sensor Time: 11:17:51    8- 9-2022 Thursday
```

*Ilustración 17. Cómo poner en hora y fecha el sensor mediante la interfaz USB*

Cada vez que el sensor se queda sin batería sufre una pérdida de la memoria, por lo que es muy importante comprobar el estado del sensor y ponerlo en hora y fecha antes de realizar cualquier tipo de prueba.

## *Software embebido en el sensor*

A diferencia del software de control desde un PC, el software embebido en la placa tiene una alta complejidad, tanto a nivel de líneas de código como de paralelismo interno. Por ello se ha desarrollado un esquema que muestra el flujo del programa. El código desarrollado se incluye principalmente en 3 ficheros: *main* (830 líneas de código C), *data-logger* (749 líneas) e interfaz USB (298 líneas).

El código embebido del módulo tiene dos funciones principales: `GetData_Thread(void const *argument)` y `WriteData_Thread(void const *argument)`. Como ambas funciones implementan hilos o *threads*, incluyen un bucle for infinito y varios semáforos. Además, se utilizan cuatro interrupciones: `HAL_GPIO_EXTI_Callback`, `dataTimer_Callback`, `BSP_AUDIO_IN_HalfTransfer_CallBack` y `BSP_AUDIO_IN_TransferComplete_CallBack`. Dos interrupciones (`HAL_GPIO_EXTI_Callback` y `dataTimer_Callback`) marcan la entrada en la función `GetData_Thread`, al activar uno de los ID del semáforo. Las otras dos (`BSP_AUDIO_IN_HalfTransfer_CallBack` y `BSP_AUDIO_IN_TransferComplete_CallBack`) comprueban si el almacenamiento del audio en la tarjeta SD está activo y llaman a la función `AudioProcess_SD_Recording()` para almacenar dichos datos.

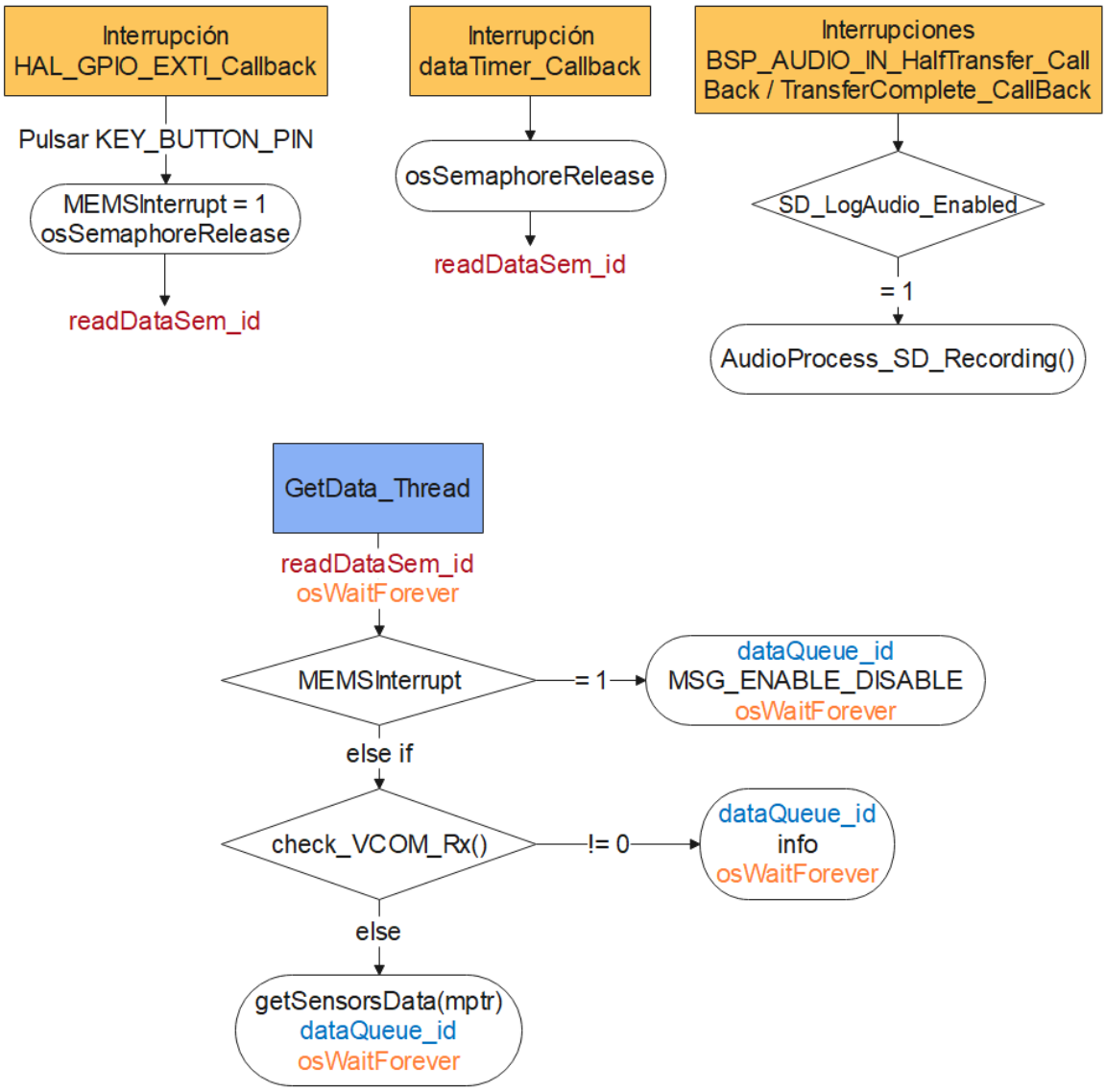


*Ilustración 18. Interrupciones y función GetData_Thread*

Una vez en la función **`GetData_Thread`** se llevan a cabo los siguientes pasos:

1. Se comprueba si `MEMSInterrupt` se encuentra a 1. Si es así se comprueba si `SD_Log_Enabled = 1`, ya que esto significa que se están almacenando datos en la SD. En caso afirmativo, se detiene la toma de datos porque que esto implica que el botón se ha pulsado. Si `SD_Log_Enabled = 0` , se siguen almacenando datos en la SD y se lanzan los mensajes `dataQueue_id` y `MSG_ENABLE_DISABLE`.
2. Se comprueba si `check_VCOM_Rx()!=0`. En caso de que se cumpla esta condición eso significa que estamos recibiendo mensajes a través del USB. A continuación, se lanzan los mensajes `dataQueue_id` e `info`.
3. Finalmente se leen los datos. En este caso, se intenta asignar un bloque de memoria y se verifica que el puntero del bloque de memoria no sea NULL. Después se lanzan los mensajes `dataQueue_id` y `mptr`.
4. Una vez comprobados todos los pasos se vuelve a empezar el bucle for y se realizan todos los pasos nuevamente.

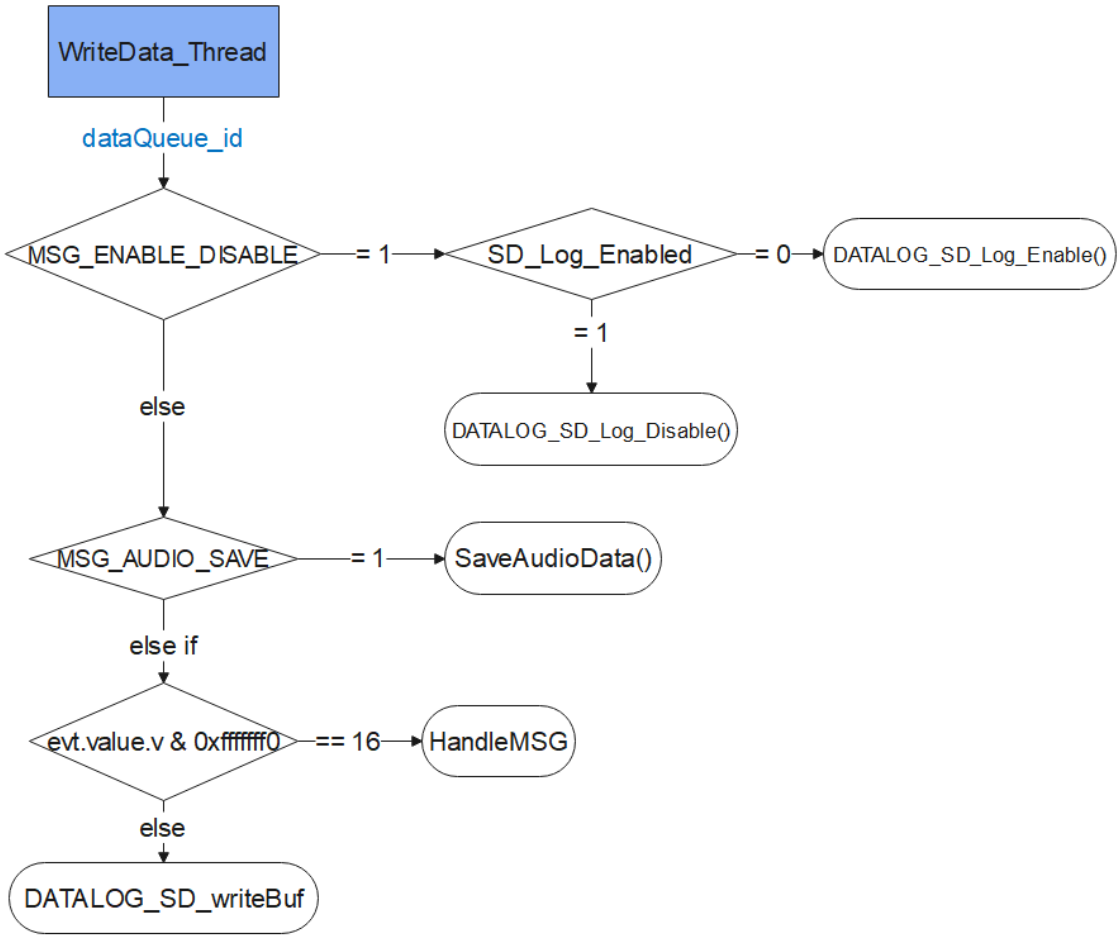


*Ilustración 19. Función WriteData_Thread*

En la función **`WriteData_Thread`** se llevan a cabo los siguientes pasos:

1. Se espera al mensaje que indica el evento, en este caso `dataQueue_id`.
2. Se comprueba si el evento es `MSG_ENABLE_DISABLE`. En caso afirmativo, se comprueba si `SD_Log_Enabled = 1`, lo que significa que se ha pulsado el botón para interrumpir la toma de datos y, por lo tanto, se llama a `DATALOG_SD_Log_Disable()`, se cierran los ficheros, se imprime el resumen y se resetean los contadores del sensor. En caso contrario, se llama a `DATALOG_SD_Log_Enable()`, ya que esto implica que se ha pulsado el botón para que el sensor tome datos y los almacene.
3. Se comprueba si el evento es `MSG_AUDIO_SAVE`. En caso afirmativo se llama a **`SaveAudioData()`**, lo que implica guardar el audio. Este evento solo existirá si ha habido una de las interrupciones de audio donde se llama a la función **`AudioProcess_SD_Recording()`.**
4. Se comprueba si el evento más una máscara es igual a 16. Esto implica que hay un mensaje que se ha recibido a través del puerto USB. En este caso se llama a **`HandleMSG()`** para procesar los mensajes recibidos y realizar las acciones que haya escogido el usuario según el menú de la *Ilustración 16*.
5. Finalmente, si no es ninguno de los casos anteriores, se comprueba el puntero de memoria y se almacenan los datos obtenidos de los distintos sensores mediante la llama a la función **`DATALOG_SD_writeBuf`**, que escribirá los resultados en un fichero de texto dentro de la tarjeta SD.

Una de las mejoras incorporadas a la última versión del software es la creación de un fichero de *logs*, donde se indican los posibles errores que surjan durante el funcionamiento del sensor, como apoyo al código de colores de los Leds que posee el módulo. Más tarde, también se decide que en ese mismo fichero de *logs* se incluya un mensaje donde se indique el momento exacto (hora, fecha y milisegundos de funcionamiento del módulo) en el que se pulsa el botón de encendido/apagado de la toma de datos y se crean nuevos ficheros. Esto permite tener una mejor sincronización con los videos tomados y, en caso de fallo del módulo, saber el momento exacto en el que sucede.

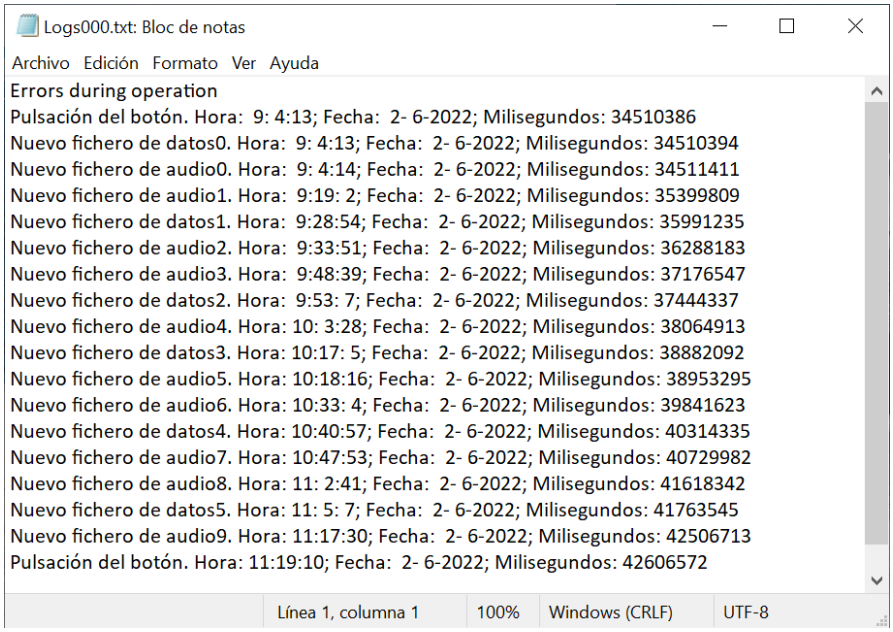


*Ilustración 20. Fichero de logs del módulo*

Los errores que pueden surgir durante el funcionamiento del sensor se reportan en el fichero de logs con tres tipos de códigos numéricos. Cada tipo identifica la parte del código a la que pertenece el evento: errores a nivel de código de usuario, errores de inicio del hardware (BSP) y errores relacionados con el reloj del sistema (RTC). A continuación, se incluye una tabla con el código que posee cada tipo de error:

| **ERRORES** | | |
|---|---|---|
| **STBOX** | **BSP (*Board support package*)** | **RTC (*Real Time Clock*)** |
| STBOX1_NO_ERROR = 0 | BSP_ERROR_NONE = 0 | HAL_OK = 0x00 |
| STBOX1_INIT_ERROR = 1 | BSP_ERROR_NO_INIT = -1 | HAL_ERROR = 0x01 |
| STBOX1_MEMS_ERROR = 2 | BSP_ERROR_WRONG_PARAM = -2 | HAL_BUSY = 0x02 |
| STBOX1_AUDIO_ERROR = 3 | BSP_ERROR_BUSY = -3 | HAL_TIMEOUT = 0x03 |
| STBOX1_FATFS = 4 | BSP_ERROR_PERIPH_FAILURE = -4 | |
| STBOX1_OS = 5 | BSP_ERROR_COMPONENT_FAILURE = -5 | |
| STBOX1_CLOCK = 6 | BSP_ERROR_UNKNOWN_FAILURE = -6 | |
| STBOX1_USBD = 7 | BSP_ERROR_UNKNOWN_COMPONENT = -7 | |
| STBOX1_VCOM = 8 | BSP_ERROR_BUS_FAILURE = -8 | |
| STBOX1_AUD = 9 | BSP_ERROR_CLOCK_FAILURE = -9 | |
| STBOX1_AUD_SAVE = 10 | BSP_ERROR_MSP_FAILURE = -10 | |
| | BSP_NOT_IMPLEMENTED = -11 | |

*Tabla 1. Tipos de errores y su codificación*

Los errores de tipo STBOX son aquellos que hacen referencia al argumento de la función **`ErrorHanlder()`**. Se ha creado la variable *errno* para poder pasar de manera cómoda el código del error al fichero de *logs*.

Los errores de tipo BSP y RTC son aquellos con los que se compara el código para mandar a la función **`ErrorHanlder()`**.

```
/* Stop Audio Recording */
if((errno = BSP_AUDIO_IN_Stop(BSP_AUDIO_IN_INSTANCE)) != BSP_ERROR_NONE) {
    ErrorHanlder(STBOX1_AUDIO_ERROR);
}
```

*Ilustración 21. Ejemplo de uso del código de errores*

```
if ((errno = HAL_RTC_Init(&RtcHandle)) != HAL_OK)
{
  // Initialization Error
    ErrorHanlder(STBOX1_CLOCK);
}
```

*Ilustración 22. Ejemplo de uso del código de errores*

Otra de las mejoras es la modificación de la duración temporal de los ficheros, ya que cuanto más largos sean estos, más difícil será el proceso de lectura y preprocesado. Por lo tanto, se toma la decisión de establecer los siguientes parámetros:

- En 1 hora se generan cuatro ficheros de audio que almacenan 15 minutos de sonido. Además, se guardan dos ficheros de datos de 155000 líneas aproximadamente y otro de 69211 líneas.
- Las variables encargadas de la sincronización, cierre y apertura de ficheros son `FORCE_WRITE` y `NEW_FILE`.

A continuación, se muestra un ejemplo del código que gestiona la apertura y cierre de ficheros de datos:

```
if(counter_aud >= FORCE_WRITE){ //Sincroniza cada 1min
    if(counter_aud2 >= NEW_FILE){ //Cierra y abre un nuevo fichero
        f_sync(&MyFileAudio);
        closeFileAudio();
        //DATALOG_SD_LogAudio_Disable();
        DATALOG_SD_LogAudio_Enable();
        counter_aud=0;
        counter_aud2=0;
    } else{
        f_sync(&MyFileAudio);
        counter_aud = counter_aud - FORCE_WRITE;
        counter_aud2++;
    }
  } else   {
   counter_aud += BytesWritten/2;
  }
```

*Ilustración 23. Ejemplo del funcionamiento del cierre y apertura de ficheros de audio*

# Capítulo 5. Calibración y toma de datos con el sensor

Antes de empezar a medir datos en campo, es necesario verificar que el sensor funciona correctamente. Para ello se ha realizado un proceso de calibración cuyo objetivo es determinar que los datos que se obtienen se corresponden con la realidad. En este proceso de calibración se han utilizado 2 sensores:

- Un sensor comercial, proporcionado por CIFA.
- El sensor descrito en los capítulos anteriores.

## 5.1. Calibración en el laboratorio

Al inicio del proyecto se nos proporciona un dispositivo desarrollado por *Digitanimal* (*SensoWave* [15]), el cual es una modificación de su producto más vendido: el "Localizador GPS para Ganado". Este dispositivo determina la posición GPS y temperatura del animal, y lo transmite mediante la red GSM o *Sigfox* al ganadero. Sin embargo, el módulo empleado en experimento almacena los datos del acelerómetro en una memoria interna del módulo (SD Card).


*Ilustración 24. Módulo de Digitanimal*

En un primer momento, el módulo de *Digitanimal* se empleó para comprobar el correcto funcionamiento de nuestro propio módulo. Para ello, se desarrolló un banco de pruebas que permitía medir el comportamiento de los sensores con movimientos predefinidos gracias a un generador de funciones y un servomotor. El sistema de medida genera movimientos angulares en un eje, con aceleración sinusoidal y un ángulo de 45 grados con la gravedad, lo que permite generar el mismo estímulo a ambos sensores, esperando que proporcionen los mismos datos de salida.

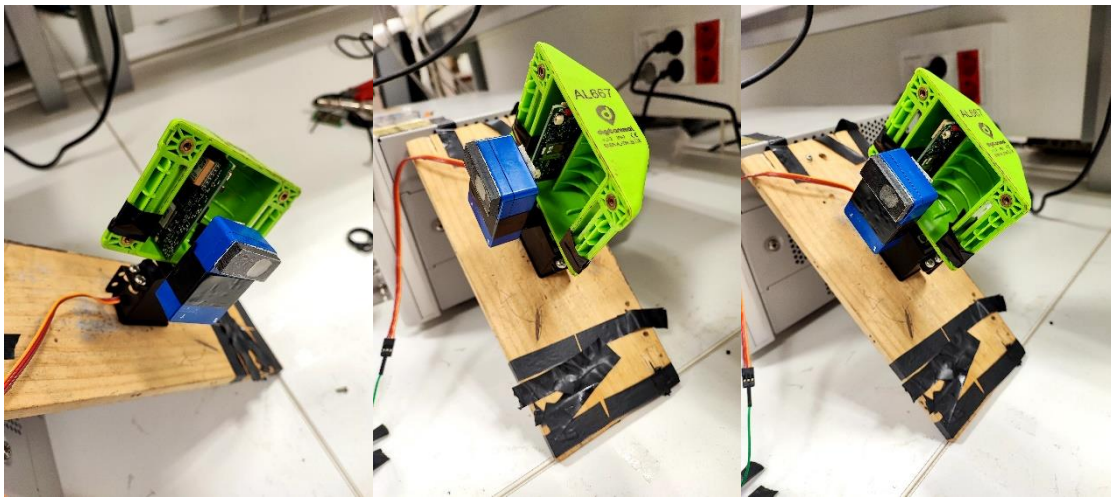

*Ilustración 25. Banco de pruebas con ambos sensores*

Cuando se realizaban pruebas cortas (menos de 1 hora), los resultados obtenidos eran los esperados. Pero si se realizaban pruebas largas, a partir de las 5 horas de grabación, el módulo de *Digitanimal* comenzaba a mostrar comportamientos extraños. Observamos que, en un primer momento, la frecuencia de muestreo aumentaba y después disminuía considerablemente.

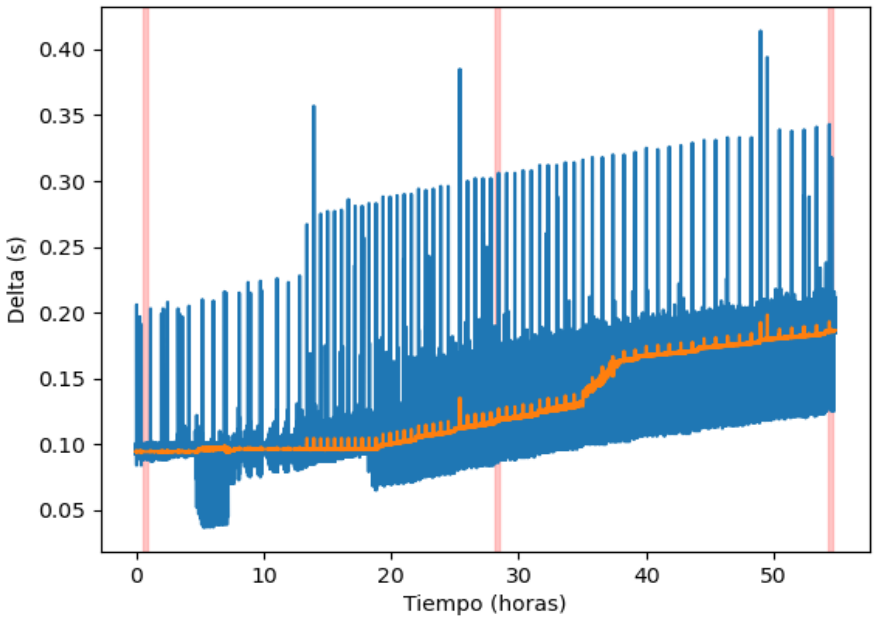


*Ilustración 26. Gráfica de la progresión temporal del funcionamiento del sensor de Digitanimal*

Como se observa en la gráfica anterior, la frecuencia de muestreo (línea azul) no es constante con el tiempo, variando su media de forma apreciable (línea naranja). Después de varias pruebas, se llega a la conclusión de que el error se debía a un fallo en el *firmware* del sensor comercial. Una vez comunicado el problema a *Digitanimal*, la empresa proporcionó un nuevo sensor con el que se volvieron a hacer pruebas para poder continuar con la calibración de nuestro propio sensor. Dicho sensor ya no tenía un comportamiento anómalo.

A continuación, se muestran los resultados obtenidos. En primer lugar, se observa la gráfica del eje Y del sensor de *Digitanimal* (*Ilustración 27*) y después la del eje Y del sensor que hemos desarrollado en el proyecto (*Ilustración 28*):

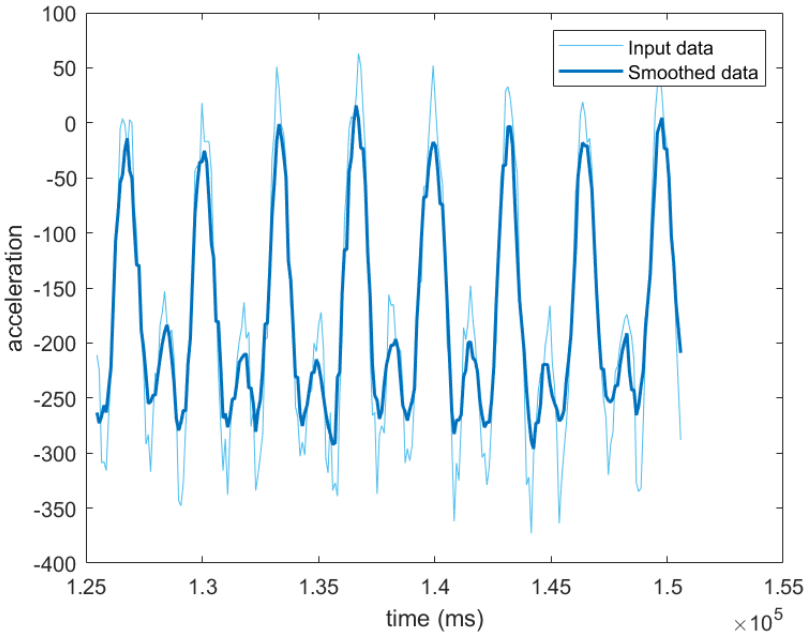


*Ilustración 27. Gráfica del funcionamiento del sensor de Digitanimal*

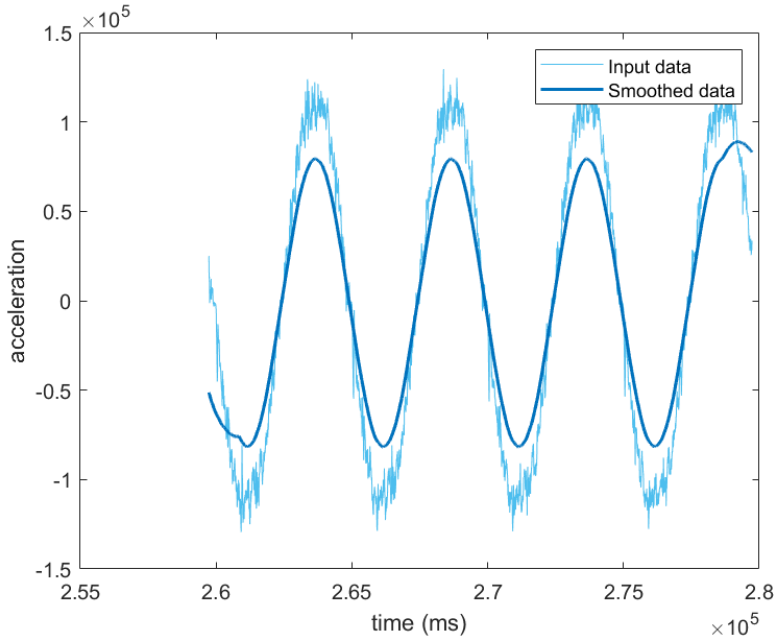


*Ilustración 28. Gráfica del funcionamiento del sensor del proyecto*

Se ha realizado un *suavizado* con un promedio móvil en ambos casos para poder apreciar mejor los datos, ya que debido a la sensibilidad de los sensores y el movimiento que produce el servomotor los datos obtenidos (azul claro) tienen bastante ruido. Se puede observar que las curvas suavizadas (azul oscuro) reflejan la forma sinusoidal de la aceleración. En el caso del sensor de *Digitanimal*, la forma es diferente porque cambia la

posición del vector aceleración de forma diferente al sensor del proyecto (distinta posición en el montaje).

Estas medidas permitieron llegar a la conclusión de que las muestras que generan ambos sensores son bastante precisas. Hay que tener en cuenta que cada sensor realiza la medida de una forma distinta, además de tener distinta frecuencia de muestreo. En este ejemplo, la frecuencia de muestreo de nuestro sensor es más alta (100Hz frente a 10Hz del sensor comercial), razón por la cual la forma de la gráfica es más precisa, sin tantos saltos.

## 5.2. Colocación de los collares en los animales

Una vez validado el sensor en el laboratorio, se pasó a realizar medidas en campo, colocando el sensor a distintas cabras y ovejas. En sensor fue colocado en el cuello del animal, utilizando un collar de nylon. Tras realizar varias pruebas en campo, surge la duda sobre cuál es la mejor forma de colocar el collar en los animales, ya que este puede tener un peso (pluma) para que el collar se mantenga siempre en la misma posición o, por el contrario, puede usarse sin el peso.

Que el collar esté colocado con peso o sin él puede influir en el rango de aceleraciones que el collar soporta y, por lo tanto, en la obtención de datos. Por esta razón se lleva a cabo una prueba en el laboratorio con dos sensores colocados mediante los collares en los brazos, uno con peso en el brazo izquierdo y otro sin peso en el brazo derecho.

Hay que tener en cuenta que la posición de los ejes en este experimento es la que aparece en la *Ilustración 29*, aunque en el caso de este proyecto la orientación de los ejes no es relevante a la hora de colocar los sensores en los collares, solo se ha tenido en cuenta la posición del micrófono.

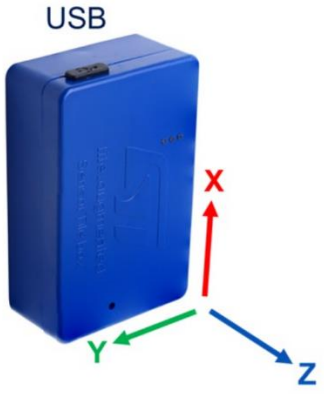


*Ilustración 29. Posición de los ejes en el sensor*

Los resultados obtenidos tras el experimento son los siguientes:

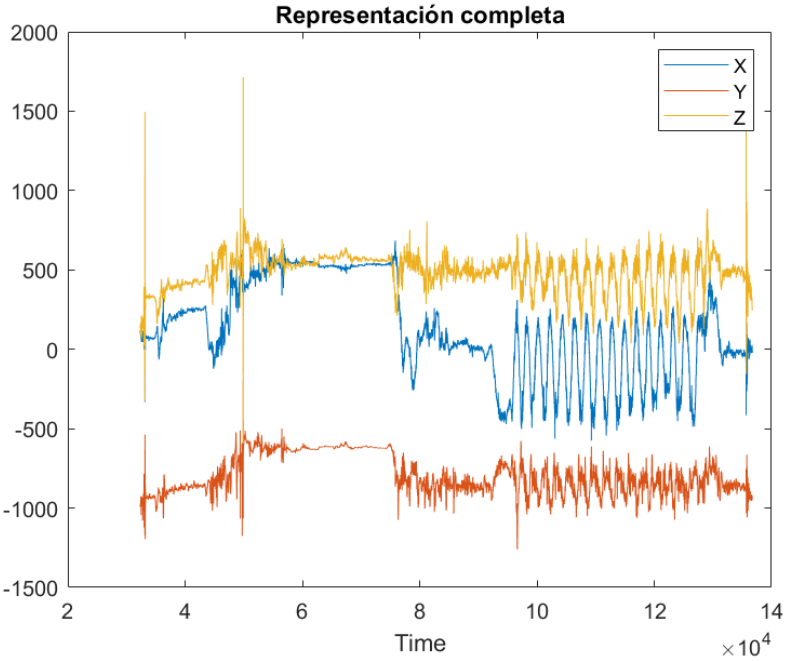


*Ilustración 30. Representación de la aceleración en un collar con peso*

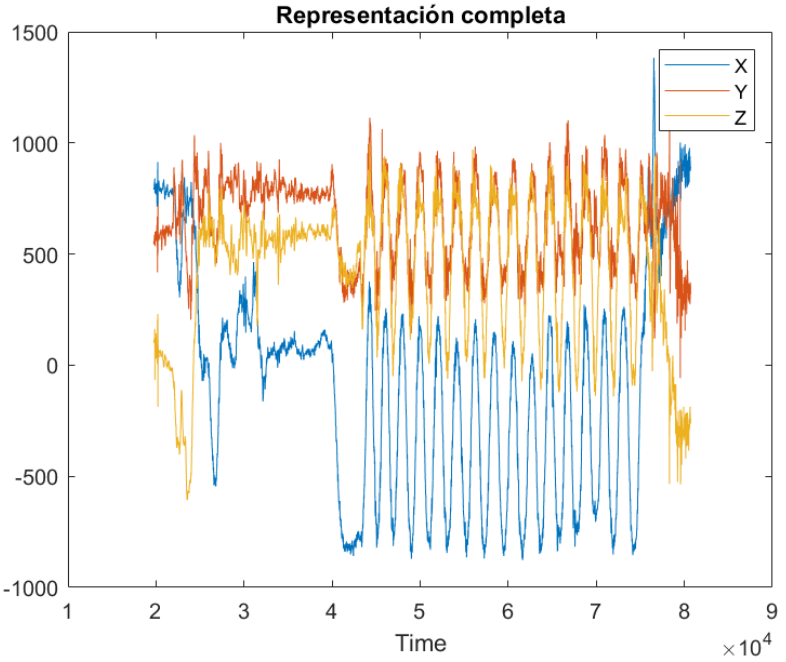


*Ilustración 31. Representación de la aceleración en un collar sin peso*

Como se puede observar, habiendo realizado el mismo movimiento con los dos brazos (movimientos verticales hacia arriba y hacia abajo), se observa claramente que el rango de variación de la aceleración del collar sin peso (*Ilustración 31*) es mucho mayor que el rango de movimiento del collar con peso (*Ilustración 30*), debido al efecto amortiguador de dicha masa. Por lo tanto, se llega a la conclusión de que la mejor forma de colocar el collar en el animal es sin el peso. También hay que tener en cuenta que la caja que sirve como aislamiento acústico, por mucho que tenga un peso reducido, también cuenta como peso a la hora de colocar el collar en el animal. Aun así, algunas pruebas en campo se han

realizado con peso en uno de los collares, para observar si siguen existiendo diferencias significativas en los resultados.

## 5.3. Captura de datos en campo

Para llevar a cabo la toma de datos en campo, se ha empleado una metodología definida por CIFA para sensores comerciales, la cual incluye 4 etapas:

1. *Sincronización de la cámara que monitoriza la actividad del animal y los sensores*.
   Para poder asociar los datos capturados con los sensores a actividades del animal es preciso disponer de un registro de dichas actividades. Este registro se obtiene mediante una grabación en video del comportamiento del animal. La cámara y los sensores se sincronizan utilizando para ello un reloj común. En este proyecto, se ha utilizado el reloj del ordenador empleado para programar los sensores. En esta etapa se han encontrado varios problemas a la hora de sincronizar las cámaras, ya que cada modelo muestra la marca temporal de diferente forma e incluso, en alguna ocasión, no se ha grabado de forma adecuada la fecha en los vídeos, lo que ha causado problemas a la hora de realizar el tratamiento posterior de los datos.

2. *Colocación de sensores en los animales y captura de datos*.
   Esta etapa se ha llevado a cabo en colaboración con CIFA, en la finca de La Jerrizuela del Gobierno de Cantabria. En dicha finca, CIFA trabaja con un rebaño formado por cabras y ovejas. Con la ayuda de los pastores y los investigadores de CIFA, se han colocado los sensores en varios animales y se ha grabado su actividad durante un periodo de tiempo superior a 15 minutos.

*Ilustración 32. Toma de datos de la actividad de una cabra en la Finca de La Jerrizuela*

3. *Preparación de los datos*
   En esta etapa se extraen los datos del sensor y se preprocesan. El sensor almacena los datos en distintos ficheros según la duración de estos y el tipo de dato que esté almacenando. En esta fase se unen los ficheros y se sincronizan con el vídeo, obteniendo de esta forma dos ficheros de muestras: el fichero con todos los datos obtenidos y la secuencia de datos que se corresponde con el vídeo grabado. Este último fichero es el que se empleará para realizar la anotación de los datos.

Para facilitar la lectura de los datos, una vez que han sido extraídos de la tarjeta SD, se ha generado un pequeño código de MATLAB donde se puede ver la representación completa de los datos del acelerómetro, magnetómetro y giroscopio del fichero que se quiera observar, así como la representación de cada eje por separado.

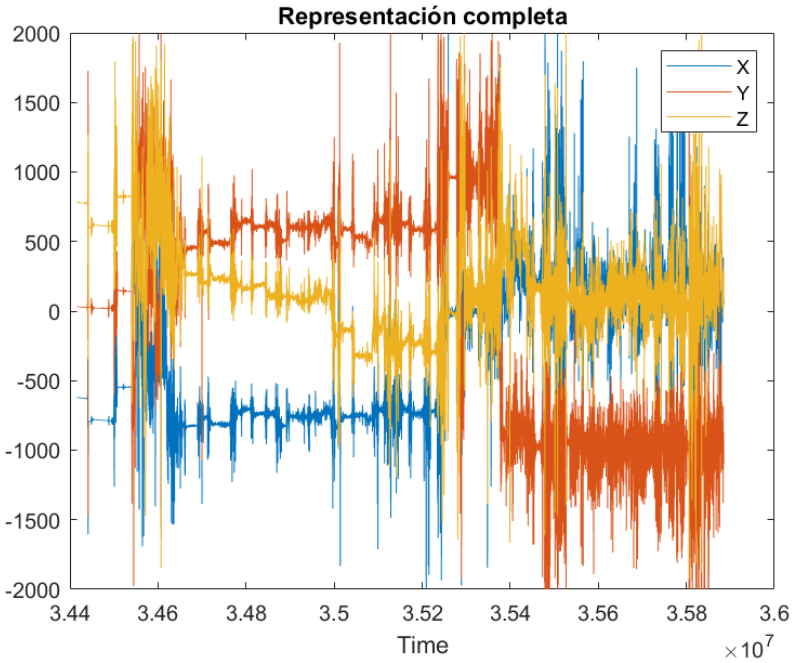


*Ilustración 33. Gráfica de la representación completa y de los tres ejes del acelerómetro*

En la figura adjunta, la gráfica muestra la evolución de los datos del acelerómetro de uno de los ficheros generado por los sensores. Estas gráficas son las que más adelante se sincronizarán con los videos y se anotarán con las diferentes actividades que estén realizando los animales en ese momento.

4. *Anotación de los datos*

Utilizando el programa BORIS (*Behavioral Observation Research Interactive Software*) [16], la secuencia de datos sincronizada con el video, obtenida en la fase anterior, se anota con las diferentes actividades que tienen lugar en el video. El resultado es un fichero de anotaciones de las actividades que se utiliza para entrenar los modelos de la inteligencia artificial.

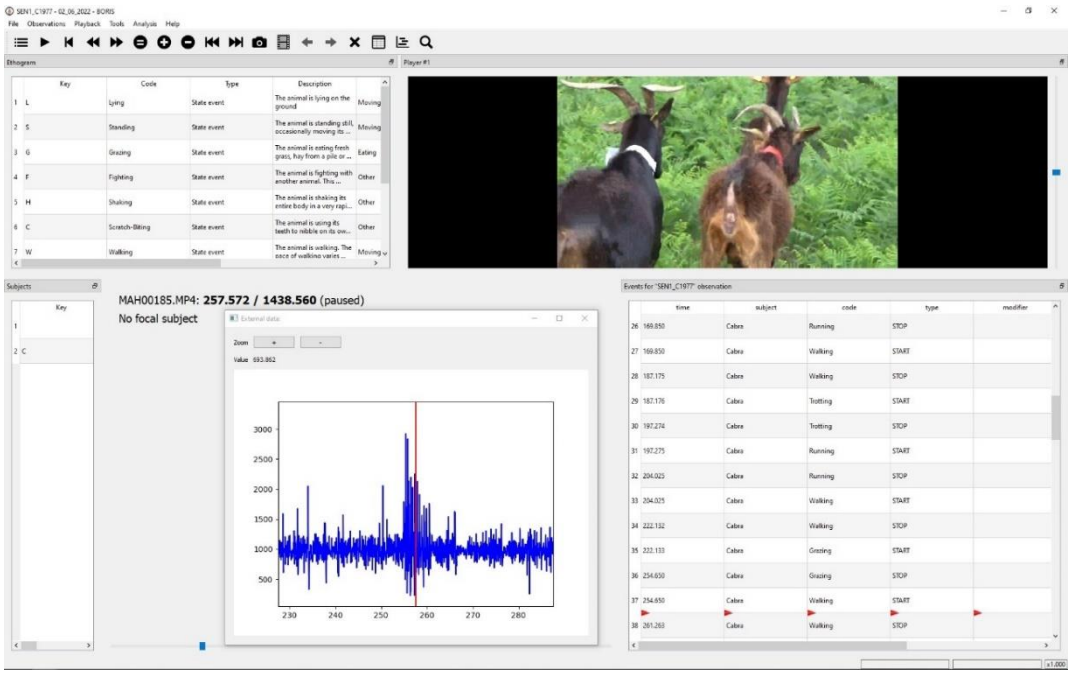


*Ilustración 34. Anotación de actividad con BORIS*

Los datos obtenidos muestran patrones similares a los de algunos conjuntos de datos públicos. La configuración empleada en el sensor ha sido la siguiente:

- Frecuencia de muestreo del acelerómetro, giroscopio y magnetómetro: 100Hz
- Rango del acelerómetro: ± 2g
- Rango del giroscopio: ± 250 dps
- Rango del magnetómetro: ± 50 gauss
- Frecuencia de muestreo del micrófono: 16 kHz
- Número de bits de las muestras del micrófono: 16 bits, sin signo

A continuación, se incluyen algunos ejemplos de las etiquetas:

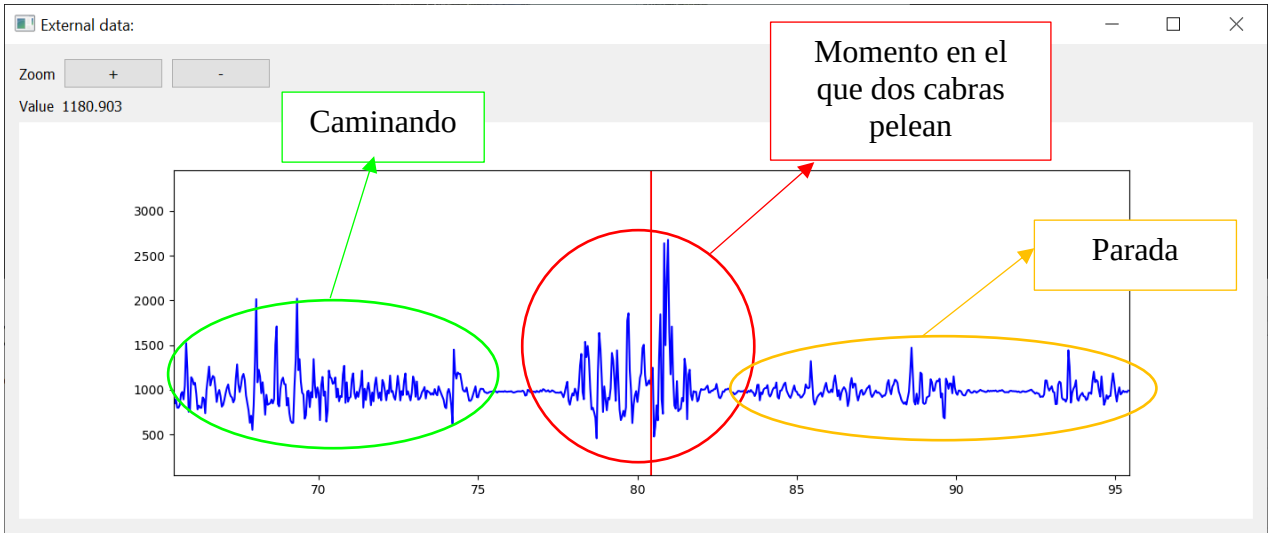


*Ilustración 35. Gráfica de la aceleración del movimiento de una cabra*

Como se puede observar en la *Ilustración 35* y en las anotaciones de la *Ilustración 36*, se aprecian distintas actividades realizadas por los animales. El cursor rojo indica en ambas ilustraciones la marca temporal.

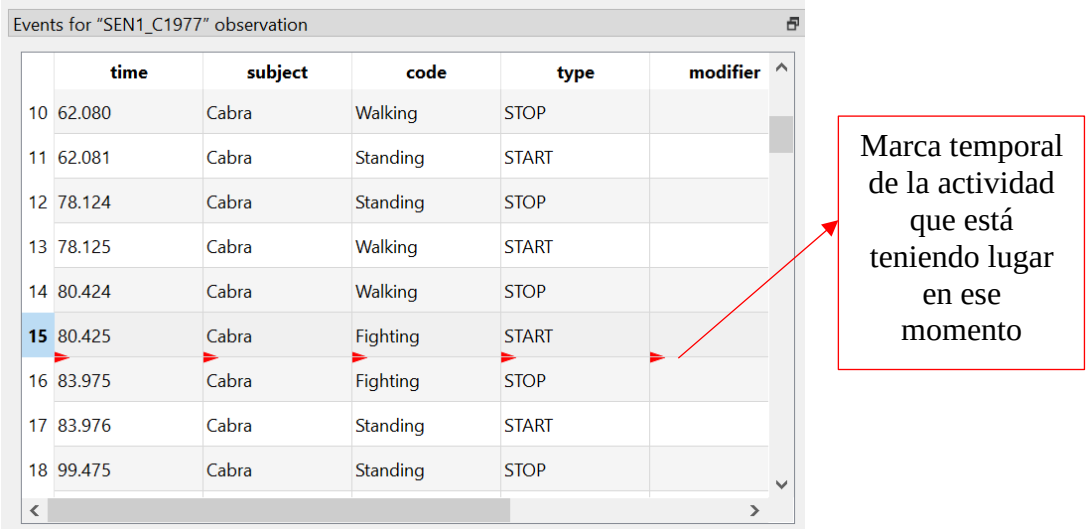


*Ilustración 36. Anotaciones de las distintas actividades que realiza el animal*

Los videos tomados durante las pruebas de los collares también son de gran ayuda a la hora de realizar las anotaciones, ya que, si el vídeo y los datos están bien sincronizados, permiten saber con mayor precisión la actividad de la que se trata a la hora de realizar las anotaciones.


*Ilustración 37. Video de apoyo durante la anotación de actividades*

# Capítulo 6. Módulo de Inteligencia Artificial integrado en el sensor

El dispositivo LSM6DSOX de STMicroelectronics incorpora un módulo hardware orientado a *machine learning*, el *Machine Learning Core* (MLC), que facilita la implementación de sistemas inteligentes. El módulo admite un conjunto de parámetros de configuración e implementa árboles de decisión capaces de implementar algoritmos de inteligencia artificial en el propio sensor [17]. Estos algoritmos permiten identificar si un patrón de datos coincide con un conjunto de clases definido por el usuario. Ejemplos típicos de aplicaciones de este componente son la detección de actividades humanas como correr, caminar, conducir, etc.

El MLC funciona con patrones de datos provenientes de los acelerómetros y giroscopios, pero también es posible utilizar datos de sensores externos, como un magnetómetro. Los datos de entrada se pueden preprocesar utilizando un bloque de que implementa filtros y el cálculo de parámetros estadísticos en una ventana de tiempo definida por el usuario.

El MLC utiliza una serie de nodos configurables, caracterizados por condiciones "*if-then-else*", donde los valores de ciertos parámetros, denominados "*features*", se evalúan frente a umbrales definidos.

El LSM6DSOX se puede configurar para utilizar hasta 8 árboles de decisión de forma simultánea e independiente, pudiendo cada flujo generar hasta 16 resultados. El número total de nodos puede llegar hasta 256 [18], [19].

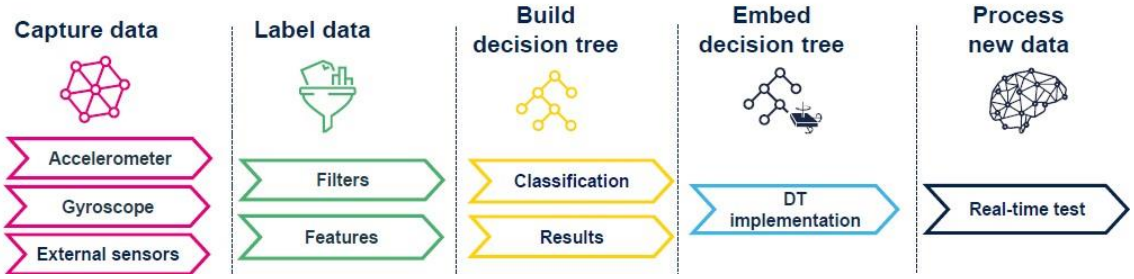


*Ilustración 38. Configuración del MLC*

## 6.1. Configuración del módulo de IA

En nuestro caso, el módulo recibe los datos de los sensores internos (acelerómetros y giroscopios) y de los sensores externos (magnetómetro) y los preprocesa. El primer paso del preprocesado de datos es el filtrado, utilizando para ello varios tipos de filtros. A continuación, se añaden a los datos filtrados 12 tipos de parámetros o *features*: media, varianza, energía, valor pico-pico, paso por cero, paso por cero positivo, paso por cero negativo, detector de pico, valor de pico positivo, pico negativo, máximo y mínimo. El módulo permite seleccionar las *features* y datos que se emplearán en el árbol de decisión para clasificar un determinado comportamiento.

Gracias a que el componente puede implementar hasta 8 árboles de decisión y permite combinar sus resultados utilizando 8 máquinas de estados, es posible realizar predicciones

precisas en tiempo real con un reducido consumo de energía. El resultado del módulo es una lista de eventos en la cual se indica el instante temporal en el que se ha detectado un determinado comportamiento.

Para configurar el módulo, el fabricante facilita un entorno denominado “UNICO-GUI”, el cual permite capturar los datos, configurar el componente, entrenar el modelo y generar el firmware necesario para su uso [20], [21].

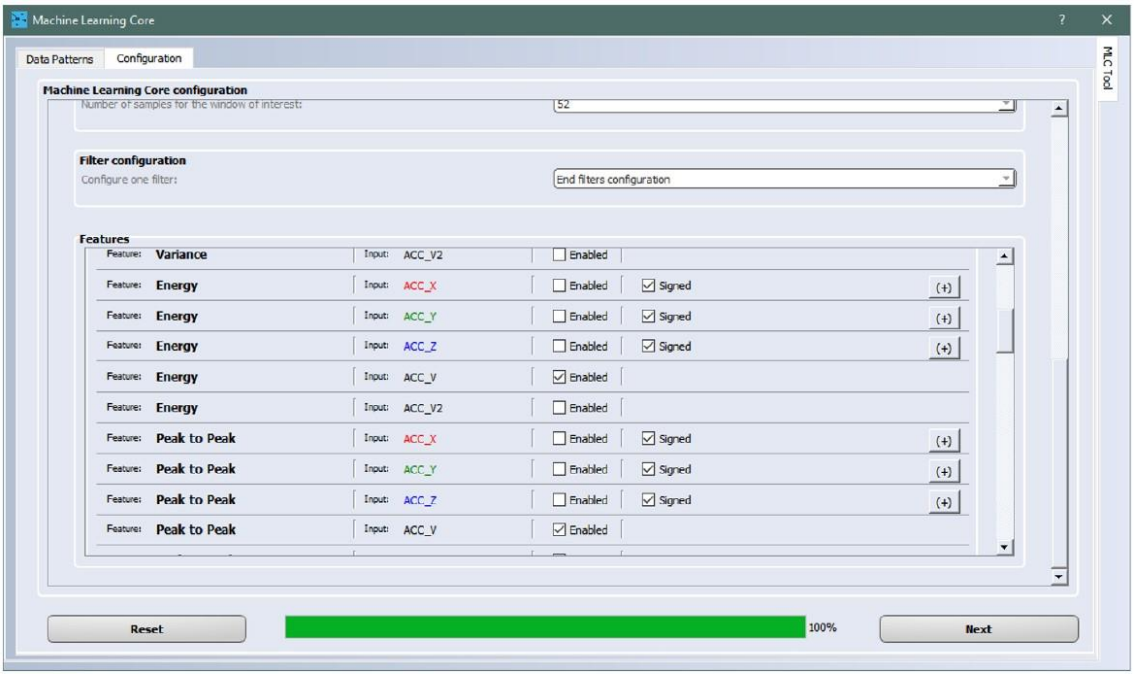


*Ilustración 39. Configuración del módulo de inteligencia artificial*

El módulo ha sido entrenado con el conjunto de datos públicos disponibles. Se ha empleado un único árbol de decisión con una ventana de 1 segundo. Se han utilizado como entradas al árbol de decisión los valores medios de los ejes de los sensores (acelerómetro, giroscopio y magnetómetro), el módulo y la energía de cada sensor. La herramienta genera un árbol de decisión con 43 ramas. La precisión es del 89.6%, aunque el sistema no señala cómo ha obtenido dicho parámetro (por ejemplo, la relación entre el conjunto de patrones de entrenamiento y validación).

# Capítulo 7. Procesado de datos y entrenamiento

## 7.1. Descripción del conjunto de datos

En este trabajo se han empleado los datos capturados por los diferentes sensores integrados en el módulo *SensorTile.box*: acelerómetro, giroscopio y magnetómetro. El módulo ha sido colocado en el cuello de los animales de un rebaño formado por cabras y ovejas, gestionado por CIFA.

Durante el desarrollo y la mejora del software del módulo se han tomado muchas muestras de datos de los distintos animales del rebaño, hasta llegar a una versión con un funcionamiento óptimo de los sensores. Es por esta razón, por la que, para realizar las grabaciones de los animales, así como su posterior etiquetado, solo se han tenido en cuenta algunas de las grabaciones realizadas. A continuación, se muestra una tabla con los datos recogidos a lo largo de las pruebas con animales, y cuáles de ellos se han usado para el etiquetado:

| Tipo | Crotal | Grabaciones de video | Etiquetado |
|---|---|---|---|
| Cabra | 1977 | 10 | Si |
| Cabra | 2368 | 2 | Si |
| Cabra | 4236 | 8 | No |
| Oveja | 8976 | 3 | Si |
| Oveja | 8040 | 1 | No |
| Oveja | 2152 | 2 | No |
| Oveja | 8903 | 3 | No |

*Tabla 2. Resumen del número de grabaciones por tipo de animal y el crotal de identificación*

### *Datos de vídeos etiquetados*

Durante el tiempo de captura de los datos, el animal fue grabado durante periodos de 20 minutos aproximadamente. En algunos casos, varias grabaciones breves forman una de 20 minutos. Posteriormente se sincronizaron los sensores y videos, ya que el sensor solo almacena información del instante en el que se encendió y el momento de creación de cada fichero de datos. La sincronización de los datos fue realizada empleando el software BORIS. Una vez que se han sincronizado los datos de los distintos sensores con el vídeo, se procede a etiquetar las distintas partes del video. En nuestro caso, solo se ha utilizado un tipo de evento, **el estado,** que indica situaciones que se prolongan durante un periodo de tiempo indefinido.

En total, se han catalogado diez tipos de actividades:

- **Tumbado**: El animal está tumbado en el suelo. Categoría: Movimiento.

- **Parado**: El animal está parado, moviendo ocasionalmente la cabeza o dando pasos muy lentos.
- **Pastando**: El animal está comiendo hierba fresca, heno de un montón o ramitas en el suelo.
- **Peleando**: El animal está peleando con otro animal. Consiste en golpear su cabeza contra la cabeza o el cuerpo de otro animal. Una cabra a menudo se para sobre la parte posterior de sus patas, se deja caer al suelo y clava sus cuernos en otro animal.
- **Sacudiendo**: El animal está sacudiendo todo su cuerpo con un movimiento muy rápido, a menudo seguido de un movimiento rápido de la cabeza durante un breve momento. En contadas ocasiones, el animal se limita a mover la cabeza.
- **Rascarse**: El animal está usando sus dientes para mordisquear su propia pie. A veces, el animal usa sus pezuñas.
- **Caminando**: El animal está caminando. El ritmo de caminar varía desde muy lento hasta casi el trote.
- **Trotando**: Esta es la fase entre caminar y correr. El animal no está galopando, pero está andando muy rápido por lo que se encuentra en estado de trote.
- **Corriendo**: El animal está galopando.

Una vez anotados los datos, se descargan en un fichero que incluye no solo las muestras sino también las marcas de anotación.

## 7.2. Entrenamiento

En este proyecto se emplea *Scikit-learn* (o *Sklearn*), una biblioteca de aprendizaje automático de software libre para el lenguaje de programación Python [22]. El entorno incluye varios algoritmos de aprendizaje supervisado y no supervisado.

La librería está construida sobre *SciPy (Scientific Python)* y está diseñada para trabajar con las bibliotecas numéricas y científicas *NumPy*, *SciPy*, *Pandas*, *Matplotlib*, *Ipython* y *SymPy* [23].

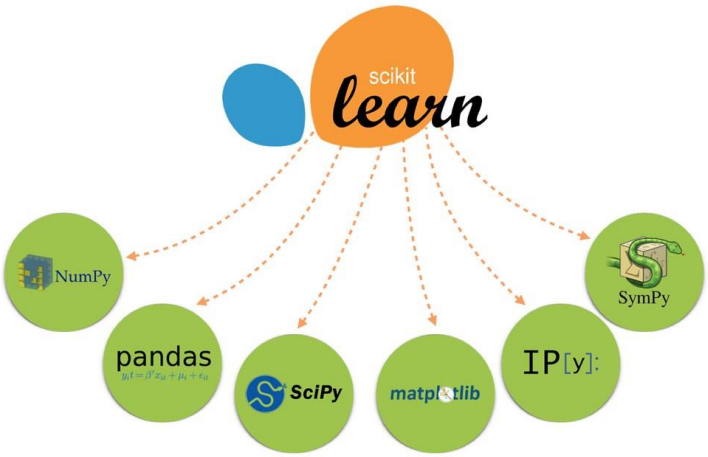


*Ilustración 40. Librerías de Scikit-learn*

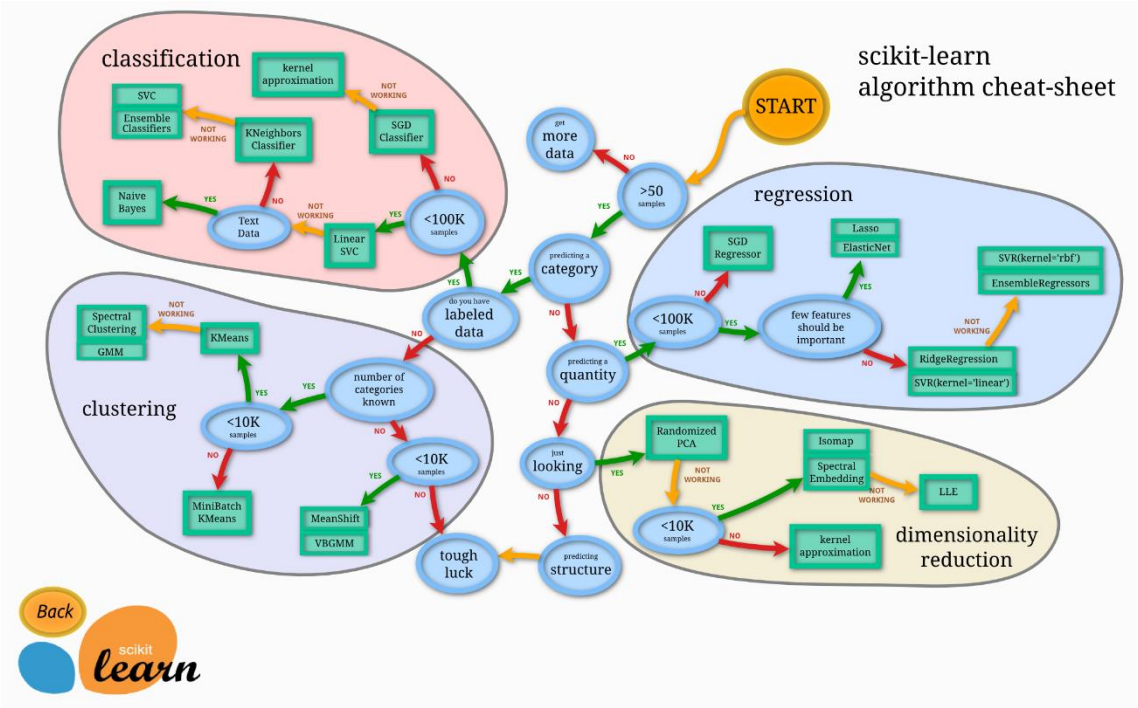


*Ilustración 41. Algoritmos de Scikit-learn*

Se han llevado a cabo los siguientes pasos para realizar la parte del entrenamiento de la IA [12]:

1. En primer lugar, se importan las bibliotecas necesarias:

```
from library import readSensorData
import numpy as np
import pandas as pd

from sklearn import tree
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import
accuracy_score,classification_report,confusion_matrix
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
      import matplotlib.pyplot as plt
```

2. A continuación, se importa el conjunto de datos anotados. En este caso, se leen los datos de los sensores y se generan las características o *features* que se van a emplear

3. Se preparan los datos, es decir, se divide la información en atributos y etiquetas. Acto seguido, se dividen los datos en conjuntos de entrenamiento y de pruebas. De esta forma se puede entrenar el algoritmo con un conjunto de datos y luego probarlo en otro conjunto de datos completamente distinto, que el algoritmo aún no ha visto.
   En este apartado, se emplea la biblioteca `model_selection` de *Scikit-Learn* que contiene el método `train_test_split`, el cual usaremos para dividir aleatoriamente los datos en conjuntos de entrenamiento y prueba.

En el código, el parámetro `train_size` especifica la proporción del conjunto de entrenamiento, que usamos para dividir el 75% de los datos en el conjunto de entrenamiento y el 25% para el test.

4. Una vez que los datos se han dividido en conjuntos de entrenamiento y prueba, el paso final es entrenar el algoritmo del árbol de decisión sobre estos datos y hacer predicciones. *Scikit-Learn* contiene la biblioteca `tree`, que contiene clases/métodos integrados para varios algoritmos de árboles de decisión. En este caso, se va a realizar una tarea de clasificación, por lo tanto, se emplea la clase `DecisionTreeClassifier`. A continuación, se llama al método `fit` de esta clase para entrenar el algoritmo con los datos de entrenamiento, que se pasan como parámetro al método.
   Una vez que el clasificador ha sido entrenado, se hacen predicciones sobre los datos de prueba. Para hacer predicciones se utiliza el método `predict` de la clase `DecisionTreeClassifier`.

5. Finalmente, se evalúa el algoritmo. Para las tareas de clasificación, algunas métricas de uso común son la matriz de confusión y la precisión. La biblioteca `metrics` de *Scikit-Learn* contiene los métodos `accuracy_score`, `classification_report` y `confusion_matrix` que se pueden emplear para calcular estas métricas.

A continuación, se incluye el código completo del entrenamiento:

```
    def trainingSensor(fileName):

        #read data
        print("Train read")
        data=genFeature(fileName,2500,1250)
        features=data['feature'].to_numpy()
        labels=data['labelRed'].to_numpy()

        #split data set
        print("Train split")
        X_train, X_test, y_train, y_test = train_test_split(features,
labels, random_state=100, train_size = 0.75)
        X_train= X_train.reshape(-1, 1)
        y_train= y_train.reshape(-1, 1)
        X_test = X_test.reshape(-1, 1)
        y_test= y_test.reshape(-1, 1)
        clf_model = DecisionTreeClassifier(criterion="gini", max_depth=4,
min_samples_leaf=4)

        #train
        print("Train !!!!")
        clf_model.fit(X_train,y_train)

        #test
        print("Traint Test ")
        y_predict = clf_model.predict(X_test)

        #report accuracy
```

```
    accuracy= accuracy_score(y_test,y_predict)
    print( " Accuracy= " , accuracy)
    conf=confusion_matrix(y_test,y_predict)
    print( " Confusion matrix= " , conf)
    matrix = confusion_matrix(y_test, y_predict)
    acc= matrix.diagonal()/matrix.sum(axis=1)
    print(acc)

    # prepare the cross-validation procedure
    cv = KFold(n_splits=10, random_state=1, shuffle=True)
    lab=labels.reshape(-1, 1)
    fea=features.reshape(-1, 1)
    # create model
    model = DecisionTreeClassifier(criterion="gini", max_depth=4,
min_samples_leaf=4)
    # evaluate model
    scores = cross_val_score(model, fea, lab, scoring='accuracy',
cv=cv, n_jobs=-1)
    # report performance
    print('Accuracy: %.3f (%.3f)' % (np.mean(scores), np.std(scores)))
    return accuracy
```

Tras ejecutar el programa, se muestran los resultados obtenidos. Como se puede observar, la precisión es del 89.6 % cuando se utiliza el mismo conjunto de datos que se ha empleado con el programa del Apartado 6.1. Con los datos obtenidos con nuestros sensores, la precisión es menor (72%).

```
>>>from library import trainCollar
>>>file='.\\workspace\\Python\\library\\test\\G1.csv'
>>>trainCollar.trainingSensor(file)

Train read
readSensorData:    Number    of    samples:    2116944        in    the
file.\workspace\Python\library\test\G1.
Train split
Train !!!!
Traint Test
 Accuracy=  0.896551724137931
 Confusion matrix=
[[48  0  0]
 [ 8 30  0]
 [ 0  1  0]]
[1.         0.78947368 0.        ]
Accuracy: 0.896 (0.059)
0.896551724137931
```

# Capítulo 8. Conclusiones y líneas futuras

El objetivo principal de este trabajo ha sido detectar las actividades que realizan los animales a partir de las mediciones realizadas con sensores colocados en el cuello de estos. La amplia literatura analizada describe limitaciones incluso a la hora de detectar acciones básicas del animal. Algunos estudios describen resultados bastante prometedores, pero las metodologías empleadas presentan ciertas limitaciones a su uso en sistemas comercializables.

Durante el desarrollo del trabajo se han encontrado algunos problemas relacionados con los sensores. En primer lugar, se encontraron fallos en el sensor comercial, proporcionado para calibrar nuestro propio sensor, lo que ha conllevado la realización de pruebas extra para detectar cuáles eran los problemas de dicho dispositivo. Otro contratiempo ha sido la necesidad de desarrollar varias versiones del software, hasta dar con la que funcionase correctamente para realizar la toma de datos.

La anotación de los datos mediante los vídeos y el programa BORIS ha sido un proceso minucioso y costoso en cuanto a tiempo. Aunque los vídeos son relativamente cortos (20 minutos) es necesario realizar su visionado varias veces para poder realizar las anotaciones correctamente y que estas estén bien sincronizadas con los datos. Esto es debido a que, en algunos momentos, se pueden dar hasta tres actividades en cuestión de segundos.

Después de solucionar los problemas encontrados con los sensores, se han entrenado árboles de decisión utilizando 2 aproximaciones: *Scikit-learn* y *UNICO-GUI*. Con ambas se obtienen resultados similares. Si se utiliza el conjunto de datos que emplean otras publicaciones, se obtiene una precisión en la clasificación de acciones básicas del 89.6%, tanto utilizando el paquete *scikit-learn* como con *UNICO-GUI*. Estos resultados son bastante buenos cuando se comparan con el 80 – 90% de precisión obtenida en otros trabajos analizados. Los conjuntos de datos públicos disponibles presentan algunas características que no tienen los datos capturados con nuestros sensores y que parecen favorecer la obtención de buenos resultados:

- La cantidad de datos que componen el *dataset* es elevado, y cada fragmento de video tiene una duración de casi tres horas.
- A diferencia de nuestro *dataset,* en donde los animales se encuentran pastando en lugares abiertos y, por lo tanto, con pendientes y entornos variables, los animales de los conjuntos de datos de otros estudios se encuentran encerrados en parcelas planas, con comederos artificiales. Esta característica obliga a tener cuidado a la hora de extraer conclusiones a partir de los resultados obtenidos con *dataset* públicos.

A pesar de que los resultados obtenidos han sido bastante cercanos a los obtenidos en otros estudios, hay que tener en cuenta varios puntos de cara a líneas futuras de investigación:

- Es necesario obtener un gran volumen de datos para entrenar los modelos. No solo es necesario incrementar las horas de grabación sino también la variedad de animales y tipos de acciones a analizar. Una gran variedad de individuos y situaciones puede incrementar la capacidad de generalización de los modelos obtenidos.

- Tras revisar varios estudios, se llega a la conclusión de que el sensor más relevante es el acelerómetro y la frecuencia de muestreo más adecuada son los 100Hz.

# Bibliografía


[1] «CuadernosEntretantos1_GanaderíaExtensiva.pdf». Accedido: 15 de julio de 2022. [En línea]. Disponible en: http://www.ganaderiaextensiva.org/wp-content/uploads/2014/10/CuadernosEntretantos1_Ganader%C3%ADaExtensiva.pdf

[2] D. Berckmans, «General introduction to precision livestock farming», *Animal Frontiers*, vol. 7, n.º 1, pp. 6-11, ene. 2017, doi: 10.2527/af.2017.0102.

[3] «Ganadería de precisión: clave para optimizar la eficiencia productiva | CONtexto ganadero». https://www.contextoganadero.com/internacional/ganaderia-de-precision-clave-para-optimizar-la-eficiencia-productiva (accedido 18 de julio de 2022).

[4] «¿Qué es la ganadería de precisión?», *BLOG*, 20 de noviembre de 2020. https://www.repuestosfuster.com/blog/ganaderia-de-precision-que-es/ (accedido 18 de julio de 2022).

[5] D. Birant y K. Yalniz, «Animal Activity Recognition From Sensor Data Using Ensemble Learning», *Emerging Trends in IoT and Integration with Data Science, Cloud Computing, and Big Data Analytics*, 2022. https://www.igi-global.com/chapter/animal-activity-recognition-from-sensor-data-using-ensemble-learning/www.igi-global.com/chapter/animal-activity-recognition-from-sensor-data-using-ensemble-learning/290080 (accedido 19 de julio de 2022).

[6] G. Bishop-Hurley *et al.*, «An investigation of cow feeding behavior using motion sensors», en *2014 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) Proceedings*, may 2014, pp. 1285-1290. doi: 10.1109/I2MTC.2014.6860952.

[7] O. R. Bidder *et al.*, «Love Thy Neighbour: Automatic Animal Behavioural Classification of Acceleration Data Using the K-Nearest Neighbour Algorithm», *PLOS ONE*, vol. 9, n.º 2, p. e88609, feb. 2014, doi: 10.1371/journal.pone.0088609.

[8] J. W. Kamminga, D. V. Le, J. P. Meijers, H. Bisby, N. Meratnia, y P. J. M. Havinga, «Robust Sensor-Orientation-Independent Feature Selection for Animal Activity Recognition on Collar Tags», *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, vol. 2, n.º 1, p. 15:1-15:27, mar. 2018, doi: 10.1145/3191747.

[9] L. Riaboff *et al.*, «Evaluation of pre-processing methods for the prediction of cattle behaviour from accelerometer data», *Computers and Electronics in Agriculture*, vol. 165, p. 104961, oct. 2019, doi: 10.1016/j.compag.2019.104961.

[10] D. Figo, P. Diniz, D. Ferreira, y J. Cardoso, «Preprocessing techniques for context recognition from accelerometer data», *Personal and Ubiquitous Computing*, vol. 14, pp. 645-662, oct. 2010, doi: 10.1007/s00779-010-0293-9.

[11] A. Géron, «Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow, 2nd Edition [Book]». https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/ (accedido 20 de septiembre de 2022).

[12] R. de la Vega, «⊳ Árboles de decisión en Python con Scikit-Learn | Pharos», 7 de febrero de 2021. https://pharos.sh/arboles-de-decision-en-python-con-scikit-learn/ (accedido 7 de septiembre de 2022).

[13] «SensorTile.box wireless multi-sensor development kit with user-friendly app for IoT and wearable sensor applications -», p. 5, abr. 2021.

[14] «Why RTOS and What is RTOS?», *FreeRTOS*. https://www.freertos.org/about-RTOS.html (accedido 20 de julio de 2022).

[15] «Localizador GPS para animales | GPS para ganado | Báscula para ganado», *digitanimal*. https://digitanimal.com/ (accedido 20 de septiembre de 2022).

[16] O. Friard y M. Gamba, «BORIS : a free, versatile open-source event-logging software for video/audio coding and live observations», *Methods Ecol Evol*, vol. 7, n.º 11, pp. 1325-1330, nov. 2016, doi: 10.1111/2041-210X.12584.
[17] «LSM6DSOX - iNEMO inertial module with Machine Learning Core, Finite State Machine and advanced Digital Functions. Ultra-low power for battery operated IoT, Gaming, Wearable and Personal Electronics. - STMicroelectronics». https://www.st.com/en/mems-and-sensors/lsm6dsox.html (accedido 22 de julio de 2022).
[18] «an5259-lsm6dsox-machine-learning-core-stmicroelectronics.pdf».
[19] «lsm6dsox.pdf».
[20] «Unico-GUI - MEMS evaluation kit software package for Linux, Mac OSX and Windows - STMicroelectronics». https://www.st.com/en/development-tools/unico-gui.html (accedido 4 de agosto de 2022).
[21] «STMems_Machine_Learning_Core/configuration_examples/example_4_stwin_stble_unico at master · STMicroelectronics/STMems_Machine_Learning_Core», *GitHub*. https://github.com/STMicroelectronics/STMems_Machine_Learning_Core (accedido 12 de febrero de 2022).
[22] «scikit-learn: machine learning in Python — scikit-learn 1.1.2 documentation». https://scikit-learn.org/stable/ (accedido 8 de agosto de 2022).
[23] L. Gonzalez, «Introducción a la librería Scikit-Learn de Python», *Aprende IA*, 2 de noviembre de 2018. https://aprendeia.com/libreria-scikit-learn-de-python/ (accedido 18 de agosto de 2022).