Universidad Zaragoza
1542

Trabajo Fin de Máster

Plataforma inalámbrica en collar para reconocimiento del comportamiento de ovejas

Wireless platform in collar for recognition of sheep behavior

Autor

Pablo Escribano García

Directores

Roberto Casas Nebra
Julio David Buldain Pérez

ESCUELA DE INGENIERÍA Y ARQUITECTURA

2020-2021

Resumen

Tras la controversia generada entre 2007 y 2010 sobre los posibles efectos adversos producidos por el uso de adyuvantes como el Aluminio en animales rumiantes durante las campañas de vacunación en Europa para hacer frente al virus de la lengua azul, diferentes grupos de investigación comienzan a trabajar para determinar si efectivamente el uso de estos adyuvantes puede producir estos efectos secundarios en los animales.

Para validar esta hipótesis, se ha de estudiar si existe un notable cambio en los patrones de comportamiento de los animales rumiantes, tras aplicar las vacunas con este adyuvante. Estos patrones de comportamiento son a lo que llamamos habitualmente "actividades". Dentro de estas actividades que realizan los animales rumiantes se encuentran: correr, andar, descansar, rumiar etc. Por lo tanto, el primer paso consiste en ser capaces de reconocer las actividades que hacen los animales para después analizar los posibles cambios que se puedan producir.

Este reconocimiento de actividades puede realizarse por medio de un experto que observe los animales en contacto directo con ellos, lo cual es un trabajo tedioso y de larga duración, o bien por medio de algún dispositivo entrenado en el reconocimiento y clasificación de estas actividades, que se encargue de hacer este trabajo.

En concreto, este trabajo fin de máster plantea el desarrollo de un sistema de adquisición de datos que, mediante sistemas de aprendizaje automático, permita el reconocimiento de actividades de ovejas, animal rumiante de estudio, en base a los datos de diferentes sensores.

La idea es que, gracias a este sistema, el proceso de reconocimiento de actividades no requiera de una persona en contacto continuo con los animales, sino que sea un sistema automático y permita en un futuro extraer conclusiones sobre el uso de los adyuvantes con Aluminio.

Además, aprovechando el uso de la tecnología BLE (*Bluetooth Low Energy*), se trabaja en nuevas ideas para detectar las relaciones entre ovejas que permitan posteriormente validar si existen cambios en las relaciones entre individuos, antes y después de que a alguno de ellos se le aplique la vacuna.

Por lo tanto, el presente estudio muestra el procedimiento seguido en la creación del dispositivo de reconocimiento de actividades de ovejas y los resultados obtenidos.

Índice

Capítulo 1

Introducción

1.1. Motivación y contexto

Las vacunas han contribuido significativamente a la salud mundial, tanto para humanos como para animales. Uno de los adyuvantes de vacunas con mayor rendimiento y más económico son las sales de Aluminio. Sin embargo, en los últimos años, el uso de este adyuvante ha suscitado cierta controversia por la generación de posibles efectos adversos no contemplados anteriormente [1].

El inicio de este cuestionamiento sobre el adyuvante de Aluminio se inició entre los años 2007 y 2010, donde se llevaron a cabo varias campañas de vacunación en Europa para hacer frente a un virus emergente, llamado virus de la lengua azul. En España se aplicaron cuatro vacunas a los animales rumiantes, las cuales contenían Aluminio (como adyuvante), además del virus ligeramente activado. Con estas cuatro vacunas a los animales se les acabaron inoculando 16 mg de Aluminio. Tras estas vacunaciones se reportó un síndrome, que previamente no se había observado y ciertos trastornos neurológicos, bajadas extremas de peso y cambios conductuales en los animales [1].

Tras estas observaciones se realizaron investigaciones al respecto, en las que algunos investigadores mantuvieron que el aluminio no suponía un riesgo para la salud de los animales, mientras que otros reportaron efectos adversos. Finalmente se llamaba a continuar las investigaciones en este campo [1].

Es en este contexto donde la Facultad de Veterinaria de Zaragoza inicia la colaboración con el grupo de investigación HowLab, para plantear una solución tecnológica que pueda ayudar en el proceso de reconocimiento de actividades que realizan las ovejas, así como los cambios sociales que aparecen, antes y después de aplicarles el adyuvante de Aluminio.

Este es el punto de partida de nuestra investigación que abarca desde el estudio del estado del arte hasta la elaboración de una herramienta de clasificación que realizar desarrollar trabajos futuros.

HOW
Universidad Zaragoza

Escuela de
Ingeniería y Arquitectura
Universidad Zaragoza

1.2. Objetivos

El principal objetivo de este Trabajo de Fin de Máster consiste en el estudio y el desarrollo de un sistema de reconocimiento de actividades de ovejas. Tras analizar el estado del arte y mediante unas primeras pruebas de campo, se verifica la viabilidad técnica del proyecto, se definen los objetivos a alcanzar y se plantea una metodología para ello. Estos objetivos son:

- Estudio del arte de técnicas actuales utilizadas para el reconocimiento de las actividades de los animales.
- Desarrollo de una plataforma para la adquisición de los datos sobre los que plantear hipótesis y algoritmos de trabajo.
- Desarrollo de hardware y firmware adaptado al proyecto y a las especificaciones que se plantean desde el grupo de investigación de la facultad de veterinaria.
- Construcción de una base de datos de interés sobre las ovejas.
- Elaboración de un sistema que facilite y agilice el proceso de etiquetado de los datos.
- Estudio y tratamiento de los datos adquiridos.
- Desarrollo del clasificador.
- Implementación del clasificador dentro de un sistema embebido que permita la detección de actividades in situ.
- Extracción de conclusiones y análisis de los resultados para abrir camino a futuros trabajos relacionados con esta tecnología.

1.3. Alcance

Para comprobar que se cumplen todos los objetivos planteados se divide el proyecto en diferentes fases con objetivos parciales. Estas fases son las siguientes:

Fase inicial: consiste en una recopilación de las investigaciones realizados hasta la fecha en relación con los sistemas de telemetría animal, tratando de recoger toda la información que pudiera ser de utilidad para el presente proyecto.

Fase dos: se apoya en el estudio realizado en la fase inicial. En esta etapa se definen los parámetros de trabajo y se hace un planteamiento inicial del dispositivo a desarrollar.

Fase tres, se basa en la elaboración del firmware y del hardware que se convertirán en la plataforma de adquisición de datos. Durante esta fase son llevadas a cabo varias etapas hasta lograr una primera versión de hardware y software

completamente funcional y optimizada. Durante esta etapa también se llevarán a cabo las sesiones de adquisición de datos de las ovejas, con las que se elaborará una base de datos propia sobre la que trabajar posteriormente.

Fase cuatro: consiste en el procesado y etiquetado de los datos. Para este punto se ha de desarrollar una herramienta gráfica de etiquetado que acelera este proceso, que por lo general es lento. Por otro lado, se analizan los datos, se normalizan, se filtran y se enventanan, para posteriormente determinar las características con las que entrenar los algoritmos de aprendizaje supervisado.

Fase cinco: trata del entrenamiento y el testeo de los clasificadores, determinando las mejores características con las que entrenar al modelo.

Fase seis: consiste finalmente, en implementar el clasificador dentro del dispositivo y verificar su correcto funcionamiento.

1.4. Estructura

Este proyecto está estructurado en capítulos como sigue:

- Capítulo 1 -Introducción

- Capítulo 2 - Estado del arte:
  En este capítulo se recoge toda la investigación realizada sobre el estado del arte en el reconocimiento de actividades animales, atendiendo a los procedimientos llevados a cabo para la creación de un equipo de telemetría de reconocimiento de actividad animal.

- Capítulo 3 - Diseño del sistema de adquisición de datos:
  En este capítulo se recogen las diferentes etapas por las que se ha pasado hasta lograr tener una versión de hardware y firmware definitivos que cumplieran las funcionalidades requeridas.

- Capítulo 4 - Sesiones de adquisición y creación de una base de datos:
  En este capítulo se explica el procedimiento seguido para la creación de la base de datos de ovejas, con los datos recogidos.

- Capítulo 5 - Procesamiento de los datos:
  En este capítulo se recogen todas las técnicas para el procesado de los datos y la determinación de buenas características para ayudar en el entrenamiento de los algoritmos de aprendizaje automático. Además, se recogen los resultados obtenidos por los algoritmos de clasificación, previo a ser implementados en el microcontrolador.

- Capítulo 6 – Implementación en el microcontrolador:

En este capítulo se detalla el proceso de implementación de los algoritmos de clasificación en el microcontrolador, y se analiza si los resultados son idénticos a los que se obtendrían en el ordenador en el que han sido entrenados.

- Capítulo 7 - Conclusiones y Líneas Futuras:
  En este capítulo se extraen conclusiones de los análisis realizados en el proyecto y se recogen las líneas futuras en las que se hará uso del sistema de telemetría planteado.

Capítulo 2

Estado del arte

2.1. Introducción

El primer acercamiento en esta materia es el análisis de cómo se ha llevado a cabo hasta la fecha el reconocimiento y clasificación de las actividades que realizan los animales. Tradicionalmente la forma de analizar cambios comportamentales es mediante la observación directa de los animales por parte de un especialista que anotan tanto en una libreta como de forma digital, las diferentes actividades que realizan [5]. Estas actividades han sido definidas previamente en un etograma.

Este método presenta tres problemas en relación con la creación de modelos precisos de los animales: El primer problema es que se requiere mucha mano de obra y un largo tiempo para realizar observaciones, lo que no siempre es posible [2]. El segundo problema, es que esta metodología no se puede aplicar por igual a todas las especies, siendo más complicada llevarla a cabo con animales salvajes que con animales domésticos [2]. El tercero es que cuando una persona va a la naturaleza para aprender de ella, su intervención puede generar perturbaciones a su alrededor dando lugar a observaciones erróneas [3].

Actualmente ya existen herramientas tecnológicas que ofrecen soluciones a estos problemas. Por ejemplo, los sistemas de posicionamiento GPS (*Global Positioning System*) han conseguido mejorar la localización animal, dejando de ser necesaria una persona cerca del animal, para saber su ubicación[2]. También se ha conseguido reducir el esfuerzo necesario en la monitorización manual de los animales gracias a las técnicas de clasificación automática procedentes del campo de la inteligencia artificial [4], las cuales se apoyan en la reducción de tamaño, peso, coste y la mejora de resolución que han experimentado los sensores, gracias su desarrollo por su introducción en el sector del automóvil [4][7][41]. Estas herramientas tecnológicas permiten aumentar el número de estudios de comportamientos sociales [2], gracias a técnicas actualizadas basadas en ZigBee o Bluetooth Low Energy [6][28]. Estas técnicas ya se han probado en granjas, facilitando las labores de reconocimiento del comportamiento animal y permitido conocer cuál es su estado de bienestar y de salud, permitiendo tomar las mejores decisiones en su cuidado [6].

Sin embargo, a pesar de las mejoras que introduce el uso de estas nuevas técnicas, también existen algunos problemas asociados. El primero es la cantidad de datos generados por equipos como los sensores inerciales [4], que agotan el espacio de las tarjetas de memoria utilizadas como almacenamiento [7]. Este problema puede solucionarse mediante la transmisión de estos datos por medio de dispositivos de radiofrecuencia, como se ve en [4] y [10] con el uso de WIFI. Sin embargo, hay situaciones, al trabajar con animales salvajes, donde se está lejos del animal y la señal de radiofrecuencia no tiene alcance, que se tendría que continuar almacenando los

datos internamente en el dispositivo, continuando con el problema del almacenamiento limitado[8].
Otro problema asociado a estas nuevas tecnologías es la limitación de la cantidad de energía que pueden almacenar las baterías en relación con su peso [4], ya que se desea que el equipo de medida sea ligero para interferir lo menos posible con el animal, pero al mismo tiempo estos sistemas deben ser lo más autónomos posible.

En este contexto, se debe trabajar para construir una metodología que permita sobrepasar estos obstáculos. En concreto, en [4] se detallan una serie de pasos comunes a la mayoría de los estudios analizados y que se describe en cinco etapas:

Etapa 0 - Definición etograma, es una etapa de definición de actividades por medio de una etograma, en el que se definen las actividades a observar y se describe cómo será reconocida cada una de ellas por los etólogos (Ver [9] [10]). Estos etogramas pueden contener una descripción muy detallada de las actividades o una descripción muy sencilla.
Ver Tabla 1 y Tabla 2

Collective Movement Activites	Description
Not active	Where the animals are gathered together in close proximity with little or no movement activities. Correspond to instances where animals are resting or sleeping.
Active	The animals are moving scattered in ther habitat with diverse movement activities.
Herd Movement	The animals are being herded at high velocity over a narrow space.

Tabla 1: Vista de un etograma sencillo [9]

Behaviour Category	Definition
Standing	The cow stands on all four legs with head erect and without swinging its head from side to side.
Lying	The cow lies in a cubicle without rumianting in any position except flat on side.
Rumianting	The cow lies in a cubicle masticating regurgitated feed, swallowing masticated feed or regurgitating feed with head erect.
Feeding	The cow places its head above the feeding table and searches, masticates or sorts the feed.
Normal walking	The cow walks straight forward with even gait.
Lame walking	The cow walks straight forward with arched back and taking shortened steps with one or more legs
Lying down	The cow bents one foreleg, lowers its forequarters, then hindquarters, and settles down in lying position.
Standing up	The cow lunges forward, lifts its hindquarters, then forequarters and rises to sand on all four feet.

Tabla 2: Etograma detallado [10]

Etapa 1- Desarrollo del dispositivo, consiste en el planteamiento del sistema hardware y firmware, determinando los sensores a utilizar para reconocer las actividades definidas en el etograma, la definición del periodo de muestreo y la selección de la mejor ubicación de los sensores en el animal.

Etapa 2 - Adquisición de los datos, abarca la fase de recopilación de datos durante un tiempo determinado en cada sesión de adquisición. El número de sesiones depende de los medios de los que se disponga y la precisión en el reconocimiento de actividades que se desee alcanzar. Es importante definir cómo se hará posteriormente la sincronización de los datos recopilados, con el vídeo que se esté grabando o con las anotaciones que estuvieran tomando los etólogos durante ese momento.

Etapa 3 - Procesado de los datos, consiste en un procesamiento de los datos por medio de su etiquetado manual con veterinarios o especialistas. Para este etiquetado

se puede haber grabado con cámaras el experimento, para no interferir en el comportamiento animal [12], o haber estado observando los etólogos durante ciertas horas del día a los animales y haber etiquetado manualmente los datos [11].

Etapa 4 - Desarrollo del clasificador, consiste en la creación y el entrenamiento del algoritmo de clasificación y su implementación posteriormente en el hardware.

Etapa 5 - Testeo y evaluación de la plataforma, durante esta etapa se vuelve a instalar el dispositivo sobre el animal y se procede a testear el grado de acierto del sistema desarrollado en el reconocimiento de actividades, si no se hubiera alcanzado el grado de acierto deseado se volverán a reproducir los pasos desde la etapa 2 hasta la etapa 5.

Esta metodología es la que se ha adoptado como metodología a seguir en el presente trabajo de fin de Máster.

2.2. Desarrollo del dispositivo

Superada la etapa 0 de definición del etograma, necesaria para determinar las actividades que han de ser reconocidas, se pasaría a la etapa 1 - Desarrollo del dispositivo, en la que los autores analizan qué hardware es el más conveniente según el tipo de animal con el que se va a trabajar, atendiendo a varios aspectos como son: sensores a utilizar, procedimiento más adecuado para guardar los datos, duración de las sesiones de toma de datos a realizar, tamaño de las baterías a utilizar, frecuencia de muestreo de los datos y tamaño de la ventana de procesamiento de esos datos.

2.2.1. Sensores para reconocimiento de actividades

Gracias a que en los último años se ha producido una mejora exponencial de la tecnología, los dispositivos de seguimiento son cada vez más pequeños, permitiendo monitorear a animales de menor tamaño y con una mayor resolución [19].

El acelerómetro es el sensor de animales que se utiliza en la mayor parte de los artículos analizados, acompañados de otros sensores como GPS, magnetómetro, giroscopio, sensores ambientales [2], micrófonos o relojes de pastoreo. Hay que destacar la utilidad del sensor de temperatura como sistema de calibración del acelerómetro [6]. En la Tabla 3 se muestran cuáles son los sensores que se utilizan en cada artículo según el animal de estudio.

Los acelerómetros son una herramienta de alto potencial para este tipo de aplicaciones [21] debido a 2 factores: permiten inferir correctamente las actividades de los animales y presentan una buena relación entre gasto energético y su forma de onda [2], que permite determinar el tiempo que un animal descansa vs el tiempo que realiza otras actividades [25]. El segundo factor, es su continua reducción en tamaño, peso y precio, debido a su incorporación en la industria del automóvil [2]. Característica

que lo hace muy adecuado para su colocación en animales pequeños y grandes, sin afectar a su comportamiento [24].

Esta reducción del tamaño de los sensores es importante ya que, para no modificar los comportamientos habituales de los animales, los biólogos apuntan a utilizar dispositivos de medición de peso inferior al 3% del peso del animal [19]. Véase la aplicación de esta regla en la medición de babuinos donde el peso del collar que se les acopló se dimensionó para que fuera inferior al 3% del peso de este [21]. En la Tabla 3 se muestra el peso de los dispositivos según el animal a estudiar.

El GPS es el sensor más utilizado para registrar la localización de un animal en el exterior, aunque presenta como desventajas un alto consumo energético y que su cobertura se reduce en zonas arboladas [6]. Para solucionar estos problemas en trabajos como [13] se proponen soluciones a este elevado consumo del GPS, utilizando el acelerómetro como mecanismo para modificar su periodo de muestreo, que habitualmente es constante, para que se adapte al nivel de activación del animal, logrando de esta manera reducir el consumo del GPS que sólo tomaría muestras si el acelerómetro indica que el animal se encuentra en movimiento [13]. De esta forma se optimiza el consumo energético del sensor, el cual es siempre una de las principales limitaciones que tienen los sistemas de adquisición de datos [5].

La técnica de utilizar GPS en combinación con el acelerómetro supone una gran ventaja al trabajar con animales pequeños, ya que estos suelen descansar en sitios donde la cobertura GPS es de baja calidad y tomar una muestra de la localización en ese sitio redundaría en un gasto energético innecesario[13]. De este modo sólo se tomarían datos si el acelerómetro indica actividad.

Tras este análisis donde se aprecia que acelerómetro y GPS son los dos sensores más utilizados para la monitorización animal, se decide incorporar el sensor de acelerómetro como un componente obligatorio en el dispositivo final. En principio, al trabajar con ovejas estabuladas se decide prescindir del GPS.

2.2.2. Frecuencia de muestreo:

Uno de los puntos más importantes a evaluar para reconocer de forma precisa las actividades de un animal, es la frecuencia de muestreo de los sensores [5], que puede ser baja o alta.

El muestreo a baja frecuencia, aunque supone un menor consumo energético, pero puede dar lugar a una pérdida de información [13]. Para evitar este problema, la solución consiste en aplicar el teorema de Nyquist-Shannon y muestrear como mínimo al doble de la frecuencia más rápida de la señal que se desee medir, para así poder reconstruir correctamente la señal muestreada.

El muestreo a frecuencias muy elevadas proporciona una gran resolución y se evita problemas de *aliasing* pero genera un gran volumen de datos y un alto consumo de batería. Es necesario el muestreo a alta frecuencia cuando se desea caracterizar de

forma precisa movimientos rápidos [14], lo que habitualmente sucede con animales de tamaño reducido[2]. Para evitar problemas, se puede considerar la conclusión extraída en [2] acerca de [14][15] y [16] donde sugiere no sobrepasar frecuencias de muestreo por encima de los 50 a 60 Hz. porque no se aporta información adicional y únicamente suponen un gasto del espacio de almacenamiento disponible. Esto no es una regla universal y debe ser estudiada para cada animal, de ahí que, que en los artículos analizados, cada autor elija, según su criterio, una frecuencia de muestreo y que posteriormente la analicen (Ver Tabla 4). Por ejemplo, en [4], aunque se realiza la toma de datos a 100 Hz para muestrear ovejas, el autor comenta que no ha observado información útil para frecuencias superiores a los 10 Hz.

Articulo	Referencia	Ar
Animal-borne behaviour classifcation for sheep and Rhinoceros	[4]	
Evaluation of sampling frequency, window size and sensor position for classification of sheep behaviour	[5]	
Using a three-axis accelerometer to identify and classify sheep behaviour at pasture	[12]	
Online Collective Animal Movement Activity Recognition	[9]	
Cow behaviour pattern recognition using a three-dimensional accelerometer and support vector machines	[10]	
ZigBee-based wireless sensor networks for classifying the behaviour of a herd of animals using classification trees	[6]	
Distinguishing Cattle Foraging Activities Using an Accelerometry-Based Activity Monitor	[11]	
Classification of behaviour in housed dairy cows using ACC monitoring system	[7]	
Using tri-axial acceleration data to identify behavioral modes of free-ranging animals: general concepts and tools illustrated for griffon vultures	[17]	
Pedalling downhill and freewheeling up; a penguin perspective on foraging	[18]	
Use of bio-loggers to characterize red fox behavior with implications for studies of magnetic alignment responses in free-roaming animals	[20]	
Identification of behaviours from accelerometer data in a wild social primate	[21]	
Fine-scale feeding behavior of Weddell seals revealed by a mandible accelerometer	[22]	

*En [5], la frecuencia de muestreo de 32Hz es la que mejor resultados les produce.

Respecto al problema de producir un gran volumen de datos por trabajar a alta frecuencia, en [23] se trabaja con la media de varias muestras consecutivas, disminuyendo la cantidad de datos a almacenar.

2.2.3. Ubicación de los sensores

Una buena ubicación del dispositivo de medida es esencial para detectar correctamente las actividades a reconocer, de lo contrario serán necesarios métodos de procesamiento de datos más sofisticados para su identificación [10]. Así, para poder detectar si un animal agacha la cabeza, puede resultar más interesante utilizar una cabezada (ubica al sensor bajo la mandíbula), que un collar (ubica el sensor en el cuello). Ver Figura 1.

Figura 1: Sistemas de sujeción de los sensores con collar [4] a la izquierda y cabezada [12] a la derecha.

Las principales soluciones para sujetar el sistema de adquisición de datos al animal, son los collares (Anexo I), los dispositivos en la oreja [5], las cabezadas o los arneses [6][7][20][19].

Los sistemas de amarre deben interferir lo menos posible con el animal, de ahí la regla propuesta en [21] de que tengan un peso inferior al 3% del peso del animal. El sistema de amarre del dispositivo, debe garantizar que el dispositivo de medida mantenga una orientación espacial determinada [10], especialmente en el caso de hacer uso de un acelerómetro para poder utilizar la información que aporta cada eje por separado (Ver Figura 2 y [5][6][7][12][17][21]). Gracias a mantener en una posición determinada el acelerómetro, en [11] pueden determinar si la vaca está en reposo o está pastando.

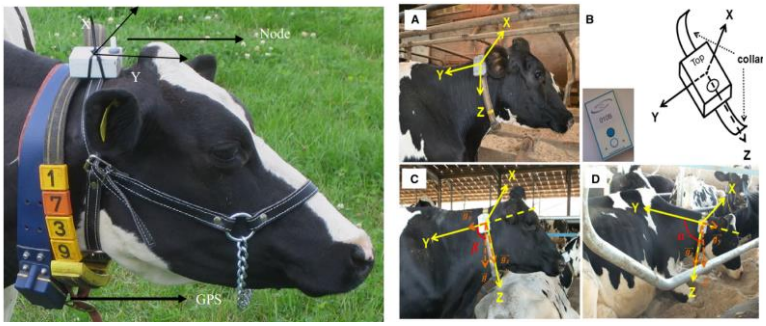


Figura 2: Dispositivos de telemetría con orientación determinada.

Para eliminar los datos producidos por pequeños movimientos ocasionados por holguras entre el animal y el sistema de adquisición, se utilizan filtros frecuenciales [2].

2.3. Adquisición y etiquetado de los datos

Para el proceso de creación de una base de datos, además de adquirir los datos de los sensores, se ha de recoger una etiqueta que indique la actividad que estaba realizando el animal durante este tiempo, para poder relacionarlos posteriormente.

El enfoque más utilizado hasta la fecha consiste en seleccionar un animal de forma aleatoria, (entre el grupo de animales de estudio) y dejarlo en un entorno controlado mientras que los etólogos, allí presentes, documentan manualmente las actividades que realiza el animal; esto es, sin hacer uso de cámara de vídeo [4][5][11][20]. La segunda aproximación, consiste en utilizar una grabación de video que permita posteriormente relacionar la información obtenida de los sensores con las actividades que realizan los animales. Visualizando el video, los expertos determinan estas etiquetas de comportamiento animal [10]. El problema de las grabaciones de video es que generan un gran volumen de datos a analizar y no son tan viables para estudiar animales salvajes [4].

Otro punto importante de las sesiones de adquisición de datos es su duración. Las sesiones de larga duración requieren de una mayor autonomía de la batería, lo que

a su vez implica un incremento en el peso de estas, que dependiendo del animal igual no es viable. Este compromiso autonomía frente a peso obliga a plantear diferentes estrategias: 1- cambios de la batería a lo largo de los días que dure el experimento [20 muestrear durante "n" segundos, en intervalos de "m" minutos en lugar de muestrear continuamente, evitando así transiciones o medidas muy repetitivas [2][11] y combinar la información de sensores de diferente consumo; como por ejemplo, combinar el acelerómetro de bajo consumo con un GPS de consumo mayor, para que el GPS sólo tome datos cuando el acelerómetro detecte movimiento, evitando tomar datos en momentos de baja actividad [21]. En la Tabla 5 se recogen diferentes artículos con la duración de los experimentos, tipos de baterías y estrategias utilizadas.

2.4. Desarrollo del clasificador

Antiguamente la clasificación de actividades animales se realizaba de forma manual, pero debido a lo tedioso del proceso y al desarrollo de las tecnologías, esta tarea se ha empezado a delegar a algoritmos de clasificación [4].

Existen estudios que tratan de inferir las actividades de los animales por medio de algoritmos de clasificación no supervisados [26], aunque el enfoque más común es el de utilizar métodos supervisados [2] que relaciona las actividades observadas de los animales en distintos instantes de tiempo con las señales muestreadas de los diferentes sensores [2]. Este método es el que se analiza a continuación más en detalle, pues es el más utilizado por los diferentes autores.

Para entrenar a estos clasificadores, en los artículos estudiados, los autores no utilizan los datos en crudo, sino que trabajan con unos datos ya procesados que aportan más información a estos clasificadores, llamados "features" o "características". Las características más habituales son la media, la derivada, la curtosis, la energía, la desviación estándar o la entropía [4][5][12] [6][20][21].
Estas características se extraen de un conjunto de datos que han sucedido en un cierto intervalo de tiempo. A la agrupación de estos datos se le llama "ventana" y hacer uso de ellas, en lugar de utilizar cada dato suelto, permite tener conocimiento no sólo de lo que sucede instantáneamente, sino de la secuencia en que ha sucedido.

El tamaño de estas ventanas es muy relevante en los resultados que obtiene el algoritmo de clasificación y depende de la frecuencia de muestreo seleccionada y de la dinámica del animal estudiado. Por ejemplo, para la clasificación de ovejas, la ventana de 7s muestreada a 32Hz es la que mejores resultados ofrece en [5], pero para la clasificación de rinocerontes es con una ventana de 6.5s con la que se obtienen los mejores resultados.

Entre una ventana y otra puede haber un conjunto de datos en común entre ellas, esta técnica es lo que se conoce como solapamiento y evita perder eventos en los datos obtenidos. Gracias a este solapamiento, en [5] con una ventana de 7s y un

solapamiento entre ventanas del 50%, se consigue mejorar los resultados de un 89 a un 95% de exactitud.

Por último, se ha de determinar el algoritmo de clasificación que se va a utilizar. Son numerosos los diferentes modelos de clasificación que existen y que se pueden aplicar, entre los que encontramos las redes convolucionales, las SVM, los árboles de decisión [7], los LDA o los LDR [4] [5][12] [9][10][6][11][7][20][21]. Cada algoritmo tiene sus ventajas y desventajas por lo que un estudio previo de sus propiedades es interesante para determinar cuál es el que mejor podría funcionar para cada aplicación.

Una vez el algoritmo de clasificación ha sido entrenado, hay que realizar unas pruebas para verificar su correcto desempeño. [9].

En la Tabla 6 se recoge como cada investigador ha aplicado cada uno de los puntos previamente tratados en esta sección, según el tipo de animal a investigar.

2.5. Conclusiones

Del estudio del estado del arte, en el reconocimiento de actividades animales, se extraen las siguientes conclusiones, que se aplican al trabajo de investigación realizado:

1- La primera tarea a realizar para llevar a cabo un reconocimiento de actividades animales consiste en definir un etograma que refleje cuáles son las actividades de interés y cómo identificarlas en el caso de que se sucedan.

2- De todos los sensores, el acelerómetro y el GPS son los más utilizados y los que mejores resultados obtienen en el reconocimiento de actividades, por lo que al menos se implementará uno de los dos en el proyecto.

3- Se ha de lograr un equilibrio entre la autonomía de la batería, el peso del dispositivo y la frecuencia de muestreo de los sensores, que permita realizar sesiones de larga duración, con dispositivos de peso inferior al 3% del peso del animal y que permita muestrear a una alta frecuencia que recoja todos los movimientos del animal de estudio.

4- El uso de cámaras de vídeo permite generar a posteriori las etiquetas de comportamiento animal sin interferir en su comportamiento normal. Esta técnica es muy útil para caracterizar el comportamiento del animal si no existen problemas de almacenamiento de los vídeos.

5- El uso de ventanas de datos y la extracción de sus características, se considera la técnica más adecuada para el procesado de los datos. En caso de existir mucho ruido en los datos se aconseja utilizar filtros frecuenciales y medias móviles.

6- Los algoritmos de aprendizaje automático están resultando ser cada vez más útiles para el reconocimiento de actividades, siendo los más implementados las SVM (*Support Vector Machines*), los Random Forest y LDA (*Linear Discriminant Analysis*).

2.5. Tablas comparativas

Articulo	Referencia	Animal Estudio	Sensores	Peso Dispositivo(g)
Animal-borne behaviour classifcation for sheep and Rhinoceros	[4]	Rinocerontes	GNS602 GPS receiver || ADXL345 tri-axial accelerometer (13 bit ±16g)	371
		Ovejas		126
Evaluation of sampling frequency, window size and sensor position for classification of sheep behaviour	[5]	Ovejas	Bosch BMI160 ±8g, 16 bit triaxial gyroscope || 16 bit triaxial accelerometer.	-
Using a three-axis accelerometer to identify and classify sheep behaviour at pasture	[12]	Ovejas	Data logger (prototype V1.0, AerobTec.) three-axis MEMS accelerometer (±8g)	-
Online Collective Animal Movement Activity Recognition	[9]	Ovejas	GPS	-
Cow behaviour pattern recognition using a three-dimensional accelerometer and support vector machines	[10]	Vacas	Three-dimensional accelerometer 8-bit (ADXL330, Analog Devices Inc., USA) ±3g	-
ZigBee-based wireless sensor networks for classifying the behaviour of a herd of animals using classification trees	[6]	Vacas	2-axis accelerometer 10bit (MTS310) and temperature sensor || GPS	-
Distinguishing Cattle Foraging Activities Using an Accelerometry-Based Activity Monitor	[11]	Vacas	LCEX s a single-axis accelerometer(0.06g to 1.94g)	-
Classification of behaviour in housed dairy cows using ACC monitoring system	[7]	Vacas	Tri-axial accelerometer (Xtrinsic MMA8451Q 3-Axis, 14-bit/8-bit Digital Accelerometer ±8 g)	250
Using tri-axial acceleration data to identify behavioral modes of free-ranging animals: general concepts and tools illustrated for griffo	[17]	Buitre	3D accelerometer measuring ACC || GPS	190 (2.4% of the mass)
Use of bio-loggers to characterize red fox behavior with implications for studies of magnetic alignment responses in free-roaming an	[20]	Zorros	±8 g accelerometer 14-bit || magnetometer ±400 µT	-
Identification of behaviours from accelerometer data in a wild social primate	[21]	Primates	Tri-Axial accelerometer	<3%

Tabla 3: Sensores utilizados en cada investigación y peso del dispositivo.

Articulo	Referencia	Animal Estudio	Frecuencia muestreo ACC (Hz)
Animal-borne behaviour classifcation for sheep and Rhinoceros	[4]	Rinocerontes	40
		Ovejas	100
Evaluation of sampling frequency, window size and sensor position for classification of sheep behaviour	[5]	Ovejas	8,16 y 32
Using a three-axis accelerometer to identify and classify sheep behaviour at pasture	[12]	Ovejas	5,10,25
Online Collective Animal Movement Activity Recognition	[9]	Ovejas	1
Cow behaviour pattern recognition using a three-dimensional accelerometer and support vector machines	[10]	Vacas	10
ZigBee-based wireless sensor networks for classifying the behaviour of a herd of animals using classification trees	[6]	Vacas	1
Distinguishing Cattle Foraging Activities Using an Accelerometry-Based Activity Monitor	[11]	Vacas	32
Classification of behaviour in housed dairy cows using ACC monitoring system	[7]	Vacas	50
Using tri-axial acceleration data to identify behavioral modes of free-ranging animals: general concepts and tools illustrated for griffon vultures	[17]	Buitre	3.3
Pedalling downhill and freewheeling up; a penguin perspective on foraging	[18]	Pinguino	6-9
Use of bio-loggers to characterize red fox behavior with implications for studies of magnetic alignment responses in free-roaming animals	[20]	Zorros	6
Identification of behaviours from accelerometer data in a wild social primate	[21]	Primates	40
Fine-scale feeding behavior of Weddell seals revealed by a mandible accelerometer	[22]	Focas	32

*En [5], la frecuencia de muestreo de 32Hz es la que mejor resultados les produce.

Tabla 4: Frecuencia de muestreo según artículo y animal de estudio.

Articulo	Referencia	Animal Estudio	Duración	Baterias	Recuperación datos	Observaciones	Camera
Animal-borne behaviour classifcation for sheep and Rhinoceros	[4]	Rinocerontes Ovejas	3 días diferentes	3.7 V 1800 mAh Baterías Ion Litio	2 GB MicroSD | Bajo consumo CC1101 GHz Transceptor RF (433MHz)	-	-
Evaluation of sampling frequency, window size and sensor position for classification of sheep behaviour	[5]	Ovejas	31h	Batería Li-Po 270 mA h	Área amplia de bajo consumo módulo de radio	-	HC-V380 64 GB SanDisk elite SDXC
Using a three-axis accelerometer to identify and classify sheep behaviour at pasture	[12]	Ovejas	5 días	Batería Li-Po 3.5 – 4.2V	Registrador de datos integrado Prototipo AML V1.0	-	Camara
Online Collective Animal Movement Activity Recognition	[9]	Ovejas	2 días	-	-	-	-
Cow behaviour pattern recognition using a three-dimensional accelerometer and support vector machines	[10]	Vacas	30 días	Batería Litio 3.7 V (2.4 Ah each)	Módulo para banda de radio de 2,45 GHz	-	Video Cámara
ZigBee-based wireless sensor networks for classifying the behaviour of a herd of animals using classification trees	[6]	Vacas	3 días	Baterías Alkaline AA total 2200mAh	Nodo Inalámbrico (2,48 GHz y máximo potencia 1 mW) y un GPS	-	-
Distinguishing Cattle Foraging Activities Using an Accelerometry-Based Activity Monitor	[11]	Vacas	4 días	CR2032 Batería Ion Litio a 3v	-	Intervalos de 1 min	No se utiliza cámara Etiquetado in situ.
Use of bio-loggers to characterize red fox behavior with implications for studies of magnetic alignment responses in free-roaming animals	[20]	Zorros	5 días (sesiones 30 min)	Li-Po (Capacidad aprox.130h)	2-GB flash unidad de memoria	Recarga de la batería cada 5 dias	Se usa un sistema de grabación

Tabla 5: Tipo de batería según la duración de la prueba. Sistema de recuperación de datos y si se hace uso de cámara o no según animal y artículo.

Articulo	Referencia	Animal Estudio	Algoritmo	Features	Rendimiento	Actividades	Tamaño ventana (s)
Animal-borne behaviour classifcation for sheep and Rhinoceros	[4]	Rinocerontes	LDA	Average signal magnitude, maximum and minumum value, mean, standard deviation, variance skewness, kurtosis,energy, spectral entropy, pairwise correlation between the axes	Accuracy [96.10%]	Standing, walking, grazing, running and lying down	6.5
		Ovejas			Accuracy [82.40%]		5.3
Evaluation of sampling frequency, window size and sensor position for classification of sheep behaviour	[5]	Ovejas	Random forest	Interquartile range, kurtosis, mean, standard deviation, minimum value, maximum value, number of zero crossing, spectral entropy, dominant frecuency, signal	Accuracy [89%-95%]	Lying, standing and walking.	7 and 5 (Overlap 50%)
Using a three-axis accelerometer to identify and classify sheep behaviour at pasture	[12]	Ovejas	Decision Tree	SMA, SVM, Movement Variation, Energy, Entropy, Pitch(°), Roll(°), Inclination (°)	Accuracy - 84.7%	Grazing, lying, running, standing and walking activities.	7
Online Collective Animal Movement Activity Recognition	[9]	Ovejas	Convolutional neural network.	Velocities of the animals and the distance of each sheep to their centroid.	77.02%(LSTM+CNN)	Active, inactive and Herd Movement.	-
Cow behaviour pattern recognition using a three-dimensional accelerometer and support vector machines	[10]	Vacas	SVM	Mean, standard deviation, skewness, kurtosis, maximum value, minimum value, and energy	Overall performance 78% precision - kappa value=0.69.	De pie, acostado, rumiando, alimentándose, caminando normal y cojo.	10
ZigBee-based wireless sensor networks for classifying the behaviour of a herd of animals using classification trees	[6]	Vacas	Decision Tree & Fuzzy logic	Kalman filter, weighted moving average window, velocity estimation and pitch	55% success rate (Decision tree) - 43% (fuzzy logic)	Active or inactive	-
Distinguishing Cattle Foraging Activities Using an Accelerometry-Based Activity Monitor	[11]	Vacas	LR & LDA	-	LDA (90.6% to 94.6%) || LR (80.8% to 91.8%)	Foraging, resting, ruminating, walking, and grooming	-
Classification of behaviour in housed dairy cows using ACC monitoring system	[7]	Vacas	decision-tree	-	**Lying** (77.42 % sensitivity, 98.63 % precision), **Standing** (88.00 % sensitivity, 55.00 % precision),	Feeding, lying and standing or transition	-
Use of bio-loggers to characterize red fox behavior with implications for studies of magnetic alignment responses in free-roaming animals	[20]	Zorros	5-nearest neighbor classifie	(1) the magnitude of the largest z-axis peak; (2) the time delay between the two largest z-axis peaks; (3) the energy in the output of a matched filter run on the z-axis; (4) the energy in the output of a 2-Hz high-pass fil□ter, averaged over the x- and y-axes.	Overall accuracy of 95.7%	Leaping, foraging, and trotting	2.5
Identification of behaviours from accelerometer data in a wild social primate	[21]	Primates	Random forest	**25 Features:** (1–3) tri-axial static acceleration; (4–5) pitch and roll; (6) vecto□rial dynamic body acceleration (VeDBA); (7) smoothed vectorial dynamic body acceleration (VeDBAs); (8–10) tri-axial partial dynamic body acceleration (PDBA); (11–13) the tri-axial PDBA-to-VeDBA ratio.(14–16) tri-axial power spectrum density (PSD); (17–19) maximum frequencies associated with the tri□axial PSDs; (20–22) the second maximum frequencies associated with the tri-axial PSDs; (23–25) the associated frequency for each axis.	Precision of 88.3% (±8.5%) and a mean recall of 70.7% (±29.3%) across all behaviours.	Resting, walking, running and foraging were all identified.	-

Tabla 6: Algoritmos y características más utilizados con su precisión y actividades a reconocer y tamaño de ventana según animal.

Capítulo 3

Diseño del sistema de adquisición de datos

3.1. Introducción

En este capítulo se muestra el procedimiento seguido en el desarrollo del firmware y el hardware del sistema de telemetría, se analizan las etapas por las que ha pasado el proyecto desde su toma de concepto a su versión final y se comenta el funcionamiento de la consola virtual UART BLE y la aplicación móvil desarrollada para facilitar el control del sistema a los investigadores de la facultad de veterinaria.

El sistema de telemetría animal quiso ser lo más exigente, en lo que a compromisos de funcionalidades eran requeridos en cada momento para las pruebas de estudio animal. En base a estos requerimientos se plantearon 4 etapas: etapa de validación de hipótesis iniciales, etapa de validación de funcionalidades, etapa de optimización y etapa de versión 1.0. Ver Figura 3.

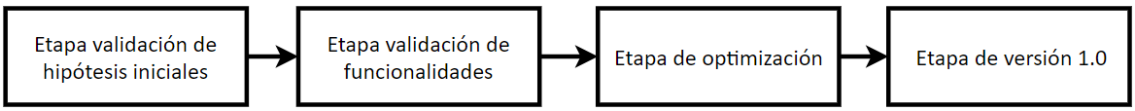


Figura 3: Etapas hardware y software del proyecto.

En cada etapa se utiliza un hardware (ver Tabla 8 en Anexo VIII) y un firmware adecuado al momento, aunque existen algunos elementos comunes a todas las etapas, como es la consola UART (*Universal Asynchronous Receiver-Transmitter*) sobre BLE (*Bluetooth Low Energy*). Por lo tanto, previo a comentar cada etapa del desarrollo. se describe el funcionamiento de esta consola que constituye un punto común a nivel de firmware en todas las etapas y es la que ha permitido la interacción entre el usuario y la electrónica de forma inalámbrica.

El enlace a todos los códigos del proyecto se encuentra: https://drive.google.com/drive/folders/1JybMIYH52VGYxqhebpuZjt7Xz7LtIorK?usp=sharing

3.2. Consola UART BLE

3.2.1. Principio de funcionamiento

Uno de los puntos comunes en todos los firmwares es la creación de una consola UART sobre BLE. Esta consola tiene como finalidad establecer una comunicación con el sistema cuando no puede ser accedido manualmente; por ejemplo, cuando está sobre la oveja. El funcionamiento general a nivel de BLE de esta comunicación se puede ver en la Figura 4.

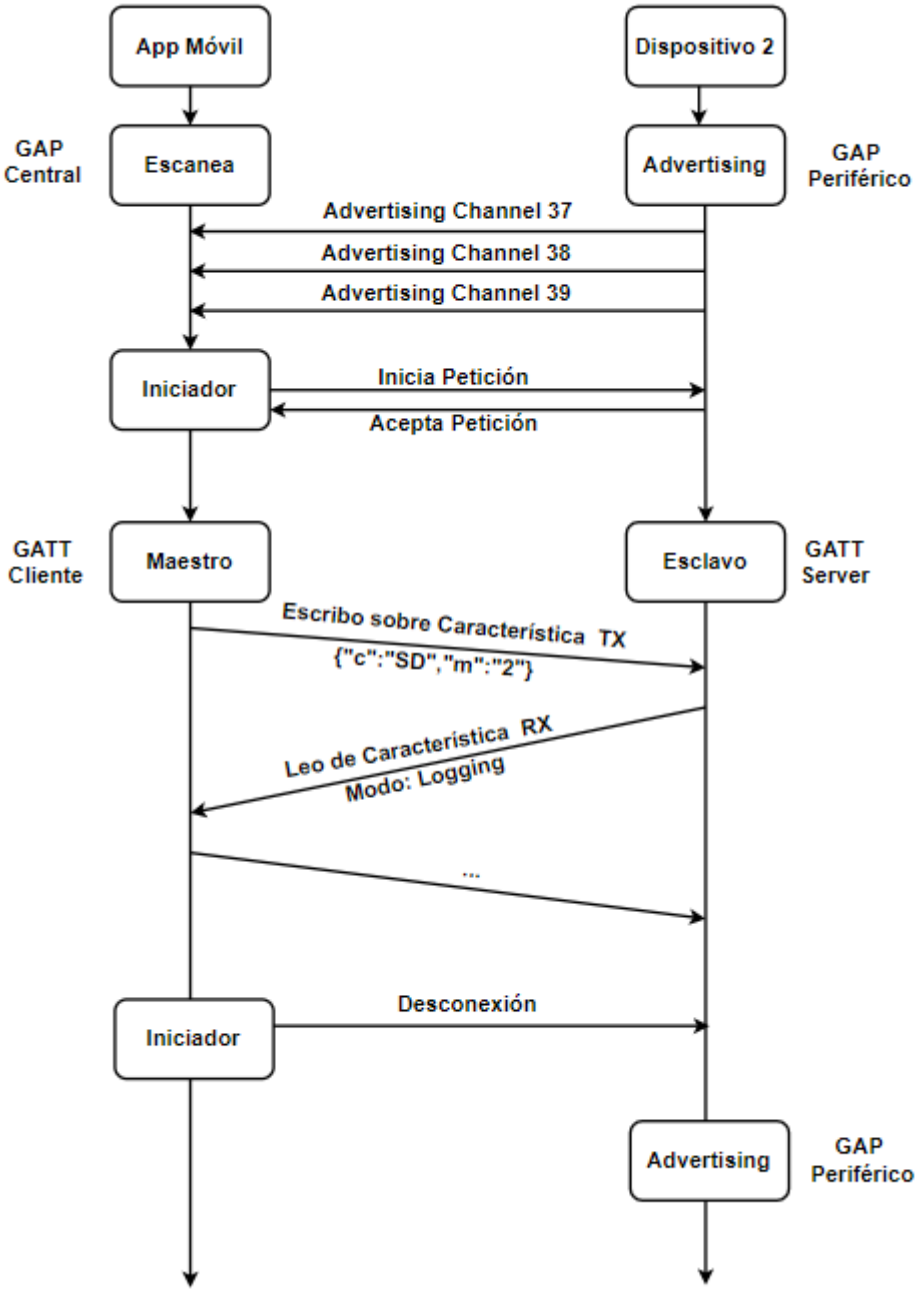


Figura 4: Representación de la comunicación BLE que sucede desde que se inicia el microcontrolador hasta que empieza en intercambio de comandos.

Son necesarios varios pasos dentro de la comunicación BLE con el dispositivo, para que el usuario pueda interactuar finalmente con la consola UART. Inicialmente, el sistema de telemetría se encuentra enviando mensajes de *advertising* indirectos (acepta conexión), en los que se incluye un identificador único que permite distinguir a ese dispositivo del resto. Este tipo de mensajes sirven para indicar al resto de dispositivos cercanos que hay un nuevo dispositivo encendido y que acepta conexiones. Este modo de funcionamiento BLE se conoce como modo "periférico" dentro de la capa GAP (*Generic Access Profile*) definida en el protocolo BLE.

Cuando algún dispositivo cercano hace una petición de conexión, esta es aceptada y el equipo cambia su role BLE y se comporta como un servidor GATT, ofreciendo varios servicios y características con las que puede interactuar el usuario que

realizado la conexión. Entre estos servicios hay uno con un UUID (*Universally Unique IDentifier*, traducido, Identificador universal único) de 128 bits que es el que alberga la consola UART BLE (Véase Figura *5*).

Este servicio contiene dos características, una llamada “característica TX” sobre la que se puede escribir, y otra de la que se puede leer, llamada “característica RX”. Estas dos características tienen un comportamiento similar al que tendría una interfaz UART física con su cable de transmisión (TX) y su cable de recepción (RX), donde se escriben comandos por la característica TX y se recibe una respuesta por RX. En el Anexo II se recoge una imagen de estas características en la aplicación NRF_Connect [29].

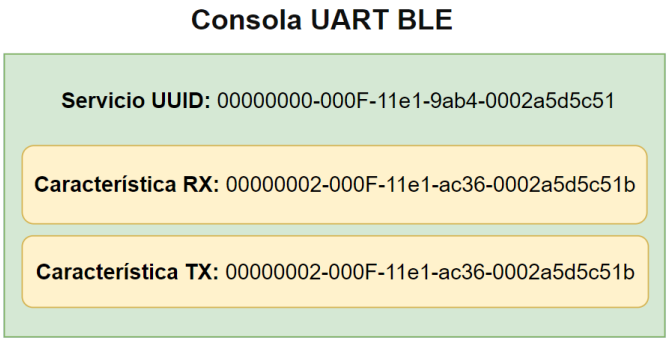


Figura 5: Imagen que muestra en términos generales como es el servicio de consola UART BLE.

3.2.2. Comandos

Los comandos que se pueden intercambiar entre el módulo y el dispositivo a través de la consola BLE están predefinidos y han sido diseñados en formato JSON (*JavaScript Object Notation*), con el propósito de poder ampliarse fácilmente en futuras versiones tanto los comandos, como los parámetros asociados a estos.

Existen dos tipos de comandos, los comandos de configuración y los comandos de estatus. Los comandos de configuración permiten modificar parámetros internos del microcontrolador, como su estado, la hora o la fecha. Los comandos de estatus permiten extraer información del microcontrolador, como su estado o el nivel de batería. Los comandos disponibles se encuentran en la Figura 6.

PROTOCOLO JSON (Sobre UART o sobre BLE en atributos RX y TX)

Comandos:
Set Device: Este comando pone en hora y selecciona el modo.

Poner en hora:
-- {"c":"SD", "t":"23050202062020"} //Se meterá asi la hora:(hora)(minuto)(segundo)(dia)(mes)(año) -> 23050202062020 = 23:05:02/02/06/2020
-- Respuesta puede ser -> T:OK o T:ERROR //Respuesta según si hubo un error o no poniendo el tiempo.

Establecer modo de funcionamiento:
-- {"c":"SD", "m":"0"} // Se mete el estado del microcontrolador 0 - (Dormido), 1 - (Espera Comandos), 2 - (Tomando datos), 3 - (Streaming)
-- Respuesta puede ser -> M:Dormido o M:ESPERA_COMANDOS o M:TOMANDO_DATOS o M:STREAMING o M:ERROR //Respuesta según estado

Get Device: Este comando devuelve la hora del sensor, el modo de funcionamiento y el estado de batería.

-- Provoca el envío de diferentes parámetros de estado (t=time, m=mode, b=batería, e= estado). Según el modo se puede mandar la respuesta de modo periódico
-- {"c":"GD","p":"b"} //Devuelve el valor de la batería
-- {"c":"GD","p":"h"} //Devuelve la hora del uC
-- {"c":"GD","p":"f"} //Devuelve el fecha del uC
-- {"c":"GD","p":"e"} //Devuelve el estado del uC

Figura 6: Imagen con los comandos de funcionamiento más utilizados.

3.2.3. Aplicación móvil

El acercamiento de la tecnología desarrollada al ámbito veterinario ha sido uno de los puntos en los que se ha prestado especial atención, desarrollando una aplicación móvil que facilita el manejo de los comandos BLE.

Con esta aplicación se pueden escanear los collares cercanos y establecer una conexión con el dispositivo que se desee, configurar la fecha, la hora y obtener el nivel de la batería.

En la Figura 7 se muestra como es la pantalla de la aplicación cuando se ha realizado un escaneo y se muestran los dispositivos cercanos, la ventana de selección de hora y fecha, cuando se pulsan estos botones de configuración y finalmente la aplicación tras haber realizado una conexión.

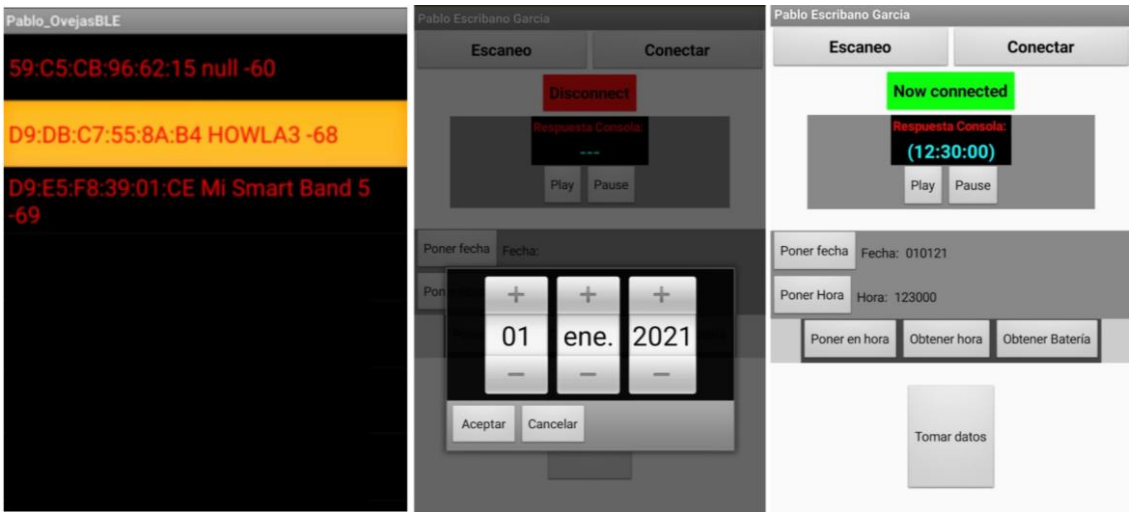


Figura 7: Capturas de pantalla de la App desarrollada tras escanearse(izquierda), tras la conexión (centro), mientras se establece la fecha (derecha).

3.3.Etapas del proyecto

Tras explicar el funcionamiento de la consola UART BLE común a todas las etapas, se profundiza el trabajo realizado en cada una de ellas. A nivel de hardware en todas las etapas se utiliza como sensor el acelerómetro combinado con el giroscopio, ya que durante el estudio del estado del arte se hacen menciones muy positivas del desempeño de estos sensores inerciales en el reconocimiento de actividades animales. Además, en cada etapa se hace uso de módulos BLE para disponer de la consola de comandos UART BLE.

3.3.1. Etapa de validación de hipótesis iniciales

En la etapa de validación de hipótesis iniciales se busca comprobar experimentalmente y verificar los resultados obtenidos por los autores citados en el

estado del arte. En esta etapa, se prioriza un desarrollo simple y rápido del sistema de adquisición.

Para el rápido desarrollo del hardware se trabaja con un microcontrolador ESP-Wroom32 [36], ya que integra dentro del propio microcontrolador el periférico de BLE además de uno de WiFi, con los que se puede interactuar con facilidad gracias a las capas de software que ofrece Arduino [35]. La placa de desarrollo en la que se encuentra integrado este microcontrolador es la TTGO-T2 [32] (ver Figura 8: Placa de desarrollo TTGO-T2) que integra una tarjeta SD para el almacenamiento de los datos. Como sensor acelerómetro y giroscopio se utiliza el sensor MPU 9250 que incluye ambos.

.

Figura 8: Placa de desarrollo TTGO-T2.

Para el desarrollo del firmware se ha trabajado en el entorno PlatformIO [34], el cual permite programar el microcontrolador utilizando tanto el software que suministra el fabricante *Espressif,* como el de Arduino. El firmware trabaja con un sistema operativo de tiempo real llamado FreeRTOS y consta de 5 tareas: “Leer sensores”, “Procesado de los datos”, “Guardar en SD”, “Levantar_server_ftp” y “Tarea Aplicación”.

La “Tarea Aplicación” gestiona una máquina de estados que suspende y activa el resto de las tareas dependiendo de este. Estos estados pueden ser: “ESPERA_COMANDOS”, “TOMANDO_DATOS”, “EXTRACCIÓN” y “DORMIDO”.
Para información más detallada de las tareas y los estados del sistema consultar Anexo III y Anexo IV.

En esta etapa se realiza un estudio teórico (Anexo VAnexo) del consumo que tenía la plataforma muestreando a una frecuencia de 50Hz. Presenta unos consumos algo elevados, alrededor de 2.3155mJ/ciclo de 20ms equivalente a 416J/h, para afrontar sesiones de toma de datos de larga duración. Como las sesiones que se realizan en esta etapa tienen una duración inferior a un día, se dimensiona la batería para tener una autonomía de 48 horas.

3.3.2. Etapa de validación de funcionalidades

En la etapa de validación de funcionalidades se trabaja con microcontroladores de ST [43]. Estos microcontroladores presentan modos de consumo muy reducido y con

la memoria suficiente para almacenar algoritmos clasificadores de actividades. La implementación realizada en esta etapa pretendía ser la definitiva; sin embargo, problemas con el tamaño de la placa de desarrollo, obligará a tener que plantear nuevas etapas.

La placa de desarrollo utilizada es la STEVAL-STWINKT1B (Ver Figura 9). Esta placa contiene un microcontrolador STM32L4R9ZIJ6 que integra un procesador Córtex ARM M4 de 32 bits.

La placa contiene los sensores inerciales de acelerómetro, giroscopio y magnetómetros conectados al microcontrolador mediante SPI (*Serial Peripheral Interface*) de 4 cables, cuenta con un módulo de comunicaciones BLE (SPBTLE-1S) y la capacidad para almacenar una tarjeta SD.

Figura 9: Placa de desarrollo STEVAL-STWINKT1B.

El proceso de elaboración del firmware en esta etapa tiene una mayor complejidad respecto al desarrollado en Arduino en la etapa anterior. En esta etapa se trabaja en C# haciendo uso de paquetes de software (ver Anexo VI) más optimizados y que requieren de unas mayores habilidades informáticas. El firmware desarrollado en esta etapa también utiliza un sistema operativo en tiempo real llamado CMSIS-RTOSv2.0 encargado de la gestionar tareas como la toma y el procesado de los datos de los sensores inerciales, la gestión de la consola BLE y el guardado en la tarjeta SD de los datos adquiridos. El firmware a nivel de tareas es muy similar al de la etapa de versión 1.0 en la que se explicará este punto en más detalle.

3.3.3. Etapa de optimización

En la etapa de optimización se trabajan aspectos relacionados con la disminución del tamaño de la placa utilizada en la etapa anterior para que tenga cabida en los nuevos collares de trabajo adquiridos por la facultad de veterinaria.

En este momento del proyecto, se plantea la opción de desarrollar electrónica propia con los mismos componentes de la placa utilizada en la etapa anterior, (lo que obligaría a destinar parte del tiempo en el aprendizaje de diseño de PCBs y a validar el hardware), o adquirir del mercado nuevas placas de desarrollo de menor tamaño que requieren modificar el firmware de la etapa anterior. Esta disyuntiva se soluciona finalmente con la adquisición de placas más pequeñas y la redefinición de parte del código de la etapa anterior.

La placa de desarrollo seleccionada es la Sensor-Tile (STEVAL-STLCS01V1) con los escudos STLCX01V1 y STLCR01V1 (ver Figura 10). Esta placa cuenta con sensores de acelerómetro - giroscopio (lsm6dsm) y magnetómetro (lsm303agr) con los que el microcontrolador se comunica a través de comunicación SPI de 3 cables, en lugar del SPI de 4 cables como sucedía en la etapa anterior. Esta placa también integra un módulo de BLE (BlueNRG-MS) para poder ejecutar la consola UART BLE y permite instalar una tarjeta SD.

Figura 10: Placa STEVAL-STLCS01V1 con los escudos STLCX01V1 y STLCR01V1.

En esta etapa se redefine parte del firmware desarrollado durante la etapa de validación de funcionalidades, teniendo que modificar algunas capas de software, como las BSP (*Board Support Package*), para gestionar la comunicación de los sensores por SPI de 3 cables en lugar de 4 cables. En esta etapa también se trabaja con sistema operativo, donde las tareas utilizadas son idénticas a las que se implementan en la etapa de Versión-1 (donde se explicarán), salvo por las tareas de LoRa y GPS que no han sido implementadas.

3.3.3.1. Escaneo de dispositivos BLE

En esta etapa se comienza a trabajar en una estrategia para obtener información de las relaciones existentes dentro del grupo de ovejas, dado que estas interacciones sociales aportan gran información al grupo de veterinaria para extraer conclusiones más elaboradas, sobre las consecuencias del uso del Aluminio como adyuvante.

En concreto, se plantea la hipótesis, medir a lo largo del tiempo, la distancia que existe entre las ovejas del grupo. De manera que aquellas que más tiempo se encuentren cerca una de la otra se presupone que es porque tienen una mayor afinidad. En este trabajo fin de máster no se ha profundizado en demostrar la validez de esta hipótesis sino en la implementación de un sistema para adquirir los datos que permitan en futuras investigaciones determinar la distancia entre ovejas y validar la hipótesis.

El uso BLE para adquirir datos de proximidad entre individuos se basa en que los módulos BLE de cada dispositivo tengan dos modos de funcionamiento. El primer modo, consiste en estar enviando mensajes de *advertising* la mayor parte del tiempo con una potencia de emisión constante y conocida, y el segundo modo consiste escanear los dispositivos BLE cercanos (cada oveja está emitiendo), anotando la potencia con la que reciben estos mensajes de *advertising*. Con la potencia de emisión y la potencia de

recepción y con una buena caracterización de la atenuación de la señal en el medio se plantea estimar la distancia entre los individuos. Ver Figura 11.

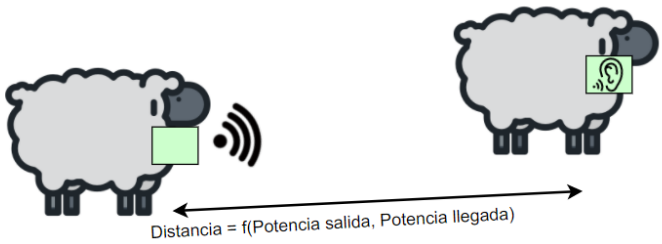


Figura 11: Representación gráfica del uso del escaneo para determinar ovejas cercanas.

La implementación ha consistido en lo siguiente: Inicialmente todos los sistemas de telemetría disponen de módulo BLE y se encuentran mandando mensajes de *advertising* cada 1.28s - 2.56s, emitiendo con una potencia de -2dBm.

Cada 5 minutos, el módulo BLE de las ovejas deja de comportarse como módulo "periférico" que hace *advertising*, para pasar a comportarse como modo "central" pudiendo escanear los dispositivos cercanos. Por lo tanto, cada 5 minutos, un equipo escanea a los dispositivos vecinos y anota cuales son los que tiene cerca y almacena esta información dentro de la tarjeta SD se anota la MAC *Address* del dispositivo cercano y el RSSI (*Received Signal Strength Indicator*). Gracias a la información del RSSI y a la potencia de envío, se puede determinar de forma aproximada si un animal está más cerca de un animal o de otro. Este proceso de escaneado dura 3 segundos tras los cuales, el dispositivo recupera su estado anterior de periférico y continúa haciendo *advertising*.

3.3.4. Etapa de versión 1.0

La etapa de versión 1.0 surge en respuesta al correcto desempeño del proyecto que hace que el equipo de veterinarios decida realizar una prueba con 30 ovejas. Al aumentar considerablemente el número de dispositivos necesarios, se decide invertir tiempo en el diseño de electrónica propia. Además, dado que se plantea la posibilidad de realizar pruebas en campo abierto se integra y trabaja con dos nuevos módulos, como son el sensor GPS y el módulo LoRa.

A nivel de hardware, en esta etapa se hace un diseño de PCB que debe contar con sensores inerciales (acelerómetro y giroscopio), módulo BLE, módulo LoRa, tarjeta SD, GPS y un sensor Hall que en presencia de campo magnético reinicie el microcontrolador. Con estos requerimientos, se decide de utilizar la placa sensor-Tile utilizada en la etapa previa como núcleo central de la electrónica, ya que incluye los sensores todos los sensores inerciales necesarios y el módulo BLE. De esta forma, se puede decir que realmente la PCB es un escudo para la placa sensor tile que extiende sus funcionalidades. En este escudo se integra un módulo GPS (L76L-M33), un módulo LoRa (RAK4270) y el conector de SD (DM3D-SF).

Para el diseño de la PCB se utilizó el programa Altium y las librerías de Howlab. Ver Figura 12. Este fue soldado a mano mediante un soldador y una pistola de calor, excepto el módulo GPS y LoRa que, al no existir stock de este, no pudo adquirirse,

careciendo la placa de estos módulos. Del GPS, se tenía un módulo externo con el que sí que se pudo trabajar. Los esquemáticos se encuentran en el Anexo VII.

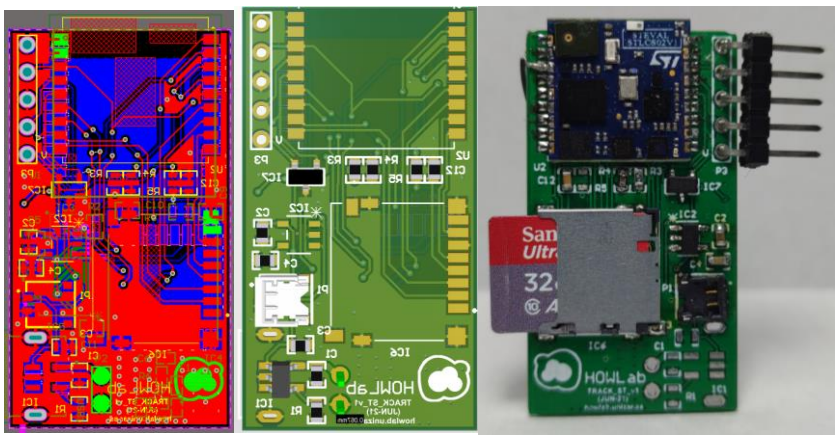


Figura 12: Dos Imagen de la PCB vistas desde Altium (imágenes de la izquierda) y PCB ya soldada.

Por lo tanto, los sensores utilizados en esta etapa final, al igual que en la anterior, han sido el sensor LSM6DSM que integra un acelerómetro que configurado para medir ±2g y un giroscopio que puede medir 2000 grados/s y el magnetómetro LSM303AGR, que trabaja ±2Gauss.

La parte de alimentación de la placa desarrollada (ver Anexo VII) incluye un regulador lineal de bajo consumo (MIC5207-3.2YM5-TR), para alimentar a los módulos LoRa, GPS y SD a 3v3. Un conversor DCDC (TPS82740ASIPT) para alimentar al microcontrolador a 1v8 desde una batería de 3v7. Por otro lado, se añade un circuito integrado llamado *level shifter* que convierte las señales de la UAR, que salen del microcontrolador a 1V8, al voltaje de los módulos LoRa, GPS y SD que funcionan a 3v3.

El nivel del firmware, el desarrollo con esta tarjeta es similar al desarrollado en la etapa anterior, ya que a se mantiene gran parte de la estructura de la placa anterior, por lo que la interacción entre microcontrolador y el resto de los componentes es la misma; a excepción de los nuevos módulos de GPS y LoRa con los que se debe interactuar.

La programación se realizó en lenguaje C# y se utilizaron los mismos paquetes del Anexo VI, teniendo que añadir el paquete RAKWireless_RUI_V2.0 [*37*] para la gestión del módulo LoRa y la librería L76X [*38*] para el control del GPS.

El firmware consta de 3 estados principales: (Tomando datos, Streaming, Espera de comandos y “Dormido) y aunque no se haya utilizado en las sesiones de toma de datos, se está trabajando en un cuarto modo, llamado modo extracción, que permite obtener un fichero guardado en la tarjeta SD, vía BLE, lo cual facilitaría el proceso de extracción de los datos sin la necesidad de retirar el collar al animal. Este modo sería similar al implementado sobre WiFi por medio de un servidor FTP (*File Transfer Protocol*) en la etapa 1- validación de hipótesis iniciales.

Los 4 estados que tiene el microcontrolador se pueden ver en la Figura 13.

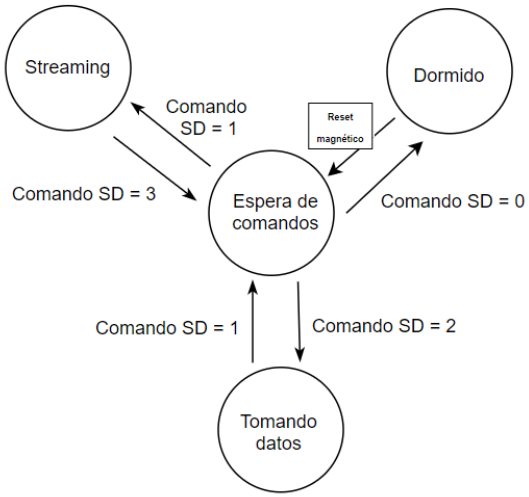


Figura 13: Estados del sistema de adquisición.

Modo Espera de comandos: este modo es utilizado para establecer una comunicación con el microcontrolador y definir los parámetros de configuración, conocer diferentes valores de estatus del microcontrolador o modificar su estado. Esta comunicación se realiza por medio de la consola UART BLE comentada previamente. Mientras se mantiene en este modo, por la característica RX se envía el tiempo en milisegundos que el microcontrolador lleva encendido (*timestamp*), para posteriormente poder hacer una sincronización correcta de los datos con el vídeo de la sesión.

Modo Dormido: es el estado en el que permanecen el sistema de adquisición de datos cuando no está parado. En este modo, el consumo es mínimo. Para activar el dispositivo es necesario acercar un imán por encima de la carcasa pasando al estado "Espera de comandos". Para volver a llevarlo a este modo "Dormido" será necesario enviar un comando a través de la consola BLE.

Modo Tomando datos: este modo es utilizado para adquirir los datos crudos del GPS, acelerómetro, giroscopio y magnetómetro que posteriormente serán procesados y enviados por BLE o guardados en la SD. Siempre que se inicia este modo se crean una carpeta con nombre "dd_mm_yy_xxxx_sen" (dd_mm_yy son día, mes y año), dentro de esta carpeta se generan tres archivos "hh_mm_ss_sens.csv" (hh_mm_ss son hora minuto y segundo), hh_mm_ss_ble.csv y hh_mm_ss_gps.csv y se reinicia la comunicación con la tarjeta SD en caso de que existiera algún error a nivel de FatFs (*FAT Filesystem*).

Modo Streaming: este modo funciona de forma idéntica al modo de Tomando_datos, con la salvedad de que los datos en lugar de ser guardados en una tarjeta de almacenamiento son enviados periódicamente por BLE.

3.3.4.1 Firmware Tareas

En esta etapa también se trabaja con CMSIS-RTOSv2.0 y la estructura del código se divide en 7 tareas. Estas tareas son: “maquina_estados”, “leer_sensores”, “procesar_datos”, “gestionar_comandos_ble”, “guardar_sd, leer_LoRa”, “leer_GPS”, “enviar_BLE” y Escaneo. La interacción entre estas tareas es por medio de semáforos y colas.

La tarea de mayor prioridad es la tarea “**gestionar_comandos_BLE”,** que es la encargada de gestionar cualquier tipo de interacción que haga el usuario a través de la consola BLE. Esta tarea sólo se inicia en caso de que suceda un evento de BLE. Ver Figura *14*.

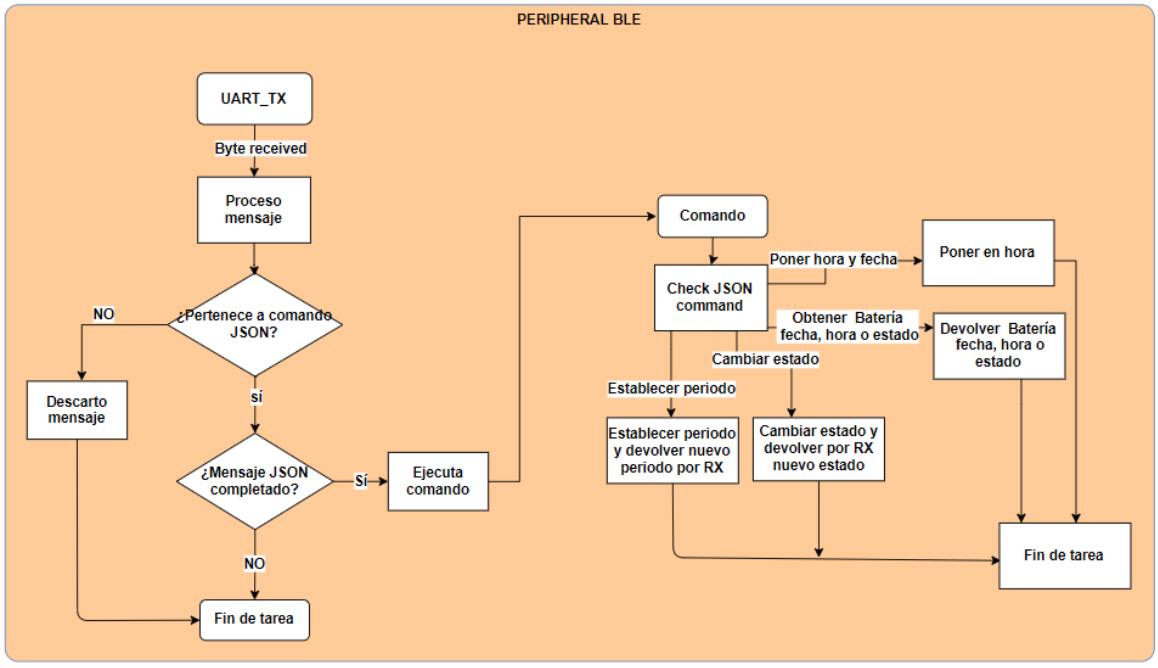


Figura 14: Diagrama Tarea gestionar comandos BLE.

La siguiente tarea de mayor prioridad es la “**máquina_estados”,** que determina en qué estado se encuentra el microcontrolador y gestiona qué tareas son las que deben permanecer activas y cuales suspendidas. Ver Figura *15*.

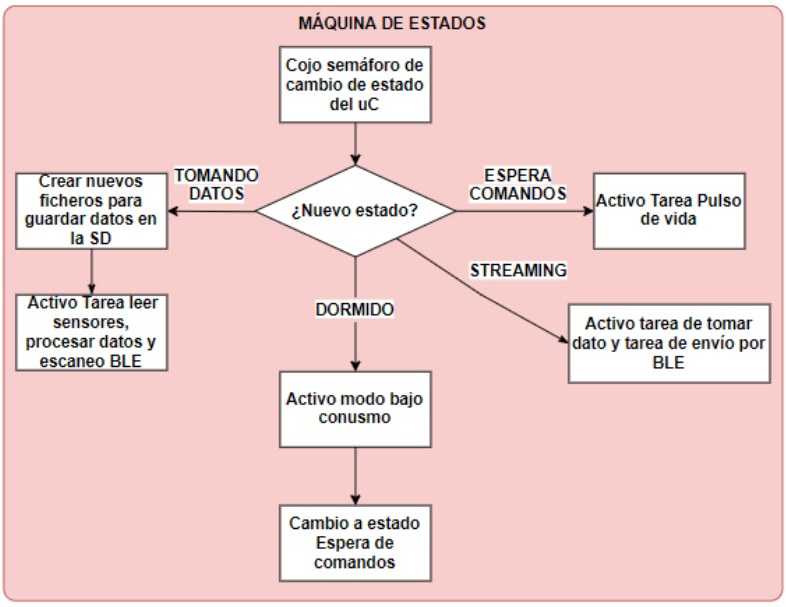


Figura 15: Diagrama tarea Máquina de estados.

La tarea de "**leer_sensores"**, se encarga de recoger la información de los sensores acelerómetro, giroscopio, magnetómetro y el *timestamp*, y ponerlos en una cola. Esta tarea sólo está activada en caso de que el estado del microcontrolador se encuentre en modo de "Tomando datos" o "Streaming". Ver Figura *16*.

La tarea de **"procesar_datos"** se encarga de extraer de la cola, en caso de que hubiera algún elemento en ella, la información de los sensores y el momento de tiempo en el que se adquirieron las muestras y procesa los datos. Este procesado se realiza mediante un vector circular y deja el dato resultante en una cola, liberando un semáforo para que la tarea de guardar en la SD se desbloque y se lleve a cabo. Ver Figura *16*.

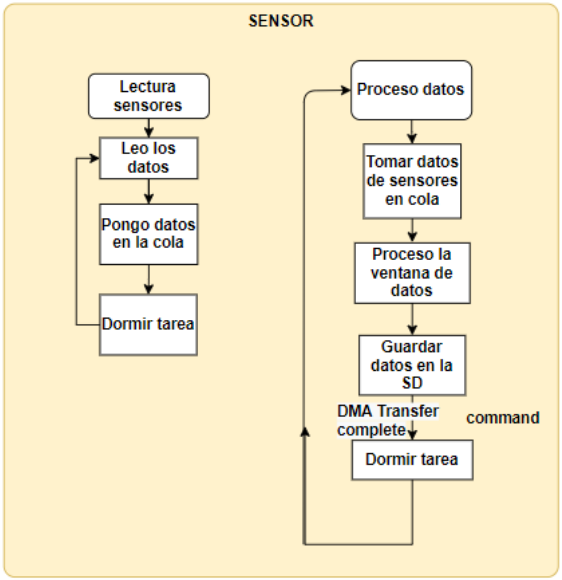


Figura 16: Diagrama tareas tomar y procesar datos.

La tarea de **“guardar_sd”** se encarga de guardar la información obtenida de la cola de datos en la tarjeta SD. La tarea de guardar en SD se mantiene bloqueada por un semáforo que es liberado por las tareas cuando hay un dato nuevo. Ver Figura *17*.

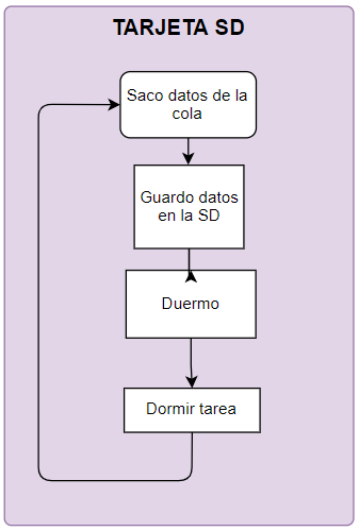


Figura 17: Diagrama tarea guardar en SD.

La tarea **“enviar_BLE”** es la encargada de enviar periódicamente los datos procesados a por la consola BLE durante el modo de “Streaming”. Durante el modo de “Espera de comandos” esta tarea también se encuentra levantada, pero enviando el timestamp.

La tarea de **“leer_LoRa”** es la encargada de levantarse y vaciar el buffer de datos que se le haya delegado para enviar en cada periodo de muestreo. Esta tarea se mantiene bloqueada mientras no llegue su periodo de muestreo o no haya datos para enviar. Ver Figura *18*. Al no haber podido adquirirse este módulo, la tarea actualemtente se mantiene en modo bloqueada.

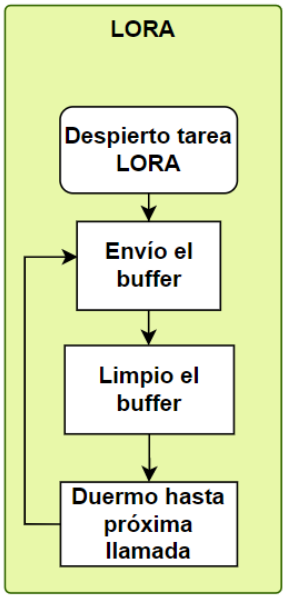


Figura 18: Diagrama tarea enviar LoRa.

La tarea **“leer_GPS”** tiene como objetivo llevar a cabo una lectura del módulo GPS. Extraer la información de este mensaje y en caso de que sea una trama $GNGGA analizar el parámetro llamado dilución horizontal. En caso de tener un valor aceptable o la conexión sea buena, guardar la información de las coordenadas en formato definido por Google Maps. Si la dilución horizontal no es lo suficientemente buena, la tarea se bloquea para reintentar tomar una nueva medida posteriormente. Ver Figura *19*.

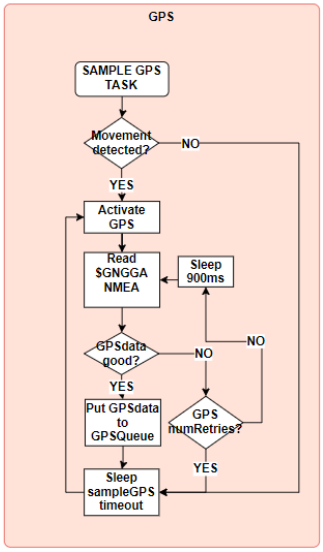


Figura 19: Diagrama tarea leer GPS.

La tarea **“Escaneo”** se activa periódicamente (cada 5 minutos) durante el modo de “*Tomando_datos*” o de “*Streaming*”, cerrando todas las conexiones existentes hasta el momento, parando el *advertising* y relanzando el módulo BLE para escanear durante 3 segundos los dispositivos BLE cercanos y guardando información del RSSI, de la MAC y del mensaje de *advertising* (este último dato puede desactivarse si se desea mediante directiva de preprocesador). Ver Figura *20*.

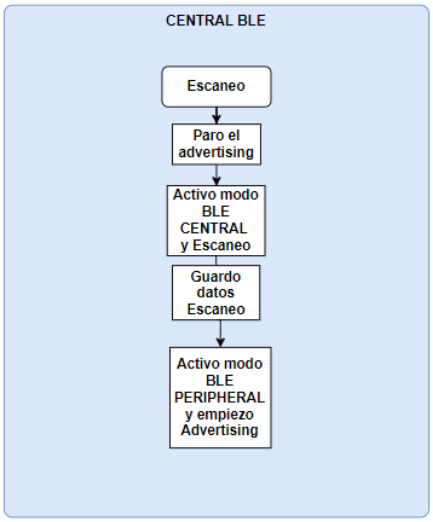


Figura 20: Diagrama tarea Escanear BLE.

Durante todo este tiempo la consola UART BLE se mantiene activa para gestionar la recepción de cualquier comando o gestionar eventos de conexión, desconexión y nuevas instrucciones.

3.3.4.2Análisis del microcontrolador y modos de bajo consumo

Para que un dispositivo alimentado a baterías tenga una autonomía aceptable, es necesario trabajar e implementar estrategias de bajo consumo. En esta última etapa, se ha trabajado para conseguir un consumo muy reducido en el estado "Dormido" del dispositivo, el cual es el estado de menor consumo. En este estado hay que dejar todos los sensores en modo de *power off,* el microcontrolador en modo *shutdown* y los reguladores lineales en modo de bajo consumo.
EL consumo mínimo logrado en este estado ha sido de alrededor de 29-30uA. El estudio teórico, así como la medida experimental que verifica este bajo consumo puede observarse en el Anexo X.

Además, en el Anexo X se explica el procedimiento establecido para caracterizar los consumos que tienen los sensores de la placa, el módulo BLE y el microcontrolador durante un ciclo de prueba (ver Figura 21), pudiendo ser aprovechados en trabajos futuros.

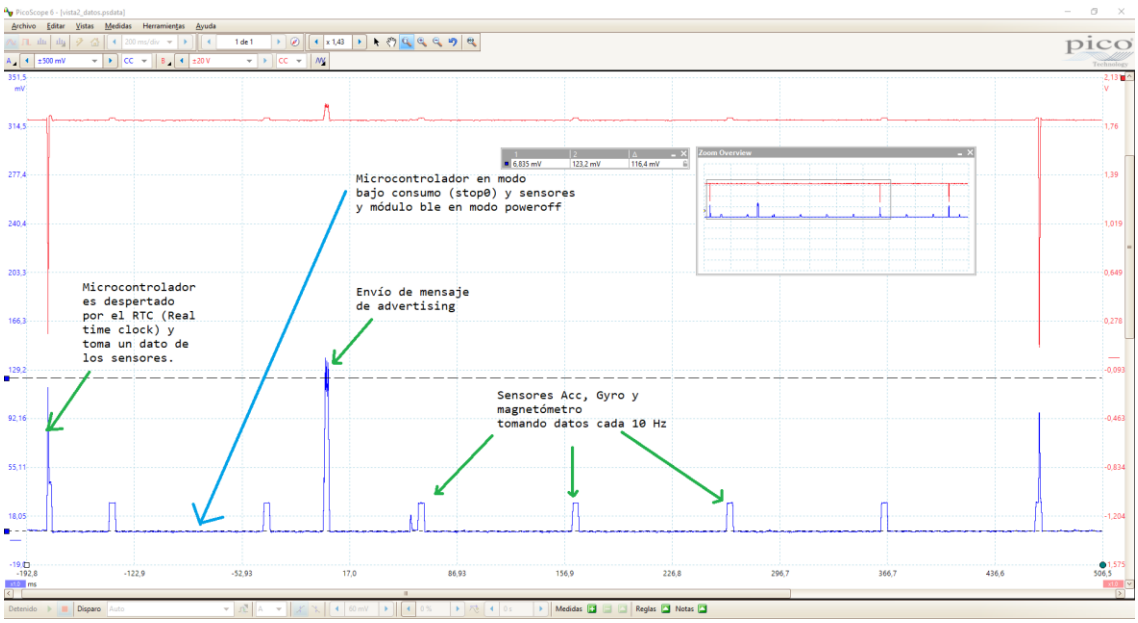


Figura 21. Ciclo de prueba para ver el funcionamiento del consumo del dispositivo. Los sensores acelerómetro, giroscopio y magnetómetro muestreados a 10Hz, BLE emite cada 900ms y el microcontrolador se despierta por el RTC (Real Time Clock) para tomar datos de los sensores

Capítulo 4

Sesiones de adquisición y creación de una base de datos

4.1. Introducción

En este capítulo se detalla el procedimiento seguido para la creación de la base de datos, se define el etograma de trabajo, se describe el entorno de desarrollo de las pruebas, se listan los materiales necesarios para las sesiones de adquisición de datos y se explican el resto de los procedimientos establecidos para evitar que exista mucha diferencia entre estas sesiones de toma de datos y la futura prueba real con las ovejas.

4.2. Definición de etograma

La definición del etograma parte de los investigadores de la Facultad de veterinaria ya que son los que tienen el conocimiento y la experiencia de qué tipo de actividades son las que más variación podían sufrir previo a aplicar la vacuna y posteriormente.

Estas actividades fueron clasificadas en dos grupos: actividades individuales y actividades sociales. Dentro de las actividades individuales se distinguieron las categorías de andando, corriendo, comiendo y parada. Dentro de las actividades sociales, se distinguen a su vez dos clases, las actividades afiliativas (descansar juntas, frotarse o montar) y las actividades agresivas (toparse).

Como primer acercamiento en el reconocimiento de actividades, en este trabajo fin de máster, se procesan los datos para tratar de reconocer las actividades individuales y se trabaja en adquirir datos de proximidad entre animales para posteriormente tratar de relacionarlos unos con otros. En la Tabla 7 se muestra el etograma con las actividades a reconocer.

Categorías de comportamiento	Definición
Andando	Oveja moviendo las 4 patas a un ritmo lento.
Corriendo	Oveja moviendo las 4 patas a un ritmo ágil.
Comiendo	Oveja rumiando o engulliendo.
Parada	Oveja sin movimiento tanto tumbada como parada de pie a 4 patas.

Tabla 7: Etograma de trabajo.

4.3. Materiales necesarios

El material necesario para realizar las sesiones de medición es el siguiente:

- Establo amplio para ovejas.
- Trípode para cámara.
- Cámara con resolución 1080p y capacidad de 2 horas de grabación.

- Tarjetas SD: 1 de 32GB para la cámara y 1 de 16GB por cada equipo de telemetría.
- Collares: 1 por cada animal.
- Carcasas para la electrónica: 1 por dispositivo con tapa y contratapa.
- Móvil con aplicación propia para enviar comandos BLE.
- Baterías de 2Ah: 1 por cada sensor.
- Equipo de telemetría: 1 placa de desarrollo con módulo BLE y sensores inerciales para cada oveja.

4.4. Frecuencia de muestreo

La frecuencia de muestreo, aunque es configurable a través de la consola BLE, se ha fijado a 100 Hz para todas las sesiones de adquisición. Esta frecuencia de muestreo, aunque elevada según varios artículos antes mencionados, permite hacer diferentes pruebas durante el procesado posterior, como aplicar filtros frecuenciales y determinar la mejor frecuencia de muestreo para esta población de ovejas o permitir hacer *data Augmentation* creando una segunda base de datos a partir de la primera.

4.5. Protocolo

Con objeto de estandarizar todos los procedimientos y que no exista una gran diferencia entre las condiciones de las sesiones de toma de datos y las que se darán durante el tiempo final de estudio, se han estandarizado raza de animal con el que se va a trabajar, las horas a las que se toman las muestras, el lugar de las pruebas, el collar a utilizar y la ubicación de la cámara dentro del corral.

4.5.1. Selección de animales

Los animales con los que se trabaja son ovejas macho de la raza “Rasa Aragonesa la cual es una especie autóctona de Aragón. Esta oveja es rústica (adaptada a la escasez de comida), está muy bien adaptada al frío y al calor, es gregaria y sus estructuras sociales son complejas (estructura de dominancia habitualmente no es lineal).

El grupo de trabajo estaba formado por un grupo de 3 a 11 individuos macho de aproximadamente 30-35 kg de peso, en edad adulta nacidos a principios de enero del 2021, los cuales ya han sido previamente castrados con lo que se reducen sus necesidades reproductivas y por lo tanto se reducen la aparición de comportamientos como el toparse o el montarse para mostrar dominancia entre ellos.

4.5.2. Horas del día a las que se toman muestras

La hora elegida para hacer las mediciones han sido por la mañana, entre las 11:00 – 13:00 de la mañana (hora española GPTM+2) y por la tarde entre las 16.00-18.00 pm. Estas horas han sido seleccionadas buscando diferentes estados de activación del animal, algo más agitado en las horas de la mañana tras haberse alimentado en grupo sobre las 7:30-8:30 Am y ya haber tenido algún tiempo para rumiar, y algo más relajado por la tarde tras haber comido y empezar la rumia. De hecho, en algunas sesiones de la

tarde tras abandonar el establo para dejar a los animales solos, las ovejas han pasado cerca de 1 hora descansando tumbadas al lado de la pared, con lo que se ha podido caracterizar relativamente bien la actividad de parada.

4.5.3. Entorno de trabajo

El lugar de realización de los ensayos han sido dos corrales de la Facultad de Veterinaria de Zaragoza. Uno de mayor tamaño (corral 003, ver Figura 22) que contaba con 6 individuos y otro de menor tamaño que contaba con 3 individuos.

Figura 22: Imagen de la puerta exterior del corral 003 e imágenes interiores de los dos corrales de trabajo.

Estos lugares reducidos son muy adecuados para el reconocimiento de actividades como son “andar”, “comer” o estar “parada”, pero presentan el problema de que ciertas actividades no se darán en gran medida como; por ejemplo, “correr”.

4.5.4. Dispositivos de medida

Los equipos de telemetría son encapsulados en unas carcasas con protección ante el agua hechas por medio de impresión 3D (Ver Figura 23). Estas cajas contienen capacidad para un microcontrolador y una batería y van insertadas en el collar que portan las ovejas.

El peso de la carcasa, con la electrónica y el collar es de aproximadamente 150 gramos con lo que se cumple que el peso es inferior al 3% del peso del animal (30 y 35 kg).

Figura 23: Imagen de las carcasas para los collares iniciales (rosa) y carcasas para el collar final (blanco).

4.5.5. Cámara y ubicación

Durante todas las tomas de datos, se dejó una cámara filmando el establo en el que se encontraban las ovejas. El uso de la cámara tenía por objetivo el abandonar el establo mientras se grababa a los animales para evitar perturbar con nuestra presencia el comportamiento habitual de estos.

Esta cámara se eleva en el suelo a una altura de 1.5m gracias a un trípode y se protege de los animales por medio de un comedero que les impide el paso (Ver Figura 24). En alguna ocasión, se utilizó una valla como protección y una mesa para conseguir que la cámara se encontrará a una altura mayor, pudiendo capturar de mejor forma el establo.

Figura 24: Imágenes de la ubicación de la cámara en el lugar de trabajo.

Uno de los problemas que se constató fue que durante los primeros 15-20 minutos, los animales se sienten atraídos por el trípode con la cámara, por lo que tienden a dejar lo que estaban haciendo y pasan a concentrarse por completo en la cámara, es por esto por lo que en las últimas sesiones se dejaba la cámara durante 15 minutos en la sala con las ovejas sin grabar, para acostumbrarse al nuevo objeto y proceder posteriormente con la grabación.

4.5.6. Problemas durante las sesiones

Los principales problemas existentes durante las sesiones de adquisición, en consideración en las sesiones posteriores, han sido la falta de tiempo para acostumbrarse, la posición de la cámara, actividades poco frecuentes y los periodos de baja actividad. Estos problemas se encuentran recogidos en el Anexo IX.

4.6. Archivos generados por cada dispositivo

Cada oveja lleva consigo un equipo de telemetría que por cada sesión genera 3 archivos en formato .csv (*comma separated values*): el archivo de datos de los sensores inerciales, el de dispositivos BLE cercanos y el de GPS. En el archivo de datos de sensores inerciales con nombre "hh_mm_ss_sens.csv", (hh_mm_ss es la hora, minutos y segundos a la que se generó el archivo), se encuentran los datos de los sensores acelerómetro, giroscopio y magnetómetro, la hora y el número de milisegundos desde que se encendió el microcontrolador en el momento de adquirir el dato (esto se denomina *Timestamp*).

En el archivo de dispositivos BLE cercanos "hh_mm_ss_ble.csv", se encontrará, juntamente con el tiempo en milisegundos en el que se encendió el microcontrolador, los datos registrados durante el proceso de escaneo de dispositivos BLE cercanos. Como se ha comentado antes estos serán RSSI, MAC Address y el tipo del mensaje.

En el archivo de GPS (hh_mm_ss_gps.csv) se ha de guardar el tiempo en milisegundos en el que se encendió el microcontrolador y el mensaje GPS recibido según el formato determinado en el protocolo NMEA-0183 (*National Electrical Manufacturers Association*). De todas las sentencias NMEA únicamente se van a registrar las referentes a --RMC y a --GGA. De la sentencia --RMC la información relevante son coordenadas universales, la hora, la fecha en GMT+0 y si el dato es válido o hay advertencia sobre esa medida y de la sentencia –GGA esta es la dilución horizontal de la posición y la altitud en metros respecto al nivel del mar.

Estos tres archivos se encuentran dentro de una nueva carpeta generada automáticamente cada vez que se activa el modo de "toma de datos" o del modo "streaming", con el nombre "hh_mm_ss_dd_mm_yy". Ver Figura 25.

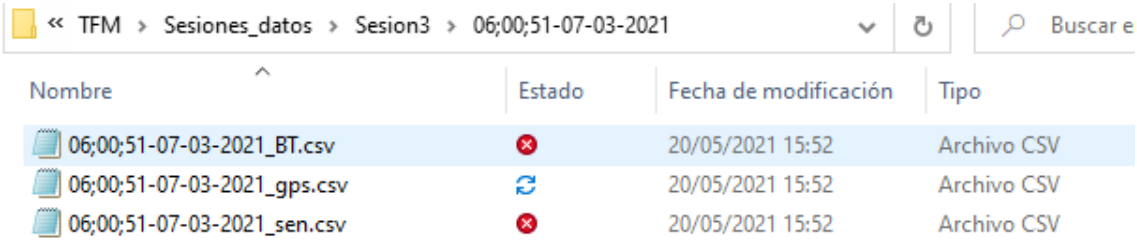


Figura 25: Captura de pantalla de los 3 archivos generados por cada dispositivo tras la adquisición de los datos.

4.7. Base de datos

La base de datos elaborada se encuentra en fase de crecimiento y ya almacena la información de 5 sesiones de adquisición. Estas sesiones suponen tener un total de aproximadamente 10 horas de vídeo y todos los datos en crudo que genera cada collar puesto a cada oveja.

De cada sesión de adquisición se ha extraído una cierta más o menos cantidad de información, dependiendo de si ha existido algún problema durante la misma. En concreto, de la sesión 1, se lleva un collar y se almacena satisfactoriamente toda la información correctamente. En la sesión 2 se llevan 3 collares, por problemas en el *firmware* sólo se puede añadir a la base de datos uno de los collares. Durante la sesión 3 y 4 se llevan dos collares y ambos dos recogen bien la información, pero existen algunos problemas para sincronizar datos con vídeo y salvo uno de los collares, los otros dos no pueden ser etiquetados. Finalmente, en la sesión 5, se llevan dos collares y se utiliza un triple sistema de sincronización (incluye la hora a la que se toma cada dato, la característica BLE y la técnica del aplauso), con lo que se consigue un ajuste fino entre vídeos y datos, pudiendo etiquetarse sin dificultad.

Además de todos los datos en crudo también se proporcionan 5 horas de datos etiquetados manualmente que pueden utilizarse para futuros problemas de ovejas. Agrupando todos los datos, la base de datos consta aproximadamente un total de 464401 datos ya etiquetados. La distribución de esta base de datos es: la actividad “Parada” supone un 69.38% de la base de datos (322193 datos), “Comiendo” supone un 25.77% (118780 datos), “Andando” supone un 3.78% (17565 datos) y “Corriendo” supone un 1.26% (5863 datos).

Como puede observarse, las clases predominantes son “parada” y “comiendo” y la más reducida es “corriendo”, lo cual parecer tener sentido pues el espacio en el que se encuentran estas ovejas es reducido.

Capítulo 5

Procesamiento de los datos

5.0. Introducción

En este capítulo se detalla el procesado de los datos adquiridos. Se describe el proceso de etiquetado, la generación de gráficos, el normalizado, los filtros frecuenciales y el enventanado de los mismos y la extracción de características de cada ventana.

5.1. Proceso de etiquetado

El proceso de etiquetado de datos es una tarea demandante que exige observar a los animales durante horas, mientras se anota las actividades que realizan. Esto se puede hacer estando cerca de los animales mientras se anota a papel la actividad que realizan o grabando en vídeo a los animales y analizando posteriormente. Para esta investigación se hace uso de una cámara que graba en formato MP4 con resolución HD (1280 x 720) a 25 *frames*/s. Una vez grabadas todas las sesiones de varias horas de duración, se procede a un visualizado del vídeo.

En las primeras sesiones, el etiquetado de los datos se realizó manualmente; es decir, se relaciona cada dato con una etiqueta de actividad. Esto supone que, si se había muestreado a 100Hz y durante un segundo la oveja estaba parada, había que poner 100 veces mediante “copia y pega” la etiqueta de parada a todos los datos pertenecientes a ese segundo. Este proceso es tedioso y lento por lo tanto sólo fue utilizado durante la primera sesión, ya que para las siguientes se creó una interfaz gráfica, mediante la librería Qt Designer v5.15.1 para Python v3.7.9. Esta interfaz gráfica permitía etiquetar tramos de vídeo de golpe, sin tener que hacer nada más que visualizar la grabación e ir pulsando en cada momento el botón con la actividad que se viera en pantalla. Como se puede ver en la Figura 26, la interfaz cuenta con simples comandos de reproducir y pausar vídeo, y la selección de tiempo de inicio y tiempo de final en los que se realiza cierta actividad.

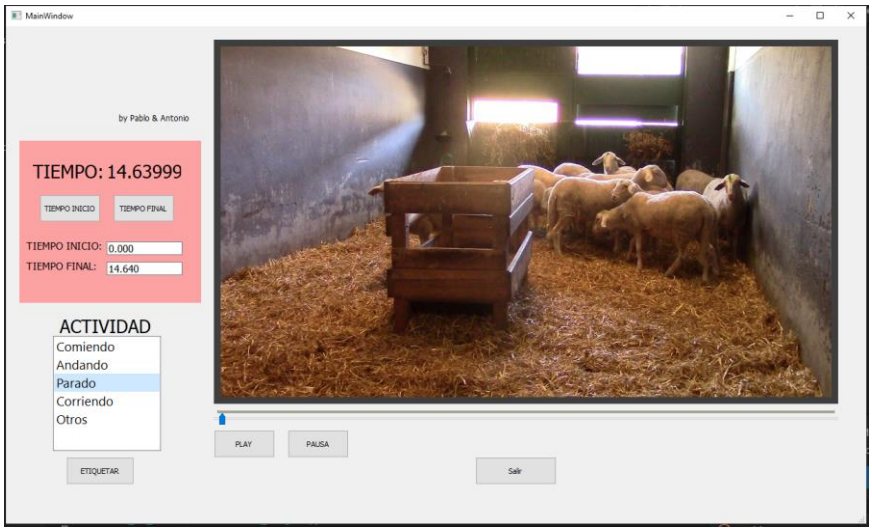


Figura 26: Imagen de la Interfaz gráfica desarrollada durante una sesión de trabajo.

Si se utiliza la interfaz gráfica y se van pulsando los botones de actividad en cada tramo de video, una vez se termina la grabación, la interfaz gráfica genera automáticamente un archivo de salida en formato. tsv (*Tab separated values*), que contendrá el tiempo de inicio y de final en segundos de cada actividad que haya ido apareciendo, como se puede ver en la Figura 27.

fichero_tiempos_201111.txt: Bloc de notas

Archivo Edición Formato Ver Ayuda

```
0.000   **      944.320
944.320 otros   952.800
952.800 andando 955.470
955.470 otros   967.000
967.000 otros   981.360
981.360 andando 982.800
982.800 otros   990.360
990.360 otros   996.760
996.760 otros   1000.040
1000.040        andando 1006.760
1006.760        **      1016.400
1016.400        otros   1033.800
1033.800        otros   1060.720
1060.720        comiendo        1232.440
1232.440        comerSuelo      1239.920
1239.920        otros   1242.480
1242.480        comiendo        1378.800
```

Figura 27: Imagen del primer archivo generado por la interfaz gráfica. La primera columna es el tiempo de inicio, la columna del medio es la columna de actividad y la tercera columna es la de tiempos de fin de actividad.

Con este archivo y la ruta al archivo con los datos en crudo de las ovejas, se genera un último fichero que contiene todos los datos en crudo, pero cada uno está asociado a la actividad que sucedía en ese instante. Ver Figura 28.

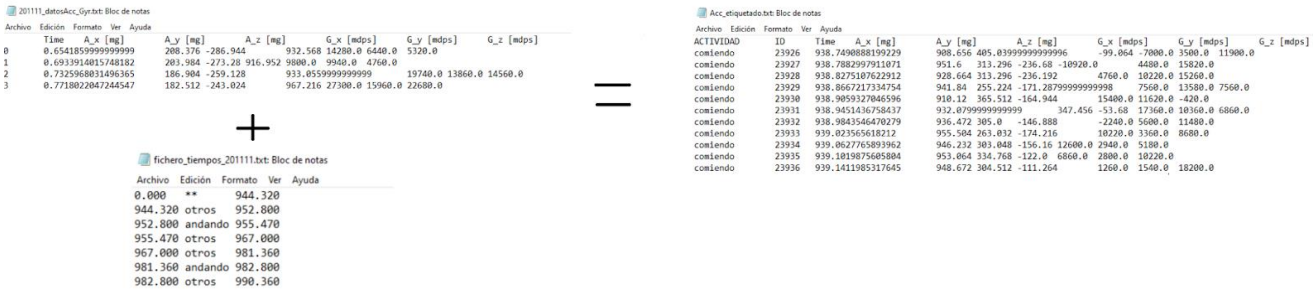


Figura 28: Imagen del resultado que se obtiene al lanzar el script de etiquetar datos. En la parte de la izquierda se encuentra el archivo con los datos en crudo y el que contiene las etiquetas. El resultado de combinarlos aparece a la derecha donde se muestra cada dato asociado a una actividad.

Con esta interfaz, el proceso de etiquetado es acelerado y permite a cualquier investigador de la facultad de veterinaria etiquetar animales de forma rápida y simple.

5.2. Generación de gráficas

La generación de gráficas resulta de vital importancia para visualizar los datos en crudo y las etiquetas de actividad conjuntamente. Esto permite reconocer posibles patrones en los datos que sean muy diferentes entre actividades con las que poder entrenar al posterior algoritmo de clasificación.
Para el graficado, se trabaja inicialmente con dos librerías de Python: Matplotlib v3.3.3 y Plotly v4.14.1 (Ver Figura 29). Aunque ambas tienen mucho potencial, se decide trabajar con Plotly ya que además de ser más intuitivo, también permite seleccionar las curvas que se desean observar y dejar invisibles el resto.

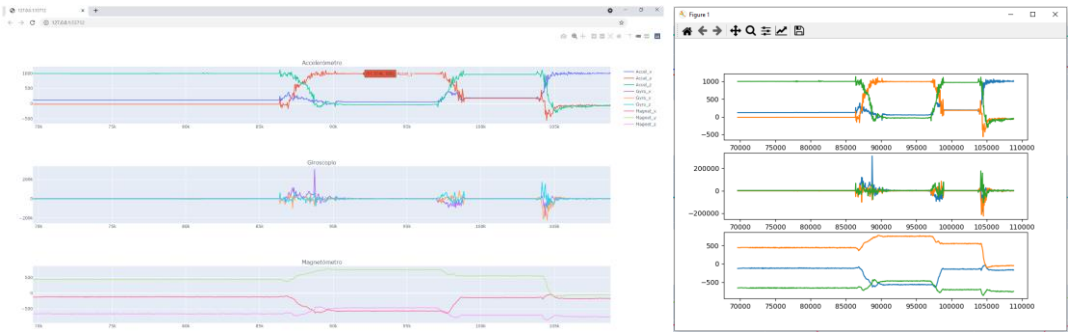


Figura 29: Dos imágenes que representan la misma señal. Una obtenida con plotly (izquierda) y otra con matplotlib (derecha).

Como se ha comentado es de gran interés poder visualizar los datos de los sensores juntamente con la etiqueta de actividad, por lo que se desarrolla un *script* de Python que colorea el fondo de cada señal de datos de un color diferente según la etiqueta de actividad. De esta forma, se pueden analizar posibles patrones en los datos. Ver Figura 30.

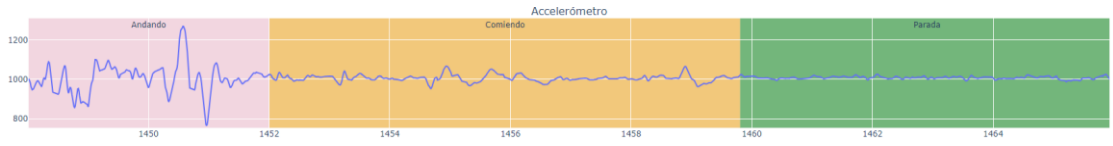


Figura 30: Visualización en plotly de los datos sobre un fondo de cada color según la etiqueta de esos datos. En este caso se representa la actividad de Andando, Comiendo y Parada.

Además de graficar las señales de los sensores inerciales, también se trabaja en la visualización de los datos adquiridos con el sensor GPS. En este caso la herramienta utilizada es Google Maps. El trabajo realizado consistió en familiarizarse con el formato de coordenadas de latitud y longitud que se envían desde los satélites GPS, (formato definido en la NMEA dddmm.mmmm) y transformarlo al formato de coordenadas que reconoce la herramienta Google Maps (formato dd.dddd). Esta conversión de formato también se realizó con un script de Python.

Una vez se tienen las coordenadas transformadas al formato especificado por Google, cada muestra de GPS puede ser graficada. Para que sea más visual, cada muestra se grafica con el nombre de "punto_x", siendo x el número de muestra de GPS y se añade la hora en la que se adquiere la muestra. Ver Figura 31.

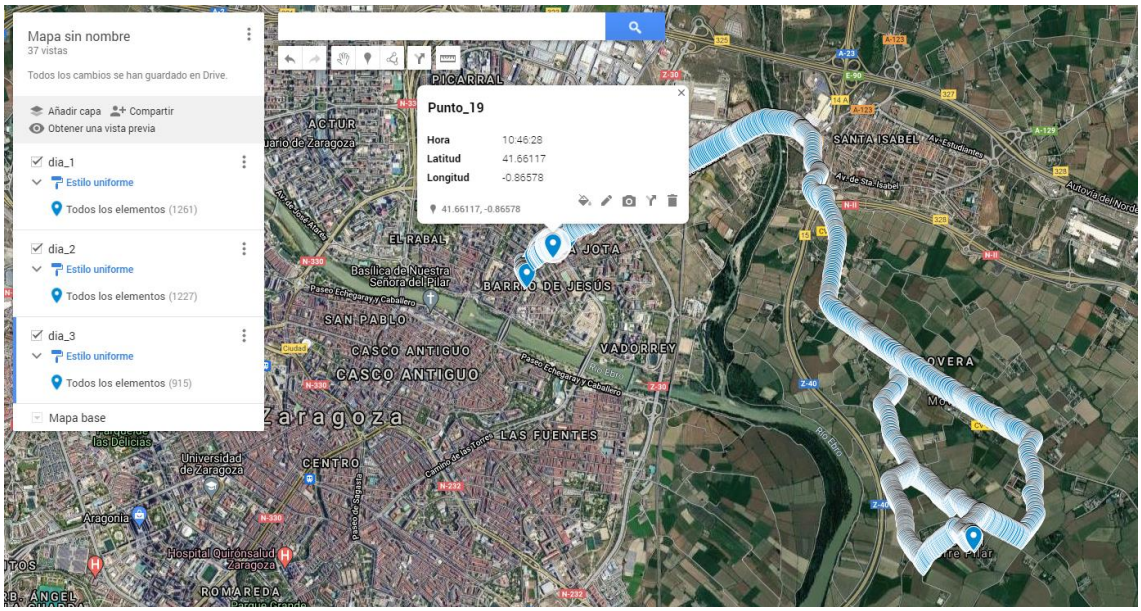


Figura 31: Imagen de la herramienta de Google Maps tras pasarle por parámetro todas las coordenadas GPS obtenidas a lo largo de 3 días de recorrido en coche.

5.3. Sincronización de los datos

Uno de los puntos a resolver a lo largo del proyecto ha sido la sincronización entre los datos de los sensores y el video grabado durante las sesiones, porque cada uno funciona con su propio reloj interno y en caso de no poder relacionarlos supone la pérdida de la información de esa sesión. Con el transcurso del proyecto se han acabado definiendo tres estrategias que permiten una correcta sincronización; estas son: el envío del tiempo que lleva encendido el microcontrolador en milisegundos a través de la consola UART BLE, el guardado de la hora en la que se adquieren los datos y el uso de la técnica "aplauso" al inicio de las grabaciones que se explican a continuación.

5.3.1. Característica BLE

Durante el estado del sistema "Espera de comandos", la característica BLE RX se actualiza periódicamente, indicando el tiempo en milisegundos desde que se encendió el microcontrolador (*timestamp*). Este tiempo es guardado en la SD junto con cada nuevo dato. De forma que, si se graba con la videocámara la pantalla de la aplicación móvil que muestra este valor, posteriormente al visualizar el video, se conoce la correspondencia de un segundo del vídeo que coincide con el *timestamp* de un dato y por lo tanto, se puede inferir qué otros segundos del video corresponde con qué otros datos.

A este valor de diferencia que hay entre el segundo que tiene el video cuando se muestra el *timestamp* en la App, se conoce como *offset* (Ver Figura 32). Gracias a este sistema se consigue sincronizar los datos y el vídeo con un error inferior a 1s (tiempo de refresco de la App móvil). Para más precisión y un ajuste fino se utiliza la técnica Aplauso.

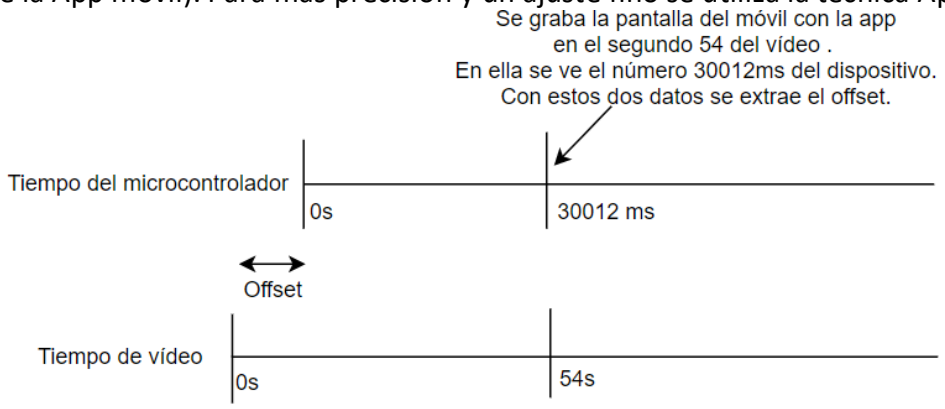


Figura 32: Esquema del sistema de sincronización por característica BLE donde aparece representado el offset.

5.3.2. Aplauso

Con esta técnica se hace un ajuste de sincronización fino entre datos y video. Esta técnica se realiza previamente a instalar el collar en las ovejas y consiste en realizar 3 movimiento rápido con el sensor. Estos acelerones son recogidos por el sensor acelerómetro y hace que en los datos aparezcan unos incrementos elevados en los valores de aceleración. De esta forma, viendo los datos se puede determinar en qué milisegundo del dispositivo se ha capturado este evento y viendo el vídeo se puede conocer el segundo exacto en el que se ha grabado el evento, por lo que ya se puede calcular el offset entre ellos.

En la imagen inferior se ilustra este procedimiento tras haber aplicado previamente la técnica de la característica BLE. Para ello se han añadido 2 líneas discontinuas verticales de color rojo, una en el segundo 59 y otra en el 63, que es el segundo en que se ha visto en el video que sucedía ese movimiento rápido del sensor. Ahora, viendo los datos se

puede apreciar cómo estas líneas no coinciden con los momentos donde tenemos los picos registrados de aceleración, que suceden un poco más tarde (segundo 60 y 64), por lo que podemos calcular el offset que es de aproximadamente 1 segundo. Ver Figura 33.

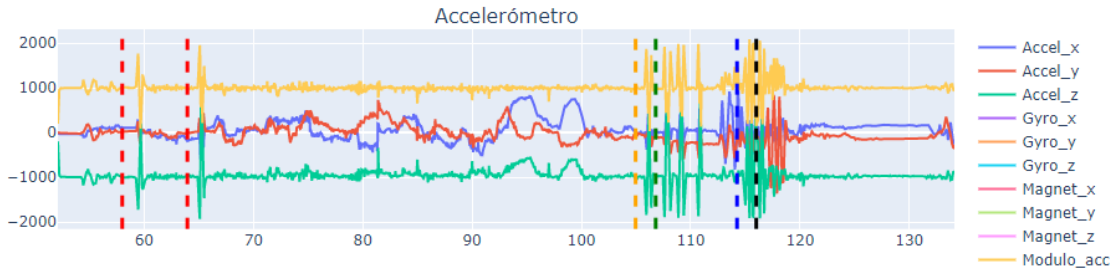


Figura 33: Representación de los datos y cursores (líneas discontínuas) que se aprecia que están desfasadas (no coincide las subidas rápidas del inicio con estas líneas).

Viendo esta diferencia de 1 segundo, lo que se hace es sumar 1 segundo al tiempo que tenía asociado cada dato y ya se aprecia que el ajuste queda perfecto (Ver Figura 34). Teniendo los datos sincronizados, se puede empezar el etiquetado con la certeza de que lo que se está etiquetando, viendo el vídeo, coincide con los datos de ese momento.

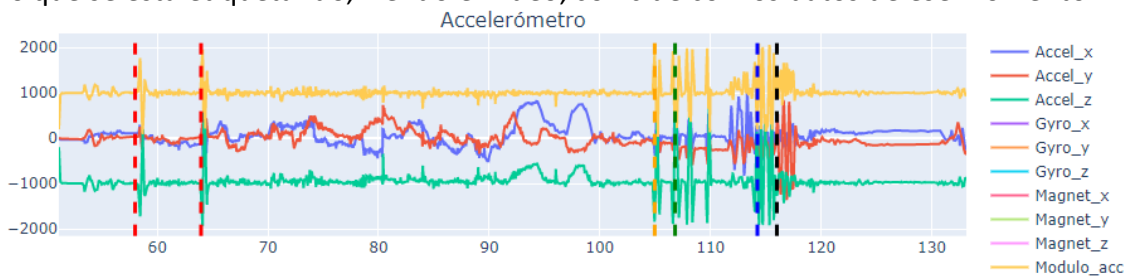


Figura 34: Representación de los datos y cursores (líneas discontinuas) que se aprecia que están perfectamente ajustadas (coinciden las subidas rápidas del inicio con estas líneas). El proceso de ajuste ha sido manual.

5.4. Procesado de los datos: Enventanados, filtros y extracción de características

Una vez se tienen los datos sincronizados con el video y se ha realizado el etiquetado, el siguiente paso es empezar a trabajar los datos para que aporten una información más relevante al posterior algoritmo de clasificación.
Este procesado de los datos consta de 4 fases: filtrado, enventanado, normalizado y extracción de características.

Como la frecuencia de muestreo seleccionada para la adquisición de los datos es muy elevada frente a los movimientos que puede realizar una oveja, esto da lugar a que los sensores capturen ruido. El filtrado es muy útil para eliminar estos ruidos y movimientos de frecuencias muy elevadas para una oveja y que no aportan información de interés. Para este propósito se ha aplicado un filtrado de primer orden y una media móvil a los datos. Con estos filtros los datos quedan suavizados y se evita que el algoritmo clasificador de actividades posterior aprenda ese ruido. Se puede ver en la Figura 35 como quedan estos datos después de filtrarse, señal real (azul) y señal suavizada (rojo).

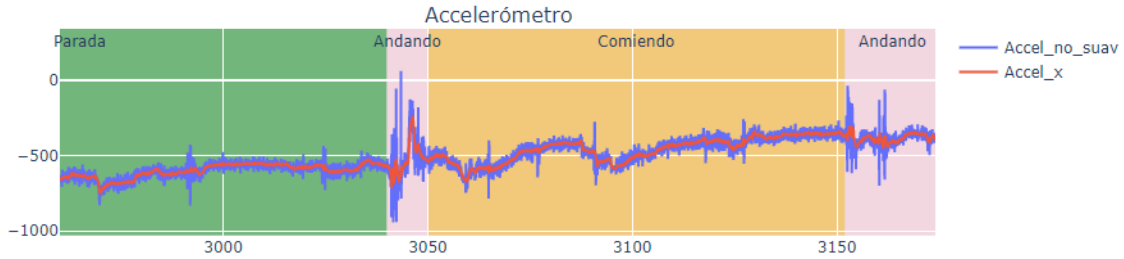
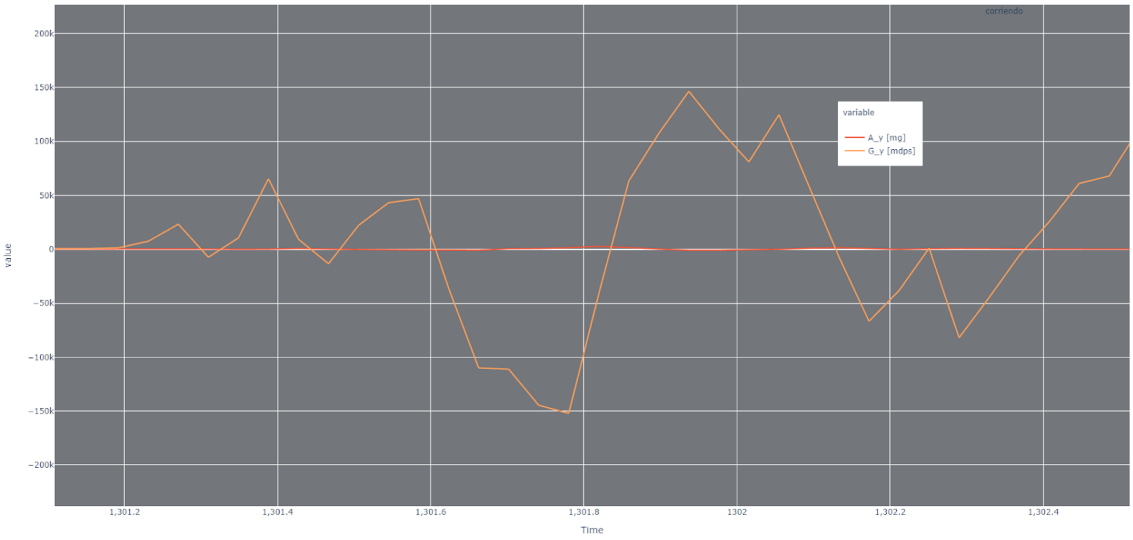


Figura 35: Señal real (azul) y señal tras aplicar el filtrado digital (rojo).

El enventanado de datos, consiste en agrupar un conjunto de muestras consecutivas en tiempo y tratar a este conjunto como un nuevo dato. A este conjunto se le conoce como una ventana temporal. Esta ventana es un nuevo dato que aporta información de la tendencia previa de los datos. El tiempo de ventana que mejores resultados ha ofrecido en términos generales ha sido la de 200 ms con un solapamiento del 20% y es el que se continúa usando para la comparativa posterior.

El siguiente paso consiste en la normalización de la base de datos. Esta normalización sirve para independizar los datos del sistema de adquisición. La técnica utilizada ha sido Z1 score, con la que se consigue una desviación de valor 1 y una media de 0, y se equilibra el peso de los sensores con diferentes mediciones. Se puede ver en la Figura 36 (imagen superior) que el valor capturado por este sensor de aceleración (señal roja) es mucho menor que el del giroscopio (señal naranja), pero que, tras aplicarles la normalización, acelerómetro y giroscopio presentan valores comparables (imagen inferior).



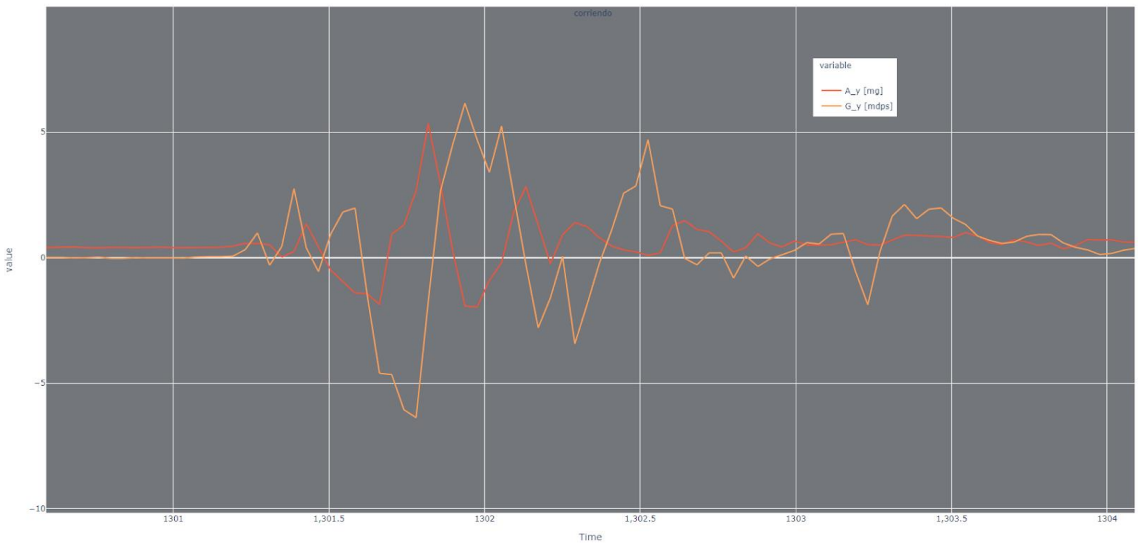


Figura 36: Se representa una ventana de datos antes (imagen superior) y después (imagen inferior) de aplicárseles la normalización Z1 Score.

El último paso en el procesado de los datos consiste en extraer de cada ventana normalizada una serie de características que van a representar a todos los datos de la ventana. En este caso siguiendo el estado del arte, las características extraídas de cada ventana han sido: la media, el valor máximo, el área bajo la curva, la máxima variación en la ventana y valor eficaz de las señales tanto del acelerómetro como del giroscopio.

5.5. Proyección de datos con UMAP

Para observar el espacio de datos que se tiene y si los datos son separables, una buena idea es proyectarlo sobre dos o tres dimensiones y analizar la proyección. Hay que tener en cuenta que estas representaciones están limitadas a 3 dimensiones, por lo que no se puede saber a ciencia cierta si las clases son fácilmente separables respecto a una dimensión diferente que no ha sido representada, pero sirve como un primer acercamiento para entender si las características elegidas son discriminantes y fiables.

Para este propósito se ha utilizado un modelo UMAP (*Uniform Manifold Approximation and Projection*), en cuya representación, para el conjunto de datos de entrenamiento de la primera sesión, se aprecia que en general los características elegidas permiten al algoritmo determinar zonas propias asociadas a cada clases (Ver Figura *37*). Sin embargo, se puede ver que, aunque la mayoría de los datos de la clase “andando” están concentrados en la parte superior, estos se encuentran en zonas donde mayoritariamente predomina otra actividad, como por ejemplo “corriendo”. Una explicación a esto puede deberse a la subjetividad de la persona que etiqueta, ya que en un espacio tan reducido es difícil determinar cuándo una oveja está realmente corriendo y cuando está andando.

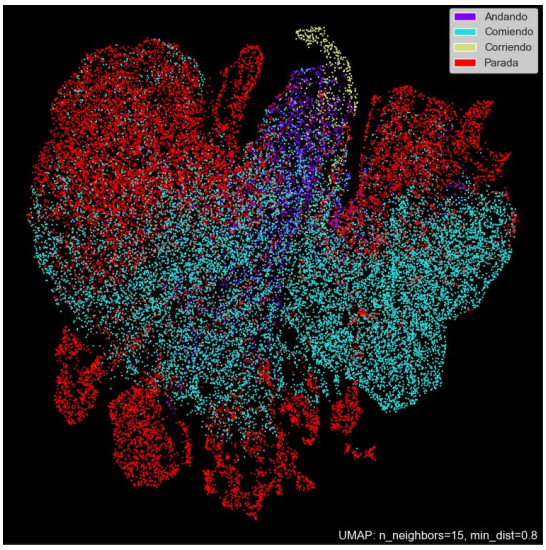


Figura 37: Proyección de los datos en UMAP con los datos de dos sesiones diferentes.

5.6. Algoritmos de clasificación

5.6.1. Introducción

El problema que se trata de resolver es un problema de clasificación de actividades de ovejas. Para determinar la mejor herramienta de clasificación se plantea una comparativa en Python de dos modelos: Bosques Aleatorios (*Random Forest*), basados en una asamblea de árboles de decisión, y un perceptrón multicapa (MLP). La selección de estos modelos es debida a que los Random Forest aparecen en repetidas ocasiones en los artículos analizados, y el modelo MLP ya había sido implementado en un momento anterior obteniendo unos buenos resultados. Para realizar esta comparativa se utiliza la misma configuración de entradas, de clases de salida y de mismos datos de entrenamiento y de prueba.

El desarrollo de todos estos modelos ha sido gracias a las librerías de Scikit Learn [45] (Random Forest) y Keras [46](MLP).

5.6.2. Random Forest

El modelo *Random Forest* está formado por un conjunto de árboles de decisión (modelos basados en reglas binarias), que han sido entrenados con muestras de datos ligeramente diferentes [40] y que todos en conjunto (*ensemble*) hacen una predicción de la actividad ante cierta entrada al algoritmo, dándole una mayor robustez que la que tendría un solo árbol como modelo.

Para el reconocimiento de actividades de ovejas, se ha implementado un *Random Forest* formado por 5 árboles de decisión utilizando el 70% de los datos para entrenamiento y el 30% de los datos para test.

Para alcanzar una mejor comprensión de cómo son estos árboles de decisión, se ha implementado un *script* de visualización (mediante la librería *Scikit Learn*), de uno de estos árboles una vez se ha entrenado. Ver Figura 38.

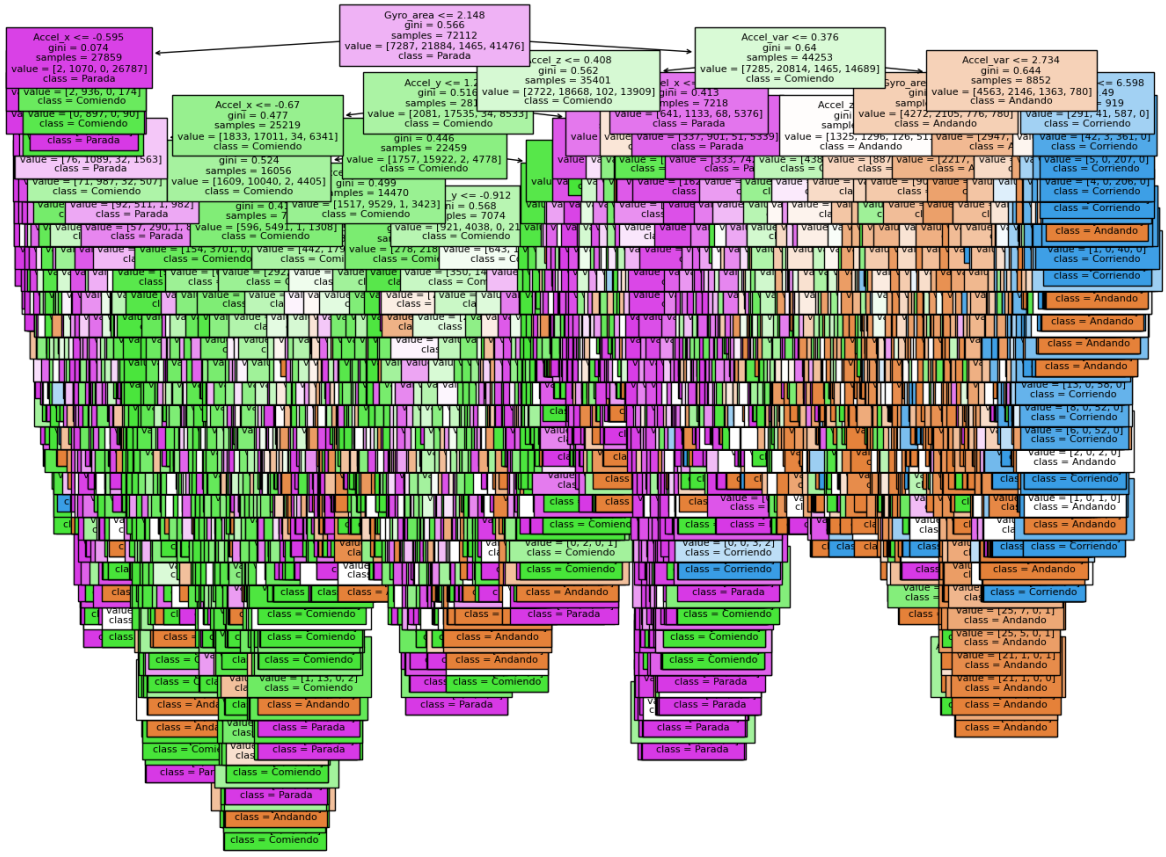


Figura 38: Representación de un árbol de decisión entrenado con los datos de las ovejas.

En la Figura 38 se aprecia el funcionamiento del árbol como un clasificador binario. Cada una de las cajas representa una comparación con un valor; por ejemplo, la caja de más arriba analiza si el área bajo la curva de una ventana es menor que el valor 2.148 y dependiendo de si se cumple o no la desigualdad, se continúa o por la flecha de la izquierda o por la de la derecha. Esto se hace sucesivamente hasta alcanzar una caja de la cual no salen flechas (hojas), en ese momento se observa la clase que está asociada a esa caja (cada clase es de un color) y esa es la etiqueta asignada a la ventana procesada.

Una vez entrenados los 5 árboles, estos son testeados con un conjunto de datos de test, previamente no mostrados a los árboles, para que los clasifiquen. Tras clasificar todos los datos se analizan las predicciones hechas frente a los valores reales y el resultado se muestra en forma de matriz de confusión con los datos de test, consiguiendo unos resultados del 93.22% de exactitud. Ver **¡Error! No se encuentra el origen de la referencia.**.

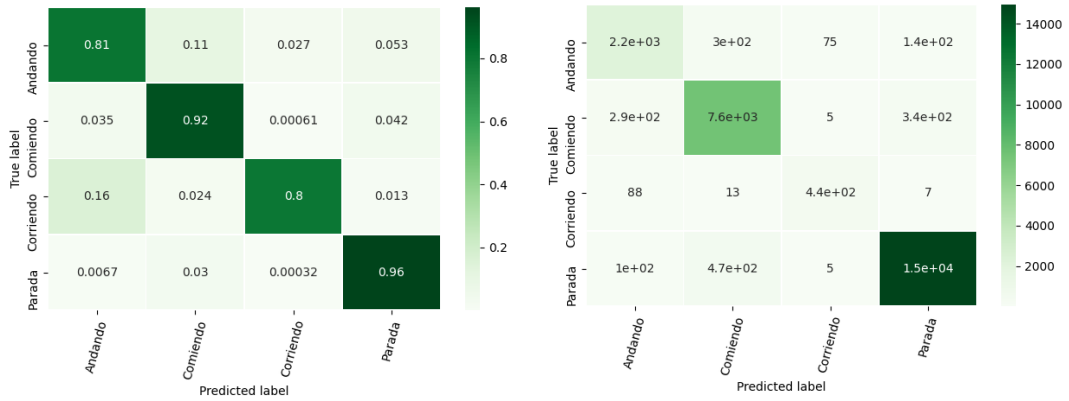


Figura 39: Representación de las matrices de confusión de los datos de test del Random Forest.

Como se puede apreciar, existe una cierta confusión entre los datos andando y corriendo, como también los reflejaba el UMAP. El motivo puede deberse a la subjetividad en la persona que etiqueta, entre lo que es un evento de andar y otro de correr.

5.6.2. MLP

Un MLP es un perceptrón multicapa y es un algoritmo basado en redes neuronales. La red implementada cuenta cuenta con dos capas ocultas y tiene una estructura 5-8-4-4. Para evitar problemas de sobre-aprendizaje se ha aplicado un *Dropout* de 0.2 a ambas capas intermedias. La función de activación es la *ReLU* en las capas ocultas y *softmax* para la capa de salida.
Los resultados del MLP se arrojan en la siguiente matriz de confusión. Ver Figura 40:

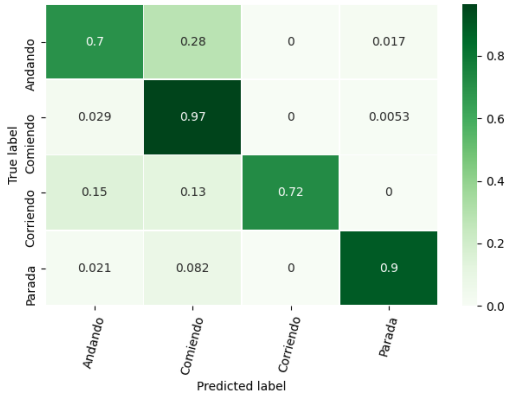


Figura 40. Matriz de confusión del MLP.

Como se puede apreciar, existe una cierta confusión entre los datos andando y corriendo, como también los reflejaba el UMAP. El motivo puede deberse a la subjetividad en la persona que etiqueta, entre lo que es un evento de andar y otro de correr.

5.7. Análisis sobre los resultados de los algoritmos

Uno de los problemas de la base de datos es la predominancia de unas clases frente a las demás. En la situación en la que se encuentran las ovejas de veterinaria, no hay gran espacio para moverse, por lo tanto, la actividad mayoritaria es “parada” y la menos predominante es “corriendo”. Esto hace que los clasificadores al tener más dato de una clase que de otra, traten de aprender primero aquella clase con más datos (para conseguir mejores rendimientos) y una vez aprenda bien esta clase, pase a tratar de clasificar el resto.

Otro de estos problemas que pueden dar lugar a error está en el proceso de etiquetado.
Este proceso, por su subjetividad, puede generar datos erróneos. En este caso la actividad “corriendo” y “andando” se confunden, lo cual puede deberse a asignar movimientos de andando algo más rápidos a los habituales y viceversa. Además, al trabajar con un collar que no tiene un gran cierre respecto al cuello del animal, se puede etiquetar cierto como comportamiento como una actividad, pero al moverse el collar por el cuello del animal los datos puedan confundir a la red.

Finalmente, y considerando los problemas de este análisis se aprecia que los mejores resultados son obtenidos por *Random Forest*. Esto en parte, puede deberse a que *Random Forest* está formado por 5 clasificadores que promedian sus resultados, lo cual reduce la probabilidad de fallo en la predicción.

Capítulo 6

Implementación en microcontrolador

6.1. Introducción

Una vez los modelos han sido entrenados en Python, estos son exportados a C# para que el microcontrolador sea el que haga las predicciones. Para la exportación de cada modelo se han utilizado herramientas diferentes:

1- Librería m2cgen [39] desarrolladas para Python que permite exportar a lenguaje C# los *Random Forest*.

2- El paquete de software STMicroelectronics. X-CUBE-AI 6.0.0, que permite la incorporación de modelos basados en Keras, ONNX y TFLite a partir de los ficheros. hfd5 que contienen los pesos de la red. Este paquete se ha utilizado para la exportación en C# del modelo MLP.

Con estas herramientas, la red previamente entrenada en un ordenador de sobremesa es incorporada en el microcontrolador, de forma que se pueda realizar la clasificación de actividades en tiempo real.

6.2. Implementación del clasificador en el microcontrolador

Una vez se tiene el algoritmo clasificador entrenado y exportado a C#, el siguiente paso es cargarlo en el microcontrolador y ver su desempeño respecto al modelo entrenado en el ordenador. Uno de los problemas que pueden surgir cuando se entrena un clasificador en un ordenador, que luego se quiere llevar a un microcontrolador, es que tengan un desempeño diferente debido a la arquitectura de la CPU y el número de bits de trabajo de cada uno (64bits habitualmente una CPU de un ordenador de sobremesa y entre 8 y 32bits una CPU de un microcontrolador). Por lo tanto, se ha de verificar que el algoritmo de clasificación tiene unos resultados similares en el microcontrolador

El procedimiento seguido para esta tarea consiste en extraer un conjunto de ventanas de datos de prueba (de los que se había utilizado para testear el clasificador en Python previamente), y se almacenan en el microcontrolador en forma de matriz(ver la Figura 41). En concreto, para este experimento se utilizan 600 ventanas de datos aleatorias cogidas de los grupos de prueba.

```
#define NUMERO_VENTANAS 600
#define NUMERO_CARACTERISTICAS 6
double input[NUMERO_VENTANAS][NUMERO_CARACTERISTICAS] = {
        {0.08318625455760273,0.013212382909290055,-0.6447607862193412,1.5822287211225046,-0.723937161807204,Parada},
        {0.07335648454437461,0.02827184176749289,-0.8390812616409716,0.4659544924685759,-0.4669721548495773,Parada},
        {0.3293854264535709,0.042237393498334316,2.2311376555341833,2.5436047754720112,-0.005655689998020245,Parada},
        {0.05527514474876626,0.01138338561895195,-0.8390255185327036,0.5012704656794518,-0.4783831922580386,Parada},
        {0.1564193577792527,0.05971295530202801,0.46405325189006014,-1.155106572948674,0.3348337774841974,Comiendo},
        {0.46197289175687567,0.11859970174141202,-0.7992528107836124,-0.46771878617371,0.2818591196417525,Parada},
        {0.057029951615503054,0.01528829927642318,-0.8423422334746395,0.6333869425109574,-0.5232329025786361,Parada},
        {0.20705601788586145,0.04840507864345737,-0.05692183798101354,-0.816970603214052,-0.012661200305746426,Parada},
        {1.7349583630216898,0.2374876167145752,1.9613967546261863,-0.4172632605125978,-0.2559040736709197,Andando},
        {0.06502736260743884,0.010492025148566347,-0.7942080594853749,0.7015872973345836,-0.59559910183989,Parada},
        {0.05334534346357836,0.008621851515563383,-0.8400288944815246,0.4946704313416818,-0.47545821115017356,Parada},
        {5.181604799224653,1.0623596279466394,0.8221469794025548,2.119089408965869,-1.2152256552949088,Andando},
        {3.687978597601642,0.5854621082928678,1.9970444723634617,-0.04606922405107276,-0.44859171801126474,Andando},
        {0.9082019987506439,0.2363299621996861,1.8761097989764177,-0.5058426687300405,-0.24294026851383863,Andando},
        {0.2099271514647657,0.03397523182966085,-0.26980476845582957,-0.6821909546322171,-0.048880410814248314,Comiendo},
        {0.06480732836032205,0.022605644349787218,-0.6722700101495125,1.6212210292495954,-0.6865985140845808,Parada},
        {0.23400412665297915,0.08481692318820773,0.15961226608534484,-0.8239469552991505,-0.1304548839335961,Andando},
```

Figura 41: Imagen de la matriz de datos implementada en C#.

Una vez se tienen todas estas ventanas dentro del microcontrolador, se implementa un código en C# para ir haciendo pasar a cada una de estas etiquetas por el *Random Forest,* de una en una. Cada vez que el *Random Forest* hace una predicción, ésta se envía a través de una UART del microcontrolador a un módulo “UART a USB”, el cual envía esta información al ordenador, haciendo uso de la herramienta Termite (es una terminal serial) [42].

Una vez han pasado todas las ventanas y se han hecho todas las predicciones en el microcontrolador, se han de seguir los mismos pasos en el ordenador hasta tener todas las predicciones de ambos realizadas. El paso final es comparar los resultados que ofrece cada uno de ellos. Para ello, se decide implementar en C# un algoritmo que calcule la matriz de confusión en cada equipo.

El resultado se puede ver en la Figura *42*, donde se aprecia como las matrices de confusión tanto en el ordenador (terminal de la izquierda) como en el microcontrolador (ventana de Termite a la derecha) son idénticas, lo cual sugiere que no existe diferencia en los resultados a la hora de hacer predicciones entre el ordenador y el microcontrolador concluyendo que el microcontrolador está capacitado y listo para reconocer actividades de ovejas.

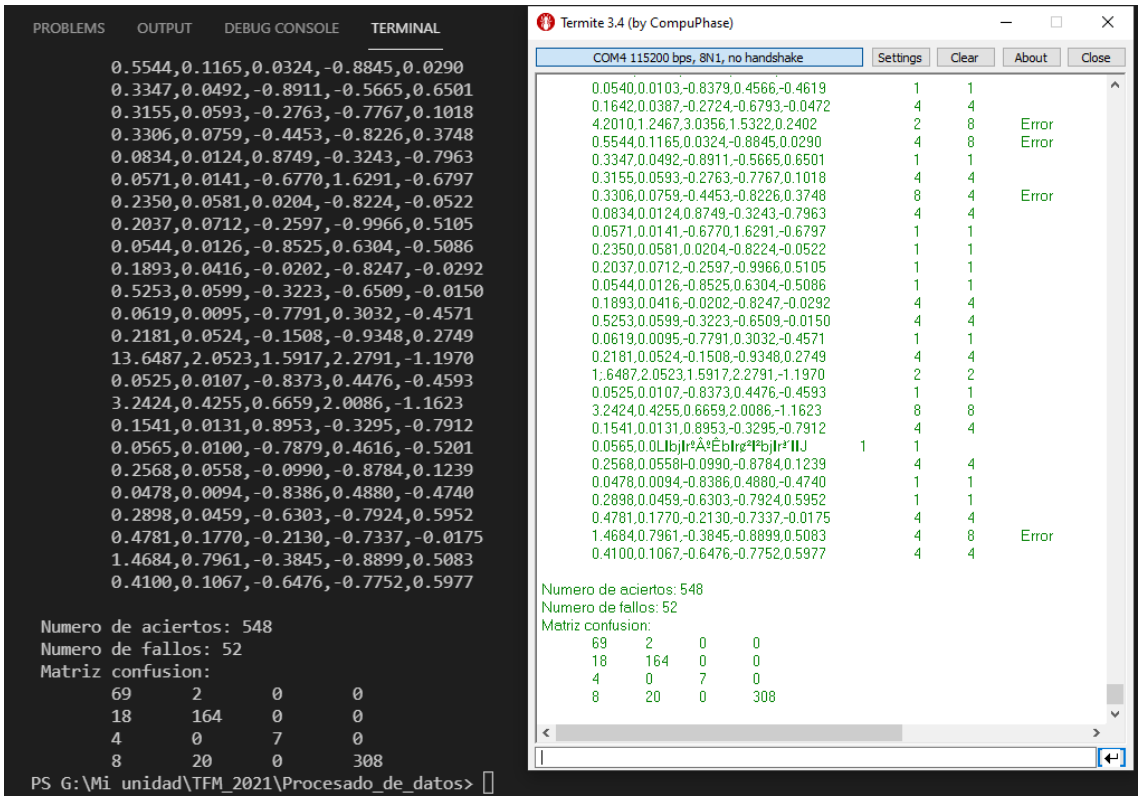


Figura 42: Comparativa del resultado de Random Forest en el ordenador y en el microcontrolador. Observar como se obtienen las mismas matrices de confusión por lo que el algoritmo funciona igual tanto en microcontrolador como en PC.

6.3. Características hardware de los modelos

Una vez se ha verificado el correcto funcionamiento del clasificador en el microcontrolador, se pasa a analizar los requisitos de memoria y el tiempo que necesitan ambos clasificadores en hacer una predicción de la actividad. Las prestaciones de memoria del microcontrolador utilizadas en

Random Forest tiene un tamaño de 671kB, lo cual implica un espacio de alrededor de 134kB por cada árbol de decisión. El tiempo de predicción del algoritmo *Random Forest* es de aproximadamente 49.3 microsegundos.

MLP ocupa un tamaño de 1.05kB, lo cual es un tamaño bastante más reducido que Random Forest. El motivo de esto se debe a que el MLP es una red relativamente pequeña y el sistema de implementación en C# parece estar más optimizado que el del Random Forest. El tiempo aproximado de ejecución es de 25 microsegundos.

Como las prestaciones de memoria del microcontrolador utilizado la etapa versión 1.0 son de 1024kB de Flash, 96kB de RAM y 32kB de RAM2 ambos clasificadores pueden ser implementados en el microcontrolador. El tiempo de predicción con valores cercanos a las decenas de microsegundos no suponen un problema puesto que el mayor periodo de muestreo que se podría establecer son 10ms entre muestra y muestra y aun con eso habría tiempo en exceso para realizar la predicción.

6.4. Ventajas de hacer inferencia en el microcontrolador

Una de las ventajas de tener un dispositivo capaz de clasificar datos es la reducción en consumo de batería que puede lograrse. A la hora de hacer comunicación del estado del animal, el sistema transmitirá únicamente los cambios de transición de estado, en lugar de los datos en crudo para ser procesados de forma externa. Esto redunda considerablemente en un ahorro energético, dado que sólo las comunicaciones imprescindibles serán ejecutadas.

Otra ventaja, es el ahorro de memoria existente en el caso de tener que almacenar internamente los datos dentro de una tarjeta SD. Esto puede apreciarse en el siguiente ejemplo:

Actualmente, el Random Forest necesita una ventana de 200 milisegundos de muestras para hacer una predicción de la actividad. Como se ha muestreado a 100Hz, una ventana de 200 ms equivale a 20 líneas de datos. Cada línea de datos contiene la hora, el timestamp, y los ejes x,yz de los sensores acelerómetro, giroscopio y magnetómetro.
Un ejemplo de una de estas líneas podría ser “15:37:04;18541;128;10; -992;3500;-6720;3150;-286;-435;85”. Esta línea contiene 57 caracteres, lo cual son 57 bytes de información. Es decir, para poder procesar a posteriori una ventana de 200ms sería necesario guardar en la tarjeta SD, 20 líneas x 57caracteres = 1140 bytes. En el caso de hacer el procesado de la ventana y guardar únicamente la etiqueta de clase, las nuevas líneas de datos se reducirían a “15:37:04;18541;Parada\r\n” que son a 23 bytes. Por lo tanto, pasamos de guardar 1200bytes cada 200ms a tan solo 23bytes.

De esta forma se concluye que hacer el procesado de los datos en el microcontrolador supone un ahorro en consumo de batería y en espacio de memoria utilizado.

Capítulo 7

Conclusiones y Líneas Futuras

7.1. Conclusiones

En este trabajo fin de máster, se ha llevado a cabo un amplio estudio del estado del arte en el reconocimiento de actividades animales. Se han analizado las técnicas más importantes para el estudio de animales de diferente tamaños analizando y se han adoptado aquellos aspectos más relevantes para el reconocimiento de actividades de ovejas.

Se ha desarrollado hardware y firmware para elaborar un dispositivo que permite la adquisición de datos de las ovejas. En concreto, se trabaja con sensores inerciales, como son el acelerómetro y el giroscopio y se ha trabajado en nuevas estrategias para intentar adquirir datos que permitan en el futuro hacer estudios sociales de los grupos de ovejas.

Esta implementación firmware y hardware, ha tratado de ser lo más personalizada al proyecto, buscando la mayor integración posible con usuarios de bajo conocimiento de electrónica, mediante el desarrollo de una aplicación móvil de fácil gestión.

Se ha creado una base de datos que almacena de los datos recogidos con los sensores y permite su estudio y análisis posterior. Para la generación de las etiquetas de estos datos, se ha elaborado una interfaz gráfica que acelera este proceso y permite de forma sencilla, hacer un etiquetado de los datos a cualquier usuario.

Para el tratamiento de los datos, se han utilizado las estrategias comentadas en el estado del arte como han sido,la normalización, el filtrado y la creación de ventanas de datos. Además, se han extraído características de estas ventanas que permiten entrenar eficientemente a los algoritmos clasificadores. Para poder realizar un etiquetado automático, se han desarrollado dos tipos de clasificadores diferentes que permiten inferir la actividad que realiza la oveja. Estos clasificadores además, han sido implementados en un microcontrolador y se ha validado su correcto funcionamiento comparándolo con el que tendría un ordenador de sobremesa.

7.2. Líneas Futuras

Las líneas futuras para considerar son las siguientes:

− Desarrollo de un Gateway para gestionar de forma automática los datos de todos los nodos de la red.

− Aumentar el número de datos para conseguir modelos más precisos y genéricos, aplicables a cualquier tipo de oveja, así como reducir fallos en la detección de “andando” vs “corriendo”.

− Realización de una prueba con un mayor número de ovejas, durante más tiempo y analizar el comportamiento social de dichas ovejas.

− Estudiar la eficacia del modelo entre ovejas macho y hembra, y entre ovejas adultas y jóvenes, ya que muy probablemente presenten comportamientos distintos.

− Dado el reducido tamaño de la red neuronal, podrían plantearse modelos más complejos basados en *Deep Learning*, utilizando capas convolucionales 1D sobre la propia señal, de forma que la extracción de características manual pase a ser automatizada por el modelo neuronal.

− Para la detección de comportamientos más sutiles, como por ejemplo “rumiando”, podría plantearse el rediseño del experimento con el dispositivo incorporado a un correaje facial.

− Planteamiento de un diseño más estandarizado de la captura de datos, incluyendo el marcado del lomo de las ovejas para su reconocimiento individual, así como una posición fija de cámara más cenital sobre el corral que permita la observación de los animales sin tantas oclusiones.

− Realización de un análisis comparativo de las señales proporcionadas por diferentes dispositivos, de forma que se puedan estandarizar las posibles derivas y/o tolerancias de los sensores que incorporan.

Referencias:

[1] Javier Asín, María Pascual-Alonso, Pedro Pinczowski, Marina Gimeno, Marta Pérez, Ana Muniesa, Lorena de Pablo-Maiso, Ignacio de Blas, Delia Lacasta, Antonio Fernández, Damián de Andrés, Ramsés Reina, Lluís Luján, Cognition and behavior in sheep repetitively inoculated with aluminum adjuvant-containing vaccines or aluminum adjuvant only, Journal of Inorganic Biochemistry,Volume 203,2020,110934,ISSN 0162-0134,https://doi.org/10.1016/j.jinorgbio.2019.110934.
https://www.sciencedirect.com/science/article/pii/S016201341930501X

[2] Brown et al.: Observing the unwatchable through acceleration logging of animal behavior. Animal Biotelemetry 2013 1:20.

[3] Schneirla TC: The relationship between observation and experimentation in the field study of behavior. Annals NY Acad Sci 1950, 51:1022–1044.

[4] le Roux, S., Marias, J., Wolhuter, R. et al. Animal-borne behaviour classification for sheep (Dohne Merino) and Rhinoceros (Ceratotherium simum and Diceros bicornis). Anim Biotelemetry 5, 25 (2017). https://doi.org/10.1186/s40317-017-0140-0

[5] Walton E, Casey C, Mitsch J, Vázquez-Diosdado JA, Yan J, Dottorini T, Ellis KA, Winterlich A, Kaler J. 2018 Evaluation of sampling frequency, window size and sensor position for classification of sheep behaviour. R. Soc. open sci.5: 171442. http://dx.doi.org/10.1098/rsos.171442

[6] ZigBee-based wireless sensor networks for classifying the behaviour of a herd of animals using classification trees E.S. Nadimia,b,, H.T. Søgaardc , T. Bakb. BIOSYSTEMS ENGINEERING 100 (2008) 167– 176

[7] Vázquez Diosdado, J.A., Barker, Z.E., Hodges, H.R. et al. Classification of behaviour in housed dairy cows using an accelerometer-based activity monitoring system. Anim Biotelemetry 3, 15 (2015). https://doi.org/10.1186/s40317-015-0045-8

[8] Shiomi K, Sato K, Mitamura H, Arai N, Naito Y, Ponganis PJ: Effect of ocean current on the dead-reckoning estimation of 3-D dive paths of emperor penguins. Aqua Biol 2008, 3:265–270.

[9] Owoeye, Kehinde & Hailles, Stephen. (2018). Online Collective Animal Movement Activity Recognition.

[10] Cow behaviour pattern recognition using a three-dimensional accelerometer and support vector machines Paula Martiskainen a, *, Mikko Ja¨rvinen a , Jukka-Pekka Sko¨n b , Jarkko Tiirikainen b , Mikko Kolehmainen b , Jaakko Mononen. P. Martiskainen et al. / Applied Animal Behaviour Science 119 (2009) 32–38

[11] Distinguishing Cattle Foraging Activities Using an Accelerometry-Based Activity Monitor

Yoshitoshi, Rena; Watanabe, Nariyasu; Kawamura, Kensuke; Sakanoue, Seiichi; Mizoguchi, Ryo; Lee, Hyo-Jin; Kurokawa, Yuzo . Rangeland Ecology and Management ; Lawrence Tomo 66, N.º 3, (May 2013): 382-386

[12] F.A.P. Alvarenga, I. Borges, L. Palkovič, J. Rodina, V.H. Oddy, R.C. Dobos, Using a three-axis accelerometer to identify and classify sheep behaviour at pasture, Applied Animal Behaviour Science, Volume 181, 2016, Pages 91-99, ISSN 0168-1591, https://doi.org/10.1016/j.applanim.2016.05.026. Keywords: Activity; Decision-tree; Grazing; Ruminant; Sensor

[13] Brown, D.D., LaPoint, S., Kays, R., Heidrich, W., Kümmeth, F. and Wikelski, M. (2012), Accelerometer-informed GPS telemetry: Reducing the trade-off between resolution and longevity. Wildlife Society Bulletin, 36: 139-146. https://doi.org/10.1002/wsb.111

[14] Sato K, Watanuki Y, Takahashi A, Miller PJO, Tanaka H, Kawabe R, Ponganis PJ, Handrich Y, Akamatsu T, Watanabe Y, Mitani Y, Costa DP, Bost CA, Aoki K, Amano M, Trathan P, Shapiro A, Naito Y: Stroke frequency, but not swimming speed, is related to body size in free-ranging seabirds, pinnipeds and cetaceans. Proc R Soc Series B Biol Sci 2007, 274:471–477

[15] Kozue Shiomi, Katsufumi Sato, Paul J. Ponganis; Point of no return in diving emperor penguins: is the timing of the decision to return limited by the number of strokes?. J Exp Biol 1 January 2012; 215 (1): 135–140. doi: https://doi.org/10.1242/jeb.064568

[16] Shepard ELC, Wilson RP, Quintana F, Laich AGM, Forman DW: Pushed for time or saving on fuel: fine-scale energy budgets shed light on currencies in a diving bird. Proc R Soc Ser B Biol Sci 2009, 276:3149–3155.

[17] Ran Nathan, Orr Spiegel, Scott Fortmann-Roe, Roi Harel, Martin Wikelski, Wayne M. Getz; Using tri-axial acceleration data to identify behavioral modes of free-ranging animals: general concepts and tools illustrated for griffon vultures. J Exp Biol 15 March 2012; 215 (6): 986–996. doi: https://doi.org/10.1242/jeb.058602

[18] Wilson, Rory & Shepard, Emily & Laich, Agustina & Frere, Esteban & Quintana, Flavio. (2010). Pedalling downhill and freewheeling up; A penguin perspective on foraging. Aquatic Biology - AQUAT BIOL. 8. 193-202. 10.3354/ab00230.

[19] Terrestrial animal tracking as an eye on life and planet Roland Kays,Margaret C. Crofoot,Walter Jetz,, Martin Wikelski Science 12 Jun 2015: Vol. 348, Issue 6240, aaa2478 DOI: 10.1126/science.aaa2478

[20] Painter, M.S., Blanco, J.A., Malkemper, E.P. et al. Use of bio-loggers to characterize red fox

behavior with implications for studies of magnetic alignment responses in free-roaming animals. Anim Biotelemetry 4, 20 (2016). https://doi.org/10.1186/s40317-016-0113-8

[21] Fehlmann, G., O'Riain, M.J., Hopkins, P.W. et al. Identification of behaviours from accelerometer data in a wild social primate. Anim Biotelemetry 5, 6 (2017). https://doi.org/10.1186/s40317-017-0121-3

[22]Yasuhiko Naito, Horst Bornemann, Akinori Takahashi, Trevor McIntyre, Joachim Plötz,Fine-scale feeding behavior of Weddell seals revealed by a mandible accelerometer, Polar Science,Volume 4, Issue 2,2010,Pages 309-316,ISSN 1873-9652,https://doi.org/10.1016/j.polar.2010.05.009.
(https://www.sciencedirect.com/science/article/pii/S1873965210000496)

[23] Shepard ELC, Wilson RP, Halsey LG, Quintana F, Gomez Laich A, Gleiss AC, Liebsch N, Myers AE, Norman B: Derivation of body motion via appropriate smoothing of acceleration data. Aqua Biol 2009, 4:235–241.

[24] Watanabe, N., Sakanoue, S., Kawamura, K., Kozakai, T., 2008. Development of an automatic classification system for eating, ruminating and resting behavior of cattle using an accelerometer. Jap. Soc. Grass Sci. 54, 231–237, http://dx.doi. org/10.1111/j.1744-697X.2008.00126.x.

[25] van Oort BEH, Tyler NJC, Storeheier PV, Stokkan K-A: The performance and validation of a data logger for long-term determination of activity in free-ranging reindeer, Rangifer tarandus L. App Anim Behav Sci 2004, 89:299–308.

[26] Sakamoto KQ, Sato K, Ishizuka M, Watanuki Y, Takahashi A, Daunt F, Wanless S: Can ethograms be automatically generated using body acceleration data from free-ranging birds? PLoS ONE 2009, 4(4):e5379

[27] Wilson RP, Shepard ELC, Liebsch N: Prying into the intimate details of animal lives: use of a daily diary on animals. Endangered Spec Res 2008, 4:123–137.

[28] Racewicz, P.; Ludwiczak, A.; Skrzypczak, E.; Składanowska-Baryza, J.; Biesiada, H.; Nowak, T.; Nowaczewski, S.; Zaborowicz, M.; Stanisz, M.; Slósarz, ´ P. Welfare Health and Productivity in Commercial Pig Herds. Animals 2021, 11, 1176. https://doi.org/10.3390/ ani11041176 Academic Editor: Melissa Hempstead Received: 15 March 2021 Accepted: 17 April 2021 Published: 20 April 2021

[29] https://www.nordicsemi.com/Software-and-tools/Development-Tools/nRF-Connect-for-mobile Accedida por última vez: 22/06/2021

[30]https://www.arduino.cc/reference/en/libraries/freertos/ Accedida por última vez: 22/06/2021

[31] https://github.com/kgabis/parson Accedida por última vez: 22/06/2021

[32] TTGO-V2: http://www.lilygo.cn/prod_view.aspx?TypeId=50033&Id=1116 Accedida por última vez: 22/06/2021

[33] STEVAL-STLCS01V1 y STLCR01V1: https://www.st.com/en/evaluation-tools/steval-stlkt01v1.html Accedida por última vez: 22/06/2021

[34] https://platformio.org/ Accedida por última vez: 22/06/2021

[35] https://www.arduino.cc/en/software/ Accedida por última vez: 22/06/2021

[36] https://www.espressif.com/en/products/modules/esp32 Accedida por última vez: 22/06/2021

[37] https://github.com/RAKWireless/Products_practice_based_on_RUI_v2.0 Accedida por última vez: 22/06/2021

[38] https://www.waveshare.com/wiki/File:L76X_GPS_HAT_code.7z Accedida por última vez: 22/06/2021

[39] https://github.com/BayesWitnesses/m2cgen Accedida por última vez: 22/06/2021

[40]

https://www.cienciadedatos.net/documentos/33_arboles_decision_random_forest_gradient_boosting_c50 Accedida por última vez: 22/06/2021

[41] J. Marek, "MEMS Technology- from Automotive to Consumer," 2007 IEEE 20th International Conference on Micro Electro Mechanical Systems (*MEMS)*, 2007, pp. 59-60, doi: 10.1109/MEMSYS.2007.4433012. Accedida por última vez: 22/06/2021

[42] https://www.compuphase.com/software_termite.htm Accedida por última vez: 22/06/2021

[43]https://www.st.com/en/microcontrollers-microprocessors/stm32-32-bit-arm-cortex-mcus.html Accedida por última vez: 22/06/2021

[44] https://www.altium.com/es Accedida por última vez: 22/06/2021

[45] https://scikit-learn.org/stable/ Accedida por última vez: 22/06/2021

[46] https://keras.io/ Accedida por última vez: 22/06/2021

Lista de Figuras

Lista de Tablas

ANEXOS

Anexo I

Imágenes de animales con collar

En este anexo se recogen más imágenes de collares implementados en animales [19]

Figura 43. Collares en diferentes animales recogidos en [19]

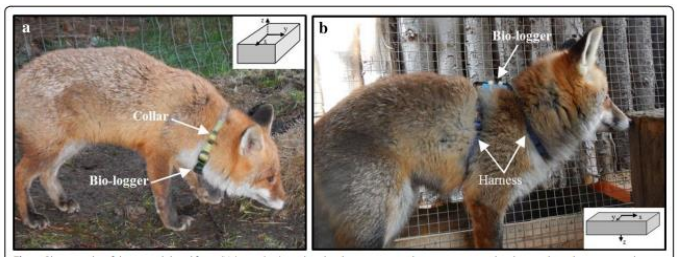


Figura 44. Collares implementados en zorros. [20]

Anexo II
Imágenes de Características RX y TX

En este anexo, se muestra de forma gráfica el dispositivo se muestra hacia el exterior vía BLE con un identificador único “HOWLA3” (ver Figura *45*).

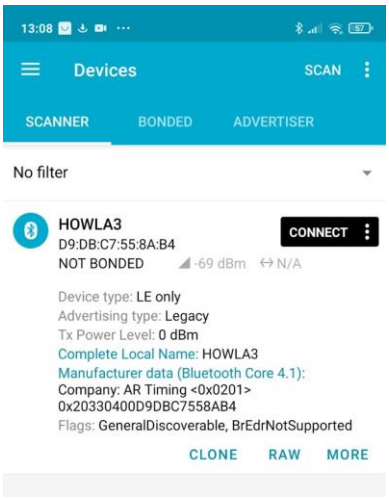


Figura 45. Captura de pantalla de como hace advertising el dispositivo enviando su identificador único.

En la Figura 46 se muestra como es el servicio que alberga la consola UART BLE. Se puede el servicio personalizado (la aplicación lo llama Unknown) de 128 bits que contiene dos características. La primera de ellas que permite ser leída o que envía una notificación cuando cambie su valor, esta es la característica RX. La segunda de ellas a través de la cual se pueden enviar comandos llamada característica TX. De hecho se puede apreciar cómo se acaba de enviar por esta característica el comando que pide que se informe del estado del dispositivo “{“c”:”GD”,”p”:”e”}” y se ha recibido en la otra que el modo es el de “ESPERA_COMANDOS”.

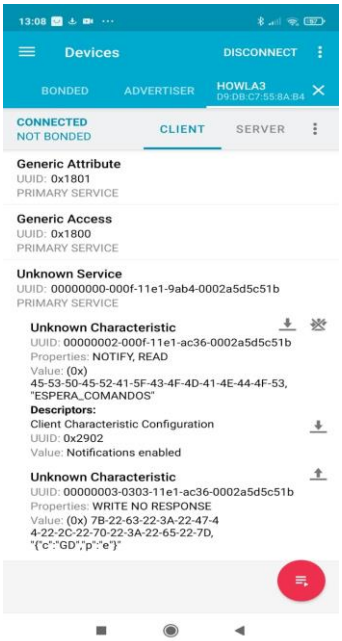


Figura 46. Captura de la consola UART BLE con sus características RX y TX.

Anexo III

DIAGRAMAS DE FIRMWARE POR ETAPAS

En cada etapa se ha ido confeccionando un firmware diferente, aunque siempre manteniendo y reutilizando el código en la medida de lo posible. En este anexo se recogen a estos firmwares a nivel de bloques para cada placa.

Diagrama del firmware de las placas de desarrollo TTGO-V2

El firmware desarrollado con esta placa es el que es incorpora alguna diferencia mayor con respecto al resto de desarrollos, puesto que el microcontrolador ESP-WROOM32 integra una interfaz para WIFI que el resto no tiene.

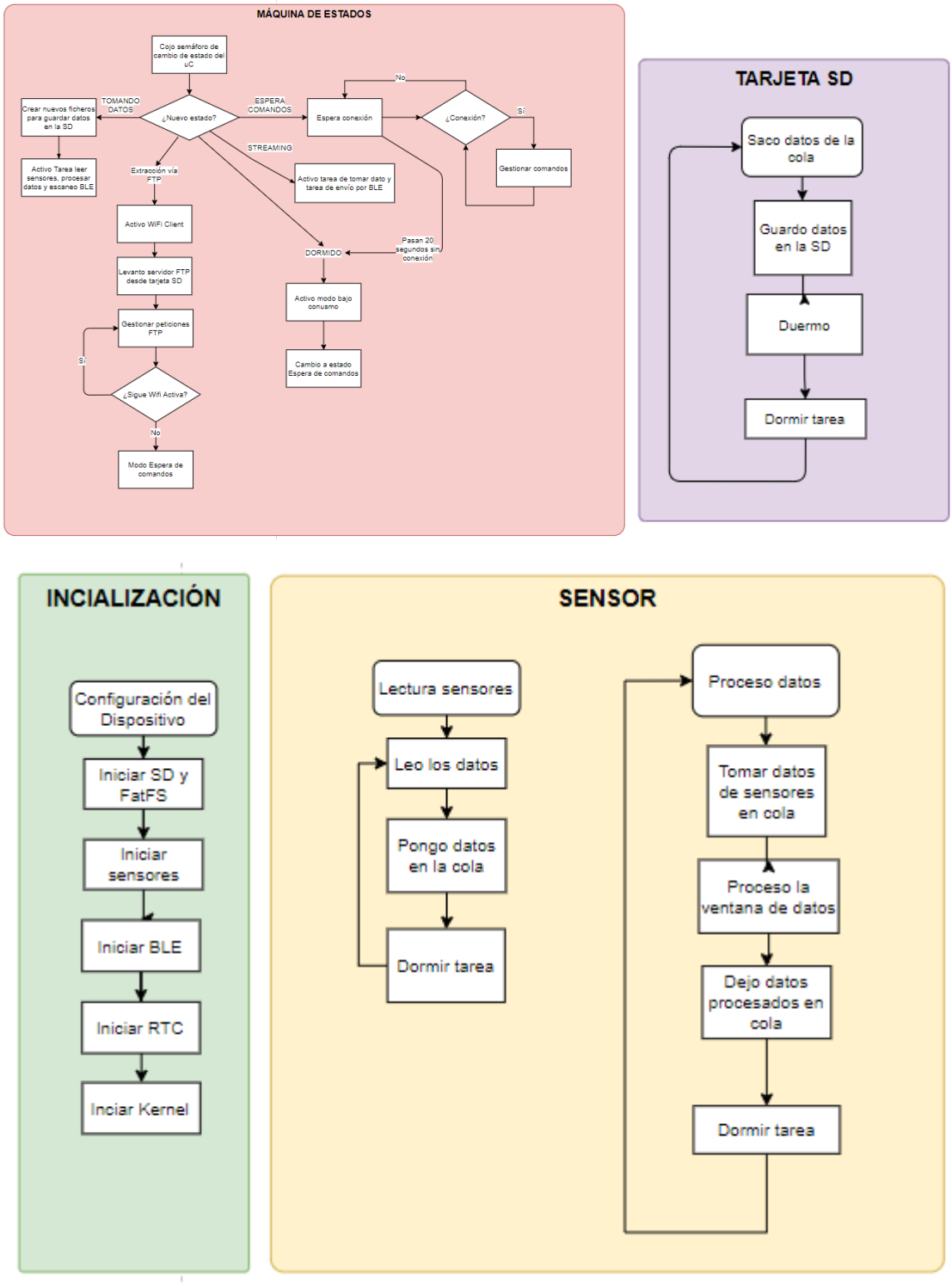
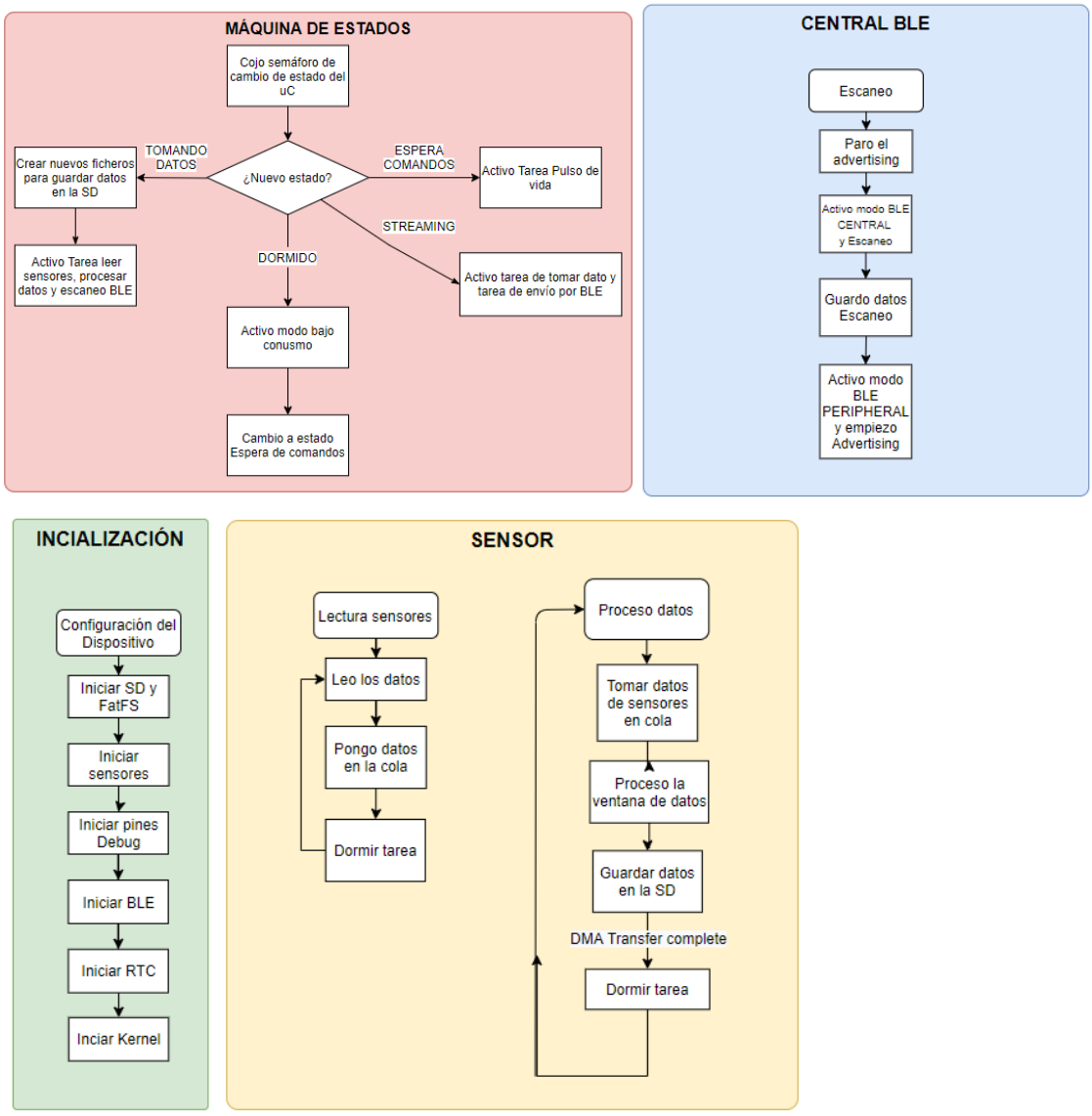
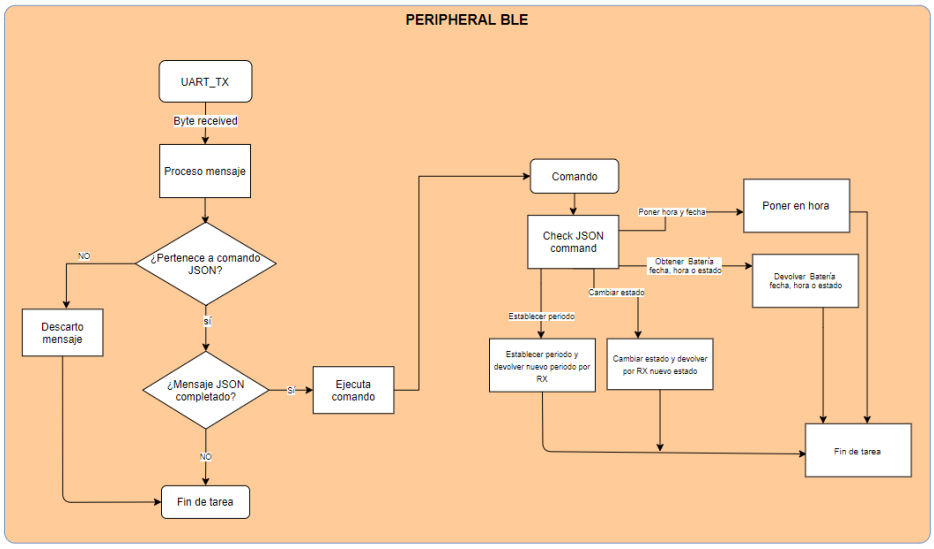


Diagrama del firmware de las placas de desarrollo STEVAL-STINKT 1B y la placa STEVAL-STLCS01V1 y STLCR01V1:

El firmware desarrollado con estas placas es muy similar al desarrollado en la etapa versión 1.0 por lo que las ideas principales pueden revisarse en la explicación del firmware de esta etapa.

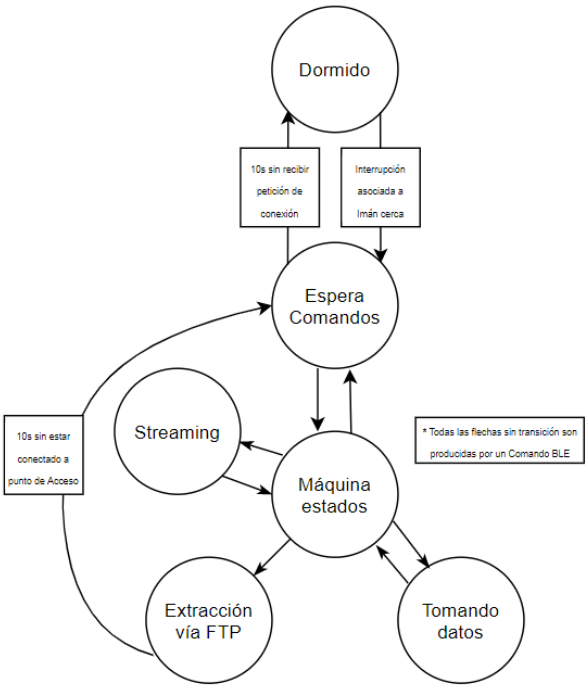



PERIPHERAL BLE
UART_TX
Byte received
Proceso mensaje
¿Pertenece a comando JSON?
NO
Descarto mensaje
SI
¿Mensaje JSON completado?
SI
Ejecuta comando
NO
Fin de tarea
Comando
Check JSON command
Poner hora y fecha
Poner en hora
Obtener Bateria fecha, hora o estado
Devolver Bateria fecha, hora o estado
Cambiar estado
Establecer periodo
Establecer periodo y devolver nuevo periodo por RX
Cambiar estado y devolver por RX nuevo estado
Fin de tarea

Anexo IV

Explicación Firmware etapa validación hipótesis

Gracias al framework de Arduino solamente se necesitaron 2 semanas para el desarrollo del firmware. Como entorno de trabajo se utilizó PlatformIO que permitía utilizar tanto el framework de Espressif como el de Arduino, añadiendo diferencias con el IDE de Arduino ya que te permite abrir declaraciones y explorar funciones. A nivel de Firmware se trabajó con FreeRTOS con el modo *preemption* activado, y constaba de 5 tareas.



La primera de estas tareas tarea recogía la información de los sensores, la segunda tarea procesaba un grupo de datos almacenados en un vector circular, la tercera tarea guardaba los datos en la tarjeta SD, la cuarta tarea gestiona una consola UART BLE (Bluetooth Low energy). Con ella se gestiona el modo de funcionamiento del sensor ("ESPERA_COMANDOS", "TOMANDO_DATOS", "EXTRACCIÓN", "DORMIDO") dependiendo de la función que se deseaba que realizara el nodo-sensor.

El modo "ESPERA_COMANDOS" dejaba al nodo en modo de bajo consumo "light-sleep", con lo que el sistema reducía su consumo manteniendo la antena de radio BLE encendida para esperar nuevas conexiones o instrucciones en caso de que ya existiera una conexión previa.

El modo "TOMANDO_DATOS" tenía como función el tomar los datos en crudo y guardarlos en la tarjeta SD para posteriormente ser extraídos.

El modo "EXTRACCIÓN" deshabilita la antena de radio usada para BLE y pasa a utilizarla para generar una red WiFi. Dentro de esa red WiFi el nodo sensor se comporta como un servidor FTP en el que se encuentran todos los archivos que contiene en su tarjeta SD (donde están los archivos con los datos). De esta forma, haciendo uso de un cliente FTP podemos descargarlos. Transcurridos 10 segundos desde que se generó la red WiFi o

tras la desconexión de todos los usuarios que se encontraran dentro de la WiFi, se reinicia el microcontrolador para volver al estado de Espera de comandos. Los parámetros de conexión a esta WiFi son almacenados dentro la memoria Flash del microcontrolador y son configurables, por lo que siempre que se haga un cambio en estos parámetros tras encender el microcontrolador se mantendrán.

Finalmente, el modo "DORMIDO", pone al microcontrolador en modo de muy bajo consumo, dejando solamente encendido el módulo RTC. Para despertar al microcontrolador es necesario hacer uso de un imán, puesto que se conectó a un sensor HALL, uno de los pines con capacidad de despertar el microcontrolador. De esta forma el microcontrolador podría ser despertado sin necesidad de acceder a dentro de la caja en la que se resguardaba.

3.2.3 - Firmware Tareas

Las tareas implementadas fueron:

Tarea 1- Leer sensores: En esta tarea se realizaba la lectura del acelerómetro, giroscopio y magnetómetro. Los datos se dejan en una cola para ser extraídos posteriormente.

Tarea 2- Procesado de los datos: Tras extraer de la cola los datos guardados estos son procesados en ventanas aplicando el procesado necesario.

Tarea 3- Guardar en SD: En esta tarea se guarda dentro de la tarjeta SD los datos procesados en un archivo **"dd_mm_yy_xxxx_sen.csv".**

Tarea 4- Levantar_server_ftp: Esta tarea se encarga de desactivar el BLE del ESP32 (ya que tienen la antena de radio y WiFi compartida), para seguidamente levantar un Punto de Acceso (AP) WiFi. Tras crear la red WiFi, el ESP32 se configura como servidor FTP desde la tarjeta SD, permitiendo a otro clientes FTP el acceder a cualquier archivo de la SD.

En caso desde que pasaran más de 25 segundos desde que se creócreo la WiFi o que todos los usuarios se desconectaran de la red WiFi esta tarea lleva al microcontrolador a modo dormido.

Tarea 5- Tarea Aplicación: Esta tarea se encarga de gestionar una máquina de estados con la que se modifica el estado del microcontrolador y por consiguiente las tareas que realiza. Para que se lance esta tarea es necesario una interrupción de BLE que tras ser procesada en la ISR (*Interrupt Service Routine*) se verifique que se ha producido un cambio de estado.

En el ANEXO II se encuentra el esquema del firmware realizado.

Anexo V

Consumos ESP32

El estudio teórico durante un funcionamiento normal en el que se muestreara a 50Hz con una distribución de 7ms para la toma y guardado de los datos mediante polling y de 13 ms de reposo, donde sólo se contempla el estado de "Toma de datos" debido a que sería el predominante da lugar a un consumo de 416J/h. En este estado las dos principales actividades a realizar son estar en reposo y la toma de datos. En la Figura 47 se puede ver la potencia consumida en miliwatios en todos los estados (no sólo en la toma de datos) para poder contemplar otros escenarios en los que el estado predominante no fuera la toma de datos.

	Nombre	Consumo(mW)
Estado 1	Espera de comandos	310,8792
Estado 2	Tomando datos	323,769
Estado 3	Extracción	444,0792
Estado 4	Streaming	419,969
Estado 5	Dormido	0,1162
Estado 6	Reposo	3,7792

Figura 47. Consumos de potencia por estados.

A la frecuencia de muestreo de 50 Hz y con una distribución de 7ms para la toma y guardado de los datos mediante polling y de 13 ms de reposo (ver Figura 48).

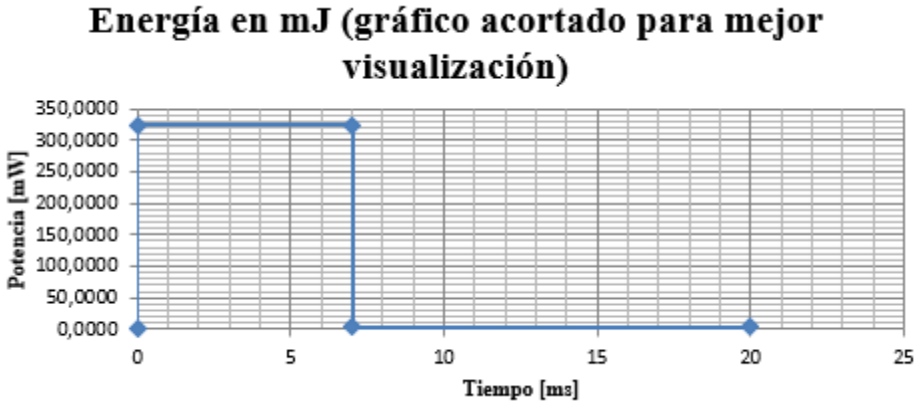


Figura 48

Supone un consumo por ciclo y por hora algo elevados para una prueba de 70 días.

Suma mJ / 1ciclo	Suma mJ/ms	suma J/h
2,3155	0,1158	416,792268

Figura 49

De hecho este sistema se desarrolló para tener autonomía de más de 1-2 días para poder hacer sesiones de 1-2h sin tener que cargar la batería:

Tipo de batería	Nº dias	Energía disponible [J]	Capacidad [mA·h]	Voltaje [V]	Precio [Eur]	Peso [g] / Dimensiones [mm]	Columna1
Ion Litio	1,0653	10656	800	3,7	6,85	20.5 / D14.2x49	KCQjwwLKFBhDPARI
Polímero de litio	2,40	23976	1800	3,7	11,90	34 / 9,7x55	-pilas-industriales/baterias

Figura 50

Aunque dada la eficiencia del conversor step down del 90% para corrientes de 80mA de salida (aproximadamente máxima corriente entregada durante el estado de toma de datos) y de un 85% durante el repososo, el sistema dura entre 1 y dos días.

Realidad	
Tipo de batería	**Nº dias**
Ion Litio	0,9588
Polímero de litio	2,1572

Figura 51

Anexo VI

Herramientas software para el desarrollo del firmware en microcontroladores ST

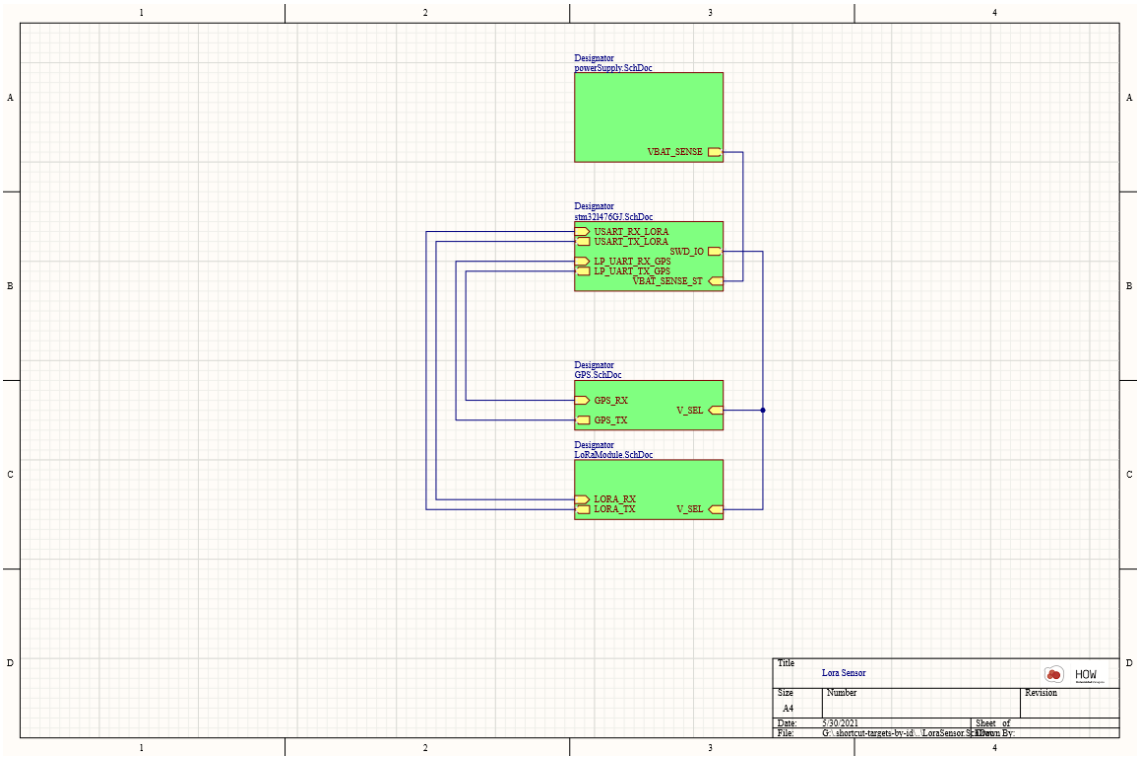
Las herramientas de trabajo de las que se hace uso en la elaboración del firmware de los microcontroladores de ST son: el entorno STM32CubeIDE versión 1.4.0. Dentro de este IDE, se trabajó en C# con las capas HAL V4.1.0 de ST liberada el 26 mayo del 2020 desarrolladas para el microcontrolador STM32L4R9. El firmware desarrollado interactúa con los sensores de ST integrados en placa por medio del paquete de software STMicroelectronics.X-CUBE-MEMS versión 1.8.1.1, utiliza CMSIS-RTOS en su versión 2.0 para la trabajar sobre un sistema operativo de tiempo real y el paquete de expansión X-CUBE-AI versión 6.0, para la implementación del modelo de Machine Learning dentro del microcontrolador. Para gestionar los comandos SPI entre microcontrolador y módulo BLE se hizo uso del paquete de software STM32CubeExpansion_BLE1_V6.1.0 liberado por ST el 3 de noviembre del 2020. Para la gestión de la tarjeta SD se hace uso del paquete de código abierto FatFs - Generic FAT file system module R0.12c. Finalmente se hace uso del paquete software Parson 1.1.0 de Krzysztof Gabis para el parseo de comandos JSON [*31*].

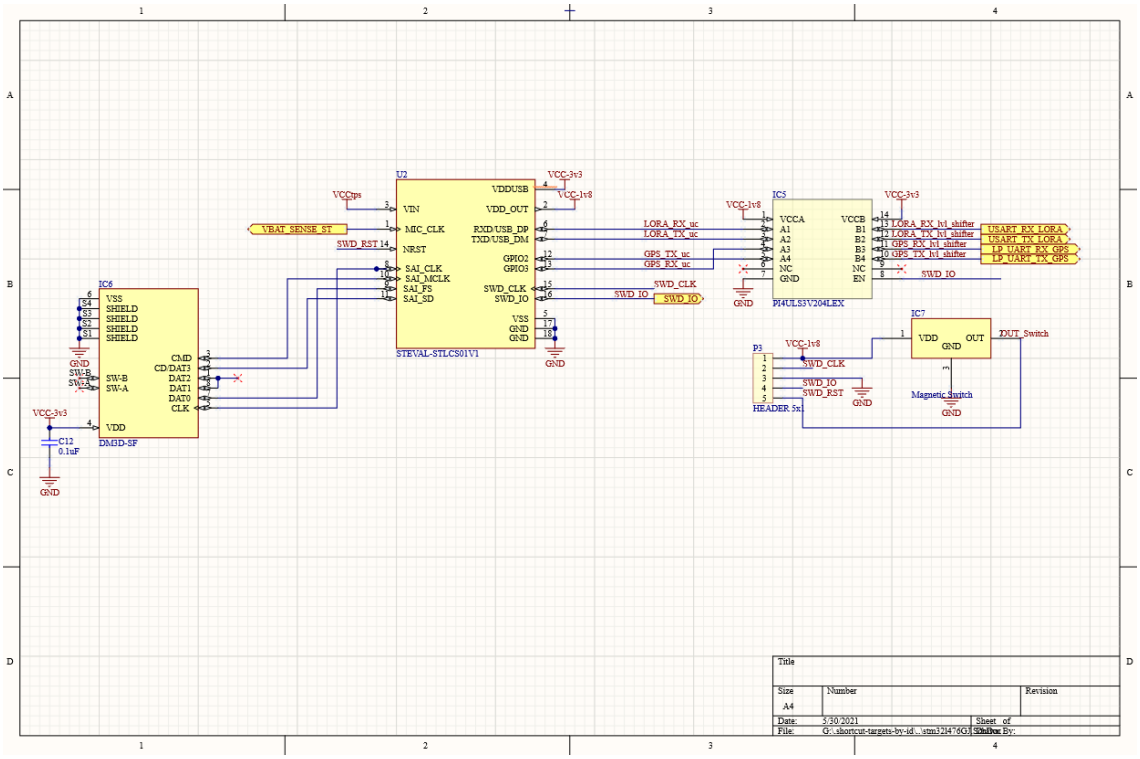
Anexo VII

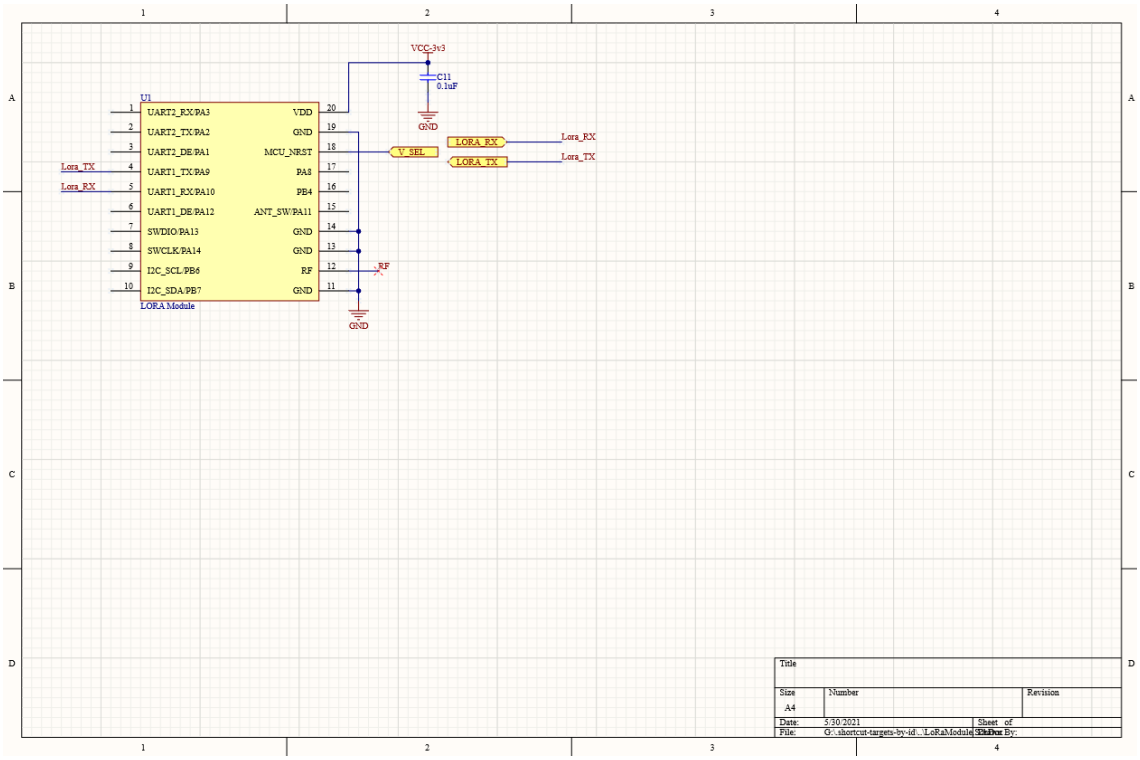
Esquemático ST_SHEEP_V0

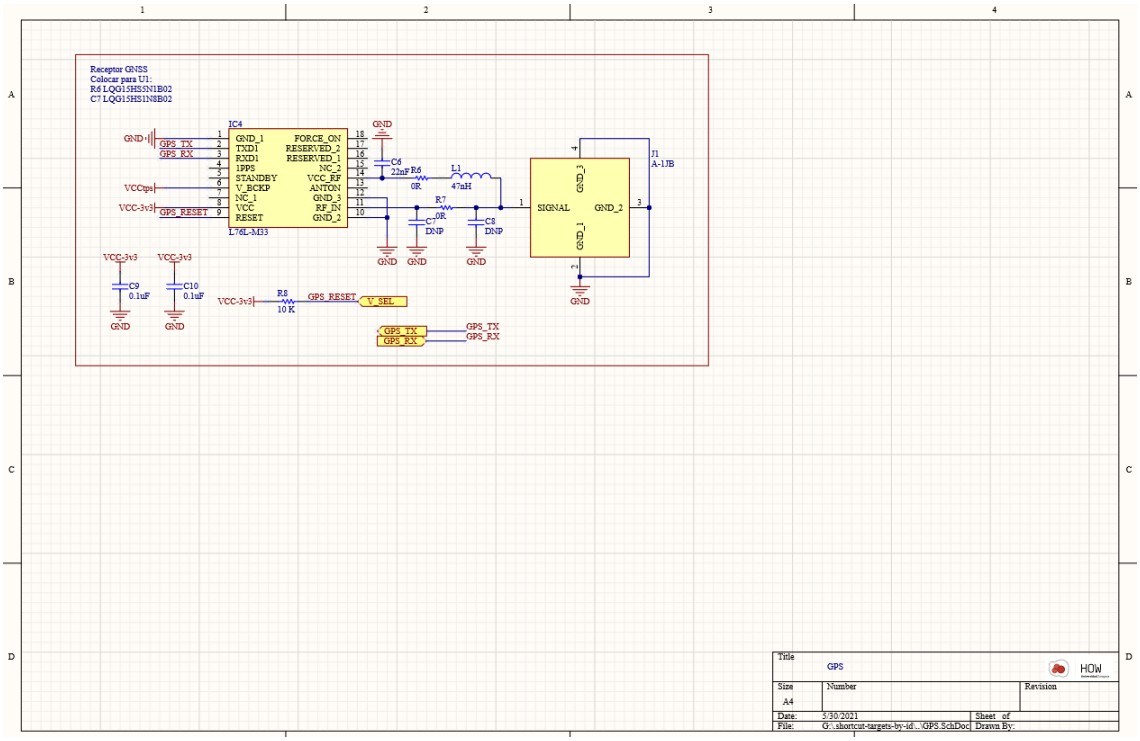
Durante la Etapa de versión 1.0 se diseña una placa que funcionará como un escudo para la placa sensor tile utilizada en la etapa de optimización. En este anexo se recoge el esquemático de esta placa.

Diagrama de bloques global

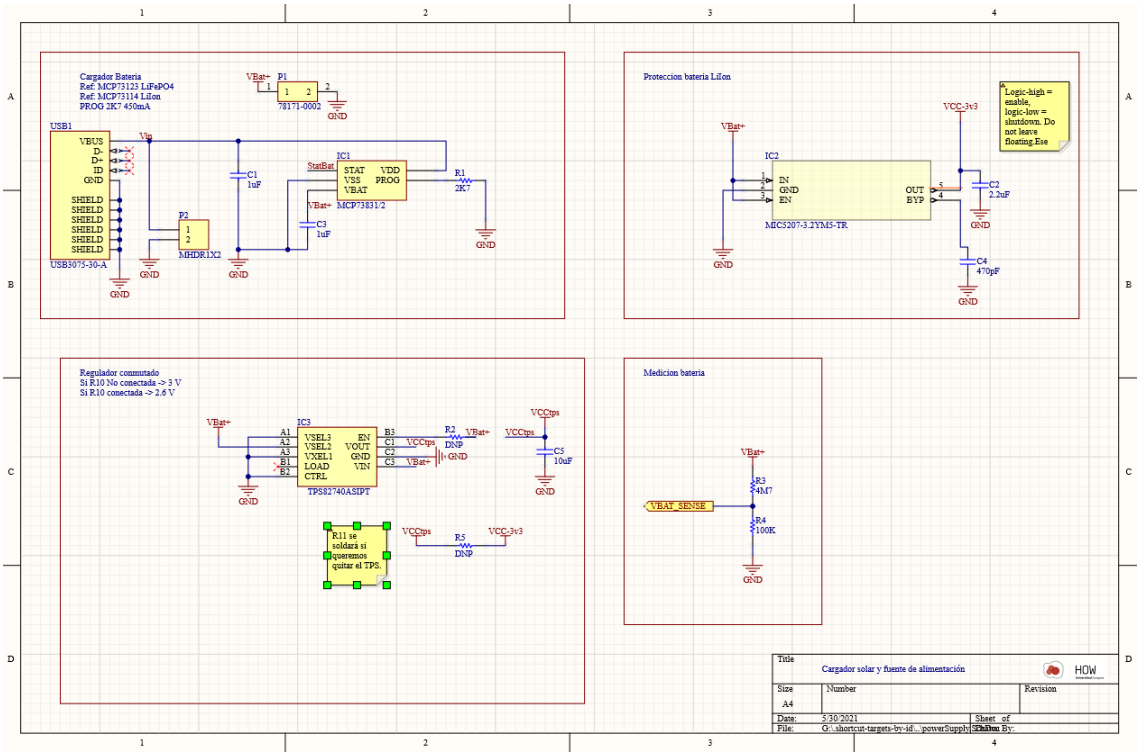



U2
STEVAL-STLCS01V1
VDDUSB
VDD_OUT
VIN
MIC_CLK
NRST
RXD/USB_DP
TXD/USB_DM
GPIO2
GPIO3
SAI_CLK
SAI_MCLK
SAI_FS
SAI_SD
SWD_CLK
SWD_IO
VSS
GND
VCC-3v3
VCC-1v8
VCC_ps
VBAT_SENSE_ST
SWD_RST
LORA_RX_uc
LORA_TX_uc
GPS_TX_uc
GPS_RX_uc
IC5
PI4ULS3V204LEX
VCCA
VCCB
A1
A2
A3
A4
NC
EN
B1
B2
B3
B4
LORA_RX_lvl_shifter
LORA_TX_lvl_shifter
GPS_RX_lvl_shifter
GPS_TX_lvl_shifter
USART_RX_LORA
USART_TX_LORA
LP_UART_RX_GPS
LP_UART_TX_GPS
IC6
DM3D-SF
VSS
SHIELD
CMD
CD/DAT3
DAT2
DAT1
DAT0
CLK
SW-B
SW-A
VDD
C12
0.1uF
IC7
VDD
GND
OUT
ZDUT_Switch
Magnetic Switch
P3
HEADER 5x1
Title
Size
A4
Number
Revision
Date:
5/30/2021
Sheet of
File:
Drawn By:

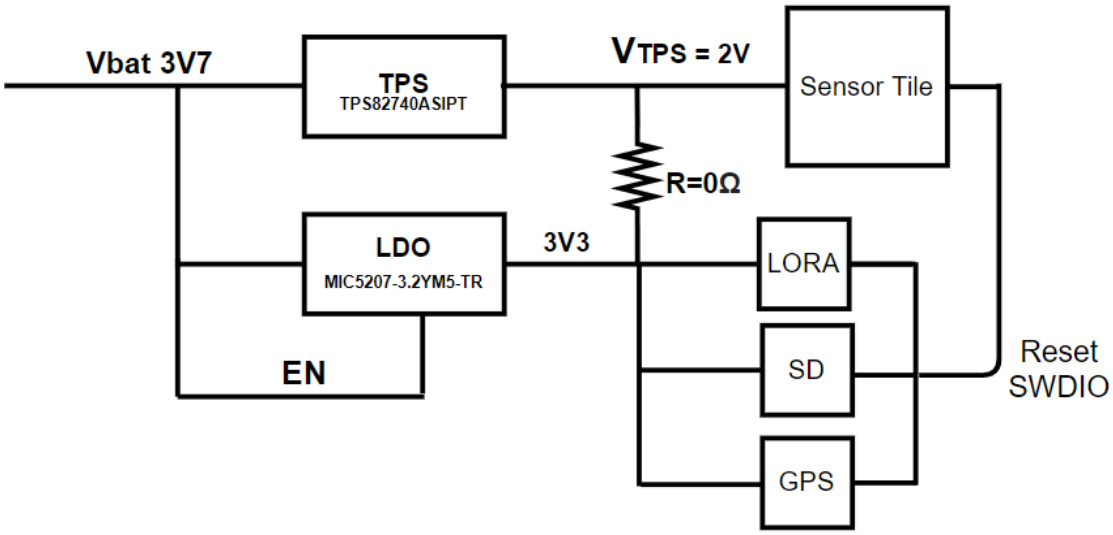

U1
LORA Module
UART2_RX/PA3
UART2_TX/PA2
UART2_DE/PA1
UART1_TX/PA9
UART1_RX/PA10
UART1_DE/PA12
SWDIO/PA13
SWCLK/PA14
I2C_SCL/PB6
I2C_SDA/PB7
VDD
GND
MCU_NRST
PA8
PB4
ANT_SW/PA11
RF
VCC-3v3
C11
0.1uF
V_SEL
LORA_RX
LORA_TX
Lora_RX
Lora_TX
Title
Size
A4
Number
Revision
Date:
5/30/2021
Sheet of
File:
Drawn By:



Alimentación de la placa



El diagrama de bloques de la parte de alimentación es el siguiente:


Vbat 3V7
TPS
TPS82740ASIPT
VTPS = 2V
Sensor Tile
R=0Ω
LDO
MIC5207-3.2YM5-TR
3V3
LORA
SD
GPS
EN
Reset
SWDIO

Anexo VIII

Hardware utilizado a lo largo de las etapas del proyecto:

	ETAPA 1 - VALIDACIÓN DE HIPÓTESIS	**ETAPA 2 - VALIDACIÓN FUNCIONALIDADES**	**ETAPA3 - OPTIMIZACIÓN**	**ETAPA 4 - VERSIÓN 1**
	TTGO- 2 - V2	**STEVAL-STWINKT1B**	**STEVAL-STLCS01V1 & STLCR01V1**	**ST_SHEEP_V0**
Microcontrolador	ESP32 - WROOM-32: 4-Mbyte Flash (SPI) 448KB ROM 520KB SRAM 8KB SRAM in RTC	STM32L4R9: 2-Mbyte Flash 640 Kbytes of SRAM	STM32L476 1-Mbyte Flash 128KB SRAM	STM32L476 1-Mbyte Flash 128KB SRAM
CPU	2 low-power Xtensa 32-bit LX6	Ultra-low-power ARM Cortex-M4 MCU at 120 MHz with FPU, 2048 kbytes Flash memory	Arm 32-bit Cortex-M4 CPU with FPU - 80 MHz	Arm 32-bit Cortex-M4 CPU with FPU - 80 MHz
Sensores	SY8089 LDO Pantalla LCD (C96_64_SSD1331) Hall Sensor	3-axis digital vibration sensor (IIS3DWB), 3D accelerometer + 3D Gyro IMU (ISM330DHCX) - ML core, ULP MEMS motion sensor (IIS2DH), ULP 3-axis magnetometer (IIS2MDC), pressure sensor (LPS22HH), humidity - temperature sensor (HTS221), low-voltage temperature sensor(STTS751), MEMS microphone (IMP34DT05)	MP34DT05-A - Microphone LSM6DSM iNEMO IMU: 3D accelerometer and 3D gyroscope LSM303AGR ultra-low power 3D accelerometer and 3D magnetometer LPS22HB MEMS nano pressure sensor	MP34DT05-A - Microphone LSM6DSM iNEMO IMU: 3D accelerometer and 3D gyroscope LSM303AGR ultra-low power 3D accelerometer and 3D magnetometer LPS22HB MEMS nano pressure sensor L76L-M33 GPS Sensor RAK4270 LoRa Sensor

MicroSD	Sí	Sí	Sí	Sí
RF communications	BLE y WiFi (on-chip)	BLE (BlueNRG) y WiFi (STEVAL-STWINWFV1)	BLE (BlueNRG-MS)	BLE (BlueNRG-MS)
Alimentación	Step down DCDC Converter - SY8089 Boost DCDC Converter - LM2733	LDO LDK130 - (1-30uA = I quiescient @ 100mV Dropout voltage)	LDO - LD39115J (35uA = Iquiescient @ 110mV Dropout voltage) LDO LDK120- (1-100uA = I quiescient @ 100mV Dropout voltage)	LDO - MIC5207-3.2YM5-TR (1-5uA = Iquiescient @ 80--400mV Dropout voltage) LDO LDK120- (1-100uA = I quiescient @ 100mV Dropout voltage) StepDown DCDC Converter - TPS82740BSIPT
Comunicaciones	- 3 x UARTS - 4 x SPI - 2 × I^2C - 2 × I^2S - 1 x IR - 1 x Two-Wire Automotive Interface	– USB OTG 2.0 full-speed, LPM and BCD – 2x SAIs (serial audio interface) – 4x I2C FM+(1 Mbit/s), SMBus/PMBus – 6x USARTs (ISO 7816, LIN, IrDA, modem) – 3x SPIs (5x SPIs with the dual OctoSPI) – CAN (2.0B Active) and SDMMC	– USB OTG 2.0 full-speed, LPM and BCD – 2x SAIs (serial audio interface) – 3x I2C FM+(1 Mbit/s), SMBus/PMBus – 5x USARTs (ISO 7816, LIN, IrDA, modem) – 1x LPUART (Stop 2 wake-up) – 3x SPIs (and 1x Quad SPI) – CAN (2.0B Active) and SDMMC interface – SWPMI single wire protocol master I/F – IRTIM (Infrared interface	– USB OTG 2.0 full-speed, LPM and BCD – 2x SAIs (serial audio interface) – 3x I2C FM+(1 Mbit/s), SMBus/PMBus – 5x USARTs (ISO 7816, LIN, IrDA, modem) – 1x LPUART (Stop 2 wake-up) – 3x SPIs (and 1x Quad SPI) – CAN (2.0B Active) and SDMMC interface – SWPMI single wire protocol master I/F – IRTIM (Infrared interface

ADC	- 12-bit SAR ADC up to 18 channels - 2 × 8-bit DAC - 10 × touch sensors	– 12-bit ADC 5 Msps, up to 16-bit with hardware oversampling, 200 µA/Msps – 2x 12-bit DAC, low-power sample and hold – 2x operational amplifiers with built-in PGA – 2x ultra-low-power comparators	– 3x 12-bit ADC 5 Msps, up to 16-bit with hardware oversampling, 200 µA/Msps – 2x 12-bit DAC output channels, low-power sample and hold – 2x operational amplifiers with built-in PGA – 2x ultra-low-power comparators	– 3x 12-bit ADC 5 Msps, up to 16-bit with hardware oversampling, 200 µA/Msps – 2x 12-bit DAC output channels, low-power sample and hold – 2x operational amplifiers with built-in PGA – 2x ultra-low-power comparators
Modos LP del uC	- Modem-sleep: The CPU is powered o-> 30 mA - 68 mA - Light-sleep -> 0.8 mA - Deep-sleep ULP co-processor is powered on -> 150 µA ULP sensor-monitored pattern -> 100 µA RTC timer + RTC memory -> 10 µA - Hibernation RTC timer only -> 5 µA - Power off The chip is powered off -> 1 µA	– 305 nA in VBAT mode: supply for RTC and 32x32-bit backup registers – 33 nA Shutdown mode (5 wakeup pins) – 125 nA Standby mode (5 wakeup pins) – 420 nA Standby mode with RTC – 2.8 µA Stop 2 with RTC – 110 µA/MHz Run mode (LDO mode) – 43 µA/MHz Run mode (@ 3.3 V SMPS mode) – 5 µs wakeup from Stop mode – Brownout reset (BOR) in all modes except shutdown	- 300 nA in VBAT mode: supply for RTC and - 32x32-bit backup registers – 30 nA Shutdown mode (5 wakeup pins) – 120 nA Standby mode (5 wakeup pins) – 420 nA Standby mode with RTC – 1.1 µA Stop 2 mode, 1.4 µA with RTC – 100 µA/MHz run mode (LDO Mode) – 39 µA/MHz run mode (@3.3 V SMPS Mode) – Batch acquisition mode (BAM)	- 300 nA in VBAT mode: supply for RTC and - 32x32-bit backup registers – 30 nA Shutdown mode (5 wakeup pins) – 120 nA Standby mode (5 wakeup pins) – 420 nA Standby mode with RTC – 1.1 µA Stop 2 mode, 1.4 µA with RTC – 100 µA/MHz run mode (LDO Mode) – 39 µA/MHz run mode (@3.3 V SMPS Mode) – Batch acquisition mode (BAM)
Precio	≈11€	≈110€	≈86€	≈60€

Tabla 8: Tabla comparativa de microcontroladores y placas utilizados por etapa del proyecto.

Anexo IX

Problemas durante la sesión de datos

Los principales problemas existentes durante las sesiones de adquisición, en consideración en las sesiones posteriores, han sido la falta de tiempo para acostumbrarse, la posición de la cámara, actividades poco frecuentes y los periodos de baja actividad. Estos problemas se detallan a continuación:

1- Falta de tiempo de acostumbrarse:
Se ha observado que durante los primeros 15 - 20 min las ovejas que no tienen collar identifican el collar como un elemento nuevo con el que interactuar por lo que tienden a lamerlo y morderlo como se aprecia en la imagen inferior con las dos ovejas de la izquierda. Este problema da lugar a datos que deben ser descartados durante el entrenamiento de la red neuronal ya que no aportan información.

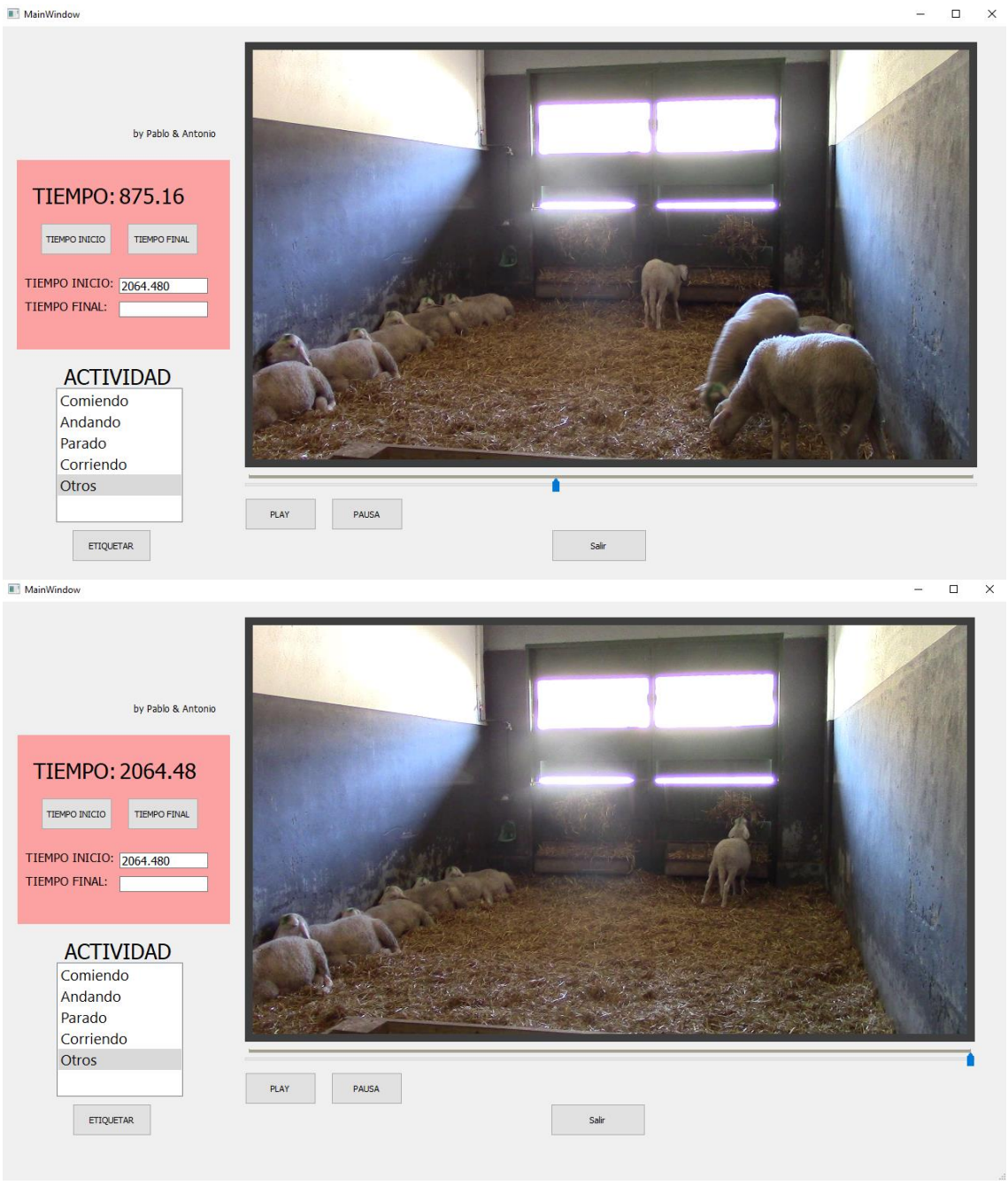
2- Posición de la cámara:
Durante las sesiones de adquisición se ha utilizado el comedero como mecanismo de separación entre las ovejas y la cámara. Este elemento puede tener el inconveniente de que aunque no haya comida las ovejas se acerquen a él perdiendo la visión de la oveja por el ángulo de la cámara.

2- Actividades que puedan dar lugar a error:

La definición de las actividades de forma precisa es esencial para que no existan problemas durante el etiquetado, ya que existen actividades que pueden realizarse conjuntamente y sólo una buena definición de las tareas puede ayudar a determinar cuál de las dos se trata.

3- Periodos de baja actividad:

La selección de la hora de trabajo es esencial ya que los animales tienen periodos de baja actividad que da lugar a que los datos tomados sean monótonos y referentes a bajos niveles de energía. Por ejemplo, en una sesión las ovejas estuvieron desde el segundo 875 de video hasta el segundo 2064 descansando, por lo que los datos tuvieron poca variedad.


MainWindow
by Pablo & Antonio
TIEMPO: 875.16
TIEMPO INICIO
TIEMPO FINAL
TIEMPO INICIO: 2064.480
TIEMPO FINAL:
ACTIVIDAD
Comiendo
Andando
Parado
Corriendo
Otros
ETIQUETAR
PLAY
PAUSA
Salir
MainWindow
by Pablo & Antonio
TIEMPO: 2064.48
TIEMPO INICIO
TIEMPO FINAL
TIEMPO INICIO: 2064.480
TIEMPO FINAL:
ACTIVIDAD
Comiendo
Andando
Parado
Corriendo
Otros
ETIQUETAR
PLAY
PAUSA
Salir

Anexo X

Curvas de caracterización de consumo

1.Modo de mínima energía

El procedimiento para caracterizar el consumo más bajo del dispositivo ha consistido en 5 pasos:

1. Se recoge los consumo experimentales que ha proporcionado el fabricante de los sensores y de diferentes circuitos integrados que tiene la placa, en sus diferentes modos de funcionamiento.

2. Desarrollar un firmware que active todos los sensores y módulos de la placa y los vaya apagando progresivamente, de uno en uno, de forma controlada. Se pueden utilizar pines digitales para saber que componente es el que se acaba de apagar.

3. Haciendo uso de un multímetro precisión, ejecutar el código e ir comprobando como se parte de una situación de alto consumo y esta se va reduciendo progresivamente.

4. Durante esta etapa se anota la reducción de consumo que experimenta el dispositivo al apagar cada componente.

5. Finalmente, observar el mínimo consumo obtenido y compararlo con el que teóricamente, según el fabricante, debiera salir. En caso de ser parecido (puede haber desviaciones respecto al dato típico de consumo) se concluye la prueba conociendo el estado de menor energía.

En el caso concreto de la placa desarrollada en la versión V1.0 se parte con la siguiente configuración:

Acelerómetro y giroscopio se inician en modo *High Power* a 0.65mA.

Acelerómetro + Giroscopio	Power Down	Low Power	Normal mode	High Power
Acelerometro	3uA	ODR=52Hz - 25uA	0,4mA	0,65mA
Giroscopio		ODR=52Hz - 290uA		

El micrófono se inicia en modo de funcionamiento a 700uA.

Micrófono - MP34DT04	Modo Apagado	Modo de funcionamiento
	1uA	700uA

El sensor HAL tiene una corriente de fuga de 1uA.

Sensor Hall	
Corriente Fugas	1uA

El regulador de la sensor tile que pasa a los 1v8 siempre se quedará encendido durante la prueba, ya que en caso de apagarse el microcontrolador se quedaría sin alimentación.

Regulador 1V8	Ld39115
Corriente Shutdown	1uA
Corriente Quiescient Il = 0 - 150mA	20uA - 70uA

El magnetómetro y el acelerómetro se inician sumando 1.2mA

Acelerómetro +Magnetómetro	Power Down	High Power
Acelerometro	2uA	ODR = 1344Hz - 185uA
Magnetómetro		ODR = 100Hz - 1130uA

El módulo BLE se inicia en modo low power consumiendo 7uA.

Una vez determinado el consumo inicial, se van llevando a modo de bajo consumo a cada componente hasta ver una corriente de alrededor de 29uA , que podría estar dentro de los posibles valores obtenidos ya que se suman los 20uA del LDO + 1uA del sensor Hall + 1 uA del micrófono +3uA del acelerómetro y giroscopio + 2uA del acelerómetro y magnetómetro = 27uA.
En la siguiente figura puede observarse como se ha logrado esta medida. Ver Figura 52.

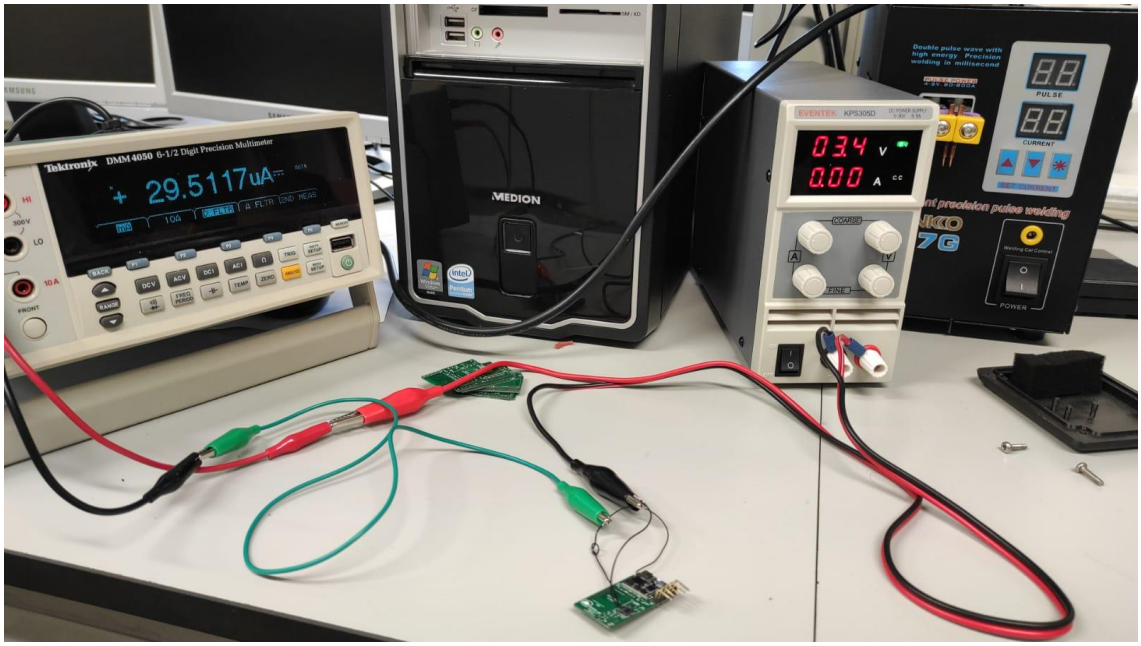


Figura 52

1.Otros modos de consumo

Para la caracterización de los consumos de los sensores se ha seguido la siguiente metodología:
Poner al microcontrolador en el modo de menor consumo y lanzar los sensores o el módulo para que empiecen a consumir.

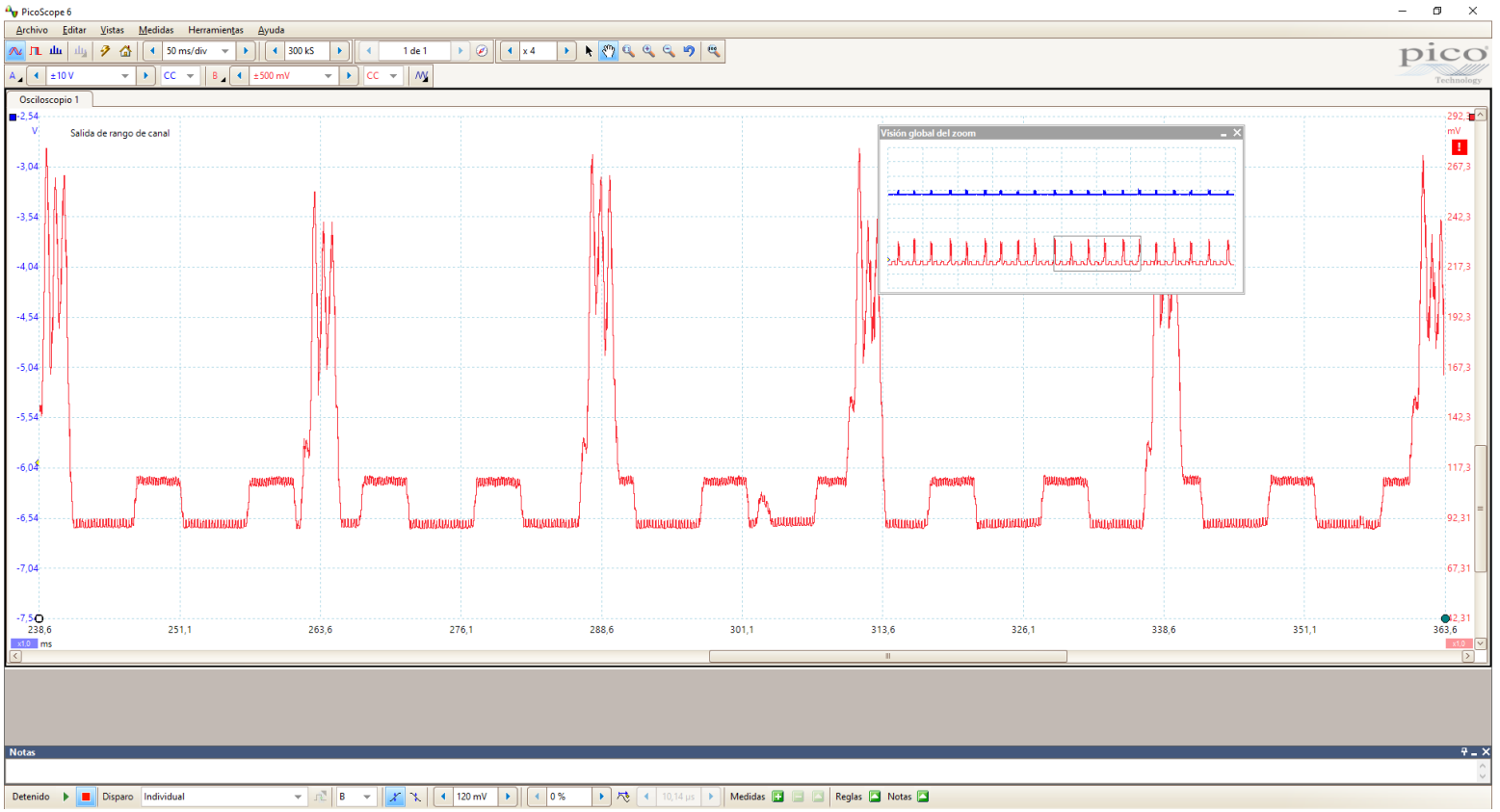
Ahora utilizando una resistencia de valor conocido de 10 ohmios se pone en serie con el micrcontrolador y se mide con un osciloscopio el voltaje que cae en esta resistencia.

Conocido este voltaje, se puede calcular la corriente que demanda el sistema por medio de la ley de ohm.

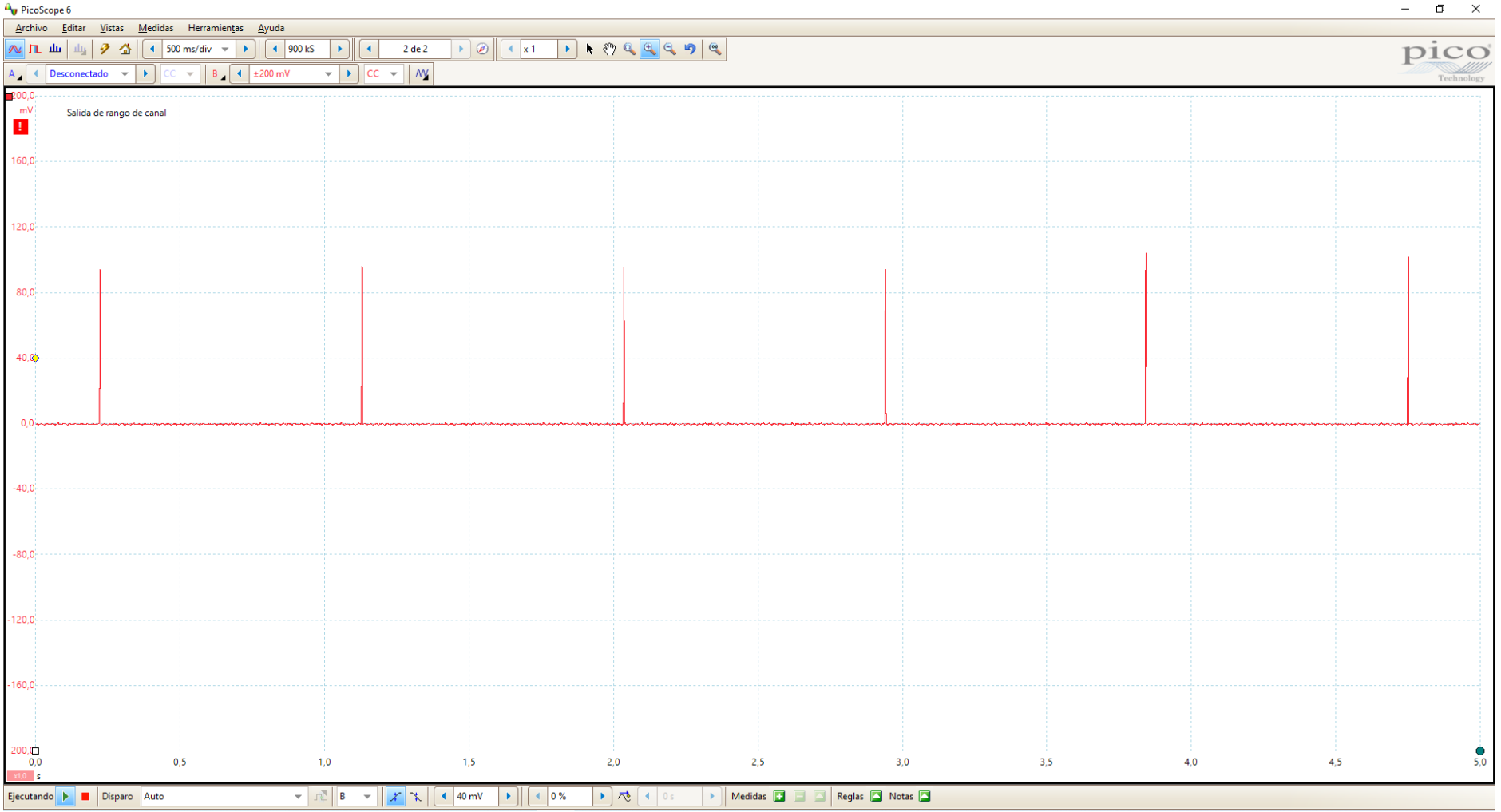
A continuación, se muestran las gráficas de consumo de los sensores, del BLE y del microcontrolador con los sensores y BLE.

Prueba 1: Caracterización del consumo con microcontrolador encendido, BLE activo y sensores activados.

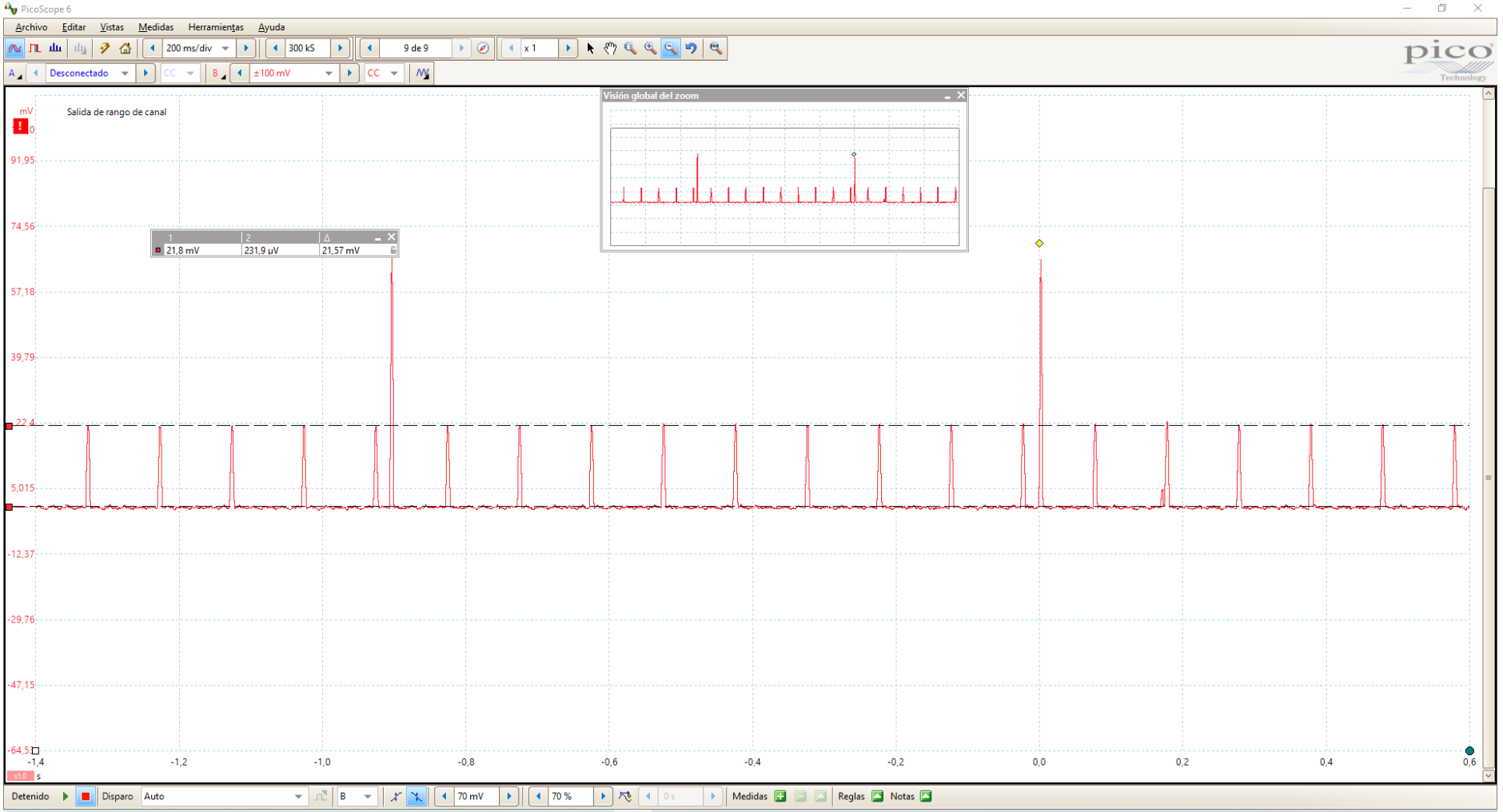
```
ret = aci_gap_set_discoverable(ADV_IND, 0x0020, 0x0020, RANDOM_ADDR, NO_WHITE_LIST_USE, sizeof(local_name), local_name, 0, NULL, 0, 0);
```



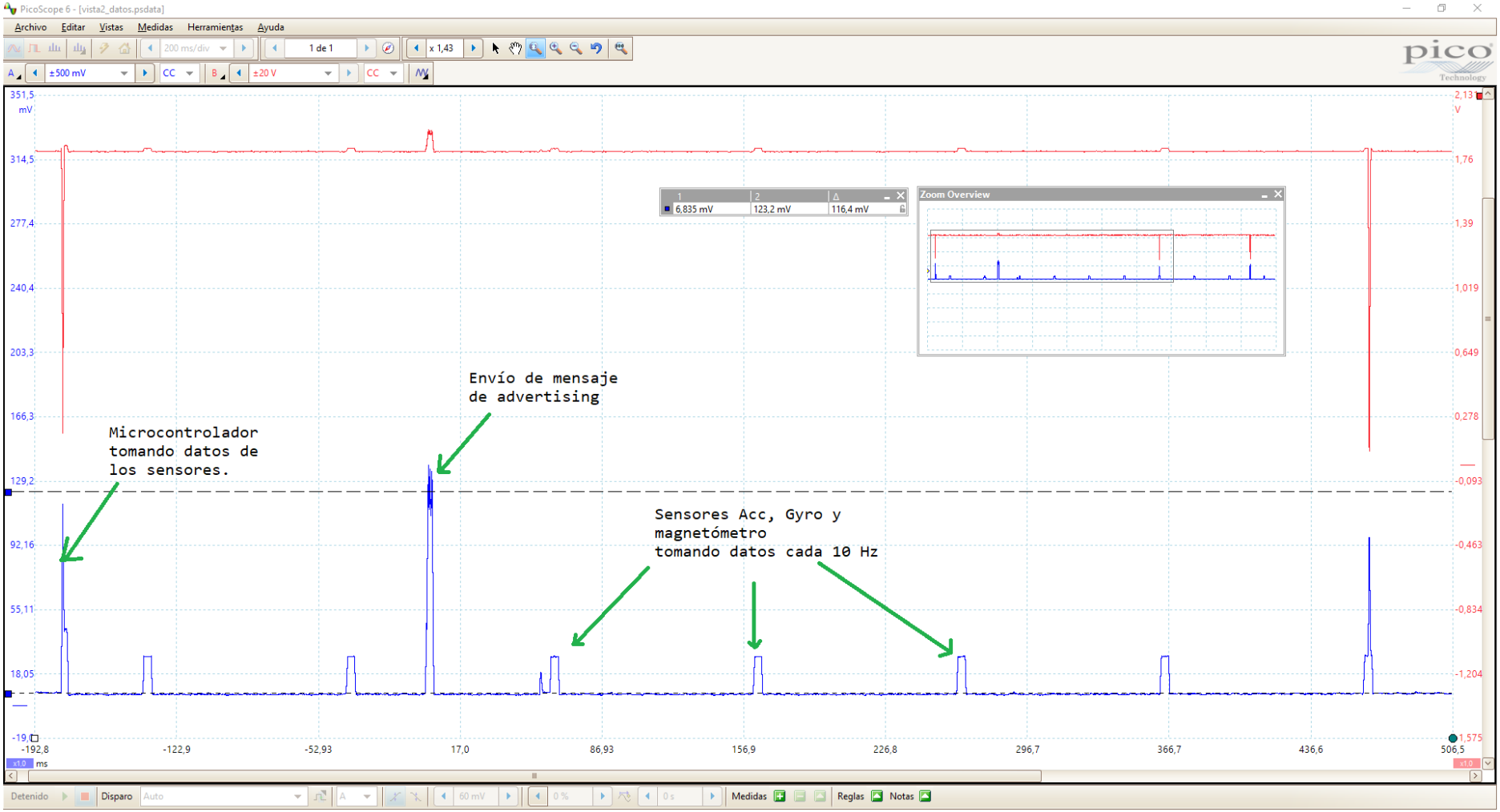
Prueba 3 : Sin sensores, sólo BLE haciendo advertising y modo shutdown para el microcontrolador.
Intentamos que vaya cada 900ms. Resultado correcto. 0x5A0 * 0.625 = 900ms


PicoScope 6
Archivo Editar Vistas Medidas Herramientas Ayuda
500 ms/div
900 kS
2 de 2
x 1
Desconectado
±200 mV
pico Technology
Salida de rango de canal
100,0 mV
160,0
120,0
80,0
40,0
0,0
-40,0
-80,0
-120,0
-160,0
-200,0
0,0 0,5 1,0 1,5 2,0 2,5 3,0 3,5 4,0 4,5 5,0
s
Ejecutando
Disparo Auto
B
40 mV
0 %
Medidas
Reglas
Notas

Prueba 4: se mide BLE haciendo advertising y con los sensores muestreando cada 10 Hz= 100ms



Modo STOP2 del microcontrolador con los sensores muestreando cada 10Hz, con BLE haciendo advertising y el microcontrolador tomando datos de los sensores:



Anexo XI

Reporte RRNN

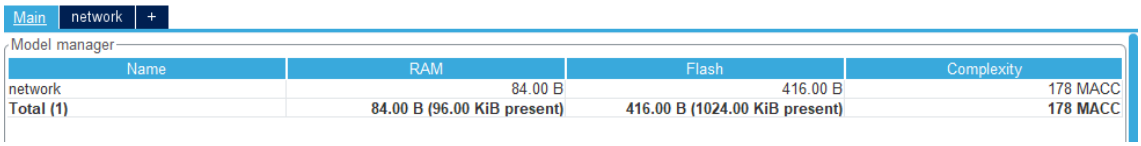
En este anexo se muestran los reportes generados automáticamente por el entorno de desarrollo STM32CUBEMX.

```
c_name            : network
compression       : None
quantize          : None
workspace dir     : C:\Users\pablo\AppData\Local\Temp\mxAI_workspace285472130636800543708184282069173l
output dir        : C:\Users\pablo\.stm32cubemx

model_name        : new2_weights2707502
model_hash        : e7f92db8d1010d4532b3a17bacef36b1
input             : input_0 [3 items, 12 B, ai_float, FLOAT32, (1, 1, 3)]
inputs (total)    : 12 B
output            : dense_2_nl [4 items, 16 B, ai_float, FLOAT32, (1, 1, 4)]
outputs (total)   : 16 B
params #          : 104 items (416 B)
macc              : 178
weights (ro)      : 416 B (416 B)
activations (rw)  : 56 B (56 B)
ram (total)       : 84 B (84 B) = 56 + 12 + 16

Model name - new2_weights2707502 ['input_0'] ['dense_2_nl']
------------------------------------------------------------------------------------------------------------------------
id   layer (type)                shape    param/size   macc   connected to   |   c_size   c_macc   c_type
------------------------------------------------------------------------------------------------------------------------
0    input_0 (Input)             (c:3)                                       |
     dense (Dense)               (c:10)   40/160       40     input_0        |                      dense()[0]
     dense_nl (Nonlinearity)     (c:10)                 10     dense          |                      nl()[1]
------------------------------------------------------------------------------------------------------------------------
2    dense_1 (Dense)             (c:4)    44/176       44     dense_nl       |                      dense()[2]
     dense_1_nl (Nonlinearity)   (c:4)                  4      dense_1        |                      nl()[3]
------------------------------------------------------------------------------------------------------------------------
4    dense_2 (Dense)             (c:4)    20/80        20     dense_1_nl     |                      dense()[4]
     dense_2_nl (Nonlinearity)   (c:4)                  60     dense_2        |                      nl()/o[5]
------------------------------------------------------------------------------------------------------------------------
model/c-model: macc=178/178  weights=416/416  activations=--/56 io=--/28

Complexity report per layer - macc=178 weights=416 act=56 ram_io=28
------------------------------------------------------------------------
id   name         c_macc                    c_rom                     c_id
------------------------------------------------------------------------
0    dense        ||||||||||||      22.5%   ||||||||||||||||  38.5%   [0]
0    dense_nl     |||                5.6%   |                  0.0%   [1]
2    dense_1      ||||||||||||      24.7%   ||||||||||||||||  42.3%   [2]
2    dense_1_nl   ||                 2.2%   |                  0.0%   [3]
4    dense_2      ||||||            11.2%   |||||||           19.2%   [4]
4    dense_2_nl   ||||||||||||||||  33.7%   |                  0.0%   [5]
Creating txt report file C:\Users\pablo\.stm32cubemx\network_analyze_report.txt
elapsed time (analyze): 0.616s
Analyze complete on AI model
```



Uso real de toda la API de ST:
Flash = 15516+3282=19344 bytes
RAM = 3828+4848= 8676 bytes