Universidad de la República
Facultad de Ingeniería

Sistema para la caracterización de la dinámica espacial en ovinos

Memoria de proyecto presentada a la Facultad de Ingeniería de la Universidad de la República por

María Victoria Campiotti, Nicolás Eduardo Finozzi,
Juan Diego Irazoqui

en cumplimiento parcial de los requerimientos
para la obtención del título de
Ingeniero Electricista.

Tutor

Julián Oreggioni ..... Universidad de la República

Tribunal

Álvaro Gómez ..... Universidad de la República
Juan Pablo Oliver..... Universidad de la República
Mariana Siniscalchi ..... Universidad de la República

Montevideo
miércoles 6 octubre, 2021

Sistema para la caracterización de la dinámica espacial en ovinos, María Victoria Campiotti, Nicolás Eduardo Finozzi, Juan Diego Irazoqui.

Esta tesis fue preparada en LaTeX usando la clase iietesis (v1.1).
Contiene un total de 152 páginas.
Compilada el miércoles 6 octubre, 2021.
http://iie.fing.edu.uy/

Agradecimientos

A nuestros padres, hermanos y familia en general por ser pilares fundamentales en nuestros avances académicos y personales.

A nuestros amigos, por su contención y apoyo incondicional. Además, en particular a Luciano y Tomás quienes nos dieron en más de una oportunidad de su tiempo y conocimiento para resolver de la mejor manera problemáticas propias del proyecto.

También queremos agradecer a nuestro tutor Julián por el seguimiento cercano, las críticas constructivas y evacuar todas nuestras dudas o, en caso de no poder hacerlo, conseguir el contacto de quien pudiera dar una mano. Por ello, también les agradecemos a todas estas personas que se tomaron un momento para darnos consejo.

A Rodolfo por darnos indicaciones precisas y consejos desde el punto de vista veterinario. Luego, más en general, a la Facultad de Veterinaria por prestar sus instalaciones para llevar a cabo las pruebas y a la CSIC junto con Antel por el apoyo financiero y tecnológico.

A todos los miembros del tribunal que evaluó el proyecto por su tiempo y las críticas en general que tendremos presentes en nuestro camino como ingenieros.

Finalmente, ya que una importante etapa de nuestras vidas finaliza, queremos agradecer a la Universidad de la República, en particular a la Facultad de Ingeniería y al Instituto de Ingeniería Eléctrica por ser el centro donde nos formamos y de donde nos llevamos vasta cantidad de experiencias y conocimiento.

Esta página ha sido intencionalmente dejada en blanco.

Resumen

Una causa importante de la baja eficiencia reproductiva del ganado ovino en nuestra región es la alta mortalidad de los corderos en las primeras horas después del parto. Predecir con cierta antelación el momento del parto le permitiría a los productores tomar acciones para prevenir este problema. Hay antecedentes que muestran que se producen cambios en la actividad de los ovinos en el período previo al parto, pero es un área todavía en investigación caracterizar adecuadamente esos cambios. Este trabajo se trata de un primer acercamiento a esta temática donde se desarrolló el prototipo de un Sistema para ser utilizado por investigadores en un área que reúne a veterinarios e ingenieros.

Se presenta el diseño, la fabricación y pruebas de funcionamiento del prototipo de un Sistema que es capaz de mostrar en tiempo real en la pantalla de un PC: la ubicación geográfica de un ovino, señales provenientes de un acelerómetro que caracterizan su actividad, y el Estado del animal (corriendo, caminando, quieto, o con la cabeza agachada).

El Sistema está compuesto por un dispositivo electrónico tipo collar, que llamaremos Equipo, que se coloca en un ovino y reporta la información adquirida a un Sistema Central. El Equipo está basado en el microcontrolador MSP-EXP432P401R y el sensor BOOSTXL-SENSORS, ambos de Texas Instruments, y el módulo de hardware LTE IoT 2 click de Mikroe. El BOOSTXL-SENSORS incluye el acelerómetro de 3 ejes BMI160 de Bosch Sensortec. El módulo LTE IoT 2 click incluye el módem BG96 de Quectel que se utiliza para la comunicación de datos mediante el protocolo de tecnología celular Narrowband IoT, que a su vez incluye un módulo para el posicionamiento global mediantes satélites (GNSS). El protocolo de comunicación entre el Equipo y el Sistema Central es MQTT. El Sistema Central recibe los datos del Equipo y los almacena en una base de datos. Se incluye en el Sistema Central una interfaz de usuario que permite visualizar los datos almacenados en tiempo real y también permite exportar datos en formato CSV.

El Equipo cuenta con dos modos de funcionamiento, el Modo Validación y el Modo Investigación. El primero busca servir de plataforma para probar y validar diferentes algoritmos de caracterización de la actividad ovina. Para ello se adquieren los datos crudos generados por el acelerómetro a una tasa de 100 Hz, y se envían al Sistema Central cada 5 segundos, junto con la ubicación geográfica que se adquiere cada 10 segundos. Este modo tiene limitada la autonomía a 48 horas aproximadamente. El Modo Investigación está pensado para dar soporte a experimentos con animales de varios días, donde el foco ya no es validar los algorit-

mos, sino estudiar el comportamiento de los animales. Para lograr esto, el Equipo transmite únicamente el Estado del animal (corriendo, caminando, quieto, o con la cabeza agachada) y registra la ubicación geográfica con menor frecuencia (2 o más minutos), logrando aumentar la autonomía a una semana. En ambos modos, el microcontrolador es el encargado de identificar los Estados del ovino utilizando los datos provenientes del acelerómetro en tiempo real. Para ello se implementó un algoritmo, propuesto previamente en la literatura, basado en el método del Análisis del Discriminante Lineal (LDA por sus siglas en inglés). Usando datos de bases de datos públicas se entrenó un clasificador en PC, y luego se lo programó en el microcontrolador.

El Sistema implementado cumple con los principales objetivos del proyecto. En particular se destaca que pruebas preliminares realizadas en ovejas en Facultad de Veterinaria muestran que la identificación del Estado tiene un 88 % de efectividad. Por otro lado, las pruebas de consumo de energía del Equipo revelan que el módulo de alimentación debe ser rediseñado en su totalidad si se pretende aumentar la autonomía.

Glosario

BMI160 Pequeña unidad de medición inercial de bajo consumo fabricada por *Bosch*.

Equipo Dispositivo encargado de la recopilación de medidas que determinan el Estado de la oveja. Va sujeto al cuello del animal mediante un elemento de fijación.

Estado Situación en la que se encuentra la oveja portadora del Equipo. Los Estados caracterizados son parada, pastando, caminado y corriendo.

GPIO *General Purpose Input Output* (del inglés), entrada-salida de propósito general de un microcontrolador.

GPS *Global Positioning System* (del inglés) es un sistema que permite, a través de una red de satélites, indicar la posición de un cuerpo en la superficie terrestre con gran precisión.

GUI Interfaz gráfica de usuario. (La sigla viene de la expresión en inglés Graphical User Interface).

I2C *Inter-Integrated Circuits*. Se le denomina a un tipo de bus serial de comunicación sincrónica, multi-maestro, multi-esclavo.

IMU *Inertial Measurement Unit* (del inglés), unidad de medición inercial.

IoT *Internet of Things* (del inglés), internet de las cosas. Es el concepto que refiere a un sistema de objetos interrelacionados conectados a Internet que pueden recopilar y transferir datos a través de una red inalámbrica. Su espectro es muy amplio incluyendo por ejemplo automatismos de maquinarias o un televisor inteligente.

LDA *Linear Discriminant Analysis* (del inglés), Análisis del Discriminante Lineal.

LPWAN *Low Power Area Newtork* (del inglés), red de área amplia de baja potencia.

ML *Machine Learning* (del inglés), aprendizaje automático.

MQTT También conocido como *Mosquitto*. Message Queuing Telemetry Transport (del inglés), es un protocolo de comunicación enfocado a aplicaciones IoT.

Módem Refiere a un dispositivo encargado de modular y demodular señales. En este caso, señales digitales a radiofrecuencia para su transferencia en el aire.

NB-IoT *Narrowband IoT*, conocido como Red del Internet de las Cosas. Es una LPWAN desarrollada para habilitar servicios para dispositivos móviles.

PCB *Printed Circuit Board* (del inglés), placa con circuito impreso.

Sistema Conformado por el par Equipo y Sistema Central.

Sistema Central Software en un servidor que intercambia datos con el Equipo y los muestra al usuario mediante la GUI.

UART *Universal Asynchronous Receiver-Transmitter* (del inglés), protocolo de comunicación serial asíncrona.

Tabla de contenidos

Tabla de contenidos

Tabla de contenidos

Capítulo 1

Introducción

1.1. Descripción del problema

En el 2019 existían en Uruguay 6,56 millones de ovinos [1], siendo la eficiencia reproductiva una de las principales limitantes para el desarrollo del sector. El complejo agroindustrial ovino forma parte del sector agroexportador: las exportaciones de lana y carne corresponden al 2,4 % y 0.9 % del total de exportaciones agropecuarias respectivamente [1]. En nuestro país, la producción ovina se desarrolla mayoritariamente en sistemas semiextensivos y extensivos, con pastoreo en campo natural a cielo abierto, con mínima intervención humana, frecuentemente compartiendo los potreros con bovinos [2].

En Uruguay el porcentaje de señalada (cantidad de corderos señalados/ovejas encarneradas) es bajo, del entorno del 70 % debido a la alta mortalidad de corderos [3], [4], [5]. Alrededor del 95 % de los corderos que mueren, lo hacen en las primeras 72 horas de vida, mayormente como consecuencia del síndrome inanición-exposición al clima [6]. Estas muertes pueden relacionarse con un peso bajo al nacer, establecimiento inadecuado del vínculo oveja-cordero o producción de calostro y leche insuficientes para cubrir las necesidades del cordero. Además, los neonatos se encuentran en una situación de mayor exposición y vulnerabilidad ante un clima hostil y/o el ataque de depredadores.

En síntesis, la eficiencia reproductiva del rubro ovino es baja fundamentalmente por la alta mortalidad de corderos, lo que implica problemas de: 1) uso ineficiente de recursos, ya que muchas ovejas gestan corderos que mueren en el posparto inmediato, habiendo consumido alimento para desarrollar la gestación durante el período del año de mayor carencia; 2) bienestar animal de las ovejas, ya que utilizan parte de sus escasas reservas energéticas en producir un cordero que no prospera, además de que despliegan un intenso comportamiento maternal hacia un cordero que muere rápidamente; 3) bienestar animal de los corderos, ya que un alto porcentaje muere poco después de nacer; 4) uso ineficiente de materiales, instalaciones, recursos humanos, etc., ya que la eficiencia de su utilización aumentaría con la cantidad de corderos producidos; 5) menores resultados productivos para el productor; 6) menores resultados productivos que los potenciales a nivel

de país, con consecuencias sobre toda la cadena productiva.

Para desarrollar estrategias de mayor cuidado alrededor de la fecha del parto, es importante contar con indicadores que permitan prever con antelación el momento del parto. Esto permitiría aplicar diferentes estrategias de manejo: utilizar espacios con protección o parideras, ofrecer dietas específicas, etc. Sin embargo, de acuerdo al conocimiento que tenemos no existe en la bibliografía internacional información sobre cambios comportamentales y/o en la dinámica y cohesión social que puedan predecir el parto con 48-72 horas de antelación, asimismo tampoco existen dispositivos comerciales que generen estos indicadores [7].

1.2. Antecedentes y estado del arte

Actualmente, se están utilizando una amplia gama de sensores en el ámbito de la ganadería. Un trabajo realizado en 2018 [8], estudia el uso de sensores en investigaciones con ovejas. Analizan 71 artículos publicados entre 1983 y 2017 con un total de 82 experimentos distintos realizados a lo largo de todos los continentes sobre 17 tipos diferentes de ovejas. El estudio muestra la amplia variedad de sensores utilizados en ovejas y un gran crecimiento del interés en los últimos años en el área.

A nivel comercial, esta tecnología de sensores en animales está mucho más desarrollada en el ámbito del ganado bovino, especialmente para producción intensiva de carne o lechería. En ganadería ovina es muy incipiente en general tanto para manejos intensivos como extensivos. El desarrollo de dispositivos utilizables en forma similar en manejos extensivos, donde justamente se busca evitar mover a todos los animales en forma innecesaria cada día, es también muy incipiente. Algunos ejemplos relevantes desarrollados en la última década se mencionan en [9].

A continuación se detallan algunos ejemplos de productos comerciales con características comunes a las que se busca desarrollar en este proyecto. Estos ejemplos dan indicios de los requerimientos que podría tener el sistema que se presenta en este proyecto. Por otra parte, cabe destacar que hasta donde sabemos la aplicación en nuestra ganadería de estos productos es impracticable debido a los altos costos.

Moocall [10] es un dispositivo estilo collar que monitorea los movimientos de una vaca y es capaz de avisar mediante SMS, aplicación móvil o vía mail una hora antes del parto estimado. Este dispositivo cuenta con batería recargable y una autonomía de aproximadamente 4 o más semanas. Este producto en particular sienta precedente para la autonomía de un producto que se pueda comercializar y con una finalidad muy similar a la de este proyecto. Se aclara que no podría ser sustituto al sistema que se plantea desarrollar ya que Moocall es sólo aplciable en vacas.

Digitanimal [11] es una empresa que aporta soluciones al sector ganadero usando Internet de las cosas (IoT). Desarrollaron una solución para el monitoreo de animales utilizando sensores. La solución que plantean consiste en un dispositivo que cuenta con diferentes sensores que permiten obtener datos sobre temperatura, ubicación, actividad del animal y distancia recorrida. Este dispositivo también se utiliza en forma de collar y cuenta con una aplicación capaz de enviar notifi-

caciones y consultar datos obtenidos. La batería tiene una vida útil de un año. Este producto hace ver la importancia de tener una interfaz para el usuario donde visualizar información con respecto a los animales. Esto difieren por ejemplo del acercamiento de Moocall el cual utiliza SMS. Por consiguiente, el presente proyecto tomará en consideración este aspecto de interfaz de usuario al desarrollar la parte del sistema que se encarga de recibir los datos.

Afimilk [12] es una empresa que trabaja en el desarrollo, fabricación y comercialización de sistemas informáticos para la producción lechera. Cuentan con un collar que entre muchas cosas puede identificar el comportamiento de una vaca siendo capaz de discriminar cuando el animal esta rumiando, pastando o moviéndose. En otras palabra, este producto es de interés ya que sienta precedente de que un collar puede caracterizar correctamente al menos tres comportamientos de un animal. De manera similar que en Moocall, Afimilk no podría sustituir el sistema desarrollado en este proyecto ya que es solo aplicable en vacas.

En los últimos años el Grupo de Microelectrónica (GME) del IIE ha llevado adelante diversos proyectos con aplicación agropecuaria usando redes de sensores inalámbricos, procesamiento de señales, e Internet de las cosas. Se destaca el reciente desarrollo de un proyecto de cercas virtuales [13], actualmente incubado en Incubaelectro. El presente proyecto se enmarca dentro de una línea de investigación conjunta entre el Prof. Rodolfo Ungerfeld (Facultad de Veterinaria) y el GME, que incluye la ejecución del CSIC I+D “Sistema electrónico para la caracterización del comportamiento de ovinos”. En ese marco, Rodolfo Ungerfeld actuó como asesor y cliente del presente proyecto.

1.3. Descripción breve de la solución propuesta

A modo de aproximarse a una solución para el problema planteado, este proyecto apunta a la creación de un dispositivo vestible por los animales que facilite el estudio e investigación de su comportamiento. En particular, se busca desarrollar un collar que contenga un dispositivo electrónico basado en un microcontrolador y un conjunto de sensores capaces de caracterizar parte de su comportamiento mediante el procesamiento de datos obtenidos de un acelerómetros utilizando algoritmos de ML (Machine Learning). Se le llamará Equipo al dispositivo que irá colocado en el cuello de la oveja y para caracterizar parte de su comportamiento se definen los Estados. El Estado determina la situación en la que se encuentra la oveja. Los Estados caracterizados en este proyecto son quieta, cabeza agachada, caminando y corriendo. Esta información se procesará y transmitirá mediante tecnologías de comunicación inalámbrica de telefonía celular (NB-IoT) a un Sistema Central para su visualización y análisis. De este modo, se espera generar una vía de estudio de los patrones de comportamiento de la oveja que lleve a determinar, en una instancia posterior a este proyecto, si una oveja está próxima a parir. Pudiendo determinar si una oveja está por parir dentro de un margen de tiempo cercano al parto, se pueden tomar medidas que mejoren la expectativa de vida de la cría.

1.4. Requerimientos conceptuales

El Equipo será empleado para adquirir datos que caractericen el Estado de la oveja. El Equipo deberá determinar la ubicación geográfica de la oveja y adquirir datos de un acelerómetro colocado en su cuello. Además, mediante el procesamiento de estos datos, determinará el Estado. Luego, estos datos serán enviados al Sistema Central. Para esto, se debe implementar una comunicación por una red que asegure cobertura en las áreas rurales donde se emplee el producto. El esquema de la figura [1.1] ilustra en general el funcionamiento del Sistema.

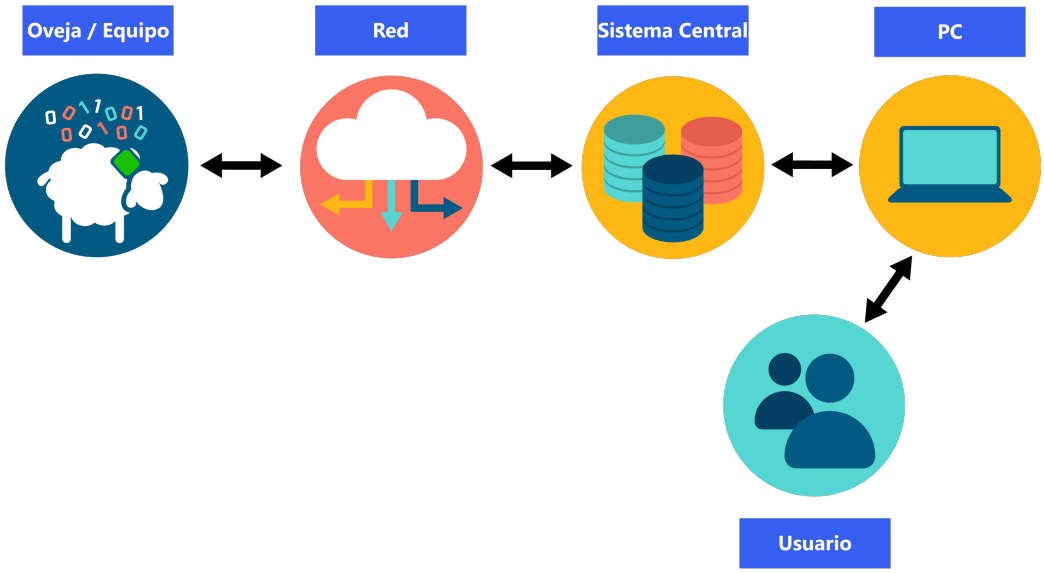

Figura 1.1: Diagrama del Sistema completo

La información recolectada debe poder ser desplegada en un formato comprensible por personal técnico veterinario, por lo que el Sistema Central contará con una interfaz de usuario (GUI).

El Equipo deberá contar con dos modos de funcionamiento, el *Modo Validación* y el *Modo Investigación.* El primero busca servir de plataforma para probar y validar diferentes algoritmos de caracterización de la actividad ovina. Este modo puede tener limitada la autonomía a pocas horas. Por otro lado, el Modo Investigación está pensado para dar soporte a experimentos con animales durante varios días, donde el foco ya no es validar los algoritmos, sino estudiar el comportamiento de los animales. Para lograr esto, el Equipo podrá transmitir únicamente el Estado del animal (corriendo, caminando, quieto, o con la cabeza agachada) y se podrá registrar la ubicación geográfica con menor frecuencia, logrando aumentar la autonomía. Queda fuera del alcance de este proyecto un tercer modo de funcionamiento, *Modo de Producción*, con objetivos comerciales, donde la autonomía deberá ser de varios meses.

En lo que respecta al elemento de fijación, debe ser tal que sea solidario al movimiento del animal de manera que los datos relevados de su cuello sean repre-

sentativos. Más aún, el Equipo no debe ser un estorbo para el animal, a modo de evitar alterar su comportamiento o que el animal lo dañe tratando de quitárselo.

1.5. Requerimientos específicos

En diálogo con los clientes y tutor se definieron los siguientes requerimientos:

- El Equipo deberá ser capaz de medir la ubicación geográfica del animal con una precisión de 5 m.
- El Sistema Central deberá desplegar en un ordenador la información en forma gráfica.
- Se deberá contar con una Interfaz de Usuario capaz de desplegar los datos adquiridos.
- Se deberá muestrear la ubicación geográfica al menos una vez cada 30 segundos.
- Se deberán muestrear datos del acelerómetro al menos una vez por segundo.
- A partir de los datos obtenidos se deberá identificar al menos un Estado de la oveja.
- La caracterización de Estados en campo debe presentar una probabilidad de error en dicha caracterización menor al 5 % con respecto a la probabilidad de aciertos teórica que tenga el algoritmo elegido. Se entiende por error el caso en que se caracteriza un Estado que no es coherente con la realidad.
- El Equipo deberá contar con un sistema de baterías que no presente riesgos que puedan causar daño físico para el animal, asegurando una autonomía de al menos una semana sin necesidad de intervención humana.
- Las pruebas realizadas en animales deben contar con la supervisión de un veterinario.
- El Equipo debe tener un peso inferior a 600 g.
- El encapsulado del Equipo debe tener dimensiones inferiores a 14 cm $\times$ 10 cm $\times$ 8 cm (largo x ancho x altura).
- La fabricación del prototipo final del Equipo debe costar menos de 10.000 pesos uruguayos.
- Se deberá desarrollar un Modo de Validación con el fin de utilizarse para validar el Sistema y un Modo de Investigación con el fin de utilizarse para hacer estudios en campo. El Modo de Producción con objetivos comerciales queda fuera del alcance del proyecto.

- El modo de funcionamiento de validación deberá trabajar con frecuencias de muestreo altas. Del orden de las decenas de milisegundos para los datos de la IMU y de los segundos para los datos del GPS con una autonomía aceptable de algunas horas.
- El modo de funcionamiento de investigación deberá contar con frecuencia de muestreo de la IMU similar al de validación y con una frecuencia de muestreo de algunos minutos para el GPS con una autonomía aceptable de 5 días.

Resumen requerimientos				
	Aceptable		**Ideal**	
Precisión GPS	5 m		1 m	
Caracterización de Estados	95 % casos		99 % casos	
Peso	600 g		300 g	
Tamaño máximo	14x10x8 cm (largo x ancho x alto)			
Costo del prototipo	<10000 pesos uruguayos		<7500 pesos uruguayos	
Modo	**Validación**		**Investigación**	
	Aceptable	**Ideal**	**Aceptable**	**Ideal**
Tiempo entre muestras IMU	1 s	1 ms	1 s	1 ms
Tiempo entre muestras GPS	30 s	10 s	5 minutos	2 minutos
Autonomía	3 horas	12 horas	5 días	3 semanas

Tabla 1.1: Tabla de requerimientos con condiciones aceptables e ideales a cumplir

1.6. Criterio de exito

- Tener el prototipo de un Equipo que sea capaz de adquirir la información de ubicación geográfica y aceleraciones de un animal dentro de la majada, y pueda transferirla hacia un Sistema Central, donde pueda ser visualizada y analizada.
- Capacidad del Equipo de identificar al menos un Estado del animal.
- Tener un prototipo robusto en términos de durabilidad en la intemperie para hace pruebas en el animal

1.7. Organización del documento

Se procede a describir los contenidos de los distintos capítulos que componen el documento a continuación.

- **Capítulo 1: Introducción.** Comprenfde la explicación del surgimiento del proyecto, la temática de estudio, antecedentes, el estado del arte y describe

los contenidos de los capítulos del documento. Además, se detalla el funcionamiento general del Sistema y en particular se listan tanto los requerimientos como el criterio de éxito del proyecto.

- **Capítulo 2: Diseño del hardware y la electrónica.** Se presentan los componentes utilizados para conformar los módulos hardware junto a sus respectivas funciones en el Sistema. Asimismo, se elabora sobre la interconexión entre los componentes, explicando las decisiones del diseño y mostrando la electrónica del dispositivo final.

- **Capítulo 3: Transmisión y recepción de datos.** En este capítulo se describen las tecnologías involucradas para la transmisión de datos: NB-IoT (Narrowband IoT) y en capa de aplicación MQTT (Message Queuing Telemetry Transport). A su vez, se describe el funcionamiento y desarrollo del Sistema Central.

- **Capítulo 4: Clasificación de Estados de la oveja.** Se describe el proceso de desarrollo y funcionamiento del módulo clasificador de Estados implementado.

- **Capítulo 5: Diseño del software embebido.** Se describe el software embebido en el microcontrolador. En particular, se presenta la arquitectura elegida, se listan los módulos con sus descripciones y se detalla el bucle principal.

- **Capítulo 6: Diseño mecánico**. Se documentan los aspectos tenidos en cuenta para el diseño del encapsulado y su sujeción, la topología de los componentes electrónicos dentro del encapsulado y el proceso para su confección.

- **Capítulo 7: Pruebas y análisis de resultados.** Se describe las pruebas para verificar los requerimientos y la validación del funcionamiento del Sistema Completo.

- **Capítulo 8: Conclusiones.** Se realiza un análisis general del desarrollo del proyecto, evaluando los resultados obtenidos con respecto a los objetivos definidos en los requerimientos. Por último, se comenta sobre los aspectos a trabajar a futuro para una siguiente iteración sobre este proyecto.

- **Apéndices.** Se anexan distintos apartados con cálculos y diseños.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 2

Diseño del hardware y la electrónica

En este capítulo se presentan los distintos módulos que conforman el Sistema. Se describe el proceso de elección de los componentes y una breve explicación sobre su funcionalidad.

2.1. Descripción del Equipo

La Figura [2.1] muestra el diagrama de bloques que componen el Equipo. El Equipo se construye en base a un microcontrolador encargado de ejecutar el software embebido diseñado y comunicarse con los demás componentes del Equipo para controlar su funcionamiento. Se divide el Equipo en 2 módulos. Por un lado el módulo encargado del muestreo, procesamiento y comunicación y por otra parte, aquel encargado de la alimentación.

El Equipo incluye el microcontrolador MSP432P401R en su placa de desarrollo producida por Texas Instruments. Dicho controlador se comunica mediante protocolo I2C con la placa BOOSTXL-SENSORS Pack de la misma empresa que cuenta con una IMU (BMI160) desarrollada por Bosch Sensortec (así como otros sensores que no serán de interés). Esta placa está diseñada para acoplarse al microcontrolador.

Por otro lado, el microcontrolador se comunica mediante protocolo UART con la placa LTE IoT 2 click, desarrollada por Mikroe, la cual cuenta con el módulo BG96 encargado de la comunicación y GPS. El BG96 es configurado para establecer la comunicación mediante la red NB-IoT y envía la información al Sistema Central empleando el protocolo MQTT (la descripción de dicha red y protocolo se dan en el capítulo 3).

Tanto la placa de desarrollo del MSP432P401R como la placa LTE IoT 2 click se conectan entre sí por medio de un PCB diseñado utilizando el programa Eagle de AutoDesk. A su vez, se cuenta con otro PCB encargado del manejo de la alimentación mediante baterías 18650 Li-ion de 3.7V y un panel solar.

Los esquemáticos y layouts de los PCB desarrollados pueden consultarse en el apéndice A.

En la sección 2.2 se detallan los componentes del módulo de muestreo, procesa-

miento y comunicación y en la sección 2.3 se detallan los componentes del módulo de alimentación.

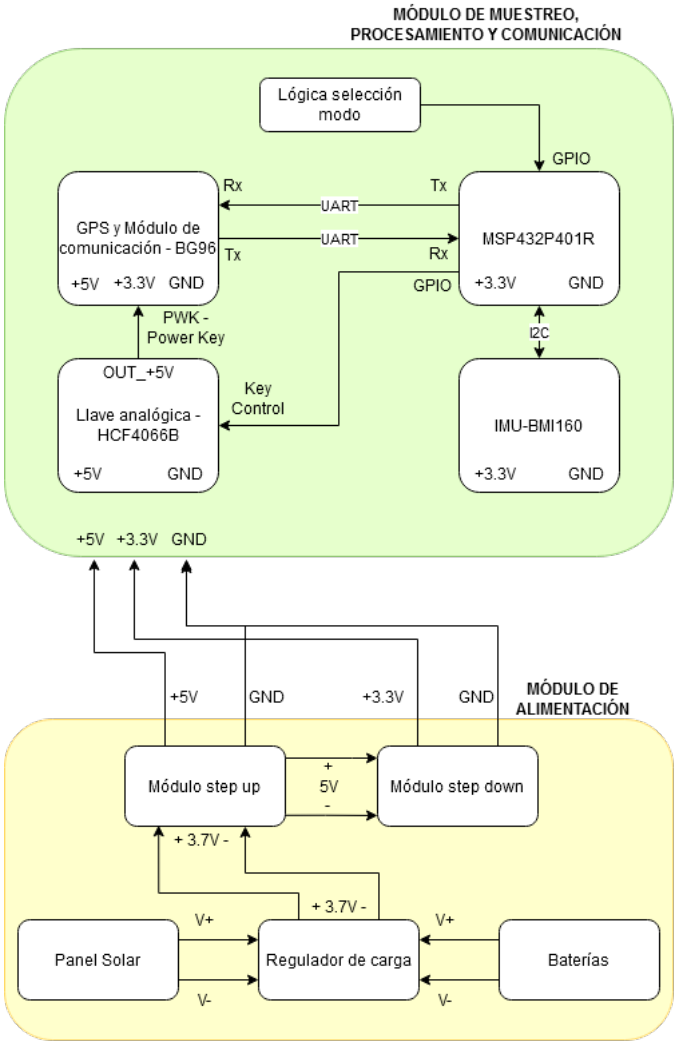

Figura 2.1: Diagrama de la arquitectura de hardware de un Equipo

2.2. Módulo de muestreo, procesamiento y comunicación

Este módulo se encarga de la obtención de datos crudos para su procesamiento y del envío de datos al Sistema Central. Los submódulos que lo constituyen y son descritos a continuación se encuentran interconectados por un PCB cuyo diseño y layout pueden encontrarse en el Apéndice [A].

En la figura [2.2] puede verse el módulo completo.

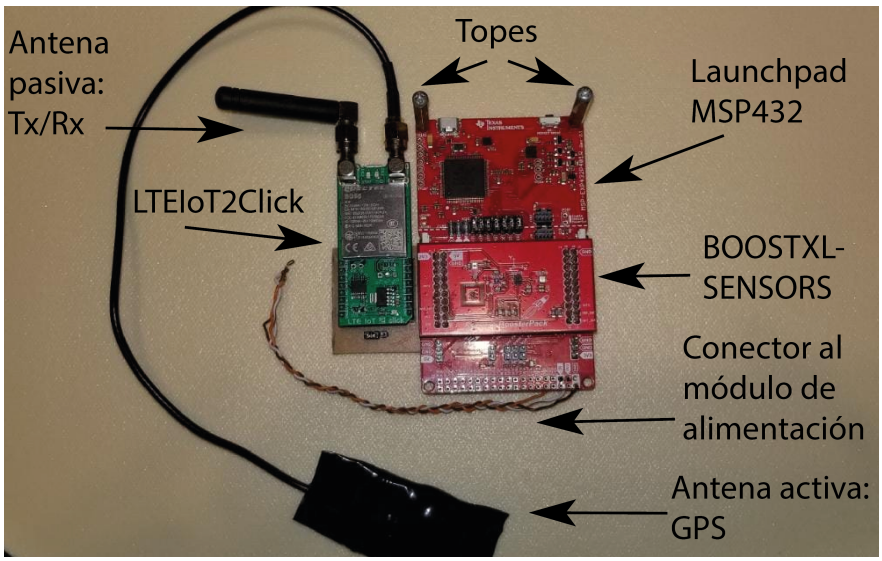

Figura 2.2: Foto de la implementación en hardware del módulo de muestreo, procesamiento y comunicación

2.2.1. Microcontrolador

Para la elección del microcontrolador, se estudiaron tres posibles alternativas. Dos de ellas dentro de la familia del MSP de Texas Instruments y por otro lado uno de la familia de microcontroladores STM32.

MSP

El MSP430 y el MSP432 son familias de microcontroladores desarrolladas por Texas Instruments. Como característica principal, se tiene que son diseñadas para trabajar en aplicaciones de sistemas embebidos de bajo consumo.

Por una parte, se considera emplear el MSP430 al ya haber adquirido experiencia usando su placa de desarrollo en trabajos anteriores (fue la plataforma en la que se desarrolló el proyecto del curso "Sistemas Embebidos en Tiempo Real" del IIE de mismos autores que este trabajo [14]). La alternativa del MSP432 surge de buscar otras opciones dentro de Texas Instruments. Con el objetivo de buscar otras placas con mayor cantidad de recursos pero que a su vez se permita adaptar fácilmente los conocimientos ya obtenidos en el curso.

En particular, se estudió el microcontrolador MSP430F5539 dado que dentro de la familia de los MSP430 es de los pocos que cuenta con un multiplicador de hardware. Dada la necesidad de realizar operaciones como la suma y la multiplicación dentro del microprocesador, se identificó la necesidad de trabajar con microprocesadores que cuenten con una unidad de punto flotante. La unidad de punto flotante es un componente del microprocesador que permite hacer operaciones básicas como la suma y la multiplicación con números flotantes. En caso de no contar con dicha unidad se identificó como indispensable contar con un multiplicador de hardware para poder realizar operaciones de multiplicación. Por otro lado,

de la familia del MSP432 se estudió el MSP432P401R. Este microcontrolador es utilizado por Mikroe para realizar sus aplicaciones por lo tanto la existencia de bibliotecas, foros y ejemplos puede facilitar su uso y entendimiento.

STM32

Como tercer alternativa se analizó usar el Kit IoT desarrollado y proporcionado por Antel basado en un microcontrolador SMT32 ARM Cortex M4 [15]. Se estudió el microcontrolador STM32L433RC dado que es el que se encuentra integrado al Kit IoT. El mayor incentivo para considerar esta opción fue el eventual apoyo que se hubiera podido tener por el equipo de Antel. Además, existen varias aplicaciones ya desarrolladas a modo de ejemplo con la interfaz de Arduino.

Elección del microcontrolador

En la tabla 2.1 se comparan los distintos microcontroladores y se destacan características que se consideran relevantes para la elección del microcontrolador.

Finalmente se optó por trabajar con el MSP432P401R. La razón de mayor peso fue la compatibilidad con los otros componentes de hardware utilizados.

Se tiene como interfaces principales:

- I2C para la obtención de los datos muestreados por la IMU.
- UART para el envío de comandos al módulo GPS y de comunicación.
- GPIO para la lectura del modo de funcionamiento (Ver capítulo 5.4)
- GPIO para el manejo de la llave analógica que maneja el encendido y apagado del módulo GPS y de comunicación.

	MSP Texas Instruments		**STM32 ST**
Micro	MSP430F5529	MSP432P401R	STM32L433xx
Procesador	16-bit RISC	ARM 32 bits Cortex M4	ARM 32 bits Cortex M4
Flash	128 kB	Hasta 256 kB de Flash Main Memory 16 kB de Flash Information	Hasta 256 kB de Single Bank Flash
Ram	8 kB	Hasta 64 kB de SRAM	64 kB de SRAM
Frecuencia	25 MHz	48 MHz	80 MHz
Interfaces **Especificaciones mínimas** **1xI2C (hacia sensores)** **2XUART (hacia BG96 y GPS)**	2xUSCI	4xEUSCI_A (UART, IrDA, SPI) 4xEUSCI_B (I2C, SPI)	3xI2C 3xSPI
Multiplicador de Hardware (HW) o FPU	HW	FPU	FPU
Consumo	Active: 290 µA/MHz @ 8 MHz LPM4: 1 µA	Active: 80 µA/MHz LPM4.5: 25 nA	Run mode: 84 µA/MHz Shutdown mode: 8 nA

Tabla 2.1: Tabla comparativa de los 3 microcontroladores estudiados

2.2.2. BOOSTXL-SENSORS

El BOOSTXL-SENSORS es un módulo plug-in del LaunchPad MSP432P401R, desarrollado por Texas Instruments. Cuenta con los siguientes sensores:

- Sensor de Luz (OPT3001)
- Sensor de Temperatura (TI TMP007)
- Sensor Ambiental: presión, temperatura y humedad (BME280)
- IMU: Unidad de Medida Inercial (BMI160)
- Magnetómetro (BMM150)

Para este proyecto se utiliza en particular la IMU, compatible con aplicaciones de bajo consumo y con interfaces I2C y SPI. La comunicación entre el microcontrolador y la IMU se realiza mediante el protocolo I2C.

BMI160: Inertial Measurement Unit:

- 16 bit IMU
- 3 ejes acelerómetro y 3 ejes giroscopio
- Resolución de acelerómetro de interés $\pm 2g : 16384 LSB/g$
- Resolución de giroscopio $\pm 2000^{o}/s : 16{,}4 LSB/^{o}/s$
- Compatible con modos de bajo consumo. Esto significa que tiene un consumo bajo y posee modos de funcionamiento que permiten reducir el consumo.
- Dimensiones 2.5 x 3.0 x 0.8 mm

2.2.3. LTE IoT 2 click

El LTE IoT 2 click es una placa que permite conexiones de red de LTE, utilizando el módulo BG96 LTE. El BG96 LTE es un módulo desarrollado por Quectel Wireless Solutions. Además, cuenta con un GPS empleado para la obtención de la ubicación geográfica.

Como características principales se tiene:

- Módulos en la placa:
  - Módulo BG96 de Quectel (módulo de LTE CAT M1/NB1 y GNSS)
  - MCP1826 (regulador de salida de baja caída (LDO) de 1A)
  - TXB0106 (traductor de nivel de voltaje bidireccional)
- Atributos principales:
  - Stack TCP/UDP/PPP
  - Soporte de tecnologías CAT NB y CAT M
  - Dos conectores SMA para antenas. Un conector SMA de antena principal (utiliza para conectar el módulo a la estación base LTE) y un conector SMA de antena secundaria (GPS).

    - Acceso a los servicios GNSS (Sistema global de navegación por satélite)

- Interfaces USB, UART
- Voltaje de entrada: 3.3 V, 5 V
- Tamaño: 57.15 mm × 25.4 mm

Mediante una de sus UART se realizan los intercambios de comandos y respuestas con el microcontrolador para el manejo de la comunicación con el Sistema Central. Para encender y apagar el módulo es necesario generar en la entrada PWK un pulso de 5 V al menos durante 650 ms.

2.2.4. Antenas

El LTE IoT 2 click utiliza 2 antenas que se conectan mediante conectores SMA. Por defecto el módulo viene con 2 antenas pasivas. Una de las antenas es utilizada para conectar el módulo a la red LTE, mientras que la otra antena es utilizada para el posicionamiento global mediante satelites (GPS). Esta segunda antena puede ser tanto pasiva como activa.

La principal diferencia que tienen las antenas pasivas y activas, es que las activas tienen componentes activos (que requieren alimentación) que permiten darle ganancia a la señal.

Por otra parte, la antena pasiva no necesita alimentación pero la señal recibida puede ser muy débil requiriendo más tiempo para ubicar los satélites y posicionarse.

Cabe mencionar que la alimentación para estas antenas se suele dar por el mismo cable coaxial por el que viaja la señal.

Resulta que el consumo del GPS es bastante elevado en comparación al resto de los componentes, lo cual se explica con mayor detalle en la sección 2.3, y la obtención de ubicación se considera crítica para la aplicación. Por ello, se busca que el tiempo en el que este esté prendido sea el menor posible para fijar ubicación. Se probó utilizando la antena pasiva que viene por defecto con el LTE IoT 2 click y su tiempo de respuesta en un ambiente exterior abierto estaba cerca de los 5 minutos mientras que en un ambiente cerrado directamente no lograba posicionarse. Por lo tanto, se decidió cambiar la antena por una antena activa y los tiempos de respuesta tanto en un ambiente cerrado como en un ambiente abierto disminuyeron a valores menores al minuto (del orden de las pocas decenas de segundos).

La antena activa elegida fue la ANN-MS de U-Blox.

La desventaja clara es el aumento en el consumo de todo el Sistema (según la hoja de datos unos 8 mA cuando el módulo de GPS está encendido). Sin embargo, en comparación al gasto de energía que implica que el GPS esté prendido más tiempo para posicionarse, resulta ser un compromiso aceptable. Además, se considera crítica la obtención de posición como dato siendo indeseable que se pierda la señal.

2.2.5. HCF4066B

El circuito integrado en cuestión es un interruptor con el propósito de transmitir o multiplexar señales analógicas o digitales. La llave se utiliza para generar el pulso (ver Sección 2.2.3) que permite prender y apagar el módulo LTE IoT 2 click. Su entrada se conecta a +5V y su salida al pin PWK del módulo LTE IoT 2 click. La entrada de control se conecta a un pin de GPIO del microcontrolador (ver Fig. [2.1]).

Como características principales, tiene una resistencia de encendido relativamente constante de aproximadamente 470 Ω y un voltaje de control de entrada de 3.5 V cuando se alimenta con 5 V (aunque se ensayó que 3.3 V es suficiente también).

La selección de este componente fue basada en principio a la simplicidad en su uso. Además, fue utilizado desde las etapas preliminares del diseño del hardware sin ningún inconveniente por lo que no hubo planteamientos ulteriores de cambiarlo. Eventualmente se podría evaluar nuevamente esta elección en pro del uso de un transistor como llave u otro circuito.

2.2.6. Selección modo

Con la finalidad de poner al Equipo en el modo de funcionamiento de Validación o Investigación (definición básica en sección 1.5 y para más detalle ver sección 5.4) se disponen tres pin headers como indica el esquema de la figura [2.3]. De esta manera, empleando un jumper, en el pin del medio se tendrá el valor 3.3 V o 0 V que, conectado a un GPIO del microcontrolador, funciona como bandera para determinar el modo.

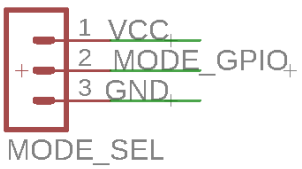

Figura 2.3: Diagrama del selector de modo

2.3. Módulo de alimentación

El módulo en cuestión se encarga de proveer la alimentación al dispositivo. Este fue diseñado de forma tal de suministrar dos niveles de tensión: 3.3 V y 5 V a partir de baterías 18650 y un panel solar. Para lograrlo, se emplearon un módulo step-up, otro step-down y un módulo de carga. Debido a que este módulo se implementó en las etapa finales del proyecto, se optó por recurrir a módulos que se consiguen en plaza, y que tienen ya soldados los chips con sus respectivos componentes externos en PCBs.

Los cálculos para el dimensionamiento en función de los requerimientos quedan explicitados en el Apéndice C.

En la Figura [2.5] puede apreciarse una foto del circuito de alimentación completo.

Se procede con la descripción de los módulos que lo constituyen y algunos aspectos del diseño.

2.3.1. Módulo de carga TP4056

Se utiliza el módulo TP4056 para cargar la batería. Este módulo tiene la ventaja de que ya trae incorporado un sistema de protección que detiene la carga evitando sobrecargas y también puede cortar el circuito evitando sobre descargas. El módulo cuenta con una entrada para conectar el panel solar y dos salidas, una para cargar las baterías y otra para alimentar el resto del Sistema.

Como características principales se tiene:

- Voltaje de entrada: 5 V
- Corriente máxima de carga: 1200 mA
- Tensión de corte de carga: 4.2 V
- Tensión de protección de descarga de la batería: 2.5 V
- Corriente de protección de sobrecarga de la batería: 3 A
- Interfaz de entrada: Micro USB
- Dimensión: 2.6 cm × 1.7 cm

2.3.2. Módulo step up

Para obtener un voltaje de entrada de 5 V, se decidió trabajar con un módulo con un convertidor de voltaje DC-DC MT3608 capaz de aumentar el voltaje provisto por la batería a 5 V, necesarios para el funcionamiento del módulo de LTE IoT 2 click. Dentro de las características más importantes se tiene:

- Voltaje de entrada amplio: 2 V-24 V
- Voltaje de salida: 5 V-28 V (ajustable)

- Corriente de salida máxima: corriente nominal 2 A
- Eficiencia: Hasta 93 %
- Dimensiones: 30 mm × 17 mm × 14 mm

2.3.3. Módulo step-down

Este módulo se encarga de proveer 3.3 V. Para generar este voltaje, se utiliza un regulador lineal Step-Down DC-DC que toma como entrada la salida del módulo step-up. Esto se debe a que según indica su hoja de datos, para dar una tensión de 3.3 V requiere una tensión de entrada mayor a 4.75 V. Este módulo cuenta con un regulador LM2596 con las siguientes características principales:

- Voltaje de entrada amplio: 3.2 V- 40 V
- Voltaje de salida: 1.25 V -25 V (ajustable)
- Corriente de salida máxima: corriente nominal 3 A
- Dimensiones: 47 mm × 26 mm × 15 mm

2.3.4. Panel solar

Se decidió incorporar al proyecto paneles solares con el objetivo de cumplir los requerimientos mínimos con respecto a la autonomía del dispositivo. Para la selección del panel solar a utilizar se realizó previamente un estudio estimado del consumo energético para el dimensionado de los componentes según se indica en el Apéndice [C]. A partir de este análisis y de aquellos paneles disponibles en plaza se decidió trabajar con un panel solar rígido epoxi estándar de silicio policristalino de 112.0 mm × 84.0 mm × 3.2 mm de 1 W de potencia, con un voltaje nominal máximo de 6 V.

2.3.5. Batería

Para la elección de la batería, es necesario saber la capacidad que cumpla con la autonomía deseada.

Se utilizarán dos baterías 18650 recargable de Litio-Ion de 3.7 V en paralelo con una capacidad de 4000mAh cada una (Ver Apéndice C).

La figura [2.4] es un diagrama que representa el módulo de alimentación y sus conexiones.

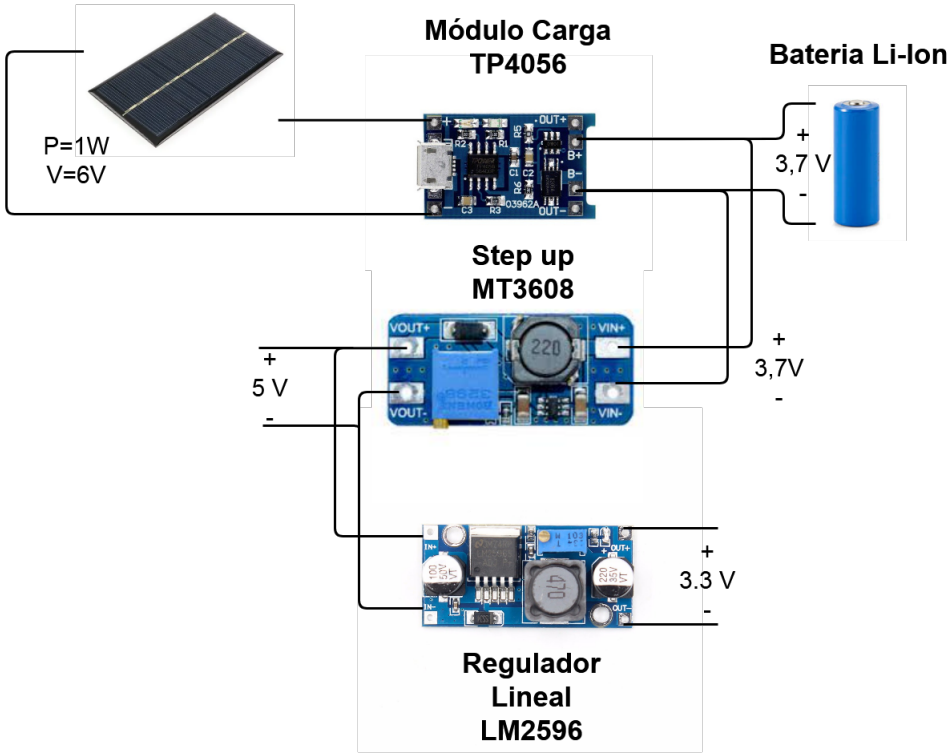

Figura 2.4: Diagrama conexiones del sistema de alimentación

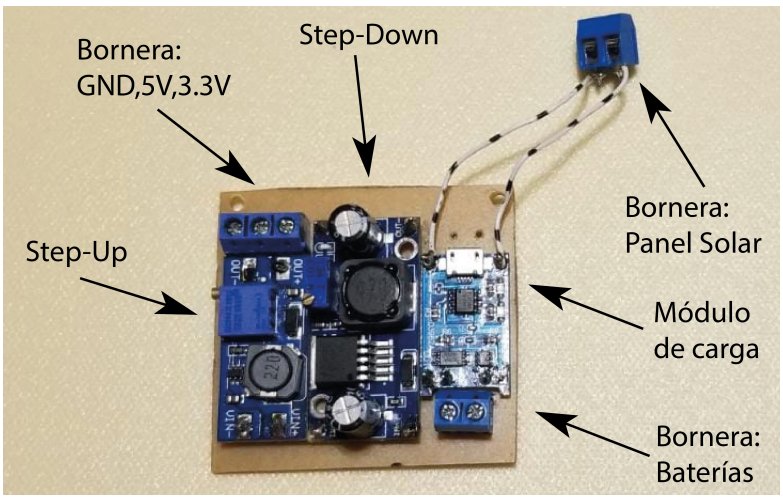

Figura 2.5: PCB-Módulo de alimentación

2.4. Ensamblado de los componentes

En la figura [2.6] puede apreciarse la disposición y conexión entre los módulos. Los topes sirven para unir y dar soporte a las placas de manera de que ocupen el

menor volumen posible, en vista de cumplir con el requerimiento de tamaño de la caja.

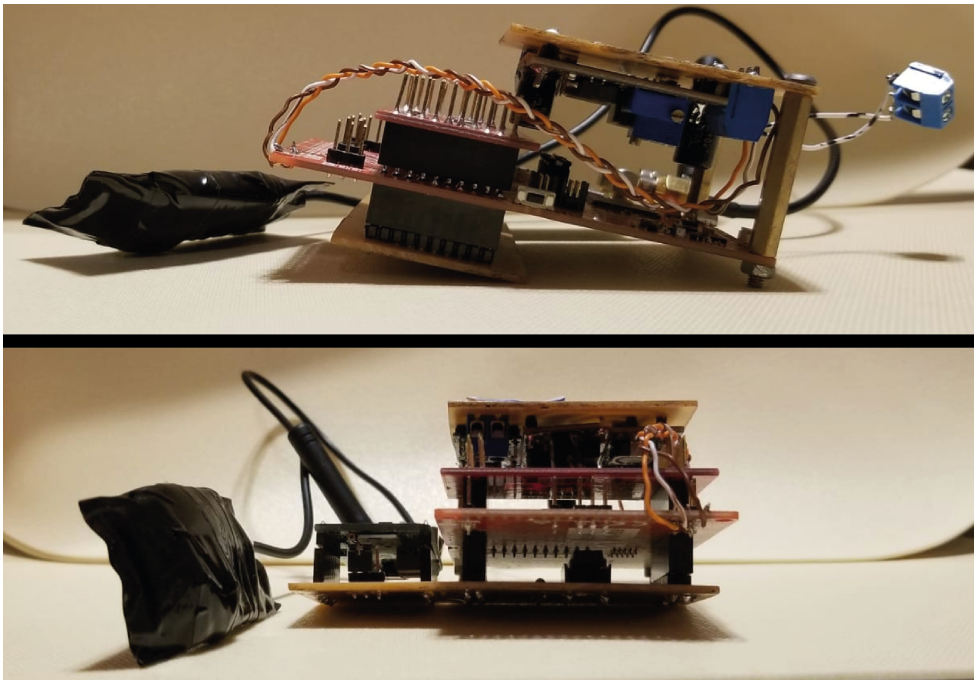

Figura 2.6: Módulos interconectados.
Arriba vista lateral, abajo vista frontal.

Toda la electrónica incluyendo baterías resulta entrar en un volumen de 115 mm x 90 mm x 58 mm.

2.5. Módulo de depurado

Ante la necesidad de depurar el código con mayor simplicidad, se desarrolló una placa con los mismos componentes que el módulo de muestreo, procesamiento y comunicación con la salvedad de que se agrega la posibilidad de observar el tráfico entre el microcontrolador y el módulo de comunicación. El diagrama de módulos expandido se muestra a continuación (ver figura [2.7]). Se procede a describir los agregados para esta placa

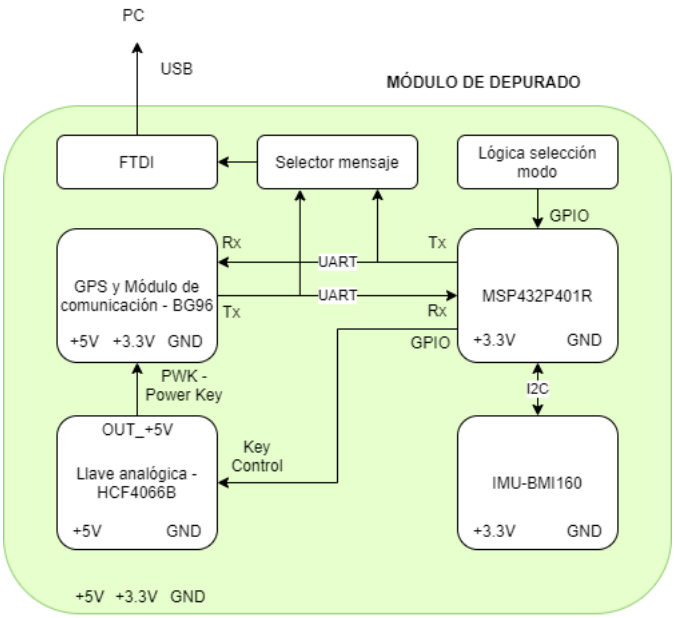

Figura 2.7: Diagrama de la arquitectura de Hardware del Equipo del módulo depurado

2.5.1. Selector mensaje

Bajo la necesidad de querer observar tanto los mensajes trasmitidos por el módulo LTE IoT 2 click como los mensajes recibidos, se emplea una lógica con pines que determina que mensajes se envían hacia la PC para su visualización. Se utilizan 3 pin headers como indica el esquema [2.8] y con un jumper entre el pin del medio y cualquiera de los otros 2 se eligen los mensajes que se desean observar.

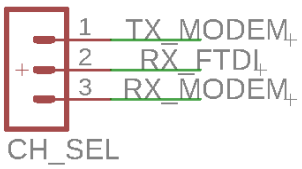

Figura 2.8: Diagrama del selector de mensaje

2.5.2. Conversor USB-TTL FTDI

El módulo de depurado cuenta con una bornera para la conexión de un módulo conversor USB-TTL basado en el controlador FT232RL de FTDI. El módulo conversor permite mandar los datos a una PC para luego poder visualizar el intercambio de mensajes entre las UART del módem y microcontrolador.

Capítulo 3

Transmisión y recepción de datos

A lo largo de este capítulo se describen conceptos básicos sobre la comunicación inalámbrica Narrowband IoT utilizada en el proyecto, así como también algunas características importantes sobre el protocolo de comunicación MQTT. Dicho protocolo fue el elegido para establecer la comunicación entre el Equipo y el Sistema Central. Además, se explica el rol que cumple el Sistema Central y su manejo de los datos recibidos. A lo largo de este capítulo las figuras que se presentan son modificaciones de figuras que se encuentran en [16] [17] [18].

3.1. Comunicación inalámbrica: Narrowband IoT

En el proyecto se implementó una comunicación inalámbrica basada en la tecnología celular conocida como Narrowband Internet of Things (abreviado cómo NB-IoT). Es una red que surge en el auge de las tecnologías LPWAN (Low-Power WideArea Network) por lo que se considera una red de largo alcance y bajo consumo. Fue creada con el objetivo de poder utilizarse con un número potencialmente alto de dispositivos con poca cantidad de datos (menos de 16 kB) a transmitir de forma esporádica, una propiedad que se puede identificar en las características del proyecto.

Una de las grandes ventajas que presenta esta tecnología es que se aprovecha de las tecnologías celulares ya existentes como lo son GSM y LTE haciendo que su despliegue sea más simple. Realiza la transmisión de datos sobre bandas de frecuencia de estas tecnologías, en distintas modalidades. La podemos encontrar en modo *stand alone* en la band de GSM ocupando 200 kHz como se muestra en la figura [3.1] o en la modalidad *in guard* o *guard band* en la banda de LTE ocupando 180 kHz, como se aprecia en la misma figura. El espacio del espectro ocupado por NB-IoT se lo conoce como PRB (Physical Resource Block). [19]

NB-IoT es una tecnología half-duplex, es decir que no puede transmitir y recibir datos simultáneamente. Se logra de forma eficiente hasta una velocidad de datos de 250 kbps para comunicaciones de enlace descendentes y de 200 kbps para comunicaciones de enlace ascendente. [20]

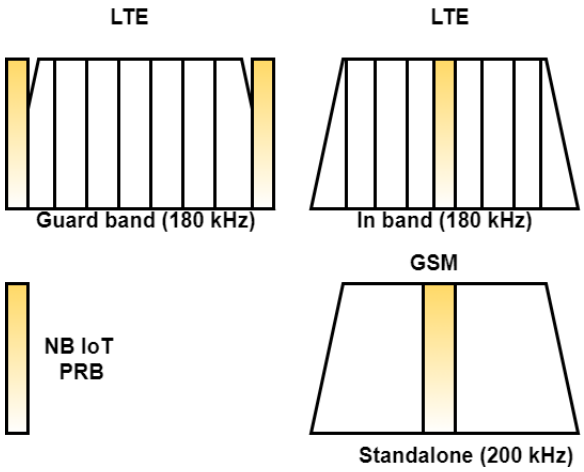

Figura 3.1: Distintas modalidades de uso de banda de Narrowband IoT

Se decide trabajar con esta tecnología debido a su bajo consumo de energía y sus mejoras en cuanto a cobertura. Es ideal para los dispositivos como el que se desarrolla en este proyecto donde se busca un largo ciclo de vida y coberturas en áreas rurales. También permite tener un gran número de dispositivos conectados, lo cual permite escalabilidad para tener un gran número de animales con el Equipo.

Por otro lado, Narrowband IoT es una tecnología nueva y prometedora. De momento, Antel en Uruguay tiene la red establecida para esta tecnología pero no se encuentra en uso comercial. Por esto se valora el estudio de esta tecnología por sobre otras anteriores.

3.1.1. Modos de bajo consumo de Narrowband IoT

La siguiente sección describe dos configuraciones de Narrowband IoT que tienen como objetivo disminuir el consumo de energía y prolongar la vida útil de los dispositivos que utilicen Narrowband. Ambas configuraciones son compatibles con el módulo BG96 utilizado en el proyecto.

PSM: Power Saving Mode

Si bien siempre es posible apagar los dispositivos conectados a las radio bases para disminuir el consumo de batería, los dispositivos tienen que registrarse nuevamente a la red cada vez que se enciendan. Esta reconexión puede implicar gastos de energía no deseados. A modo de evitar este problema, surge el PSM (Power Saving Mode).

Este modo permite poner el dispositivo en reposo profundo, similar a apagado, por un período de tiempo definido sin perder el registro de la conexión a la red. En este reposo el dispositivo apaga toda comunicación, pero permanece registrado en la red a pesar de ser inalcanzable por la red. De este modo, no es necesaria una reconexión cuando el dispositivo decida despertarse para transmitir.

Un equipo conectado debe comunicar periódicamente su ubicación a la red, a esto se lo conoce como Tracking Area Update (TAU). Para emplear PSM el dispositivo debe negociar con la red por cuanto tiempo estará apagado e inalcanzable. Este modo se caracteriza con dos timers, llamados T3412 y T3324. El T3412 o también conocido como TAU timer define el intervalo de tiempo entre dos TAU. Cada TAU es seguido de un timer T3324 también conocido como Active Timer, el cual genera un período de tiempo donde el dispositivo permanece accesible por la red. Finalizado ese tiempo el dispositivo se va a un estado de reposo y deja de ser accesible. La imagen [3.2] ilustra el funcionamiento de PSM.

Para hacer uso de PSM, los períodos entre TAU se pueden configurar comunicándose con la red. Se pueden configurar valores tan largos que el período de TAU puede llegar a ser de días. En todo ese tiempo el dispositivo puede permanecer en reposo pero con la contrapartida de ser inalcanzable por la red. [17]

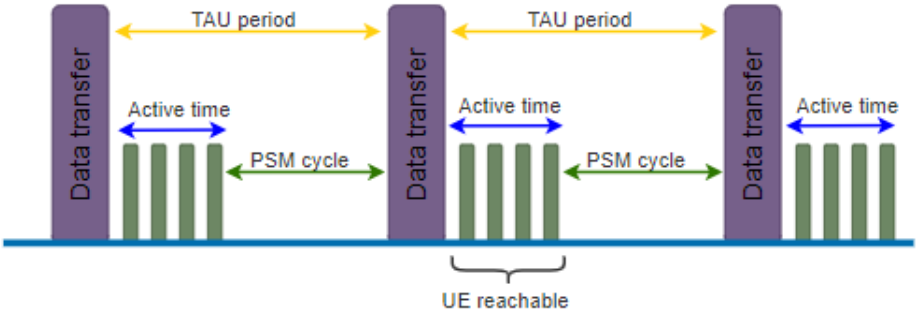

Figura 3.2: Esquema de tiempos de PSM - Power Save Mode

De todas maneras, como se mencionará posteriormente, esta configuración no es utilizada en este proyecto y se presenta solo por completitud y por haberse considerado.

eDRX: Extended Discontinuous Reception

El eDRX es una extensión de una propiedad de LTE conocida como DRX (Discontinuous Reception). Dicha propiedad consiste en apagar momentáneamente la recepción desde la red. En LTE, los tiempos en los que se apaga la recepción son períodos muy cortos (de hasta 2.56 s) pero que logran reducir el consumo de energía significativamente para equipos con tráfico constante de datos. El modo eDRX hace lo mismo pero con períodos de tiempo mucho mayores, pensado para aplicaciones IoT que no requieren ser alcanzables por la red permanentemente.

Se caracteriza con dos parámetros denominados eDRX cycle y PWT (Paging Window Timer). El PWT es el período durante el cual el dispositivo habilita la recepción y es accesible por la red. Se llama *paging* a una función que realiza la red para localizar un dispositivo en su área de cobertura y avisar si tiene mensajes

para este. El eDRX cycle es el tiempo entre dos períodos de PWT, durante el cual se deshabilita la recepción de datos. Se ilustran estos tiempos en la figura [3.3] [17]

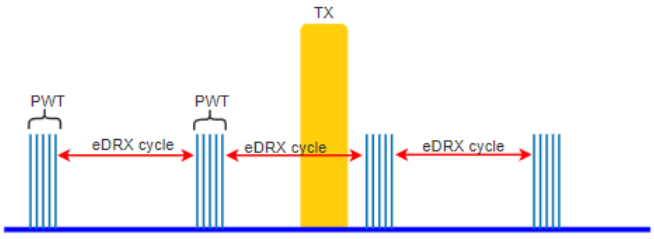

Figura 3.3: Esquema de tiempos de eDRX - Extended Discontinuous Reception

Tanto PSM como eDRX pueden emplearse por sí solos pero están diseñados para poder ser usados en conjunto. De este modo, se optimiza el consumo durante el tiempo activo de PSM empleando eDRX. Se ilustra el funcionamiento de ambos modos en conjunto en la imagen [3.4].

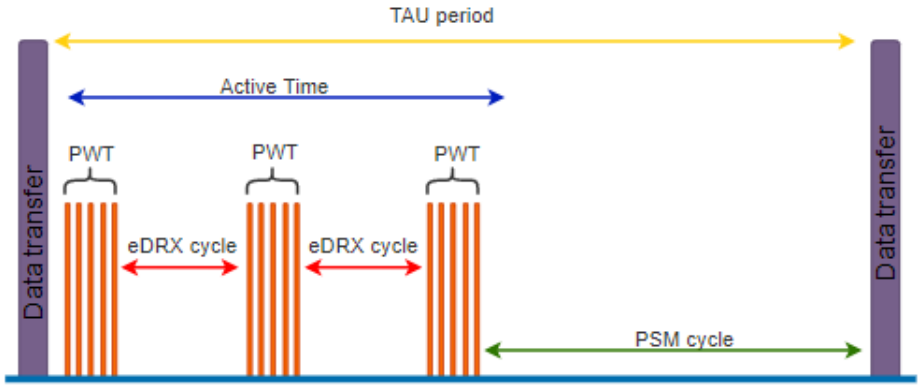

Figura 3.4: Implementación conjunta: PSM y eDRX

En lo que respecta al presente proyecto esta configuración sí fue empleada. Dado que en principio no es de interés que el Equipo sea alcanzado por la red se configuran los ciclos eDRX solicitando los más largos posibles (10485.76 segundos). Sin embargo, queda en potestad de la red, en este caso de Antel, aceptar dicha solicitud o en caso contrario, asignar el tiempo de ciclo eDRX máximo que se permite para los dispositivos de la red (varía dependiendo de la red).

3.2. Protocolo de comunicación: MQTT

Dentro de la capa de aplicación, el protocolo seleccionado fue el protocolo MQTT. El protocolo MQTT, por sus siglas en inglés Message Queue Telemetry Transport, se basa en un sistema de suscriptor/publicador. Los mensajes se dividen por tópicos. El tópico se representa mediante una cadena de caracteres y tiene una estructura jerárquica. Cada jerarquía se separa con '/'. Por ejemplo, *"datos/oveja1/ax"*. Un cliente publica un mensaje en un determinado tópico. Este mensaje le llega a cualquier otro cliente que se encuentre suscrito a ese mismo tópico. Esta comunicación se hace a través de un servidor central conocido como broker. El broker es el encargado de gestionar la red y de transmitir los mensajes a través de una comunicación TCP/IP. El broker es el responsable de mantener registro de los clientes conectados. La figura [3.5] representa el funcionamiento del protocolo MQTT. [18]

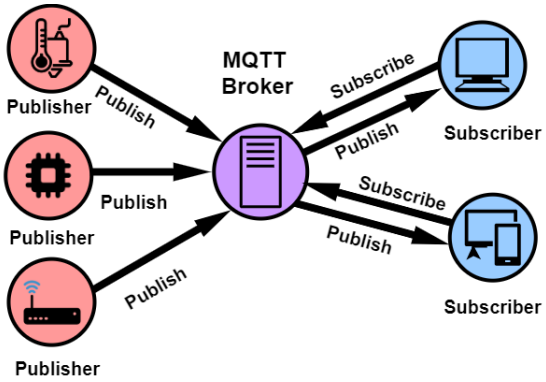

Figura 3.5: Diagrama de funcionamiento del protocolo MQTT

Para este proyecto se implementa MQTT de modo que el equipo de la oveja publica su información en un determinado tópico al cual una computadora se suscribe permanentemente para leer estos mensajes.

3.3. Comandos AT

En esta sección se presenta un resumen de los comandos AT empleados entre el microcontrolador y el módulo BG96 de la placa LTE IoT 2 click. Mediante estos comandos se dan las instrucciones para operar el módem y que este devuelva la información deseada. Estos comandos forman parte de un lenguaje conocido como *Conjunto de comandos Hayes* debido a que todos los comandos son precedidos por los caracteres "AT" (proveniente de la palabra Atención del ingles Attention) también se los conoce como *Comandos AT*. Son cadenas de caracteres que deben ser transmitidas por UART al módem del LTE IoT 2 click y este es capaz de interpretarlos.

3.3.1. Conexión a la red Narrowband IoT de Antel

Para establecer la conexión del módem LTE IoT 2 click con la red Narrowband IoT de Antel se debe seguir una secuencia de comandos. Estos configuran parámetros de la conexión y definen el punto de acceso y operador al cual conectarse. El procedimiento correspondiente para poder conectarse se detalla en el anexo E. Cabe destacar que dicha secuencia se debe realizar una única vez, luego queda guardada esta configuración en memoria del módem.

3.3.2. Conexión al broker MQTT de ANTEL

En este proyecto el broker utilizado es proporcionado por Antel. Utilizando comandos AT se ingresa al gateway MQTT de Antel con el puerto 1883. El flujo de comandos consiste en establecer la conexión a la red MQTT identificándose con un nombre de cliente (nombre con el cual se identificará el nodo que se esté conectando), un usuario y contraseña para luego publicar mensajes al tópico que se desee. Con la conexión establecida se puede publicar cualquier número de mensajes. Si no se desea seguir transmitiendo, se debe cerrar la conexión. Los comandos a emplear son los siguientes:

AT+QMTOPEN Abre la red MQTT ingresando URL y puerto especificado

AT+QMTCONN Ingresa usuario y contraseña de la red MQTT

AT+QMTPUB Publica un mensaje al tópico de la red MQTT especificado

AT+QMTDISC Cierra la conexión MQTT

Este procedimiento se encuentra explicado con un ejemplo en el anexo E y en el sitio web de Antel [15].

3.3.3. Inicializaciones

Estos comandos se corren una única vez cuando se enciende el módem. Realizan configuración necesaria para la utilización del módem.

ATE0 Deshabilita el eco de comandos

AT+QMTCFG="keepalive",0,3599 Configura el tiempo de vida con la red Narrowband

Por defecto el módem se enciende con el eco de comandos habilitado. Esta función hace que cuando se le da un comando al módem este responde ese exacto comando antes de realizar lo que se ordena. Esto no es necesario ya que además introduce retardos y aumenta el consumo, por ende se desactiva. Por otro lado, el *keepalive* es un parámetro que se negocia con la red Narrowband IoT cada vez que el módem se conecta. El parámetro *keepalive* es el intervalo de tiempo máximo en segundos en el que se puede estar sin comunicarse con el broker y que no se pierda

la conexión. En caso de superar este tiempo y que el broker no haya recibido nada del cliente, el broker da por finalizada la conexión. Se configura un tiempo largo para que el módem gaste menos energía en dar la señal de *keepalive* a la red. Por lo tanto se configura el tiempo máximo que es 3599 segundos.

3.3.4. Comandos relacionados a la función GPS

Para obtener información del GPS, se sigue el siguiente flujo de comandos. Consiste en activar la función de GPS dentro del BG96, pedirle la información y apagarlo si así se desea.

AT+QGPS=2 Prende el GPS en modo asistido por la red

AT+QGPSGNMEA="GGA" Pide posición geográfica y hora actual al GPS

AT+QGPSEND Apaga el GPS (Opcional)

Luego de enviado el primer comando el GPS se prende. El GPS trata de fijar su posición buscando señal con los satélites. El modo asistido por la red significa que el módulo toma información de la radio-base a la cual está conectado por Narrowband IoT. Así logra fijar su posición más rápidamente y obtiene información horaria más precisa. El segundo comando es el encargado de pedir información al GPS el cual devuelve en formato de sentencias NMEA (acrónimo de *National Marine Electronics Association*) de tipo GGA. Esto define la información que devuelve y en qué formato. Es una cadena de caracteres que el microcontrolador obtiene por UART. El formato GGA en cuestión devuelve la información en la siguiente secuencia de caracteres ASCII:

$$\$GGA, hora, latitud, N/S, longitud, E/W, ... \tag{3.1}$$

Incluye más información a continuación, pero no es de interés a la aplicación. El software embebido implementado se encarga de desglosar esta información y guardarla en un formato manejable para el código.

3.4. Sistema Central

La Figura [3.6] muestra el diagrama de bloques que componen el Sistema Central. El Sistema Central se compone de 3 módulos. Por un lado el módulo encargado de la recepción, procesamiento y guardado de datos. Por otro lado la base de datos donde se guardan todos los datos y finalmente el módulo encargado de generar una interfaz gráfica para la visualización de los datos.

3.4.1. Base de datos

Para la base de datos se utiliza un sistema de gestión de base de datos llamado Postgres. Postgres utiliza un lenguaje llamado PL/pgSQL (Procedural Language/PostgreSQL Structured Query Language) el cual permite ejecutar comandos

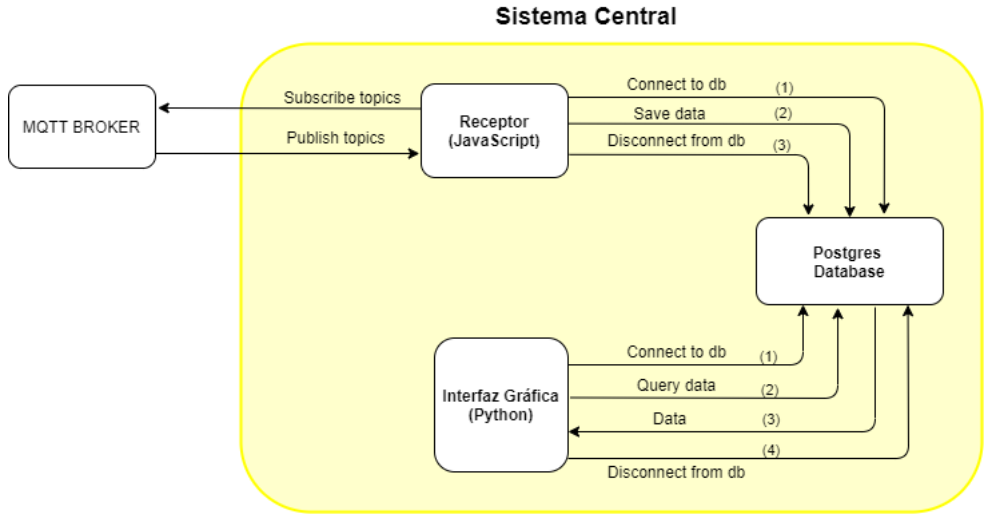

Figura 3.6: Diagrama de módulos del Sistema Central

SQL (Structured Query Language en español conocido como Lenguaje de Consulta Estructurado). A través de Postgres se crea nuestra propia base de datos donde se guarda toda la información. Postgres tiene dos grandes ventajas. La primera es que es un programa gratis, por lo tanto cualquier persona tiene acceso al software y la segunda ventaja es que posee una interfaz fácil de entender y facilita consultas SQL que permiten gestionar toda la información.

3.4.2. Receptor

El módulo receptor es el encargado de recibir los datos, procesarlos y guardarlos en la base de datos. Dentro de los datos que llegan y se guardan en la base de datos se pueden identificar 3 tipos de datos: los datos provenientes de los 3 ejes del acelerómetro, los datos del GPS y el resultado de la identificación de Estado. La recepción de datos se crea de manera que sea escalable. Se implementa un script en JavaScript que genera una base de datos y guarda la información recibida en la tabla correspondiente al tópico del cual se recibe el mensaje. De modo que es fácil determinar qué información se adquiere y para agregar otra clase de datos alcanza con crear otro tópico. El diagrama de la Figura [3.7] muestra el funcionamiento del módulo receptor.

3.4.3. Interfaz gráfica

Para el despliegue de los datos de una forma visual, se opta por trabajar en el lenguaje de programación Python, usando una librería llamada Tkinter. En Python se pueden implementar distintas variedades de interfaces de usuario siendo Tkinter una de ellas. Tkinter tiene la ventaja de que es compatible con Windows, macOS y Linux. Además es una librería que ya viene por defecto instalada en Python. Si bien es una librería estándar con funcionalidades muy básicas, para el alcance de

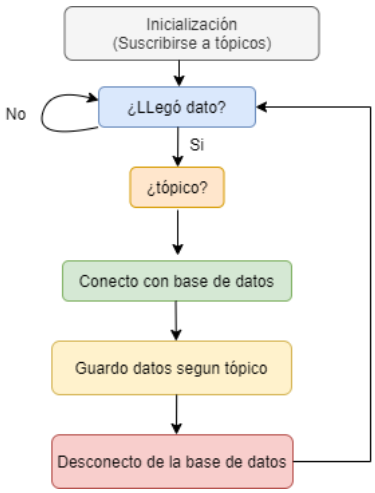

Figura 3.7: Diagrama del módulo Receptor

este proyecto cumple con las características necesarias.

Además de Tkinter se utilizan otras dos librerías. Se utiliza Matplotlib que permite generar todas las gráficas pertinentes y se utiliza Basemap que permite genera un mapa geográfico y graficar las coordenadas geográficas.

La interfaz se diseña para que sea dinámica y el usuario a través de los distintos botones pueda navegar por distintas pantallas visualizando los datos que desee. Además se puede configurar para que la interfaz actualice los datos cada cierto período de tiempo. También se incorpora una funcionalidad donde el usuario puede exportar los datos de la base de datos a archivos .CSV para su uso personal. La figura [3.8] muestra las diferentes pantallas de la interfaz gráfica. En el Anexo F se encuentra un instructivo de uso del Sistema Central.

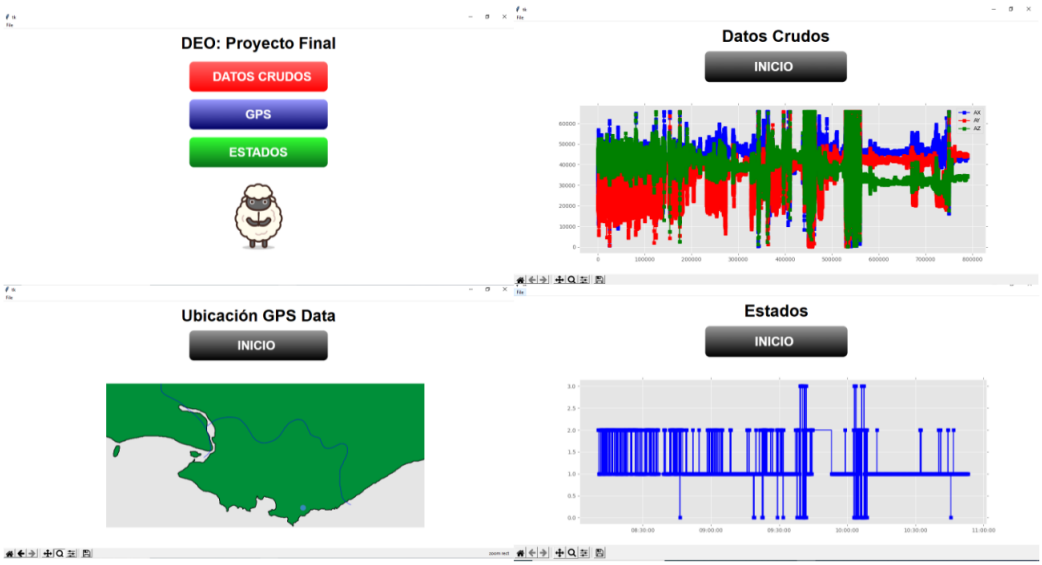

Figura 3.8: Interfaz gráfica

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 4

Clasificación de Estados de la oveja

En general, la problemática del presente trabajo se reduce a identificar Estados en los que se encuentra la oveja en cada momento para que en un futuro pueda encontrarse una correlación entre estos y el parto. Se procede entonces a describir el proceso de desarrollo y funcionamiento del módulo clasificador de Estados implementado. El primer enfoque fue el de caracterizar el comportamiento relevando el ángulo de la posición del cuello del animal mientras que el segundo se centra en aplicar un algoritmo de aprendizaje automático. Para la solución final se decidió optar por este segundo método.

Mayor desarrollo sobre el funcionamiento del primer clasificador puede encontrarse en el Apéndice B.

Se accedió a un conjunto de datos de dos ovejas etiquetados según su comportamiento (Kamminga et al., 2017 [21]) así procesándolos con un algoritmo de aprendizaje supervisado (Le Roux et al., 2017 [22]).

El método empleado se denomina análisis discriminante lineal y busca asignar a un vector de atributos d-dimensional $x = \{x_1, ..., x_d\}$ una de las k clases en las que se pueden dividir los datos, aplicando un límite de decisión lineal. A su vez, el método sirve para reducir la dimensión del vector x hallando la proyección en un espacio de dimensión $K - 1$ o menor que maximice la variabilidad entre clases y minimice la variabilidad dentro de cada clase. Es decir, pensándolo en términos geométricos, se busca que las clases proyectadas sean tales que los datos de cada una estén apreciablemente separados según su clase pero que los datos dentro de cada una estén lo más próximos posible. Las hipótesis que el método toma son que las clases tienen una distribución gaussiana multivariada, lo cual cualitativamente es una generalización de la distribución gaussiana para dimensiones mayores a uno y que las clases posean la misma matriz de covarianza.

Considérense las siguientes definiciones:

- $P(y = k|x)$: probabilidad de que teniendo un cierto vector de atributos calculado (x) , la clase a la que pertenece (y) sea igual a la clase k.
- ω_k y ω_{k0}: matrices que se obtienen del entrenamiento.
- Cst: es una constante independiente de la clase.

Entonces, puede demostrarse (mayor detalle sobre los aspectos matemáticos pueden encontrarse en el Apéndice B.2) que el clasificador se reduce a evaluar para cada clase k la siguiente expresión:

$$log(P(y = k|x)) = \omega_k \cdot x + \omega_{k0} + Cst \tag{4.1}$$

De aquí que la clase a la que pertenece el vector de atributos es la que maximiza $log(P(y = k|x))$ y por consiguiente es la de mayor probabilidad. Además, por la forma de la expresión presentada, se ve que la superficie de decisión del clasificador es lineal.

Se observa también que Cst no es necesario considerarlo en la decisión del máximo ya que es el mismo para todas las clases.

La técnica fue elegida por su simplicidad en la implementación y su bajo costo computacional ya que tal como se aprecia en la ecuación [4], consiste en realizar algunas multiplicaciones matriciales y hallar un máximo a nivel del microcontrolador.

4.1. Análisis sobre algoritmos de clasificación

4.1.1. Condiciones de prueba

Se analiza la efectividad sobre los dos algoritmos de clasificación mencionados. El primer acercamiento, de ahora en más llamado Algoritmo por filtrado, utiliza los datos de un acelerómetro junto con los datos de un giroscopio para calcular el ángulo de la posición del cuello del animal utilizando un filtro complementario. El filtro complementario combina los datos del acelerómetro con los del giroscopio para una mejor estimación. Para más detalle sobre el primer algoritmo ver apéndice B. El segundo acercamiento, cuya implementación es explicada en la sección 4.1.3, se basa en aplicar un algoritmo de aprendizaje automático (ML). De aquí en más este algoritmo será llamado Algoritmo por ML,

Para ambos algoritmos de clasificación propuestos, se realizaron pruebas sobre bases de datos publicas de comportamiento en ovejas vinculadas con el estudio [21]. Dichas bases de datos contienen información obtenida de acelerómetro y giroscopio (entre otros sensores) registrada con un collar, en dos ovejas. Los datos están etiquetados en el tiempo con la actividad que realiza el animal, para que se corresponda con el momento en que se relevó la información. Las etiquetas según el estudio refieren a las siguientes actividades:

- Caminando
- Parada
- Pastando
- Corriendo
- Trotando

Los datos fueron procesados en Python con funciones principalmente de las librerías *sklearn*, *numpy* y *pandas*. El entorno elegido para verificar los algoritmos de esta manera fue *Visual Studio Code*.

En una etapa de verificación anterior, el clasificador que consiste en implementar un filtro complementario, también tuvo una prueba cualitativa de funcionamiento con herramientas de *Code Composer Studio*.

4.1.2. Clasificador de Estados: Algoritmo por filtrado

Para validar el Algoritmo por filtrado, en principio se generaron datos moviendo el Equipo con un aparato hecho en madera donde se fija la placa y se tiene movimiento angular según los tres ejes de coordenadas (ver figura [4.1]). El mecanismo construido pretendía simular comportamientos rudimentarios de una oveja. Inicialmente se realizó una inspección de datos de las variables y buffers de interés en la ventana de expresiones del *Code Composer Studio*. Luego, para poder tener una visualización en tiempo real, se introdujo un breakpoint en el lazo principal con la opción *refresh all windows* de CCS así pudiendo ver los cambios cada vez que el código pasa por dicho breakpoint. El estudio cualitativo mencionado arrojó resultados aparentemente positivos por lo que prosiguió un estudio sobre datos más representativos de la realidad. En esta instancia se decidió descartar este algoritmo optando por el Algoritmo por ML.

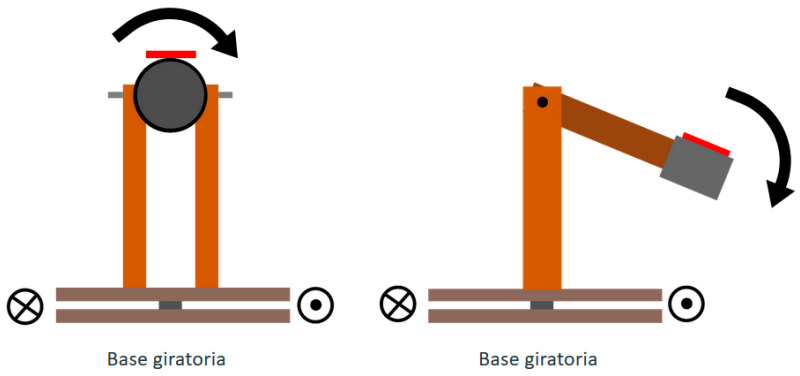

Figura 4.1: En rojo la placa con los sensores adosada al mecanismo hecho en madera para emular los movimientos.

El algoritmo fue desarrollado en el marco del proyecto final de la asignatura Sistemas Embebidos para tiempo real [14]. La demostración del funcionamiento de este algoritmo puede verse en [23].

Los supuestos principales del funcionamiento del algoritmo fueron que el pastar del animal estaría estrechamente relacionado con la posición del cuello y que los sensores de adquisición se encontrarían en la parte superior del cuello. Por lo tanto, para el análisis realizado con Python, se reetiquetaron los datos de forma tal que pastar correspondiera a cuello bajo y, en caso contrario, a cuello alto. Luego de realizar la manipulación de datos acorde y calcular los ángulos del cuello

de la oveja mediante el filtro complementario (detalles sobre el procesamiento se explicitan en el apéndice B) se decidió realizar el gráfico de la dispersión de los ángulos calculados con las muestras en función de su clase. Lo esperable sería que hubiera mayor concentración de datos de cabeza alta en un rango de ángulos y de cabeza baja en otros. En la figura [4.2] se deja el gráfico de lo mencionado anteriormente.

Es importante destacar que no se realizó ningún tipo de procesamiento de corrección del ángulo inicial del cuello de la oveja. Esto significa por una parte que siempre se consideró al animal en un terreno plano y por otra que la posición del collar con respecto al cuello de la oveja tampoco se tomó en cuenta (aun si el collar pudiera haberse movido durante la captura de los datos de la base). Sin ningún tipo de procesamiento, el ángulo calculado depende en gran medida de la inclinación del terreno y de la posición del collar con respecto al cuello.

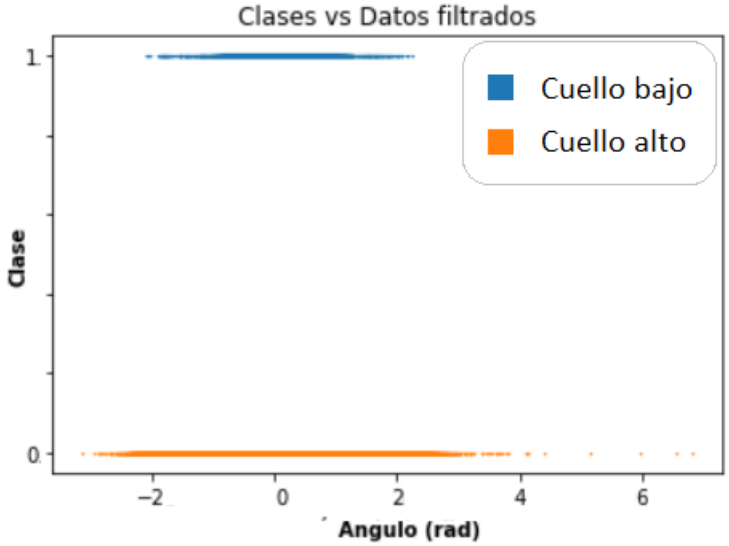

Figura 4.2: Gráfico con ángulos calculados contra su clase.

Analizando la gráfica [4.2] se observa una mayor concentración de puntos en cierto rango cuando la oveja está con lo que se supone es el cuello bajo. Sin embargo, los ángulos correspondientes al cuello alto se encuentran en gran parte en el mismo rango que los de cuello bajo lo cual haría muy difícil la distinción entre un caso y el otro. Se le atribuye la causa de esto por un lado a que, tal como dice [21], el collar no se encuentra fijo en la parte superior del animal a lo largo de la adquisición. Por otra parte, del gráfico [4.3], donde se hizo un nuevo etiquetado diferenciando entre todos los Estados de la base de datos, puede observarse que para las actividades que involucran mayor movimiento el rango de ángulos que ocupan es mayor. Con respecto al nuevo etiquetado, en este se consideran el trotar y el correr como lo mismo ya que es insuficiente la información que brinda la base de datos para saber que consideraron en su momento como un Estado u el otro.

Esto no implica directamente que el cuello de la oveja haya pasado por todos esos ángulos si no que permite inferir que frente a movimientos abruptos el

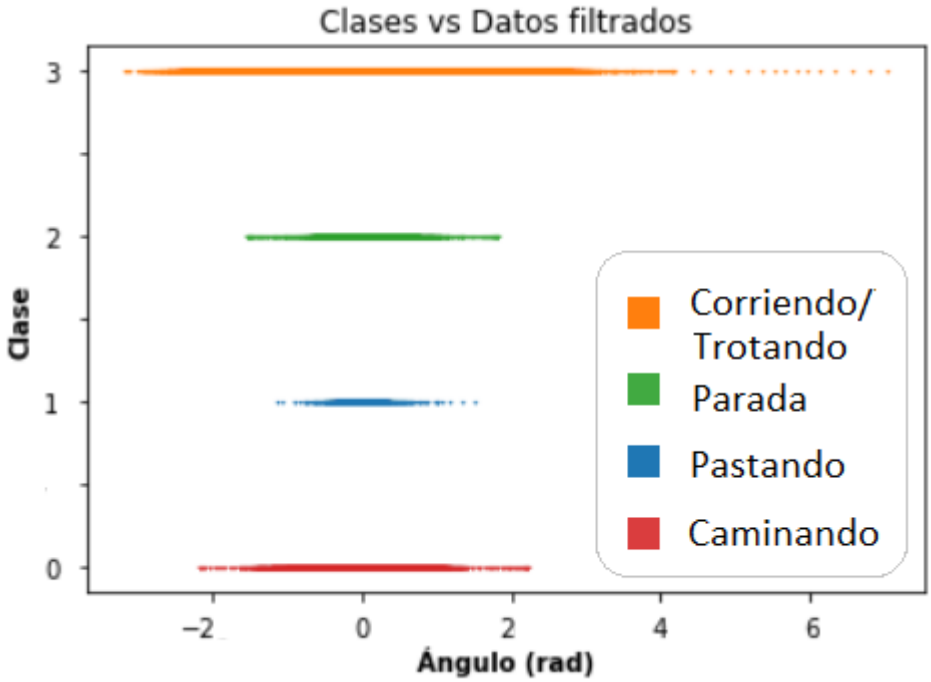

Figura 4.3: Gráfico con ángulos calculados contra su clase nueva.

filtro complementario arroja resultados impredecibles. Resulta claro entonces que el algoritmo no puede ser implementado como era deseado. Podría analizarse si un pre-procesamiento del tipo de aplicar un filtro pasa-bajo a los datos del acelerómetro y otro pasa-alto a las adquisiciones del giroscopio mejora la habilidad de clasificar. Sin embargo, se decidió no ahondar en este método pues además es deseable que la información a obtenerse a partir de los sensores sea indiferente a cambios en la posición del collar o a la inclinación del terreno. Por consiguiente, se opta por un enfoque vinculado al aprendizaje automático sin siquiera realizar pruebas en campo con este algoritmo.

4.1.3. Clasificador de Estados: Algoritmo por ML

Al no ser posible que el método de clasificación anterior cumpla con lo requerido con algún ajuste sencillo, se opta por aplicar el método de aprendizaje automático (ML) del Análisis del Discriminante Lineal (LDA). Se toma como base el trabajo de Le Roux et al [22], que estudió el compromiso entre cantidad de atributos calculados con datos de un acelerómetro y tasa de éxito al emplear un clasificador LDA.

Puede encontrarse mayor explicación sobre el método de ML en el Apéndice B. Las pruebas hechas por [22] de mayor relevancia para este proyecto consistieron por una parte en considerar vectores de 12 atributos con operaciones poco costas en términos de procesamiento computacional para clasificar. En dicho escenario, tuvieron una tasa de éxito en la predicción del 82,40 % de datos. Se entiende por

éxito a que la predicción del Estado del animal coincida con el Estado real en ese momento. Por otro lado, consideraron diferentes parámetros estadísticos que pudieran identificar las clases sin importar la posición en el cuello del collar con los sensores. Para dicho análisis consideraron 31 posibles atributos y los resultados del compromiso se reproducen en la gráfica de la figura [4.4].

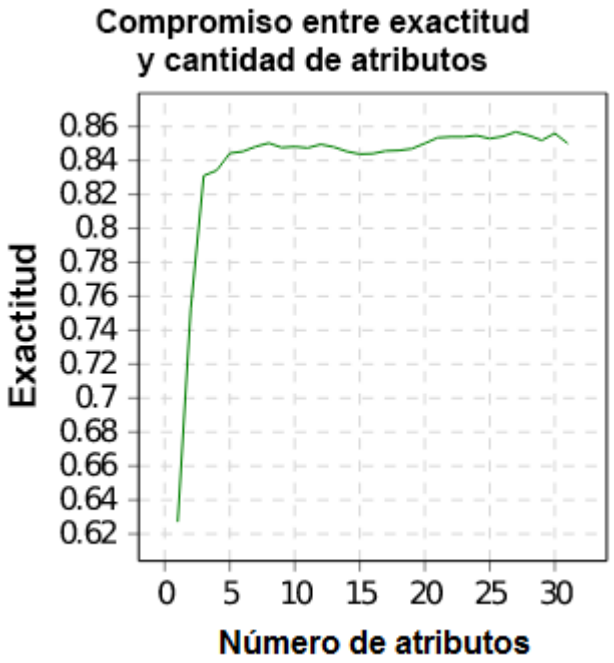

Figura 4.4: Gráfico del compromiso entre atributos y exactitud traducida y reproducida de [22]

Se define exactitud a la cercanía de una medida al valor real. En este caso, se mide el porcentaje de aciertos en la clasificación de Estados. Con los primeros cinco atributos que emplearon (se muestran en la tabla 4.1) la exactitud obtenida es mayor al 80 %. Cabe destacar que el mínimo del eje X y el mínimo del eje Y cuentan como 2 atributos independientes.Sin embargo, ellos no llegaron a implementarlo en el sistema en el collar si no que el análisis fue hecho en laboratorio en una computadora posteriormente.

Partiendo de estos resultados de laboratorio, se propone implementar un algoritmo para clasificar el Estado del animal en tiempo real que continúe con esta linea de estudio.

Atributo	**Ecuación**
Desviación estándar (eje Z): σ	$\sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \bar{x})^2}$
Valor mínimo (ejes X e Y)	$min(x)$
Varianza (eje Z)	σ^2
Entropía espectral (eje Z)	$\sum_{i=1}^{N} P(X_i) log \frac{1}{P(X_i)}$

Tabla 4.1: Atributos de mayor peso para la clasificación. La FFT de x se denota como X y $P(X)$ corresponde a la densidad espectral de potencia de X.[2]

Las pruebas para este acercamiento emplearon las mismas herramientas en Python que para la primera parte, con la adición de la librería *sklearn*. Estas pruebas buscarían determinar qué atributos calcular con los datos de la IMU para aplicar el LDA.

Para definir completamente el Algoritmo por ML se buscó decidir que vector de atributos emplear analizando los siguientes puntos:

- Considerar un vector de atributos de los datos crudos y ver su tasa de éxito para clasificar.
- Verificar si el empleo de datos recabados con el giroscopio en algoritmo impacta positivamente en la tasa de éxito.
- Verificar la tasa de éxito con atributos similares a los de la tabla 4.2 (se descarta la entropía espectral presentada en 4.1 por la complejidad de su cálculo en un Sistema Embebido).

Para todos los entrenamientos y pruebas con el LDA se tomaron de los datos 80 % para entrenar y el otro 20 % para verificarlo.

Por consiguiente, se tuvo que luego de entrenar el LDA con datos crudos del acelerómetro de una oveja sola, la predicción resultaba ser de un 82,1 %. Sin embargo, tan buen resultado se presume sea causa solamente de la consistencia que existe en las aceleraciones dadas por un mismo animal. Al intentar aplicar el algoritmo ya entrenado sobre los datos de un segundo animal, rápidamente se tiene una tasa de éxito del 44,75 %. Aún así, resulta claro que al menos para un mismo animal, las aceleraciones según cada Estado contienen información suficiente para caracterizarlo. Se descarta entonces usar un vector de atributos de datos de una oveja sola ya que la predicción no es generalizable a más de un animal.

Posteriormente, se decidió considerar los datos de ambas ovejas a la vez para no tener una tasa de éxito inflada por la correlación entre aceleraciones de un mismo animal. El algoritmo fue luego entrenado con los atributos que se indican en la tabla 4.2 para cada eje, tanto del acelerómetro como del giroscopio (similar al vector que utiliza [22] pero con la adición de los datos de giroscopio). Se utilizan los mismo atributos en el giroscopio y se agregan al vector de atributos. Con el

[2] $P(X_i) = \frac{|X_i|^2}{\sum_i |X_i|^2}$

fin de reducir posibles costo de procesamiento en el microcontrolador se optó por quitar el atributo de entropía.

Contrario a la intuición, el aporte de información del giroscopio (con las variables consideradas) hace que el resultado de predicción sea de tan solo 69.72 % de aciertos.

Atributo	**Ecuación**
Desviación estándar (eje Z): σ	$\sqrt{\frac{1}{N}\sum_{i=1}^{N}(X_i - \bar{X})^2}$
Valor mínimo (ejes X e Y)	$min(x)$
Varianza (eje Z)	σ^2

Tabla 4.2: Atributos aplicados a los tres ejes del acelerómetro y del giroscopio

Finalmente, visto los resultados anteriores, se decide descartar el aporte del giroscopio y entrenar el algoritmo con los cuatro atributos de la tabla 4.2.

Se obtiene un resultado de 85,7 % de aciertos que coincide en buena medida con el ilustrado en la gráfica [4.4] que muestra el compromiso entre la exactitud y la cantidad de atributos. Donde se puede apreciar que para 4 atributos la exactitud esta por arriba del 80 %. Demostrando ser el mejor de entre los analizados.

Luego, es interesante ver la matriz de confusión de la clasificación del algoritmo entrenado (Tabla 4.3). Se recuerda que la matriz de confusión es una herramienta para visualizar el desempeño de un algoritmo donde cada columna representa el número de predicciones de cada clase, mientras que cada fila representa lo observado en la realidad. Se considera la siguiente relación entre Estados y variables categóricas:

- caminando;
- parada;
- pastando;
- corriendo o trotando;

Obs\Pre	**Caminando**	**Parada**	**Pastando**	**Corriendo**	**Total**
Caminando	166	0	65	0	231
Parada	3	652	32	1	688
Pastando	47	118	664	1	830
Corriendo	3	0	0	180	183
Total	219	770	761	182	1932

Tabla 4.3: Matriz de confusión del algoritmo entrenado. Observaciones contra predicciones.

Se observa que las mayores confusiones se dan entre el Estado de pastando y parada. No obstante lo anterior, se recuerda que además de las etiquetas, no se

posee ninguna especificación ulterior de que implica cada Estado por lo cual un mayor análisis podrá hacerse luego de efectuadas las pruebas de campo. Además, puede observarse cierta consistencia entre la matriz de confusión y el gráfico tridimensional de los vectores de atributos proyectados con el LDA [4.5]. Más aun, resulta interesante como el aspecto probabilístico de pertenecer a una clase u otra se ve desde un ámbito geométrico. En otras palabras, es coherente creer que los datos correspondientes al Estado caminando se encuentren entre los datos de correr y estar parada tal como se ve. También resulta intuitivo pensar que los datos productos del correr se encuentren más dispersos tal como se observa. Finalmente, la oveja parada o pastando resulta en vectores proyectados muy cercanos lo cual también es consistente.

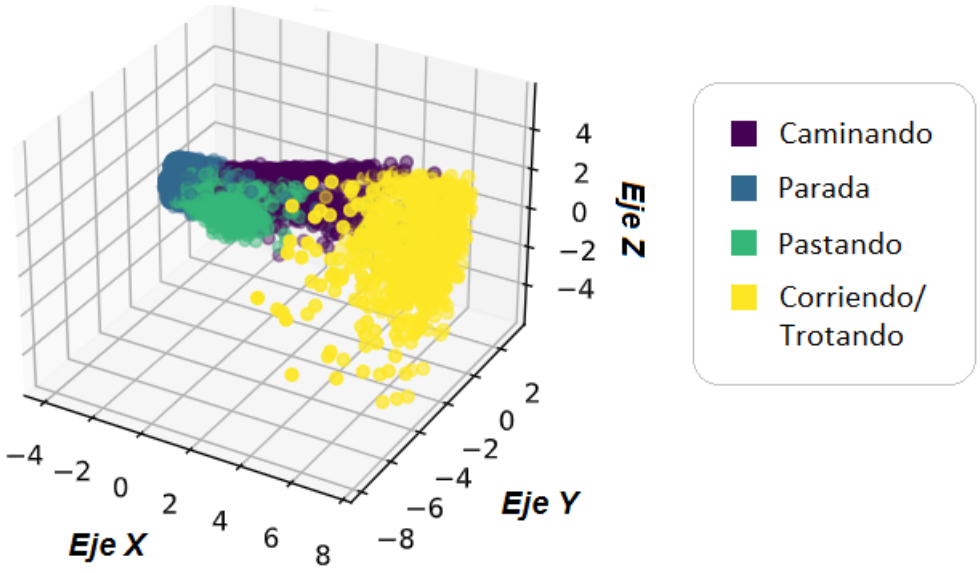

Figura 4.5: Gráfico tridimensional de los vectores de atributos según su clase transformados con LDA

4.2. Conclusiones

Dadas las pruebas y los antecedentes (Le Roux et al., 2017 [22]) el vector de atributos que se decidió considerar se obtiene tomando ventanas de 500 valores muestreados a 100 Hz de los datos de aceleración únicamente. El vector consiste de:

- Desviación estándar de la aceleración en el eje z.
- Mínima aceleración en el eje x.
- Mínima aceleración en el eje y.
- Varianza en el eje z.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 5

Diseño del software embebido

5.1. Descripción general

El presente capítulo detalla el proceso de diseño del software embebido del Equipo[1], abarcando tanto su arquitectura más general como la descripción de los módulos que lo componen. El software embebido fue programado utilizando el lenguaje C en el entorno de desarrollo Code Composer Studio (CCS) versión 10.1.1, de Texas Instruments.

El programa se encarga de adquirir datos de la unidad de medición inercial (IMU), procesarlos y clasificarlos implementando un algoritmo de aprendizaje automático (Machine Learning, ML) así pudiendo caracterizar el Estado de la oveja en un momento dado. También, se controla la interacción con el GPS del módulo LTE IoT 2 click para obtener la posición geográfica. Una vez obtenida cierta cantidad de datos, dependiendo del modo de funcionamiento, se arma un paquete que contiene las clasificaciones de Estado, ubicación geográfica y hora actual. Luego, intercambiando mensajes con el módulo de transmisión LTE IoT 2 click mediante comunicación UART, se envía este paquete o paquetes por la red Narrowband IoT al Sistema Central utilizando el protocolo MQTT en capa de aplicación.

El Equipo cuenta con dos modos de funcionamiento con distintos fines. El modo empleado es determinado únicamente al momento que se enciende el Equipo e implica diferencias en la lógica del software. Esta implementación de modos será explicada a lo largo del capítulo.

5.2. Arquitectura del software embebido

La arquitectura de software elegida es Round-Robin con interrupciones. Dicha arquitectura se basa en un bucle principal donde se consultan distintas banderas indicadoras de diferentes eventos. Para cada evento señalizado por una bandera se tiene una serie de acciones correspondientes. Además, se incorpora el uso de un semáforo para evitar colisiones en las secuencias de mensajes a enviar de

[1]Puede accederse al repositorio en: https://gitlab.fing.edu.uy/nicolas.finozzi/deo-tesis

los módulos que comparten la UART. Se presenta el diagrama de flujo de esta arquitectura en la figura [5.1]. Cabe destacar que el diagrama es lo suficientemente genérico como para representar ambos modos de funcionamiento, indicándose donde es pertinente si la implementación difiere entre un modo y el otro, aún si la lógica general ilustrada la comparten.

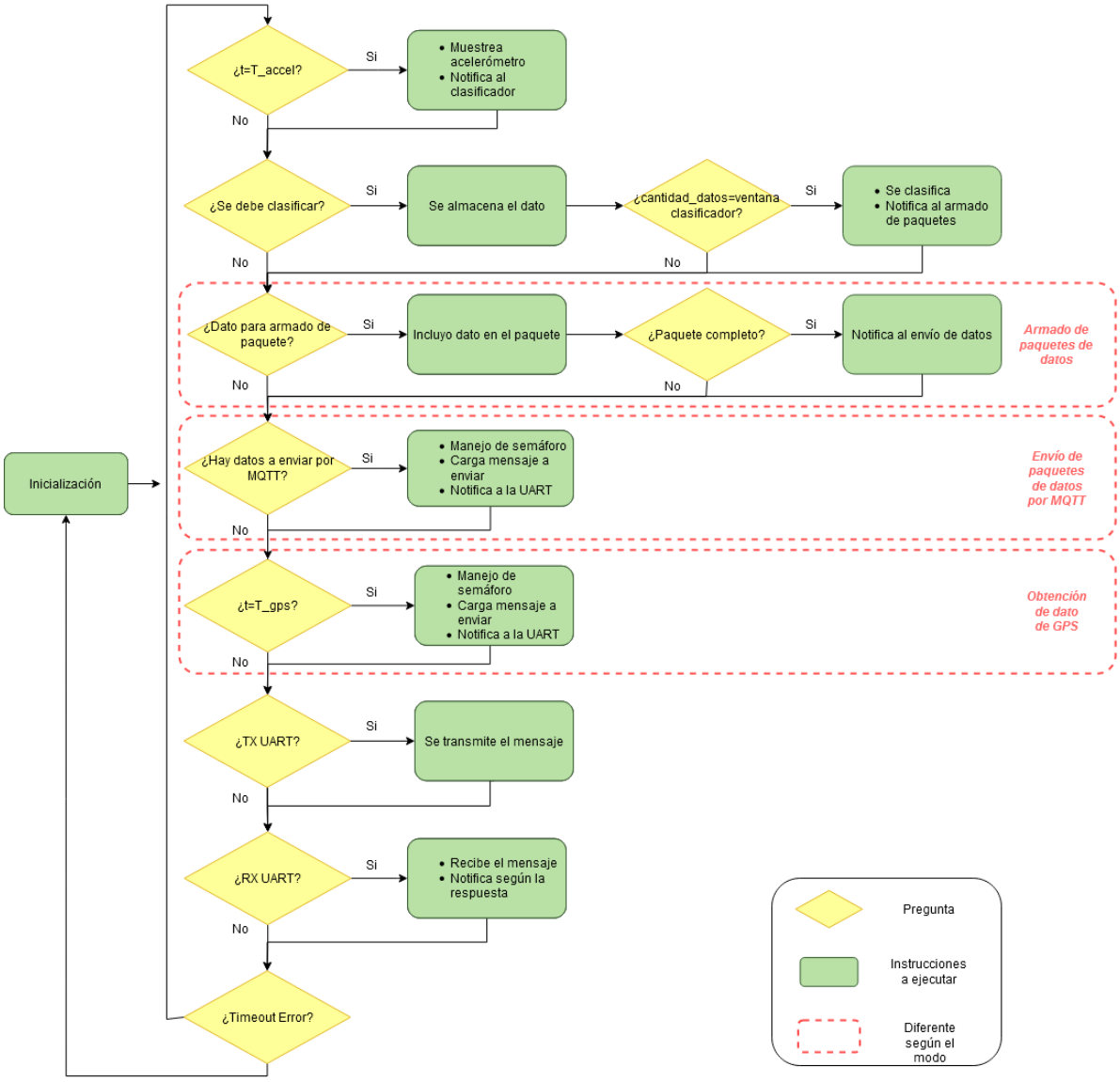

Figura 5.1: Diagrama de flujo de la arquitectura Round-Robin

En el diagrama de la figura [5.1] se representan todos los eventos que determinan el flujo del código, junto con sus acciones correspondientes. A continuación se explica en términos generales la lógica de este flujo.

Primero se consulta la bandera "flagTmrAccel", la cual es encendida por un temporizador que interrumpe una vez pasado un tiempo predefinido T_accel, de-

terminando el momento de efectuar un muestreo del acelerómetro. Luego de muestrear, debe devolverse el dato correspondiente y notificar al módulo clasificador si hay un mensaje a clasificar. Entonces, el módulo clasificador almacena el dato y en caso de tener la cantidad suficiente para realizar una clasificación, devuelve el resultado notificando al encargado de armar los paquetes de datos. El armado de paquetes consiste en incluir el dato nuevo y notificar cuando el o los paquetes armados estén completos para que se envíen por MQTT. Cuando esto sucede, el módulo MQTT prepara los mensajes a enviar encapsulando el paquete en los comandos correspondientes al Módem. Por otra parte, si debe realizarse un muestreo en el GPS, se devolverá el comando para el Módem correspondiente a dicha operativa. Tanto los comandos referentes a MQTT como al GPS son enviados (TX) por UART y esta también se encarga de su recepción (RX) generando la interrupción acorde. Al final del flujo se consulta si se detecta un error del tipo *Timeout*. Esta es una medida que soluciona casos de error en que alguna parte de la lógica está tomando más tiempo del debido o no sucede un cierto evento esperado.

Las interrupciones interactúan en el flujo de dos formas. Mediante interrupciones generadas por el módulo timer se encienden las banderas relacionadas con el muestreo "flagTmrAccel" y "flagTmrGPS". Esta última bandera funciona de manera análoga a "flagTmrAccel" pero para indicar el momento de obtener un dato de GPS es decir, luego de pasado un tiempo T_gps. Además, la comunicación UART con el módulo de transmisión y GPS se implementa con interrupciones. Una vez que todas las banderas están apagadas, no hay acciones a realizar y se pone al microcontrolador en un modo de bajo consumo. Del cual sale cuando una interrupción lo despierte.

Adicionalmente, se incorpora el uso de una lógica de "semáforo" (ver sección 5.3) por limitaciones en la comunicación entre módulos hardware. Se tiene una única vía de comunicación serial (UART) para controlar dos funciones externas al microcontrolador, que son la transmisión de datos y muestreo del GPS. El semáforo evita que estas se interrumpan, mientras otros procesos siguen corriendo. A nivel de código, el semáforo ocupa una variable cuando se inicia una secuencia de comandos y se libera al terminar.

La arquitectura en cuestión fue elegida principalmente por su flexibilidad y simpleza para integrar los modos de funcionamiento. El código resulta fácilmente escalable ante la necesidad de agregar nuevas tareas que se puedan realizar mediante interrupciones.

Se continúa comentando sobre los modos de funcionamiento y explicando las diferencias que se indican en el diagrama de la figura [5.1] con el punteado rojo.

5.3. Semáforo

La importancia de esta implementación recae sobre el hecho de tener una única UART para comunicarse con el módulo. Cada mensaje recibido debe ser interpretado de forma distinta en contexto de la tarea que se está realizando.

Tanto el GPS como el módem NB-IoT están contenidos en un mismo módem de hardware, LTE IoT 2 click. Sin embargo, el módem tiene una única vía de

comunicación con el microcontrolador, la comunicación serial UART. Para manejar ambas funcionalidades, que podrían llegar a funcionar en simultáneo en el módulo, se implementa en el software embebido un semáforo.

El semáforo consiste en una bandera denominada *token*, la cual los módulos que se ocupan del GPS y transmisión gestionan y verifican antes de utilizar la UART.

En caso de que la bandera ya estuviera tomada por algún módulo, este podrá seguir su lógica de funcionamiento cuando corresponda mientras que el otro módulo seguirá consultando si la bandera se liberó cada vez que se recorra el bucle principal. Esto impide que las tareas relacionadas al módulo LTE IoT 2 click se obstruyan mutuamente, caso que ocurriría por ejemplo si se quisiera pedir un dato al GPS mientras se está esperando una respuesta de confirmación de la transmisión por parte del LTE IoT 2 click. Para que la bandera *token* se libere tiene que haberse finalizado la tarea, dando paso a otras.

Adicionalmente, resulta que el muestreo de datos del acelerómetro se efectúa a través de interrupciones I2C las cuales podrían llegar a ocasionar pérdidas en los mensajes recibidos por la UART (ya que mientras se atiende una interrupción se dejan de escuchar nuevas). Por lo tanto, el semáforo cumple una segunda funcionalidad donde los módulos encargados de muestrear, clasificar y armar paquetes deben consultar que el *token* está libre para proseguir con sus funciones.

5.4. Modos de funcionamiento

En un principio, el procesamiento de datos del acelerómetro en el microcontrolador para generar el Estado del animal tiene como justificación el disminuir la cantidad de paquetes enviados. Ello resulta deseable ya que por un lado la red Narrowband IoT para uso comercial no es gratuita y por otro se aumenta la autonomía del dispositivo sin tener que cambiar las baterías.

Resulta que para cada clasificación se requieren 1500 datos crudos (números enteros) que no serían necesarios de enviar durante la investigación del comportamiento (500 muestras por eje del acelerómetro). Sin embargo, en la etapa preliminar de validación del algoritmo, se requieren los datos crudos al igual que el Estado para verificar que lo clasificado coincida con la realidad.

Ante esta situación, se integran dos modos de funcionamiento en el software embebido según se indica en la tabla 5.1.

Las principales diferencias en el flujo del código según el funcionamiento se dan en los lugares que se señalan en el diagrama de la figura [5.1] con punteado rojo

Armado de paquetes de datos

Para ambos modos, resulta que junto a la llegada de un dato del clasificador también llegan los datos crudos que se utilizaron para calcularlo. Para el Modo Validación, se arman cuatro paquetes a ser transmitidos. Según sus tópicos (ver sección 3.2 donde se explica el funcionamiento de MQTT) estos son:

- oveja1/info: trae la clasificación, el dato actual del GPS y la hora.

Modo	Descripción	Observaciones
Investigación	Reúne una cierta cantidad de datos clasificados más un arreglo de caracteres para la hora y otro para la posición, arma un paquete y lo envía al Sistema Central.	Mientras más datos se empaqueten, menor la cantidad de transmisiones lo cual impacta en un ahorro energético notorio. No se puede hacer ningún análisis sobre los datos crudos porque no son enviados.
Validación	Con cada Estado que se clasifica (utilizando 500 datos por eje de acelerómetro) se arman cuatro paquetes. Uno con la hora, posición y clasificación y los otros tres cada uno con los datos crudos de un eje del acelerómetro.	Al tener tantos paquetes en simultáneo deben enviarse ni bien se arman aumentando la cadencia de transmisión y consecuentemente el consumo. Permite analizar los datos crudos y efectuar procesamiento ulterior al del Equipo, en una PC.

Tabla 5.1: Modos de funcionamiento del software embebido

- oveja1/ax: Trae los datos de la aceleración según el eje X de la IMU.
- oveja1/ay: Trae los datos de la aceleración según el eje Y de la IMU.
- oveja1/az: Trae los datos de la aceleración según el eje Z de la IMU.

En el Modo Validación una vez que se tiene un Estado clasificado, con la bandera correspondiente se notifica que hay paquetes prontos para ser trasmitidos. Son 4 paquetes. Uno contiene el resultado de la caracterización de Estado junto con la posición GPS obtenida y la hora actual. Los otros 3 paquetes contienen los 500 datos crudos de cada eje del acelerómetro con los cuales se calculó la caracterización de Estado.

Por otra parte, en el Modo Investigación, los datos crudos no son transmitidos y no se transmite cada caracterización individualmente. Se guardan estos resultados de caracterización y se lleva una cuenta. Una vez se obtiene un número definido de resultados, se arma un paquete incluyendo a estos resultados la hora actual y posición GPS. Se notifica con una bandera que se puede enviar el paquete, al tópico de MQTT correspondiente, que en este caso es:

- oveja1/GPSclasifData

Se hace notar que aprovechando el sistema de publicación y suscripción a tópicos el sistema sería rápidamente escalable para varias ovejas ya que su información se identifica según el tópico al que publican (oveja1/ en este caso por ejemplo). Más aun, el Sistema Central puede saber el modo de funcionamiento en el que está cada dispositivo fácilmente ya que los tópicos a los que se publican los datos son diferentes en cada modo.

Envío de paquete de datos por MQTT

En esta sección, la diferencia sustancial es que para el Modo Investigación, al llegar un paquete que debe transmitirse desde el módulo que arma los paquetes, debe encender el módulo LTE IoT 2 click ya que está apagado mientras se procesan Estados y se guardan. Este no es el caso para el Modo Validación donde el módulo permanece prendido siempre. Además, para controlar el flujo de envío de mensajes MQTT, se cuenta la cantidad de paquetes enviados. Lo cual difiere según el modo, siendo 4 paquetes transmitidos en el Modo Validación y solo uno en el de Investigación.

Obtención de dato de GPS

Finalmente, para esta sección, sucede que en el Modo Validación se ejecuta exactamente lo que se muestra en el diagrama sin más. Por otra parte, para el Modo Investigación, en caso de que se haya muestreado un dato del GPS y la posición este fijada, es decir que el GPS devuelve algún dato y no tiene el mensaje vacío, el módulo LTE IoT 2 click se apaga hasta que se deba realizar la siguiente transmisión con las clasificaciones y el dato recién adquirido.

5.5. Módulos del software

El diagrama de módulos del software embebido en el Equipo puede verse en la figura [5.2], a su vez muestra la relación con los periféricos del microcontrolador utilizados y los módulos hardware externos con los que estos periféricos se comunican. Los módulos coloreados en azul refieren al software independiente de hardware, aquellos en verde son módulos dependientes de hardware, los bloques amarillos son periféricos del microcontrolador y por último los rojos representan los módulos hardware externos. Cada módulo en el diagrama hace uso de aquellos con los que se vincula por flechas.

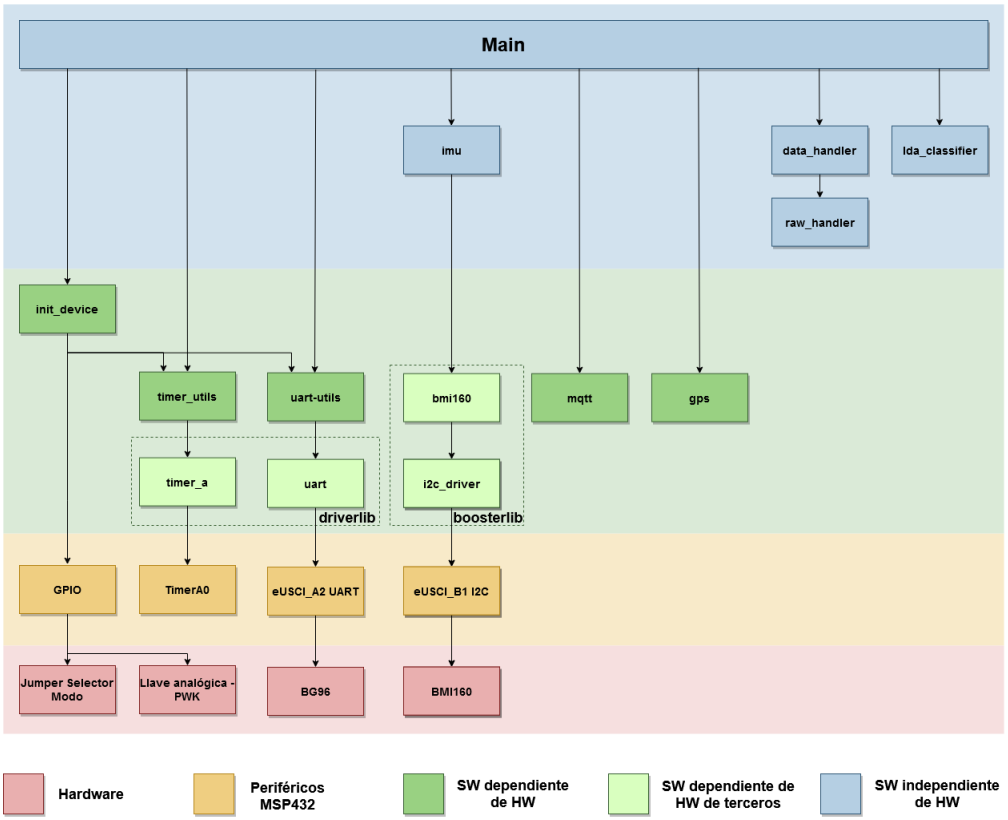

Figura 5.2: Diagrama de módulos del software embebido

Se procede a realizar una breve descripción de cada módulo para posteriormente hacer énfasis en los aspectos de mayor relevancia para el funcionamiento.

5.5.1. init_device

Se encarga de la inicialización e intercambio de mensajes entre el microcontrolador y el BG96 que ponen en funcionamiento el Equipo. Según las capacidades que ofrecen las funciones se categorizan en:

- Flujo de código: funciones encargadas de la asignación de banderas necesarias en la lógica del envío de datos por UART y para gestionar Time-outs.
- Configuración: envío de los mensajes de configuración de la comunicación MQTT, GPS, comando ATE0 para apagar el eco del módulo LTE IoT 2 click y apagado de pines en desuso. Para transmitir estos mensajes de configuración se requiere invocar funciones propias del módulo *uart*.
- Procesamiento: verificación de las respuestas recibidas por UART. Dicha verificación resulta en que el flujo de inicialización continúe normalmente o deban realizarse reenvíos o incluso reiniciar el sistema.
- Operación: se encarga del encendido y correcta apertura de la comunicación MQTT manejando posibles errores ocasionados por rechazos de la conexión o tiempos de espera elevados en el intercambio de mensajes de inicialización con el módem. El manejo de estos tiempos de espera se emplea utilizando funciones del módulo timer_utils.

Se abstrae esta parte de la lógica de los módulos *mqtt* y *gps* que se encargan de la transmisión de datos al Sistema Central en sí y la obtención de datos de GPS respectivamente. Esto se debe a que las funciones que implementa *init_device* solo aplican para configuraciones iniciales.

Mayor especificidad en el flujo de comandos AT se da en la sección 3.3.

5.5.2. timer_utils

El módulo de *timer_utils* tiene el propósito de configurar el periférico del MSP432 "TimerA0" para que interrumpa a una frecuencia de 100 Hz. Contiene además la rutina de interrupción asociada a este timer. El propósito de tener un timer interrumpiendo cada 1 ms es poder utilizar dicha frecuencia para generar banderas para la atención a distintas tareas con frecuencias que sean múltiplos de esa frecuencia.

Según las capacidades de las funciones se tiene:

- Flujo de código: funciones que asignan las banderas necesarias para indicar cuando los distintos contadores terminan.
- Configuración: en particular la función inicializarTimer() la cual fija el reloj SMCLK a 12 MHz con un divisor de 4 generando una frecuencia de 3 MHz. Considerando interrupciones cada 3000 ticks, se obtiene una frecuencia de interrupción de 1 ms.
- Operación: el timer es empleado para llevar cuenta de la frecuencia a la cual muestrear la IMU, el GPS y también para llevar el control de diferentes tiempos de *time-out*.

Como se observa en el diagrama de la figura [5.2], el módulo depende a su vez de la librería *timer_a* la cual se encuentra en el conjunto *driverlib*. En particular las funciones de interés sirven para configurar el reloj SMCLK.

Mayor detalle de la implementación de este módulo puede verse en 5.7.

5.5.3. driverlib

El módulo *driverlib* es un conjunto de módulos desarrollado por Texas Instruments utilizadas para configurar y manipular los periféricos del hardware MSP432. Sin necesidad de modificar ninguno de sus módulos se utilizan los siguientes dos módulos:

- *uart*: provee funciones para configurar la comunicación UART utilizada por el periférico eUSCI. Además, cuenta con funciones para la transmisión y recepción de a un byte. Estas sirven como base para crear funciones que transmitan cadenas de caracteres como se tiene en el módulo *uart_utils*.
- *timer_a*: provee múltiples funciones para trabajar con el timerA. Como más relevantes se tienen Timer_A_configureUpMode(), la cual se encarga de la configuración a aplicar en el modo Up del timerA y Timer_A_startCounter() que sirve para iniciar la cuenta.

5.5.4. uart-utils

Este módulo contiene las funciones que implementan la comunicación UART con la placa Mikroe LTE IoT 2 click incluyendo la rutina de atención a interrupciones. Se implementa con una única UART la comunicación con la placa Mikroe, la cual implica comunicación con el GPS y con el módulo de comunicación Narrowband IoT. Categorizando las funciones según las capacidades que ofrecen se tiene:

- Flujo del código: el módulo provee funciones para el manejo de la bandera que indicará si hay un mensaje nuevo recibido en el bucle principal y funciones para el manejo de buffers empleados en la comunicación.
- Configuración: la función de inicialización define los parámetros de la comunicación y configuración del periférico. Se efectúa valiéndose de funciones de la librería *uart* que también pertenece al conjunto *driverlib*. En dicha configuración se establece que la comunicación será UART: 8N1 sin control de flujo y con una velocidad de comunicación de 115200 baudios.
- Operación: la implementación elegida utiliza como carácter de fin de línea tanto en transmisión como recepción el "\r". Esto implica en transmisión, la cual es iniciada con la función transmitir_cadena(), que la interrupción encargada de mandar los caracteres del mensaje solo dejará de interrumpir al leer como último carácter aquel de fin de línea. Por otra parte, en recepción significa que se levantará una bandera para copiar a un buffer auxiliar y dejar pronto el buffer de recepción para recibir nuevos datos.

La implementación elegida trae como beneficio que los mensajes pueden ser de largo variable en tanto terminen con el carácter elegido. Por otra parte trae como desventaja que ese carácter no puede estar en la carga útil del mensaje (a menos que se lo sustituya y se indique dicha sustitución, ver 5.5.10).

5.5.5. boosterlib

El *boosterlib* es un conjunto de librerías también desarrollado por Texas Instruments, en el que implementan un variedad de funciones para utilizar con el BOOSTXL SENSORS. Sin necesidad de modificar ninguno de sus módulos se utilizan los siguientes dos módulos:

- *bmi160*: provee funciones para trabajar con el sensor BMI160. Como ejemplo se tiene bmi160_config_running_mode() que se encarga de configurar el modo de funcionamiento. También se destaca bmi160_read_accel_xyz() que se ocupa de retornar los datos del acelerómetro de los tres ejes en un tipo de estructura también definida en el módulo. Este módulo está basado en otro del mismo nombre provisto por Bosch pero incorporando funciones más específicas por ejemplo para la comunicación I2C.

- *i2c_driver*: provee funciones para configurar la comunicación I2C, recibir y transmitir mensajes entre el microcontrolador y el BoosterPack. Sus funciones son implementadas en el módulo *bmi160* para definir como escribir y leer en I2C. Estas funciones son readI2C() y writeI2C().

5.5.6. imu

El conjunto de funciones presentes en *imu* implementan la inicialización del periférico junto con el manejo de datos del acelerómetro. Este módulo utiliza la librería provista por Texas Instruments, *boosterlib*.

Agrupando las funciones por categoría se tiene:

- Flujo de código: asignan las banderas con las cuales se manejarán el muestreo de datos y se indicará si se completó el buffer de muestras para que sean clasificadas.

- Configuración: inicializan la configuración de la comunicación I2C utilizando la función initI2C() que se encuentra en el módulo *i2c_driver* además de las configuraciones correspondientes al modo de funcionamiento para el sensor BMI160. Para configurar el BMI160 se utilizan las funciones bmi160_initialize_sensor() y bmi160_config_running_mode() que se pueden encontrar en el módulo *bmi160*. La configuración del BMI160 es en este caso STANDARD_UI_IMU, la cual corresponde a un muestreo de 100 Hz.

- Operación: para el manejo de datos del acelerómetro se utiliza la función bmi160_read_accel_xyz() del módulo *bmi160*, la cual se encarga de obtener un dato nuevo. A su vez mediante un manejo de banderas el módulo *imu* es capaz de avisar cuando hay una ventana de 500 dato nueva para ser utilizada en la clasificación.

En otras posibles configuraciones la frecuencia de muestreo puede variar, incluir configuraciones para un magnetómetro que funcione en conjunto e incluso se tiene a disposición un stack FIFO de 1024 bytes de datos propio del sensor.

5.5.7. mqtt

Este módulo implementa todas las funcionalidades referentes al manejo de la transmisión de datos con el protocolo MQTT sobre la red Narrowband IoT. Se categorizan las funciones según las capacidades que ofrecen:

- Flujo del código: funciones encargadas de la asignación de banderas necesarias en la lógica del bucle principal.
- Configuración: Las funciones de inicialización que determinan los parámetros a establecer para la comunicación, apertura y establecimiento de la conexión no se encuentran en este módulo sino que están en el módulo *device_init*.
- Procesamiento: Contiene la función que interpreta las cadenas de caracteres recibidas por MQTT. En base a las respuestas se maneja de manera interna al módulo si debe proseguirse enviando el siguiente comando del flujo, debe efectuarse un reenvío o levantarse una bandera que indique que el Equipo debe reiniciarse.
- Operación: Contiene las funciones encargadas de acondicionar los paquetes de transmisión así como también verificar el flujo de intercambio de comandos. Para transmitir información, primero se abre el tópico generado a nivel de protocolo MQTT y la información se envía una vez recibida la señal de habilitación del módulo Quectel.

Mayor detalle en el flujo necesario para el intercambio de datos con la red puede consultarse en la sección 3.3.

5.5.8. gps

Este módulo implementa todas las funcionalidades referentes al manejo del GPS. El funcionamiento se basa en la interpretación de sentencias NMEA que devuelve el GPS para obtener la información de interés. En particular, aquella de interés y de las más comunes es la GGA. Esta contiene información sobre posición, elevación, hora y número de satélites utilizados.

Se categorizan las funciones según las capacidades que ofrecen:

- Flujo del código: funciones encargadas de la asignación de banderas necesarias en la lógica del bucle principal como lo es “flagTmrGPS”.
- Configuración: la función de inicialización pertenece al módulo *device_init*.
- Procesamiento: funciones que reciben las sentencias de respuesta del GPS en forma de cadenas de caracteres y las interpretan, obteniendo la información pertinente. Como datos de mayor interés se tienen la hora y la posición. También potencialmente se tienen la cantidad de satélites fijados para la triangulación de la posición.

- Operación: el módulo cuenta con funciones que definen la secuencia de comandos AT a emplear en la comunicación UART. Dada la aplicación, el comando AT a enviar es solo uno el cual sirve para pedirle al GPS la sentencia NMEA GGA para su posterior análisis.

  Se desarrolla sobre el flujo de comandos AT a intercambiar con el módem en 3.3.

5.5.9. data_handler

El módulo se encarga del armado de paquetes de datos según el modo de funcionamiento. Sus funciones según las capacidades que ofrecen pueden clasificarse de la siguiente manera:

- Flujo de código: funciones para el manejo de banderas que se utilizan en el bucle principal asociadas al armado de paquetes de datos a enviar al Sistema Central empleando MQTT.
- Procesamiento: para el armado de los paquetes que contengan los datos crudos del acelerómetro se utiliza el módulo *raw_handler* según se indica en la siguiente subsección.
- Operación: el módulo contiene las funciones encargadas de armar los paquetes de datos a enviar identificando en qué modo está operando el sistema (Modo Validación o Modo Investigación) y en base a eso la lógica que se implementa para generar los paquetes de datos a enviar.

5.5.10. raw_handler

El módulo *raw_handler* fue creado con el fin de codificar los datos crudos para no generar errores relacionados a caracteres especiales utilizados en el código y en la comunicación UART con el módem. Esto es debido a que los datos provenientes del acelerómetro son números binarios sin codificar que, al ser enviados como la carga útil del mensaje MQTT, son interpretados como su equivalente ASCII y puede generar problemas en la trasmisión. Consecuentemente, al haber caracteres que no pueden utilizarse debe idearse un método de escape para sustituirlos y decodificarlos en recepción. Sus únicas 2 funciones se clasifican según su funcionalidad en:

- Procesamiento: función que recibe los datos crudos del acelerómetro y arma el paquete con los datos codificados
- Operación: función encargada de que codificar un dato crudo del acelerómetro

Se identificaron 5 caracteres que no se pueden enviar. Estos 5 caracteres son:

- NULL (0x00): Ya que el código se vale de funciones que requieren este carácter de fin de línea, por ejemplo, al concatenar cadenas de caracteres.

- Retroceso (0x08): El módem recopila los caracteres del mensaje que se quiere enviar hasta que se le envía un carácter de fin de linea y en ese momento hace efectiva la transmisión. Si se le envía un retroceso, el carácter inmediatamente anterior es borrado.
- Retorno de carro (0x13): Es el carácter de fin de linea elegido para transmitir por UART.
- SUB (0x26): Es el carácter de fin de linea que tiene el módem para saber que debe mandar la cadena de caracteres que recopiló hasta el momento.
- ESC (0X27): Es el carácter para cancelar el envío de caracteres por UART al módem y que el mensaje no sea enviado.

Para solucionar el problema se decidió implementar una codificación de datos tal que en caso de necesitar enviar alguno de los caracteres ya mencionados se sustituye por un 0x30, correspondiente al 0 en ASCII (podría haber sido cualquier otro carácter siempre que no fuera de los reservados). Se agrega un byte de control encargado de avisar si el carácter fue modificado por un 0x30 o el carácter recibido es auténtico. Debido a que cada dato ocupa 2 bytes se utilizó el mismo byte de control para identificar si alguno de los 2 bytes de datos fue alterado. Se dividió el byte de control en 2. Los 4 bits menos significativos del byte de control son los encargados de avisar el estado del byte de datos menos significativo y los 4 bits más significativos del byte de control son los responsables del byte de datos más significativos. Si el byte a transmitir corresponde con alguno de los caracteres, los 4 bits que le corresponden en el byte de control se modifican con el código de la tabla 5.2. La tabla 5.3, presenta algunos ejemplos de la codificación utilizada.

Carácter (hexadecimal)	**Código en binario**
NULL (0x00)	0001
Retorno de carro (0x13)	0010
SUB (0x26)	0011
ESC (0x27)	0100
Retroceso (0x08)	0101
Ninguno de los anteriores	1111

Tabla 5.2: Código de codificación

Por lo tanto, el módulo *raw_handler* contiene la función que codifica el dato crudo del acelerómetro y la función encargada de encolar los datos crudos al paquete de datos a enviar del módulo de *data_handler*.

5.5.11. lda_classifier

Este módulo implemente todas las funciones relativas al algoritmo de aprendizaje automático que se utiliza para clasificar un Estado según los datos crudos del acelerómetro. Las funciones se categorizan según las funcionalidades que ofrecen:

Dato a enviar			Dato codificado (dato enviado)		
Decimal	Primer byte (hex)	Segundo byte (hex)	Control byte (hex)	Primer byte (hex)	Segundo byte (hex)
4904	0x13	0x26	0x23	0x30	0x30
10259	0x28	0x13	0xF2	0x28	0x30
17	0x00	0x11	0x1F	0x30	0x11
6168	0x18	0x18	0xFF	0x18	0x18

Tabla 5.3: Ejemplos de datos codificados

- Flujo de código: funciones encargadas de la asignación de banderas necesarias en la lógica del bucle principal
- Configuración: función que permite actualizar los parámetros de la matriz de clasificación en caso de ser necesario
- Procesamiento: función que multiplica matrices por vectores
- Operación: funciones encargadas de clasificar el Estado del animal a partir de una ventana de datos crudos del acelerómetro calculando el vector de atributos primero y multiplicándolo por la matriz clasificadora después. La ventana de datos es de 500 muestras por eje, es decir, 1500 números enteros. Dicha ventana es elegida ya que es la que se emplea para entrenar el algoritmo de ML. A su vez mediante un manejo de banderas el módulo es capaz de avisar cuando hay un dato nuevo clasificado para que se añada al paquete a enviar armado en el módulo *data_handler*

Cabe destacar que el módulo ya tiene cargado como variables las matrices y vectores que se obtiene del aprendizaje automático que se necesitan para realizar la clasificación del Estado.

Debe mencionarse además, que el BMI160 trabaja con 16 bits, así devolviendo sus datos en el rango de [-32768, 32768]. Por lo tanto, los datos crudos provenientes del sensor se encuentran dentro de ese rango. Luego, el acelerómetro trabaja dentro del rango [-2g, 2g], dos veces la aceleración gravitatoria. Para escalar los datos crudos al rango correspondiente este módulo se encarga de efectuar la siguiente ecuación:

$$a_x = a_{xcrudo} \cdot \frac{2}{32768} = \frac{a_{xcrudo}}{16384} \tag{5.1}$$

Donde se define a_{xcrudo} como el dato que se obtiene del BMI160 sin ningún tipo de procesamiento y a_x como el dato mapeado al rango correspondiente.

Parece coherente efectuar esta operativa en el módulo de clasificación ya que del módulo *imu* es necesario que los datos que salgan sean los crudos para enviarlos como números enteros.

El algoritmo 1 calcula el vector de atributos de una ventana de datos crudos del acelerómetro. El algoritmo 2 realiza la clasificación y determina el Estado

del animal a partir del vector de atributos. Se desarrolla el funcionamiento del algoritmo de ML en mayor profundidad en el capítulo 4.

Algoritmo 1: Calcula el vector de atributos

Input: ax, ay, az
Output: Vector atributos
Result: Recibe los datos crudos del acelerómetro y calcula el vector de atributos

```
// Convierte datos crudos en rango correspondiente
```
for $i \leftarrow 0$ ***to*** ax_{size} **do**
  $ax_i \leftarrow ax_i/16384 \times 9{,}8$
  $ay_i \leftarrow ay_i/16384 \times 9{,}8$
  $az_i \leftarrow az_i/16384 \times 9{,}8$
end
$ax_{min} \leftarrow ax_0$; `// Busca mínimo en ax`
for $i \leftarrow 1$ ***to*** ax_{size} **do**
  if $ax_i < ax_{min}$ **then**
    $ax_{min} \leftarrow ax_i$
  end
end
$ay_{min} \leftarrow ay_0$; `// Busca mínimo en ay`
for $i \leftarrow 1$ ***to*** ay_{size} **do**
  if $ay_i < ay_{min}$ **then**
    $ay_{min} \leftarrow ay_i$
  end
end
$az_{mean} \leftarrow az_0$; `// Calcula media de az`
for $i \leftarrow 1$ ***to*** az_{size} **do**
  $az_{mean} \leftarrow az_{mean} + az_i$
end
$az_{mean} \leftarrow az_{mean}/az_{size}$; `// Calcula varianza de az`
for $i \leftarrow 0$ ***to*** az_{size} **do**
  $az_{var} \leftarrow az_{var} = (az_i - az_{mean}) \times (az_i - az_{mean})$
end
$az_{var} \leftarrow az_{var}/az_{size}$
$az_{std} \leftarrow \sqrt{az_{var}}$; `// Calcula desviación en az`
return $[az_{std}, ax_{min}, ay_{min}, az_{var}]$

Algoritmo 2: Clasifica el Estado

Input: vec (Vector atributos)
Output: estado
Result: Recibe el vector de atributos y clasifica el Estado. El vector vec2std, mean y intercept junto con la matriz mtx son datos ya cargados obtenidos del ML

```
// Estandarizo el vector de atributos
for i ← 0 to vec_size do
    std_vec_i ← (vec_i − mean_i)/vec2std_i
end
// Multiplica matriz por vector estandarizado
rslt_mtx_vec ← mtx × std_vec
// Me quedo con el Estado de mayor probabilidad
estado ← 0
prob_estado ← rslt_mtx_vec_0 + intercept_0
for i ← 1 to cant_estados do
    if rslt_mtx_vec_i + intercept_i > prob_estado then
        estado ← i
    end
end
return estado
```

5.6. Implementación comandos AT en el software embebido según el modo de funcionamiento

En base a las secuencias de comandos desarrollada en 3.3, se explica en esta sección la forma en que se emplea el flujo de comandos a nivel de código para cumplir con las tareas de muestreo GPS y transmisión de datos según el modo de funcionamiento. En la figura [5.3] se representa el funcionamiento del software embebido solo a nivel de comandos AT, diferenciando según el módulo encargado de realizar dicha secuencia.

Se separa el primer bloque de comandos para diferenciar los comandos de inicialización que no corresponden directamente a transmitir por MQTT ni obtener datos por GPS. Cabe destacar que en nuestra implementación nunca se necesita apagar el GPS o desconectarse de la red MQTT. En el Modo Validación se reitera cada función indefinidamente, entonces no se apaga ni desconecta nunca de forma voluntaria; solo se apaga en casos de detección de un error que se soluciona con un reinicio. En el Modo Investigación simplemente se apaga todo el módulo hardware una vez obtiene posición del GPS, lo cual automáticamente desconecta de la red MQTT. La próxima vez que se tenga suficientes datos para transmitir, se repetirá todo el flujo.

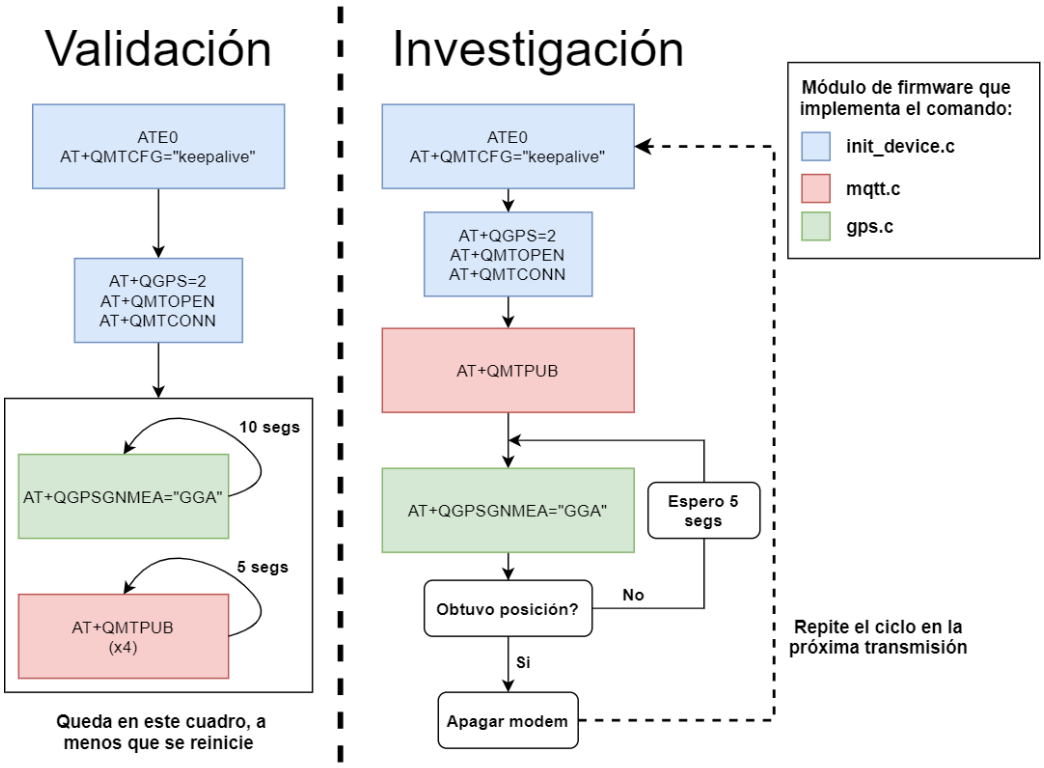

Figura 5.3: Flujo de comandos AT empleado en el software embebido

5.7. Contadores y Timers

En esta sección se describe la implementación del timer y contadores asociados para generar períodos de muestreo y otras funciones del código. Para esto se emplea el TimerA0 del microcontrolador, configurado en *Up Mode* de modo que genere una interrupción periódica cada 1 ms.

La interrupción del timer simplemente incrementa un contador asignado a cada función que se quiera tener temporizada. Cuando este contador alcanza un límite configurado, se prende una bandera para indicar al bucle principal que transcurrió esa cantidad de tiempo y se debe realizar cierta tarea. El timer interrumpe cada 1 ms e incrementa la cuenta en 1. Por ende, el valor límite del contador es el valor en milisegundos que tendrá el temporizador.

El límite de estos contadores es fácilmente configurable dado que se requiere cambiar un único valor. Además resulta intuitivo que se incremente cada 1 ms para fácilmente definir cualquier tiempo que se requiera. Sin embargo, en vista de que 10 es el mínimo límite de cuenta empleado, se podría reconfigurar el timer para que interrumpa cada 10 ms. Esto potencialmente ahorra consumo del microcontrolador, pero no fue implementado y el efecto sería despreciable en vista de los consumos relevados y presentados en la sección 7.2.

Excepto el muestreo del acelerómetro, los demás contadores no están permanentemente activos. Cada uno puede ser habilitado o deshabilitado según se re-

quiera su función en el flujo del código. El timeout y delay se implementan en forma de función que permite definir el límite de cuenta cuando es invocada de forma de poder generalizar su uso para distintas funcionalidades.

void start_timeout(**int** t_timeout_ms)

void delay_ms(**int** time_ms)

Las funciones de start_timeout y delay_ms se configuran de forma tal que habilitan al TimerA0 de prender las banderas correspondientes en caso de haber trascurrido el tiempo asociado.

En la tabla 5.4 se presentan todos los contadores que se emplean para el funcionamiento del código. Se utilizan distintos tiempos de Delay a lo largo de todo el código con distintas funcionalidades. El resto de los contadores se mantiene constante en todo momento diferenciado la frecuencia de muestreo del GPS según el modo de funcionamiento.

Función del contador	**Límite de cuenta**
Muestrear acelerómetro	10
Muestrear GPS	5000 (Investigación) 10000 (Validación)
Timeout	10000
Delay	Valores entre: 25 y 1700

Tabla 5.4: Contadores que incrementa el timer, según su función y límite de cuenta que realizan Cada cuenta equivale a 1 ms

5.7.1. Implementación de un Timeout para control de errores

Para gestionar errores de comunicación entre microcontrolador, el GPS y el módem se implementa un Timeout. Esto es un temporizador que, si finaliza su cuenta, indica que hubo un error. Se inicia con una función en cierto punto del código y se frena en otro punto. Si no se frena antes que termine la cuenta, hay una falla en el flujo y se necesita tomar una acción en respuesta.

Lo que se busca solucionar son los casos en que se espera una respuesta del módem que nunca llega, sea en el flujo usual de transmisión de datos o en la inicialización y conexión a la red. Por esto se empieza y frena el timeout en las tareas del bucle principal relacionadas al módulo, y en la rutina de inicialización del mismo.

En la implementación actual del código, el timeout levanta la bandera de reinicio del módem. No se lograron identificar casos de timeout que sean confiablemente solucionables sin emplear un reinicio. Es una solución fuerte pero segura. Luego, estos eventos no suceden con frecuencia suficientemente elevada como para que el reinicio comprometa el funcionamiento del Equipo. Eventualmente previo a un reinicio, se podría implementar una lógica de reintentos y que luego de cierta cantidad de reintentos fallidos, se reinicie el módem.

5.7.2. Implementación del Watchdog Timer (WDT)

El Watchdog Timer, o "temporizador de perro guardián" es un periférico del microcontrolador, tiene como propósito reiniciar el microcontrolador en caso que este se quede trancado en algún punto del código de manera no prevista. Funciona como un timeout igual que el explicado previamente. Cuando expira este temporizador, genera una señal de reinicio para el microcontrolador. Esta funcionalidad es esencial en cualquier equipo que funcione de forma autónoma por largos períodos de tiempo, dado que permite sacar al código de cualquier tipo de tranca o problema imprevisto.

Se controla este módulo configurando registros del microcontrolador. Se implementa en el bucle principal del código, donde en cada vuelta se reinicia la cuenta de este temporizador. Que el temporizador finalice quiere decir que la cuenta no fue reiniciada a tiempo, por lo que el código está trancado en alguna tarea y no retorna al bucle principal. Ante este problema la única solución es reiniciar el microcontrolador.

El WDT se configura para funcionar a partir del SMCLK del microcontrolador, el cual es un sub-reloj que opera a 1/4 de velocidad que la frecuencia del CPU. El CPU funciona a 48 MHz, por lo tanto el SMCLK a 12 MHz. Por otro lado se debe configurar el valor máximo de la cuenta del WDT, es decir el valor al que debe llegar para reiniciar el microcontrolador. Se tiene un número limitado de opciones configurables, de las cuales se elige 128 millones. De estos parámetros de obtiene el tiempo en segundos que dura el temporizador.

Frecuencia SMCLK	Cuenta WDT	Tiempo del WDT
12 MHz	128 M	10.67 segundos

Tabla 5.5: Duración del WDT en base a los parámetros configurados

Si cuenta 12 millones de veces por segundo, tarda 10.67 segundos en llegar a 128 millones. Esta duración probó ser útil en la práctica y por eso se decidió usarla.

Es preciso remarcar que el Watchdog Timer y el Timeout de implementación propia solucionan problemas distintos. El primero actúa en cualquier situación que detenga el código de forma indeseada. Mientras que el segundo actúa cuando el código sigue corriendo pero no sucede un evento esperado.

5.8. Problemas y mejoras

Modularización

Se logró una buena implementación del software embebido, logrando un buen nivel de modularización, siendo algunos de estos completamente independientes del hardware. Sin embargo todavía quedan algunos módulos que requieren un trabajo mas profundo para lograr tal independización. Esto implica generar capas de abstracción que permitan invocar funciones de los módulos a nivel más alto, separado del hardware.

Tareas bloqueantes

Uno de los problemas que presenta el código, el cual deberá analizarse para futuros trabajos, es trabajar con tareas bloqueantes. La lógica de semáforo implementada genera que haya ciertas tareas que bloquean el código. Esas tareas son aquellas que utilizan la UART. El código está implementado de forma de actuar según la respuesta a los comandos AT que le llegan del módem. Se priorizó que el código quede esperando la respuesta correspondiente y que este no pueda ser interrumpido por otras tareas, ya que esto puede generar pérdida de datos en la recepción. En el Modo Validación, esto no presenta ningún inconveniente dado que los tiempos no interfieren con el tiempo de muestreo de las demás tareas. El problema se presenta en el Modo Investigación, donde se apaga el módem y por lo tanto cada vez que se prende se requiere enviar los comandos **AT+QMTOPEN** y **AT+QMTCONN** que su respuesta sí presenta tiempos considerables. Para trabajos futuros se sugiere que el flujo del código no dependa enteramente de la respuesta recibida, en la aplicación aquí planteada el código compara la cadena de caracteres de la respuesta recibida con la cadena de la respuesta esperada y para esto es fundamental recibir el mensaje en su completitud sin ninguna alteración. Sin embargo, se podría implementar algún método donde la pérdida de algún byte en la respuesta no genere inconveniente y el código pueda seguir con su flujo normal de trabajo.

Método de escape

En referencia al método de escape que se implementó en el módulo *raw_handler* explicado en la sección 5.5.10 se identificó un potencial punto débil. Se analizó y se estudió una posible alternativa a implementar en sustitución al método actualmente implementado.

La solución implementada descrita anteriormente consiste en agregar a todos los datos (de 2 bytes) un byte de control que avisa si los siguientes caracteres se encuentran alterados o no. Esto resulta en un aumento de la carga del paquete a enviar en Modo Investigación en un 50 %. Dado el alcance del presente proyecto esto resulta aceptable, sin embargo para una reducción en el consumo ulterior debe de replantearse.

El siguiente método que se pensó como alternativa para implementar a futuro consiste en enviar un byte de control solo en los casos que el caracter haya sido modificado. Reduciendo la cantidad de byte de control enviados sólo a los casos problemáticos.

Teniendo en cuenta de antemano cuales son esos caracteres y sabiendo que hay dos caracteres continuos que no se pueden enviar, se pensó trabajar con una lógica donde los datos van a ser redondeados con una diferencia de dos unidades. La lógica consiste en utilizar 0xFF como byte de control y redondear los caracteres a 2 valores por arriba si corresponden a la parte inferior del rango dinámico o a 2 valores por debajo si corresponde a la parte superior del rango dinámico. Entonces cada vez que se tenga que redondear un caracter se envía 0xFF a continuación del byte redondeado. Si el byte no contiene ningún 0xFF antes significa que el caracter

no fue modificado y su valor es el mismo, si posee un 0xFF significa que su valor fue alterado y habrá que restar o sumar 2 dependiendo del rango al que pertenece.

Este método tiene la contrapartida de que se agrega un problema donde no lo había, ya que ahora el 0xFF pasa a ser considerado como un byte de control. Se genera un byte más al que se le deberá aplicar el método de escape. Utilizando la misma lógica el byte 0xFF se deberá redondear 2 unidades hacia abajo y se le agregará el byte de control cada vez que se necesite.

En comparación con el método que está implementado actualmente, la carga de datos a enviar que se agrega al incorporar el 0xFF como nuevo caracter a escapar es despreciable frente al byte de control que se le agrega a cada dato. El aumento de carga útil del nuevo método queda dependiente de la probabilidad de que se tenga que enviar alguno de esos caracteres, si su probabilidad es baja la carga que se aumenta es baja. En el método actual la carga siempre se aumenta en la misma proporción que los datos que se manden.

Incompatibilidad PSM con MQTT

El módulo BG96 desactiva los contextos sobre los cuales funciona MQTT cuando entra en modo PSM o se apaga, por lo cual es necesario buscar una manera de evitarlo y así disminuir tiempos de conexión y consecuentemente, el consumo. Se proponen entonces dos caminos para subsanarlo.

La primera opción es configurar el tiempo de keepalive del protocolo TCP si la red así lo permite. De esta manera, aún si el intercambio de datos no se da durante cierto tiempo, el contexto se mantendrá encendido y la red MQTT podrá mantenerse. Según la hoja de datos del BG96 el comando para efectuar dicha configuración fue introducido en una revisión efectuada en el 2020. Sin embargo, el sistema actual no cuenta con ella.

La segunda opción consta en analizar el protocolo MQTT-SN (MQTT for sensor networks). A diferencia del protocolo MQTT, donde debe establecerse una conexión de manera tal que se tengan ciertas garantías sobre la transmisión y recepción de los paquetes, este otro permite enviar mensajes a un tópico sin previa conexión pero con la posibilidad de que este se pierda. Entonces resulta de interés investigar el uso de este protocolo con el BG96 y el compromiso entre redundancia en el envío de datos para que no se pierdan y la tasa de pérdida de paquetes.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 6

Diseño mecánico

El Equipo fue concebido con el objetivo de ir sujeto al animal en todo momento durante la adquisición de datos. Por este motivo, fue necesario tener cuidado en aspectos tales como el tamaño, peso y método de sujeción para no dañar al animal ni alterar su comportamiento.

En este capítulo se detallan las principales características del modelado tridimensional del encapsulado del Equipo, los materiales y el método de sujeción. Su representación tridimensional puede apreciarse en la Figura [6.1]

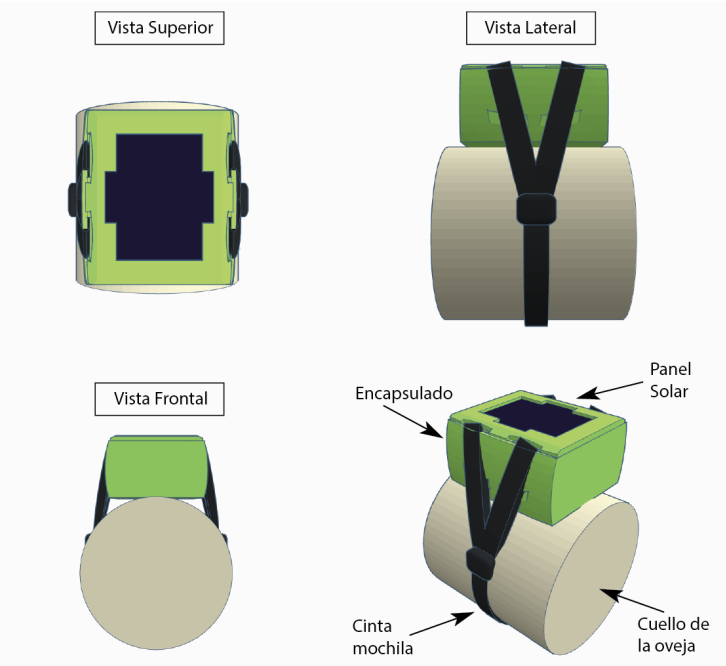

Figura 6.1: Collar en 3D diseñado en Tinkercad

6.1. Diseño del encapsulado

Para el diseño de las piezas en 3D se trabajó con el software de diseño Tinkercad de Autocad. Se escogió esta plataforma dado el buen compromiso entre sencillez de uso y versatilidad para diseños variados. A su vez, Tinkercad es de uso en línea, por lo cual no requiere la instalación de programas que potencialmente tengan altos requerimientos.

Para comenzar con el diseño, se tuvo en cuenta por una parte el espacio ocupado por la electrónica y por otra las dimensiones del panel solar rígido en su parte superior (ver secciones 2.4 y 2.3.4). El encapsulado (ver figura [6.2]) se diseñó como una caja con una tapa removible que encastrase en la canaleta del cuerpo principal (ver figura [6.3]). La forma de las solapas para que la tapa encastre tienen forma trapezoidal, en particular para que el encastre sea sencillo. Resulta que la impresión 3D deja asperezas que a priori podrían dificultar el acople entre tapa y parte inferior especialmente cerca de los vértices. Por otra parte, la tapa posee un orificio de forma tal que el panel solar que irá debajo pueda recibir luz (ver figura [6.2.D]).

Luego, para la sujeción del encapsulado, se previeron cuatro empuñaduras por donde pasar cintas (ver figura [6.2.B] o [6.4]), dos de cada lado del encapsulado. Se decidió finalmente incorporar orificios para tornillos tal que fije la tapa al cuerpo principal y otro para el interruptor del Equipo (ver figura [6.4]). Estos orificios serían efectuados luego de impresa la caja para poder presentar la disposición de la electrónica y disminuir los imprevistos.

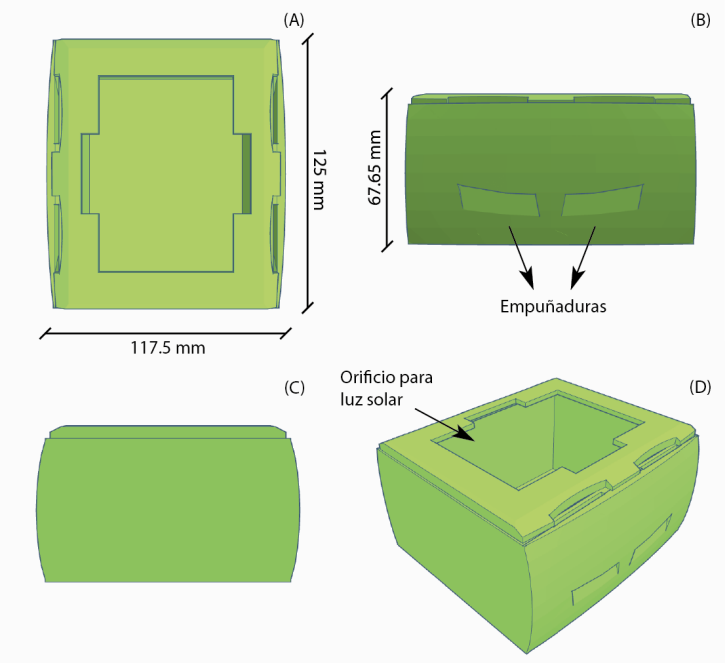

Figura 6.2: Diseño 3D del encapsulado

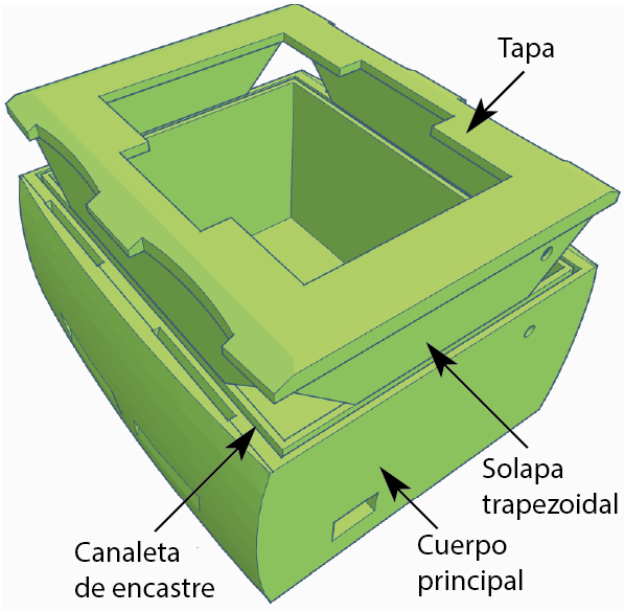

Figura 6.3: Encapsulado y representación de la canaleta con la tapa

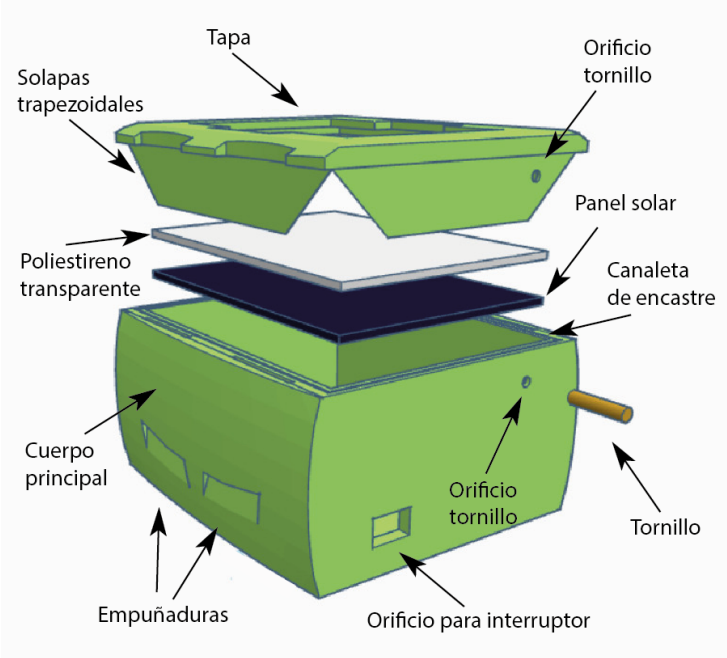

Figura 6.4: Representación del encastre entre las partes

6.1.1. Fabricación y ensamble del encapsulado

Para la fabricación del encapsulado se decidió utilizar los servicios de impresión 3D de la empresa IALarquitectura. El material elegido para la impresión fue

PLA+, una variante del tradicional PLA (ácido poliláctico) con mayor resistencia y flexibilidad. Además, resulta ser suficientemente resistente, económico y de naturaleza biocompatible (hecho a base de recursos renovables como el almidón de maíz o la caña de azúcar). Una vez con las piezas, se procedió a realizar los orificios necesarios para los tornillos y el interruptor.

Además, en vista de la fragilidad del panel, se incorporó una lámina de poliestireno de 1 mm transparente para evitar su exposición a la intemperie y aumentar la robustez del encapsulado. Las junturas de la lámina así como las del interruptor fueron luego selladas con silicona fría.

Véase en la Figura [6.4] esquemáticamente la disposición y encastre de los componentes.

Las dimensiones finales del encapsulado son de 125.0 mm × 117.5 mm × 67.65 mm y su peso con la electrónica dentro es de 465 g. Ver fotografía del Equipo en la Figura [6.5]).

Figura 6.5: Fotografía del encapsulado

6.2. Fijación del encapsulado al animal

En lo que respecta a la sujeción del encapsulado al animal se decidió fabricar con cinta mochila al ser lo suficientemente flexible para ajustarse al cuello del animal, una hebilla de plástico y reguladores de cinta como se puede apreciar en la figura [6.6]. Estos materiales no dañan al animal y tras las pruebas de campo realizadas tampoco parecieran alterar su comportamiento. El ajuste puede variar entre 25 cm y 40 cm de largo.

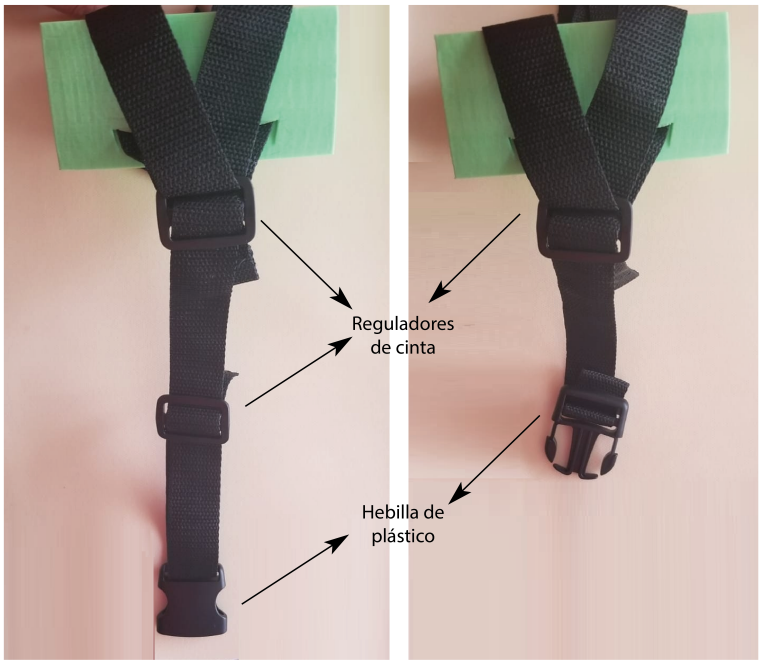

Figura 6.6: Fotografía con ambos extremos de la cinta para fijar el Equipo.

El prototipo final en campo puede apreciarse en las fotografías de la Figura [6.7]

Figura 6.7: Fotografía del prototipo final en ejemplar ovino ubicado en Facultad de Veterinaria.

Capítulo 7

Pruebas experimentales y análisis de resultados

El presente capítulo reúne las pruebas relacionadas al consumo de energía del Equipo y a la efectividad de los algoritmos empleados.

Se presentan los resultados y evaluaciones de las pruebas de campo realizadas con el Sistema Completo en ovejas.

Por otro lado, se busca estudiar cómo impactan en el consumo del Equipo las diferentes tecnologías empleadas y componentes de hardware. Para esto se reportan medidas de consumo y se evalúa según modos de operación y parámetros configurables de las tecnologías. Estas son la comunicación por Narrowband IoT sobre protocolo MQTT y posicionamiento GPS. A nivel de módulos hardware se evalúa el consumo del módulo LTE IoT 2 click y el sistema de alimentación.

7.1. Pruebas de campo

Resultados globales:

Las pruebas de campo se realizaron el día 3 de junio del 2021, en el predio principal de la Facultad de Veterinaria en Montevideo. Las pruebas comenzaron a las 8:00 AM y se terminaron aproximadamente a las 11:00 AM del mismo día. Se contó con la asistencia de el cliente Rodolfo Ungerfeld y un funcionario de la Facultad de Veterinaria que asistió con la colocación del collar y el manejo de las ovejas.

La prueba consistió en colocarle el collar a una de las ovejas y comparar el comportamiento de la oveja registrado visualmente con el resultado obtenido a través del procesamiento de datos en el Equipo, reportado en tiempo real al Sistema Central. Durante la prueba se determinaron los 4 Estados que el algoritmo es capaz de identificar. Estos Estados se van a identificar como quieta, con la cabeza agachada, caminando y corriendo. Se desea evaluar si la clasificación de los Estados es acorde a la actividad real de la oveja. Para esto se llevó un registro visual a través de una grabación de video como también un registro escrito de lo observado.

Durante la prueba, además de registrar los Estados de la oveja, se fueron almacenando en una base de datos los datos provenientes de los ejes de los acelerómetros y las coordenadas geográficas del GPS junto con la hora.

Las pruebas en animales fueron realizadas y supervisadas por el cliente y personal de Facultad de Veterinaria. Entre las 8:00 AM y las 9:00 AM se les ofreció alimento (forraje), y durante ese período los animales se dedicaron esencialmente a comer, lo cual implica que caminen poco, se queden quietas y agachen la cabeza. Entre las 9:35 AM y las 9:45 AM y entre las 10:05 AM y 10:15 AM se intervino en la majada para que los animales caminen y corran. En el resto del tiempo no se intervino y se dieron las interacciones habituales.

En la figura [7.1] se ven gráficas de los Estados clasificados en función del tiempo.

- caminando
- quieta (asimilable a parada)
- cabeza agachada (asimilable a pastando)
- corriendo

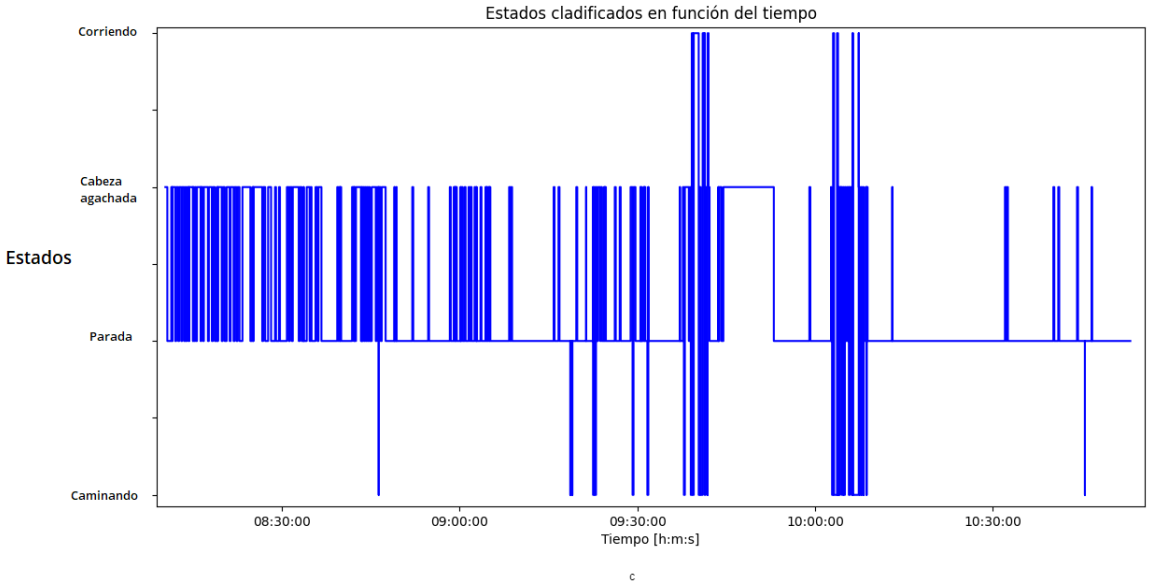

Figura 7.1: Estados clasificados en función del tiempo

A simple vista los Estados parecen ser los esperados. Se puede observar como los Estados que predominan son el 1 y el 2 correspondientes a estar quieta y agachar la cabeza y es acorde a lo esperado ya que el animal estuvo mayormente en esos Estados. La primera hora corresponde a la hora donde se alimenta el animal por lo tanto es cuando más veces se puede ver que agacha la cabeza. La secuencia de

cambios de Estados que se presentan en los minutos 9:40 y 10:03 pertenecen al momento en el que se intervino para que corra y camine así pudiendo evaluar el algoritmos en su completitud.

Hay 3 momentos para destacar que no corresponden con ningún Estado y son los siguientes. En el minutos 8:37 a 8:38 por problemas con la computadora no se registraron datos. En el minuto 9:22 a 9:23 se separó a la oveja y se le ajustó el collar y por último en los minutos 9:45 a 9:53 por problema de conexión de red tampoco se pudieron guardar datos.

Se analiza en detalle algunos intervalos específicos de tiempo para hacer un análisis más profundo. Se compara en detalle el Estado clasificado por el algoritmo y el Estado observado visualmente en el video que se grabó durante la prueba.

7.1.1. Oveja comiendo

En la gráfica de la figura [7.2] se representa en verde los Estados observados en el video y en azul los clasificados por el algoritmo.

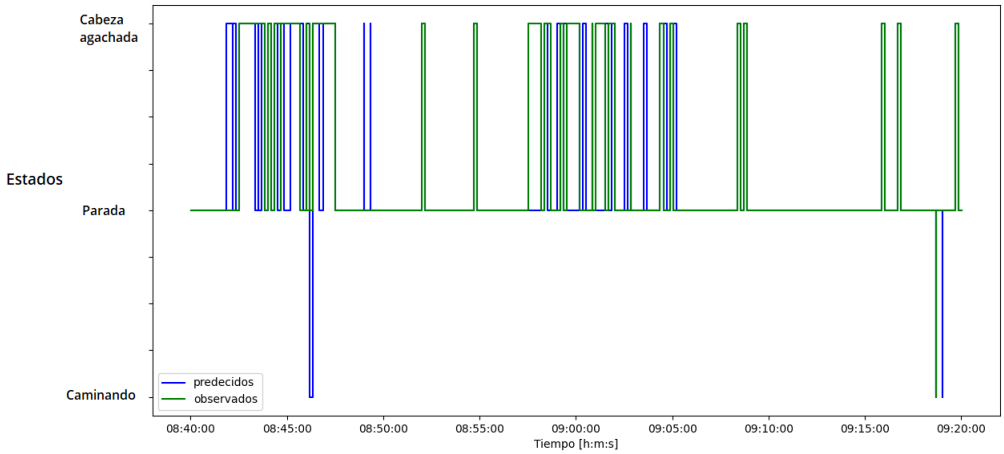

Figura 7.2: Estados clasificados contra observados 8:40-9:20

A partir de los datos observados con los datos clasificados se crea la matriz de confusión 7.1, la diagonal de la tabla representa los Estados acertados mientras que el resto son los Estados mal clasificados. Analizando en detalle la matriz de confusión se obtienen los resultados de la tabla 7.2. Se destaca que el valor máximo posible es 1.

7.1.2. Oveja corriendo

Se realiza el mismo estudio en otro rango de tiempo. En este caso en la gráfica de la figura [7.2], los Estados clasificados corresponden al azul mientras que los

Obs\Pre	Caminando	Parada	Cabeza agachada	Corriendo	Total
Caminando	1				1
Parada	2	311	15		328
Cabeza agachada		34	52		86
Corriendo				0	0
Total	3	345	67	0	415

Tabla 7.1: Matriz de confusión 8:40-9:20

	Caminado	Parada	Cabeza agachada	Corriendo	Promedio
Exactitud por estado	0.33	0.90	0.78	—	0.873
Sensibilidad por estado	1	0.95	0.60	—	0.877
Efectividad total	0.88				

Tabla 7.2: Resultados de análisis 8:40-9:20

verdes corresponden a los Estados observados.

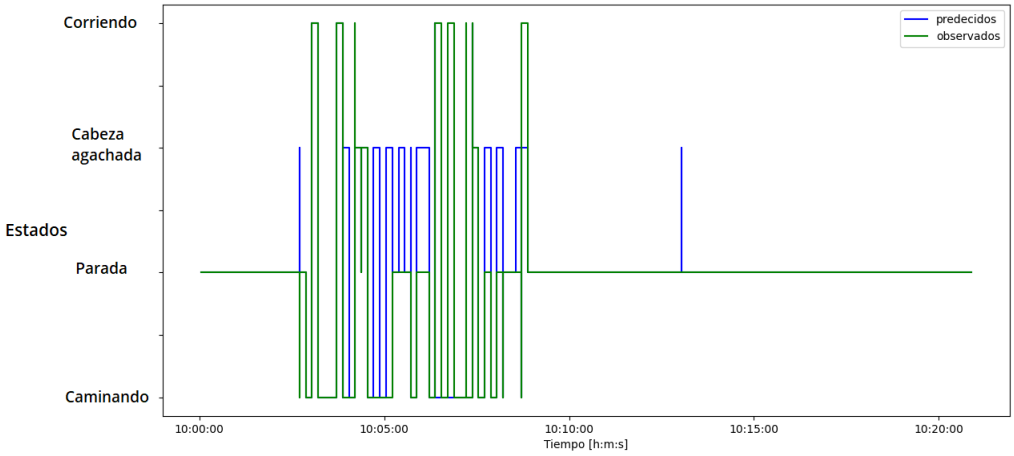

Figura 7.3: Estados clasificados vs observados 10:00-10:20

En la tabla 7.3 se representa la matriz de confusión para este rango y en la tabla 7.4 sus respectivos resultados.

Se estudiaron estos dos rangos de tiempos porque se considera que son los más característicos para su análisis. El primero fue elegido ya que es donde podemos encontrar la mayor frecuencia de cambio de Estados entre parada y cabeza

Obs\Pre	Caminando	Parada	Cabeza agachada	Corriendo	Total
Caminando	26		6		32
Parada		160	11		171
Cabeza agachada		1	3		4
Corriendo	5		1	5	11
Total	31	161	21	5	218

Tabla 7.3: Matriz de confusión 10:00-10:20

	Caminado	Parada	Cabeza agachada	Corriendo	Promedio
Exactitud por estado	0.83	0.99	0.14	1	0.85
Sensibilidad por estado	0.81	0.93	0.75	0.45	0.79
Efectividad total	0.89				

Tabla 7.4: Resultados de análisis 10:00-10:20

agachada. El segundo rango fue elegido porque presenta los Estados caminando y corriendo fundamentales para un estudio completo. Se puede observar cómo en ambos casos se tiene una efectividad mayor a 88 %, lo cual significa que tiene una gran probabilidad de acierto en general.

Mirando más en detalle se puede observar como el Estado que tiene peor exactitud es el Estado 2 correspondiente a cabeza agachada. Si bien no se identifica una razón particular para esto, se destaca que el algoritmo fue entrenado con un dataset público de un tercero. Lo cual significa que esos datos fueron obtenidos de otro collar, otro acelerómetro y otras condiciones. Por lo tanto, se puede llegar a pensar que los datos que fueron generados en estas pruebas con el acelerómetro puede tener pequeñas diferencias con los del dataset generando predicciones erróneas. Entonces, existe la posibilidad de que pequeños movimientos con la cabeza se identifican como cabeza agachada cuando visualmente no parece estarlo.

Las condiciones del lugar no fueron las ideales para las observaciones con respecto al Estado 3. El lugar no era lo suficientemente grande como para que la oveja corriera todo lo esperado. Esto provocó que sus corridas fueran de períodos cortos de tiempo dificultado al algoritmo para clasificarlo correctamente ya que para detectar que estuviera corriendo primero tienen que pasar al menos 5 segundos. Además, en muchas ocasiones la diferencia entre una corrida y una caminata rápida era muy confusa de identificar visualmente. Es por eso que se puede observar cómo la sensibilidad del Estado 3 no es muy buena.

Por otro lado, las pruebas fueron realizadas en un único animal y en un solo

día. Para lograr conclusiones más fuertes sobre el desempeño del clasificador es necesario hacer más pruebas que permitan evaluar más variables. Es decir, probar el collar en distintas ovejas de distintos tamaños y en franjas horarias distintas. Igualmente, los resultados de las pruebas se consideran exitosos, concluyéndose que el algoritmo es capaz clasificar correctamente en gran medida.

Como un trabajo a futuro para lograr una mejor exactitud del Sistema, se puede entrenar el algoritmo con los datos del acelerómetro obtenidos en la prueba de campo. Con esos nuevos parámetros del algoritmo volver a realizar la misma prueba y comprobar que los resultados sean mejores. También se recomienda un lugar con más espacio para realizar las pruebas y que la oveja pueda correr más libremente.

Además se observa que el algoritmo no hace uso de la información temporal como ser la continuidad entre un Estado y otro. Eventualmente podría introducirse una etapa de post procesamiento donde se corrijan datos sabiendo que no es coherente con aquellos próximos en el tiempo (por ejemplo un filtro pasa-bajos).

7.1.3. Resultados GPS

Para una visualización de las ubicaciones geográficas relevadas del GPS, en la figura 7.4 se representan sobre un mapa de imagen satelital las coordenadas de 500 muestras.

Figura 7.4: 500 datos relevados del GPS posicionados sobre un mapa del sitio de pruebas

Esta imagen se obtiene del programa web *maps.co*, que permite darle como entrada un archivo .CSV con todas las coordenadas que se quiere posicionar sobre el mapa de imagen satelital. El corral en el que se realizaron las pruebas es reconocible desde la imagen y se dibujan aproximadamente los bordes del mismo sobre la figura.

Se destaca que hay una mayor densidad de puntos sobre la zona en que había comida y los puntos que salen del rectángulo corresponderían a la zona por la que entraron los animales al corral.

No se hicieron pruebas exhaustivas para determinar la precisión del GPS. Sin embargo, flexibilizando el requerimiento de tener una precisión de 5m, los resultados presentados alcanzan para validar el uso del GPS. El rectángulo trazado representa un área de $300\,\mathrm{m}^2$, dentro del cual se puede identificar con claridad las zonas por las que la oveja estuvo. Esto es en definitiva lo que se espera del GPS, poder saber en que zona se encuentra dentro de un área de dimensiones similares o mayores a esta.

Entonces, se evalúa cualitativamente el requerimiento establecido de 5 m de precisión comparando el área de incertidumbre con el área de pruebas visualizada. Al definir el requerimiento se esperaba que, dado un par de coordenadas, el Equipo se pudiera ubicar en cualquier punto dentro de una circunferencia de 5 m de radio. En la figura [7.4], observando el área que representa esta precisión en el mapa, alcanza para una buena intuición de que el requerimiento se cumple. Aún así, a futuro sería deseable realizar un análisis más exhaustivo. Esto podría realizarse por ejemplo contrastando los datos del Equipo con los de otro dispositivo fiable como podría ser el GPS de un celular.

Por otro lado, la precisión que se planteó como ideal, 1 m, resulta aún menos verificable con precisión con solo este estudio. De todas maneras, se puede asumir que ese nivel de precisión es sobredimensionado. No es necesaria tanta precisión para obtener información útil sobre la posición del animal dentro de un área de las dimensiones de un corral.

7.2. Caracterización del consumo de corriente del Equipo

Las pruebas de consumo de corriente se realizaron empleando el instrumento OTII Arc, de Qoitech [24]. Junto con su software para computadora, el instrumento actúa como fuente de poder y analizador de consumo.

Se busca caracterizar el consumo de corriente del Equipo mediante el estudio individual de los modos de funcionamiento y tareas que se realizan. Mediante estas pruebas se evalúa el uso de distintas configuraciones que tiene Narrowband IoT para aplicaciones de bajo consumo explicados en la sección 3.1.1. Es de interés estudiar el consumo del módulo de comunicación en ciclos de transmisión, de stand-by y apagado. Así con esta información poder estimar el consumo del Equipo, cualquiera sea el modo de funcionamiento y configuración.

7.2.1. Configuración para las pruebas

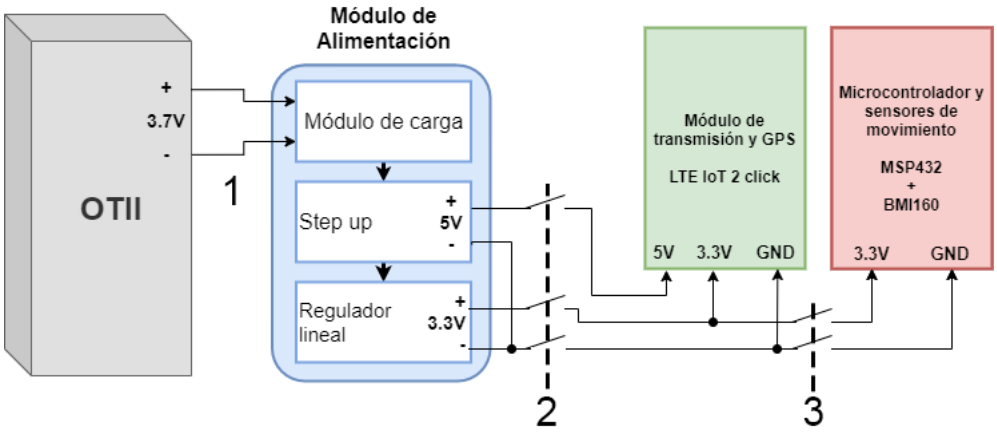

Figura 7.5: Configuración de conexiones para medida de consumo con el instrumento OTII Arc

Se presenta en la figura [7.5] un esquema simplificado de las conexiones para realizar las medidas de consumo. Para las medidas se desconecta la parte del *debugger* del microcontrolador, removiendo los conectores correspondientes. En la sección 2.3 se detalla el armado del sistema de alimentación. Las líneas punteadas representan las conexiones que se hacen para cada parte del estudio. Se realiza el estudio en tres partes, que se definen a continuación:

- Parte 1: Se conecta el OTII únicamente al sistema de alimentación, sin que este alimente al resto del Equipo.
- Parte 2: Se agrega la conexión al módulo de transmisión y GPS para caracterizar su consumo según sus diferentes configuraciones. Aún sin conectar el microcontrolador y sensores.
- Parte 3: Se agrega la conexión al microcontrolador con los sensores de movimiento para realizar medidas de consumo del Equipo completo.

7.2.2. Consumo del sistema de alimentación en vacío

Primero se mide el consumo que introduce el sistema de alimentación por sí mismo, cuando este no alimenta ninguna carga posterior. Esto generará una noción del mínimo consumo de corriente que siempre se tendrá. Dado que tres módulos de hardware que no están diseñados para el bajo consumo componen al sistema de alimentación, se espera tener un consumo de corriente elevado.

Se conecta el OTII Arc entregando 3.7 V al sistema de alimentación y se obtiene el resultado de la tabla 7.5.

	Corriente promedio consumida (mA)	Corriente esperada (mA)
Sistema de alimentación	15.4	11.36

Tabla 7.5: Medidas de consumo de corriente promedio del sistema de alimentación Cálculo corriente esperada: Apéndice C.

Se presentan los cálculos para el valor esperado en el apéndice C.2.2. Se destaca que el consumo es elevado para ser empleado en un sistema embebido de bajo consumo. Este resultado es producto de distintos factores. Primero, emplear tres módulos de hardware para modificar el nivel de tensión inevitablemente impone ineficiencias. Por otro lado, los módulos empleados no son los más indicados para una aplicación de estas características. En particular, el step down no está diseñado específicamente para electrónica de bajo consumo. Sirve para voltajes mayores en aplicaciones para las que consumos de corriente del orden de 15 mA no son un problema. Se usó este componente porque no se encontraron mejores alternativas en venta en plaza.

7.2.3. Caracterización de consumo del módulo LTE IoT 2 click

Se busca caracterizar el consumo del módulo de forma independiente al microcontrolador y al flujo del código ya que no se cuenta con documentación oficial acerca de su consumo de corriente.

Se conecta al sistema de alimentación el módulo LTE IoT 2 click que requiere de ambos niveles de tensión, 3.3 V y 5 V. Primero se mide el consumo base para el estudio. Esto quiere decir que, una vez conectado, el módulo BG96 se encuentra por defecto apagado, por lo que el consumo que se desea medir es aquel dado por la circuitería agregada.

	Corriente promedio consumida (mA)	Corriente esperada (mA)
Sistema de alimentación	15.4	11.36
Sist. de alimentación +LTE IoT 2 click apagado	20.6	14.84

Tabla 7.6: Corriente consumida por el módulo LTE IoT 2 click. Cálculo corriente esperada: Apéndice C.

Esta corriente representa una noción del piso de consumo que estará presente en cualquier implementación de software embebido, lo cual compromete la vida útil del Equipo. El consumo que impone incluso estando apagado se explica por componentes que tiene la placa que inevitablemente consumen corriente. Uno es

un LED que está constantemente prendido, indicando que el módulo recibe alimentación. También tiene reguladores de voltaje y conversores de nivel de voltaje que pueden imponer este consumo.

Evaluación de modo bajo consumo eDRX

Se tiene el modo de bajo consumo eDRX como alterativa para mitigar el consumo del módulo LTE IoT 2 click, explicado en la sección 3.1.1. En base a mediciones de consumo se busca corroborar su funcionamiento y dados los resultados, determinar la mejor forma de emplear este modo.

En la siguiente figura se puede ver gráficamente el consumo generado durante un ciclo eDRX habitual.

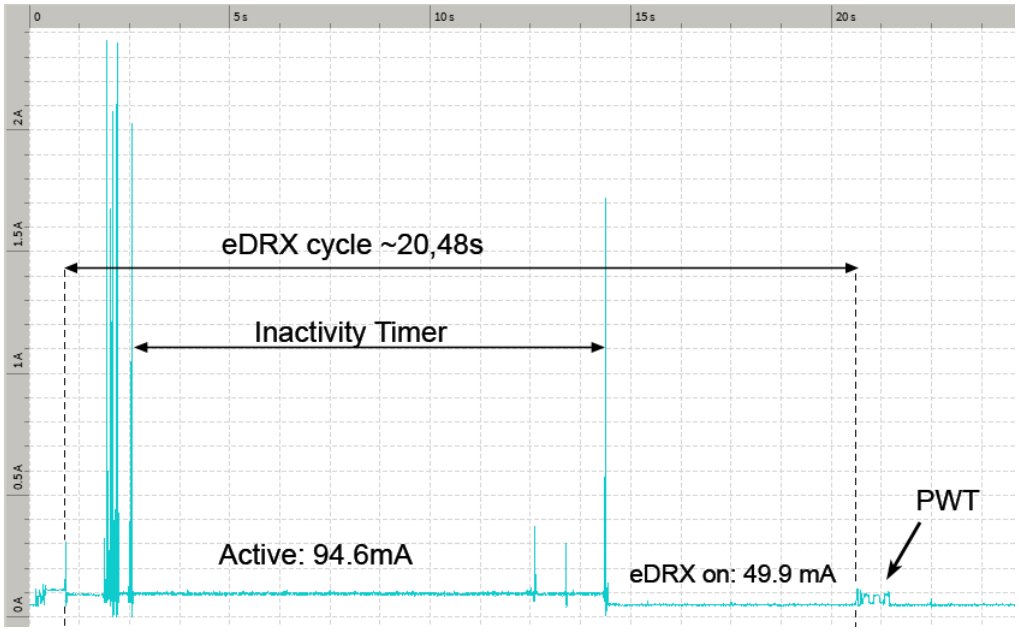

Figura 7.6: Consumo medido y tiempos de un ciclo eDRX

Se prueba configurando el ciclo eDRX en 20.48 s, uno de los 16 valores que permite configurar el módulo. Este tiempo coincide con la gráfica obtenida. En la figura [7.6] se puede ver que al comienzo del ciclo, marcado por la línea negra punteada, el módulo se encuentra en modo activo preparado para recibir o enviar mensajes. Luego hay un evento entre el tiempo t=1 s y t=2 s, sea transmisión o recepción, que genera picos de consumo elevados.

Finalizado este evento, entre t=14 s y t=15 s, comienza a correr el timer de inactividad (*inactivity timer*). Una vez vencido el timer, el módulo entra en el modo eDRX que es cuando efectivamente deshabilita la recepción de mensajes. Se puede ver como esto hace bajar el consumo a casi la mitad comparado con el modo activo que está constantemente escuchando a la red. Por último, una vez finalizado el tiempo del ciclo eDRX sin que haya más actividad, se hace la PTW (Paging Time Window).

La duración del ciclo es configurable y en la aplicación desarrollada el módulo no debe recibir ningún mensaje, mas que los que emplee la red para su propio control y configuración. Por ende, lo mejor es configurar este ciclo lo más grande posible así no se desperdicia carga en prender la antena de recepción para el PTW.

De las mediciones realizadas se presenta el siguiente cuadro de resultados:

	Consumo de corriente (mA)		
	Medido	BG96 (consumo típico)[1]	Diferencia: consumo medido y típico (LTE IoT 2 click encendido)
Modo Activo	94.6	47.0	47.6
Modo eDRX On	49.9	15.0	34.9

Tabla 7.7: Resultados de consumo eDRX

Se confirma que este modo disminuye significativamente el consumo para aplicaciones en que no sea de interés recibir datos frecuentemente. Lo cual se orienta a aplicaciones tipo IoT donde el interés es principalmente reportar información.

No se tiene documentación del consumo del módulo LTE IoT 2 click de Mikroe. Por consiguiente se decide hacer un ensayo con una carga resistiva de valor conocido. De esa manera se busca determinar si el consumo agregado es producto de la placa de desarrollo o del sistema de alimentación.

Para dicha prueba se consideró una resistencia de 100 Ω la cual, sometida a una tensión de 5 V, consumiría 50 mA, similar al consumo típico en modo activo del BG96 como puede verse en la tabla 7.7. Véase en la figura [7.7] un diagrama de la configuración.

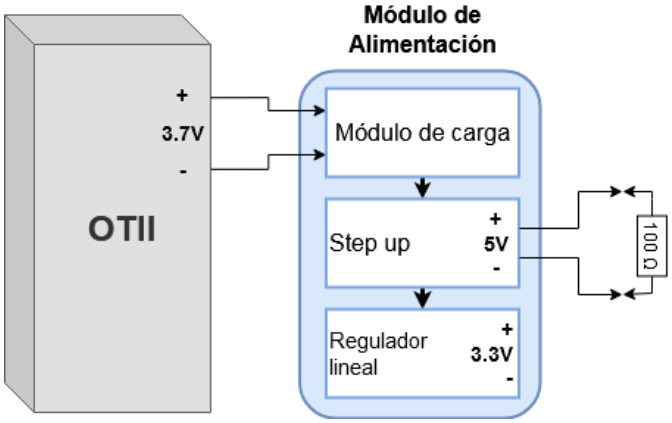

Figura 7.7: Configuración de conexiones para ensayo sobre resistencia

La medida de consumo de corriente para este ensayo puede verse en la figura [7.8].

Puede observarse que el resultado de la medida es aquel reportado en la tabla 7.8. Este es muy similar al obtenido con el LTE IoT 2 click (tabla 7.7) pudiéndose

[1]Consumo típico reportado en [25]

concluir que es el circuito de alimentación el que impone las diferencias entre consumos medidos y aquellos de la documentación.

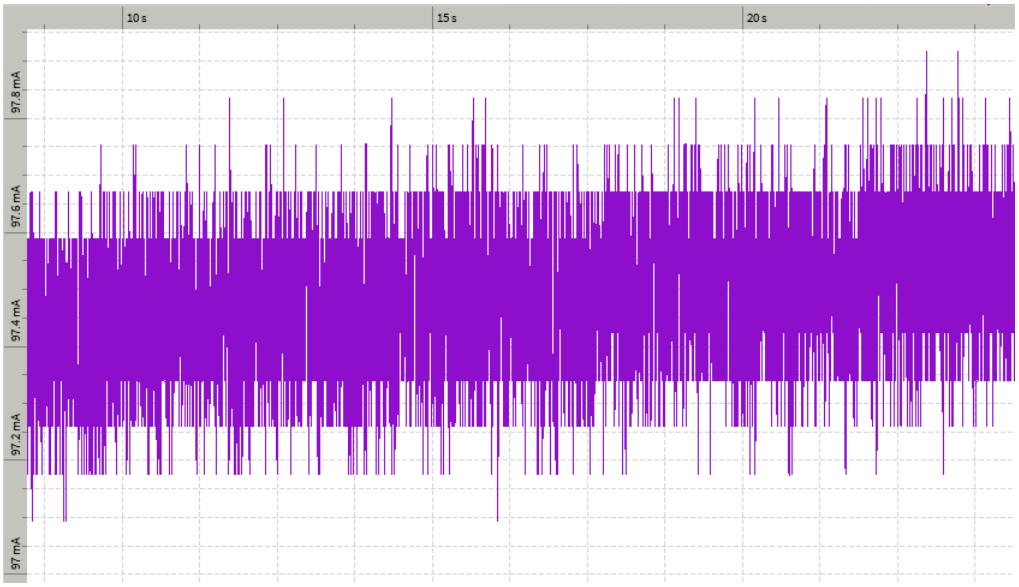

Figura 7.8: Consumo medido con resistencia de $100\,\Omega$

Consumo esperado de la carga ($100\,\Omega$)	Consumo medido de la carga (con módulo de alimentación incluido)
50 mA	97.3 mA

Tabla 7.8: Medida de consumo para ensayo sobre una resistencia de $100\,\Omega$ a $5\,\mathrm{V}$ con el módulo de alimentación.

Consumos de conexión a la red MQTT

Se quiere medir cuánto consume el módulo LTE IoT 2 click para establecer la conexión a la red MQTT y cuánto consume en transmitir cierta cantidad de datos. Estas medidas serán útiles para definir ciertos parámetros de la aplicación aquí propuesta. Primero, se presenta una medida de consumo durante la conexión a la red en la figura [7.9] .

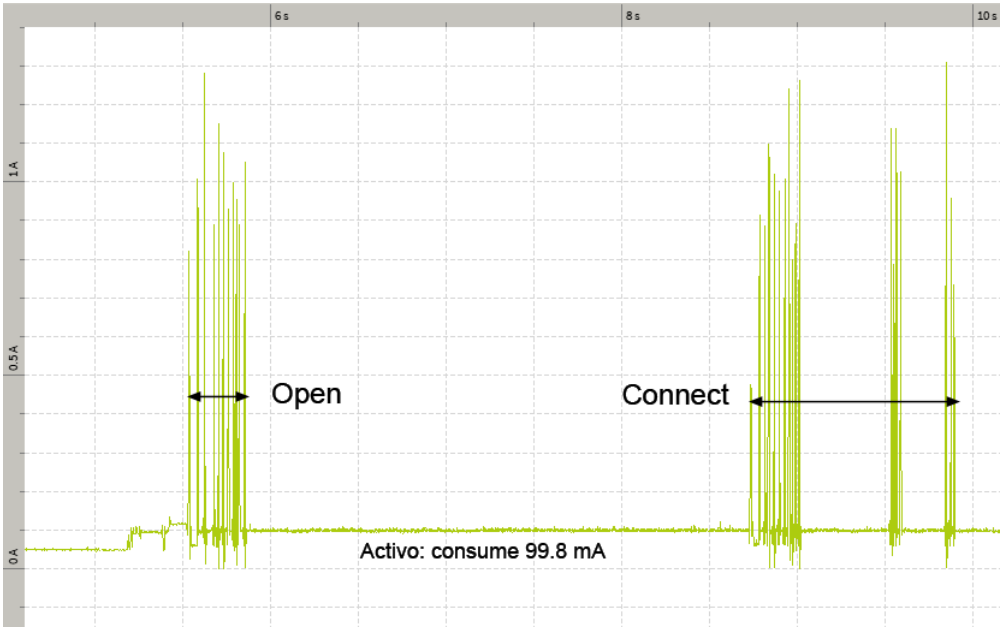

Figura 7.9: Consumo medido en establecimiento de conexión a MQTT

Para establecer conexión a la red MQTT se requiere ejecutar dos comandos AT en el módulo, que son *Open* y *Connect*. Se realiza un intercambio de información con la red que genera el consumo de corriente visto en la figura, donde se diferencia en tiempo el consumo generado por cada comando. Sumado al consumo propio de realizar la transmisión de información, el módulo permanece en modo activo durante este tiempo. Este procedimiento implica un consumo no despreciable, por lo que sería deseable minimizar la cantidad de conexiones a la red que realiza nuestra aplicación.

De las medidas se obtienen los resultados de la tabla 7.9.

	Tiempo (ms)	Consumo promedio (mA)
Open	336	162
Connect	1174	126

Tabla 7.9: Consumo de corriente en comandos de conexión a la red MQTT

Estos resultados sirven para tener una idea del consumo que implica este procedimiento. Este caso representa un mínimo en cuanto a tiempo y consumo, dado que en el establecimiento de conexión la negociación con la red puede implicar una mayor o menor cantidad de intercambios dependiendo de las condiciones de la red y señal, pudiendo entonces implicar más tiempo y más consumo. Se concluye entonces que se debe minimizar la cantidad de veces que se realiza este procedimiento, a modo de evitar perder tiempo y carga en establecer conexión durante la ejecución de la aplicación diseñada.

Se analiza el impacto de este procedimiento según el modo de funcionamiento del Equipo, Validación e Investigación (ver sección 5.4). Para el Modo Validación

en el cual se deben transmitir 4500 bytes de datos crudos cada 5 segundos, se concluye que no es viable conectarse y desconectarse de la red periódicamente. A lo largo del desarrollo del proyecto se observa que el procedimiento de conexión toma aproximadamente 2 segundos en los mejores casos, con casos en que toma varios segundos más. Por ende, para tener suficiente tiempo para transmitir toda la información periódicamente en ciclos de 5 segundos, el módulo debe permanecer constantemente prendido y conectado. Por otro lado, en el Modo Investigación se transmite un único paquete de datos (hasta 1500 bytes) en intervalos de tiempo de entre 2 y 10 minutos. Si cada transmisión requiere conectarse a la red, el consumo asociado a la transmisión de cada paquete tiene una gran componente correspondiente a la conexión sumada al envío del paquete en sí mismo.

Por lo anterior, se busca una forma de evitar realizar una reconexión cada vez que se quiera transmitir información. Se encuentra como potencial solución el empleo del modo PSM, descrito previamente en la sección 3.1.1. Si bien el análisis sobre este modo será realizado, la aplicación final no lo incluye debido a lo que será expuesto a continuación.

Uso de Power Save Mode (PSM)

La intención del uso de esta configuración del BG96 es mantener la conexión del Equipo a la red Narrowband IoT mientras el módulo LTE IoT 2 click entra en un modo de bajo consumo o como se le denominó en la sección 3.1.1, reposo profundo. Para esto, el módulo negocia previamente tiempos con la red, avisando cuando tiempo estará inalcanzable y así evitar que la red lo desconecte.

	Corriente promedio consumida (mA)
Módulo apagado	20.6
Módulo en PSM	20.6

Tabla 7.10: Consumo módulo apagado vs PSM

Se confirma con los resultados obtenidos que el modo PSM es, a efectos prácticos, equivalente a apagar el módulo por completo (ver tabla 7.10). Por lo tanto, es una solución acorde a las necesidades para la reducción de consumo. Sin embargo, se encuentran impedimentos para su uso.

Para el Modo Validación, los requerimientos de tiempos son tan elevados que el módulo nunca tiene una ventana de tiempo suficientemente grande para poder irse a reposo o apagar el módulo LTE IoT 2 click. Para activar PSM se debe avisar a la red y un timer de inactividad debe cumplirse para que el módulo decida apagarse. No hay tiempo para que se cumpla este timer siendo que se transmite cada 5 segundos una gran cantidad de datos.

Para el Modo de Investigación, se encuentran conflictos con el empleo de PSM sobre el protocolo MQTT. Se pueden negociar los parámetros de PSM con éxito, pero al momento de mandar a reposo al módulo se pierde la conexión a la red MQTT. Al activar PSM, el módulo responde con los siguientes mensajes:

+QIURC: "pdpdeact",0
+QMTSTAT: 0,1

El primer mensaje indica que la red desactiva el contexto PDP del módulo, es decir Packet Data Protocol. La conexión a MQTT funciona sobre protocolo TCP y desactivar este contexto implica la pérdida de conexión. La segunda respuesta simplemente significa que la conexión a MQTT se cerró. No se encontró forma de entrar en PSM manteniendo el contexto PDP activado, por lo tanto no se logra mantener abierta la conexión.

Entonces lo único que es capaz de aportar el modo PSM es ahorrar consumo en reconectarse a la red Narrowband, pero no a MQTT. En nuestro sistema, el consumo que se quisiera evitar entrando a PSM es en realidad aquel dado por la conexión a MQTT dado que es bastante significativo como fue visto anteriormente. Al no ser este el caso y resultar que las medidas de consumo del modo PSM no presentan mejora perceptible frente a los consumos de MQTT se opta por sencillamente apagar el módulo.

Consumo en transmisión 1000 bytes vs 1500 bytes

En el Modo Validación se transmiten todos los datos crudos obtenidos del acelerómetro. 1548 bytes es el limite de bytes que se pueden transmitir en la publicación de un mensaje mediante MQTT, con el módulo empleado. De momento se transmiten 1500 bytes (casi el máximo) de datos por mensaje bajo la lógica de: 500 datos compuestos por 3 bytes, donde 2 bytes corresponden a datos crudos del acelerómetro y 1 byte de control (la funcionalidad del byte de control se explica en la sección 5.5.10). Modificando la lógica de transmisión se podría reducir a 1000 bytes como mínimo si se eliminan los bytes de control. La desventaja sería perder fidelidad en los datos, dado que se debe evitar transmitir ciertos valores que representan caracteres ASCII usados como caracteres de control.

Por esto se compara el impacto a nivel de consumo entre la transmisión de 1000 y 1500 bytes.

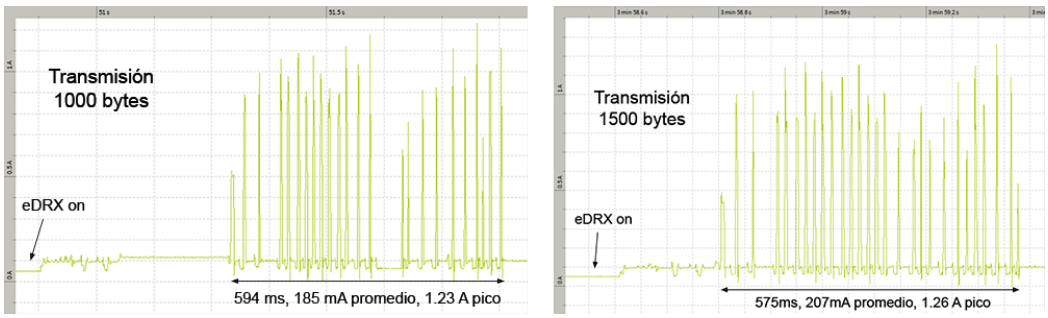

Figura 7.10: Comparación de consumo entre transmisión de 1000 bytes y 1500 bytes

Lo primero que se puede notar de los resultados es que el tiempo de transmisión del mensaje es, no solo similar, sino que mayor en el caso de transmitir 1000 bytes.

Se puede ver cómo la transmisión de 1000 bytes tiene un tiempo de inactividad en el medio, en donde la antena no transmite. Sin embargo esto no se da en el caso de 1500 bytes.

	Tiempo (ms)	Consumo promedio (mA)	Pico de corriente (A)
1000 bytes	594	185	1.23
1500 bytes	575	207	1.26

Incremento de corriente promedio	10 %

Tabla 7.11: Mediciones de transmisión de mensaje 1000 bytes y 1500 bytes

Los picos de corriente son prácticamente iguales, estos varían caso a caso y se llega a medir 1.37 A como máximo. La diferencia entre ambas transmisiones recae en el valor promedio de la corriente durante este tiempo, que resulta levemente superior en la transmisión de 1500 bytes. Ampliando el margen de tiempo de la transmisión de 1500 bytes para equiparar, se tiene que la corriente promedio pasa a ser de 204 mA, de donde se obtiene como resultado que la transmisión de 1500 bytes consume un 10 % más de corriente, a pesar de ser un mensaje 50 % más largo.

Puede concluirse entonces que el ahorro en enviar menor cantidad de bytes no es tan significativo. Aún así queda un factor de influencia que no pudo ser cuantificado, y son las retransmisiones. El protocolo MQTT emplea TCP por lo cual se harán retransmisiones en caso de que se pierdan datos. Mandar mayor cantidad de datos implica una mayor probabilidad de que alguno se pierda y por lo tanto requerir una retransmisión.

Fuera del marco de este proyecto, puede quedar como trabajo futuro la optimización del armado de mensajes de transmisión para reducir la cantidad de bytes enviados.

7.2.4. Consumo del Equipo según modo de funcionamiento

Habiendo estudiado el consumo del módulo de comunicación por si solo, se procede a tomar mediciones del Equipo completo en funcionamiento, diferenciando según el modo de operación.

Consumo del Equipo en Modo Validación

Primero se presenta la figura [7.11], correspondiente al Equipo configurado en Modo Validación, midiendo desde el inicio de ejecución del programa. Se puede ver como al principio, t=0 s, el consumo es muy reducido dado que el módulo de transmisión se encuentra apagado. Luego se enciende para realizar las configuraciones necesarias y la primer ventana de consumo elevado (aproximadamente t=16 s) corresponde al establecimiento de conexión a MQTT ejecutando los comandos de *open* y *connect*. Luego se ven 5 ventanas de transmisión y muestreo del GPS en lo que resta del tiempo. En toda la medida presentada en esta figura, el GPS aún

no ha logrado fijar su posición. Es decir que está activamente buscando satélites para triangular su posición, lo cual podría implicar un consumo mayor que cuando los encuentre.

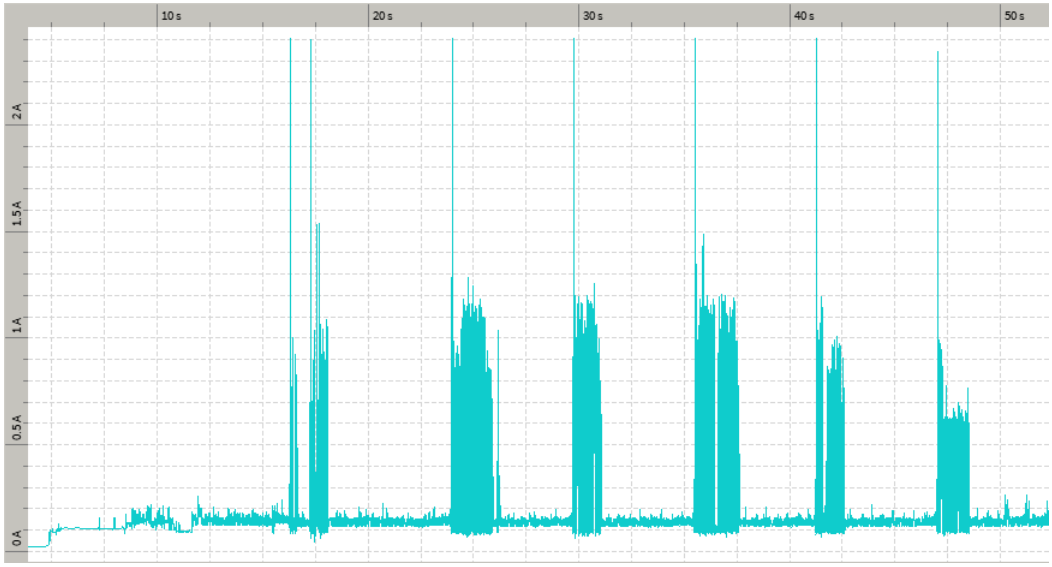

Figura 7.11: Consumo del Equipo en Modo Validación en inicio de ejecución

El Modo Validación (ver sección 5.4), consiste en enviar datos cada 5 segundos. Se puede ver que los picos de consumo son bastante mayores que al medir el consumo del módulo de transmisión por si solo. Esto se explica porque ahora se tiene el sistema completo en funcionamiento, en particular la antena de GPS también está encendida. Se nota que en stand-by el consumo también es más elevado y ruidoso. Llamando Idle al tiempo en que no se está transmitiendo, y no se ven relieves en la gráfica. De los datos relevados se obtienen los valores de la tabla 7.12.

Corriente Idle	132 mA
Corriente promedio	175 mA
Corriente máxima	2.41 A

Tabla 7.12: Consumo Modo Validación, GPS sin posición fijada

El consumo es elevado, pero para este modo se prioriza la transmisión de la totalidad de los datos relevados y posicionamiento GPS por sobre la vida útil del Equipo. También llama la atención el valor máximo de corriente que llega a medirse. Tener que entregar 2.41 A de corriente implica restricciones sobre el Equipo de alimentación que debe tolerar este requerimiento.

Esto generó problemas con el sistema de alimentación diseñado. El módulo regulador de carga que se compró protege las baterías contra demandas tan altas de corriente, por lo qué el módulo de transmisión no obtenía la corriente que requería

y fallaba la transmisión. Se solucionó alimentando el circuito directamente desde las baterías.

En la figura [7.12] se presenta otra gráfica de medida de consumo que presenta un comportamiento particular:

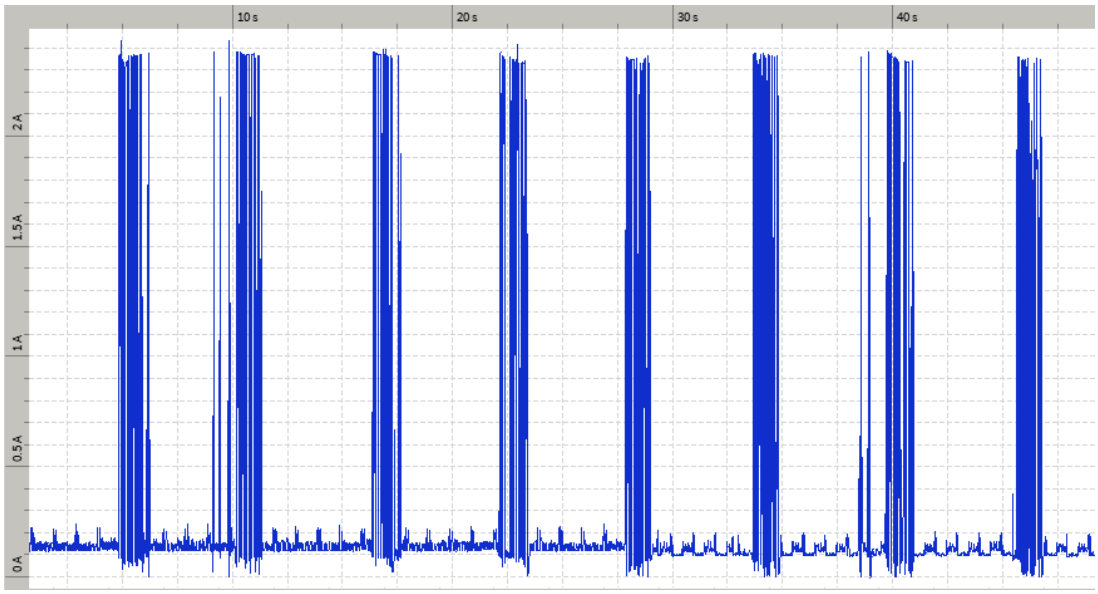

Figura 7.12: Consumo del Equipo en Modo Validación, cambio en el GPS de *searching* a *tracking*

Se puede ver cómo a la mitad de la medida, la corriente de stand-by se reduce notoriamente mientras que los picos de transmisión se mantienen iguales. La explicación de este comportamiento debe recaer sobre la funcionalidad GPS del módulo, dado que el comportamiento de esta funcionalidad es el único que no se controla directamente y todo lo demás opera en un régimen constante.

De la figura [7.12], se obtienen los resultados de la tabla 7.13.

	Principio	Final
Corriente stand-by	132 mA	113 mA
Corriente promedio	177 mA	158 mA
Corriente máxima	2.44 A	

Tabla 7.13: Medidas de consumo en Modo Validación con diferentes comportamientos del GPS

La diferencia es notoria. Se atribuye esta diferencia al modo de funcionamiento en que se encuentra actuando el GPS. Estos son dos, llamados *searching* y *tracking*. El primero refiere a cuando el GPS busca comunicación con al menos 3 satélites para poder triangular su posición. Una vez que encuentra estos satélites, pasa a modo de tracking en el cual solo debe mantener la señal con los satélites que conoce. Durante tracking el consumo del GPS es menor, lo cual explica la diferencia observada.

Esta suposición se verifica con el consumo reportado por la documentación del chip BG96, donde se reportan los consumos de la tabla 7.14.

	Conditions	Current consumption typ. (mA)
Searching	Cold start (passive antenna)	41.7
	Lost state (passive antenna)	42
Tracking	Instrument environment	21.7
	Open sky (passive antenna)	36
	Open sky (active antenna)	35

Tabla 7.14: Consumos del GPS reportados por Quectel [25]

La información no es muy detallada pero se ve que existe la posibilidad de tener una diferencia de hasta 20 mA según el modo, tal como dan los resultados de medición. En este caso, se tiene una antena activa y la condición de inicio es *cold state*. No se encuentra una definición de *Instrument environment*, pero las medidas tampoco son relevadas a cielo abierto sino en ciudad y bajo techo.

De estos resultados se concluye que es beneficioso asegurar que el GPS mantenga fijada la cantidad de satélites necesaria para permanecer en modo de tracking y así ahorrar consumo. Lo cual justifica el uso de una antena de buenas características para lograr esto. En particular, se usa una antena activa qué asegura mejor señal que una pasiva. Encuentra señal en tiempos mucho menores que una antena pasiva y no pierde señal tanto como la pasiva.

En base a los resultados presentados en la tabla 7.13, se calcula la vida útil del Equipo. Se asume tener una batería con 8000 mAh de carga. Diferenciando según el estado del GPS, se obtienen los siguientes resultados límite:

	Corriente promedio consumida	Vida útil con baterías de 8000 mAh de carga
GPS searching	177 mA	45.2 horas
GPS tracking	158 mA	50.6 horas

Tabla 7.15: Estimación de vida útil del Equipo en Modo Validación

Se presentan dos valores correspondientes a la vida útil del Equipo si el GPS estuviese siempre en modo searching o siempre en modo tracking. Esto presenta dos limites posibles, aunque en la práctica el valor real se debería acercar al de tracking. No es posible estimar cuánto tiempo pasaría en cada modo porque depende de condiciones que no están bajo nuestro control. Pero a cielo abierto, seguro pasa la mayor parte del tiempo en tracking.

Dada la alta demanda de transmisión de datos de este modo, estos resultados se consideran satisfactorios porque cumple con los requerimientos necesarios para validar el algoritmo, con una vida útil superior a un día.

Consumo del Equipo en Modo Investigación

En este modo se busca extender la vida útil del Equipo para obtener una autonomía mayor. Por esto se enciende el módulo de transmisión por ventanas cortas de tiempo, separadas varios minutos. Se transmiten únicamente los Estados clasificados junto con ubicación geográfica. La duración de estas ventanas de tiempo depende de cuanto tiempo tarda el GPS en fijar su ubicación geográfica, por esto es beneficioso usar una antena activa.

Se presenta a continuación una figura con el consumo en corriente relevado para este modo. Para estas pruebas, se configura el modo para transmitir información cada 2 minutos.

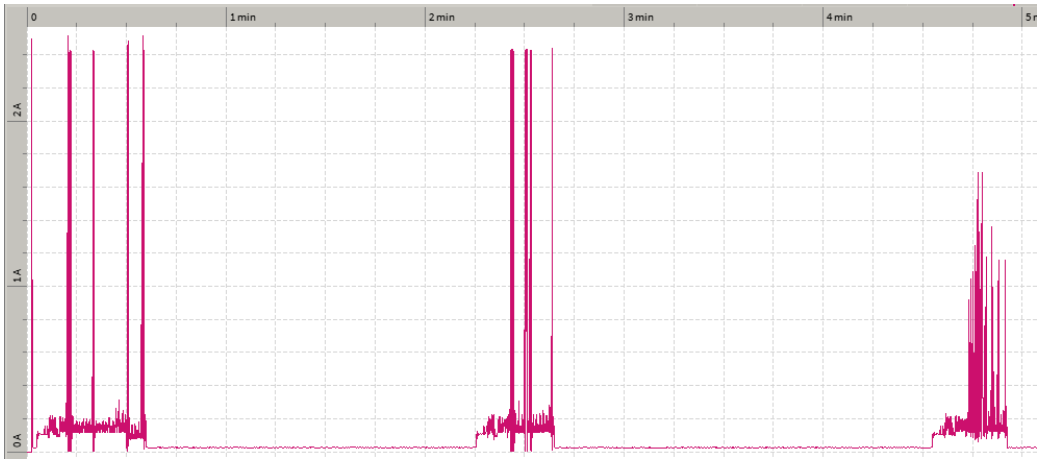

Figura 7.13: Consumo del Equipo en Modo Investigación

Se observa el comportamiento esperado para este modo, transmitiendo información cada 2 minutos. Se tienen ventanas de alto consumo correspondientes al tiempo en que se enciende el módulo LTE IoT 2 click, separadas aproximadamente 2 minutos. En cada ventana se establece la conexión al servidor MQTT, se transmiten todas las caracterizaciones relevadas en los últimos 2 minutos y se mantiene encendido hasta fijar ubicación geográfica. Por esto se puede ver que la primer ventana es más ancha que las siguientes, el GPS tarda más tiempo en fijar su ubicación cuanto más tiempo estuvo apagado.

De este estudio se relevan los valores de la tabla 7.16.

	Corriente consumida promedio	Duración promedio
Módulo LTE apagado	25 mA	113 s
Ventana de transmisión	140 mA	24 s
Ciclo completo	45 mA	137 s

Tabla 7.16: Medidas de consumo en Modo Investigación, ciclo de 2 minutos

Se destaca que, aunque el ciclo se configura teóricamente en 2 minutos, la duración real es algunos segundos superior porque el firmware del Equipo recomienza

cada ciclo una vez transmitida la información del anterior.

El consumo del Equipo con el módulo LTE IoT 2 click apagado es elevado pero coincide con las medidas presentadas anteriormente. Reducir este consumo de 25 mA es sin duda una de las modificaciones más importantes que se le debería hacer a este proyecto a modo de trabajo futuro.

El consumo promedio en un ciclo completo está ligado a la duración configurada para el ciclo y la duración de la ventana de transmisión y GPS. Cuanto más largo se configure el ciclo, más tiempo se tendrá apagado el módulo. Se realizan las pruebas con un ciclo de 2 minutos pero se puede configurar a elección, hasta 60 minutos limitado por la implementación del código. Una mayor duración del ciclo compromete la frecuencia con la que se recibe ubicación geográfica. Definir esta frecuencia quedaría a discreción quien haga uso del Equipo y el estudio que se quiera realizar.

Midiendo los tiempos obtenidos en el relevamiento de consumo, presentados previamente en la tabla 7.16, es posible estimar la vida útil del Equipo para cualquier duración de ciclo que se defina. Para esto se calcula el consumo promedio por ciclo asumiendo que la duración de la ventana de transmisión será siempre la misma. Se presentan estos resultados a continuación.

Tiempo de ciclo configurado	Consumo promedio	Vida útil con baterías de 8000 mAh
2 minutos	45.1 mA	7.4 días
5 minutos	33.7 mA	9.9 días
10 minutos	29.5 mA	11.3 días

Tabla 7.17: Estimación de vida útil del Equipo en Modo Investigación según tiempo de ciclo

Se logra el objetivo de superar una autonomía de 5 días, siendo necesario para esto contar con 8000 mAh de carga, lo cual se logra con 2 baterías tipo 18650 de 4000 mAh.

Se puede ver el incremento de vida útil al aumentar el tiempo de ciclo. Este incremento es decreciente a medida que la corriente promedio se acerca al valor 25 mA, siendo este el consumo del Equipo con el módulo de transmisión apagado.

No se considera para estos cálculos el aporte del panel solar. Dado el elevado consumo que se tiene, el panel solar no puede aportar carga comparable y solo es capaz de añadir algunas horas de vida útil. Se calcula el aporte del panel solar en la sección C.2.2. Por lo tanto, se pueden considerar estos resultados como el peor caso.

Esta página ha sido intencionalmente dejada en blanco.

Capítulo 8

Conclusiones

En este capítulo se realiza un análisis general del desarrollo del proyecto contrastando los resultados obtenidos con los requerimientos y criterios de éxito preestablecidos (Ver Capítulo de Introducción [1]). Por último, se plantean mejoras al Sistema y ampliaciones que podrían aplicarse al proyecto en trabajos futuros.

8.1. Análisis general del desarrollo del proyecto

Se diseñó y construyó un dispositivo capaz de relevar y procesar datos de una IMU y un GPS para obtener Estados que permiten reflejar el comportamiento de una oveja, con un buen porcentaje de aciertos en la prueba de campo realizada. Con distintos niveles de exactitud, estos Estados son: caminando, quieta, cabeza agachada y corriendo. Dicho procesamiento fue implementado con un algoritmo de aprendizaje automático denominado Análisis del Discriminante Lineal. Además, se implementaron dos modos de funcionamiento tal que en uno se envíen los datos procesados junto a los del GPS a un Sistema Central y en otro también los datos crudos. Para dicha transmisión el sistema utiliza la red Narrowband IoT y sobre ella el protocolo en capa de aplicación MQTT. Luego, para el otro extremo de la comunicación, se programó una interfaz de usuario tal que se pudieran visualizar los datos y guardar los datos en una bases de datos así completando un sistema de gran utilidad en el ámbito de la investigación.

Se concibió el Equipo como la interconexión de diversos módulos hardware a cumplir diversas funciones. Luego de definir la funcionalidad de cada módulo, se procedió a la elección de los componentes que los implementaran. Los módulos son aquel encargado de la adquisición, procesamiento y transmisión mientras que el otro se ocupa de la alimentación. Estos fueron luego probados individualmente para posteriormente ser integrados. La interconexión de los módulos hardware requirió el diseño y fabricación de dos PCBs, cada uno con los submódulos y componentes elegidos. El Equipo entonces consta de estos módulos los cuales se encuentran dentro de una caja impresa en 3D, la cual a su vez posee una cinta y hebilla para sujetar el Equipo al animal. Las pruebas de campo en animales arrojaron resultados excelentes en términos del procesamiento, adquisición y no alteración

del comportamiento habitual del animal.

Se utilizó el microcontrolador MSP432P401R como unidad central del Equipo. Se desarrolló el software embebido en el entorno de desarrollo integrado CCS, utilizando el lenguaje C y eligiendo como arquitectura el Round-Robin con interrupciones. Al igual que el hardware, el software consta de diversos módulos que buscan centralizar funcionalidades. Estos módulos se dividen en software dependiente e independiente del hardware. El software logra entonces controlar todos los periféricos para cumplir el cometido y además maneja potenciales errores ocasionados por problemas en comunicación entre módulos o períodos de inactividad prolongados (timeouts).

Se concibió el Sistema como herramienta para investigaciones futuras, en particular relacionadas con el parto en ovejas, arrojando resultados muy prometedores a bajo costo de procesamiento. Además, se trabajó buscando incorporar tecnologías novedosas como lo son el aprendizaje automático y la red Narrowband IoT y el diseño e impresión 3D.

En lo que respecta a la gestión general del proyecto, la situación sanitaria influyó fuertemente. Se decidió hacer uso de las herramientas de comunicación a distancia y se optó por un enfoque en base a la metodología ágil. El trabajo se organizó en sprints, los cuales son ciclos de una o dos semanas donde se dividen las tareas, se realizan y una vez vencido el plazo, se planifica el nuevo ciclo en base a lo que se logró y si se deben considerar nuevas tareas. La herramienta elegida para la gestión ágil fue el software de Atlassian Jira. Adicionalmente, se optó por emplear componentes de fácil obtención en plaza para la fabricación del Equipo así disminuyendo los imprevistos.

8.2. Evaluación de los resultados respecto a los requerimientos del proyecto

A modo de concluir los resultados del proyecto se evalúan los requerimientos planteados inicialmente. La tabla 8.1 presenta un resumen de todos los requerimientos establecidos en 1.5.

- *El Equipo deberá ser capaz de medir la ubicación geográfica del animal con una precisión de* 5 m.
  Las pruebas no fueron exhaustivas en este aspecto por lo que no se puede afirmar con resultados experimentales. Pero de todos modos se puede asumir que se tiene una precisión suficiente o mayor, una que logra el objetivo buscado del GPS, el cual es poder diferenciar claramente las posiciones del animal dentro de un corral de dimensiones comparables al de las pruebas (área 300 m^2). Se puede llegar a esta conclusión en vista de que los cientos de datos GPS siempre se sitúan dentro del perímetro del corral donde se estuvo moviendo, nunca fuera, y por zonas donde se vio al animal durante

[1] El análisis de costos se puede encontrar en el anexo G

Resumen resultados				
	Aceptable		Obtenido	
Precisión GPS	5 m		Aproximadamente 5m	
Caracterización de Estados	95 % casos		88 % casos	
Peso	600 g		465 g	
Tamaño máximo	14x10x8 cm (largo x ancho x alto)		12.5 x 11.75 x 6.765 cm	
Costo del prototipo	<10000 pesos uruguayos		9182 pesos uruguayos[1]	
Modo	Validación		Investigación	
	Aceptable	Obtenido	Aceptable	Obtenido
Tiempo entre muestras IMU	1 s	10 ms	1 s	10 ms
Tiempo entre muestras GPS	30 s	10 s	5 minutos	2 minutos
Autonomía	3 horas	>1 día	5 días	5 días

Tabla 8.1: Resumen de resultados de requerimientos

las pruebas. Por otro lado, el objetivo ideal de 1 m de precisión resulta ser sobredimensionado, por más que sea alcanzable o no con el GPS empleado.

- *El Sistema Central deberá desplegar en un ordenador la información en forma gráfica. Se deberá contar con una Interfaz de Usuario capaz de desplegar los datos adquiridos.*
  Son dos requerimientos relacionados que logran ser cumplidos mediante la implementación de aplicaciones en una computadora actuando como Sistema Central. Estos programas se encargan de recibir todos los datos transmitidos por el Equipo, guardar la información de forma ordenada en una base de datos y desplegarlos de forma visual. Se puede visualizar en tiempo real que se reciben los datos, obtener gráficas y posicionar las coordenadas del GPS sobre un mapa.

- *Se deberá muestrear la ubicación geográfica al menos una vez cada 30 segundos.*
  Este requerimiento genera complicaciones en el desarrollo del proyecto por parte del consumo de corriente. 30 segundos es una frecuencia alta que impone un consumo elevado dadas las características del GPS empleado. Se logra cumplir en el Modo Validación muestreando el GPS cada 10 segundos, lo cual permite conocer con gran certeza los desplazamientos del animal. Esta frecuencia de muestreo puede resultar bastante útil para distintos estudios que se quieran realizar. Más que nada tratándose de ovejas que tienden a moverse poco y en grupos grandes. Por otro lado el Modo Investigación se implementa para utilizar frecuencias menores de muestreo GPS (tiempos mayores 2 minutos) de modo tal de tener mayor autonomía.

- *Se deberán muestrear datos del acelerómetro al menos una vez por segundo.*
  Este requerimiento resulta trivial a medida que se define mejor la aplicación a desarrollar. Para hacer uso del algoritmo entrenado por aprendizaje au-

tomático se muestrea el acelerómetro a una velocidad de 100 muestras por segundo. Esto se requiere para operar bajo las mismas condiciones en las que el algoritmo fue entrenado y que la clasificación sea coherente con el Estado del animal. No hubo problemas en este aspecto, el acelerómetro empleado y los tiempos de ejecución de las tareas del software embebido no generaron limitantes.

- *A partir de los datos obtenidos se deberá identificar al menos un Estado de la oveja.*
  En inicios del proyecto, se quiso identificar solamente el Estado del animal pastando. El enfoque del proyecto cambió con la implementación de aprendizaje automático, optándose por identificar 4 Estados del animal. Con diferente nivel de éxito en estos Estados, los resultados fueron ampliamente satisfactorios en cuanto a la identificación de Estados. En base a los datos relevados se puede determinar con claridad cuando el animal está pastando. Y se tiene además amplio margen para mejorar e iterar, no es una solución fija.

- *La caracterización de Estados en campo debe presentar una probabilidad de error en dicha caracterización menor al 5 % con respecto a la probabilidad de aciertos teórica que tenga el algoritmo elegido. Se entiende por error el caso en que se caracteriza un Estado que no es coherente con la realidad.*

  Este requerimiento surge de la necesidad de validar que el algoritmo de aprendizaje automático sea efectivo en su caracterización. Se contrastan los resultados de caracterización de aplicar el algoritmo en pruebas de campo contra el algoritmo aplicado a la base de datos sobre la que se desarrolló. Los resultados son buenos, obteniéndose una probabilidad de error en la caracterización dentro del margen establecido. Con lo cual se puede validar que el algoritmo efectivamente logra caracterizar Estados a partir de los datos de aceleración relevados.

- *El Equipo deberá contar con un sistema de baterías que no presente riesgos que puedan causar daño físico para el animal, asegurando una autonomía de al menos una semana sin necesidad de intervención humana.*

  Se usaron baterías recargables de Litio-Ion tipo 18650, las cuales son suficientemente seguras para su uso en el Equipo diseñado. La autonomía de una semana se logra en el Modo Investigación mediante el empleo de una gran carga de baterías. Este es un aspecto a mejorar dado que se tiene un consumo más elevado de lo esperado, producto principalmente del hardware empleado. Se logra sin embargo llegar a un tiempo aceptable de autonomía, pero lejos de ideal.

- *El Equipo debe tener un peso inferior a* 600 g. *El encapsulado debe tener dimensiones inferiores a* 14 cm × 10 cm × 8 cm *(largo x ancho x altura).*

  Estos dos requerimientos puramente mecánicos fueron desafiantes para el diseño del encapsulado. Optando por impresión 3D, logramos cumplir el reque-

rimiento de tamaño satisfactoriamente y con un diseño estéticamente bueno. Se obtuvo como resultado un Equipo que tiene 125.0 mm × 117.5 mm × 67.65 mm de dimensiones Las baterías empleadas añaden bastante peso al Equipo, aún así se logra un peso aceptable de 465 g. Hubiese sido deseable obtener un Equipo más chico y liviano pero resulta imposible dado el hardware empleado. Mejoras en este aspecto dependen de miniaturizar el hardware, fabricando un circuito impreso de solo los componentes estrictamente necesarios. Esto escapa el alcance del proyecto planteado.

- *Las pruebas realizadas sobre el animal deben contar con la supervisión de un veterinario.*

  Se realizaron las pruebas con el cliente Rodolfo Ungerfeld que es un reconocido investigador en el área, con amplia experiencia en experimentación animal. También se evaluaron los requerimientos y aspectos del proyecto en forma conjunta. Se logra confirmar que el proyecto tiene validez y utilidad a fines de investigación veterinaria en ovinos.

- *La fabricación del prototipo final debe costar menos de 10.000 pesos uruguayos.*

  Se logró mantenerse por debajo del límite de costo establecido, con un costo total de $9181 pesos uruguayos. Lo cual es importante para que el Equipo tenga un potencial uso, tanto para trabajos de investigación como eventualmente producción. Para el prototipo se emplearon módulos proporcionados por la facultad, que facilitó su obtención pero genero costos elevados ya que la mayor parte del costo recae en los 3 componentes que se obtuvieron de la facultad. En el apéndice G se puede encontrar un análisis mas detallado de los costos del prototipo.

- *Se deberá desarrollar un Modo de Validación con el fin de utilizarse para validar el Sistema Completo y un Modo Investigación con el fin de utilizarse para hacer estudios en campo por personal capacitado.*

  Se pudieron desarrollar ambos modos con resultados satisfactorios. Contar con estos dos modos es algo que surgió durante el curso del proyecto y sirvió para definir bien las metas del proyecto. La necesidad de un Modo de Validación se hizo evidente cuando decidimos usar algoritmos de aprendizaje de máquinas, donde se buscó priorizar la posibilidad de re-entrenar el algoritmo y proveer al usuario toda la información posible. Esto es lo que permite que el dispositivo sea una herramienta de investigación y relevamiento de información sobre la cual se puede seguir trabajando.

8.3. Conclusión final

El Sistema desarrollado es un prototipo, una versión preliminar de la herramienta para la caracterización de comportamiento ovino que se buscó desarrollar. No se buscó desarrollar un producto final ni comercial. Como tal, los resultados

son satisfactorios, cumpliendo con el objetivo de caracterizar ciertos Estados de la oveja e incluso poder transmitir los datos relevados. Este aspecto da la posibilidad de proveer información útil para cualquier aplicación posterior y también seguir trabajando sobre el algoritmo de caracterización. El tiempo de vida del Equipo es suficiente para un entorno de investigación, en el cual se asume supervisión y fácil acceso al Equipo. Aún así, hay amplio margen para mejorar.

Se logró una buena implementación y estudio de distintas tecnologías que aportan versatilidad y escalabilidad. El protocolo MQTT resultó ser simple de implementar y se ajusta a las necesidades del proyecto. Se valora haber tenido la posibilidad de trabajar con la red celular Narrowband IoT de Antel, la cual sigue en despliegue. Apostar por estas nuevas tecnologías también rinde frutos a nivel de estudio y formación. Haber implementado algoritmos de aprendizaje automático da al proyecto una mayor versatilidad y posibilidad de trabajos futuros que no se pensó inicialmente. La caracterización lograda no es una solución única sino que se puede seguir iterando.

Por otro lado, haber desarrollado cada parte del Sistema no permitió optimizar cada una, habiendo aspectos para revisar y mejorar. El código del software embebido puede modificarse para reducir tareas bloqueantes que condicionan el muestreo. Principalmente, el hecho de reconectarse a la red MQTT cada vez que se quiere transmitir, lo cual también consume energía. El sistema de alimentación del Equipo necesita ajustes y se podría evaluar el uso de un módulo de comunicación y GPS distintos. Para esto se destaca que, aunque la electrónica sea distinta, la lógica de comandos AT empleada debería ser igual.

8.4. Trabajo futuro

El Sistema diseñado busca satisfacer la necesidad de tener una herramienta para el análisis y procesamiento de datos, en principio de ovejas, pero potencialmente para otros animales. Es por esto que se dejaron las bases de un sistema con ciertas funcionalidades esenciales pero que tiene muchos aspectos donde se puede expandir. Por ende, se procede a mencionar temáticamente los puntos que se consideran los próximos pasos en desarrollos ulteriores.

Hardware

Tanto el módulo de muestreo, procesamiento y comunicación como el de alimentación fueron armados empleando plataformas de desarrollo. No fueron efectuados diseños de miniaturización por lo que vale la pena tomar en consideración los componentes (periféricos, sensores y demás) involucrados en la aplicación particular y realizar una placa a medida tal que compacte la electrónica. Más aún, en vista de lo expuesto en el capítulo 7 de pruebas, se considera necesario otra propuesta para el módulo de alimentación. Resulta necesario tener ambos niveles de tensión, 5 V y 3.3 V si se pretende continuar con el resto de la electrónica intacta por lo cual podría partirse del diseño dado en [26].

Transmisión y recepción

Dada la implementación actual, resulta fácil escalar el Sistema para varias ovejas ya que el tópico al que publican sus datos identifica cada Equipo. No obstante lo anterior, sería deseable también poder recibir datos, por ejemplo de configuración. Es por esto que en el módulo MQTT del software embebido se debe de implementar una lógica de suscripción a los tópicos de interés. Se observa que dada la arquitectura de MQTT, no es posible alcanzar un dispositivo si éste no está suscrito al tópico, el cual sería el caso de los Equipos en contraposición al Sistema Central, por lo cual los cambios de configuración se darían cada cierto tiempo. En caso de que una aplicación futura no tolere lo mencionado, se deja planteada la necesidad de evaluar otros protocolos en capa de aplicación.

Por otra parte, como se menciona en la sección 5.8, el BG96 desactiva la conexión al MQTT cuando entra en modo PSM o se apaga. Se deberá abordar dicho problema si se desea disminuir el consumo del módem. Para esto se proponen 2 alternativas. La primer solución consiste en configurar el tiempo de *keep alive* de forma de que los períodos en los cuales no se genere intercambio de datos sean suficientemente largos para no perder la conexión sin limpiar la sesión (configurar el parámetro *clean session* a cero). La segunda alternativa que se plantea es el uso del protocolo MQTT-SN el cual tiene la ventaja de que no se necesita establecer una conexión previo a enviar mensajes. Pero tiene la contrapartida de que no existen garantías de que los paquetes lleguen a destino.

Sería recomendable también implementar un nuevo método de escape de datos. El que se implementó en este proyecto tiene la desventaja de generar un aumento de la carga del paquete a enviar en Modo Investigación en un 50 %. En 5.8 se propone un nuevo método, el cual logra disminuir considerablemente el aumento de carga útil ya que reduce el aumento a 1 byte solo en los casos problemáticos.

Procesamiento

En el ámbito del procesamiento de los datos, la aplicación particular y pruebas futuras indicarán los aspectos más importantes a considerar. A raíz de las pruebas, las cuales arrojaron resultados prometedores, se recomienda en primer lugar realizar una etapa de post procesamiento.

Al extraer información de 1500 datos y ser capaz de reportarla en un solo dato (el Estado del animal) resulta claro que la clasificación en el collar tiene un gran peso en términos de consumo y cadencia de transmisión. Sin embargo, utilizando información temporal, la cual no es considerada por el clasificador, podría mejorarse notoriamente la predicción del comportamiento. A modo de ejemplo, suponiendo que la oveja tiene períodos donde su comportamiento es muy marcado como ser al pastar, errores entre el Estado oveja quieta y oveja con cabeza agachada no afectarían ya que es claro lo que esta está haciendo en esa ventana de tiempo. Análogamente, si se tiene que el animal está corriendo y de manera intermitente aparecen datos de la oveja con la cabeza agachada, es claro que estos son un error y cual es la tendencia del comportamiento. La implementación más básica para incorporar la variable temporal sería el aplicar un filtro pasabajos ya que se

considera que las transiciones entre los comportamientos serán bastante continuas.

Por otra parte, sería recomendable entrenar nuevamente el algoritmo con datos recabados de más ovejas y explorar otras posibles variables que mejoren la exactitud del LDA. Como opción que surge de inmediato es analizar como utilizar la información dada por el giroscopio de la IMU para que mejore la precisión. Eventualmente, también se podría emplear el algoritmo del Análisis del Discriminante Cuadrático, donde la superficie de decisión es cuadrática tal como indica su nombre y por ello se pueden conseguir clasificadores más precisos.

Apéndice A

Esquemáticos y layouts de PCBs

A.1. PCB: Módulo muestreo y procesamiento

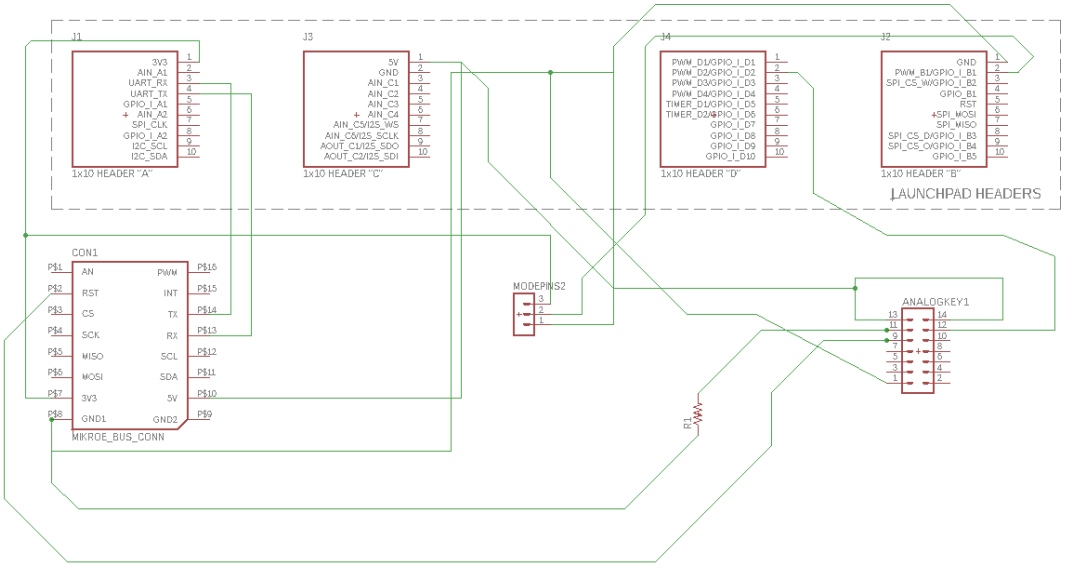

Figura A.1: Esquemático PCB de muestreo y procesamiento

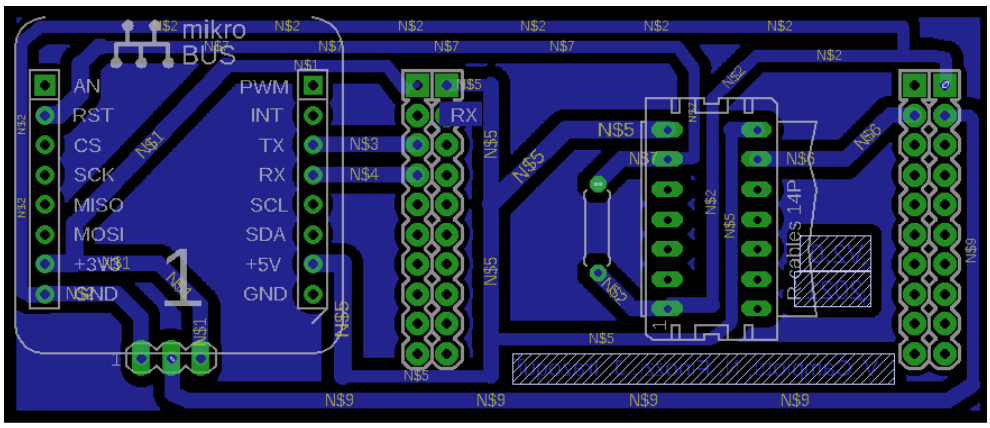

Figura A.2: Layout PCB de muestreo y procesamiento

A.2. PCB: Módulo de alimentación

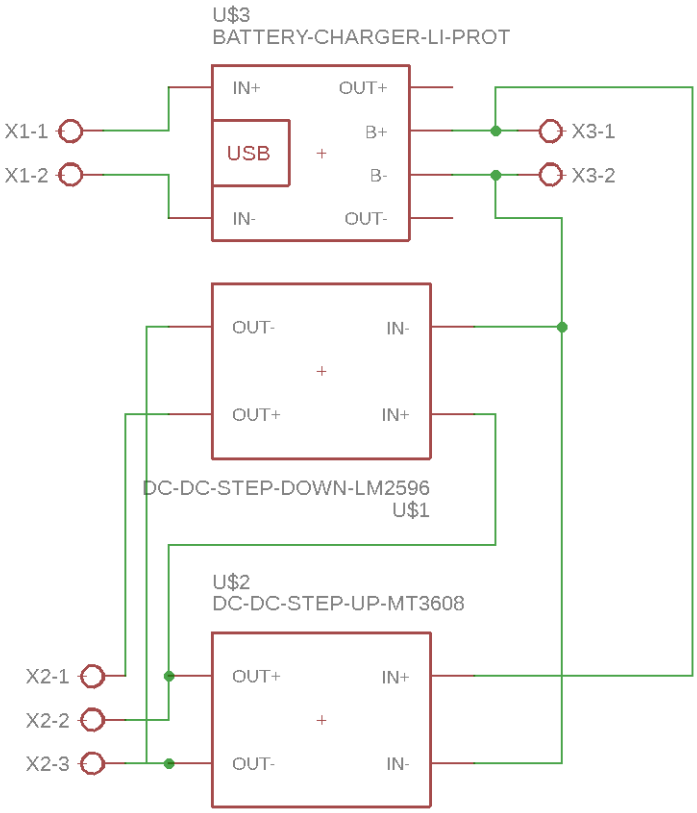

Figura A.3: Esquemático PCB de alimentación

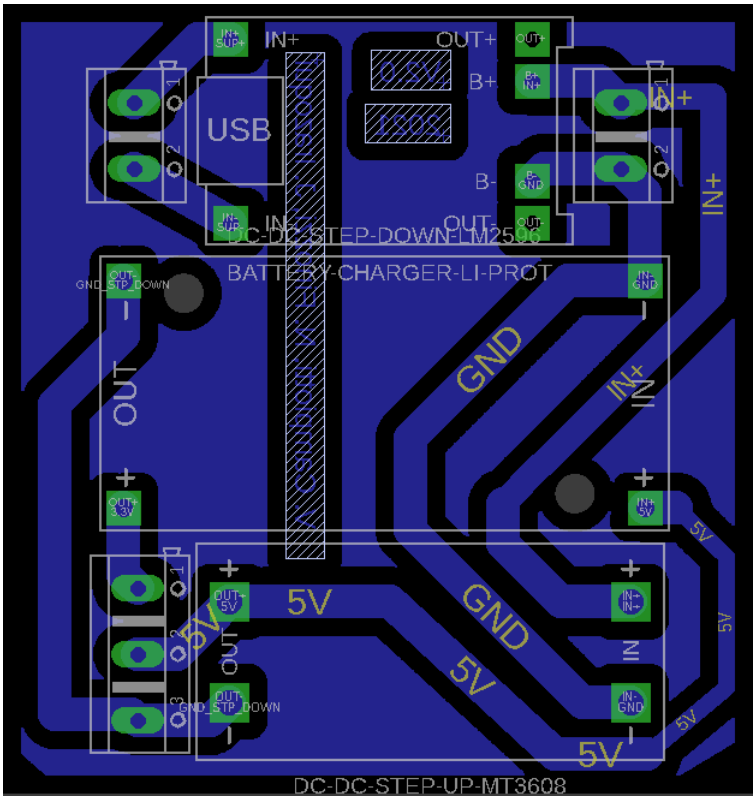

Figura A.4: Layout PCB de alimentación

Esta página ha sido intencionalmente dejada en blanco.

Apéndice B

Procesamiento y cálculos para la clasificación

B.1. Primer acercamiento: Filtro complementario

B.1.1. Cálculos: primer acercamiento

Cálculo de los ángulos del acelerómetro

El acelerómetro mide las aceleraciones de cada eje. Esto significa que si el acelerómetro esta quieto, la única aceleración que detectará es la gravedad. Es por eso que conociendo la dirección del vector de gravedad podemos detectar el ángulo de inclinación. Aplicando trigonometría y posibles movimientos en los 3 ángulos se llega a las siguientes expresiones.

$$\begin{cases} \Theta_x = atan\left(\frac{a_x}{\sqrt{a_y^2+a_z^2}}\right) \\ \Theta_y = atan\left(\frac{a_y}{\sqrt{a_x^2+a_z^2}}\right) \end{cases} \tag{B.1}$$

Cabe destacar que al estar calculando los ángulos a partir de la gravedad, es imposible calcular los movimientos en el eje Z. Para esto hace falta utilizar algún otro componente como es el magnetómetro.

Cálculo de los ángulos del giroscopio

El giroscopio nos da la velocidad angular por lo tanto para calcular el ángulo se deberá integrar la velocidad, se deberá conocer el ángulo inicial.

$$\begin{cases} \Theta_x = \Theta_{xo} + w_x\Delta t \\ \Theta_y = \Theta_{yo} + w_y\Delta t \\ \Theta_z = \Theta_{zo} + w_z\Delta t \end{cases} \tag{B.2}$$

La frecuencia de muestreo se configura para que sea $f = 100 \quad Hz$, obteniendo así un $\Delta t = 0{,}01s$. Es importante destacar que cuando hablamos de ángulo X en el

giroscopio nos referimos al giro en torno al eje X, como se puede apreciar en figura B.1.

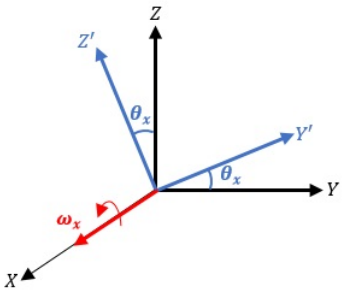

Figura B.1: Representación de los ángulos del giroscopio

Filtro complementario

El filtro está dado por la siguiente ecuación recursiva:

$$y[n] = a \cdot (y[n-1] + dato\_gyro \cdot \Delta t) + (1-a) ang\_accel \tag{B.3}$$

donde se definen:

- $y[n]$ es la enésima muestra.
- a es una constante dada entre 0 y 1 que define la ponderación y el efecto de filtros pasa-bajos y pasa-altos a la vez.
- $dato\_gyro$ es la velocidad angular dada por el giroscopio.
- Δt es el período de muestreo.
- ang_{accel} es el ángulo obtenido a partir de la medida del acelerómetro.

Se observa en la ecuación (B.3) que el filtro pondera el ángulo obtenido a partir de los datos del acelerómetro con aquel obtenido mediante el giroscopio. El ángulo a partir de los datos del giroscopio (ang_{gyro}) se obtiene con la aproximación discreta de la integral de la velocidad angular empleando como extremo inferior de integración el ángulo obtenido en la muestra anterior ($y[n-1]$):

$$ang_{gyro} = y[n-1] + dato\_gyro \cdot \Delta t \tag{B.4}$$

La expresión para el filtro surge de un simple estudio en frecuencia donde se superponen los efectos de los filtros pasa-bajo (sobre los datos del acelerómetro) y pasa-alto (sobre los datos del giroscopio) y se aplica la antitransformada Z.

La constante a de la ecuación (B.3) se elige a partir de la siguiente expresión:

$$a = \frac{\tau}{\tau + t_s} \tag{B.5}$$

donde se tiene:

- τ es la constante de tiempo elegida para tener la frecuencia de corte (f_c) correcta en los filtros pasa-bajo y pasa-alto que se superponen. $\tau = \frac{1}{f_c}$.
- t_s es la frecuencia de muestreo.

Para ver el desarrollo matemático completo es recomendado referirse al artículo "Filtro complementario para estimación de actitud aplicado al controlador embebido de un cuatrirrotor" [27].

B.1.2. Procesamiento: primer acercamiento

Para procesar los datos de la oveja se empleo Python. Más especificamente las librerías principales fueron *numpy* y *pandas*. Además, se optó por utilizar el formato de documento de *Jupyter Notebooks* el cual permite escribir fragmentos de código en distintos cuadros para ejecutarlos selectivamente. De esta manera se pueden aislar operaciones y ejecutarlas cuantas veces se desee de manera aislada sin tener que correr todo el código.

Se comienza cargando la base de datos y descartando las columnas que no sean la de etiquetas, aceleraciones en tres ejes y datos del giroscopio en tres ejes.

Luego, por una parte se crea una nueva columna donde se asigna el valor 1 si la oveja esta pastando (Estado que se supuso corresponde al animal con el cuello bajo) y 0 en cualquier otro. Por otra parte, se crea una columna donde a cada etiqueta se le asigna un número del 0 al 3 (los Estados de la base de datos son cinco pero trotar y correr se consideran como el mismo).

Posteriormente, se calcula el ángulo de interés a partir de las aceleraciones y por otra parte a partir del giroscopio operando como ya se indico en la sección anterior. La condición inicial para el cálculo del ángulo con datos de giroscopio se toma como el primer ángulo calculado con el acelerómetro.

Se procede aplicando el filtro complementario con un bucle *for* y se guarda en una columna de la base de datos para guardarlos en el ordenador. De esta manera todo el procesamiento a partir de este punto se puede hacer cargando el archivo con esta columna extra sin tener que realizar los cálculos nuevamente.

Luego, se separan los datos según el etiquetado de cabeza alta o baja y se eligen 10000 muestras de cada una para graficar. Emplear la totalidad de los datos en la gráfica puede conllevar a que el programa deje de correr.

Análogamente, se separan los datos según el etiquetado por clase para realizar otra grafica donde observar la dispersión de los puntos según la clase.

B.1.3. Implementación en el Equipo: Primer acercamiento

En primera instancia, se supuso que la la posición del cuello del animal fuera indicador suficiente para discriminar, por ejemplo, el Estado de pastando así como eventualmente otros. Siguiendo esta linea, se planteó una solución con ciertas hipótesis que simplificaran el problema para que, una vez se tuviera una primera versión funcional, se pudiera escalar para levantar las suposiciones. Dichas hipótesis iniciales fueron:

- El collar será siempre dispuesto de la misma manera de forma tal que el módulo con la IMU quede en la parte superior del cuello.
- El terreno sobre el que se encuentra la oveja será plano.
- El collar permanecerá fijo en el cuello del animal a lo largo del tiempo.
- El ángulo en el que se encuentra el cuello del animal con respecto a un plano vertical es representativo de su actividad.

El algoritmo utiliza en conjunto datos relevados por el giroscopio y acelerómetro de la IMU la cual se encontraría adosada a algún elemento que pudiera cambiar su ángulo en el plano vertical. El sistema luego podría discriminar entre dos Estados: ángulo alto y ángulo bajo.

Para la mejor comprensión de este acercamiento se definen los ángulos de navegación pitch, roll y yaw (ver Fig. [B.2]) con los cuales se caracteriza el ángulo del cuello de la oveja.

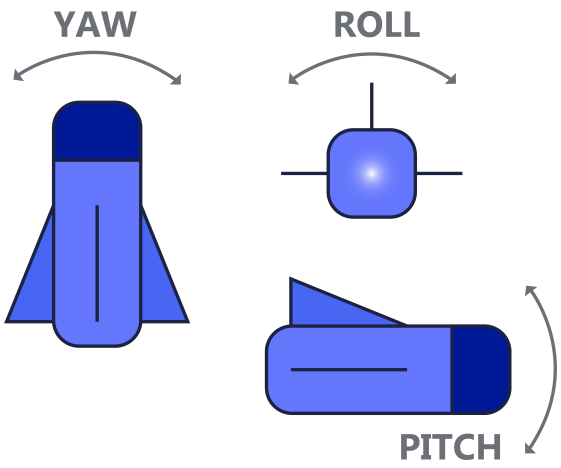

Figura B.2: Ilustración de los ángulos Yaw, Pitch y Roll.

El acelerómetro mide en tres ejes las aceleraciones ejercidas sobre la placa mientras que el giroscopio mide la velocidad de rotación en radianes por segundo. Las medidas son reportadas por comunicación serial cableada por protocolo I2C.

Se emplea una frecuencia de muestreo de 100 Hz asumiéndose suficiente para identificar los ángulos.

El dato del ángulo depende de las lecturas que hagan los sensores pudiendo ser afectadas por ciertos errores o interferencias. Por ejemplo, el acelerómetro puede detectar cualquier otra aceleración que no sea la gravedad la cual podría influir en la lectura de la medida. Por otro lado, para el cálculo del ángulo con el giroscopio, es inevitable que se genere un pequeño error y que se vaya acumulando a medida que se integra (ya que se obtiene la posición angular a partir del dato de la velocidad angular), generando notorias divergencias. A este error se lo conoce como *drift*. Es por esto que se requiere de un procesamiento para compensar esos defectos.

Por lo tanto, los datos relevados por ambos sensores son tratados por un filtro complementario como el de la ecuación B.3.

Para determinar el valor de a, se optó por ensayar empíricamente iterando qué valor daba un mejor desempeño. Resultó que el mejor filtrado se dio para valores en el entorno de $a = 0,93$.

Cabe mencionar que solo los ángulos de pitch y roll pueden ser tratados mediante el empleo de este filtro ya que el principio de funcionamiento de los acelerómetros no permiten la obtención de un ángulo yaw. Se recuerdan entonces las hipótesis sobre la fijación del collar de lo cual resulta que de todas maneras el ángulo yaw no afectaría en la clasificación de cuello alto o bajo.

En lo que respecta a la implementación del módulo se cuentan con tres sub bloques como indica la figura [B.3].

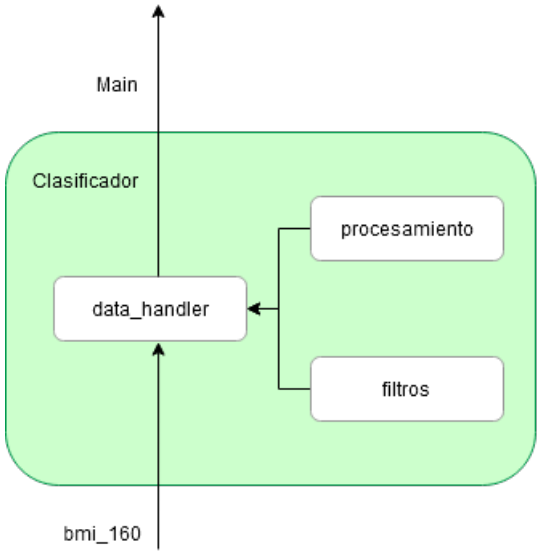

Figura B.3: Diagrama de módulos. Primera implementación del clasificador

El módulo *data_handler* (no debe ser confundido con el módulo del mismo nombre empleado en la implementación final) utiliza la librería de BOSCH del *bmi_160*, para obtener los datos crudos de los sensores y luego utiliza procesamiento y filtros para obtener el valor de los ángulos.

El módulo incluye tanto el filtrado como el procesamiento de datos para obtener los datos finales. Resulta de interés ver el pseudocódigo de la función gen_ang_data, definida en este módulo, encargada de generar los ángulos unificando todo. (Ver algoritmo 3)

En el módulo de procesamiento se encuentran las funciones que utilizan los datos crudos de los sensores y los convierten a datos de ángulos.

El BMI160 trabaja con 16 bits, esto genera un rango de datos de [-32768, 32768]. Por lo tanto, los datos crudos provenientes de los sensores se encuentran

Algoritmo 3: gen_ang_data: Generación de los ángulos a partir de los filtros y procesamiento e identificación del umbral que se establece para el cuello erguido o agachado.

Input: ***a***: coeficiente filtro complementario; ***thshld***: umbral de identificación de la posición del cuello. Dado por un *#define* al inicio del main para su cambio con sencillez.

Output: Ángulos luego del filtro complementario; Ángulos a partir del acelerómetro solo; Ángulos a partir del giroscopio solo; flag_posicion indicador de la posición del cuello de la oveja con respecto al umbral

Result: Mediante variables locales al módulo se obtienen los ángulos del giroscopio, acelerómetro y los ángulos del filtro complementario dependiente de ***a*** así como también la posición del cuello al comparar el ángulo absoluto contra ***thshld***. Las variables locales pueden ser luego devueltas al main por otras funciones del módulo

gen_ang_data(a, thshld){
obtener datos acelerómetro;
aceleraciones(datos acelerómetro);
ángulos acelerómetro(aceleraciones);
obtener datos giroscopio;
velocidades angulares(datos giroscopio);
diferenciales de ángulo giroscopio(velocidades angulares);
filtro complementario(ángulos acelerómetro, diferenciales de ángulo giroscopio);
flag_posicion = (ángulo absoluto > ***thshld***) ? 1 : 0;
}

dentro de ese rango. Por un lado, el acelerómetro se configura para que trabaje dentro del rango [-2g, 2g], 2 veces la aceleración gravitatoria. Para mapear los datos crudos al rango correspondiente se realiza la siguiente ecuación:

$$a_x = a_{xcrudo} \cdot \frac{2}{32768} = \frac{a_{xcrudo}}{16384} \tag{B.6}$$

Donde se define a_{xcrudo} como el dato que se obtiene del BMI160 sin ningún tipo de procesamiento y a_x como el dato mapeado al rango correspondiente.
El mismo razonamiento se aplica con el giroscopio que trabaja en un rango de [-2000 rad/s, 2000 rad/s]:

$$g_x = g_{xcrudo} \cdot \frac{2000}{32768} = \frac{g_{xcrudo}}{16{,}384} \tag{B.7}$$

En esta caso g_{xcrudo} corresponde al dato obtenido del BMI160 sin ningún procesamiento y g_x al dato escalado al rango de trabajo.

Finalmente, en el módulo de filtro se encuentran las funciones del filtro complementario. La descripción completa quedó dada anteriormente.

Las pruebas relacionadas a esta implementación con sus respectivos análisis quedan explicitadas en la sección 4.1.2.

B.2. Segundo acercamiento: Aprendizaje automático

El aprendizaje automático es una disciplina del ámbito de la inteligencia artificial que busca que las computadoras puedan aprender. Se entiende como aprender a una mejora en el desempeño de una resolución a un problema mientras más experiencia se adquiera. En un caso más concreto, si se tiene una base de datos con cierta información etiquetada (a lo cual se le llama aprendizaje supervisado) se espera que luego de un entrenamiento, pueda generarse un algoritmo tal que frente a nueva información pueda identificarse que etiqueta le corresponde. Mientras mayor sea la cantidad de información, mejores resultados se podrían lograr.

Para el algoritmo elegido, se aplicó el denominado análisis del discriminante lineal. Este presenta límites de decisión de carácter lineal para sus diferentes clases pudiéndose aprecia claramente en la Figura [B.4].

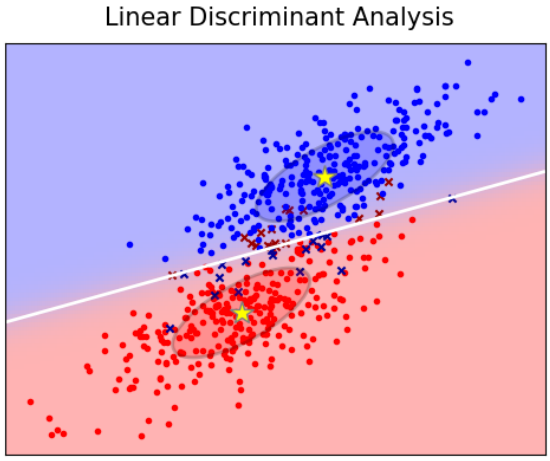

Figura B.4: Ejemplo de límite de decisión lineal para datos en dos dimensiones
Fuente: scikit-learn.org

El método empleado se denomina análisis discriminante lineal y busca asignar a un vector de atributos d-dimensional $x = \{x_1, ..., x_d\}$ una de las K clases que se consideran, aplicando un límite de decisión lineal. A su vez, el métodos sirve para reducir la dimensión del vector x hallando la proyección en un espacio de dimensión $K - 1$ o menor que maximice la variabilidad entre clases y minimice la variabilidad dentro de cada clase. Las hipótesis que el método toma son que las clases tienen una distribución gaussiana multivariante, lo cual cualitativamente es una generalización de la distribución gaussiana para dimensiones mayores a uno y que las clases posean la misma matriz de covarianza.

El clasificador se deriva directamente del modelo probabilístico que modela la probabilidad condicional de cada clase $P(X|y = k)$ para cada clase k. Utilizando la fórmula de Bayes para cada muestra del entrenamiento $x \in R^d$ resulta:

$$P(y = k|x) = \frac{P(x|y = k)P(y = k)}{P(x)} = \frac{P(x|y = k)P(y = k)}{\sum_l P(x|y = l)P(y = l)} \tag{B.8}$$

De aquí, que la clase a la que pertenece el vector de atributos en cuestión es la que presenta la mayor probabilidad.

De manera más específica, dada la hipótesis de que $P(x|y)$ se modela como una distribución Gaussiana multivariante se tiene que su densidad es:

$$P(x|y = k) = \frac{1}{(2\pi)^{d/2}|\Sigma_k|^{1/2}} exp\left(-\frac{1}{2}(x - \mu_k)^t \Sigma_k^{-1}(x - \mu_k)\right) \tag{B.9}$$

donde d es el número de atributos.

Luego, las Gaussianas para cada clase se suponen con la misma matriz de covarianza tal que $\Sigma_k = \Sigma$ para toda k. Esto reduce la ecuación (B.9) a:

$$log(P(y = k|x)) = -\frac{1}{2}(x - \mu_k)^t \Sigma^{-1}(x - \mu_k) + logP(y = k) + Cst \tag{B.10}$$

Luego, según como se indica en [28], resulta que la ecuación anterior puede expresarse como:

$$log(P(y = k|x)) = \omega_k \cdot x + \omega_{k0} + Cst \tag{B.11}$$

donde $\omega_k = \Sigma^{-1}\mu_k$ y $\omega_{k0} = -\frac{1}{2}\mu_k^t \Sigma^{-1}\mu_k + logP(y = k)$ y Cst no es necesario considerarlo ya que es el mismo para todas las clases (la clasificación se efectúa encontrando para cual es mayor la probabilidad).

Tanto ω_k como ω_{k0} son parámetros que pueden ser obtenidos con el empleo de la librería de Python sklearn.

Resulta claro de la expresión B.11 que la superficie de decisión del clasificador es lineal.

Apéndice C

Cálculos de consumo

Para la decisión de qué baterías utilizar junto a qué panel solar respetando los requerimientos planteados en el Capítulo [1] debió estimarse un consumo promedio.

Por consiguiente, primero se realizó el estudio del consumo utilizando los datos proporcionados por las hojas de datos correspondientes, luego se identificó cuanto sería el consumo según el modo de funcionamiento para 3.3 V y 5 V y finalmente, según las características de los conversores DC-DC, se estimó el consumo promedio de la batería para así poder dimensionarla.

Las siguientes secciones refieren al proceso recién descrito.

C.1. Datos para cálculo de corriente promedio consumida

Se reportan los consumos de los componentes del módulo de muestreo, procesamiento y comunicación, es decir, el microcontrolador, el BMI160, el módulo BG96, el GPS y la antena activa en la Tabla C.1.

Componente		**Consumo**
Microcontrolador		80 µA/MHz
BMI160		925 µA
BG96	Power Save Mode	10 µA @Real Network
	e-I-DRX	15 mA @20.48s, Real Network
	Active State	47-203 mA Data transfer@Real Network
GPS	Searching	42 mA @Lost state, Passive antenna
	Tracking	36 mA @Instrument Environment
Antena activa		8 mA

Tabla C.1: Consumos para cada componente

Deben realizarse entonces las siguientes observaciones. En primer lugar, el consumo del BG96 no contempla otros componentes de la placa de desarrollo LTE IoT 2 click. Se estima que la mayoría de consumo adicional que pueda presentar será ocasionado por los tres LEDs que posee. Para estimar dicho consumo se midió

la caída en la resistencia que poseen en serie y de allí se reportan los resultados en la Tabla C.2. Los nombres de los LED son según se indican en el esquemático provisto por Mikroe:

- TXD: LED que indica cuando se transmite. Se alimenta de 5V.
- STAT: LED que indica el estado de conexión de la red. Se alimenta de 5V.
- PWR: LED que indica si el módulo está alimentado. Se alimenta de 3.3V.

LED	**Voltaje**	**Resistencia**	**Consumo**
TXD	2.2 V	2.2 kΩ	1 mA
STAT	2.2 V	2.2 kΩ	1 mA
PWR	1.25 V	470 Ω	2.65 mA

Tabla C.2: Consumos por LED, módulo LTE IoT 2 click.

Luego, resulta que el estudio del consumo del GPS en actividad provisto por el fabricante es insuficiente para la aplicación aquí planteada. La antena que se emplea es activa y en este caso la hoja de datos dice TBD, es decir *to be determined*, del inglés: se debe determinar. De allí que para el dimensionamiento se utilizará para estimar los datos provistos en la Tabla C.1 sumándole el consumo particular de la antena empleada.

Se diferencian a continuación los componentes según su alimentación (Tabla C.3).

Alimentación de 5V	**Alimentación de 3.3V**
TXD LED	PWR LED
STAT LED	Microcontrolador
BG96 (GPS, Antena, TX)	BMI160
Step-Down	

Tabla C.3: Componentes según su alimentación.

Por otra parte, los componentes del módulo de alimentación influyen directamente en el consumo de las baterías dada la eficiencia que presentan los conversores y sus corrientes quiescentes. Los datos relevantes para efectuar los cálculos se reportan en la Tabla C.4.

Step-Up		**Step-Down**	
I_{qUp}	1.6 mA	I_{qDown}	5 mA
η_{Up}	93 %	η_{Down}	73 %
I_l	50 µA	I_{led}	1.67 mA

Tabla C.4: Datos de relevancia, módulo de alimentación.

Se aclara:

- I_{qUp}: Corriente quiesciente Step-Up.
- I_{qDown}: Corriente quiesciente Step-Down.
- η_{Up}: Eficiencia Step-Up.
- η_{Down}. Eficiencia Step-Down.
- I_l: Corriente de fuga.
- I_{led}: Corriente consumida por el LED en la placa del Step-Down.

C.2. Consumo promedio

A partir de los datos reportados en la sección anterior, junto a los modos de funcionamiento del Equipo, pueden estimarse los consumos promedio.

C.2.1. Modos de funcionamiento

El GPS tiene la característica de que demora aproximadamente 15 segundos en prender y posicionarse por lo que se divide el estudio en dos casos.

1. GPS cada 10 segundos. Esto implica que no se pueda apagar el módulo (A esta configuración se la denominará Modo Validación.
2. GPS con frecuencia de dos minutos. Esto permite el apagado del GPS entre la obtención de muestras (a esta configuración se la denominará Modo Investigación).

Se elige dos minutos como caso representativo de obtención de la posición con cierta coherencia con la aplicación. Resulta claro que el consumo para frecuencias que obtengan la posición cada un tiempo mayor disminuiría más.

En el primer caso, Modo Validación, se tiene un consumo permanente estimado de 44 mA por parte del GPS. En el segundo caso, Modo Investigación, durante un máximo de 15 segundos se consumirán 50 mA y luego 44 mA hasta que se apague el módulo.

Más aún, el Modo Investigación permite hacer uso de los modos de bajo consumo e-DRX y PSM en cada ciclo. La especificación con mayor profundidad de cada modo queda dada en el capítulo donde se describe el software embebido 5.

Considérese las notaciones de la tabla C.5.

Los modos se definen entonces de la siguiente manera

Validación

- f_{TX} de 5 segundos. Se desconoce el consumo en modo activo según la cantidad de datos por lo que se toma el promedio del rango dado en la tabla C.1.

TX	GPS
f_{TX}: frecuencia de transmisión del BG96	f_{GPS}: frecuencia de encendido del GPS
t_{on}: tiempo BG96 en modo connected	t_s: tiempo GPS en modo searching
t_{off}: tiempo BG96 apagado	t_t: tiempo GPS en modo tracking
t_{sby}: tiempo BG96 en standby por el e-I-DRX	

Tabla C.5: Nomenclatura utilizada

- $1/f_{GPS}$ de todo el período. A no confundirse con la frecuencia de obtención de dato de GPS que es de 10 segundos. Implica que el GPS se mantenga encendido siempre y a su vez el módulo LTE IoT 2 click.
- t_{on} de todo el ciclo.
- t_s no se considera ya que solo se da una vez al prender el dispositivo y en los reinicios.
- t_{off} no es posible en este modo.
- t_t de todo el ciclo.
- t_{sby} no es posible en este modo.

Investigación:

- f_{TX} de 2 minutos.
- f_{GPS} de 2 minutos.
- t_{on} se estima un peor caso de 30 segundos. Incluye la conexión a la red MQTT y el envío de los datos.
- t_s se estima de 15 segundos.
- t_{off} se estima de 1 minuto y medio.
- t_t se estima de 5 segundos.
- t_{sby} tampoco es posible en este modo. Aquí resulta innecesario al suponer que las transmisiones son tales que se oscile entre modo activo y apagado en todo momento.

C.2.2. Cálculos

Considérense las siguientes definiciones:

- Io_{Down}: Corriente a la salida del Step-Down.
- Ii_{Down}: Corriente a la entrada del Step-Down (sin contar la quiesciente).

- Io_{Up}: Corriente a la salida del Step-Up.
- Ii_{Up}: Corriente a la entrada del Step-Up (sin contar la quiesciente).
- I_{5V}: Corriente consumida por los componentes a 5 V (sin contar el consumo del Step-Down en cascada).
- $I_{3,3V}$: Corriente consumida por los componentes a 3.3 V.

Entonces puede definirse la eficiencia de los conversores como:

$$\eta_k = Po_k/Pi_k = \frac{Vo_k \cdot Io_k}{Vi_k \cdot Ii_k} \tag{C.1}$$

Comenzando con la operativa:

$$Io_{Down} = I_{PWRLED} + I_{micro@48MHz} + I_{BMI160} = 7.415\,\text{mA} \tag{C.2}$$

A partir de las ecuaciones (C.1) y (C.2) se obtiene:

$$Ii_{Down} + I_{qDown} = \frac{Vo_{Down} \cdot Io_{Down}}{Vi_{Down} \cdot \eta_{Down}} + I_{qDown} = 11.7\,\text{mA}$$

Por otra parte:

$$\begin{aligned} Io_{Up} &= Ii_{Down} + Iq_{Down} + I_{LED} + I_{TX} + I_{GPS} + I_{TXDLED} + I_{STATLED} \\ Io_{Up} &= I_{TX} + I_{GPS} + 20{,}37mA \end{aligned} \tag{C.3}$$

Donde los consumos producto de transmitir y del GPS en un ciclo pueden calcularse respectivamente como:

$$\bar{I_{TX}} = \frac{I_{on} \cdot t_{on} + I_{off} \cdot t_{off}}{1/f_{TX}} \qquad \bar{I_{GPS}} = \frac{I_t \cdot t_t + I_s \cdot t_s}{1/f_{GPS}} \tag{C.4}$$

Según el modo se obtienen los resultados de la tabla C.6.

Modo		**Consumo**
Validación	GPS	44 mA
	TX	125 mA
Investigación	GPS	6 mA
	TX	31.25 mA

Tabla C.6: Consumo de GPS y transmisión según el modo

Se tiene entonces:

$$\begin{aligned} Io_{UpValidacion} &= 189.37\,\text{mA} \\ Io_{UpInvestigacion} &= 57.62\,\text{mA} \end{aligned} \tag{C.5}$$

Luego, resulta que el consumo de las baterías será I_B dado por:

$I_{BValidacion}$	$I_{BInvestigacion}$
276 mA	85.15 mA

Tabla C.7: Consumo de GPS y transmisión según el modo

$$\begin{aligned} I_B &= Ii_{Up} + I_{qUp} = \frac{Vo_{Up} \cdot Io_{Up}}{Vi_{Up} \cdot \eta_{Up}} + I_{qUp} \\ I_B &= 1{,}45 \cdot Io_{Up} + 1.6\,\text{mA} \end{aligned} \tag{C.6}$$

Finalmente resulta:

La capacidad de la batería según requerimientos era deseable que fuera de una semana en Modo Investigación sin intervención humana. Sin embargo el consumo semanal para ese modo de funcionamiento es de:

$$Consumo \times semana = I_{investigacion} \cdot 24\,\text{h} \cdot 7\,\text{d} \tag{C.7}$$

$$Consumo \times semana = 14\,300\,\text{mA h}$$

Suponiendo se emplearan dos baterías 18650 de 4000 mA h aún se requeriría una fuente adicional de 6300 mA h. Por una parte se decide flexibilizar el requerimiento del consumo para el dimensionamiento según lo conversado con el veterinario Rodolfo Ungerfeld a una autonomía de cuatro días resultando en:

$$Consumo \times cuatro\,dias = 8174\,\text{mA h}$$

En estas condiciones aun se requieren de 174 mA h haciendo necesaria la incorporación de un panel solar.

Elección del panel solar

La energía consumida a diario necesaria se obtiene de la siguiente manera:

$$Energia\,consumida = corriente\,por\,hora \times voltaje\,bateria \tag{C.8}$$

$$Energia\,consumida = 174\,\text{mA h} \times 3.7\,\text{V} = 645.28\,\text{mW h}$$

Los fabricantes de paneles proporcionan la potencia nominal del panel, esta potencia corresponde a la potencia máxima que es capaz de entregar el panel en las horas picos de sol (HSP). En Uruguay se puede considerar que la cantidad de HSP es de 3 horas.

$$Potencia\,del\,panel = \frac{Energia\,consumida}{HSP} \tag{C.9}$$

$$\begin{aligned} Potencia\,del\,panel &= \frac{645.28\,\text{mW h}}{3HSP} \\ Potencia\,del\,panel &= 0.215\,\text{W} \end{aligned}$$

En estas condiciones resulta que un panel de 1 W de potencia es suficiente e incluso potencialmente prolonga la autonomía un poco más de cuatro días.

Se recuerda que el sistema fue sobre dimensionado para todo el análisis omitiendo por ejemplo la entrada en modo de bajo consumo del microcontrolador y asumiendo que durante el tiempo que el BG96 se encuentra activo la mitad está transmitiendo. Por lo que en estas condiciones se espera una duración un poco mayor a los cuatro días.

Consumo en vacío y consumo con LTE IoT 2 click apagado

Puede resultar de interés calcular el consumo en vacío del sistema de alimentación para tener una noción de la mínima corriente que se tendrá presente siempre en la aplicación.

Considérense la ecuación C.1 y la tabla C.4. Resulta entonces que la corriente promedio consumida a las baterías puede calcularse como:

$$I_B = \frac{V_{OUp} \cdot (I_l + I_{led} + I_{qDown})}{V_{iUp} \cdot \eta_{Up}} + I_{qUp} = 11.36\,\text{mA} \tag{C.10}$$

Por otra parte, otro dato de interés es el consumo del sistema cuando el módulo LTE IoT 2 click está conectado pero se encuentra apagado. En estas condiciones el consumo principal que presentará es el del LED de PWR. Resulta que el consumo promedio a las baterías puede calcularse como:

$$I_B = \frac{V_{OUp} \cdot (I_l + I_{led} + I_{iDown} + I_{qDown})}{V_{iUp} \cdot \eta_{Up}} + I_{qUp} = 11.36\,\text{mA}$$

Donde I_{iDown} se obtiene como:

$$I_{iDown} = \frac{V_{ODown} \cdot I_{PWR}}{V_{iDown} \cdot \eta_{Down}} = 2.4\,\text{V}$$

Finalmente se tiene:

$$I_B = 14.84\,\text{mA} \tag{C.11}$$

Esta página ha sido intencionalmente dejada en blanco.

Apéndice D

Alimentación del MSP432P401R

La placa fue diseñada para incorporar distintos métodos de alimentación. La más común es a través de la placa XDS110 conectada mediante el puerto USB. Se alimenta con 5 V y ese voltaje se regula a 3.3 V para el funcionamiento general del microcontrolador. Alternativamente, puede alimentarse de forma externa, sin utilizar la placa XDS110. La placa tiene un rango de operación de 1.62 V a 3.7 V. La imagen [D.1] representa a la izquierda el método de alimentación a través de la placa XDS110 y la derecha el método de alimentación externa.

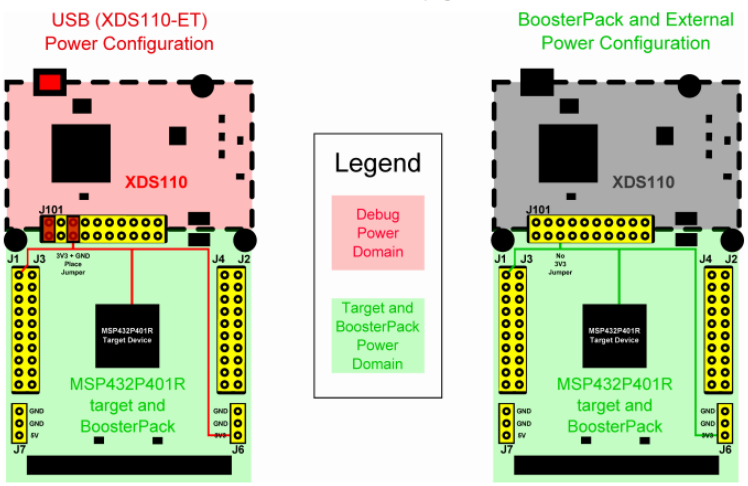

Figure 10. MSP-EXP432P401R Power Block Diagram

Figura D.1: Diagramas de alimentación del MSP432P401R
Fuente: https://www.ti.com/

Esta página ha sido intencionalmente dejada en blanco.

Apéndice E

Comandos AT

La secuencia de comandos para la conexión Narrow Band de Antel junto con el ejemplo de conexión al servidor MQTT de Antel que se presentan a continuación se obtuvo de la pagina de Antel del Kit IoT [15].

E.1. Conexión Narrowband de Antel

Se debe enviar la siguiente secuencia de comandos en orden. En paréntesis hay un pequeño comentario del comando y en gris se representa la respuesta esperada.

AT (comando de testeo)
OK

AT+CPIN? (verifico si la sim no tiene pin)
+CPIN: READY

AT+CFUN=4 (deshabilito transmisión y recepción de datos)
OK

AT+QCFG="nwscanmode",3,1 (habilito a que el modem solo busque conexiones LTE)
OK

AT+QCFG="iotopmode",1,1 (especifico a redes LTE Cat NB1)
OK

AT+CGDCONT= 1,"IP","testnbiot","0.0.0.0",0,0 (creo contexto PDP e ingreso APN de la red NB de ANTEL)
OK

AT+CFUN=1 (restablezco las funcionalidades del modem)
OK

AT+COPS=1,2,“74801”,9 (especifico manualmente que se conecte al operador ANTEL con red NB)
OK

AT+CEREG? (verifico que se haya establecido la conexión a la red)
+CEREG: 0,1

AT+QNWINFO (pregunto por el tipo de red)
+QNWINFO: “CAT-NB1”,“74801”,“LTE BAND 3”,1290

AT+CGACT=1,1 (Activo contexto PDP 1 a la red APN)
OK

AT+CGPADDR=1 (Pregunto por IP la asignada por al contexto PDP 1)
+CGPADDR: 1,167.108.195.136

E.2. Conexión al servidor MQTT de Antel

El ejemplo a continuación muestra como conectarse al servidor MQTT de Antel, enviar un mensaje y desconectar la conexión. En paréntesis hay un pequeño comentario del comando y en gris se representa la respuesta esperada.

AT+QMTOPEN=0,”kitiot.antel.com.uy”,1883 (Abre red MQTT con la URL y puerto ingresados)
+QMTOPEN: 0,0

AT+QMTCONN=0,”nodo”,”kitiot”,”micontraseña”(Ingreso usuario y contraseña de la red MQTT)
+QMTCONN: 0,0,0
OK

AT+QMTPUB=0,0,0,0,”kitiot/holaMundo”(publico un mensaje al tópico kitiot/holaMundo, 0-QoS)
>

(Luego de obtener la respuesta anterior podemos ingresar nuestro mensaje, para finalizar y publicar el mensaje hay que escribir el carácter “Ctrl+Z”que en hexadecimal es ”1a”)
+QMTPUB: 0,0,0
OK

AT+QMTDISC=0 (cerramos conexión)
+QMTDISC: 0,0

Apéndice F

Instructivo de uso del Sistema Central

En esta sección, se explica con mayor profundidad cómo trabajar con el Sistema Central. El Sistema Central consta de 3 partes. Una base de datos donde se guardan todos los datos recibidos, un script de JavaScript que se suscribe a los tópicos y guarda los datos en la base de datos y finalmente un programa en Python para la visualización de datos en tiempo real.

F.1. Base de datos: Postgres

Como primer requisito para el funcionamiento del Sistema Central se deberá tener instalado el programa Postgres, el cual se encuentra disponible para descargar en la página oficial [29] junto con un instructivo de su instalación. Durante la instalación, se deberá crear un **usuario** y una **contraseña** que el usuario debe tener siempre presente.

Se le proveerá al usuario un archivo Sistema_Central.zip con todos los archivos necesarios. Si no es la primera vez que se utiliza el programa se puede saltar a la sección F.2.2.

F.2. Inicialización

F.2.1. Crear base de datos de cero

Deberá descomprimir el archivo .zip y ejecutar el archivo llamado ScriptPrincipal. Ese archivo abrirá una consola que se encargará de obtener cierta información relevante para el funcionamiento del Sistema Central. En primer lugar deberá informar si es la primera vez que se está ejecutando el programa ya que de ser así se necesitará crear la base de datos. En rojo se representan los datos que el usuario debe ingresar.

"Starting shell script..."

First time? [y / n] y

Luego se le pedirá que ingrese el usuario y la contraseña. El usuario y la contraseña deben ser las que se crearon cuando se instaló el Postgres (F.1).

Enter user: usuario
Enter password: contraseña

Finalmente se le pedirá que elija un nombre para la base de datos. Si la base de datos no existe se creará desde cero. En el caso de que ya existiera una base de datos con ese nombre se eliminará y se creará una nueva con ese nombre.

dbname: nombre para la db

F.2.2. Base de datos ya fue creada

En el caso de que el programa ya haya sido ejecutado, y ya existe una base de datos activa, el intercambio de información con la consola va a ser la siguiente

"Starting shell script..."
First time? [y / n] n

F.3. Ejecución de los programa

Luego de la inicialización se abrirán 2 consolas automáticamente. Una de ellas va a estar ejecutando el script de JavaScript que va a estar suscripto a todos los tópicos necesarios y cuando reciba datos los va a estar guardando en la base de datos creada en la sección (inicialización). Solo mientras esa consola esté corriendo los datos recibidos van a estar siendo almacenados. La otra consola es la encargada de ejecutar el programa Python para desplegar los datos en una interfaz gráfica. Ambas consolas funcionan de manera independiente, si alguna de las 2 se cierra solo dejará de funcionar la parte asociada a esa consola. Por ejemplo si se cierra la consola que ejecuta el Python, se dejará de ver la interfaz gráfica pero los datos seguirán siendo almacenados en la base de datos. Si se cierra la consola que ejecuta el JavaScript ya no se estarán recibiendo datos pero se podrá ver en la interfaz todos los datos obtenidos hasta el momento que se dejó de ejecutar el JavaScript.

F.4. Interfaz gráfica

La interfaz gráfica consiste en un conjunto de pantallas, en el cual se puede navegar a través de distintos botones.

Primero encontramos la pantalla principal, con 3 botones. Según el botón que se presiona a que pantalla nos deriva. El primer botón Datos Acelerómetro nos lleva a una ventana en la cual se grafican los datos provenientes de los 3 ejes del acelerómetro. Junto con un boton Inicio que nos permite volver a la pantalla principal.

El segundo botón GPS, nos lleva a una ventana donde se grafican las posiciones geográficas, en base a los valores de longitud y latitud. Incluye también el botón Inicio que nos permite volver a la pantalla principal.

Finalmente el tercer botón Estados, nos lleva a una pantalla donde se grafican los Estados clasificados en función del tiempo. Incluye también el botón de Inicio para volver a la pantalla principal.

Figura F.1: Pantalla Principal de la Interfaz Gráfica

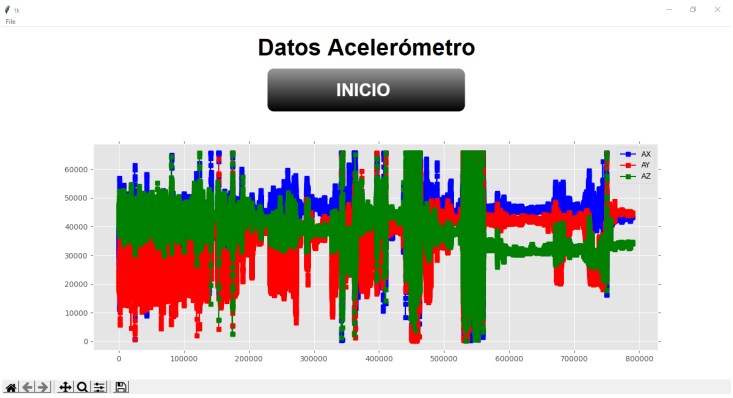

Figura F.2: Pantalla Datos del acelerómetro de la Interfaz Gráfica

Se incluye una funcionalidad para poder exportar los datos en formato CSV. Para poder exportar los datos se deberá presionar *File* > *Save*.

Una vez presionado *Save* deberá saltar una ventana como la imagen [F.6]. Esto significa que los datos fueron exportados y guardados correctamente dentro de la carpeta *Sist_Central/Interfaz/data*. Esos archivos pueden ser manipulados por el usuario sin ningún inconveniente.

Figura F.3: Pantalla Datos del GPS de la Interfaz Gráfica

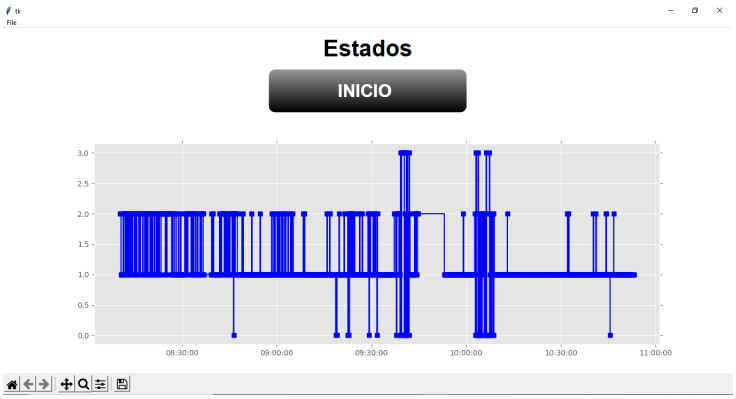

Figura F.4: Pantalla Datos del Estados de la Interfaz Gráfica

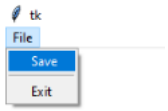

Figura F.5: Botón textitSave para exportat datos

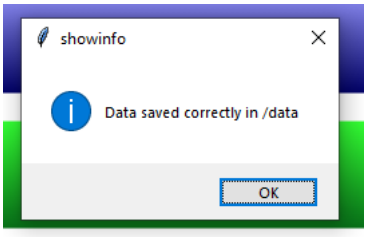

Figura F.6: Datos Exportados

Apéndice G

Análisis de costos

Una vez construido el dispositivo completo, se realizó un análisis sobre sus costos. Se presentan los costos más detallados en las tablas [G.1] y [G.2].

El dispositivo en su completitud (sin incluir mano de obra) tuvo un costo de USD 211,1. Si bien dentro de los requerimientos, está dentro de los costos aceptables se identifican algunos puntos importantes a destacar.

Los gastos de este proyecto recae en su mayoría en estos 3 componentes:

- Microprocesador MSP-EXP432P401R
- LTE IoT 2 click
- BOOSTXL-SENSORS

Entre la suma de estos 3 componentes hay un gasto aproximado de U$D100. Estos 3 componentes no fueron comprados, sino que fueron provistos por el Instituto de Ingeniería Eléctrica. Debido a la emergencia sanitaria se priorizo trabajar con componentes que ya teníamos a disposición. Haciendo un análisis más profundo de mercado y de usabilidad de los componentes se considera un gasto que se podría reducir ampliamente.

Por un lado el microcontrolador utilizado se encuentra en un Launchpad que contiene muchas más funcionalidades de las que se utilizan en este proyecto y cuesta U$D 20. Por ejemplo el debugger que tiene el launchpad, aparte de no ser utilizado le agrega un costo a la placa innecesario. Por lo tanto, como trabajo a futuro se podría crear nuestra propia placa que incluya como base el mismo micro, logrando reducir su costo a la mitad dado que el micro solo cuesta alrededor de U$D 10.

Por otro lado, el BOOSTXL-SENSORS utilizado cuesta alrededor de U$D 25. Un costo algo elevado para utilizar solamente la IMU. Existe en el mercado IMUs de mucho menor costo con características similares que pueden ajustarse perfectamente al proyecto, con costos menores a U$D 10.

Por último el LTE IoT 2 click es el componente más caro de los que se utilizan. Con un precio aproximado de U$D 50. La base del LTE IoT 2 click es el módulo BG96 de Quectel. Se encontró como una posible alternativa que Quectel cuenta

con un Mini PCIe [30] que cuesta alrededor de U$D 20 y cuenta con las mismas funcionalidades que el LTE IoT 2 click. El único inconveniente es que no incluye las antenas. Pero debido a que igualmente se tiene que comprar una antena activa para el proyecto, la compra de una antena pasiva adicional cuesta menos de U$D 10. Reduciendo el costo total del LTE IoT 2 click casi a la mitad.

El resto de los componentes utilizados fueron comparados en Uruguay. Se llegó a la conclusión de que de haber importado el resto de los componentes el costo de los componentes podría reducirse un 76 %. Además si se le suma el factor de compras al por mayor estos precios pueden reducirse aún más.

Se concluye que aplicando las medidas mencionadas anteriormente se podría llegar a alcanzar el precio ideal establecido al comienzo del proyecto.

Tipo de cambio:	43,51

Prototipo Tesis					
	Componente	Unidades	Provedores	Costo USD	Coso $
módulos	msp432p401r	1	Texas Instrument*(1)	19,9	865,849
	boosterxl-sensor	1	Texas Instrument*(1)	24,99	1087,3149
	lte iot 2 click	1	Mikroe*(1)	51,75	2251,6425
Subtotal				96,64	4204,8064
componentes	antena	1	*(1)	17,25	750
	hcf4066b	1	Eneka	0,391	17
	t04056	1	Eneka	2,783	121
	step up	1	Eneka	3,335	145
	step down	1	Eneka	4,6	200
	panel solar	1	Eneka	7,59	330
construcción PCB	Zócalo de 10 pines	4	Eneka	3,22	140
	Headers 10 pines	4	Eneka	3,68	160
	Bornera 2 terminales	2	Eneka	0,69	30
	Bornera 3 terminales	1	Eneka	0,621	27
	Separadores PCB	2	Eneka	2,76	120
	Portapilas 18650	1	Eneka	3,45	150
	Pila Li-Ion 6800mAh	2	Eneka	11,155	485
	Switch on/off	1	Eneka	0,805	35
	Estaño (1 metro)	1	Eneka	1,242	54
	Placa Cobre	2	Eneka	2,3	100
Subtotal				65,872	2864
construcción caja mecanica	impresión caja	1	Ial arquitectura	36,8	1600
	impresión tapa	1	Ial arquitectura	10,12	440
	cinta	1	Casa greco	0,805	35
	pasador	3	Casa greco	0,184	8
	hebilla	3	Casa greco	0,69	30
Subtotal				48,599	2113
Total				211,111	9181,8064
				U$D	$

Tabla G.1: Tabla costos prototipo
*(1)Provisto por el instinto de ingeniería eléctrica

Tomando Medidas					
	Componente	Unidades	Provedores	Costo USD	Coso $
módulos	msp432p401r	1	Texas Instrument*(1)	19,9	865,849
	boosterxl-sensor	1	Texas Instrument*(1)	24,99	1087,3149
	lte iot 2 click	1	Mikroe*(1)	51,75	2251,6425
Subtotal				96,64	4204,8064
componentes	antena	1	Gnssmall store *(2)	4,95	215,3745
	hcf4066b	1	Ljsix store *(2)	0,35	15,2285
	t04056	1	Great it electronic *(2)	0,246	10,70346
	step up	1	Great it electronic *(2)	0,48	20,8848
	step down	1	Great it electronic *(2)	0,92	40,0292
	panel solar	1	Great it electronic *(2)	1,44	62,6544
construcción PCB	Zócalo de 10 pines	4	Shop2885226 Store *(2)	0,38	16,5338
	Headers 10 pines	4	Shop2885226 Store *(2)	0,224	9,74624
	Bornera 2 terminales	2	Shop2885226 Store *(2)	0,1	4,351
	Bornera 3 terminales	1	Shop2885226 Store *(2)	0,085	3,69835
	Separadores PCB	2	Hihigh Store *(2)	0,1075	4,677325
	Portapilas 18650	1	XIA011 MI Store *(2)	0,01	0,4351
	Pila Li-Ion 9900mAh	1	Golden 7direct factory Store *(2)	3,25	141,4075
	Switch on/off	1	Shop2885226 Store *(2)	0,742	32,28442
	Estaño (1 metro)	1	Great it electronic *(2)	0,8	34,808
	Placa Cobre	2	Great it electronic *(2)	1,67	72,6617
Subtotal				15,7545	685,478295
construcción caja mecanica	impresión caja	1	Ial arquitectura	36,8	1600
	impresión tapa	1	Ial arquitectura	10,12	440
	cinta	1	Casa greco	0,805	35
	pasador	3	Casa greco	0,184	8
	hebilla	3	Casa greco	0,69	30
Subtotal				48,599	2113
Total				160,9935	7003,284695
				U$D	$

Tabla G.2: Tabla costos prototipo aplicando algunas medidas
*(1)Provisto por el instinto de ingeniería eléctrica
*(2) Obtenido en AliExpress

Referencias

[1] DIEA, "Producción animal: ganadería vacuna y lanar." *Anuario Estadístico Agropecuario*, 2020. [En linea]. Disponible en: https://descargas.mgap.gub.uy/DIEA/Anuarios/Anuario2020/ANUARIO2020.pdf

[2] R. Cardellino. (2015) La producción ovina en Uruguay. Accedido: 24/06/2017. [En linea]. Disponible en: fhttp://www.dohnetresarboles.com.uy/newsletters/bob/agronomia2015.pd

[3] J. J. Mari, "Pérdidas perinatales en corderos," 1eras. Jornadas Ovinas Veterinarias 1-12, Tacuarembó, Uruguay, 1979.

[4] J. Olivera-Muzante, "¿Es posible mejorar la supervivencia de corderos en nuestros sistemas ovinos?" *Revista Cangüé digital*, vol. 36, no. 15-17, 2015.

[5] Diario "El Pueblo", Salto, Uruguay. (2017) Javier Otero, gerente del SUL: el producto bruto nos indica que es mejor negocio el ovino que el vacuno. Accedido: 13/07/2021. [En linea]. Disponible en: https://diarioelpueblo.com.uy/javier-otero-gerente-del-sul-el-producto-bruto-nos-indica-que-es-mejor-negocio-el-ovino-que-el-vacuno/

[6] M. Azzarini, "Una propuesta para mejorar los procreos ovinos." *Boletín de difusión SUL*, no. 3-35, 2000.

[7] R. Ungerfeld, "Comunicación personal," entrevista con Julián Oreggioni, Febrero 2020.

[8] E. Fogarty, D. Swaina, G. Croninb, and M. Trottera, "Autonomous on-animal sensors in sheep research: A systematic review," *Computers and Electronics in Agriculture*, vol. 150, pp. 245–256, July 2018.

[9] P. Castro-Lisboa, J. Oreggioni, and E. Machado, "System and device for monitoring the reproductive activity of animals," Mar. 2020, US Patent 10,575,501.

[10] Insprovet, España. (2020) Avisador de partos Moocall. Accedido: 13-07-2021. [En linea]. Disponible en: http://www.insprovet.es/moocall/

[11] Digitanimal. (2020) Localizador GPS para animales y ganado. Accedido: 14-07-2021. [En linea]. Disponible en: https://digitanimal.co/

[12] Afimilk. (2020) Cow monitoring solutions. Accedido: 14-07-2021. [En linea]. Disponible en: https://www.afimilk.com/cow-monitoring

[13] N. Acosta, N. Barreto, P. Caitano, R. Marichal, M. Pedemonte, and J. Oreggioni, "Research platform for cattle virtual fences," in *2020 IEEE International Conference on Industrial Technology (ICIT)*, 2020, pp. 797–802.

[14] V. Campiotti, N. Finozzi, and J. D. Irazoqui, "DEO: Dinámica espacial de una oveja," 2020, Proyecto Final del curso "Sistemas Embebidos para tiempo real", Udelar.

[15] Antel. (2020) Kit IoT Antel. Accedido: 14-07-2021. [En linea]. Disponible en: https://kitiot.antel.com.uy/

[16] M. Chen, Y. Miao, M. Yiming, and Y. HaoKai Hwang, "Narrow band internet of things," *IEEE Access*, pp. 20 557 – 20 577, September 2017. [En linea]. Disponible en: https://www.researchgate.net/publication/319869218_Narrow_Band_Internet_of_Things

[17] F. Michelinakis, A. S. Al-selwi, A. S. Capuzzo, A. Zanella, K. Mahmood, and A. Elmokashf, "Dissecting energy consumption of NB-IoT devices empirically," *IEEE Internet of Things Journal*, pp. 1224 – 1242, January 2020. [En linea]. Disponible en: https://ieeexplore.ieee.org/document/9159902

[18] MQTT. (2021) The standard for IoT messaging. Accedido: 14-07-2021. [En linea]. Disponible en: https://mqtt.org/

[19] E. Rastogi, N. Saxena, A. Royc, and D. R. Shinb, "Narrowband internet of things: A comprehensive study," *Computer Networks*, vol. 173, p. 107209, May 2020. [En linea]. Disponible en: https://www.sciencedirect.com/science/article/abs/pii/S1389128619313593?via%3Dihub

[20] J. Chen, K. Hu, Q. Wang, Y. Sun, Z. Shi, and S. He, "Narrow-band internet of things: Implementations and applications," *IEEE Internet of Things Journal*, pp. 2309 – 2314, December 2017. [En linea]. Disponible en: https://ieeexplore.ieee.org/abstract/document/8076876

[21] J. W. Kamminga, H. C. Bisby, D. V. Le, N. Meratnia, and P. J. M. Havinga, "Generic online animal activity recognition on collar tags," in *Proceedings of the 2017 ACM International Joint Conference on Pervasive and Ubiquitous Computing and Proceedings of the 2017 ACM International Symposium on Wearable Computers*, ser. UbiComp '17, September 2017, p. 597–606. [En linea]. Disponible en: https://doi.org/10.1145/3123024.3124407

[22] S. P. Le Roux, J. Marias, R. Wolhuter, and T. Niesler, "Animal-borne behaviour classification for sheep (dohne merino) and rhinoceros (ceratotherium simum and diceros bicornis)," *Animal Biotelemetry*, vol. 5, no. 25, November 2017. [En linea]. Disponible en: https://animalbiotelemetry.biomedcentral.com/articles/10.1186/s40317-017-0140-0

[23] V. Campiotti, N. Finozzi, and J. Irazoqui. (2020) Demostración Proyecto Final del curso "Sistemas embebidos para tiempo real", Udelar. [En linea]. Disponible en: https://www.youtube.com/watch?v=cEKwzG_4UJs

[24] QOITECH. (2021) Otii Arc. Accedido: 14-07-2021. [En linea]. Disponible en: https://www.qoitech.com/otii/

[25] Quectel, "Quectel BG96 power consumption report V1.0 important," 2017. [En linea]. Disponible en: https://www.quectel.com/product/lte-bg96-cat-m1-nb1-egprs/

[26] Ceech. (2016) Power path manager and battery charger with 5 V and 3.3 V output. Accedido: 14-07-2021. [En linea]. Disponible en: https://www.openhardware.io/view/291/Power-path-manager-and-battery-charger-with-5V-and-33V-output#tabs-instructions

[27] J. Redolfi, D. Gaydou, and H. Agustin, "Filtro complementario para estimación de actitud aplicado al controlador embebido de un cuatrirrotor," in *Congreso Argentino de Sistemas Embebidos*, Marzo 2011, p. 124. [En linea]. Disponible en: https://www.researchgate.net/publication/275033545_Filtro_complementario_para_estimacion_de_actitud_aplicado_al_controlador_embebido_de_un_cuatrirrotor

[28] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern classification*, 2nd ed. Wiley, 2001.

[29] The PostgreSQL Global Development Group. (2021) PostgreSQL: The world's most advanced open source relational database. Accedido: 14-07-2021. [En linea]. Disponible en: https://www.postgresql.org/

[30] Quectel. (2021) Quectel BG96 mini PCIe. Accedido: 14-07-2021. [En linea]. Disponible en: https://www.quectel.com/wp-content/uploads/pdfupload/Quectel_BG96_Mini_PCIe_LPWA_Specification_V1.0.pdf

Esta página ha sido intencionalmente dejada en blanco.

Índice de tablas

Índice de figuras

Esta es la última página.
Compilado el miércoles 6 octubre, 2021.
http://iie.fing.edu.uy/