TESIS DE MAESTRÍA EN INGENIERÍA

DESARROLLO DE UN DISPOSITIVO DE SENSORES INTEGRADOS PARA EL ESTUDIO DEL COMPORTAMIENTO ANIMAL

Ing. Andrés Oliva Trevisan
Maestrando

Dra. Karina Laneri
Director

Ing. Nicolás Catalano
Co-director

Miembros del Jurado
Dr. Luis Horacio Arnaldi (IB-CAB-CNEA)
Dr. Fabricio Pablo Alcalde Bessia (IB-CAB-CONICET)
Dr. Eduardo Luis Blotta (Universidad Nacional de Mar del Plata)

Junio de 2023

Física Estadística e Interdisciplinaria–Ingeniería en Telecomunicaciones

Instituto Balseiro
Universidad Nacional de Cuyo
Comisión Nacional de Energía Atómica
Argentina

A todos aquellos
que buscan un sueño
cuando se les dice
que no les corresponde

Resumen

El monitoreo del comportamiento de la tortuga terrestre *Chelonoidis chilensis* en su hábitat natural, es esencial para recopilar información sobre su movimiento y elaborar directrices para la conservación de la especie, debido a que actualmente se encuentra en estado vulnerable. En esta Tesis, se presenta el desarrollo de una familia de dispositivos de bajo costo y bajo consumo de energía, formada por un dispositivo de monitoreo que se coloca sobre el animal, una estación colectora y un rastreador de radio frecuencia. Sus diseños fueron elaborados de forma de ser fácilmente adaptables para el monitoreo de otras especies animales en otros contextos. Cada dispositivo de la familia está compuesto por un transceptor compatible con protocolos de internet de las cosas (IoT) en la banda de frecuencia Sub-1 GHz, un receptor de sistema de navegación por satélite global (GNSS), un magnetómetro, así como sensores de temperatura e inercia. El dispositivo no supera el 5 % del peso del animal para evitar perturbar su comportamiento. El peso de la placa de circuito impreso, junto a la batería y el receptor GNSS, es de 44,9 g y sus dimensiones son de 48,7 mm x 63,7 mm. La autonomía que puede variar entre una semana y un mes, dependiendo de las tasas de muestreo de los sensores, la tasa de la señal de radio frecuencia y la del receptor GNSS. La placa fue diseñada para funcionar como un dispositivo de monitoreo, una estación de recopilación de datos y un rastreador, mediante la adición de pequeñas piezas de hardware. Se presenta aquí el diseño del circuito electrónico y del firmware asociado, pensados para permitir la extensión de las funciones del dispositivo. Se realizaron mediciones en el laboratorio y en el campo para evaluar la autonomía y el alcance del enlace de radio frecuencia, así como el consumo de energía y el error de posicionamiento asociado. Se informan esos valores y discuten las limitaciones y ventajas del dispositivo, publicando este desarrollo abierto para su uso por parte de otros grupos de investigación que trabajan en proyectos similares.

Índice de contenidos

Índice de contenidos v

Índice de acrónimos ix

Índice de figuras xi

Índice de tablas xv

Organización de la tesis 1

1. Introducción 3
- 1.1. Monitoreo del movimiento animal y su importancia . . . 3
- 1.2. Relevamiento y trabajo en campo . . . 5
  - 1.2.1. Relevamiento de campo . . . 5
  - 1.2.2. Impacto del trabajo . . . 7
- 1.3. Solución propuesta . . . 11
  - 1.3.1. Ventajas . . . 12
  - 1.3.2. Consideraciones durante el desarrollo del proyecto . . . 12

2. Especificaciones de diseño 15
- 2.1. Especificaciones generales del proyecto . . . 15
- 2.2. Diagrama de bloques y restricciones . . . 16
  - 2.2.1. Restricciones de diseño . . . 18
- 2.3. Especificaciones de *hardware* . . . 18
  - 2.3.1. Especificaciones de *firmware* . . . 20

3. Diseño de *hardware* 23
- 3.1. Selección de microcontrolador . . . 23
  - 3.1.1. Especificaciones del SoC MCU . . . 24
  - 3.1.2. Relevamiento del mercado . . . 25
  - 3.1.3. Comparación entre SoC de ST y Texas Instruments . . . 26
  - 3.1.4. SoC CC1312R de Texas Instruments . . . 28

3.2. Selección de otros componentes . . . 30
3.2.1. Sensores inerciales y magnetómetro . . . 30
3.2.2. Receptor GNSS . . . 32
3.2.3. Memoria . . . 32
3.2.4. Batería . . . 34
3.2.5. Regulador de tensión . . . 35
3.2.6. Cargador de batería . . . 35
3.2.7. Switch digital . . . 35
3.2.8. Sensor de luz . . . 35
3.2.9. Micrófono . . . 36
3.2.10. Selección de antenas . . . 36
3.2.11. Módulo Bluetooth . . . 37
3.3. Diseño del PCB . . . 37
3.3.1. Diseño de circuito esquemático . . . 37
3.3.2. Esquemático . . . 39
3.3.3. Posicionamiento de módulos . . . 39
3.3.4. Diseño de PCB . . . 39
3.3.5. Elementos para la fabricación de cada dispositivo . . . 42
3.4. Estimación de alcance . . . 43

4. Diseño de *firmware* 45
4.1. Diagrama de bloques . . . 45
4.2. Características y elementos del *firmware* . . . 45
4.2.1. Criterios de diseño . . . 45
4.2.2. Sistema operativo en tiempo real . . . 47
4.2.3. Interacción entre tareas del RTOS . . . 48
4.2.4. Máquina de estados . . . 49
4.2.5. Modos de funcionamiento asociados al consumo . . . 52
4.2.6. Estructura de memoria y almacenamiento . . . 53
4.2.7. Interfaces al usuario . . . 55
4.3. Características de las tareas del RTOS . . . 56
4.3.1. Estructura general de tareas de adquisición . . . 57
4.3.2. Estructura general de tareas no periódicas . . . 60
4.4. Tareas del sistema operativo . . . 61
4.4.1. Tarea *System monitor* . . . 64
4.4.2. Tarea PWM leds . . . 65
4.4.3. Tarea IMU ACCEL/GYRO . . . 65
4.4.4. Tarea IMU MAG . . . 66
4.4.5. Tarea Temperatura . . . 66

4.4.6. Tarea Receptor GNSS 66
4.4.7. Tarea SD Store 70
4.4.8. Tarea de Print UART 71
4.4.9. Tarea RF data send 73
4.4.10. Tarea RF KA (*Keep Alive pulse*) 75
4.4.11. Tarea actividad animal 76
4.5. Estimación de consumo y autonomía 77
4.5.1. Estimación de consumo 77
4.5.2. Estimación de autonomía 77
4.6. Decodificación de datos 78
4.7. Tareas de la DSC y el TD 80

5. Resultados **83**
5.1. Fabricación 83
5.1.1. Circuito impreso 83
5.1.2. Montaje de componentes 83
5.1.3. Costos de fabricación del MD 85
5.2. Ensayos en laboratorio 85
5.2.1. Consumo, autonomía y memoria de almacenamiento 85
5.2.2. Caracterización de radiofrecuencia 87
5.2.3. Caracterización de antena del MD 88
5.2.4. Alcance 91
5.3. Mediciones en campo 94
5.3.1. Autonomía y tiempo de carga de batería 94
5.3.2. Alcance 94
5.3.3. Sensores de temperatura 96
5.3.4. Precisión en GNSS 96
5.3.5. Tiempo medio de *fix* en receptor GNSS 98
5.3.6. Algoritmo de detección de movimiento animal 99
5.4. Publicación de los resultados 101

6. Conclusiones y trabajo futuro **103**

Bibliografía **107**

Agradecimientos **111**

Índice de acrónimos

ADC Analog-to-Digital Converter (Conversor Analógico Digital) 37, 45, 47, 62, 64

BGA Ball-Grid Array 27

DCS Data Collector Station (Estación Colectora) 11, 12, 18, 20, 25, 27, 36, 42, 43, 55, 56, 68, 73, 74, 80, 81, 83, 91, 92

DSSS Direct Sequence Spread Spectrum (Espectro Ensanchado por Secuencia Directa) 24, 74

GNSS Global Navigation Satellite System (Sistema global de navegación por satélite) 4, 6, 7, 11, 19, 32, 33, 35, 37, 41, 42, 48, 50, 53, 56, 64, 66–71, 74, 77, 78, 81, 83, 86, 96–98

GPIO General Purpose Input/Output (Entrada/Salida de Propósito General) 42, 47, 67, 68

GPS Global Positioning System (Sistema de Posicionamiento Global) 32, 96, 97

HDOP Horizontal Dilution of Precisión (Dilución de la Precisión Horizontal) 67, 68, 74, 97

I2C Inter-Integrated Circuit 25, 39, 45, 57, 59, 62, 65

I2S Inter-IC Sound 25, 30, 36, 39, 45

IC Integrated Circuit (Circuito Integrado) 23, 35

IMU Inertial Measurement Unit (Unidad de medición inercial) 30–32, 39, 54, 65, 66, 77, 96

IoT Internet of Things (Internet de las cosas) 19, 24–26, 28, 30

MCU Microcontroller Unit (Microcontrolador) 19, 21, 24, 25, 31, 32, 35, 37, 38, 47, 48, 54, 66, 75, 78, 87, 96

MD Monitoring Device (Dispositivo de Monitoreo) 11, 12, 15, 16, 18, 19, 23, 25, 27, 34, 36, 42, 45, 51, 55, 56, 74, 75, 80, 81, 83, 85–88, 90–92, 94, 96–98, 101

NMEA National Marine Electronics Association 66, 67

PC Personal Computer (Computadora Personal) 11, 18, 43, 56

PCB Printed Circuit Board (Placa de Circuito Impreso) 18–20, 23, 32, 37, 39, 41–43, 83, 101

RF Radio Frecuency (Radio Frecuencia) 6, 8, 11, 16, 17, 19, 20, 24–27, 42, 47, 65, 66, 73–75, 80, 81, 85, 88, 96

RSSI Received Signal Strength Indicator (Indicador de Nivel de Señal) 20, 73, 80, 88, 92, 94

RTOS Real Time Operative System (Sistema Operativo en Tiempo Real) 20, 25, 47–50, 54–57, 59–64, 71, 73–75, 77, 80

SMD Surface-Mount Device (Dispositivo de Montaje en Superficie.) 39

SoC System on Chip (Sistema en Chip) 23–28, 30, 46, 48, 55, 66

SPI Serial Peripheral Interface (Interfaz Periférica Serial) 25, 45, 47, 60–62

TD Tracking Device (Dispositivo de Seguimiento) 11, 12, 18, 20, 30, 32, 36, 37, 42, 43, 52, 55, 75, 80, 81, 83

UART Universal Asynchronous Receiver-Transmitter (Transmisor-Receptor asíncrono universal) 19, 25, 37, 45, 47, 56, 62, 66–68, 70, 73, 80, 81

USB Universal Serial Bus (Bus Universal en Serie) 64

UTC Coordinated Universal Time (Tiempo universal coordinado) 74

Índice de figuras

1.1. Región habitada por la especie de tortuga terrestre *C. chilensis* en Argentina junto a la ubicación de la zona donde se realizaron los estudios de campo. . . . 5
1.2. Método del transmisor. . . . 8
1.3. Elementos utilizados en el método de los hilos. . . . 8
1.4. Método de monitoreo con dispositivo prototipo. . . . 8
1.5. Diagrama comparando en forma cualitativa los diferentes métodos y la información que proveen. . . . 9
1.6. Imagen del hábitat de la tortuga *C. chilensis*. La vegetación del terreno consta de pastizales de entre 20 y 40 cm de altura y arbustos o arboles de hasta 2 metros. Además en el terreno se encuentran diferentes tipos de depresiones y desniveles. . . . 10
1.7. Familia de dispositivos propuesta interactuando entre si: dispositivo de monitoreo (MD), dispositivo rastreador (TD) y estación colectora (DCS). 12

2.1. Diagrama de bloques donde se muestran los diferentes módulos interactuando entre sí. . . . 17

3.1. Diagrama de bloques y prestaciones del microcontrolador CC1312R de Texas Instruments . . . 29
3.2. Batería tipo LiPo de 3.7 V y 600 mAh. . . . 34
3.3. Antenas a utilizar para los dispositivos DCS (a) y TD (b). . . . 36
3.4. Filtros diseñados para módulo receptor GNSS. . . . 38
3.5. Divisor diseñado (dentro de rectangulo rojo) para implementar la medición de la tensión de la batería en forma segura. . . . 38
3.6. Esquemático de la familia de dispositivos de monitoreo animal. . . . 40
3.7. Imagen ilustrativa de posicionamiento de los diferentes módulos en el PCB. . . . 41
3.8. Diseño del PCB de la familia de dispositivos a utilizar. . . . 42
3.9. Modelo 3D del PCB donde se pueden observar los diferentes elementos del diseño. . . . 43

4.1. Diagrama de bloques donde se muestran los diferentes módulos interactuando entre sí y los protocolos utilizados para comunicarse con cada periférico. 46
4.2. Interacción entre los diferentes elementos del *firmware* utilizando un RTOS y máquina de estados. 49
4.3. Máquina de estados del MD. 51
4.4. Curva de batería (tensión en el eje de ordenadas, porcentaje de energía en eje de abscisas) tipo LiPO de 3.7 V y 600 mAH, relevada experimentalmente. 52
4.5. Elementos de la estructura de datos utilizada para almacenar los valores adquiridos. 53
4.6. Código de *header* de estructura de memoria. 54
4.7. Ejemplo de estructura de memoria para las muestras obtenidas con el magnetómetro. 55
4.8. Estructura general de las tareas de adquisición utilizada en este proyecto. 58
4.9. Parte principal del código para tarea de adquisición de magnetómetro. . 60
4.10. Estructura general de tareas aperiódicas y su interacción con una tarea periódica. 62
4.11. Diagrama reducido de la función encargada de recolectar y analizar los mensajes recibidos por el módulo GNSS. 69
4.12. Estructura propuesta para la tarea aperiódica print UART. 72
4.13. Estructura correspondiente a la tarea de comunicación RF. 75
4.14. Contenido de archivo generado por MD al almacenar los datos adquiridos en una memoria microSD. 78
4.15. Ejemplo de log de dispositivo DCS, donde se observan los mensajes recibidos por RF de dos MD. 81
4.16. Ejemplo de mensaje completo enviado por MD y recibido por DCS. . . 81

5.1. PCB diseñado en el Capítulo 3 fabricado por la firma Mayer. 84
5.2. Versión implementada del MD. 84
5.3. Valores recibidos (en dBm) para una transmisión de un tono único en diferentes valores de frecuencia (Hz). 89
5.4. Valores de potencia (dB) recibidos por el analizador de espectro para transmisiones realizadas por el dispositivo MD a diferentes valores de frecuencia (MHz). 89
5.5. Potencia recibida (en dB) para cable curvo (rombos naranja) y cable recto (cuadrados azules) en función de largo del cable (en metros), realizando transmisiones en iguales condiciones. 91
5.6. Imagen de antena tipo monopolo dentro del campus del Instituto Balseiro. 93

5.7. Estación receptora colocada en campo con antena monopolo de 2 metros de altura. 95
5.8. Valores de temperatura adquiridos de la IMU (rojo) y el MCU (azul) del MD. 97
5.9. Trayectoria construida a partir de muestras de geolocalización adquiridas con el MD (rojo) y con el receptor GPS de mano Garmin (azul), junto a los valores de HDOP para cada punto. 97
5.10. Valor medio de longitud/latitud obtenidos con el MD (rojo) y con el receptor GPS de mano Garmin (azul), junto a los valores de desviación estándar obtenidos para la medición de cada coordenada. 98
5.11. Histograma de tiempo de *fix* para receptor GNSS de MD luego de 48 horas de actividad en campo. 99
5.12. Ejemplo de señal de acelerómetro adquirida de una tortuga silvestre, donde los ejes X, Y, Z corresponden a los colores rojo, azul y verde respectivamente. 100
5.13. Resultado de clasificación de actividad animal para la señal del acelerómetro. 100

Índice de tablas

3.1. Matriz de comparación donde se observan en forma resumida las principales características de los diferentes SoCs evaluados. 26
3.2. Matriz adicional de comparación con un resumen de las características de RF de los diferentes SoCs evaluados. 27
3.3. Comparación de parámetros de interés entre SoCs de Texas Instruments y ST. 28
3.4. Integrados presentados como combinaciones de acelerómetro (A), giróscopo (G) y magnetómetro (M). 31
3.5. Opciones de selección de receptor GNSS. 32
3.6. Elementos asociados a cada tipo de dispositivo de acuerdo a sus funcionalidades. 43

4.1. Tareas del sistema, listando parámetros de interés cómo P. (prioridad), tiempo de ejecución, de vencimiento y periodo en segundos, valor de stack y tipo de tarea. 63
4.2. Períodos y *bytes* a escribir por de las tareas periódicas que llaman a la tarea aperiódica de escritura en microSD, junto al tiempo de ejecución y factor de utilización asociado. 64
4.3. Consumo estimado para cada tarea. 78
4.4. Autonomía estimada para diferentes modos de funcionamiento empleando una batería de 600 mAH. 79

5.1. Consumo de las diferentes tareas y sensores medidos en el MD. 86
5.2. Valores de autonomía calculados a partir de los valores obtenidos de las mediciones en laboratorio para una batería de 600 mAh. 87
5.3. Valores recibidos de potencia para diferentes combinaciones de transmisores y receptores. 90
5.4. Resultados de pruebas de alcance realizadas en el campus del Instituto Balseiro. 92
5.5. Valores máximos de alcance y RSSI para los ensayos realizados en campo. 96

Organización de la Tesis

En el Capítulo 1 se presentará el contexto en el cual se realiza esta Tesis, introduciendo las metodologías utilizadas por el grupo de investigación, empleadas en el estudio de tortugas terrestres y explicando, las limitaciones de las mismas junto a las necesidades del grupo. Luego se introducirá la solución propuesta, consistiendo ésta en el diseño de una familia de dispositivos para el monitoreo animal.

En el Capítulo 2 se presentarán las especificaciones de diseño del *hardware* y *firmware* de la familia de dispositivos a partir de lo expuesto en el capítulo anterior.

En el Capítulo 3 se detallaran, a partir de las especificaciones mencionadas en el Capítulo 2, los componentes seleccionados para la implementación del *hardware* junto a los diseños del esquemático y placa de circuito impreso.

En el Capítulo 4 se detallarán los criterios de diseño del *firmware* obtenidos a través de las especificaciones mencionadas en el Capítulo 2, para luego explicar la forma en la que el mismo fue implementado.

En el Capítulo 5 se mostrarán los resultados obtenidos al realizar la implementación de los dispositivos, contrastando las estimaciones realizadas en forma teórica con los resultados obtenidos en las pruebas realizadas en el campus del Instituto Balseiro y en la región donde se encuentran las tortugas terrestres estudiadas.

Finalmente en el Capítulo 6 se mencionarán las conclusiones obtenidas a partir de los resultados expuestos en el capítulo anterior junto a las futuras mejoras propuestas.

Capítulo 1

Introducción

Esta Tesis de Maestría presenta la especificación, desarrollo, construcción y caracterización de un dispositivo de bajo costo y bajo consumo de energía, diseñado para el monitoreo del movimiento animal. La información recabada por estos dispositivos permitirá ayudar a determinar patrones de movimiento y comportamiento del animal de estudio.

Para este fin, el dispositivo de monitoreo contará con una serie de sensores que serán utilizados para extraer información tal como posición, temperatura y aceleración, que se utilizará para clasificar diferentes comportamientos animales como caminar, alimentarse o copular. En las secciones posteriores nos ocuparemos de ahondar en las características técnicas que deberá satisfacer esta familia de dispositivos.

Antes de entrar en los detalles técnicos del proyecto, primero se introducirá el contexto para el cual se desarrollará esta solución, junto al relevamiento y selección de los componentes electrónicos.

1.1. Monitoreo del movimiento animal y su importancia

Los patrones de movimiento animal son el resultado de la interacción entre los recursos ambientales y las necesidades energéticas de un individuo, junto con la interacción con conespecíficos e individuos de otras especies [1, 2]. Las primeras observaciones fueron realizadas por investigadores en el campo, a veces con la consecuencia de alterar el comportamiento de los animales, pero más recientemente comenzaron a utilizarse sistemas automatizados, con sensores adheridos a los individuos o bien desplegados en el campo [3, 4]. En este sentido, se han desarrollado varios dispositivos para monitorear el movimiento animal [3, 5]. Por ejemplo, para monitorear el movimiento, área visitada, migración e interacciones de tortugas box orientales (*Terrapene carolina*) en el noroeste de Ohio (EE.UU.), se ensamblaron receptores de sistema global de navegación por

satélite (GNSS) de bajo costo basados en tecnología Arduino [6]. Otro sistema utilizado en tortugas en entornos semi naturales, se basó en un dispositivo programable de bajo consumo energético con un sistema de comunicación de radio junto con acelerómetros y receptor GNSS. Este sistema permitió a los investigadores detectar la actividad de excavación de nidos de tres especies de tortugas mediterráneas: (*Testudo hermanni, Testudo graeca y Testudo marginata*), transcribiendo las coordenadas geográficas al detectar ciertos patrones característicos en el acelerómetro [7, 8]. Esta información permitió localizar los huevos de las tortugas y protegerlos de los depredadores [8]. Los dispositivos de monitoreo tienen muchos beneficios para estudiar la ecología y conservación de los animales [4]. Sin embargo, en algunos países como Argentina, el acceso a dispositivos de monitoreo comerciales es a menudo demasiado costoso. Además, estos dispositivos no suelen ser versátiles, en el sentido de que no siempre es posible hacer modificaciones de *software* o *hardware* para adaptarlos a necesidades particulares.

En este sentido, este proyecto fue motivado por la necesidad de desarrollar una tecnología propia, de bajo costo y versátil para el monitoreo de la tortuga terrestre *Chelonoidis chilensis*, siendo la tortuga continental más austral del mundo [9], cuya historia natural es poco conocida [10–13]. Esta especie habita las regiones del Chaco seco, llanuras y mesetas de Monte [13, 14] de Bolivia, y desde el oeste de Paraguay hasta el norte de la Provincia de Chubut (figura 1.1), en Argentina [11]. Su población actualmente se ve seriamente afectada por el avance de la frontera agrícola en el Chaco Seco y por la extensa cría de cabras y ganado en sectores del sur, además es el reptil más afectado por el mercado ilegal de mascotas en Argentina [15]. Por esas razones, esta tortuga fue categorizada como vulnerable a nivel nacional [15], así como internacionalmente [16].

Es importante mencionar que los huevos de las poblaciones más australes tienen un período de incubación muy largo bajo tierra variando de 10 a 16 meses, que junto con la reciente introducción de ganado y jabalíes, hacen que estas poblaciones sean aún más vulnerables [15].

En particular esta especie es la tortuga continental más austral del mundo y en esta Tesis se desarrollará una familia de dispositivos que serán utilizados en el monitoreo de ejemplares de la población en el extremo sur de su distribución geográfica (figura 1.1). Puntualmente en esta latitud, las tortugas tienen un periodo de baja actividad entre los meses de mayo y octubre, denominado brumación, por lo cuál su actividad se concentra en los meses de noviembre a marzo. Durante los meses de noviembre y diciembre ocurre el apareamiento, luego entre enero y marzo las hembras buscan sitio donde depositar sus huevos y entre marzo y abril comienza la búsqueda de los sitios donde brumaran durante los meses de mayo a octubre, cuando la temperatura disminuye considerablemente. En este sentido es importante recordar que las tortugas son reptiles y por lo tanto la temperatura del ambiente determina su temperatura corporal

(ectotermos), condicionando significativamente el grado de actividad del animal.

Con esta motivación, el desarrollo presentado en esta Tesis fue orientado y diseñado para el estudio de la tortuga *C. chilensis*. No obstante, todas las especificaciones y desarrollos de los dispositivos (criterios de diseño, circuito esquemáticos, códigos, etc.) fueron diseñados con una filosofía de código abierto, pensados de modo que otros grupos de investigación puedan adaptar este desarrollo a sus necesidades, según la especie animal a monitorear.

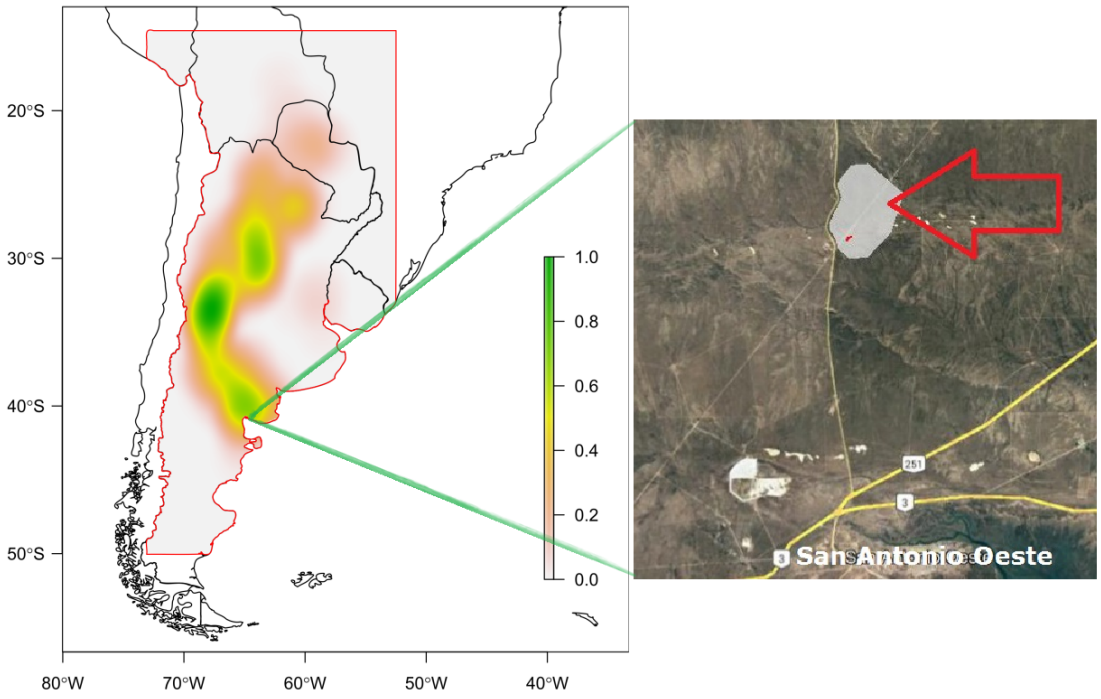


Figura 1.1: Región habitada por la especie de tortuga terrestre *C. chilensis* en Argentina junto a la ubicación de la zona donde se realizaron los estudios de campo.

1.2. Relevamiento y trabajo en campo

Para la definir los requerimientos del proyecto fue necesario interactuar con los diferentes miembros del grupo interdisciplinario de estudio de comportamiento animal. Por este motivo, participe en dos campañas de conservación de tortugas terrestres, realizando las mismas tareas que los demás integrantes del equipo, para poder traducir las necesidades del grupo en especificaciones técnicas, como se verá más en detalle en el Capítulo 2.

1.2.1. Relevamiento de campo

A continuación se detallará y analizará el trabajo en campo de modo de poder determinar e ilustrar cuáles son las complejidades, dificultades y requerimientos del

mismo. Es a partir de este relevamiento que se determinarán las necesidades del grupo de investigación, de modo de poder establecer qué criterios debe cumplir la solución tecnológica a desarrollar.

Se utilizan cuatro métodos para la investigación del comportamiento y movimiento de tortugas terrestres:

1. **Observación directa**: se realizan observaciones en campo de las tortugas, a una distancia que permita ver el comportamiento sin perturbar al animal, por ejemplo mediante el uso de binoculares. En un principio se busca distinguir si el animal está activo o inactivo (si se encuentra o no realizando una acción), y en caso de detectar actividad, catalogar a que tipo corresponde (por ejemplo, si está caminando, comiendo, bebiendo o copulando). Es importante mencionar que las tortugas se encuentran dispersas en un área de aproximadamente 4,5 km^2 donde no es usual encontrar más de un ejemplar en la misma línea de visión. Por lo cual, este método requiere de un observador por cada tortuga, presentando un limitante importante. Además, las mismas se encuentran en refugios (puntos elegidos por las tortugas para descansar durante la noche o refugiarse del calor durante el día) que no son estáticos, sino cambiantes incluso en un mismo día, dificultando aun más la observación.

2. **Marcación con rastreador por radio frecuencia**: se coloca sobre el caparazón de la tortuga un transmisor de radiofrecuencia (figuras 1.2a y b) y se utiliza un equipo rastreador (1.2c) con el fin de encontrar más fácilmente a los especímenes a observar durante una campaña.

   El rastreador utilizado consta de un receptor de RF (banda 148 a 152 MHz) junto a una antena direccional tipo Yagi-Uda. Si la dirección a la cual se apunta la antena coincide con la del transmisor (el cual emite pulsos a una frecuencia única cada 2 segundos), el receptor emite un sonido que es proporcional al nivel de potencia recibida. Esto permite al investigador, ajustando la ganancia del receptor, determinar la posición actual del animal al caminar en la dirección donde se escucha el sonido del receptor con mayor intensidad. De esta forma se puede registrar la posición del animal utilizando un receptor GNSS manual u otro método de los mencionados a continuación.

3. **Seguimiento con bobinas de hilos**: se coloca un carrete de hilo sobre el caparazón de la tortuga y se ata al extremo de una rama (figura 1.3a). Al moverse la tortuga, el hilo se desenrolla y queda sujeto a la vegetación del suelo, dejando marcada el trayectoria de la misma. Luego, con un receptor GNSS de mano se geolocalizan los vértices de la trayectoria formada por el hilo para obtener el recorrido realizado(figura 1.3b). En general el hilo se coloca al anochecer cuando la

tortuga se refugia. Luego, el hilo se desenrolla durante las primeras horas cuando la tortuga comienza su actividad, o después de que se va a su refugio (cuando el sol comienza a ocultarse y baja la temperatura). Una de las limitaciones de ésta técnica es que el carrete solo tiene 100 metros de hilo.

4. **Monitoreo con dispositivo prototipo**: se realiza el monitoreo con un dispositivo prototipo previamente desarrollado por el grupo de investigación conformado por un módulo microcontrolador (con batería y circuitería de carga), sensores inerciales (acelerómetro y giróscopo), sensor de temperatura y un receptor GNSS (figura 1.4a). El mismo registra los datos de los sensores en tiempo real (con excepción de la posición que es adquirida cada un intervalo de al menos 10 minutos) y los almacena en una memoria microSD, con una autonomía de hasta 18 horas debido a la capacidad de la batería. El dispositivo se coloca sobre la tortuga con cinta camuflada para evitar la atracción de depredadores (figura 1.4b). Este procedimiento se realiza cuando la tortuga está inactiva, a las primeras horas del día, y se retira al final del día, tomando nota del estado de actividad del animal al momento de realizar tanto el encendido como el retiro del dispositivo. Luego se extraen los datos binarios de la memoria microSD y se convierten a un formato legible. Finalmente, se recarga cada dispositivo para poder reutilizarlos al día siguiente.

   Debido a su limitada autonomía, el dispositivo es colocado durante las primeras horas de luz solar (cuando las tortugas están inactivas) y es retirado cuando baja la temperatura, aproximádamente una hora antes del crepúsculo. Los transmisores mencionados en el punto 2 contribuyen sustancialmente a facilitar la tarea de localización de las tortuga para la recuperación y colocación diaria de este dispositivo.

En la figura 1.5 se puede observar un diagrama donde se visualiza cómo se entrecruza el tipo de información obtenida mediante los diferentes métodos utilizados. Es importante entender que la tarea de observación directa es la que permite asociar los valores registrados por los diferentes sensores del dispositivo prototipo con el tipo de actividad que realiza el animal.

1.2.2. Impacto del trabajo

Como se expuso en la sección anterior, los métodos utilizados por el grupo de investigación no permiten recolectar información por intervalos mayores a 18 horas. Además tanto las características del terreno como su vegetación (figura 1.6) y área del mismo (4.5 km^2) dificultan aún más la tárea de observación y de localización de las tortugas.

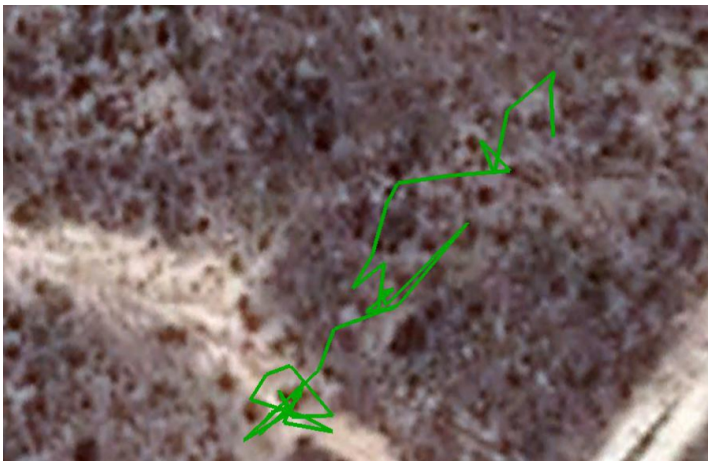
(a) Transmisores de RF.

(b) Tortuga con transmisor colocado.

(c) Investigador con antena tipo Yagi-Uda detectando la dirección de arribo de la señal.

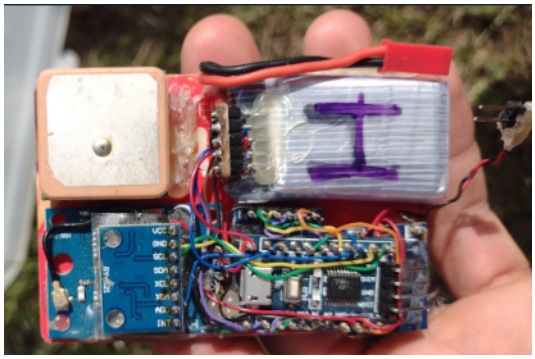
Figura 1.2: Método del transmisor.

(a) Tortuga con hilo colocado.



(b) Trayectoria del hilo.

Figura 1.3: Elementos utilizados en el método de los hilos.



(a) Dispositivo prototipo de monitoreo.

(b) Dispositivo colocado en tortuga.

Figura 1.4: Método de monitoreo con dispositivo prototipo.

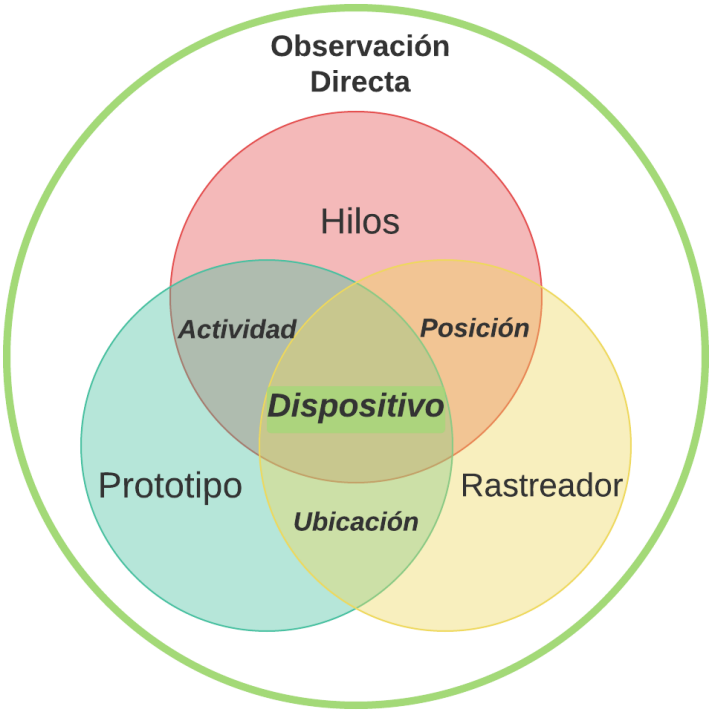


Figura 1.5: Diagrama comparando en forma cualitativa los diferentes métodos y la información que proveen.

El conjunto de estos factores generan complicaciones en términos de logística, limitando considerablemente la cantidad de animales a monitorear, debido principalmente al tiempo necesario para poder localizar a los individuos con radiotransmisor y luego colocarles el dispositivo prototipo en forma diaria. Además las tareas de colocado deben realizarse en una franja horaria donde los individuos se encuentran inactivos (primeras horas de salida del sol).

Por este motivo, el principal objetivo de este proyecto es poder realizar una versión rediseñada del dispositivo prototipo que permita obtener el mismo tipo de información que proveen los métodos que actualmente utiliza el equipo de investigación, buscando que este nuevo desarrollo facilite las tareas de campo y en consecuencia disminuya el esfuerzo de muestreo.

Los costos aproximados del equipamiento electrónico utilizado por el grupo de investigación son:

- Dispositivo prototipo: 50 USD.
- Rastreador:

  Receptor ATS track R410 Receiver 148-152 MHz: 1200 USD.

  Antena Yagi de 3 elementos plegable 148-152 MHz : 150 USD.
- Transmisores A5000 Avian Tarsal-Jess: 200 USD.

Figura 1.6: Imagen del hábitat de la tortuga *C. chilensis*. La vegetación del terreno consta de pastizales de entre 20 y 40 cm de altura y arbustos o arboles de hasta 2 metros. Además en el terreno se encuentran diferentes tipos de depresiones y desniveles.

- Receptor GNSS Garmin 2xS/3S: 250 a 350 USD.

Implicando que el monitoreo de cada tortuga con transmisor y dispositivo prototipo tiene un costo de 250 USD por individuo, además de los costos comunes de 1700 USD. Como ventaja adicional, seria deseable que la solución propuesta baje estos costos.

1.3. Solución propuesta

Como se mencionó al principio de este capítulo, el objetivo de este trabajo es la creación de una familia de dispositivos que permitan realizar el monitoreo del animal en la forma descripta anteriormente. Para eso, se propuso como solución una familia compuesta por un dispositivo de monitoreo (MD), un dispositivo rastreador (TD) y una estación colectora (DCS). El MD es aquel que va colocado en el animal a monitorear, recolectando diferentes datos de interés, y enviando dos tipos de señales por radio frecuencia; una que contiene parte de la información recolectada para permitir al investigador conocer el estado de los dispositivos a través de la DCS mientras los MD realizan la adquisición, y otra sin información denominada "*Keep Alive pulse*", que permite realizar la recuperación del dispositivo mediante el uso del TD. La DCS es aquel miembro de la familia encargado de recolectar la información enviada por radio frecuencia por los diferentes MD activos, de modo de permitir al investigador visualizar esta información en tiempo real por medio de una PC conectada a la misma, datos como el nivel de batería o geolocalización de cada MD, junto a la geolocalización de la propia DCS. Por otro lado, el TD es quién permite al investigador la localización del dispositivo utilizando el mismo principio explicado al inicio de este capítulo en el método "*Marcación con rastreador por radio frecuencia*", en donde el transmisor es el MD y el rastreador consiste en un TD conectado a una antena Yagi-Uda junto a una interfaz, como puede ser un *smartphone* enlazado por Bluetooth a la TD, que permita al usuario seleccionar que dispositivo desea rastrear y notificar al mismo en forma cualitativa la cercanía al dispositivo.

En la figura 1.7 se esquematiza la interacción entre los diferentes dispositivos, donde se encuentran varios MD colocados sobre los caparazones de tortugas recolectando datos y emitiendo una parte de estos datos por RF, siendo recibidos por la antena omnidireccional del DCS. A su vez, un investigador se encuentra utilizando un dispositivo celular enlazado al TD junto a una antena direccional tipo Yagi-Uda para localizar a un individuo en particular.

El alcance de esta Tesis consistirá en el diseño del hardware para toda la familia de dispositivos y la implementación del firmware abarcará al MD y las funcionalidades básicas de recepción y decodificación de información del TD y la DCS.

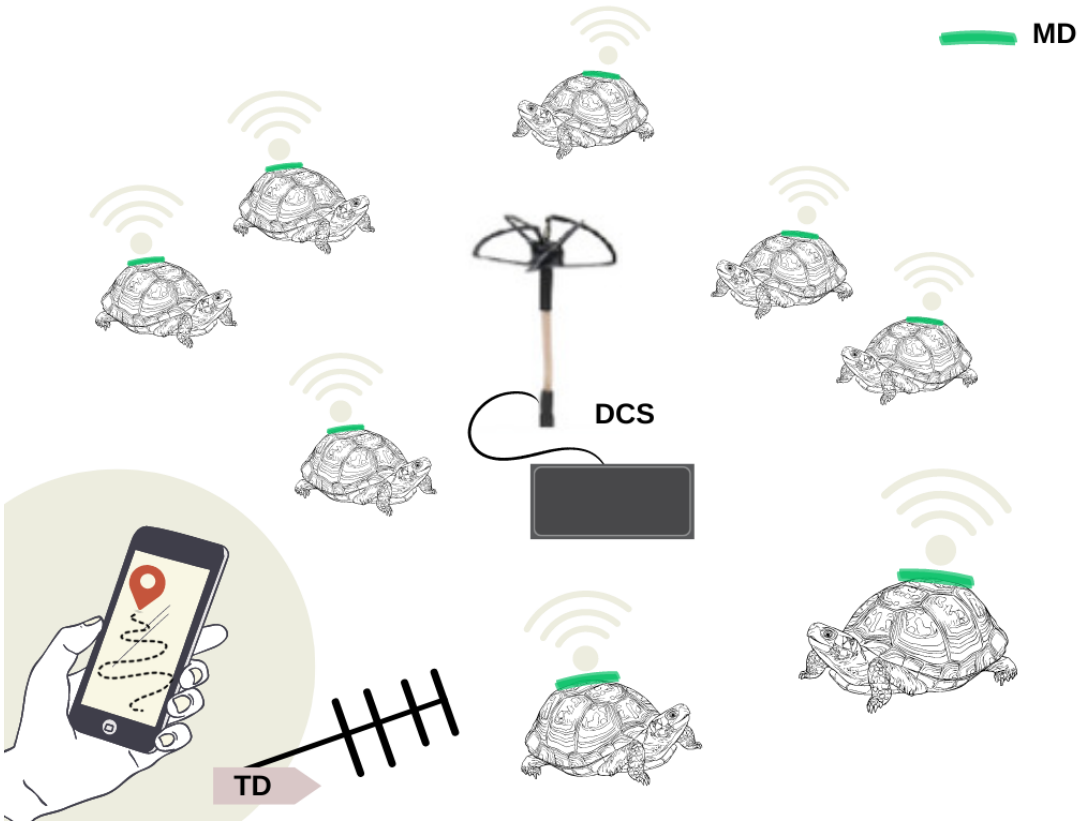


Figura 1.7: Familia de dispositivos propuesta interactuando entre si: dispositivo de monitoreo (MD), dispositivo rastreador (TD) y estación colectora (DCS).

1.3.1. Ventajas

Algunas de las ventajas de la solución propuesta son:

- Prolongar el tiempo de monitoreo en los animales sin la necesidad de molestarlos al lograr una solución con mayor autonomía.
- Poder acceder en forma remota a la información adquirida.
- Reducir el tiempo necesario para la colocación de los dispositivos disminuyendo el esfuerzo de muestreo.
- Posibilidad de configurar los sensores en forma remota.
- Fortalecer la colaboración y vínculo con otros grupos de investigación, ofreciendo la solución para el monitoreo de otras especies de interés.

1.3.2. Consideraciones durante el desarrollo del proyecto

Es importante mencionar que el presente trabajo fue desarrollado en el contexto de una crisis de abastecimiento de componentes electrónicos a nivel global originada como consecuencia de la pandemia de COVID-19, implicando demoras, aumento de costos y

limitaciones en la adquisición de los componentes necesarios para la fabricación de la familia de dispositivos. En los Capítulos 3 y 5 se profundizará en mayor detalle respecto del impacto de esta crisis en el presente trabajo [17–19].

Capítulo 2

Especificaciones de diseño

2.1. Especificaciones generales del proyecto

A partir de la información dispuesta en el Capítulo 1 y de la participación en una campaña de conservación, se establecieron las siguientes especificaciones de diseño para la familia de dispositivos. Las mismas fueron establecidas luego de varias reuniones con los integrantes del grupo de trabajo interdisciplinario.

1. Ensamble: el dispositivo deberá poder ensamblarse en Argentina. Es deseable que se pueda ensamblar o reparar con las instalaciones y equipamiento disponibles en el Instituto Balseiro.

2. Datos: el dispositivo de monitoreo (MD) debe recolectar y almacenar datos que se consideren de utilidad para determinar el comportamiento animal. La información debe corresponder a, al menos, 1 semana de actividad.

   Deberá poder adquirir los siguientes tipos de datos:

   - Temperatura
   - Aceleración
   - Velocidad de ángulo de giro
   - Geolocalización

3. Actividad: es deseable que el sistema permita realizar detección de actividad (animal en reposo, en movimiento, etc). Es deseable que el sistema sea compatible con la tecnología de aprendizaje automático en sistemas embebidos (*TinyML*) [20], permitiendo la ejecución de éstos modelos en dispositivos de pocos recursos y baja potencia. Es importante mencionar que esta tecnología cuenta con el soporte de la plataforma *EdgeImpulse* [21], utilizada por el grupo de investigación. La misma ofrece herramientas tanto para adquirir, preprocesar y etiquetar datos,

como para entrenar y evaluar modelos de aprendizaje automático, de modo de facilitar considerablemente la creación de modelos *TinyML*.

4. Comunicación de radio frecuencia (RF): todos los dispositivos deberán permitir una comunicación bidireccional de RF en la banda de 150 MHz, al ser esta frecuencia compatible con el equipamiento que utiliza el grupo de investigación. El enlace de radio deberá tener el mayor alcance posible, siendo deseable un alcance de al menos 500 metros. A su vez, el MD deberá transmitir los siguientes datos con un intervalo mínimo de un minuto:

   - Identificador del dispositivo
   - Temperatura
   - Posición
   - Nivel de Batería
   - Datos del acelerómetro o código de tipo de actividad

   A su vez, es deseable que los MD puedan comunicarse entre ellos a través de un protocolo estándar.

5. Batería: todos los dispositivos deberán contar con una batería recargable.

6. Diseño reutilizable: el diseño del MD deberá permitir su reutilización en diseños de dispositivos a utilizar en otros animales.

7. Peso: el peso del MD no deberá superar el peso de 75 g, correspondiente al 5 % del peso del animal, que se estima en 1.5 kg.

8. Autonomía: el MD deberá tener una autonomía de al menos 24 horas.

9. Dimensiones: el dispositivo a desarrollar deberá tener un tamaño menor al dispositivo prototipo utilizado por el grupo (72 x 54 mm). Es deseable que el diseño se pueda reutilizar para crear versiones más pequeñas del dispositivo a utilizar en otras especies animales.

10. Indicadores: la familia de dispositivos deberá contar con interfaces, como un sistema de luces, para informar al usuario respecto del estado o funcionamiento del dispositivo.

2.2. Diagrama de bloques y restricciones

A partir de las especificaciones listadas, se elaboró el siguiente diagrama de bloques (figura 2.1) que ilustra los diferentes elementos con los que cuenta la familia de dispositivos.

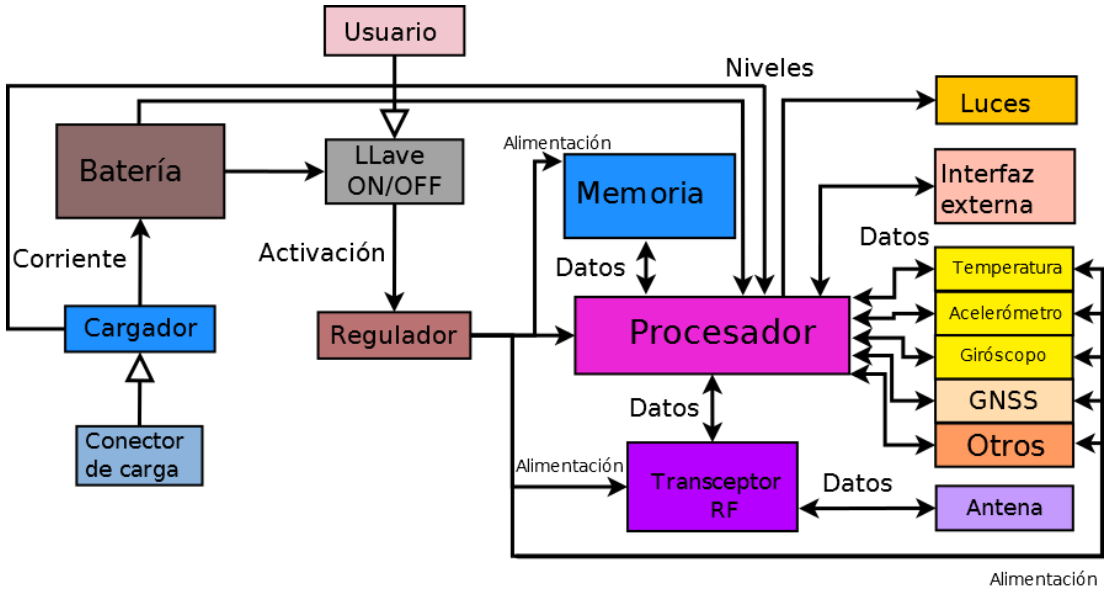


Figura 2.1: Diagrama de bloques donde se muestran los diferentes módulos interactuando entre sí.

- Procesador: gestiona la interacción de los diferentes bloques. Configura y recibe los datos de los diferentes sensores de los cuales se extraen los datos a recolectar y almacenar.
- Batería: provee el suministro de energía a los demás elementos del sistema.
- Cargador: permite realizar la carga de la batería del dispositivo.
- Conector de carga: permite proveer al cargador de una fuente de energía externa para que este realice la carga de la batería.
- Regulador: permite adaptar la tensión de alimentación suministrada por la batería a los valores utilizados por los diferentes módulos del sistema.
- Transceptor RF: transmite o recibe datos por radio frecuencia.
- Antena: actúa como interfaz entre el transceptor RF y el aire por el cual se propagan las ondas de RF a transmitir y recibir.
- Memoria: consiste en la unidad en la cual se almacenarán los datos.
- Llave ON/OFF: permite al usuario encender o apagar el dispositivo.
- Bloques de sensores: representan los diferentes módulos que adquieren los datos a utilizar para determinar el comportamiento animal. El bloque “Otros” contempla cualquier tipo de sensor que el grupo considere de interés (como de sonido) y que sea factible a añadir en etapas posteriores del diseño.

- Interfaces: representa a una salida a una interfaz externa, como un celular o computadora personal (PC), que permita al usuario interactuar con el dispositivo en forma directa. Esta interfaz debe permitir además programar el dispositivo desde una PC.

2.2.1. Restricciones de diseño

A partir del diagrama de bloques expuesto en la figura 2.1 y considerando los criterios de diseño mencionados anteriormente, se listan las siguientes restricciones de diseño:

- Tensión de alimentación: es deseable que sea la misma para todos los módulos.
- Memoria: de ser posible, se prefiere una del tipo extraíble tipo microSD, al permitir una extracción de los datos fácil y rápida.
- Batería: deberá ser recargable y adquirible localmente.
- Cargador de batería: mientras no afecte a la restricción de tamaño del dispositivo, deberá estar integrado a la placa del mismo.
- Tipo de conector de cargador: deberá ser un tipo de conector estándar del cual se pueda disponer fácilmente, como las fichas tipo USB micro B.

A partir de las especificaciones generales del proyecto expuestas y del diagrama de bloques y restricciones, se confeccionaron las siguientes especificaciones que deberán cumplir el *hardware* y el *firmware*, cuyos diseños se expondrán en los Capítulos 3 y 4.

2.3. Especificaciones de *hardware*

A continuación se listan los requisitos que el *hardware* deberá satisfacer, el cual incluye la placa de circuito impreso (PCB) junto a los demás componentes, módulos y conectores que van soldados a la misma.

1. Desarrollo: el mismo diseño y PCB deberá poder usarse tanto como MD, dispositivo rastreador (TD) y estación colectora (DCS), observados en la figura 1.7, en función de los componentes o módulos que se le suelden. A su vez deberá posibilitar su utilización como placa de desarrollo, para poder hacer una depuración del *firmware* del sistema.

2. Dimensiones: el PCB deberá ser del menor tamaño posible, sin ser mayor que el tamaño del dispositivo prototipo utilizado por el grupo de investigación (72 x 54 mm). Se deberán utilizar componentes y circuitos integrados del menor tamaño posible que permitan una manipulación manual.

3. Comunicación RF: el dispositivo deberá estar adaptado a la banda de 150 MHz, soportando una comunicación bidireccional. Es deseable que soporte estándares de internet de las cosas (IoT), junto a la posibilidad de crear redes *mesh* entre MD y MD.

4. Peso: deberá ser lo más liviano posible y menor al 5 % del peso promedio de una tortuga de 1.5 kg, es decir que deberá pesar 75 g o menos. A su vez es deseable que el peso, sin considerar el receptor de sistema global de navegación por satélite (GNSS) del dispositivo y la batería, sea menor a 4 gramos, lo cual facilitaría la adaptación del diseño para monitorear animales más pequeños como lagartijas.

5. Costo: Es deseable que el costo total de fabricación no supere la suma del costo de un transmisor de radiofrecuencia junto al del dispositivo prototipo utilizado actualmente por el grupo (250 USD).

6. Tensión de alimentación: es deseable que se utilice la misma tensión eléctrica para la alimentación de todos los circuitos integrados, de modo de disminuir la cantidad de circuitos integrados y pistas necesarios en el PCB.

7. Batería: deberá incluir y permitir la conexión de una batería, con una ficha de alimentación USB micro B, que permita realizar la recarga de la misma.

8. Interfaz: deberá contar con al menos 3 luces leds testigos manejados por el MCU y una luz led testigo que indique si el dispositivo esta recibiendo alimentación desde la ficha de alimentación para la carga de la batería. Además tiene que permitir la conexión de un módulo externo Bluetooth por comunicación UART sin que esto requiera remover elementos del PCB.

9. Llave: deberá tener una llave que permita encender o apagar el dispositivo al conectar o desconectar la batería.

10. Sensores: deberá incluir los siguientes sensores:

    - Receptor GNSS
    - Acelerómetro
    - Giróscopo
    - Termómetro
    - Magnetómetro
    - Sensor de Luz
    - Micrófono

11. Potencia RF: deberá permitir medir el valor de indicador de nivel de señal (RSSI) de las señales de RF recibidas.

12. Antena: la antena deberá ser flexible y del menor tamaño posible.

13. PCB: de ser posible, se resolverá el ruteado de las pistas en una sola cara del PCB, dejando el otro lado como plano de tierra. Es deseable que el PCB sea fabricado con un sustrato flexible de modo de facilitar la colocación del dispositivo sobre el animal.

14. Detección de actividad: es deseable que el *hardware* sea compatible con la tecnología *TinyML*, requiriendo tener una unidad de coma flotante (FPU) en el microcontrolador para su funcionamiento adecuado, o que la arquitectura del *hardware* provea instrucciones optimizadas para la implementación de algoritmos de procesamiento de señales, como sucede en las arquitecturas ARM cortex M3 o M4 [20, 22–24].

2.3.1. Especificaciones de *firmware*

1. Sistema operativo: deberá contar con un sistema operativo en tiempo real (RTOS), siendo deseable que el sistema sea compatible con FreeRTOS. Todas las tareas deberán ser diseñadas de forma que la mayor cantidad de procesos sean no bloqueantes, es decir que no deshabiliten el uso del procesador cuando no se estan realizando tareas de procesamiento, y estén claramente desacoplados de otras tareas.

2. Bajo consumo: el diseño deberá disminuir el consumo de energía del sistema al mínimo, utilizando los diferentes modos de energía que soporte el *hardware*, de modo de lograr diferentes configuraciones de energía. La autonomía deberá ser de al menos 24 horas.

3. Información: la información recolectada por cada tipo de dispositivo deberá permitir determinar en forma unívoca a que sensor pertenece. La información almacenada en la memoria deberá contener caracteres legibles que ayuden al proceso de depuración.

4. Comunicación RF: deberá permitir una comunicación inalámbrica bidireccional. Deberá poder enviar pulsos de radio frecuencia, permitiendo mandar tanto datos como un pulso sin información (denominado "*Keep Alive*") que pueda ser detectado tanto por los dispositivos TD y DCS, como por los receptores utilizados por el equipo de trabajo. La frecuencia en la que se envían los pulsos en la radio deberá ser configurable y única para cada dispositivo.

5. Configuración dinámica: el sistema deberá posibilitar que se pueda modificar la configuración de los sensores ya sea por un comando externo al dispositivo (como por radio frecuencia) o en forma interna (por ejemplo, a través de un algoritmo que determine el nivel de actividad).

6. Interfaz: el *firmware* deberá evaluar y notificar al usuario de diferentes estados del sistema o eventos que se consideren de interés. Estos estados o eventos incluyen desde el resultado de verificaciones básicas al inicio del sistema (como sí el nivel de batería o el estado de los sensores es el adecuado para comenzar con la adquisición de datos), hasta eventos que sucedan durante la adquisición y que condicionen el funcionamiento del dispositivo (como que la batería se encuentre cerca de agotarse). En el Capítulo 4 se entrará más en detalle sobre estos estados.

7. Exportabilidad: es deseable que la arquitectura del *firmware* facilite la migración a otro tipo de MCU mediante la implementación de capas de abstracción.

Con estas especificaciones en mente se procedió al diseño del *hardware* y el *firmware*, como se detallará en los próximos capítulos.

Capítulo 3

Diseño de *hardware*

A partir de las especificaciones obtenidas en el Capítulo 2 para el diseño del *hardware*, se desarrollará en este capítulo la selección de los circuitos integrados (IC) junto al diseño de la placa de circuito impreso (PCB).

Antes de continuar con la selección del microcontrolador y los demás circuitos integrados, es importante mencionar que se consideraron los siguiente criterios para su elección:

- Bajo consumo y soporte de modos de ahorro de energía.
- Tamaño de encapsulado pequeño, baja cantidad de pines y de componentes adicionales necesarios.
- Facilidad de montaje y manipulación.
- Disponibilidad de librerías.
- Disponibilidad de placas de desarrollo o módulos.
- Tensión nominal de operación de 3.3 V, valor estándar tanto para alimentación como lógica digital.
- Bajo costo.

3.1. Selección de microcontrolador

Debido a que en este proyecto se busca tener un PCB del menor tamaño posible, así como también contar con la posibilidad de reutilizar el diseño para diseñar versiones más pequeñas del MD, se utilizaron sistemas integrados (SoC). Esto permite integrar la mayor cantidad de elementos del sistema expuesto en el diagrama de bloque de la figura 2.1. Tras hacer un relevamiento inicial, se encontraron varios SoC que integran

tanto el microcontrolador (MCU) como el transceptor de radiofrecuencia (RF) en un solo circuito integrado.

Radiofrecuencia

Previo a continuar con el relevamiento de mercado y la selección de los componentes, es importante mencionar las técnicas de modulación de RF e internet de las cosas (IoT) adecuadas para el entorno físico en el cual se utilizarán los dispositivos con banda angosta (Narrowband o NB-IoT) y largo alcance (LoRa®). Ambas técnicas de modulación, orientadas al bajo consumo y largo alcance, disminuyen la degradación de la señal de RF debido a atenuaciones, pérdidas por propagación y rebote de la señal, provocada por características del entorno como desniveles en el terreno, presencia de vegetación y árboles [25].

El estándar banda angosta, como indica su nombre, utiliza un ancho de banda reducido de modo de realizar la transmisión de datos a una velocidad menor para lograr un alcance mayor, convirtiendo a este estándar en una solución eficiente en términos de consumo de energía [26].

Por otro lado, LoRa® es una tecnología de comunicación inalámbrica de largo alcance que utiliza modulación de espectro ensanchado (DSSS) para transmitir datos a través de largas distancias con un bajo consumo de energía, siendo especialmente adecuada para aplicaciones en áreas rurales, así como para aplicaciones que requieren una larga vida útil de la batería y una transmisión de datos de baja velocidad. Esta tecnología se basa en lograr diferentes combinaciones de factores de espectro esparcido y tasa de transmisión de datos, permitiendo que el transmisor ajuste estos parámetros para incrementar el alcance. Una característica muy interesante de LoRa® es que existen transceptores compatibles con esta tecnología que pueden recibir señales de RF con diferentes combinaciones de factores de espectro esparcido y tasas de transmisión de datos en simultáneo, a los cuales denominaremos como LoRa® *Gateway*, sin necesidad de realizar configuraciones en el mismo. Por lo cual, el lado transmisor puede ajustar estos valores cuando lo requiera y el LoRa® *Gateway* continuará recibiendo los mensajes [27–29].

3.1.1. Especificaciones del SoC MCU

A partir de las especificaciones de *hardware* y *firmware*, se obtuvieron las siguientes especificaciones para la elección del SoC.

- El MCU deberá presentar un bajo consumo cuando funcione a su capacidad nominal de procesamiento. A su vez, deberá proveer diferentes modos de funcionamiento (como *sleep*) que permitan cambiar el consumo del microcontrolador en tiempo real.

- Tener un transceptor de RF con una frecuencia mínima de operación en 150 MHz.
- La arquitectura del MCU debe soportar un sistema operativo en tiempo real (RTOS).
- Soportar interfaces UART, I2C y SPI, siendo deseable que soporte una interfaz de sonido como I2S.
- Soportar el uso de estándares IoT orientados al bajo consumo y largo alcance como LoRa® o Narrow Band IoT. Es deseable que además soporte estándares de IoT que permitan a los MD actuar como intermediarios o repetidores en la comunicación de otros MD y la DCS, conocidos como topologías tipo *mesh*. Este tipo de topologías dan la posibilidad de incrementar el alcance máximo del enlace.
- Es deseable que soporte la tecnología *TinyML*, requiriendo tener una unidad de coma flotante, de modo de poder utilizar ésta tecnología para clasificar el tipo de actividad del animal monitoreado en tiempo real [20, 22–24].

3.1.2. Relevamiento del mercado

A partir de un relevamiento del mercado, se encontraron cuatro SoCs de interés para este proyecto. Los mismos se muestran en la tabla 3.1. Es importante mencionar que todos los integrados soportan las interfaces de comunicación especificadas, trabajan en un rango de tensión de 1.8 a 3.6 V, la velocidad de sus procesadores es de 48 MHz, permiten medir la potencia de las señales recibidas de RF y soportan diferentes tipos de modulación en banda angosta, junto a la posibilidad de aplicación de técnicas basadas en espectro esparcido que puedan ayudar a incrementar el alcance del enlace en ciertos escenarios.

De esta primer comparación se puede observar como los microcontroladores ofrecidos por los fabricantes ST, Texas Instruments y Microchip proveen diferentes modos de bajo consumo que son de interés para este proyecto. Adicionalmente, los SoCs de Texas Instruments y ST, cuentan con un procesador dedicado específicamente para el manejo de la radio RF. Esto permite utilizar las funciones de la radio en forma no bloqueante, evitando que estas tareas utilicen los recursos del procesador principal, disminuyendo la carga en el mismo.

En la tabla 3.2 se muestran las características de la radio RF, incluyendo los valores de energía requeridos al transmitir o recibir señales de radiofrecuencia, junto a una breve descripción de la información encontrada respecto del entorno de desarrollo, librerías, proyectos y actividad de comunidades. Es importante mencionar que el valor de sensibilidad mostrado, es el valor mínimo encontrado en la hoja de datos de cada integrado, el cual depende del valor de la tasa de transferencia de datos y del factor de

Fabricante y Soc	Tamaño $[mm]$	Memoria $[kB]$	CPU(s)	Modos de Consumo $[uA]$	Costo, IC y Placa de desarrollo $[USD]$
Texas Instruments CC1312R	7 x 7 5 x 5	352 flash 88 RAM	M4F M0+(radio)	0,8 standby 500 idle 2900 active	IC: 8 Placa: 53
ST STM32WLE5	7 x 7 5 x 5 (BGA)	256 flash 64 RAM	M4F M0+ (radio)	0,3 standby 200 low CPU 3420 active	IC: 10 Placa: 42
Microchip ATSAMR34J18	6 x 6 (BGA)	256 flash 40 RAM	M0+	1 standby 4500 active	IC: 6.5 Placa:120
ASR ASR6501	6 x 6	128 flash 16 RAM	M0	3 sleep	N/a

Tabla 3.1: Matriz de comparación donde se observan en forma resumida las principales características de los diferentes SoCs evaluados.

espectro esparcido utilizado para su cálculo. De esta tabla también se puede observar como los integrados de ST y Texas Instruments presentan el menor consumo para recepción y valores similares de consumo para transmisión.

A partir de lo expuesto, se seleccionaron dos de los cuatro SoCs: el CC1312R de Texas Instruments y el STM32WLE5 de ST. Ambos SoCs comparten características muy similares respecto del consumo, modos de ahorro de energía, consumo de radio RF, procesador dedicado para la misma y rango de frecuencia, cantidad adicional de componentes necesarios para el montaje, mayor cantidad de memoria RAM respecto de las demás opciones, cantidad y variedad de soporte de protocolos de comunicación digitales y de IoT, junto a la inclusión de un sensor de temperatura y de nivel de fuente del integrado embebidos en el SoC y una amplia información respecto de sus entornos de desarrollo y librerías. A su vez, ambos dispositivos cuentan con un núcleo ARM-Cortex M4F, el cual soporta a nivel *hardware* operaciones con coma flotante y, por lo tanto, es compatible con la tecnología *TinyML*.

3.1.3. Comparación entre SoC de ST y Texas Instruments

Con el fin de determinar cuál de los dos SoC es el más adecuado para el proyecto, se compararán otras características (tabla 3.3) tales como el tamaño del integrado, el enlace de radiofrecuencia, la disponibilidad de módulos de desarrollo, los costos, junto a la disponibilidad de información y herramientas de desarrollo.

Como se mencionó anteriormente, una consideración importante respecto de LoRa® es que permite realizar combinaciones de factores de espectro esparcido y tasa de datos junto a la posibilidad de utilizar un LoRa ® *gateway* para recibir múltiples señales con

Fabricante y SoC	RF e IoT	Consumo Rx [mA] y sensibilidad mín. [dBm] Consumo Tx [mA] y maxPout [dBm]	Herramientas de desarrollo entorno, librerías, comunidad)
Texas Instruments CC1312R	IEEE 802.15.4g, SigFox,MIOTY WiSUN	Rx: 5,8 mA, -121 dBm Tx:14/24 mA, +10/14 dBm	Si. Librerías Comunidad
ST STM32WLE5	IEEE 802.15.4g Sigfox, MIOTY, LoRa®	Rx: 5.4 mA, -148 dBm Tx:17/24 mA, +10/14 dBm Tx: 120 mA, +22 dBm	Si Librerías Proyectos
Microchip ATSAMR34J18	IEEE 802.15.4g LoRa® WiSUN	Rx: 10,3 mA, -148 dBm Tx:10/28 mA, +10/13 dBm	Si Comunidad
ASR ASR6501	LoRa®	Rx: 10,1 mA, -140 dBm Tx:59/118 mA, +10/22 dBm	____

Tabla 3.2: Matriz adicional de comparación con un resumen de las características de RF de los diferentes SoCs evaluados.

diferentes combinaciones de estos parámetros sin la necesidad de cambiar la configuración de este transceptor. Por lo cual si se desea sacar provecho de está característica se debe utilizar un LoRa ® *gateway* en la DCS , siendo un integrado distinto al SoC de ST propuesto para el MD, permitiendo que la DCS reciba en simultáneo diferentes combinaciones de tasa de datos y factor de espectro esparcido, haciendo posible que cada MD adapte éstos parámetros de acuerdo a la situación. Los LoRa ® *gateway*, además de tener un costo superior, no vienen presentados en formato SoC. A su vez, LoRa® no cuenta con soporte para redes *mesh* entre SoCs.

Es importante mencionar que los valores del enlace listados en la tabla 3.3 fueron obtenidos como la diferencia entre la potencia transmitida (14 dBm) y la sensibilidad de cada SoC para una tasa de transferencia de datos de 5.2 kbps. Esta tasa de transferencia es un valor razonable para el proyecto y es el valor más bajo para el cual Texas Instruments provee información respecto de la sensibilidad asociada en el modo de comunicación de largo alcance que admite el integrado. De la tabla 3.3 se puede observar que el enlace de RF tiene una diferencia de 6 dB entre ambos SoCs, lo cual corresponde a una diferencia del doble de alcance en condiciones de línea de visión sin obstáculos, siendo la opción de ST la de mayor valor de alcance, permitiendo realizar un ajuste de la tasa de datos para incrementar el alcance gracias a la tecnología de LoRa®.

Por otro lado, el integrado de Texas Instruments provee tamaños más pequeños disponibles para el SoC, ya que la versión de 5x5 mm de ST solo está disponible en empaquetado del tipo *Ball-Grid Array* (BGA), el cual no permite realizar un mantenimiento ni soldadura manual en las instalaciones del laboratorio de Telecomunicaciones.

Integrados	Tama -ño [mm]	En- lace [dB]	Disponibilidad y costo SoC y kits	Disp. y costo módulos	*mesh*	Desarrollo
CC1312R	5x5 7x7	135	SoC Tx, Rx: 8 USD Kits: Tx, Rx: 53 USD	Tx, Rx: 10 USD	Si	Comunidad, información concentrada, soporte activo
STM32-WLE5	7x7	141	SoC/integrado Tx,Rx: 10 USD (9 Meses de demora) *Gateway*: 60 USD Kits Tx,Rx: 42 USD *Gateway*:100 USD	Tx, Rx: 20 USD *Gateway*: 120 USD	No	Protocolo IoT estándar utilizado ampliamente (LoRa®) Librerías y comunidad dispersa

Tabla 3.3: Comparación de parámetros de interés entre SoCs de Texas Instruments y ST.

También permite la posibilidad de desarrollo de redes *mesh* entre dispositivos junto a la disponibilidad de una amplia cantidad de librerías y una comunidad con soporte del fabricante para el desarrollo del *firmware* muy activa.

Selección

A partir de las comparaciones mencionadas anteriormente, se observa que ambos SoCs comparados poseen prestaciones suficientes para la realización de este proyecto, aunque debido a la posibilidad de obtener un mayor alcance en iguales condiciones y poca diferencia de tamaño, se considera como opción principal a elegir el SoC de ST. Sin embargo, es importante mencionar que el mismo se encontraba disponible con 9 meses de demora al momento de realizar la compra. Al considerarse este tiempo de demora como inaceptable para los tiempos de desarrollo de este proyecto, se decidió utilizar el SoC CC1312R de Texas Instruments.

3.1.4. SoC CC1312R de Texas Instruments

En la figura 3.1 se puede observar un diagrama provisto por el fabricante con las diferentes características del SoC CC1312R de Texas Instruments.

Para facilitar la programación, depuración y uso del SoC, Texas Instruments ofrece la placa de desarrollo LAUNCHXL-CC1312R1, la cual combinada con el entorno de desarrollo provisto por el fabricante (*Code Composer Studio*) ofrece diferentes herramientas y funcionalidad que ayudan en el desarrollo de aplicaciones en el integrado. La misma puede ser utilizada como placa programadora, depuradora y además permite al usuario medir la corriente de consumo de los módulos conectados a la misma en un rango que va de los μA a cientos de mA en tiempo real a través de la tecno-

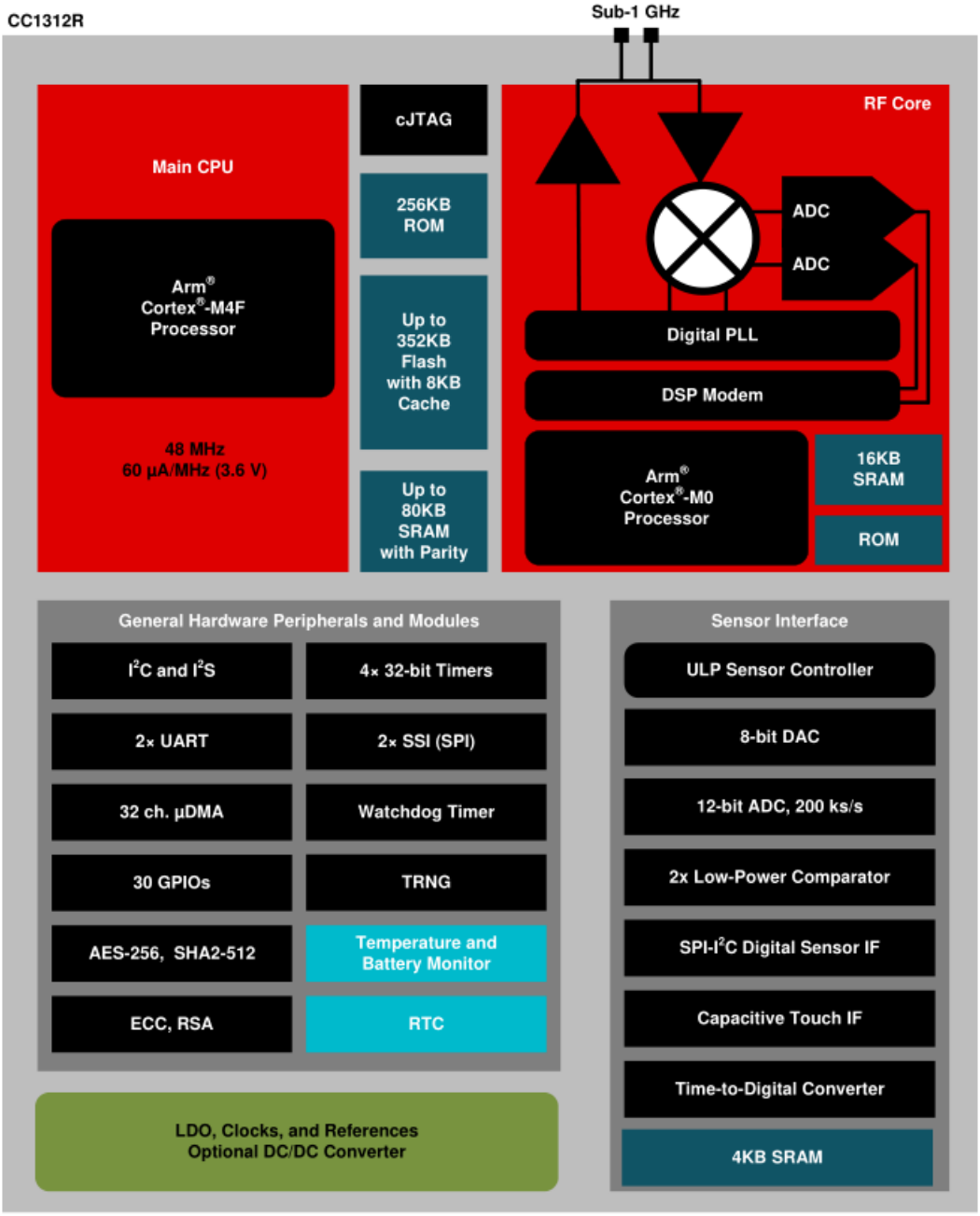


Figura 3.1: Diagrama de bloques y prestaciones del microcontrolador CC1312R de Texas Instruments

logía EnergyTrace ™, siendo de gran utilidad para este proyecto. *Code Composer Studio* además provee interfaces que facilitan el monitoreo de las tareas del sistema operativo en tiempo real, ya sea para ver los tiempos de ejecución o evolución del *stack* del sistema o de cada tarea en particular, junto a otros parámetros del sistema. El kit de desarrollo de software (SDK, por su sigla en inglés) contiene una vasta cantidad de ejemplos y librerías, que van desde el manejo de interfaces de memoria y sonido hasta la creación de redes de IoT tipo *mesh* con el estándar WISUN o tipo estrella con el estándar SimpleLink TI 15.4.

Es interesante destacar que este SoC posee un tercer procesador de 20 MHz de menor consumo, permitiendo optimizar el consumo del sistema en los casos en los cuales los valores de consumo de los sensores se encuentran próximos a los del procesador, con la consideración de que debe ser programado utilizando un entorno de desarrollo diferente y con diferentes librerías a las utilizadas en *Code Composer Studio*. Además contiene soporte de la interfaz I2S, la cual es utilizada para interactuar con integrados que manejan señales de sonido.

3.2. Selección de otros componentes

A continuación se detallarán los diferentes componentes elegidos para la implementación del *hardware* de este proyecto junto a los motivos de dichas elecciones. Los mismos corresponden a los diferentes sensores a utilizar para la adquisición de los datos de interés detallados en el Capítulo 2, junto a un sensor de luz y de sonido, al considerarse de interés para futuras aplicaciones más allá del alcance de este proyecto. Además se incluyen la elección de la batería, memoria, fuente *DC-DC* y *switch* digital, junto a la elección de antenas para cada dispositivo y módulo Bluetooth, de modo que posibilite la interacción del TD con dispositivos móviles.

3.2.1. Sensores inerciales y magnetómetro

Se denominan sensores inerciales a aquellos sensores que miden aceleración y velocidad angular a través del uso de acelerómetros o giróscopos, respectivamente. En el mercado existen tanto integrados que cuentan con uno solo de estos sensores como otras versiones que incluyen a ambos tipos, estos últimos denominados unidades de medición inercial (IMU). Es importante mencionar que algunas IMU pueden además incluir un sensor de temperatura o un magnetómetro.

Para la selección de la IMU se evaluó el uso de diferentes combinaciones de acelerómetro, giróscopo y magnetómetro. Se decidió añadir a los magnetómetros ya que se encontraron varios integrados que incluían los tres tipos de sensores mencionados y podrían proveer información de interés para la investigación. Por lo cual cuando nos

IC	A	G	M	Consumo [μA]	Sleep	Interfaces	Disponible
ICM-20948	X	X	X	A:68 G:1230 M:80	Si	I2C/SPI	No
LSM9DS1	X	X	X	G:1900 A+M:600	Si	I2C/SPI	Si* (demora)
Mpu-9255	X	X	X	A:19 G:3200 M:280	Si	I2C	No
BMI270	X	X		A:6 G:420	Si	I2C/SPI	No
LSM6DSO32	X	X		A:6 G:400	Si	I2C/SPI	No
LSM303AGR	X		X	A:5 M:50 a 200	No	I2C/SPI	No
Lsm303dlhc	X		X	A+M:110	Si	I2C/SPI	No
ADXL362	X			A: 1.2	No	I2C/SPI	No
LIS331HHTR	X			A: 10	No	SPI	No

Tabla 3.4: Integrados presentados como combinaciones de acelerómetro (A), giróscopo (G) y magnetómetro (M).

refiramos en secciones posteriores al término IMU, nos referiremos a un integrado que incluye a los sensores acelerómetro, giróscopo y magnetómetro.

En la tabla 3.4 se pueden observar, tanto las opciones que ofrecen los tres sensores en un solo componente, como los casos en donde se incluye uno o dos sensores en el integrado. Es interesante mencionar cómo algunos de los encapsulados que contienen un solo tipo de sensor exhiben un menor consumo que aquellos que combinan más de uno en el integrado, con la contrapartida de que si se utilizan tres integrados en lugar de uno (que incluya los tres sensores), se requirieren un mayor número de componentes adicionales, y por lo tanto se ocupa un mayor espacio físico.

Debido al problema de faltantes de existencias para la mayoría de los integrados al momento de decidir la selección de componentes (como se mencionó en el Capítulo 1), se optó por elegir el LSM9DS1, ya que el grupo contaba con 5 de estos integrados. Este integrado posee características de interés para el proyecto, como bajo consumo, disponibilidad de módulos de desarrollo, librerías, múltiples ejemplos de uso en internet y modos ahorro de energía. Otra característica interesante del mismo es que cuenta con un *buffer* interno que permite almacenar hasta 32 muestras de cada eje de giróscopo y acelerómetro, permitiendo hacer una transferencia en ráfagas desde la IMU al MCU, aliviando la carga del sistema. El mismo también incluye un sistema de interrupciones por umbral en el cual el MCU puede programar a la IMU un valor de umbral por tipo de sensor, y permite ser configurado de modo que la misma genere interrupciones de

Módulo	Consumo $[mA]$	Tracking $[mA]$	Sleep $[mA]$	Módulo Disp.	Costo $[USD]$	Tamaño $[mm]$
Neo M6M	49	12	1	Si	25	23x30
Neo M7M	47	10	1	Si	25	23x30
Neo M8M	12	3,7	1	Si	30	23x30
L80-M39	25	3	1	No	25	10x10
GPS- L70	12	1.4	0,01	No	40	10x10
MTK3339	25	20	———	Si	25	23x35
MAX2769	25	10	0,001	No	No	9x9

Tabla 3.5: Opciones de selección de receptor GNSS.

hardware al MCU. Esta prestación posibilita el desarrollo de estrategias más sofisticadas de bajo consumo, como apagar todo el sistema y que la IMU lo active si un valor del acelerómetro supera el valor de umbral.

3.2.2. Receptor GNSS

Los receptores de sistema global de navegación por satélite (GNSS) son dispositivos electrónicos que reciben y procesan las señales provenientes de las constelaciones de satélites de navegación, obteniendo a partir de estos, valores como la geolocalización y velocidad del receptor. Los mismos están conformados por una antena y un integrado que realiza la recepción y procesamiento de las señales de radiofrecuencia emitidas por los satélites, que se encuentran en el orden de los gigahertz. Esto implica que la placa donde está montado el integrado debe funcionar correctamente a esa frecuencia, siendo un diseño no trivial. Para facilitar el diseño del PCB y permitir una mayor flexibilidad en el diseño, se decidió utilizar un módulo receptor que incluya el integrado presentado como un módulo junto a su antena, facilitando además quitar el mismo en los casos donde no es necesario (como es el caso del TD).

En la tabla 3.5 se listan los módulos receptores de GNSS, en donde el tamaño corresponde al del módulo de desarrollo donde está montado el integrado. Es importante mencionar que al momento de realizar el relevamiento de los integrados, la mayoría de estos no se encontraban disponibles en el mercado, por lo cual se decidió comprar las versiones disponibles localmente de los módulos Neo 7M del fabricante u-blox. Estos módulos son receptores GNSS que utilizan el sistema de posicionamiento global GPS, con amplia disponibilidad de documentación, librerías y ejemplos debido a su uso masivo.

3.2.3. Memoria

La unidad de memoria del sistema es aquella que permitirá almacenar los datos obtenidos de los diferentes sensores mencionados hasta el momento, siendo los valores

obtenidos con el receptor GNSS, acelerómetro, giróscopo, magnetómetro y sensor de temperatura. Por lo tanto, antes de evaluar el tipo de memoria a utilizar, se realizará una estimación del valor de memoria mínimo necesario.

Estimación del tamaño de la memoria

Para estimar el tamaño mínimo de memoria requerido, se tomará en cuenta el tamaño de los datos a almacenar a partir de los circuitos integrados seleccionados para los sensores, a saber:

- Localización: latitud y longitud en formato double (8 bytes), dando un total de 16 bytes.
- Sensores inerciales: se utilizan 2 bytes para almacenar cada eje. Teniendo 3 instrumentos (acelerómetro, giróscopo y magnetómetro) con 3 ejes (X, Y, Z), da un total de $3 * 3 * 2 = 18$ bytes.
- Temperatura: 1 byte.
- Estampa de tiempo: corresponde a los valores de tiempo en los cuales se han adquirido las muestras de los sensores. Se utilizarán un total de 2 bytes.

Consideramos:

- Tasa de muestreo de 0,1 segundos para los sensores acelerómetro, giróscopo y magnetómetro y para la estampa de tiempo.
- Tasa de muestreo de 10 minutos para localización y temperatura.

Si calculamos para una hora:

$$6 * 16 + 6 * 1 + 3600 * 10 * (18 + 2) = \tag{3.1}$$

$$= 0,72\ MB \tag{3.2}$$

Si extendemos a una semana, al ser un valor de tiempo de interés para el grupo de investigación, se obtiene:

$$7 * 24 * 0,72 = \tag{3.3}$$

$$= 120,97\ MB \tag{3.4}$$

Por lo tanto, la memoria mínima que deberá tener la unidad de almacenamiento debe ser de 120,97 MB o 967,81 Mb.

Selección del tipo de Memoria

Se analizaron las tecnologías de memoria de montaje superficial disponibles en el mercado, donde se encontró que las memorias *flash* fabricadas con circuitos tipo *NAND* presentan el menor consumo para lectura/escritura y modos de ahorro de energía. El tamaño más pequeño de integrado para estas memorias, con una capacidad de 1 Gb, es de 6x8 mm. Una consideración importante respecto a este tipo de memorias es que no pueden ser removidas del dispositivo y requieren del diseño de una interfaz para la extracción de información de las mismas, lo cual complejiza la lectura de los datos adquiridos.

Por otro lado, las memorias microSD permiten realizar una extracción directa de su contenido, al ser removibles del dispositivo y compatibles con interfaces estándar de computadoras personales. Es interesante mencionar que las mismas están diseñadas con memorias *flash* tipo *NAND*, siendo adecuadas para aplicaciones de bajo consumo. Su tamaño es de 11 x 15 mm y su capacidad puede llegar a los cientos de gigabits, ofreciendo una capacidad de almacenamiento mucho mayor respecto de las memorias de montaje superficial.

Debido a las ventajas mencionadas que ofrecen estas memorias respecto de las memorias de montaje superficial, con el costo de tener un tamaño casi cuatro veces mayor, se decidió utilizar memorias microSD como unidad de almacenamiento en este proyecto.

3.2.4. Batería

La batería a utilizar deberá permitir al menos un día de autonomía en el MD, y a la vez cumplir con las restricciones de tamaño y peso máximo del dispositivo (75 g). Para este proyecto se eligió una batería tipo LiPo de 3.7 V y 600 mAh (figura 3.2), cuyo tamaño es de 42 x 30 x 5 mm y su peso de 12 g. La misma fue seleccionada debido a su disponibilidad local y al hecho de que es la batería de mayor capacidad que cumple con las restricciones del dispositivo con respecto a tamaño y peso.

Figura 3.2: Batería tipo LiPo de 3.7 V y 600 mAh.

En el Capítulo 5 se analiza el consumo estimado, en donde en la tabla 4.3 se puede

observar que el consumo estimado promedio es de 9 mAh, por lo cual para lograr un día de autonomía se requiere una batería de 216 mAh.

3.2.5. Regulador de tensión

Debido a que el rango de tensión de funcionamiento de las batería tipo LiPO es entre 2.7 y 4.2 V, siendo este mayor al valor máximo de tensión soportado por el MCU (3.8 V) y distinto al valor de tensión nominal elegido para todos los circuitos integrados (3.3 V), se debe utilizar un regulador *DC-DC* de topología *buck-boost* que permita ajustar el valor de tensión al nominal del sistema.

El regulador elegido fue LTC3531ES6-3.3 (5 USD), debido a su disponibilidad, tamaño (3x3 mm), empaquetado (*TSOT*), baja cantidad de pines (6), baja cantidad de componentes adicionales (3), alta eficiencia (hasta un 90 %), baja corriente parásita (15 μA) y posibilidad de encendido/apagado utilizando lógica digital.

3.2.6. Cargador de batería

Se utilizó el IC LTC4057 (3,5 USD), al tener poca cantidad de pines y requerir de pocos componentes adicionales para su montaje. El mismo es compatible con una tensión de entrada de hasta 6 V, permitiendo el uso de fuentes estándar de USB de 5 V como alimentación para la carga de la batería.

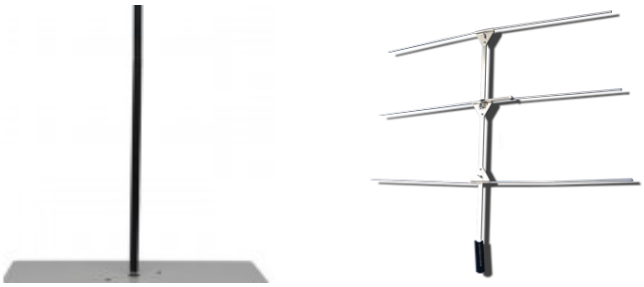
3.2.7. Switch digital

Debido a que el módulo receptor GNSS a utilizar en este proyecto no presenta un modo de apagado, y el modo de ahorro de energía disponible implica un consumo mayor al de los demás sensores utilizados, se decidió incluir en el diseño un switch para poder cortar la alimentación del módulo cuando no se utiliza. Ya que para el caso de la microSD no se encontró información clara respecto del modo de ahorro de energía y el consumo del mismo, también se decidió incluir un switch para la misma.

Se utilizaron los switch tipo MOSFET SiP32431DR3 por ser los componentes encontrados de menor costo (0,7 USD), menor corriente de fuga (10 nA) y de funcionamiento (1 μA), junto a un tamaño de empaquetado pequeño.

3.2.8. Sensor de luz

Para el sensor de luz se eligió el TSL2571 (2 USD), debido a que es un módulo diseñado para la medición de luces en ambientes naturales, bajo costo, tamaño adecuado (2x2 mm) y disponibilidad de librerías.



(a) Antena monopolo. **(b)** Antena Yagi-Uda.

Figura 3.3: Antenas a utilizar para los dispositivos DCS (a) y TD (b).

3.2.9. Micrófono

Para el micrófono se seleccionó el integrado SPH0645LM4H-B (1,6 USD) debido a la disponibilidad de librerías, módulos de desarrollo, soporte de interfaces I2S , bajo consumo (600 μA), rango en frecuencia de muestreo de hasta 48 kHz y tamaño adecuado (3.5 x 2.7 mm).

3.2.10. Selección de antenas

Debido a que cada dispositivo es utilizado para funciones y contextos diferentes, se evaluaron diferentes tipos de antenas (figura 3.3) a utilizar para cada uno. En el caso de la DCS, como la misma debe estar fija y recibir señales desde diferentes localizaciones, el tipo de antena más adecuado es el omnidireccional. Por el contrario, en el TD se busca utilizar antenas direccionales que permitan realizar la recuperación del MD tal como se describió en el Capítulo 1. Finalmente, en el caso del MD se busca un tipo de antena que se adapte al cuerpo de la tortuga sin sobresalir y que presente, en lo posible, un comportamiento omnidireccional.

Para el DCS se decidió utilizar la antena omnidireccional monopolo disponible en el grupo de investigación.

Para el TD se utilizará una antena tipo Yagi-Uda como se mencionó en el Capítulo 1, al permitir al sistema vincular cualitativamente el ángulo de arribo de la señal con la potencia recibida.

En el caso del MD se utilizará como antena un cable de longitud menor a 50 cm, al ser necesario emplear una antena flexible que se pueda colocar en el caparazón de la tortuga sin que sobresalga del mismo.

Los resultados de la selección de las antenas evaluadas serán expuestos en el Capítulo 5.

3.2.11. Módulo Bluetooth

Para posibilitar la implementación de la interfaz del TD, se decidió elegir el módulo Bluetooth JBtek HC-05. El mismo posee un excelente rendimiento, provee comunicación bidireccional y se encuentran varios ejemplos de aplicación en la web, junto a librerías ya desarrolladas, además de encontrarse disponible en el mercado local.

3.3. Diseño del PCB

En esta sección se detallará el diseño del PCB a partir del diagrama de bloques (figura 2.1) mencionado en el Capítulo 2 y la elección de los componentes realizada en este capítulo, mencionando los criterios y lineamientos considerados a la hora de llevar a cabo el diseño de la placa de circuito impreso.

3.3.1. Diseño de circuito esquemático

Previo a continuar con los detalles del diseño del PCB, se describirán algunos circuitos adicionales agregados de modo de asegurar una correcta interacción entre los diferentes integrados elegidos.

Filtro de receptor GNSS

Debido a que los receptores GNSS son sensibles a la interferencia electromagnética (como la que puede ser generada por el microcontrolador u otros dispositivos), se añadieron un filtro tipo L-R-C para la alimentación del módulo, junto a filtros tipo R-C para las líneas de UART, permitiendo funcionar a una tasa de transferencia de 9600 baudios, siendo el valor estándar utilizado para este tipo de módulos y suficiente para esta aplicación. En la figura 3.4 se observa el diseño de estos filtros.

Medición de tensión de batería y corriente de carga

Debido a que la batería estará conectada a través de una llave y una fuente *switching* al MCU, y el pin ADC utilizado para la adquisición del nivel de batería se encuentra conectado directamente al MCU, puede ocurrir que el mismo no reciba alimentación mientras la batería aún tiene tensión. Al ser la máxima tensión soportada por el MCU cuando no recibe alimentación de 0,3 mV, valor considerablemente menor al mínimo de tensión de la batería, puede ocurrir que el MCU se dañe al estar el pin del ADC conectado directamente a la batería.

Para medir esta tensión sin que implique riesgo de dañar al MCU, se añadió un divisor resistivo (figura 3.5) de 1,5 MΩ de impedancia, lo que implica una corriente de fuga de 2,8 μA, siendo un valor de consumo despreciable. Este divisor escala el máximo

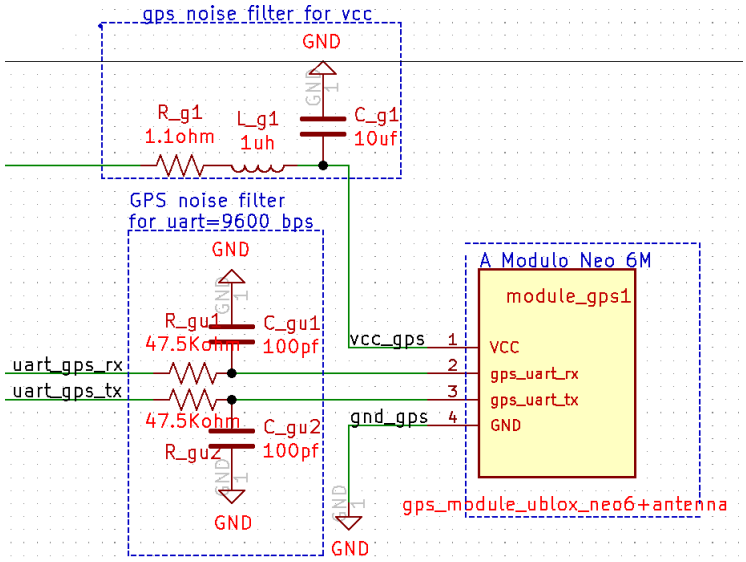


Figura 3.4: Filtros diseñados para módulo receptor GNSS.

valor de tensión de la batería (4,2 V) a 0,25 mV, evitando que pueda dañar el MCU en cualquier circunstancia.

Se diseñó un divisor similar para la medición de la corriente de carga generada por el integrado cargador de batería siguiendo el mismo principio.

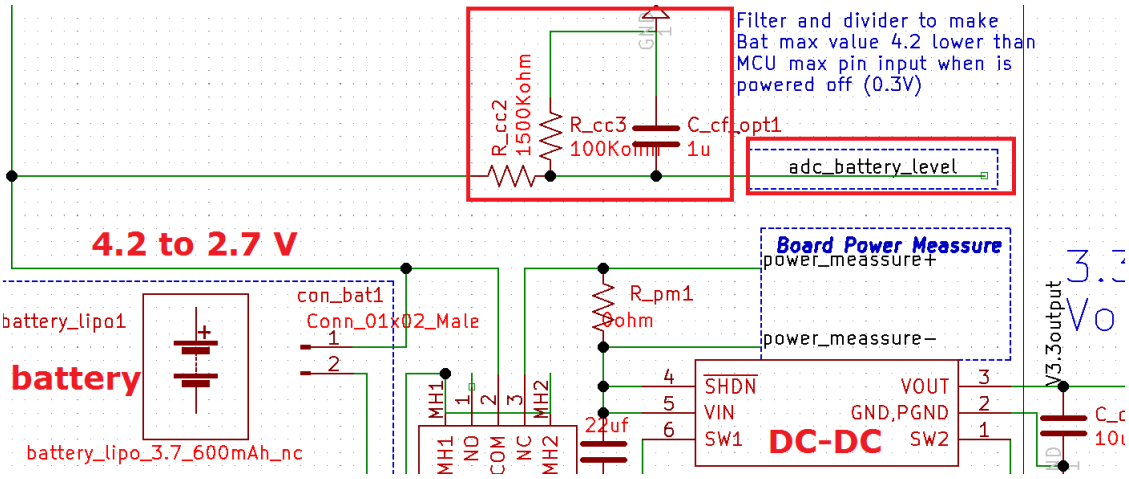


Figura 3.5: Divisor diseñado (dentro de rectangulo rojo) para implementar la medición de la tensión de la batería en forma segura.

Puntos de medición

Para facilitar la pruebas, verificaciones y depuración del dispositivo, se colocaron diferentes puntos de medición que permiten la conexión de instrumental de medición. Los puntos son:

- Tensión de la batería.
- Corriente de carga de la batería.

- Señales de *bus* I2C.
- Señales de *bus* I2S.
- Señales de interrupción de la IMU.
- Puestas a tierra en diferentes puntos de placa.

3.3.2. Esquemático

En la figura 3.6 se puede observar el esquemático diseñado utilizando el software KiCad, en el cual se encuentran los diferentes conjuntos de componentes asociados a cada sensor y otros integrados importantes como la fuente *DC-DC* y el cargador de la batería. Esta separación de los componentes en grupos facilita el diseño del PCB, como se verá a continuación.

3.3.3. Posicionamiento de módulos

Previo a realizar el diseño del PCB en su totalidad, se diseñaron primero las diferentes partes del PCB correspondientes a cada módulo (en este contexto, corresponde a los integrados mencionados junto a sus componentes necesarios para funcionar), con el fin de tener una estimación del tamaño real de cada uno. Luego se utilizaron los tamaños estimados para plantear posibles ubicaciones de los módulos dentro de la placa, de forma que resulten funcionales a las necesidades del grupo de investigación. En el diseño se utilizaron componentes de montaje superficial (SMD) de tamaño de empaquetado 0805 (tamaño 1.25 x 2 mm) siendo un tamaño adecuado para realizar modificaciones manuales al *hardware*.

Las consideraciones para el posicionamiento de los módulos fueron las siguientes:

- El acceso a los leds de visualización y memoria microSD debe estar del lado trasero de colocación en el animal.
- Es deseable que la IMU esté posicionada cerca de la cabeza del animal, de modo que la misma sea sensible a movimientos cercanos al cuello.

En la figura 3.7 se puede observar como se posicionaron los módulos a partir de los criterios mencionados y tras varias iteraciones realizadas con los integrantes del grupo de trabajo. El tamaño estimado fue de 60 x 55 mm.

3.3.4. Diseño de PCB

A partir de la selección del posicionamiento de los diferentes módulos, se realizó el diseño del PCB tomando en cuenta las siguientes consideraciones:

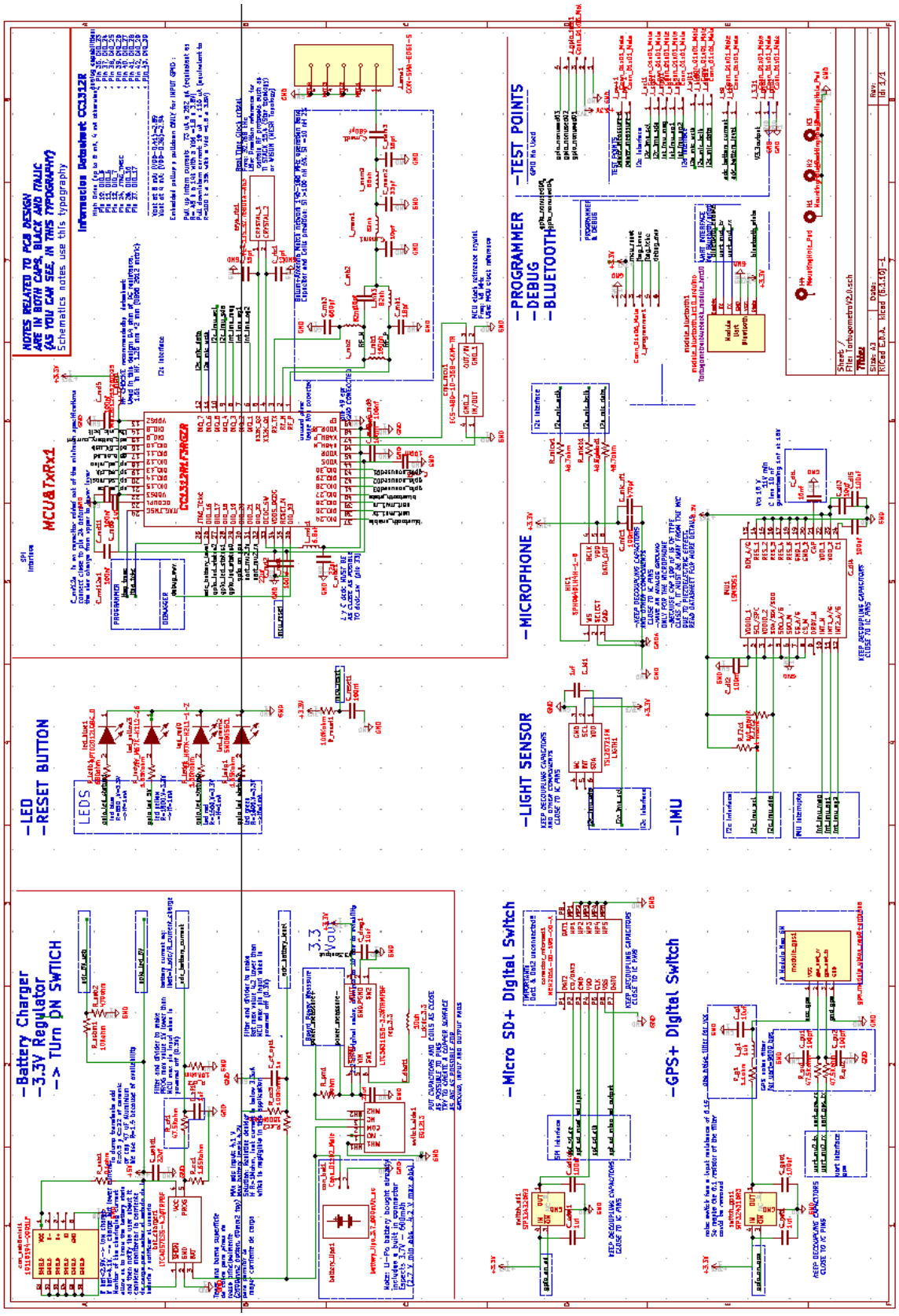


Figura 3.6: Esquemático de la familia de dispositivos de monitoreo animal.

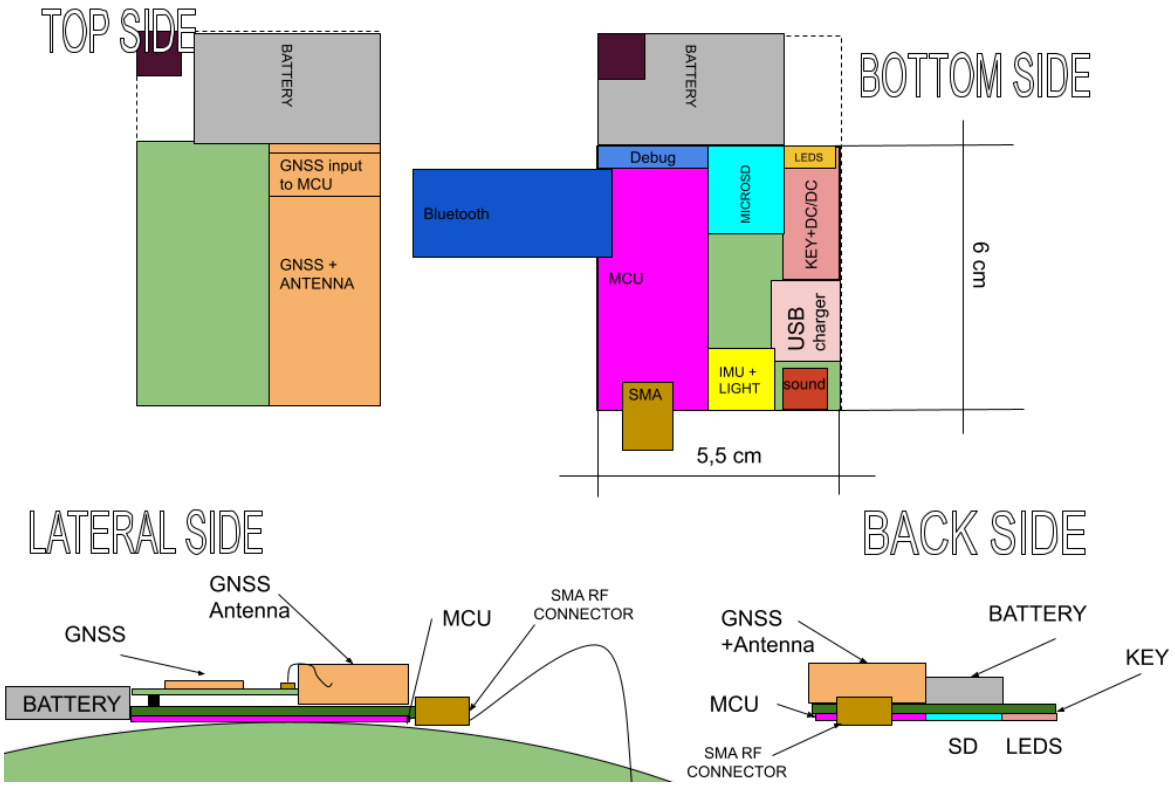


Figura 3.7: Imagen ilustrativa de posicionamiento de los diferentes módulos en el PCB.

- Se diseñó utilizando 2 capas al ser más económico para la fabricación.
- El plano de masa se encuentra solo del lado posterior del dispositivo, sin que se suelden componentes de montaje superficial de ese lado del PCB.
- El tamaño de pista mínimo fue de 0,15 mm. Es importante mencionar que el tamaño de la pista fue incrementado en las secciones donde el consumo de corriente es mayor, como es el caso del receptor GNSS o el cargador de batería, para disminuir la resistencia de la pista.
- El ruteo se realizó de modo de evitar el cruce de señales de alta velocidad o generar divisiones en el plano de masa.
- La red de adaptación del transceptor fue implementada siguiendo el diseño obtenido a través del foro de consultas técnicas de Texas Instruments, garantizando la adaptación tanto para transmisión como recepción en una carga de 50 Ω dentro de la banda 143 a 176 MHz. Es importante mencionar que Texas Instruments no provee los valores de impedancias de los terminales de transceptor para la frecuencia de operación utilizada en este proyecto, ni los criterios utilizados para diseñar la red de adaptación, siendo esta información fundamental para poder modificar la misma y motivo por el cual se utilizó el diseño provisto por el fabricante.

- La sección de la placa correspondiente al transceptor RF debe tener un plano de masa continuo debajo de donde van colocados los componentes. Además incluye el *footprint* de un conector de RF SMA, de modo de permitir la conexión de antenas en los casos del TD y DCS.
- Se añadió, entre la batería y la fuente *DC-DC*, una resistencia en serie con la fuente de modo que se pueda medir el consumo de corriente del dispositivo.
- Los pines GPIO no utilizados (siendo un total de 3) fueron ruteados junto a un terminal de masa y de tensión a un zócalo, para poder utilizarlos de ser necesario.

En la figura 3.8 se puede observar en la cara donde van montados los componentes. El tamaño total obtenido en el diseño final fue de 48,7 × 63,7 mm.

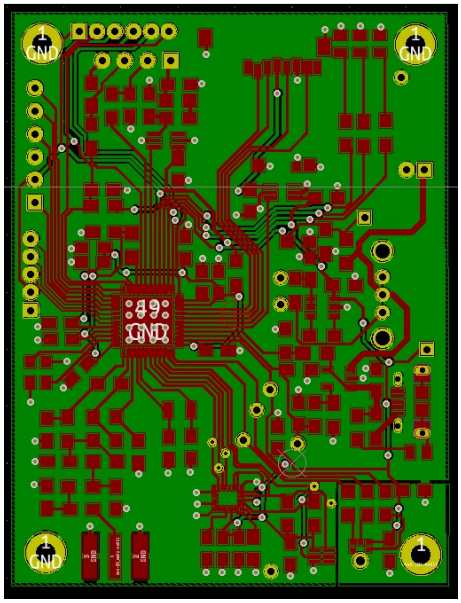


Figura 3.8: Diseño del PCB de la familia de dispositivos a utilizar.

Finalmente, en la figura 3.9 se puede observar el modelo 3D del diseño, donde se indica la ubicación de los principales elementos mencionados en este capítulo.

3.3.5. Elementos para la fabricación de cada dispositivo

A partir de la selección de componentes y de las diferencias en el funcionamiento de cada dispositivo integrante de la familia, se confeccionó la tabla 3.6 en la cual se puede observar qué elementos, ya sean módulos, integrados o antenas y baterías, forman parte de cada tipo de dispositivo. En la misma se observa cómo el mismo PCB es utilizado en los tres dispositivos, donde solo el MD requiere tener los sensores utilizados para la adquisición de datos, con la excepción del receptor GNSS que es utilizado tanto por el MD como por el DCS para que este último pueda registrar su posición al recibir las posiciones de los diferentes MD. Respecto a las interfaces, como se mencionó en

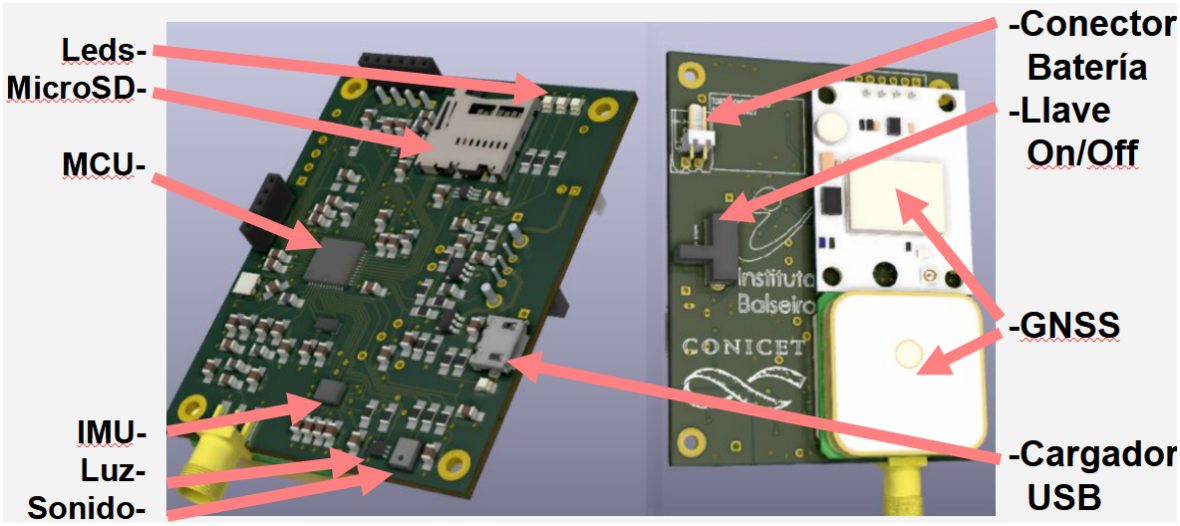


Figura 3.9: Modelo 3D del PCB donde se pueden observar los diferentes elementos del diseño.

el Capítulo 1 se observa que el TD utiliza un Bluetooth para implementar la interfaz mientras que para la DCS va conectado a una PC. Finalmente, ya que la DCS estará colocada en una posición fija durante el tiempo que se encuentre recolectando datos, se utilizará una batería de mayor capacidad, de modo que incremente su autonomía, y que estará conectada por el puerto USB del que dispone la placa de acuerdo a las especificaciones ya mencionadas.

Elementos	Dispositivo de monitoreo (MD)	Dispositivo rastreador (TD)	Estación colectora (DCS)
PCB	X	X	X
sensores	X		
Receptor GNSS	X		X
Bluetooth		X	
PC			X
Conector SMA		X	X
Tipo de antena	"Cable"	Direccional (Yagi)	Omnidireccional
Tipo de batería	LiPo 600 mAh	LiPo 600 mAh	Banco de carga USB de 10000 mAh

Tabla 3.6: Elementos asociados a cada tipo de dispositivo de acuerdo a sus funcionalidades.

3.4. Estimación de alcance

Para realizar la estimación del alcance se utilizó una hoja de cálculo provista por Texas Instruments, con los siguientes parámetros:

- Potencia de transmisión: 14 dBm.

- Ganancia en la antena transmisora de 0 dB. Ganancia en la antena receptora de 0 dB.
- Antenas en posición horizontal.
- Antena receptora a un metro de altura, antena transmisora a 10 cm.
- Modo de transmisión de largo alcance 5.2 kbps (-121 dBm de sensibilidad).
- Sin obstáculos entre receptor y transmisor.

El resultado del alcance mínimo esperado en condiciones ideales fue de aproximadamente 5.2 kilómetros.

Capítulo 4

Diseño de *firmware*

En este capítulo se profundizará sobre el diseño del *firmware* de la familia de dispositivos. Se entrará en detalle sobre los criterios de diseño y la forma en la que se implementan los mismos a partir del uso de un sistema operativo en tiempo real, como elemento principal en el soporte de la interacción entre las diferentes funciones del sistema.

Es importante recordar que el *firmware* es el código que deberá ejecutar el microcontrolador para llevar a cabo las diferentes funciones del dispositivo. En el caso del MD por ejemplo, deberá comunicarse con los diferentes sensores que componen el *hardware* para configurarlos y almacenar los datos en la memoria microSD. A su vez también deberá monitorear el estado del dispositivo (por ejemplo, a través de la medición del nivel de batería) y notificar al usuario, por medio de las luces leds, de su correcto funcionamiento o eventos de interés, tal como un bajo nivel de batería.

4.1. Diagrama de bloques

En la figura 4.1 se puede observar una versión reducida del diagrama general del sistema (figura 2.1) expuesto en el Capítulo 2. En él se muestran los diferentes elementos de *hardware* del sistema comunicándose a través de diferentes protocolos. En este caso, los canales de comunicación requeridos son uno de SPI, I2C y I2S, dos canales de UART y tres entradas de ADC.

4.2. Características y elementos del *firmware*

4.2.1. Criterios de diseño

Es importante mencionar que varios de los criterios y decisiones de diseño se basaron en el libro "*Design Patterns for Embedded Systems in C*" de Douglass [30]. A

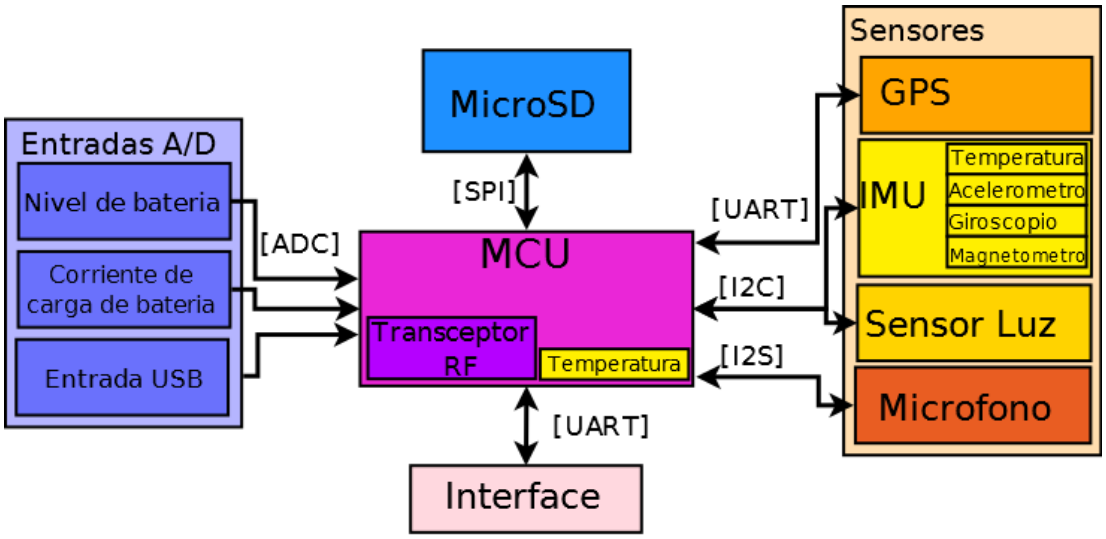


Figura 4.1: Diagrama de bloques donde se muestran los diferentes módulos interactuando entre sí y los protocolos utilizados para comunicarse con cada periférico.

continuación se listan los criterios aplicados en el diseño del firmware [30]:

- Escalabilidad y diseño modular: la arquitectura deberá permitir la habilitación/ inhabilitación de funciones y tareas (como, por ejemplo, qué tipos de sensores se utilizarán para adquirir un determinado tipo de muestra) de forma que no requiera una reescritura completa del código, buscando utilizar una filosofía de encapsulamiento para los diferentes elementos del sistema.

- Migración: es deseable que el *firmware* se diseñe de modo que permita realizar una migración a otros sistemas integrados (SoC) de la forma más fácil posible. Por este motivo, se busca utilizar capas de abstracción tanto para el sistema operativo en tiempo real como para los diferentes módulos y *drivers* del sistema.

- Bajo consumo: permitir el uso y manejo de diferentes modos de bajo consumo de los elementos que componen el sistema.

- Identificación: la información almacenada en la memoria del dispositivo debe contener suficiente información como para identificar a qué dispositivo pertenece y permitir la conversión de los datos almacenados de los sensores en formato binario a decimal.

Antes de profundizar en cómo se implementarán las funciones del *firmware*, primero se mencionarán los elementos, funciones y herramientas que permitirán su implementación.

4.2.2. Sistema operativo en tiempo real

Un sistema operativo en tiempo real (RTOS) es un tipo de sistema operativo que se utiliza en aplicaciones (como sistemas de adquisición, computadoras de automóviles o sistemas de soporte vital) en las que el tiempo en que se realizan los procesos o tareas del sistema es un factor importante o crítico. Se busca que las mismas se ejecuten garantizando un plazo de tiempo determinado. El uso de estos sistemas permite manejar múltiples tareas y prioridades, posibilitando que las aplicaciones se ejecuten de manera más eficiente y predecible. En este contexto, una tarea o hilo es una unidad de trabajo independiente que se ejecuta en el RTOS [31–33].

Características de TI-RTOS

El principal RTOS soportado por el MCU es TI-RTOS, desarrollado por Texas Instruments, el cual ofrece una amplia gama de características y funcionalidades para el desarrollo de sistemas embebidos, siendo algunas de sus características distintivas las siguientes [34–36]:

- Gestión de recursos y bajo consumo: TI-RTOS proporciona un conjunto completo de librerías para gestionar y controlar los recursos de *hardware* tales como GPIO, UART, ADC, SPI, etc. A su vez, permite la creación de políticas de consumo (denominadas *PowerPolicy*), las cuales automatizan la selección del modo de consumo del procesador en función de qué recursos de *hardware* se encuentran activos. Un ejemplo de políticas de consumo consiste en seleccionar el modo de consumo entre los disponibles: si *standby* (2 μA) o *idle* (500 μA), en el que entrará el MCU cuando el RTOS no encuentre más tareas para ejecutar. Es importante mencionar que cuando algunos *drivers* están activos, como la radio RF, el MCU no puede entrar en ciertos modos de bajo consumo ya que podrían generar un mal funcionamiento del *driver*.

- Comunicación entre tareas: TI-RTOS proporciona mecanismos de comunicación ínter-tarea para que las mismas puedan compartir datos e información, como semáforos o *mailboxes*.

- Soporte de *drivers* de dispositivos: TI-RTOS cuenta con una amplia variedad de controladores de dispositivos precompilados, lo que facilita la integración de diferentes dispositivos periféricos, como memorias microSD o *flash NAND*, sensores de luz o de sonido.

- Integración con herramientas de desarrollo: TI-RTOS está integrado con el entorno de desarrollo *Code Composer Studio* de Texas Instruments, lo que facilita el proceso de desarrollo y depuración.

- Compatibilidad con POSIX y FreeRTOS: TI-RTOS es compatible con el estándar POSIX, siendo ésta una capa de abstracción para sistemas operativos utilizados en varios lenguajes de programación en ordenadores. El uso de esta capa también es compatible con el renombrado RTOS FreeRTOS [37, 38].

Capa de abstracción e implementación

Es importante mencionar que TI-RTOS es un RTOS que es compatible sólo con los MCU de Texas Instruments, por lo que los desarrollos que usen este sistema operativo solo serán compatibles con productos de este fabricante. Además, la capa de abstracción provista por el fabricante (POSIX) no permite implementar funciones en el planificador, lo cual es de fundamental importancia para optimizar el consumo en proyectos de ultra bajo consumo. Para facilitar la migración de este proyecto a otros tipos de SoCs se desarrolló una capa de abstracción que se basa en utilizar funciones de igual nombre (como con las funciones de retardo) que las correspondientes en FreeRTOS. Esto se debe a que FreeRTOS es uno de los RTOS más utilizados y cuenta con una gran cantidad de algoritmos de planificación implementados disponibles.

4.2.3. Interacción entre tareas del RTOS

Como se mencionó al principio de este capítulo, se buscó que las tareas funcionen de forma modular e independiente. Para poder asegurar que las tareas del RTOS interactúen correctamente entre sí, se utilizaron semáforos entre las mismas y una máquina de estados.

Un semáforo es una herramienta de sincronización que se utiliza en los sistemas operativos para garantizar el acceso exclusivo a recursos compartidos entre procesos o tareas, lo cual previene que ocurran fallas en el programa. Cuando una tarea intenta acceder a un recurso compartido, debe verificar que el mismo esté libre a través del semáforo, y de estarlo, tomar el recurso. De haber más de una tarea que intente acceder, deberán esperar a que la tarea que tomó el recurso lo libere para que esté disponible nuevamente. Es importante mencionar que este mecanismo fue implementado cuidando que la interacción de tareas por medio de semáforos, evite la ocurrencia de bloqueos o *starvation* de una tarea, al depender de la liberación de un recurso compartido.

En la figura 4.2 se pueden observar de forma resumida las diferentes tareas del sistema interactuando entre sí. Es importante mencionar que el uso de semáforos permite, por ejemplo, crear listas de espera para gestionar el acceso de diferentes tareas del sistema a recursos compartidos, como es la unidad de memoria microSD. A su vez, la máquina de estados, la cual explicaremos en el próximo punto, posibilita recolectar información de diferentes elementos del sistema (como del receptor de sistema global de navegación por satélite (GNSS) o del estado de la memoria microSD) y permite

que otras tareas puedan conocer el estado de estos sensores y actúen acorde, como es el caso de la tarea que maneja las luces leds, permitiendo indicar al usuario el resultado de estas verificaciones. En las próximas secciones se desarrollarán las tareas que conforman el *firmware* y su interacción.

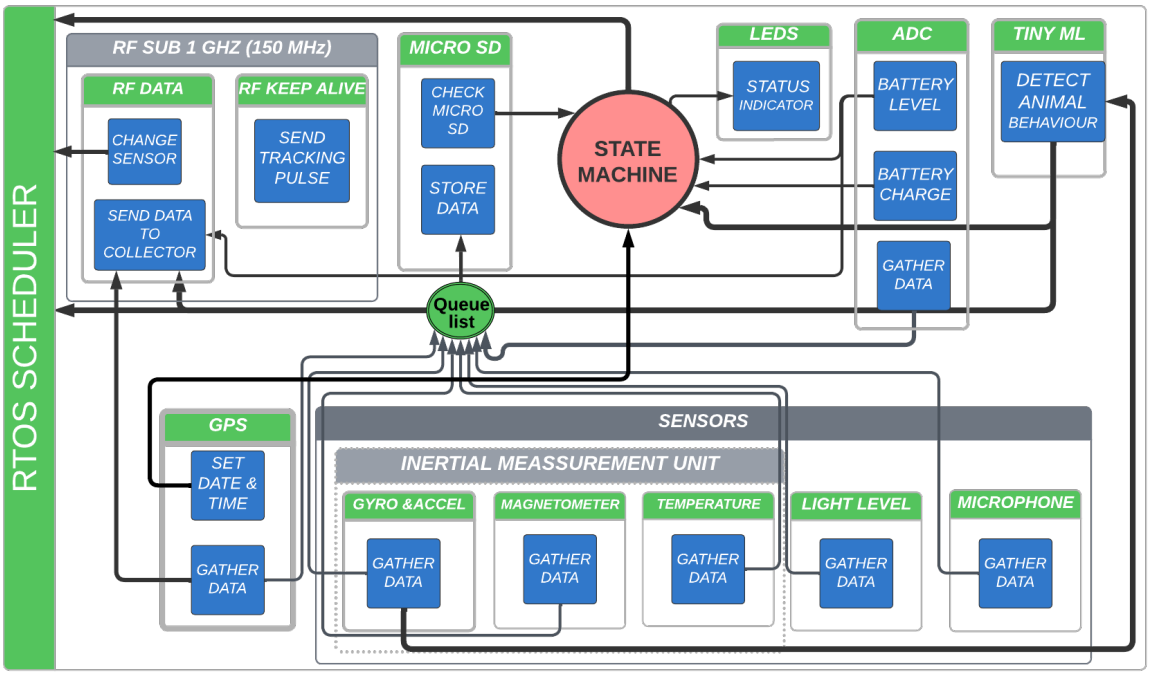


Figura 4.2: Interacción entre los diferentes elementos del *firmware* utilizando un RTOS y máquina de estados.

4.2.4. Máquina de estados

Una máquina de estados es un modelo matemático que se utiliza para representar el comportamiento de un sistema en términos de estados, transiciones y eventos. En una máquina de estados, el sistema se modela como un conjunto finito de estados y las transiciones entre dichos estados, que se activan por eventos externos o internos. Por lo cual, cada estado representa una condición o modo de operación del sistema, mientras que las transiciones representan el cambio de un estado a otro debido a algún evento o entrada, como pueden ser valores de sensores. La máquina de estados permite llevar un registro de los estados de interés del dispositivo, de modo que todas las tareas puedan conocerlos y actuar acorde. Una extensión al formalismo convencional de las máquinas de estado finitas es *StateCharts*, un lenguaje de modelado gráfico que agrega características adicionales al modelo básico de las máquinas de estado finitas, permitiendo representar de manera más efectiva el comportamiento y la interacción de los elementos del sistema. Este paradigma permite organizar los diferentes estados en jerarquías, asociar acciones y condiciones a las transiciones de los mismos, y modelar

la concurrencia y la comunicación entre partes del sistema. El uso de máquinas de estados basadas en este paradigma permite describir comportamientos complejos de manera clara, así como mostrar el diseño y facilitar la comprensión y la verificación del sistema.

En la figura 4.3 se observa el diagrama de estados. El mismo fue creado utilizando el *software Yakindu StateChart Tools*. Esta herramienta permite crear máquinas de estados de forma visual, donde se conectan diferentes bloques y los eventos que generan las transiciones de estado diagramados. La máquina luego puede exportarse en lenguaje C, C++ o Python. Así, el usuario puede incluirla en su proyecto y alimentarla con los eventos definidos en el diagrama para generar las transiciones correspondientes entre estados. El uso de esta herramienta facilita el mantenimiento y entendimiento de la máquina de estados [39].

Para favorecer el uso de la máquina de estados en el RTOS, se añadió una capa de abstracción que integra la misma con funciones del RTOS y funciones de luminosidad de los tres leds. De este modo, si se modifica la máquina de estados, basta con importar la misma desde el entorno provisto por *Yakindu StateChart Tools* y agregar las funciones deseadas a la capa de abstracción.

Además, el uso de la máquina de estados permite llevar un registro de los diferentes estados de interés del dispositivo (figura 4.3). El primer chequeo que se realiza es sí el dispositivo se encuentra (o no) conectado a una fuente de alimentación, para alertar al usuario de que se está realizando una carga de la batería. En este caso, el dispositivo se mantendrá en este estado hasta que la batería este llena.

De no estar conectado, se verifica si el nivel de batería está por encima del nivel mínimo requerido para funcionar correctamente. Si esta comprobación es correcta, se procede a realizar una verificación de la memoria microSD conectada, intentando crear un archivo dentro, escribir contenido en él y luego borrarlo. Si esta verificación resulta satisfactoria, el dispositivo enciende el receptor GNSS y aguarda a que el mismo logre obtener una localización valida o "*fix*", proceso que puede demorar hasta 20 minutos dependiendo de las condiciones del entorno. Si el receptor logra un estado de *fix*, se produce la transición al estado de adquisición, estado *ADQ_DATA*, en el cual cada sensor comienza a adquirir datos. Durante este proceso una de las tareas del sistema (*System Monitor*) se ocupará de verificar el estado de la batería regularmente. Si se detecta que el nivel de la batería disminuye a un valor por debajo de cierto umbral, induce un cambio de estado a *BAT_LOW*. Mientras esto ocurre, las tareas periódicas verifican el estado actual de la máquina de estados, cambiando su comportamiento en función del mismo. En las próximas secciones de desarrollará más sobre este último punto.

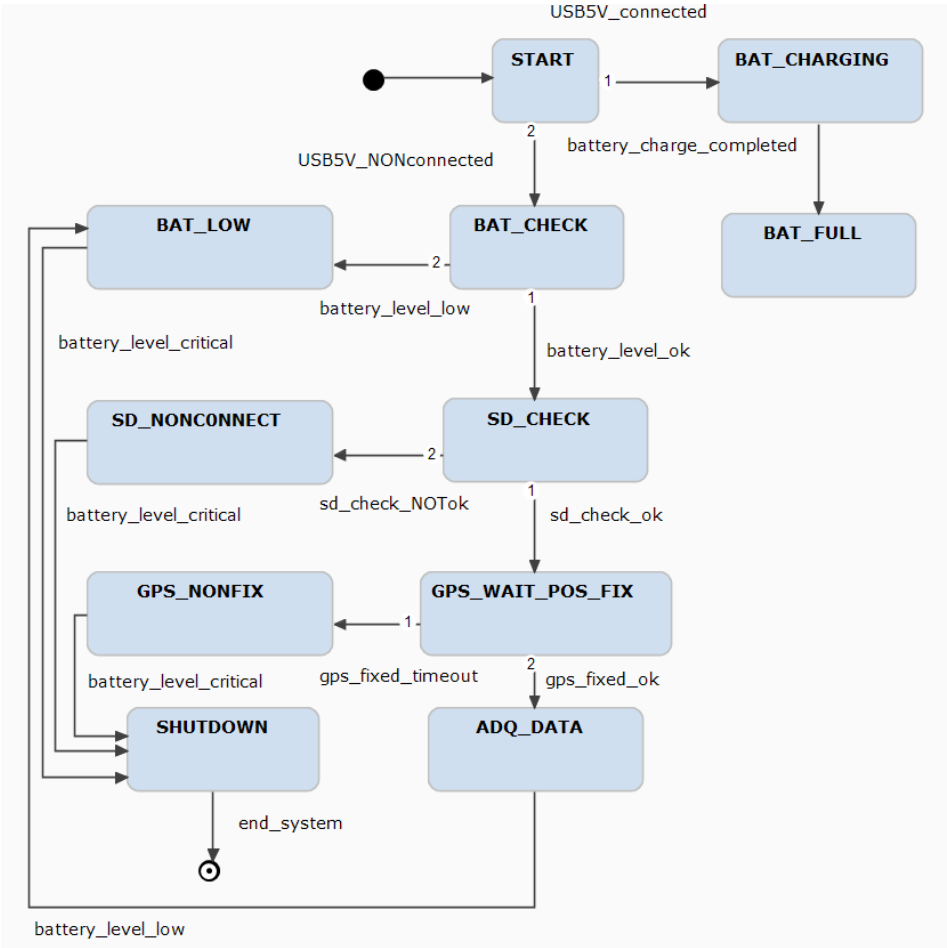


Figura 4.3: Máquina de estados del MD.

4.2.5. Modos de funcionamiento asociados al consumo

Antes de mencionar estos modos es importante entender cómo se asocian los valores de tensión al valor porcentaje de energía disponible en la batería, como se observa en la curva obtenida experimentalmente (figura 4.4). En ella se ve como el punto de caída donde la batería está cerca del 5 % es de 3.3 V, por lo cual se considerará 3.4 V como valor de batería baja.

Como se mencionó en el diagrama de estados, el valor de la tensión de la batería condiciona la actividad del dispositivo. Por este motivo, se definieron los siguientes modos de consumo asociados a la máquina de estados:

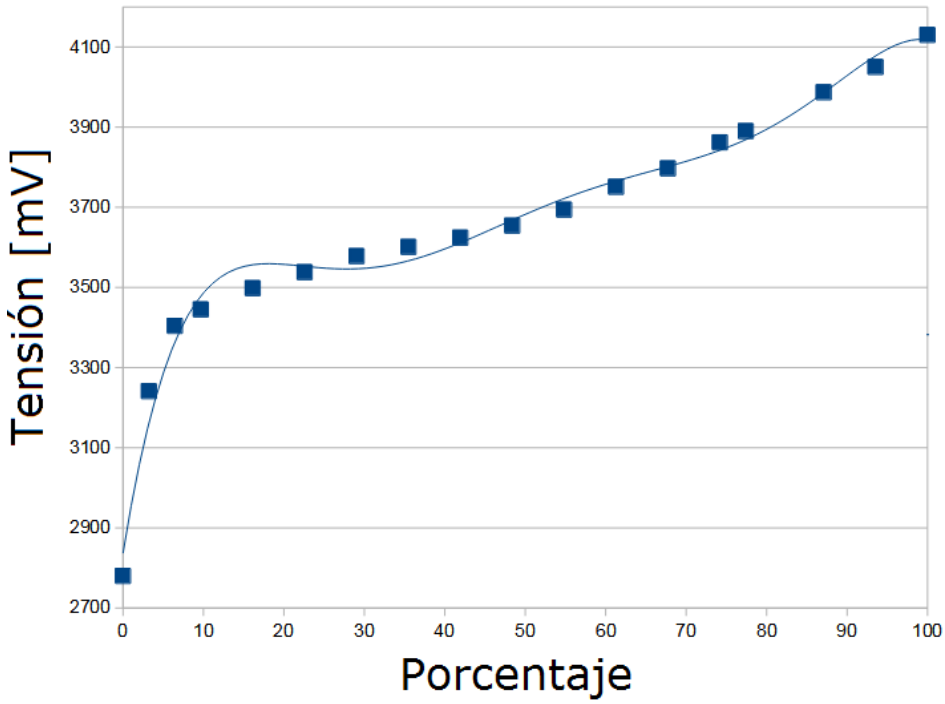


Figura 4.4: Curva de batería (tensión en el eje de ordenadas, porcentaje de energía en eje de abscisas) tipo LiPO de 3.7 V y 600 mAH, relevada experimentalmente.

- Nivel de batería correcto: la tensión de la batería es mayor a 3.4 V. El dispositivo actúa de acuerdo al diagrama de estados. Una vez alcanzado el estado DATA_ADQ, comienza a obtener y a guardar toda la información de los sensores.

- Batería baja (estado BAT_LOW): ocurre cuando el valor de batería es menor a 3.4 V, esto corresponde a aproximadamente el 5 % de una batería tipo LiPO. El dispositivo apaga todos sus sensores y se limita únicamente a enviar mensajes de radiofrecuencia para poder ser localizado por medio de un TD.

- Apagado (Estado *SHUTDOWN* en la figura 4.3): el dispositivo se apaga por completo debido a que la batería ha alcanzado un nivel crítico. En la batería

utilizada, este valor es de 2.7 V, por lo cual de decidió tomar 2.9 V como valor conservador.

Además de estos modos vinculados al valor de la batería, se presentan dos modos de optimización de consumo utilizando un algoritmo de detección de movimiento a partir de las señales obtenidas por el acelerómetro para determinar entre dos tipos de actividad animal:

- Animal en movimiento: se utilizan todos los sensores para realizar la adquisición.
- Animal en reposo: si este estado permanece por más de 20 segundos, se apaga el giróscopo (al ser el segundo sensor de mayor consumo) y solo se deja encendido el acelerómetro y el magnetómetro para realizar el monitoreo del animal. Si este estado permanece por más de 10 minutos, no se enciende el receptor GNSS, al tener un consumo considerablemente mayor que los demás sensores y debido a que no es necesario obtener una geolocalización nueva al no haber cambiado desde la última adquisición (hace 10 minutos) debido a que el animal se mantuvo en reposo. El uso de esta estrategia permite optimizar considerablemente el funcionamiento del sistema, al apagar los sensores de mayor consumo cuando el animal no se encuentra en movimiento.

4.2.6. Estructura de memoria y almacenamiento

Para poder estandarizar la forma en la que se almacenan los datos en la memoria microSD, y por lo tanto, estandarizar su lectura, se diseñó una estructura de datos que es común para todos los *buffer* de almacenamiento, como se puede observar en la figura 4.5. Estos datos son volcados en un archivo .DAT, que contiene tanto datos en binario como datos en formato *ASCII*, con el fin de facilitar su lectura y la depuración del *firmware* en etapas de desarrollo.

Figura 4.5: Elementos de la estructura de datos utilizada para almacenar los valores adquiridos.

La estructura consta de los siguientes elementos:

- *Header*: contiene información general del tipo unidad de datos a almacenar. Este elemento es común a todas las estructuras de almacenamiento. En la figura 4.6 se pueden observar sus campos definidos en lenguaje C, siendo:
  - *id_head*: conformado por los dos caracteres $\backslash r \backslash n$. Estos caracteres indican el inicio de una escritura de datos.

- *id_name*: identificador de hasta 12 caracteres. Este identificador permite determinar de qué sensor o tarea fue extraída la información almacenada, siendo único para cada estructura de almacenamiento. Ej: $"IMU0M"$ corresponde al identificador de los valores de campo magnético obtenidos con el magnetómetro incluido en la IMU LSM9DS1.
- *block_size*: cantidad de bytes total de tamaño que tiene la estructura de memoria a almacenar en la microSD. Por convención con otros formatos de almacenamiento, se utilizan potencias de dos (como 512, 2048 o 4096) para indicar el tamaño del bloque.
- *data_written*: contiene la cantidad de elementos del arreglo de datos que fueron escritos en la memoria.
- *acq_time_start* y *acq_time_end*: representan los valores de tiempo del MCU para los cuales se adquirieron la primera y la última muestra almacenada respectivamente. Estos valores están expresados en cantidad de *ticks* del RTOS. Utilizando estos valores y la cantidad de datos (*data_written*), es posible reconstruir los valores de tiempo para cada muestra obtenida con aquellos sensores que adquieren información en forma periódica.

- *Config*: contiene los parámetros de configuración del sensor del cual se obtuvieron los datos a almacenar, tales como la tasa de muestreo, factor de escala de los datos binarios, etc. El tamaño de este campo es único para cada estructura de almacenamiento.

- *Data*: contiene un arreglo de tamaño fijo, donde cada elemento de ese arreglo corresponde con el tipo de estructura que contiene los datos en el formato perteneciente al sensor que los suministra, buscando que este formato sea del menor tamaño posible. Por ejemplo, en el caso de los valores adquiridos por el magnetómetro, la estructura de 1 dato corresponde a 3 números de 2 bytes cada uno correspondientes a los ejes X, Y, Z.

```
typedef struct {
    char                id_head[MEMMORY_STRUCT_START];
    char                id_name[MEMMORY_STRUCT_ID_SIZE];
    uint16_t            block_size;//
    uint16_t            data_written;
    uint32_t            acq_time_start;
    uint32_t            acq_time_end;
}mem_data_header;
```

Figura 4.6: Código de *header* de estructura de memoria.

En la figura 4.7 se puede observar un ejemplo de la estructura de los datos en formato binario almacenados correspondiente al sensor magnetómetro, cuyo tamaño corresponde a 2048 bytes (*IMU0M_BLOCK_UNIT_SIZE*), permitiendo almacenar hasta 310 muestras (*IMU0M_DATA_ARRAY_SIZE*) para los ejes X, Y, Z.

```
typedef struct {
    uint8_t              sample_rate;
    uint8_t              scale;
    char                 message_to_user[9];
}data_header_imu_mag;//6 bytes struct
//
typedef struct{
    mem_data_header        head;
    data_header_imu_mag    conf;
    struct  IMU_3xasis     d[IMU0M_DATA_ARRAY_SIZE];
}mem_imu0_mag;
//create an union of all to facilitate manipulation
typedef union {
    mem_imu0_mag           m;
    unsigned char          b[IMU0M_BLOCK_UNIT_SIZE];
}mem_imu0_mag_union;//union of all structures
```

Figura 4.7: Ejemplo de estructura de memoria para las muestras obtenidas con el magnetómetro.

Primera escritura de referencia

A su vez, para permitir una correcta decodificación de los datos, antes de escribir información de algún sensor, el sistema al iniciarse graba un mensaje en la microSD que contiene la siguiente información:

- ID del dispositivo: 8 bytes, correspondientes a la dirección MAC del SoC.
- Tipo de dispositivo y versión: MD, dispositivo rastreador(TD) o estación colectora (DCS), junto a versión de *hardware* y *firmware* asociada al tipo de dispositivo.
- Escala de *ticks* del RTOS, permitiendo convertir los valores de *ticks* a milisegundos.

4.2.7. Interfaces al usuario

Como se mencionó en la sección anterior, el sistema presenta tres modos de funcionamiento asociados al nivel de batería, junto a dos estados transitorios que suceden

mientras el sistema realiza la verificación de correcto funcionamiento de la unidad de memoria microSD y estado de *fix* del receptor GNSS. Para que el usuario del dispositivo pueda entender en qué estado actual se encuentra funcionando el dispositivo, se utilizaron las tres luces leds que contiene el *hardware*. Cada uno de los estados observados en la figura 4.3 tiene asociado un código de luces con estos leds, utilizando diferentes patrones y velocidades de parpadeo en cada luz para indicar el estado.

Además, el sistema cuenta con una interfaz tipo UART que es utilizada para ayudar en la depuración del *firmware* (Por ejemplo: al informar al usuario el valor de una variable de interés) y permitir al MD y DCS interactuar con otros dispositivos que manejen comunicación serial, tales como el módulo Bluetooth mencionado en el Capítulo 3 o una PC. También, cuando se enciende el dispositivo, se utiliza este puerto para enviar mensajes de verificación de arranque del *hardware* y sus elementos, indicando el estado general del sistema al iniciar.

4.3. Características de las tareas del RTOS

Antes de entrar en detalle sobre la implementación de las diferentes tareas del RTOS, se realizará una breve introducción a los parámetros fundamentales que definen a estás tareas, junto a la estructura general utilizada para su implementación. En la sección siguiente se explicará en mayor detalle como se aplican estos diseños para cada tarea del RTOS.

Parámetros importantes de las tareas

Las tareas o hilos pueden tener diferentes prioridades, tiempos de ejecución y tiempos de vencimiento, lo que permite al sistema operativo gestionar múltiples tareas simultáneamente y garantizar que se cumplan los requisitos de tiempo crítico. Estos parámetros se definen cómo[31]:

- Prioridad: es la importancia relativa de una tarea o hilo en comparación con otras tareas o hilos en el sistema. Las tareas con prioridades más altas se ejecutan antes que las tareas con prioridades más bajas. Por ejemplo, una tarea con prioridad 15 se ejecuta antes que una de prioridad 1.
- Periodo: intervalo de tiempo en que se debe llamar a la ejecución de la tarea.
- Tiempo de ejecución: cantidad máxima de tiempo necesario para que una tarea se complete.
- Tiempo de vencimiento: tiempo límite para que una tarea se complete desde la última vez que fue ejecutada. Si una tarea no se completa antes de su tiempo de

vencimiento, se considera que ha fallado. Dependiendo del tipo de tarea, el hecho de que falle puede tener diferentes consecuencias. Por ejemplo, si la tarea *IMU MAG*, de tiempo de vencimiento de 0,2 segundos, no se ejecuta en ese intervalo debido a que hay otra tarea de mayor prioridad ejecutándose y ocupando el procesador, entonces la tarea falla y la consecuencia es que se pierde una muestra del sensor.

- Periódica/Aperiódica: la tarea es periódica cuando tiene un periodo de ejecución fijo y estático, y aperiódica cuando este periodo es aleatorio en el tiempo.
- *Stack*: cantidad de memoria máxima asignada a la tarea en kB. Si alguna de las tareas sobrepasa el valor de stack asignado, puede provocar un fallo crítico en el RTOS, causando que el mismo colapse.

4.3.1. Estructura general de tareas de adquisición

En la figura 4.8 se puede observar en forma simplificada la estructura general utilizada para las tareas de adquisición:

- Inicialización: la tarea carga sus parámetros predefinidos en el código (como tasa de adquisición del sensor) e inicializa las variables.
- Verificación: se verifica el estado del sensor conectado, buscando detectar si funciona correctamente.
- Estado de adquisición: la tarea espera hasta que el estado *ADQ_DATA* sea alcanzado por la máquina de estados para comenzar a adquirir. Una vez en este estado, se realizan las siguientes acciones:
  - Adquirir: envía las instrucciones al sensor para adquirir los datos. Previamente debe tomar el semáforo que utiliza el *driver* de comunicación correspondiente. Por ejemplo, en la tarea del magnetómetro, como se accede utilizando el *driver* de I2C, primero se debe solicitar el semáforo asociado a ese *driver*.
  - Guardar: verifica si la cantidad de datos adquiridos está próxima a llenar el *buffer*. De ser así, almacena los datos en la memoria microSD utilizando un proceso no bloqueante.
  - Otros: incluye otras acciones a realizar por la tarea, siendo distintas según el caso. Por ejemplo, el cambiar configuraciones del sensor a partir de detecciones de movimiento.
  - Batería baja: verifica si la máquina de estados está en el estado *BAT_LOW*. Si la verificación es positiva, se realizan las siguientes acciones:

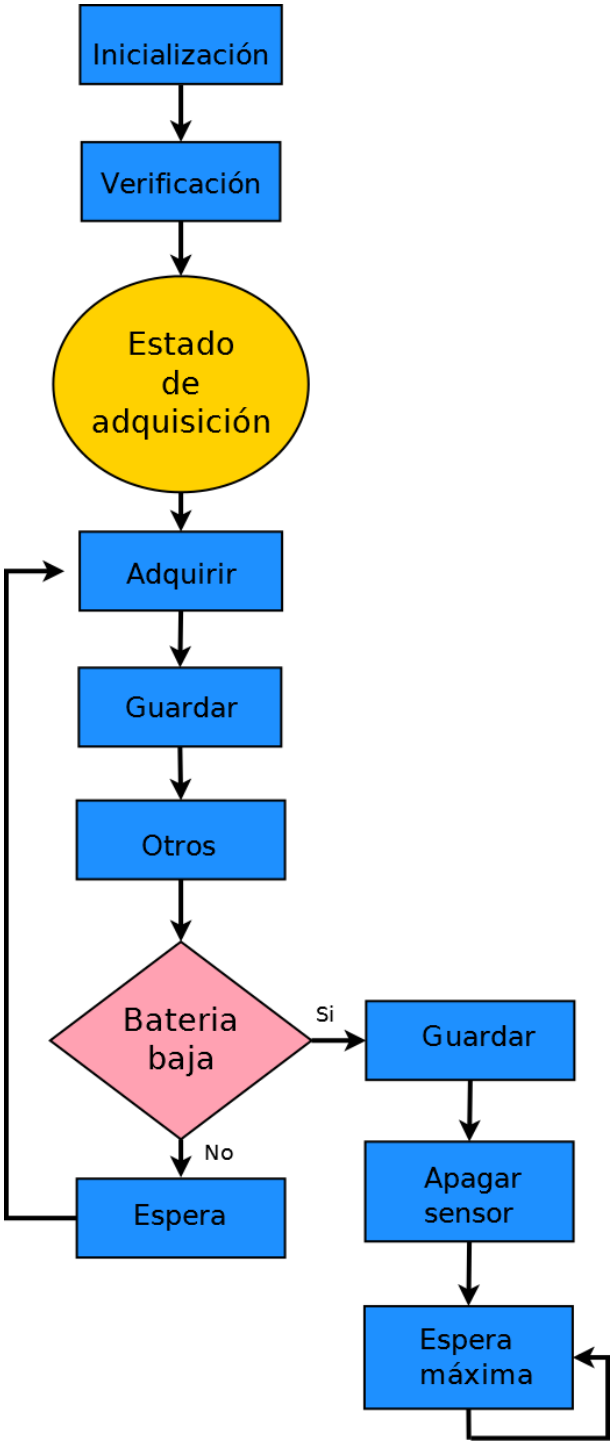


Figura 4.8: Estructura general de las tareas de adquisición utilizada en este proyecto.

- Guardar: almacena en memoria todos los datos adquiridos hasta el momento.
- Apagar sensor: envía las instrucciones al sensor para que se apague y deje de adquirir. A su vez, apaga los periféricos (como el I2C en el caso del magnetómetro) utilizados para su control.
- Espera máxima: una vez apagado el sensor, el sistema queda en un bucle en el cual la tarea queda en un estado de sueño o *sleep* por el máximo tiempo permitido por el RTOS, indicando al sistema operativo que no debe ejecutar esa tarea.

- Si la verificación es positiva, se realizan las siguientes acciones:
  - Espera: la tarea demora su ejecución hasta que haya alcanzado el periodo de tiempo desde la última vez que se ejecutó para continuar con el bucle principal.

Es importante mencionar que al crear tareas en un RTOS no se aconseja eliminarlas porque esto podría afectar el rendimiento y la confiabilidad del sistema. Esto se debe a que cuando una tarea es creada, se asigna un bloque de recursos específico para su ejecución, y si esta tarea se elimina, los recursos asignados quedan inutilizables hasta que se liberen explícitamente. Esto puede provocar la falta de recursos necesarios para otras tareas, lo que podría causar un comportamiento impredecible o incluso fallas en el sistema. Por este motivo, en el presente proyecto no se eliminan tareas, sino se que utilizan semáforos o demoras de tiempo para gestionar la activación o configuración de las mismas [40].

Finalmente, en la figura 4.9 se puede observar un ejemplo del código utilizado para el caso del magnetómetro, en donde se observan funciones correspondientes a varios de los diferentes bloques del diagrama 4.8. En la misma figura se puede observar cómo la tarea, antes de comenzar con la adquisición, verifica que la máquina de estados se encuentre en estado de adquisición, el cual es alcanzado una vez que el sistema ha realizado las verificaciones mencionadas en la sección 4.2.4. La consulta al estado del *StateChart* fue implementada a través de una librería creada para este proyecto que encapsula la máquina de estados e implementa el uso de funciones del sistema operativo como semáforos o demoras, permitiendo una interacción fácil y segura con el *StateChart*. Una consideración importante es que la función *VTaskDelayUntil* permite introducir un retardo de un número de *ticks* específicos del RTOS en la ejecución, a diferencia de la función *VTaskDelay*, que introduce un retardo fijo que puede variar, sin contemplar lo que las otras tareas del sistema puedan demorar el momento en el que se la llama. Esto ayuda significativamente a que la tarea se ejecute y adquiera las muestras cada un periodo regular.

```
 state_machine_wait_for_state(state_machine.SM_ADQ_DATA);
//--------------
 start_data_acquisition(&buffer_data_mag,&imu);
 while (1) { //adquire data and stores into global buffer
            acquire_data(&buffer_data_mag,&imu);
            //----------------------//store data
            store_data(&buffer_data_mag,IMU0M_DATA_LIMIT);
            //check to continue if data acq should continue
            if(state_machine_is_lowpower()){break;}//exit p
            //-----------------delay until time of acq beg
            vTaskDelayUntil(&xLastWakeTime, xFrequency);//d
           }
 //end of task. If this point is reached, -->store all the
 memory_store(&buffer_data_mag);
 stop_data_acquisition(&imu);
```

Figura 4.9: Parte principal del código para tarea de adquisición de magnetómetro.

4.3.2. Estructura general de tareas no periódicas

En el contexto de un RTOS, se denomina a una tarea como no periódica cuando no es llamada en forma regular o se llama en tiempos aleatorios. Un ejemplo de esto es la tarea utilizada para el almacenamiento en la unidad microSD, ya que es llamada por diferentes tareas en diferentes tiempos. Nos referiremos a este caso en particular como ejemplo general de la estructura de las tareas no periódicas.

Para poder entender por qué el almacenamiento en microSD es implementado como una tarea y no como una función, es importante analizar como trabajan las tareas al momento de compartir recursos. Cuando una tarea toma un recurso compartido (como el *driver* SPI) lo hace a través de un semáforo, el cual debe ser tomado y luego liberado por la misma tarea. Si múltiples tareas intentan hacer esta acción, lo que sucede es que su ejecución se ve demorada o bloqueada hasta que logran tener acceso al recurso. En el caso de funciones como la escritura en microSD, que puede ser llamada por múltiples tareas, puede implicar demoras demasiado prolongadas en la ejecución de las tareas de adquisición, consecuencia que podría llegar a provocar la perdida de datos a adquirir. Para evitar este tipo de situaciones, en lugar de implementar el proceso como una función, se implementó como una tarea aperiódica, cuya ejecución es habilitada a través del llamado de una función que no bloquea a la tarea que la llama. Por este motivo, tareas como la de escritura en microSD fueron implementadas como tareas no periódicas, permitiendo que sean llamadas por otras tareas sin que esto bloquee la actividad de la tarea llamante. Esto permite que las tareas de adquisición continúen adquiriendo mientras la tarea de memoria se ocupa de grabar la información en la microSD.

En la figura 4.10 se puede observar un diagrama resumido donde se implementa este mecanismo. La tarea aperiódica realiza el proceso de inicialización cuando el planificador del sistema operativo es activado, y entre estas tareas, toma el semáforo de

ejecución de sí misma y espera indefinidamente a que sea liberado. Cuando la tarea periódica desea realizar una acción llama a una función dentro de su bucle de ejecución. Esta función toma el semáforo asignado al *driver* (en el caso de la escritura en memoria, es el *driver* SPI). Luego realiza una copia de la información a comunicar (en este caso, los datos a almacenar del sensor) al *buffer* propio de la tarea aperiódica (siendo el *buffer* de la tarea encargada de escribir en la microSD) y libera el semáforo de Ejecución. Esto se puede ver en la flecha y letra A en el diagrama de la figura 4.10.

Al liberar este semáforo, el planificador detecta que el semáforo de ejecución asignado a la tarea aperiódica está libre y que la tarea misma está intentando tomarlo. Al suceder esto, la tarea aperiódica realizará sus acciones (en este caso, es escribir la información copiada en el *buffer* de la tarea en la memoria microSD), para luego liberar el semáforo asignado al *driver* (flecha con letra B en la figura 4.10), indicando que ha completado sus acciones y permitiendo que otras tareas llamen a la función asignada a la tarea aperiódica (de escritura en microSD, siguiendo este ejemplo) o utilicen el recurso compartido (en este caso, el *bus* SPI). Como la tarea aperiódica no libera el semáforo de ejecución después de tomarlo, la misma vuelve al estado inicial, donde deberá esperar que otras tareas llamen a la función correspondiente para habilitar el mecanismo.

Es importante mencionar que cuando la función llamada por la tarea se completa, la misma continua realizando sus demás acciones (en el caso de la tarea correspondiente al magnetómetro, continúa adquiriendo datos). Dependiendo de las tareas que estén en lista de espera de ejecución y el orden de prioridades de las mismas, en algún momento el planificador ejecutará la tarea aperiódica (en este caso, la escritura en microSD). Esto también implica que si otra tarea desea escribir en memoria en este punto, deberá esperar a que la primera escritura se complete.

4.4. Tareas del sistema operativo

En esta sección se mencionarán las diferentes tareas que componen el *firmware*, junto a detalles significativos de como interactúan entre sí. Es importante mencionar que para facilitar la implementación del sistema operativo y la gestión de las tareas, se creó una estructura de datos junto a la capa de abstracción del RTOS que almacena diferentes parámetros de interés (periodo de la tarea, prioridad, tamaño de stack, posición en memoria de stack, función de ejecución, etc.) para cada tarea. Esta estructura es propia de cada tarea y solo puede ser accedida por sí misma o por el sistema operativo. El uso de este tipo de estructuras permite facilitar y sistematizar la forma en la que se inician o configuran las tareas.

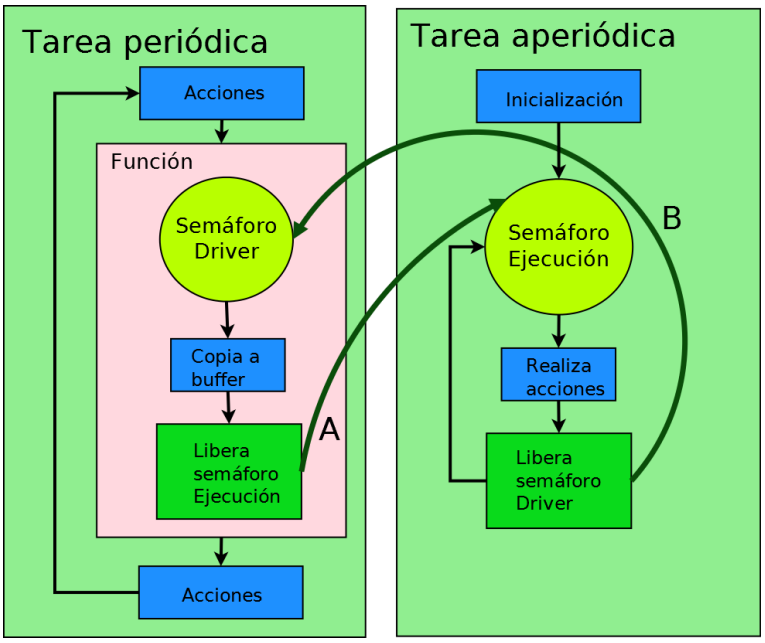


Figura 4.10: Estructura general de tareas aperiódicas y su interacción con una tarea periódica.

Inicialización del sistema operativo

A continuación se enlistan los pasos a realizar por el sistema operativo. El resultado de la instancia de verificación es notificado al usuario utilizando el puerto UART.

1. Se toman los parámetros de las tareas definidos (prioridad, periodo, tiempo de vencimiento) y se asignan a sus estructuras.

2. Se pasan los punteros de todas las estructuras a un arreglo, para permitir la iteración de las estructuras de las tareas.

3. Utilizando este arreglo se comienza a iterar una tarea. Se asigna un lugar de memoria estática al valor de stack de la tarea. Se verifica que el stack utilizado hasta al momento no sobrepase el valor máximo. Se notifica al usuario si esto sucede y se aborta la inicialización del sistema.

4. Se crea la tarea del RTOS, pasando los parámetros de la estructura de la tarea a la función de creación. Se avisa al usuario el resultado de la creación de la tarea.

5. Si hay más tareas por crear, se continua repitiendo el proceso de iteración desde el paso 3.

6. Una vez creadas todas las tareas, se inicializan los elementos auxiliares del sistema, como por ejemplo, los semáforos utilizados por los *drivers* pertenecientes a los protocolos de comunicación propios del *hardware* (ADC, I2C, SPI, etc.).

7. Se inicia el sistema operativo. Cada tarea comienza su ejecución.

Tarea	P.	Ejecución	Periodo	Vencimiento	Stack	Tipo
PWM Leds	14	0,001	1,5	1	512	Periódica
Temperature	15	0,002	30	60	1024	Periódica
System monitor	5	0,01	60	80	2048	Periódica
IMU ACC/GYRO	13	0,010	1,067	2,134	2048	Periódica
IMU MAG	14	0,004	0,1	0,2	1024	Periódica
GNSS RX	12	60	600	700	2048	Periódica
RF TX DATA	10	0,1	60	80	2048	Periódica
RF KA	11	0,015	2,5	4	1248	Periódica
SD store	13	0,084	—-	—-	2048	Aperiódica
Actividad animal	1	0,002	—-	—-	1248	Aperiódica
PRINT UART	1	——	—-	—-	1248	Aperiódica

Tabla 4.1: Tareas del sistema, listando parámetros de interés cómo P. (prioridad), tiempo de ejecución, de vencimiento y periodo en segundos, valor de stack y tipo de tarea.

Lista de tareas

En la tabla 4.1 se pueden observar todas las tareas que contiene el sistema operativo. El funcionamiento de las mismas será desarrollado en esta sección.

Factor de utilización y planificabilidad

La planificabilidad se refiere a la capacidad de un RTOS para planificar y gestionar las tareas de manera que se cumplan todos los requisitos de tiempo crítico. Un RTOS es planificable si se puede garantizar que todas las tareas cumplirán sus tiempos de vencimiento. El factor de utilización es un parámetro que se utiliza para determinar si un RTOS es planificable, definiéndose como la suma de los tiempos de ejecución de todas las tareas en el sistema, dividida por el periodo de tiempo total disponible para la ejecución de las tareas. Si el factor de utilización es mayor que 1, entonces el sistema operativo no es planificable y no se pueden garantizar los tiempos de vencimiento. En general, se considera que un RTOS es planificable si el factor de utilización es inferior a un valor umbral determinado [31, 41, 42].

El valor umbral depende de la política de planificación utilizada por el sistema operativo y de otros factores como el número de tareas, sus tiempos de ejecución y sus tiempos de vencimiento. Por ejemplo, en el caso de *Rate Monotonic* es de 0,69, mientras que para *Earliest Deadline First* es 1.

Para poder calcular el factor de utilización de forma adecuada para este proyecto se debe estimar el correspondiente a las tareas aperiódicas, siendo las relevantes las de actividad animal y la de Sd store. En el caso de la tarea de actividad animal, como tiene un periodo igual al de la tarea del *IMU ACCEL/GYRO*, su factor de utilización es de 0,004/1,067=0,0037. En el caso de la SD store, se estimó el factor de utilización a partir de las tasas de escritura y la cantidad de bytes escritos para cada sensor (tabla

Tarea	*Bytes*	Periodo [s]	Tiempo [s]	Factor de utilización
IMU MAG	2048	30	0,046	0,00153
IMU ACCEL/GYRO	4096	20	0,084	0,0042
Temperatura	256	1800	0,010	0,0000055
GPS	1024	600	0,026	0,0000433
ADC	50	900	0,01	0,0000111
Total	—	—	—-	0,0057855

Tabla 4.2: Períodos y *bytes* a escribir por de las tareas periódicas que llaman a la tarea aperiódica de escritura en microSD, junto al tiempo de ejecución y factor de utilización asociado.

4.2), dando un valor de 0,0057.

El factor de utilización obtenido teniendo en cuenta todas las tareas mencionadas fue de 0,143. Considerando el hecho de que las tareas listadas son no bloqueantes, permitiendo que se ejecuten otras tareas sin que afecte su funcionamiento, junto a los valores de factor de utilización ya mencionados, se puede afirmar que el sistema es planificable.

4.4.1. Tarea *System monitor*

Como su nombre lo indica, esta tarea se encarga de monitorear diferentes parámetros del sistema. La misma utiliza el *driver* del ADC para poder medir periódicamente los valores de:

- Tensión de la batería.
- Corriente de carga de la batería.
- Tensión de alimentación en el conector USB.

Es a partir de estos valores, que la tarea de monitoreo acciona las transiciones a los diferentes modos de energía (nivel de batería correcto, batería baja y apagado) mencionados al principio de este capítulo.

La tarea sigue el mismo tipo de estructura mostrado en la figura 4.8, con las siguientes consideraciones:

- El bloque *Inicialización*, inicia a partir del valor de los *ticks* del RTOS, el valor de fecha y hora del dispositivo, valor que será actualizado por la tarea receptor GNSS una vez que obtenga una posición válida. Además, obtiene la ID del dispositivo y el número de dispositivo asignado por el usuario a esa ID a través de una tabla contenida en el *firmware* del dispositivo.
- El bloque *Verificación*, se encarga de realizar las verificaciones mencionadas en la máquina de estados hasta alcanzar el estado de adquisición.

- El bloque *Guardar*, almacena los valores de la tensión de la batería en la memoria microSD.
- En el bloque *Otros*, se encarga de generar las transiciones a los modos de *SHUTDOWN* y *LOW_POWER* si el valor de la batería es bajo. A su vez, actualiza el valor de batería en una variable compartida con la tarea de comunicación RF. Este valor es expresado en porcentaje, a través del uso de una tabla que asigna los valores de tensión medidos con valores porcentuales de la cantidad de energía disponible en la batería relevados experimentalmente (figura 4.4). También se ocupa de seguir los *ticks* de reloj del sistema operativo, indicando si ocurre un *overflow* y guardando un mensaje en la microSD.
- Es importante mencionar que el bloque de decisión *Baja Batería* no está incluido en este bucle, al ser esta tarea una tarea vital que debe funcionar hasta que el dispositivo se apague.

4.4.2. Tarea PWM leds

Esta tarea se ocupa de manejar el patrón de luz de los 3 leds del *hardware*. La mayoría de estos leds son manejados por la capa de abstracción de la máquina de estados utilizando el *driver* PWM. Sin embargo, debido a que este *driver* presenta un consumo muy elevado para la aplicación, una vez que el estado de adquisición es alcanzado, se apaga este módulo y parpadea únicamente un solo led del *hardware*, para indicar que el dispositivo está adquiriendo datos hasta que el mismo entre en estado *SHUTDOWN* o se apague.

4.4.3. Tarea IMU ACCEL/GYRO

Esta tarea se encarga de configurar y recolectar la información del giróscopo y acelerómetro en la IMU LSM9DS1. El manejo de este periférico es realizado utilizando una clase basada en la librería de código abierto desarrollada por SparkFun. La misma fue modificada para que pueda realizar chequeos en el integrado para que sea compatible con la capa de abstracción implementada para el *driver* I2C. Se tuvieron en cuenta las recomendaciones de las políticas de consumo provistas por Texas Instruments, por lo cual todas las funciones que involucran al *driver* I2C se ocupan de abrir y cerrar el mismo antes y después de su uso.

Además, cada vez que se utiliza la clase declarada por esta librería se debe tomar el semáforo común utilizado para el puerto I2C.

Consideraciones respecto del diagrama en la figura 4.8:

- El bloque *Verificación* revisa el estado de la comunicación con la IMU y notifica al usuario.
- El bloque *Otros*:
  - Comunica los datos a la tarea aperiódica de actividad animal para su procesamiento. Luego, si está habilitado el modo de bajo consumo de la tarea, consulta el estado de actividad animal (en reposo o en movimiento) y el tiempo en el que ha estado para tomar decisiones. Por ejemplo, apagar el giróscopo por falta de actividad, dado que es el segundo sensor de mayor consumo.
  - Actualiza una variable compartida con la tarea de comunicación RF, con el tipo de actividad animal y el tiempo en el que ha estado en ese tipo de actividad.

4.4.4. Tarea IMU MAG

Esta tarea toma las muestras del magnetómetro y las almacena, siguiendo el diagrama 4.8 explicado en la sección anterior.

4.4.5. Tarea Temperatura

Esta tarea toma las muestras de los sensores de temperatura pertenecientes a la IMU y al SoC.

Consideración respecto del diagrama en la figura 4.8:

- En el bloque *Otros* se actualizan los valores de temperatura obtenidos con ambos sensores en una variable compartida con la tarea de comunicación RF.

4.4.6. Tarea Receptor GNSS

Esta tarea se ocupa del manejo del modulo receptor GNSS u-blox Neo7M. El mismo envía mensajes del tipo NMEA (*National Marine Electronics Association*) al MCU a través de la interfaz serial UART. Estos mensajes contienen información como la geolocalización actual (latitud y longitud), la velocidad, la altitud y la hora actual, entre otros datos. Los mensajes NMEA tienen un formato estándar que consta de una cadena de caracteres *ASCII* que comienza con el símbolo "$" seguido de un código de cinco letras que indica el tipo de mensaje (por ejemplo, "GPGGA" para información de posición). Después de este código, el mensaje contiene una serie de campos con datos específicos asociados al mismo y separados por una cantidad definida de comas, seguido por el carácter asterisco al cual se le suma 1 *byte* expresado en formato hexadecimal que

corresponde al *checksum* del mensaje enviado por el módulo. Un ejemplo de mensaje NMEA es el siguiente:

*$GPGGA,172814.0,3723.46587704,N,12202.26957864,W,2,6,1.2,18.893,M,-25.669, M,2.0 0031*4F*

Con el fin de desacoplar el análisis de los mensajes del proceso de adquisición, se implementaron dos clases. Una que maneja el *driver* GPIO para encender o apagar el módulo junto al *driver* UART, utilizado para configurar el mismo, y poder así determinar parámetros (tipo de mensajes a recibir y tasa de adquisición). La otra clase analiza el contenido de estos mensajes recibidos, verifica si son correctos y extrae información de los mismos.

Es importante mencionar que debido al consumo del módulo respecto de los demás periféricos y a partir de información recabada de su uso en la región de estudio, la función encargada de obtener los mensajes del módulo GNSS implementa un tiempo límite de 1 minuto para recibir los mensajes, para luego apagar el módulo hasta que pasen 9 minutos, de este modo se limita el consumo de la batería del dispositivo.

La clase desarrollada para analizar los mensajes, basada en la clase de código abierto TinyGPSPlus, permite:

- Detectar mensaje tipo: GPGGA, GPRMC y GPGSA, permitiendo obtener información de fecha, hora, geolocalización y precisión cualitativa de la misma (como la dilución de la precisión horizontal o HDOP), entre otros.
- Validar la integridad de las tramas del punto anterior, verificando la presencia del caracter de inicio $, número de comas esperado, valor correcto de *checksum* y que el contenido sea proveniente de un estado de *fix*. Este estado se alcanza cuando el módulo se ha enlazado correctamente con suficientes satélites para obtener la información del mensaje, como posición, de forma correcta.
- Poder extraer información como fecha, hora, coordenadas de geolocalización, etc.
- Llevar un registro de cuándo se han leído los datos adquiridos (como último mensaje válido, última geolocalización obtenida, entre otros) por el programa principal, de modo que facilite la manipulación de los mismos.

La tarea de adquisición presenta una arquitectura similar a la propuesta en el diagrama 4.8 de tareas, con las siguientes consideraciones:

- Inicialización: inicia el módulo receptor GNSS. Luego espera a que la máquina de estados llegue al estado *GPS_WAIT_TO_FIX*. Una vez alcanzado ese estado, se configura y activa el módulo receptor GNSS. Luego, se espera a recibir los mensajes mencionados anteriormente una vez que el módulo logre un fix, sin aplicar un tiempo límite. Adquiridos los mensajes válidos, genera una transición

de la máquina de estados al estado *ADQ_DATA* y actualiza en el mensaje RF TX DATA (tarea encargada de enviar datos al DCS) la geolocalización, HDOP, fecha y hora, y comunica a la tarea *System Monitor* la fecha y hora obtenidas.

- Otros:
  - Al adquirir valores nuevos de geolocalización, HDOP, fecha y hora, los comunica a la tarea RF TX DATA .
  - De estar habilitado el modo de consumo en función de la actividad animal detectada, consulta si el animal ha estado en reposo por más de 10 minutos, para determinar si debe encender o no el módulo receptor GNSS.

Es importante mencionar que los valores almacenados en memoria son los valores recibidos sin realizar ningún tipo de procesamiento. Esto se debe a que en otros trabajos realizados por el grupo con módulos receptores GNSS, se encontraron mensajes que no contenían los valores correctos, por más que pasaran todas las pruebas de validación mencionadas.

Función de adquisición

En la figura 4.11 se observa el funcionamiento de la función desarrollada para obtener la información del receptor GNSS. Esta función primero inicializa la salida GPIO que maneja el encendido/apagado del switch del módulo. El *driver* UART en modo no bloqueante, utilizando una función de *Callback*, habilita la recepción de mensajes UART, y luego toma posesión del semáforo asignado al *Callback*.

Este modo de funcionamiento de la UART implica que cuando llegue un mensaje que supere el tamaño umbral del *buffer* de recepción (30 bytes en este caso, siendo 100 el máximo), la función de *Callback* es llamada y los bytes recibidos son copiados mediante acceso directo de memoria a un *buffer* como parámetro de entrada en la función, el cual puede ser leído por el código dentro de la función *callback*. En esta aplicación, la función *Callback* al ser llamada copia los valores de bytes ingresados a un *buffer* compartido con la aplicación (denominado *buffer Callback*) y el número de bytes añadidos. Finalmente, libera el semáforo *Callback* para indicar a la función principal que se recibió un mensaje y que puede procesarlo, tal como se observa en la figura 4.11.

Una vez recibidos los bytes y liberado el semáforo *Callback*, los datos son copiados a la clase que se ocupa de realizar el análisis de los mensajes del receptor GNSS verificando si se ha recibido un mensaje completo válido. A partir del resultado del análisis del mensaje, se descarta o almacena el mismo. Si este es correcto, se almacena en la estructura de memoria junto al tipo de mensaje, el número de mensaje de ese tipo que hay en la estructura, y el tiempo (en ciclos de reloj) en el que fue recibido desde que se invocó la función de recepción. Considerando que se deben almacenar tres tipos de

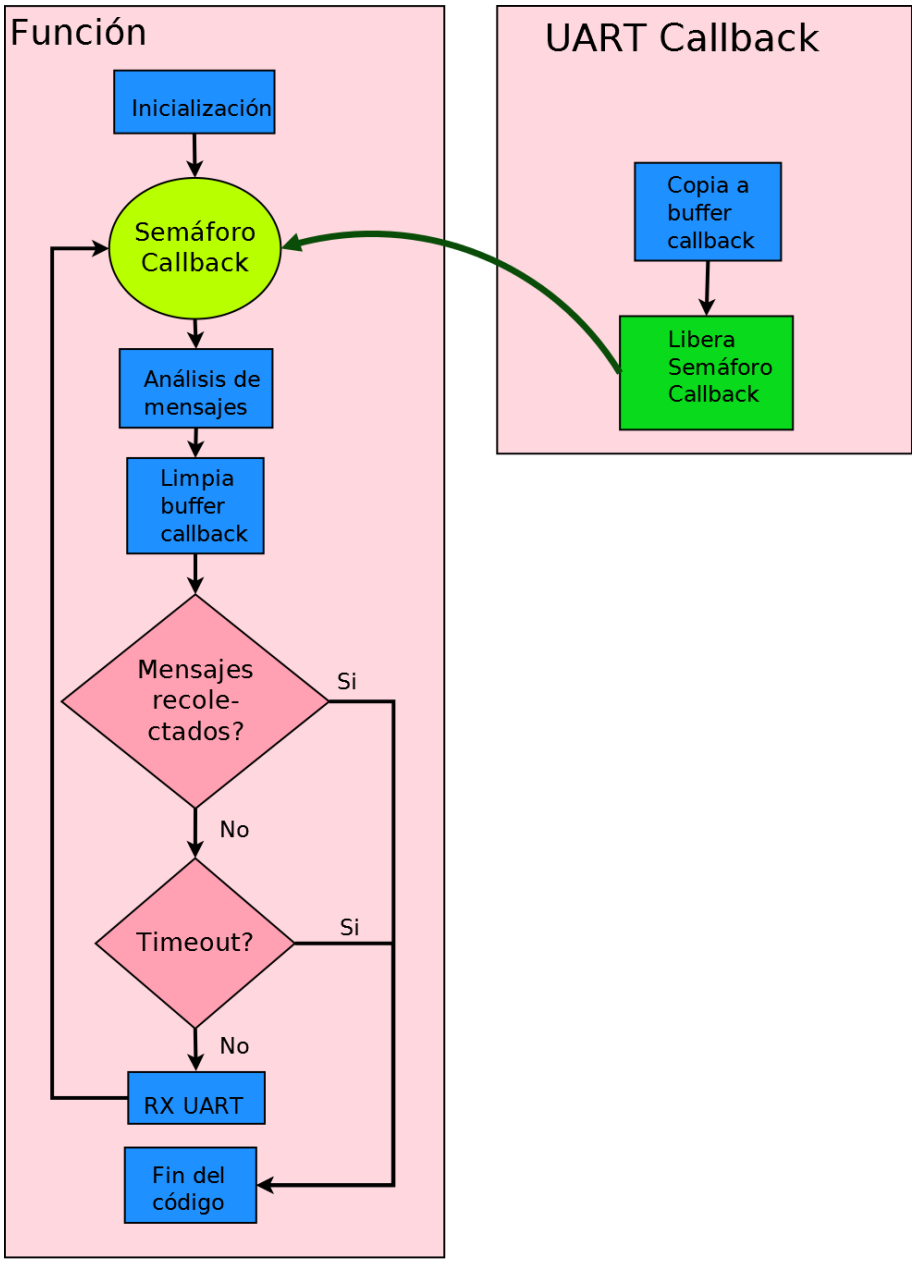


Figura 4.11: Diagrama reducido de la función encargada de recolectar y analizar los mensajes recibidos por el módulo GNSS.

mensajes (GPGGA, GPRMC y GPGSA) y tres mensajes de cada tipo para obtener redundancia en la información (dando un total de nueve mensajes a almacenar), este mecanismo permite llevar un control sobre en qué momento se recibió cada tipo de mensaje y en qué orden. El número de mensajes a recolectar asociado a cada tipo fue elegido debido a que se ha encontrado que a pesar de que la información del receptor GNSS provenga de un estado de *fix* y cumpla satisfactoriamente todas las verificaciones mencionadas al principio, a veces puede contener errores, como posiciones alejadas a miles de kilómetros del lugar donde se encuentra el dispositivo.

Es importante mencionar que el semáforo de la función de *Callback* contiene un tiempo límite de 1 segundo para ser tomado. Si el semáforo es liberado a causa de este tiempo límite, significa que no se han recibido mensajes desde el módulo por UART, por lo que se considera que no se están recibiendo mensajes desde el receptor y, por lo tanto, se reinicia el mecanismo de análisis de los mensajes.

A partir de la recepción del mensaje y de su análisis, se verifica si se han recolectados todos los mensajes necesarios. De no ser así, se verifica si el tiempo desde el cual se ha llamado a la función, ha superado el tiempo límite o *Timeout* correspondiente a 60 segundos. Este mecanismo fue diseñado para evitar que el receptor GNSS agote la batería del dispositivo cuando el mismo no puede encontrar un valor válido de *fix* en un tiempo prudencial. Una vez que se alcanza el final del código, la función retorna a la tarea, retornando un valor de verdadero o falso dependiendo de si se recolectaron todos los mensajes esperados. Independientemente de este retorno, la tarea guarda todos los mensajes adquiridos en la memoria microSD.

4.4.7. Tarea SD Store

Para la implementación de la escritura sobre la microSD, se implementó una capa de abstracción sobre una versión de la librería FatFs (siendo compatible con microcontroladores ARM) provista por Texas Instruments, la cual presenta las siguientes prestaciones y limitaciones [43]:

- Permite leer/escribir/abrir/borrar archivos y sub carpetas.
- Los nombres de los archivos, carpetas y sub carpetas no deben tener más de ocho (8) caracteres.
- Las extensiones de los archivos no deben tener más de tres (3) caracteres.
- Solo funciona con memorias SD/microSD en formato FAT o FAT32, limitando el tamaño de las memorias a 32 GB como máximo.

La capa de abstracción, junto a las funciones provistas, permiten realizar escrituras de datos utilizando como entrada la estructura propuesta en la figura 4.5, o escribiendo

un arreglo de bytes sin la necesidad de que el RTOS sea inicializado. En todos los casos los datos son escritos sobre el mismo archivo, añadiendo los datos al final de la última escritura.

El funcionamiento de la tarea trabaja con la estructura observada en la figura 4.10, en donde la tarea de inicialización consiste en:

- Esperar a que la máquina de estados entre en el estado *SD_CHECK*, donde verifica que la microSD conectada funcione correctamente. De ser así, realiza una transición al estado *GPS_WAIT_TO_FIX*.

- Luego, aguarda a que el sistema llegue al estado *DATA_ADQ* para realizar la primera escritura del sistema. Esta escritura se lleva a cabo creando una carpeta con el número asignado al dispositivo, ej.: TORTO005. En ella se crea una subcarpeta con la fecha adquirida a través del módulo receptor GNSS, por ejemplo, si la recepción ocurrió el 2 de Enero de 2023, la subcarpeta se llamará 20230102. En esta carpeta se guardarán todos los archivos que se creen a partir de este punto. En el archivo creado (que lleva el mismo nombre que la carpeta, en este caso TORTO005), se escriben los parámetros del sistema que permiten identificarlo (tipo de dispositivo, identificador (ID), versión de *firmware*, *hardware* y escala de *ticks* del sistema) junto al valor de batería en mV, el horario y fecha en la que se adquirió el primer valor válido de posición.

- Finalmente, inicializa y toma el semáforo de la aplicación, quedando a la espera de que alguna tarea llame a la función correspondiente para realizar el volcado de datos.

4.4.8. Tarea de Print UART

Esta tarea cumple el propósito de enviar caracteres por el puerto serie en forma no bloqueante. Está diseñada para facilitar el uso de este periférico en la comunicación con otros dispositivos, como módulos Bluetooth o computadoras personales, y para ayudar a la depuración del *firmware*. Utiliza un esquema similar al propuesto en la figura 4.10, con la diferencia de que presenta un sistema de colas para evitar que las tareas tengan que esperar a que finalice el proceso de impresión, para poder llamar a la función de ejecución.

El sistema de colas implementado se ilustra en la figura 4.12. Las letras A, B y C asignadas a las flechas indican las partes del diagrama de flujo en donde se libera un semáforo asociado, permitiendo que el mismo pueda ser tomado por una función o tarea.

En este caso, la tarea que desea realizar la impresión debe llamar a la función correspondiente, la cual toma el semáforo que contiene la lista de elementos a imprimir

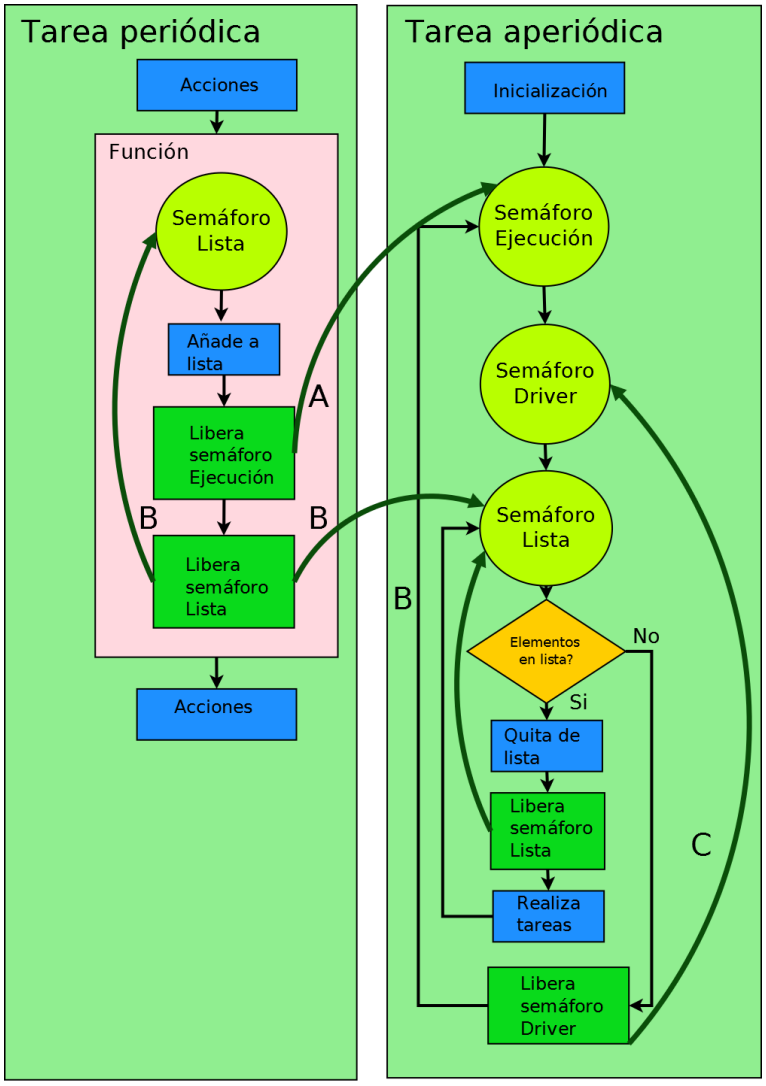


Figura 4.12: Estructura propuesta para la tarea aperiódica print UART.

junto con el número de elementos en la lista. En este punto se añade el elemento a la lista y se actualiza el contador, verificando que el tamaño no sobrepase el número máximo de caracteres o el límite de mensajes en la lista.

Luego, se libera el semáforo de ejecución junto al semáforo utilizado para añadir o examinar los elementos en la cola. Esto permite que la tarea aperiódica encargada de la impresión, tome el semáforo correspondiente al *driver* de la UART y comience a enviar los mensajes por el puerto serie. Una vez que logra tomar el semáforo del *driver*, la tarea entra en un bucle en el cual revisa si hay elementos en la lista, quita un elemento, libera el acceso a la lista, y luego realiza la tarea asignada siendo, en este caso, la impresión del mensaje por el puerto serie. Este proceso se repite hasta que ya no hay más elementos en la lista, donde el sistema libera el semáforo asignado al *driver* UART y queda a la espera de que otra tarea llame a la función para liberar el semáforo de ejecución.

Es importante mencionar que este sistema tiene la ventaja de no tomar el *driver* a la hora de añadir elementos a la lista, y, por lo tanto, reducir el tiempo de ejecución de la función con el costo de requerir memoria adicional para soportar un número determinado de pedidos en la lista. Este es el motivo por el cual, no se utilizó este esquema para la tarea encargada del guardado en memoria microSD.

4.4.9. Tarea RF data send

Esta tarea es la encargada de enviar por RF al DCS parte del los datos recolectados por los diferentes sensores. Antes de entrar en detalle respecto de su implementación, se introducirá la configuración y manejo del *driver* o radio RF.

Configuración de radio RF

Para el manejo de la radio RF se utilizó la capa de abstracción y módulo *EasyLink* provista por el fabricante, la cual facilita significativamente el uso del *driver* de RF y añade funciones como el filtrado de mensajes recibidos por direcciones asignadas (siendo un campo transmitido en cada mensaje) y lectura simple del valor de RSSI de los mensajes recibidos, entre otras. Se realizaron modificaciones a este módulo para que permita al usuario utilizar la banda de frecuencias debajo de los 430 MHz, debido a que el entorno de desarrollo no permite utilizarla en librerías de alto nivel como es el caso de *EasyLink* y de varios protocolos propietarios que ofrece Texas Instruments para la implementación de redes basadas en internet de las cosas. Además, la librería *EasyLink* solo soporta el uso de un tipo de configuración radio RF, teniendo que estar predefinida por el usuario al momento de compilar y sin permitir cambios en tiempo real, junto al hecho de funcionar únicamente con el RTOS TI-RTOS. Por este motivo se modificó la librería de modo que los parámetros de configuración de la radio (como son

su frecuencia, modulación, *buffer* de recepción, etc.) sean almacenados en la tarea que utiliza el *driver*, y se le comuniquen a la misma por medio de punteros, permitiendo que cada tarea pueda utilizar una configuración diferente para realizar la comunicación RF. Además, se añadieron las funciones de la capa de abstracción del RTOS diseñada para este proyecto, permitiendo al usuario seleccionar mediante un MACRO si la librería utilizará TI-RTOS o la capa de abstracción desarrollada como RTOS.

Para la transmisión, se utilizó la configuración correspondiente al estándar de bajo consumo y largo alcance *Wireless M-Bus Protocol*, siendo sus características principales:

- Frecuencia central: 149.950 MHz.
- Modulación: 2-GFSK (modulación por desplazamiento de frecuencia gausiana).
- Valor de desviación de frecuencia para modulación: 2.4 kHz.
- Tasa de datos: 4.8 kbps.
- Ancho de banda en recepción: 17 kHz.

Se utilizó este estándar debido a que se evaluó experimentalmente que en la banda de 150 MHz permitía un mayor alcance (un 60 % mayor) que al utilizar el protocolo provisto por Texas Instruments (*SimpleLink Long Range*) basado en técnicas de DSSS.

Implementación de tarea

Como se menciono, ésta tarea envía periódicamente un mensaje a la DCS con información de interés para el usuario:

- Identificación: contiene el tipo de mensaje (1 byte, indicando si corresponde a datos de adquisición u otro tipo de mensaje), tipo de dispositivo (MD) e ID del dispositivo (8 bytes).
- Datos de sensores: temperatura, coordenadas geográficas junto al valor de HDOP y hora en UTC en la cual los datos del receptor GNSS fueron adquiridos.
- Detección de actividad animal: un byte correspondiente a si el animal se encuentra en movimiento (valor 2) o se encuentra en reposo (valor 1).
- Fecha y hora de envió: fecha y hora del día en la cual se envió el mensaje.
- *Checksum*: conformado por 2 bytes.

En la figura 4.13 se observa el diagrama que indica el funcionamiento general de la tarea. Es importante mencionar que al realizar la transmisión del paquete (*Transmitir*), se inicializa la radio RF luego de tomar control del *driver*, se realiza la transmisión y

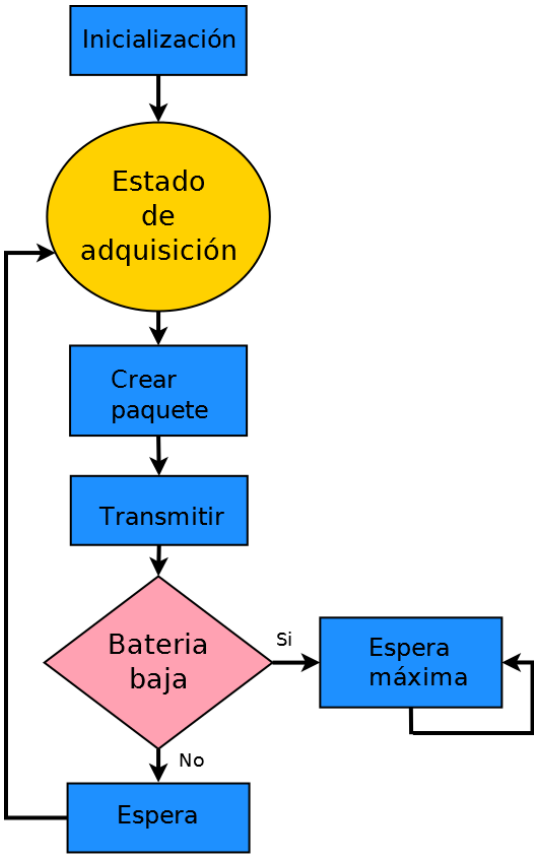


Figura 4.13: Estructura correspondiente a la tarea de comunicación RF.

se espera a que una función de *Callback* notifique a la tarea a través del uso de un semáforo compartido, como se vio en los diagramas de las tareas aperiódicas. Este diseño permite que la tarea sea no bloqueante y el MCU pueda realizar otras funciones mientras la radio RF, que posee un procesador distinto del principal, envía el mensaje. Una vez finalizada la transmisión, se apaga el *driver* RF y se libera el recurso.

4.4.10. Tarea RF KA (*Keep Alive pulse*)

Esta tarea es la encargada de enviar en forma periódica el pulso de RF que permitirá la recuperación del MD mediante el uso de un TD o el receptor ATS (junto a una antena Yagi-Uda) tal como se menciono en el Capítulo 1. Por este motivo se la considera como una tarea vital, por lo cual comienza su ejecución apenas se inicia el RTOS y solo deja de funcionar cuando el dispositivo entra en estado *SHUTDOWN* o se agota la batería, garantizando que siempre se transmita el pulso que permite la recuperación del dispositivo.

La transmisión del pulso es realizada cada 2,5 segundos. Este pulso presenta una modulación más reducida en espectro (1 kHz) respecto de la utilizada en la tarea de transmisión de datos *RF data send*. Esto se debe a que la frecuencia de cada pulso KA es única para cada dispositivo al estar asociada a la ID del dispositivo en forma manual

por medio de una tabla cargada en el *firmware*. Cuanto menor sea el ancho de banda de los pulsos, se pueden asignar valores más próximos entre sí, de modo que se hace un mejor uso del espectro. La duración del pulso es de 16 ms, que es el menor valor de pulso admisible por la librería utilizada.

4.4.11. Tarea actividad animal

Para poder realizar la detección de actividad, se implementó un algoritmo de detección de umbral combinado con una resta del promedio de valores para una ventana de 1 segundo, aplicado en los valores correspondientes a los tres ejes del acelerómetro. Este método permite utilizar un valor constante como umbral de comparación, al desacoplar de la entrada el valor de aceleración promedio. La importancia de esta última consideración radica en el hecho de que debido a la presencia de la fuerza gravitatoria terrestre, la misma es medida por el acelerómetro y su valor se acopla a las mediciones de los diferentes ejes del mismo, siendo este valor función del ángulo del sensor respecto de la la superficie terrestre. Esto significa que cada medición tendrá acoplado un valor constante que dependerá de este ángulo de inclinación. Por lo cual, la implementación de este filtro permite separar este valor de los provenientes a causa del movimiento del animal.

El algoritmo consta de las siguientes etapas:

1. Se reciben los datos correspondientes a 1 segundo de adquisición para los ejes X, Y, Z.

2. Se selecciona un eje con datos sin procesar.

3. Se añade un dato a un *buffer* deslizante asignado a ese eje.

4. Se realiza la resta de la muestra actual con el valor promedio del *buffer*. Si la resta del nuevo valor con el promedio de los anteriores supera el umbral configurado, se coloca una bandera indicando que el resultado fue afirmativo.

5. Se añade la muestra procesada al *buffer* deslizante, quitando la muestra más antigua.

6. Se computa el promedio del *buffer* deslizante.

7. Se repite el paso 3 para todas las muestras de un eje.

8. Si el resultado de la detección hasta el momento fue negativo, se repite el paso 2 para todos los ejes.

Los resultados posibles de este algoritmo son: “1” para indicar que no se detectó actividad y por lo tanto el animal esta en reposo, y “2” para indicar que hubo movimiento.

Una vez que se realiza el procesamiento, se almacena el valor del resultado y del tiempo (en *ticks* de RTOS) al momento en que se detectó la actividad, en una variable interna protegida por un semáforo. Por lo cual, si otra tarea desea consultar el valor de actividad actual se debe llamar a una función que accede a la variable utilizando el semáforo y computa el tiempo que ha pasado desde que hubo un cambio en el estado del animal, retornando el estado actual junto a la cantidad de segundos en la que ese estado ha permanecido. Por ejemplo, la tarea puede consultar este proceso y recibir como respuesta: 1 (en reposo), 250 segundos. Esto implica que el animal ha estado en reposo por 250 segundos , por lo cual hubo movimiento en un intervalo menor a 10 minutos (600 segundos) y, por ende, se debe encender el receptor GNSS para adquirir una muestra de posición.

Es importante mencionar que el diseño de la capa de abstracción fue pensado para tener compatibilidad con el código generado utilizando la plataforma de *TinyML* de *EdgeImpulse*, que no fue implementado en este proyecto.

4.5. Estimación de consumo y autonomía

4.5.1. Estimación de consumo

En la tabla 4.3 se pueden observar los valores estimados de consumo correspondientes a cada tarea, estimados a partir de los valores obtenidos de las hojas de datos del fabricante para los modos de funcionamiento de los integrados, junto a los tiempos de duración de las tareas y el periodo de las mismas.

Es importante mencionar que para el cálculo de consumo de la tarea de la microSD, se utilizó el factor de utilización (0,00578) calculado anteriormente.

4.5.2. Estimación de autonomía

En la tabla 4.4 se pueden observar los valores estimados de autonomía para una batería de 600 mAh en diferentes modos de funcionamiento, considerando los casos donde todos los módulos del sistema están activos, cuando el sistema no utiliza el receptor GNSS, cuando además el giróscopo esta apagado (y solo se adquieren datos con el acelerómetro y magnetómetro de la IMU), y, finalmente, el caso en el que ningún sensor está activo o enviando datos por radiofrecuencia (es decir, cuando solo se envía el pulso KA y se monitorea el estado del sistema).

Es importante mencionar que debido a que se utiliza una fuente con una eficiencia

Tarea	Consumo $[mA]$	Tiempo $[s]$	Periodo $[s]$	Consumo $[mAh]$
LEDS	1	0,001	1	0,001
MCU	2,9	0,13	1	0,37
IMU GYRO	1,9	———	1,067	1,9
IMU ACEL	1,9	———	1,067	—
IMU MAG	0,1	0,004	0,1	0,6
GNSS	47	60	600	4,7
RF TX DATA	24	0,100	60	0,06
RF KA	24	0,016	2.5	0,15
SD store	25	0,00578	—-	0,14
ACT. ANIMAL	2,9	0,004	1,067	0,011
PRINT UART	——	0,011	—-	
TOTAL	———	——	—-	8,12

Tabla 4.3: Consumo estimado para cada tarea.

del 90 % para las condiciones de carga usuales del sistema, se debe aplicar un factor de 1,11 para obtener el consumo efectivo. También es importante mencionar que cuando se estima el valor de consumo con el receptor GNSS apagado, se restan 0,29 mA correspondientes a la carga de procesamiento en el MCU.

4.6. Decodificación de datos

Como se mencionó al principio de este capítulo, los datos adquiridos por los sensores son almacenados en un archivo dentro de la memoria microSD, conteniendo al principio del mismo la información del sistema junto con información para el usuario (como los mensajes de detección de actividad animal) y la información de los sensores, donde la mayoría del contenido esta en binario. En la figura 4.14 se puede observar una captura del archivo .DAT generado, donde se visualiza el primer mensaje de escritura (*SYSTEM INFO*) junto a mensajes correspondientes al almacenamiento de información de diferentes sensores (*IMU0M*, *IMU0AG* y *GPSNEO7N*).

```
SYSTEM INFO :Device Type:MD_TORTOISE,HardV:2.0,FirmV:1.0, Device ID "0C
!NEW ANIMAL ACTIVITY DETECTED, CURRENT STATE: 2, at time: 54.251 [sec]
GPSNEO7M     ┘-   Z┘  `Î   X x       $GPRMC,115021.00,A,4034.99065,S,064
!NEW ANIMAL ACTIVITY DETECTED, CURRENT STATE: 1, at time: 55.318 [sec]
IMU0AG       ┼:   ~Ï  ←#     ┐ õ ,,SR=1,GY=1
Œÿqüž┘£¿nÿ,  ÿoü ┘ó¿fÿ⁋ „ÿyü¬┘È¿fÿ┬ Œÿzü'┘Œ¿Šÿ¶ †ÿlü·┘‰¿›ÿ⁋ ▯ypü›┘™¿§ÿ$ ▯
›┘›¿▯ÿ1 ‰ÿmü•┘§¿Šÿ# ‹ÿsü¡┘²¿˜ÿ “ÿzü˜┘Š¿˜ÿ' “ÿeüŸ┘•¿ÿ2 „ÿnü•┘§¿…ÿ -ÿrü›┘
¢ÿ+ ƒÿbüž┘-¿™ÿ" “ÿwüŒ┘¯¿²ÿ1 ˆÿyü¢┘±¿,ÿ+ £ÿoü›┘·¿¶ÿ/ ‹ÿuüŽ┘³¿–ÿ⁋ -ÿmü§┘
IMU0M        ■F    Kx  ÜÜ   ┘┘          Á•ÕúŒýx|´ûä÷x|´ûä÷x|´ûä÷x|´ûä÷x|´ûä
åû½÷M|åû½÷M|åû½÷M|åû½÷M|åû½÷M|åû½÷M|åû½÷M|åû½÷M|åû½÷G|▯ûÕ÷G|▯ûÕ÷G|▯ûÕ÷G|▯ûÕ÷G
IMU0AG       ┼F    #   ¢z     ┐ õ ,,SR=1,GY=1     ¦ÿ,  šÿqüœ┘¡¿ª ÿ(  ▯ÿtü'┘£
```

Figura 4.14: Contenido de archivo generado por MD al almacenar los datos adquiridos en una memoria microSD.

Para decodificar los datos, se realizó un programa en Python que lee los datos y

Módulos/tareas activas	Consumo $[mA]$	Consumo efectivo $[mA]$	Autonomía $[horas]$	Autonomía $[días]$
GNSS GYRO ACCEL MAG SD store TX DATA KA	8,12	9,03	66,44	2,78
GYRO ACCEL MAG SD store TX DATA KA	2,93	3,25	184	7,66
ACCEL MAG SD store TX DATA KA	1,03	1,14	526	21,92
KA	0,37	0,41	1461	60,97

Tabla 4.4: Autonomía estimada para diferentes modos de funcionamiento empleando una batería de 600 mAH.

genera archivos de salida en formato .csv. El motivo de la elección de este lenguaje es por su fácil manejo, su amplio uso en la comunidad científica y porque es el lenguaje utilizado actualmente por la mayoría de los miembros del grupo de trabajo.

El algoritmo de decodificación consta de los siguientes pasos:

1. Se analiza el archivo .DAT en el cual fueron guardados los datos, como un archivo de texto ASCII.

2. Se obtienen los parámetros del sistema de la primera escritura mencionados al principio de este capítulo.

3. Se busca la ocurrencia de los caracteres $\backslash r \backslash n$.

   Se decodifica la información a partir de determinar a qué fuente corresponde leyendo el contenido en el campo *head* de la estructura ilustrada en la figura 4.6, convirtiendo los valores de binario a decimal a partir del sensor detectado. Por ejemplo, si se encuentra el grupo de caracteres *IMU0M*, se determina que la información corresponde a la obtenida por medio del magnetómetro del integrado LSM9DS1. Por medio de los datos contenidos en el *head* de la estructura, se guardan la cantidad de caracteres detallados en el campo *block_size*. Utilizando el número almacenado en el campo de *data_written*, se itera el arreglo extrayendo los diferentes datos. Utilizando los datos en la estructura *config*, se obtiene la escala del magnetómetro para convertir los valores en binario a decimal. Finalmente, con los valores de estampa de tiempo en el *header* y escala de *ticks* del RTOS obtenida de la lectura de la primera escritura, se generan los valores de tiempo para cada muestra.

4. Se realiza el paso 3 hasta terminar de leer todos los datos de todos los sensores.

5. Se generan archivos de salida en .csv que contienen los datos.

6. Se realiza un gráfico para cada tipo de dato extraído.

4.7. Tareas de la DSC y el TD

Tanto la estructura de la tarea de recepción de RF de la DCS como del TD fueron diseñadas utilizando un esquema similar al presentado en la figura 4.13, con la diferencia de que en lugar de transmitir, se reciben los mensajes del MD y se envía su contenido expresado en caracteres ASCII al puerto serie UART, para que puedan ser leídos por el usuario. También se almacenan dentro de la memoria microSD, en un archivo que denominaremos *log*, indicando la fecha, hora y RSSI al momento de recibir el mensaje, junto a otra información adicional. En el caso del TD, esta información adicional es

la frecuencia portadora de RF a la cual se están recibiendo los pulsos *Keep Alive*, y en el caso de la DCS, corresponde a la información enviada por radio frecuencia por el MD. Además en el caso del TD, el puerto serie permite al usuario seleccionar la frecuencia de recepción del dispositivo, utilizando un diseño basado en el expuesto en la figura 4.11 para el caso de la tarea del receptor GNSS. Esta implementación utiliza el mismo principio de interacción del código principal de esta tarea con la función *Callback* asociada al módulo UART, en donde la función *Callback* notifica la recepción de caracteres a la función principal que luego implementa el cambio en el valor de frecuencia de recepción.

En la figura 4.15 se puede observar el contenido del archivo *log* generado por la DCS, donde se observa que han llegado mensajes de diferentes dispositivos, siendo aquellos cuyas IDs fueron asignadas a los números 3 y 6 como se observa en el campo "*NUM*". En la figura 4.16 se puede observar un mensaje completo con todos los datos transmitidos por el MD.

```
DCS Rx Date UTC:20230223 00:36:54, TX: 12:15:40, RSSI:-39, MSJ:1,DE_type:1,NUM: 6,
RF:         €              20230223¦◘  Ù                SFª K↕ d→┤┐  202302256VþU  8(
DCS Rx Date UTC:20230223 00:37:27, TX: 12:16:40, RSSI:-33, MSJ:1,DE_type:1,NUM: 6,
RF:         €              20230223Ç◘  ß                SFª K↕ d→├ ┐ 20230225TVþU  8
DCS Rx Date UTC:20230223 00:37:54, TX: 12:17:06, RSSI:-62, MSJ:1,DE_type:1,NUM: 3,
RF:         €              20230223â◘  Â                ËKª K↕ d←┬ 1 20230225aV¤S  P
DCS Rx Date UTC:20230223 00:38:27, TX: 12:17:40, RSSI:-30, MSJ:1,DE_type:1,NUM: 6,
RF:         €              20230223└      â             SFª K↕ d→┐♪ 20230225rVþU
DCS Rx Date UTC:20230223 00:38:54, TX: 12:18:06, RSSI:-58, MSJ:1,DE_type:1,NUM: 3,
RF:         €              20230223       Æ             ËKª K↕ d↑  1 20230225▯V¤S
DCS Rx Date UTC:20230223 00:39:27, TX: 12:18:40, RSSI:-38, MSJ:1,DE_type:1,NUM: 6,
```

Figura 4.15: Ejemplo de log de dispositivo DCS, donde se observan los mensajes recibidos por RF de dos MD.

```
DCS Rx Date UTC:20230223 01:15:27, TX: 12:54:40, RSSI:-43, MSJ:1,DE_type:1,NUM: 6,
ID:00:12:4B:00:1C:AA:46:53|activity: 2, act.time: 0 [sec], BAT: 100, T_MCU: 29,
T_IMU: 26, GPS lat: -40.5835, lon: -64.9987,hdop: 2.4, fix at date:20230225 12:53:48
```

Figura 4.16: Ejemplo de mensaje completo enviado por MD y recibido por DCS.

Capítulo 5

Resultados

En este capítulo se mostrarán los resultados obtenidos al momento de realizar la fabricación del dispositivo y sus costos asociados, junto a los datos relevados al realizar pruebas tanto en el laboratorio del Instituto como en la campaña de conservación de tortugas realizada en febrero del año 2023 en San Antonio Oeste.

5.1. Fabricación

En esta sección se detallará la fabricación de los dispositivos y sus costos asociados. Debido a la disponibilidad de componentes y presupuesto, se realizó la fabricación de siete dispositivos: 5 dispositivos de monitoreo (MD), un dispositivo rastreador (TD) y una estación colectora (DCS).

5.1.1. Circuito impreso

La fabricación del circuito impreso (PCB) observado en la figura 5.1, fue realizada por el la empresa Mayer Circuitos Impresos al ser la que presentó la propuesta de menor costo dentro de los fabricantes de PCB consultados en Argentina. El material utilizado fue el tipo FR4 con un espesor de 1,6 mm.

5.1.2. Montaje de componentes

El montaje y soldado de los componentes de montaje superficial fue realizado por la empresa ASSISI. En la figura 5.2 se observa el resultado final, en donde se agregaron la batería y módulo receptor de sistema global de navegación por satélite (GNSS) al PCB con sus componentes de montaje superficial soldados. Las medidas del dispositivo son de 50 x 64 mm, acorde a lo esperado en el diseño detallado en el Capítulo 3. El peso total del mismo es de 44,9 g, siendo menor al valor máximo de 75 g mencionado en el Capítulo 2.

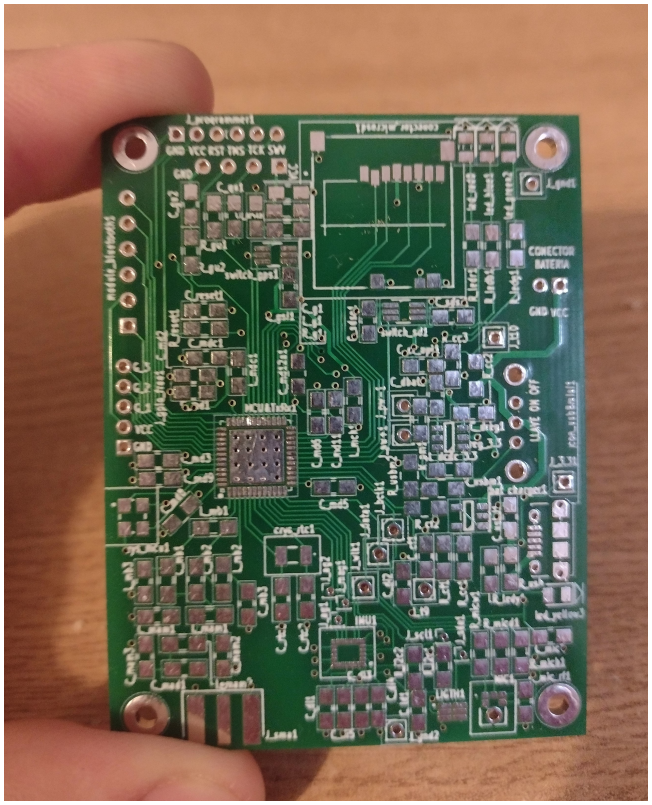


Figura 5.1: PCB diseñado en el Capítulo 3 fabricado por la firma Mayer.

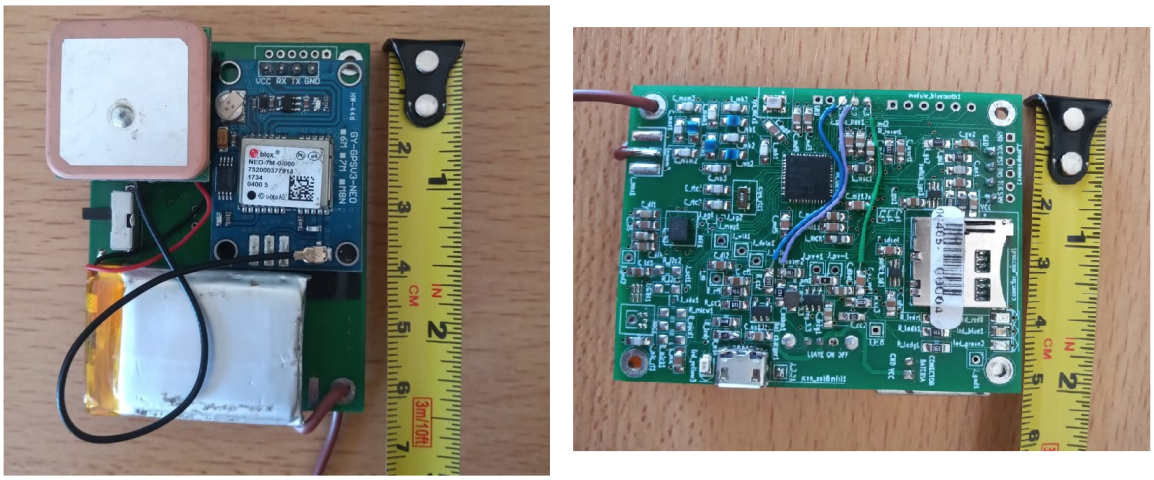


Figura 5.2: Versión implementada del MD.

5.1.3. Costos de fabricación del MD

A continuación se desglosan los costos para la fabricación de 10 MDs.

- Componentes: 600 USD
- Receptor GNSS: 270 USD
- Batería 40 USD
- PCB: 125 USD
- Montaje: 200 USD

Siendo un total de 1135 USD, de manera qué el costo por dispositivo corresponde a 113,5 USD, resultando en un costo inferior al valor de los dispositivos usados anteriormente, de 250 USD (mencionados en el Capítulo 1) correspondiente a un dispositivo prototipo y un transmisor RF.

Dificultades encontradas

Es importante mencionar que los costos de los componentes fueron más elevados como consecuencia de la disminución de *stock* de componentes electrónicos a causa de la pandemia de COVID-19, como se mencionó en el Capítulo 3. Este contexto también implicó demoras en los envíos de los mismos debido a los cambios volátiles en el *stock* de los proveedores. En otras palabras, en el intervalo en el cual se realizaba la compra y el proveedor realizaba el envío, se agotaba la existencia de algunos componentes críticos, debiendo aguardar hasta su reposición o buscar proveedores alternativos más costosos en ciertos casos.

5.2. Ensayos en laboratorio

En esta sección se detallarán las pruebas realizadas en el laboratorio de Telecomunicaciones para poder medir varios parámetros de interés del dispositivo, como son: su consumo, autonomía, alcance y respuesta en radiofrecuencia.

5.2.1. Consumo, autonomía y memoria de almacenamiento

En la tabla 5.1 se observan los consumos obtenidos para las diferentes funciones del dispositivo. Los mismos fueron medidos utilizando la tecnología de medición de consumo EnergyTrace ™ junto a la placa de desarrollo LAUNCHXL-CC1312R1 mencionados en el Capítulo 3.

Tarea	Consumo [mA]	Tiempo [s]	Periodo [s]	Consumo [mAh]
Leds	1	0,001	1	0,001
MCU	2,9	0,147	1	0,312
IMU GYRO	1,9	———	1,067	1,9
IMU ACEL	0.1	———	1.067	0,1
IMU MAG	0,1	0,004	0,1	0,1
GNSS	74	60	600	7,4
RF TX DATA	30	0,100	60	0,05
RF KA	23	0,016	2,5	0,147
SD store	41	0,00578	—-	0,23
Act. animal	2,9	0,004	—-	0,096
PRINTU	—-	—-	—-	
TOTAL	–	—-	—-	10,44

Tabla 5.1: Consumo de las diferentes tareas y sensores medidos en el MD.

Es importante mencionar que el valor de consumo para la tarea RF TX DATA es mayor debido a que se transmite a una potencia de 14 dBm en lugar de 12,5 dBm como es el caso de la tarea RF KA. También cabe destacar que el valor de consumo medio asociado a la tarea*SD store* encargada de realizar la escritura en la microSD, es mayor a los 25 mA que se encontró en diferentes hojas de datos del fabricante. En el caso del receptor GNSS se midió un valor de 74 mA en lugar de los 47 mA detallados por la hoja de datos del fabricante. En forma adicional, se encontró que el consumo del magnetómetro y del acelerómetro es considerablemente menor al detallado en la hoja de datos.

Autonomía

En la tabla 5.2 se observan los valores de consumo medidos para los diferentes modos de funcionamiento, junto a los valores de autonomía asociados para la batería de 600 mAh. Se observa que el valor de consumo para el caso en el que receptor GNSS y el giróscopo se encuentran apagados, es menor al mostrado en la tabla 4.4, mientras que en los otros casos es mayor.

Es importante mencionar que la tasa de muestreo cuando el giróscopo está desactivado cambia de 14.9 a 10 Hz y por lo tanto cambia también la tasa con la cual se guardan los datos, disminuyendo significativamente el consumo de la microSD.

Además al momento de realizar las mediciones se encontró un consumo parásito de 150 μA cuando la microSD está en modo *shutdown*, lo que ocurre luego de que se finaliza la escritura y se deja de utilizar la memoria. Es importante mencionar que el tiempo para la primera escritura, después de que la microSD recibe alimentación es 4 veces mayor que para las escrituras subsiguientes, lo que implica multiplicar por cuatro el consumo asociado a la tarea *SD Storage*. Por lo tanto, el apagar y encender

Módulos/tareas activas	Consumo $[mA]$	Consumo efectivo $[mA]$	Autonomía $[horas]$	Autonomía $[días]$
GNSS GYRO ACCEL MAG SD store TX DATA KA	10,44	11,58	51,91	2,16
GYRO ACCEL MAG SD store TX DATA KA	3,04	3,37	178	7,41
ACCEL MAG SD store TX DATA KA	0,82	0,91	659	27,47
KA	0,51	0,56	1071	44,67

Tabla 5.2: Valores de autonomía calculados a partir de los valores obtenidos de las mediciones en laboratorio para una batería de 600 mAh.

la microSD utilizando el *switch* incluido en el *harware* implica valores de consumo mayores a los obtenidos, siendo el motivo por el cual no se utilizará esta funcionalidad.

También se midió una corriente de 0,21 mA proveniente del MCU cuando los demás sensores se encuentran apagados. Además cada vez que el dispositivo genera una transición de estado de sueño a despertar se genera un consumo de 8,1 μA, lo cual ocurre al menos cada 500 ms cuando no se ejecutan tareas en esa ventana de tiempo.

Memoria utilizada

La cantidad de memoria utilizada por el MD para almacenar una semana de datos adquiridos fue de 74,45 MB, siendo un valor menor al estimado en el Capítulo 3 de 120,97 MB.

5.2.2. Caracterización de radiofrecuencia

Los ensayos realizados a continuación se hicieron utilizando el instrumental disponible en el laboratorio de telecomunicaciones del departamento de Ingeniería en Telecomunicaciones, el cual consiste en un analizador de espectro Keysight Technologies N9320B y un generador de radiofrecuencia Keysight Technologies N9310A.

Se realizaron las siguientes mediciones para evaluar las características de la radio RF del dispositivo:

- Sintetizador de frecuencia: la resolución en frecuencia del sintetizador es de 110 Hz. También es importante mencionar que se encontró un corrimiento en frecuencia de 9700 Hz tanto en la placa de desarrollo como en todos los dispositivos fabricados. Es decir, envía un comando al *driver* de la radio para que sintonice una frecuencia de 150,0097 MHz, el valor sintonizado resultante sera de 150,0000 MHz.

- Ancho de banda en recepción: en este caso, se utilizaron dos MD para caracterizar el ancho de banda en recepción: uno funcionando como receptor con una frecuencia de portadora fija y un ancho de banda de 17,4 kHz, registrando la potencia recibida (RSSI), y otro enviando mensajes de RF, cambiando el valor de frecuencia de portadora en pasos de 500 Hz. A partir de los diferentes valores de RSSI obtenidos en esté barrido para cada frecuencia portadora, se confeccionó el gráfico observado en la figura 5.3, siendo el ancho de banda efectivo de 17 kHz, acorde al valor configurado al realizar esta prueba.

- Adaptación: el diseño de la red de adaptación de RF fue obtenido a través del foro de consultas técnicas de Texas Instruments (como se menciono en el Capítulo 3), donde el diseño garantiza la adaptación tanto para transmisión como recepción en una carga de 50 Ω dentro de la banda 143 a 176 MHz. Para verificar la adaptación, se conectó el dispositivo en modo transmisor, transmitiendo un tono en 150 MHz y 14 dBm de potencia, a un analizador de espectros con un cable de conector BNC. El valor medido con el analizador de espectro fue de 13,4 dBm, estando dentro del rango de desviación esperado. Se pueden observar los valores de potencia recibidos para otras frecuencias de interés en la figura 5.4.

- RSSI medida por el dispositivo: es importante mencionar que los valores de RSSI registrados al recibir un mensaje utilizando el MD como receptor son 10 dB mayores al recibido realmente.

5.2.3. Caracterización de antena del MD

Para caracterizar la antena del MD, se realizaron las mediciones posicionando el transmisor y el receptor a un metro de altura y a 3,5 metros de distancia en todos los casos. Se colocó el cable que funciona como antena del receptor formando un semicírculo, para simular la geometría utilizada al colocar el transmisor en el animal, como se observa en la figura 1.2 del Capítulo 1.

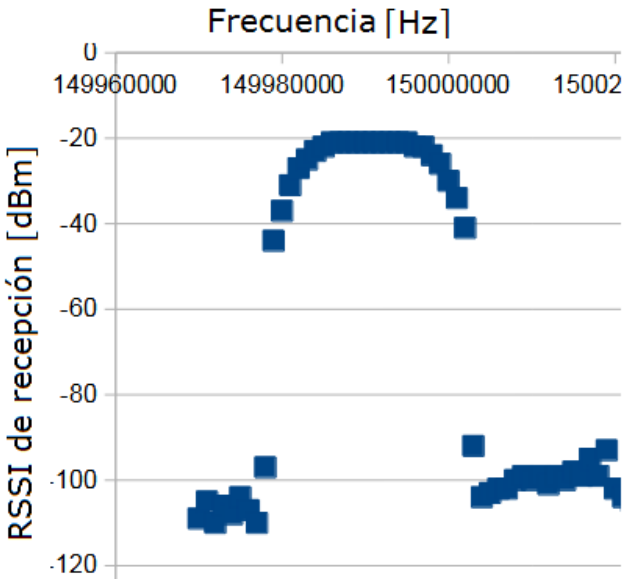


Figura 5.3: Valores recibidos (en dBm) para una transmisión de un tono único en diferentes valores de frecuencia (Hz).

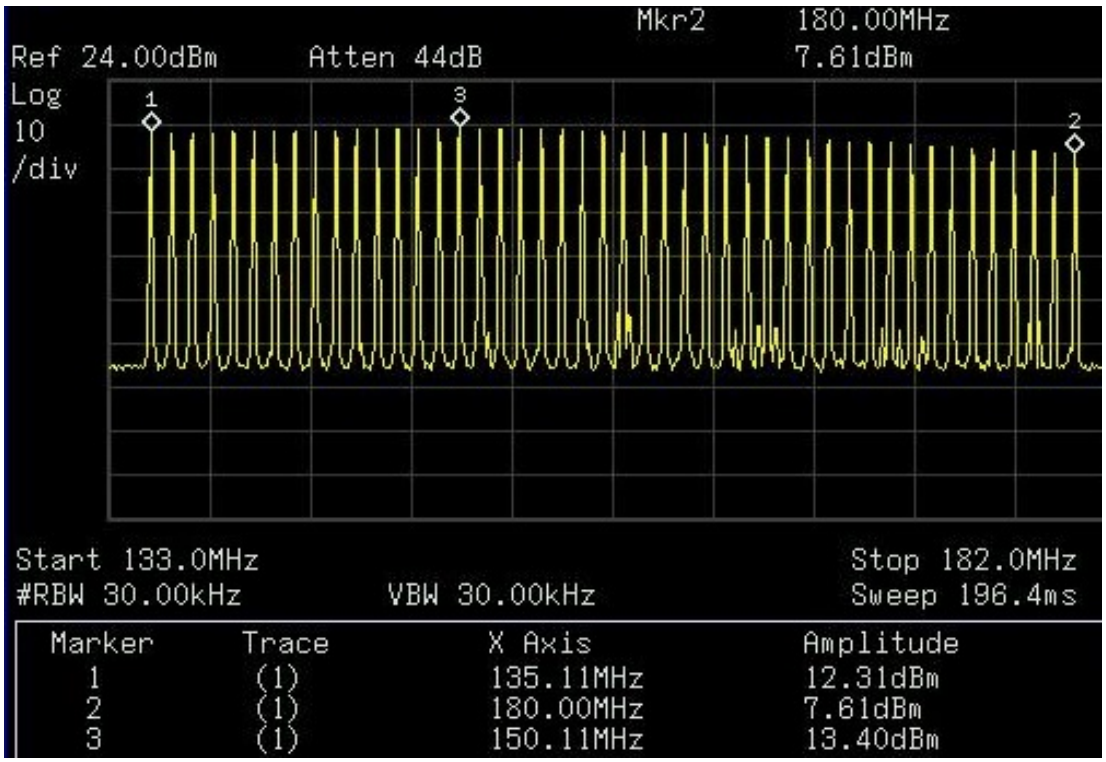


Figura 5.4: Valores de potencia (dB) recibidos por el analizador de espectro para transmisiones realizadas por el dispositivo MD a diferentes valores de frecuencia (MHz).

Transmisor	Receptor	Potencia Recibida [dBm]
Generador de funciones conectado a antena monopolo	Analizador de espectro conectado a cable de 28 cm	-36
Generador de funciones conectado a cable de 28 cm	Analizador de espectro conectado a antena monopolo	-36
MD conectado a cable de 28 cm	Analizador de espectro conectado a antena monopolo	-58

Tabla 5.3: Valores recibidos de potencia para diferentes combinaciones de transmisores y receptores.

Los resultados se presentan en la tabla 5.3, en la cual se observa que la red de adaptación del MD presenta pérdidas significativas cuando no se conecta a una antena que presente una impedancia de carga de 50 Ω, como es el caso de un cable.

Rendimiento de antena del MD

Para realizar esta medición, se utilizó al dispositivo MD transmitiendo un tono en la frecuencia de 150 MHz a 12,5 dBm, utilizando una antena tipo cable de 28 cm dispuesto en forma recta en este caso y en posición vertical, a un metro de altura. La recepción fue realizada utilizando un analizador de espectro conectado a una antena tipo Yagi-Uda en posición alineada con la antena transmisora para asegurar una máxima recepción de potencia. La distancia entre el transmisor y el receptor fue de 3,5 metros.

La potencia recibida por el analizador fue de -44 dBm. Para poder calcular las pérdidas, se debe considerar el efecto de las pérdidas por propagación en el aire, que corresponden a 27 dB para frecuencia de 150 MHz y una distancia de 3,5 metros.

Por lo tanto, el valor de pérdida de potencia se calcula como:

$$Pérdidas = 12{,}5 - (-44{,}5) - 27 \tag{5.1}$$

$$Pérdidas = 30\ dB \tag{5.2}$$

Es importante mencionar que estas pérdidas incluyen tanto las pérdidas de la antena transmisora del MD como las asociadas a la antena Yagi-Uda.

Comparación de recepción entre cable recto y cable curvo

Debido a los valores de pérdidas obtenidos en el ensayo anterior y a que las mediciones se realizaron utilizando el cable en una posición recta, distinta de la que se utilizará con el dispositivo (curva), se realizó una caracterización de la respuesta del sistema empleando diferentes disposiciones y largos de cable manteniendo el mismo banco de medición descripto anteriormente.

En la figura 5.5 se observan los valores de ganancia para diferentes largos de cable según si está colocado en forma recta (cuadrados azules) o formando una curva (rombos naranja), tal y como se colocaría en el caparazón de la tortuga. De este ensayo se observa que el valor óptimo de ganancia se obtiene para 58 cm de largo. Sin embargo, para valores de largo menores a 28 cm, la diferencia de ganancia entre el caso en el que se coloca el cable en forma recta y el caso curvo es prácticamente nula, por lo que el largo mínimo del cable recomendado es de 28 cm.

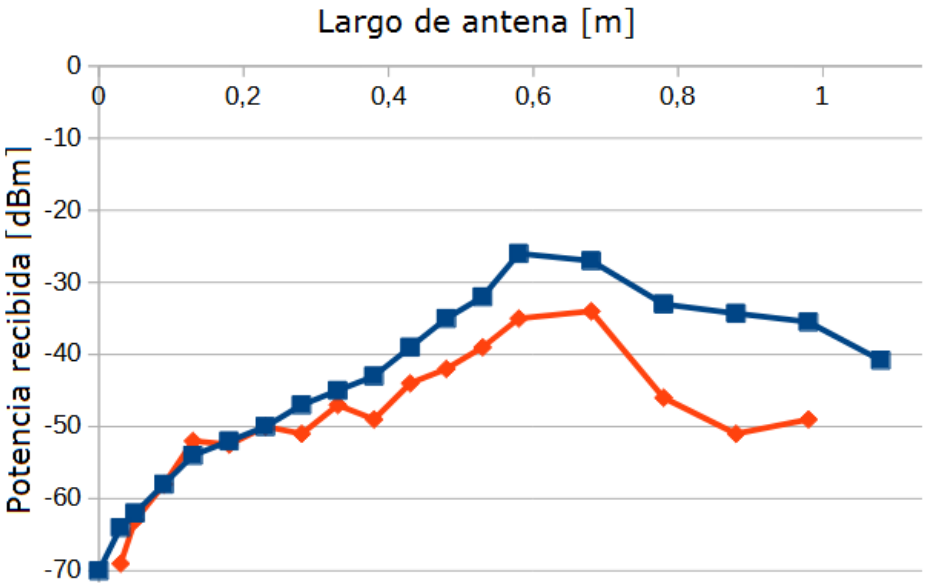


Figura 5.5: Potencia recibida (en dB) para cable curvo (rombos naranja) y cable recto (cuadrados azules) en función de largo del cable (en metros), realizando transmisiones en iguales condiciones.

Es importante mencionar que luego de realizar este ensayo, se evaluaron otras topologías con el fin de incrementar la potencia recibida, como realizar una bobina con un diámetro más reducido, obteniendo un peor rendimiento respecto de utilizar el cable dispuesto en forma curvada o recta. También se probó añadir una red L-C para mejorar la adaptación, sin obtener mejoras consistentes debido a las dificultades presentadas al momento de intentar caracterizar la impedancia del cable, al depender su comportamiento en radio frecuencia fuertemente de su geometría de disposición y orientación respecto del receptor. En este sentido, la disposición del cable en forma curva presentó un rendimiento mejor tanto con la antena direccional Yagi-Uda como con la antena omnidireccional monopolo.

5.2.4. Alcance

Las primeras pruebas de alcance se realizaron en el campus del Instituto Balseiro. Para las mismas se utilizaron las antenas monopolo (figura 5.6) y Yagi-Uda como receptoras conectadas a un DCS. Como transmisor se utilizó un MD con un cable

Antena Receptor	Alcance $[m]$	Potencia Rx $[dBm]$
Yagi-Uda	310	-98
monopolo	240	-97

Tabla 5.4: Resultados de pruebas de alcance realizadas en el campus del Instituto Balseiro.

de 50 cm como antena, realizando las mediciones con el dispositivo colocado en el suelo con el cable colocado en forma curva como se posicionaría sobre el caparazón de una tortuga, y colocando el dispositivo lejos de árboles u objetos metálicos ya que se comprobó al realizar el ensayo, que éstos, al posicionarse a menos de un metro del dispositivo, actúan como una antena que incrementa el patrón de radiación de la antena del MD y aumenta el valor de potencia transmitida.

Es importante mencionar que los valores de RSSI registrados en los ensayos de alcance fueron aquellos para los cuales se recibían paquetes en la DCS sin registrar errores en la recepción, con el MD localizado a la máxima distancia posible. En la tabla 5.4 se observan en forma simplificada los resultados de los ensayos realizados con línea de visión (LOS, por su sigla en inglés) entre el transmisor y el receptor.

Es importante mencionar que el alcance obtenido, cuando el transmisor y el receptor se encuentran sin obstáculos entre ellos, es considerablemente menor al esperado (5,2 km) en la estimación obtenida al final el Capítulo 3, lo cual se puede explicar debido al mal rendimiento de la antena transmisora del MD. Si se consideran las pérdidas medidas para esta antena (30 dB) en la hoja de cálculo utilizada en el Capítulo 3 para estimar el alcance, el valor máximo cambia a 213 metros sin obstáculos en la línea de visión, siendo un valor cercano al obtenido en los ensayos.

Figura 5.6: Imagen de antena tipo monopolo dentro del campus del Instituto Balseiro.

5.3. Mediciones en campo

En esta sección se mencionarán los resultados obtenidos en la mediciones hechas durante la campaña de conservación realizada en febrero del año 2023 en las cercanías de San Antonio Oeste, como se mencionó en el Capítulo 1 (figura 1.1).

5.3.1. Autonomía y tiempo de carga de batería

Durante la campaña se realizaron mediciones de autonomía con todos los sensores activos y sin utilizar las políticas de bajo consumo mencionadas en el Capítulo 4, debido al interés científico en recolectar datos de todos los sensores por periodos mayores a un día. Por este motivo, el valor máximo de autonomía obtenido fue de 2,38 días, siendo un valor acorde al calculado en la tabla 5.2.

El tiempo de carga de la batería fue de 1 hora en todos los casos, de acuerdo con lo esperado por el diseño.

5.3.2. Alcance

En la figura 5.7 se observa la estación colectora junto a la antena monopolo. Al momento de realizar los ensayos, se realizaron las mediciones en diferentes puntos del campo, escogidos de modo de poder tener una muestra de valores que represente mejor los diferentes tipos de obstáculos que se encuentran en el terreno, como son la altura de la vegetación o los desniveles del mismo.

En la tabla 5.5 se observan los valores de alcance máximos obtenidos con las antenas monopolo y Yagi-Uda. Los ensayos se realizaron siguiendo el mismo procedimiento que en las pruebas realizadas en el campus del Instituto Balseiro. El mismo consistió en colocar el dispositivo en el suelo, con la antena dispuesta en forma semi-circular simulando la geometría del caparazón de las tortugas terrestres. Es importante mencionar que el largo de antena del MD utilizado fue de 28 cm, debido a que es el mayor largo que permite colocarse en los caparazones de las tortugas terrestres a estudiar.

Como se observa en la tabla 5.5, el máximo alcance registrado fue de 270 metros con una RSSI de -103 dBm para el caso de la antena monopolo, y de 340 metros para la antena Yagi-Uda con una RSSI de -104 dBm, en ambos casos sin LOS, al no encontrarse zonas sin vegetación u obstáculos en el campo. Los valores más bajos de alcances máximos registrados al realizar estos ensayos en diferentes partes del campo fueron de 80 metros con la antena monopolo y 98 metros con la antena Yagi-Uda, observándose una variabilidad significativa del alcance máximo en función de donde se realizaban los ensayos.

Es importante mencionar que en la zona donde se recibió la señal con mayor alcance, la vegetación no sobrepasaba el metro de altura y no habían desniveles significativos

Figura 5.7: Estación receptora colocada en campo con antena monopolo de 2 metros de altura.

Antena Receptor	Alcance máximo $[m]$	Potencia Rx $[dBm]$
Yagi-Uda	340	-104
monopolo	270	-103

Tabla 5.5: Valores máximos de alcance y RSSI para los ensayos realizados en campo.

en el terreno. Por el contrario, en las zonas donde los árboles o arbustos alcanzaban la altura de la antena (2 metros) o se encontraban en el terreno desniveles se registraron los menores valores de alcance máximo. También se encontró en reiteradas oportunidades durante la realización de los ensayos que algunos obstáculos (como árboles de mucho volumen y altura) implicaban un menor alcance a pesar de que la distancia al receptor fuera menor que en otros puntos donde no se encontraban esa clase de obstáculos y el alcance máximo registrado era mayor.

El alcance registrado para la antena Yagi-Uda fue en promedio un 25 % superior al caso del monopolo. A su vez, se pudo utilizar exitosamente el receptor ATS R410 mencionado en el Capítulo 1, para detectar y localizar la señal del pulso KA proveniente del MD, verificando la compatibilidad del desarrollo realizado con el receptor RF utilizado habitualmente por el grupo de investigación.

En forma adicional, se realizaron pruebas al medir la potencia recibida cuando se coloca el dispositivo sobre el animal, y no se observaron diferencias significativas con respecto a colocar el mismo sobre la superficie de la tierra.

5.3.3. Sensores de temperatura

En la figura 5.8 se pueden observar los valores obtenidos con los sensores de temperatura correspondientes a la IMU y MCU durante un período de 2 días. La resolución de la medición del sensor del MCU es de 2 grados Celsius y su exactitud de $\pm$ 2,5 grados Celsius, mientras que el sensor de la IMU posee una resolución de 0,125 grados Celsius. Se visualiza una diferencia de aproximádamente dos grados entre ambos sensores, siendo mayor el valor adquirido por el sensor del MCU y encontrándose esta diferencia en el rango de exactitud de la medición del sensor del MCU. Es importante mencionar que independientemente del valor absoluto de temperatura adquirido, ambos sensores registran la misma variabilidad térmica.

5.3.4. Precisión en GNSS

Se evaluó la precisión del receptor GNSS utilizando un receptor GPS de mano Garmin eTrex 20. En la figura 5.9 se pueden observar las mediciones realizadas en campo comparando los valores de geolocalización obtenidos con el MD (rojo) y el receptor

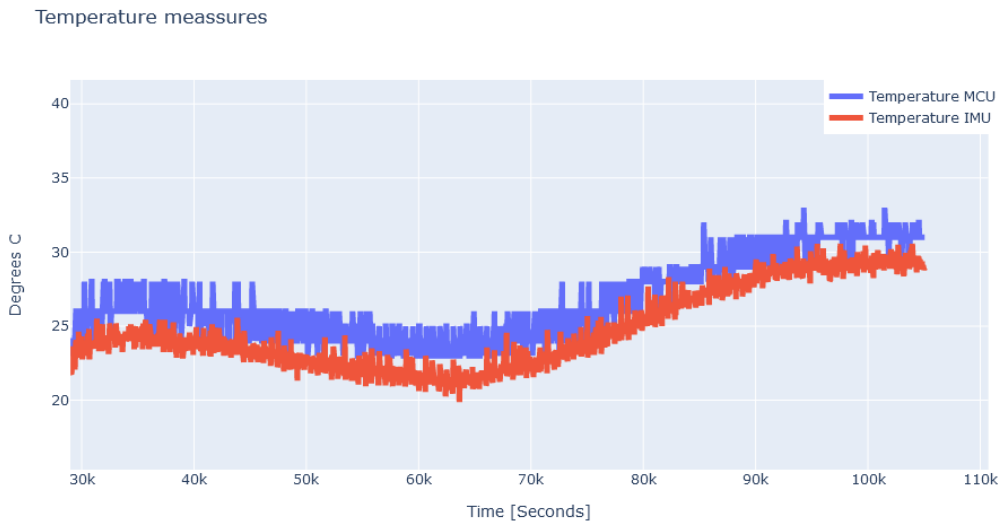


Figura 5.8: Valores de temperatura adquiridos de la IMU (rojo) y el MCU (azul) del MD.

Garmin (azul), junto a los valores de HDOP asociados. Se realizaron las mediciones en una línea recta de 50 metros, tomando las medidas cada 5 metros con el GPS Garmin y colocando el MD en las mismas posiciones durante 5 minutos de modo de adquirir la mayor cantidad de muestras para cada localización. Luego se realizó el promedio de los valores para obtener la localización y su desviación estadística asociada. Es importante mencionar que un valor de HDOP cercano a uno se asocia a una precisión estimada de 2,5 metros, y que la precisión con la que se configuró el receptor GNSS del MD es de 10 metros.

Figura 5.9: Trayectoria construida a partir de muestras de geolocalización adquiridas con el MD (rojo) y con el receptor GPS de mano Garmin (azul), junto a los valores de HDOP para cada punto.

En la figura 5.10 se pueden observar los valores de coordenadas de geolocalización (latitud y longitud) obtenidos, donde las líneas rectas representan el valor de error para el caso del receptor Garmin (azul) y desviación estándar en el caso del MD (rojo). Se observa que los valores medios son similares entre sí. En el caso del MD, la desviación estándar corresponde a un valor de 8 metros, siendo un valor esperado de acuerdo con la configuración (10 metros de precisión) utilizada en el MD.

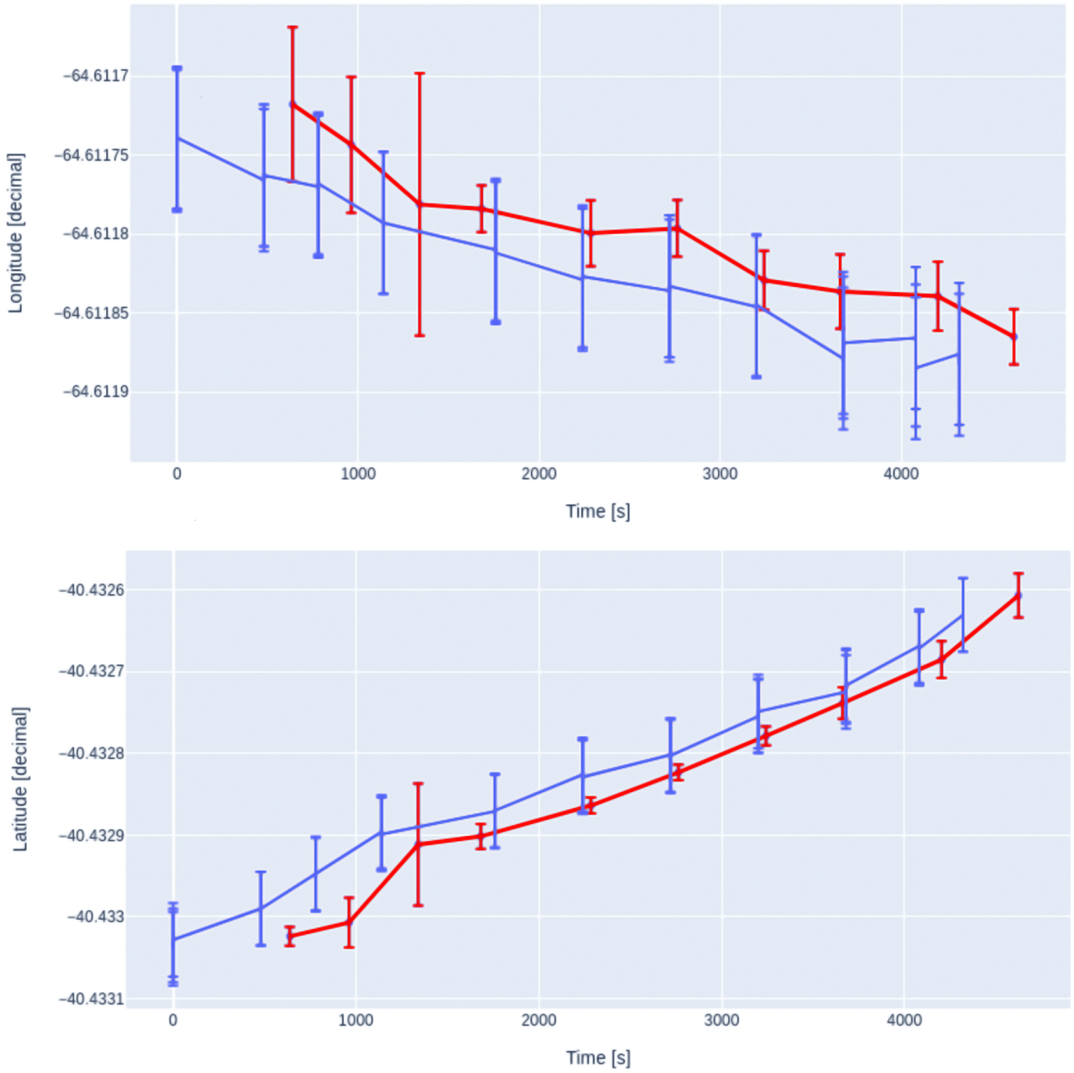


Figura 5.10: Valor medio de longitud/latitud obtenidos con el MD (rojo) y con el receptor GPS de mano Garmin (azul), junto a los valores de desviación estándar obtenidos para la medición de cada coordenada.

5.3.5. Tiempo medio de *fix* en receptor GNSS

A partir de los datos de geolocalización obtenidos al colocar el MD en una tortuga silvestre por más de 48 horas, se construyó el histograma de la figura 5.11 donde se grafica el tiempo que tardó el receptor GNSS en obtener una geolocalización válida

(denominado como "Tiempo de *fix*") para cada muestra del sensor tomada cada 10 minutos. Se observa cómo casi todos los tiempos para que el receptor logre entrar en *fix* son menores a 80 segundos, valor coherente con el utilizado para realizar la estimación de consumo y autonomía (60 segundos) en el Capítulo 4.

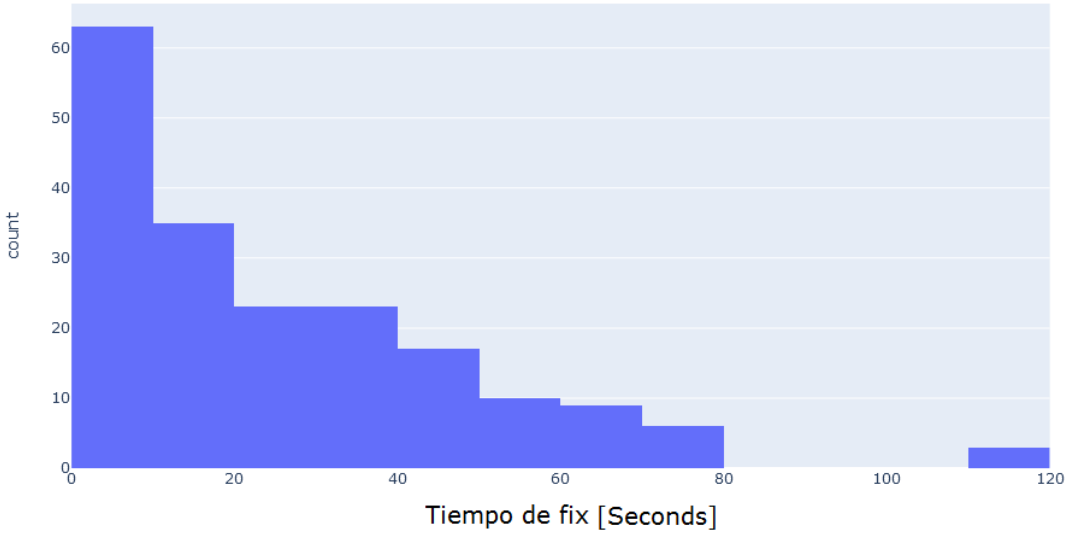


Figura 5.11: Histograma de tiempo de *fix* para receptor GNSS de MD luego de 48 horas de actividad en campo.

5.3.6. Algoritmo de detección de movimiento animal

En la figura 5.12 se pueden observar los valores obtenidos para los tres ejes del acelerómetro en una tortuga silvestre, a una tasa de muestro de 14,9 Hz. En la figura 5.13 se puede observar el resultado del algoritmo de detección de movimiento, aplicado sobre las señales del acelerómetro de la figura 5.12. Por convención, el valor 2 representa que se detectó movimiento en un intervalo muestreado de un segundo, mientras que el valor 1 significa que no se detectó movimiento en ese intervalo. Se observa que el movimiento ocurre con un cambio abrupto en el valor de alguno de los tres ejes y es acompañado por una oscilación si la tortuga continúa el movimiento y comienza a caminar. Por el contrario cuando el animal está en reposo se observa una señal plana con ruido de muy poca amplitud. Las pruebas realizadas con este algoritmo en las tortugas terrestres permiten el uso de los modos de bajo consumo mencionados en el Capítulo 4.

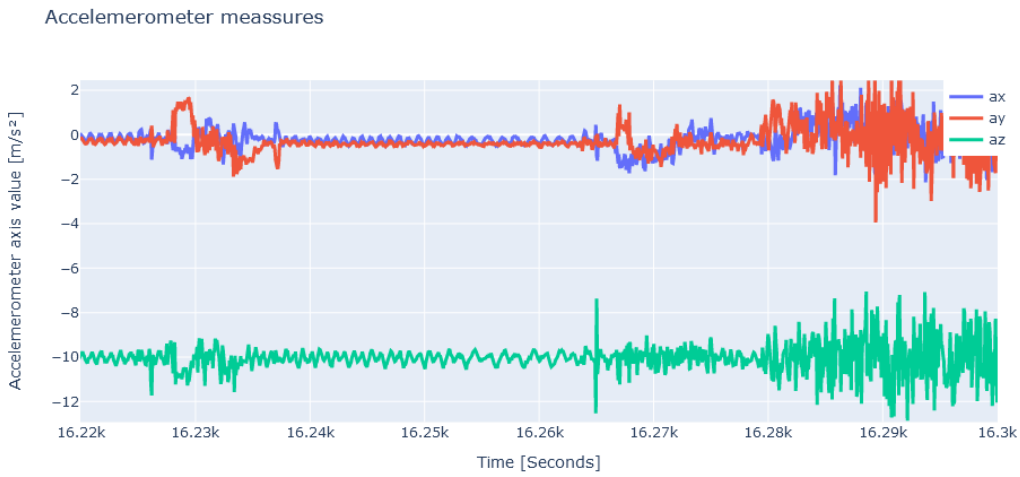


Figura 5.12: Ejemplo de señal de acelerómetro adquirida de una tortuga silvestre, donde los ejes X, Y, Z corresponden a los colores rojo, azul y verde respectivamente.

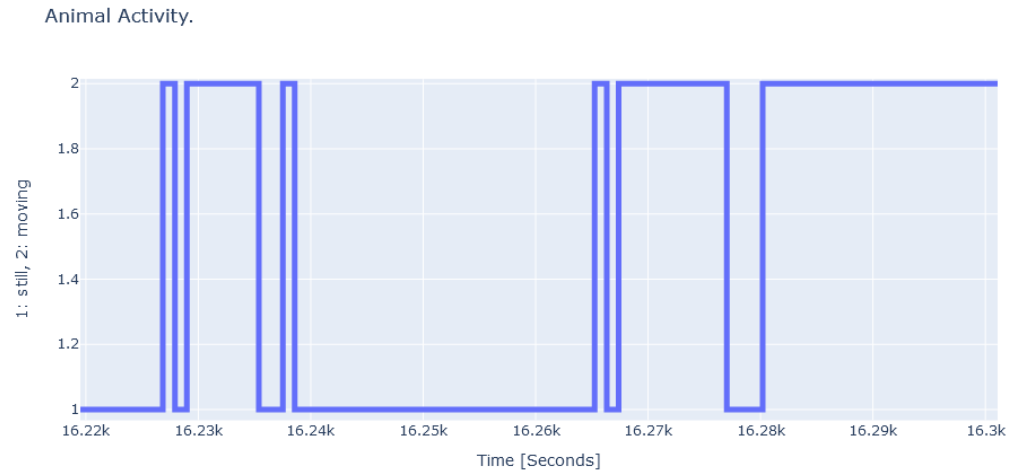


Figura 5.13: Resultado de clasificación de actividad animal para la señal del acelerómetro.

5.4. Publicación de los resultados

Los resultados de esta Tesis se publicaron en la revista Sensors, con referato internacional, compartiendo con la comunidad científica tanto el diseño del dispositivo como el *firmware* y los criterios elegidos para el mismo [44].

Además, se presentaron los avances del primer año de esta Tesis en la Reunión Nacional de Física Argentina en septiembre del año 2022, en forma de póster con título:"*Desarrollo de una red de sensores para el monitoreo de tortugas terrestres*".

Durante el desarrollo de este proyecto, el grupo se presentó al concurso "*IoT Into the Wild Contest for Sustainable Planet 2022*", obteniendo el primer puesto dentro de la categoría "*Innovative and Creative Proyect Award* " auspiciado por la comunidad hackster.io, siendo el premio del concurso una nota de crédito de 500 USD para la compra de productos de la empresa Seedstudio, la cual comercializa módulos y soluciones vinculadas a la tecnología LoRa®, internet de las cosas y *TinyML* [45].

Es importante mencionar que a partir del desarrollo de este trabajo se colaboró con el grupo para lograr el desarrollo y fabricación de una versión de menor tamaño del MD para utilizar en lagartijas, fortaleciendo el vínculo con otros grupos de investigación.

Finalmente, tanto los diseños del PCB como el código fuente del *firmware* expuestos en esta Tesis se encuentran disponibles en el siguiente repositorio de github: git@github.com:TortoisesSAO/CodesDevice.git

Capítulo 6

Conclusiones y trabajo futuro

Durante el desarrollo de esta Tesis de Maestría, se diseñó e implementó una familia de dispositivos para el monitoreo de comportamiento animal que fue utilizada exitosamente para conocer el movimiento de la tortuga terrestre *Chelonoidis chilensis* en su hábitat natural. Este desarrollo de bajo costo y de bajo consumo energético, resultó de suma importancia, en primer lugar porque dicha tortuga se encuentra en estado vulnerable y por lo tanto el conocimiento de su comportamiento contribuirá a poner en práctica estrategias para su conservación, y en segundo lugar porque el dispositivo despertó el interés de otros grupos de investigación del país, interesados en monitorear otras especies animales.

A continuación se resumen las principales conclusiones de cada capítulo de esta Tesis:

- En el Capítulo 1 se explico la importancia del monitoreo de la tortuga terrestre *Chelonoidis chilensis*, el contexto y la necesidad del equipo de investigación, resaltando la importancia de la participación del autor en su primera campaña de conservación de tortugas terrestres, lo que permitió fortalecer el trabajo interdisciplinario entre los miembros del grupo de investigación y hacer un relevamiento exitoso de las necesidades del mismo. A partir de este relevamiento se propuso una solución de bajo costo y de bajo consumo energético. La misma consiste en varios módulos de monitoreo que se colocan sobre las tortugas, una estación colectora de datos y un rastreador inalámbrico.

- En el Capítulo 2 se detallaron los requerimientos propuestos en función del relevamiento realizado. Cada módulo de monitoreo contiene un receptor GNSS, sensores inerciales (IMU), magnetómetro, giróscopo, sensor de temperatura y batería. Las especificaciones del *hardware* estuvieron limitadas principalmente por el peso máximo del dispositivo que no debió superar los 75 g para no perturbar el comportamiento del animal. Otro limitante importante fueron los costos y la

disponibilidad en el mercado. Es importante mencionar que la falta de disponibilidad de componentes a causa de la pandemia de COVID-19 presentaron tanto demoras como limitaciones en el desarrollo de este proyecto. Un claro ejemplo de este impacto es el caso del módulo GNSS utilizado, siendo uno de los de mayor consumo. Al ser el sensor de mayor consumo, y estar implementado en formato de módulo gracias a las decisiones de diseño, realizar un cambio del mismo por las opciones evaluadas no requeriría de cambios en la placa e impactaría significativamente en la autonomía del dispositivo. Lo mismo ocurre con la IMU, el segundo sensor de mayor consumo después del módulo GNSS, con la consideración de que se requeriría un rediseño de la placa de circuito impreso.

- En el Capítulo 3 se explicó en profundidad el diseño del *hardware*. Se realizó un exhaustivo estudio de mercado tanto para la elección del microprocesador como de los demás componentes, priorizando, además de los costos y de la disponibilidad, la facilidad para el desarrollo y depuración del *firmware*. Es importante aclarar que el posicionamiento de los módulos se realizó consultando permanentemente con el equipo interdisciplinario de investigación, de manera de facilitar el uso del dispositivo en el campo, por ejemplo facilitando la carga, la notificación al usuario a través de las luces led, o el prendido y apagado del mismo.

- En el Capítulo 4 se expusieron los detalles de la programación del *firmware*. Se diseñó una estructura con soporte de sistema operativo en tiempo real, planteando un esquema para la interacción de las diferentes tareas del sistema junto al soporte de diferentes modos de consumo funcionales en tiempo real basados en detección de actividad animal. Además se programó un *script* de decodificación de los datos en Python, para que los demás miembros del grupo de investigación puedan modificarla fácilmente en el futuro para el análisis de los datos registrados en la memoria del dispositivo de monitoreo.

- En el Capítulo 5 se expusieron los resultados de las pruebas del dispositivo desarrollado tanto en el Instituto Balseiro como en el campo, en las cercanías de San Antonio Oeste. En primer lugar se hace una estimación de los costos de fabricación del dispositivo, resultando de 113,5 USD, valor muy inferior al costo del prototipo que venía utilizando el grupo de investigación (250 USD). La participación en una segunda campaña de conservación permitió probar el dispositivo desarrollado en el campo, lográndose obtener una autonomía de más de dos días con todos los sensores adquiriendo a su tasa nominal, sin habilitar ninguno de los modos de optimización de consumo en función de la actividad de la tortuga. Luego se logró implementar exitosamente un algoritmo básico de detección de actividad animal, que permitió incrementar la autonomía del dispositivo consi-

derablemente.

En el caso del alcance, los resultados obtenidos en los ensayos en el Instituto Balseiro, muestran que el principal limitante proviene del mal rendimiento de la antena del dispositivo de monitoreo, que depende considerablemente tanto del largo del cable como de su disposición geométrica. En este sentido, la red de adaptación utilizada con el transceptor, provista por el fabricante, esta diseñada para una carga de 50 Ω, por lo cual se debería estudiar y analizar cómo rediseñar esta red para mejorar el rendimiento del cable como antena. Es interesante mencionar que en los ensayos se encontró que el valor de alcance se incrementaba cuando la altura de la antena de la estación colectora es mayor a la de la vegetación, por lo cual incrementar la altura de la misma puede favorecer el valor máximo de alcance.

A pesar de las limitaciones mencionadas, se logró cumplir con los objetivos generales del proyecto, y promover la difusión del trabajo en la comunidad científica al participar en un congreso de alcance nacional, realizar una publicación en una revista internacional, haber obtenido el primer puesto en un concurso de desarrollo y colaborar en el diseño de una versión más pequeña del dispositivo para el monitoreo de lagartijas.

Por estos motivos, se consideran cumplidos los objetivos planteados, brindando al grupo no solamente un sistema funcional, sino una plataforma como base para futuros desarrollos y aplicaciones que se adapten a las diferentes necesidades del grupo de investigación, presentando una mejora significativa respecto de las herramientas con las que contaba el mismo.

Como trabajo a futuro se plantea la implementación de herramientas de aprendizaje automático, con las cuales se podrían detectar otras actividades de las tortugas, entrenando una red neuronal con actividades clasificadas mediante observaciones de campo. En este sentido, el *hardware* del dispositivo desarrollado soporta la implementación de redes neuronales generadas con *TinyMl*, permitiendo embeber algoritmos de procesamiento más complejos en el equipo. En particular la detección de los nidos es de vital interés para la conservación de la tortuga estudiada. Finalmente, la publicación abierta de este desarrollo, se espera que contribuya a la implementación y mejora de este dispositivo en otros proyectos de investigación similares que apunten a la conservación.

Bibliografía

[1] Morales, J. M., Fortin, D., Frair, J. L., Merrill, E. H. Adaptive models for large herbivore movements in heterogeneous landscapes. *Landscape Ecology*, **20** (3), 301–316, 2005. 3

[2] Smouse, P. E., Focardi, S., Moorcroft, P. R., Kie, J. G., Forester, J. D., Morales, J. M. Stochastic modelling of animal movement. *Philosophical Transactions of the Royal Society B: Biological Sciences*, **365** (1550), 2201–2211, 2010. 3

[3] Baratchi, M., Meratnia, N., Havinga, P. J. M., Skidmore, A. K., Toxopeus, B. A. G. Sensing solutions for collecting spatio-temporal data for wildlife monitoring applications: A review. *Sensors*, **13** (5), 6054–6088, 2013. URL https://www.mdpi.com/1424-8220/13/5/6054. 3

[4] Hebblewhite, M., Haydon, D. T. Distinguishing technology from biology: a critical review of the use of gps telemetry data in ecology. *Philosophical Transactions of the Royal Society B: Biological Sciences*, **365** (1550), 2303–2312, 2010. 3, 4

[5] Marin, F. Human and animal motion tracking using inertial sensors. *Sensors*, **20** (21), 6074, 2020. 3

[6] Cain, P. W., Cross, M. D. An open-source hardware gps data logger for wildlife radio-telemetry studies: a case study using Eastern box turtles. *HardwareX*, **3**, 82–90, 2018. 4

[7] Barbuti, R., Chessa, S., Micheli, A., Pallini, D., Pucci, R., Anastasi, G. Tortoise@: a system for localizing tortoises during the eggs deposition phase. *Atti Societa Toscana Scienze Naturali, memorie B*, **119**, 89–95, 2012. 4

[8] Barbuti, R., Chessa, S., Micheli, A., Pucci, R. Localizing tortoise nests by neural networks. *PloS one*, **11** (3), e0151168, 2016. 4

[9] Cei, J. M. Reptiles del centro, centro-oeste y sur de la Argentina: Herpetofauna de las zonas áridas y semiáridas, tomo 4. Museo regionale di scienze naturali Torino, 1986. 4

[10] Richard, E. Espectro trófico de *Chelonoidis chilensis* (Chelonii: Testudinidae) en la provincia fitogeográfica del Monte (Mendoza, Argentina). *Cuadernos de Herpetología*, **8**, 1994. 4

[11] Richard, E. Tortugas de las regiones áridas de Argentina. L.O.L.A., 1999. 4

[12] Waller, T., Micucci, P. Land use and grazing in relation to the genus Geochelone in Argentina. En: Proceedings: Conservation, Restoration, and Management of Tortoises and Turtles-An International Conference, págs. 2–9. 1997.

[13] Sánchez, J., Alcalde, L., Bolzan, A. D., Sanchez, R. M., del Valle Lazcóz, M. Abundance of Chelonoidis chilensis (Gray, 1870) within protected and unprotected areas from the Dry Chaco and Monte Eco-regions (Argentina), 2014. 4

[14] Burkart, R., Bárbaro, N., Sánchez, R., Gómez, D. Ecorregiones de la Argentina. Administración de Parques Nacionales y Secretaría de Recursos Naturales y Desarrollo Sustentable. Argentina: Buenos Aires. *Ecological Engineering*, (30), Pp–43, 1999. 4

[15] Prado, W. S., Waller, T., Albareda, D. A., Cabrera, M. R., Etchepare, E. G., Giraudo, A. R., *et al.* Categorización del estado de conservación de las tortugas de la República Argentina. *Cuadernos de herpetología*, **26**, 2012. 4

[16] IUCN. The iucn red list of threatened species, n.d. URL https://www.iucnredlist.org/species/9007/1294968/. 4

[17] Zhang, Y., Li, Z., Liu, W., et al. A comprehensive analysis of the 2021 global chip shortage. *IEEE Access*, **9**, 55825–55838, 2021. 13

[18] Wei, Q., Jiang, Z., Xie, Y., et al. Understanding the impact of COVID-19 on the global semiconductor industry. *Journal of Electronic Materials*, **50** (6), 3475–3482, 2021.

[19] Su, X., Zeng, X., Wang, Y., et al. A survey on the 2021 global semiconductor shortage: Causes, impacts, and solutions. *IEEE Transactions on Semiconductor Manufacturing*, **34** (2), 163–179, 2021. 13

[20] Shin, W., Lee, S., Kim, S., Lee, S., Kim, H.-j., Lee, S., *et al.* TinyML design challenges: Opportunities and risks. *arXiv preprint arXiv:2009.07161*, 2020. 15, 20, 25

[21] Maarouf, A., Jong, T. v. d. EdgeImpulse: A platform for Tiny Machine Learning on edge devices. *IEEE Consumer Electronics Magazine*, **10** (3), 48–54, 2021. 15

[22] Bai, H., Gao, J., Wang, Y., Liu, J., Li, M., Zhang, F. TinyML on OS-Enabled MCUs: Challenges, opportunities, and tools. *IEEE Embedded Systems Letters*, **12** (4), 100–103, 2020. 20, 25

[23] Dostert, K., Boubekeur, M., Chen, L., Winkelmann, S. Cortex-M4 processor optimization for real-time digital signal processing. *IEEE Transactions on Industrial Electronics*, **65** (1), 248–256, 2018.

[24] Hemmati, H. R., Amiri, A. Real-time digital signal processing on ARM Cortex-M4-based microcontrollers. *International Journal of Electronics*, **103** (4), 648–663, 2016. 20, 25

[25] Bisio, A., Sciarrone, A., Lavagetto, F., Sabella, R. Evaluation of NB-IoT and LoRa for smart city applications. *IEEE Internet of Things Journal*, **5** (2), 858–867, 2018. 24

[26] Maniezzo, D., Cominelli, L., Palazzi, V., Mamei, M. Design and evaluation of a narrowband IoT system for environmental monitoring. *IEEE Internet of Things Journal*, **4** (6), 1752–1762, 2017. 24

[27] Lora alliance® overview. URL https://lora-alliance.org/about-lorawan/. 24

[28] Lorawan™ specification. URL https://lora-alliance.org/resource_hub/lorawan-specification-v1-1/.

[29] Lorawan technology overview. URL https://www.semtech.com/lora/lora-applications. 24

[30] Douglass. Design Patterns for Embedded Systems in C: An Embedded Software Engineering Toolkit. Newnes, 2010. 45, 46

[31] Buttazzo, G. Hard real-time computing systems: predictable scheduling algorithms and applications. Springer, 2011. 47, 56, 63

[32] Xiang, M., Zhou, M. A survey of scheduling algorithms in real-time embedded systems. *IEEE Access*, **5**, 11178–11193, 2017.

[33] Baruah, S. Response time analysis of real-time systems. *Foundations and Trends in Real-Time Systems*, **1** (1), 1–144, 2011. 47

[34] TI-RTOS Wiki. URL https://processors.wiki.ti.com/index.php/TI-RTOS. 47

[35] Texas Instruments. TI-RTOS User's Guide. 2021. URL https://www.ti.com/lit/ug/spruhd4s/spruhd4s.pdf.

[36] Ravindran, S., Srikantan, N. Analysis of TI-RTOS on TI's Sitara AM335x Processor. En: 2018 International Conference on Embedded Systems (ICES). 2018. 47

[37] Texas Instruments. SimpleLink MSP432 SDK user's guide. https://software-dl.ti.com/simplelink/esd/simplelink_msp432_sdk/2.40.00.10/docs/tiposix/Users_Guide.html, 2018. 48

[38] LinuxHint. Posix standard explained, n.d. URL https://linuxhint.com/posix-standard/. 48

[39] Yakindu statechart tools. URL https://www.itemis.com/en/yakindu/state-machine/. 50

[40] Rajkumar, R., Lee, I. Real-time computing: the challenge of future automotive systems. *Proceedings of the IEEE*, **85** (3), 366–377, 1997. 59

[41] Liu, J. W. S. Real-Time Systems. Wiley, 2017. 63

[42] Clements, A. An Introduction to Real-Time Systems: From Design to Networking with C/C++. Springer, 2011. 63

[43] Fatfs - generic fat filesystem module tools. URL http://elm-chan.org/fsw/ff/00index_e.html. 70

[44] Kazimierski, L., Oliva Trevisan, A., Kubisch, E., Laneri, K., Catalano, N. Design and development of a family of integrated devices to monitor animal movement in the wild. *Sensors*, **23** (7), 3684, 2023. Oliva Trevisan and Kazimierski contribuyeron igualmente. 101

[45] Kazimierski, L., Kolton, A., Kubisch, E., Laneri, K., Echave, M., Catalano, N., *et al.* Study of animal movement equipment design and development, 2022. URL https://www.hackster.io/471203/study-of-animal-movement-equipment-design-and-development-febb17. 101

Agradecimientos

A mis directores, sin los cuales habría sido absolutamente imposible haber realizado esta tesis, por su motivación, compromiso y calidad humana.

A la Dra. Laila Kazimierski por su enorme colaboración, motivación y espíritu inquieto.

A Erika Kubisch y María Eugenia Echave, por su apertura y espíritu humano.

A mi familia, por su apoyo incondicional. A mi tío Valentín, por ser un ejemplo de profesional y persona.

A Cali y Franco por su apoyo.

A Gasti y Horacio por su inspiración y empuje en perseguir mi crecimiento profesional.

A José Quinteros y Lucas Grigolato por su confianza.

A Marcos y Dimitri, por su incondicional apoyo y amistad.

Al Dr.José Urriza y Dr.Francisco Paez por adentrarme en este vasto mundo que es el de los sistema operativos en tiempo real.

A la Universidad Nacional de Mar del Plata por permitirme formarme como ingeniero electrónico. Al Instituto Balseiro por permitirme realizar esta maestría. Al Estado Nacional Argentino, por permitirme formarme en esta maravillosa profesión que es la ingeniería.

Y a todas las personas que alguna vez me ayudaron.