Universidad Tecnológica Nacional

Proyecto Final

Sistema autónomo de monitorización acústica para ganadería

Autor:

- *Horst Tabacchi, Matías*

Director:

- *Dr. Martínez Rau, Luciano*

Codirector:

- *Ing. Maggiolo, Gustavo*

Proyecto final presentado para cumplimentar los requisitos académicos para acceder al título de Ingeniero Electrónico.

en la

Facultad Regional Paraná

Noviembre de 2023

Declaración de autoría:

Yo declaro que el Proyecto Final “Sistema autónomo de monitorización acústica para ganadería” y el trabajo realizado son propios. Declaro:

- Este trabajo fue realizado en su totalidad, o principalmente, para acceder al título de grado de Ingeniero Electrónico, en la Universidad Tecnológica Nacional, Regional Paraná.
- Se establece claramente que el desarrollo realizado y el informe que lo acompaña no han sido previamente utilizados para acceder a otro título de grado o pregrado.
- Siempre que se ha utilizado trabajo de otros autores, el mismo ha sido correctamente citado. El resto del trabajo es de autoría propia.
- Se ha indicado y agradecido correctamente a todos aquellos que han colaborado con el presente trabajo.
- Cuando el trabajo forma parte de un trabajo de mayores dimensiones donde han participado otras personas, se ha indicado claramente el alcance del trabajo realizado.

Firmas:

-
-
-

Fecha:

Agradecimientos:

Al Dr. Martínez Rau, Luciano, por brindarme la oportunidad de desarrollar este proyecto. También por su generosidad y por estar siempre dispuesto a ayudarme con sus vastos conocimientos.

Al Ing. Maggiolo, Gustavo, por su ser un gran guía en momentos cruciales. Sus amplios conocimientos y experiencia en el campo de los microcontroladores fueron de gran ayuda.

Mi sincero agradecimiento al Instituto de investigación en señales, sistemas e inteligencia computacional (*sinc(i))* por su invaluable contribución en el área. Agradezco a la Facultad Regional Paraná por brindarme una formación académica excepcional.

A mi familia, que me ha apoyado desde el inicio de mi formación y han contribuido a poder dedicarme de lleno al estudio.

A mi novia por acompañarme y darme el soporte que necesite en muchos momentos.

A mis amigos/as, y compañeros de trabajo con los que he compartido en esta gran etapa.

Horst Tabacchi, Matías

Universidad Tecnológica Nacional

Abstract

Facultad Regional Paraná

Ingeniero en Electrónica

Sistema autónomo de monitorización acústica para ganadería

Horst Tabacchi, Matías

Abstract:

The present Autonomous Acoustic Monitoring System for livestock was developed using the BluePill development board that incorporates the STM32F103C8T6 microprocessor. It includes a georeferencing system and a transmission system, which utilizes long-range technology (LoRa).

The system implements the CBEBA machine learning algorithm, capable of classifying 4 types of jaw movements. The algorithm utilizes the instantaneous power envelope of the input signal and incorporates features that enhance noise immunity.

A prototype embedded system was achieved that presented a great efficiency in the classification of events, serially processing audio samples corresponding to recordings of cattle in the process of free grazing.

Keywords:

Microcontroller, processing, algorithm, machine learning.

Resumen:

El presente Sistema autónomo de monitorización acústica para ganadería, se desarrolló mediante el uso de la placa de desarrollo BluePill que incorpora el microprocesador STM32F103C8T6. Incluye un sistema de georreferenciación, y un sistema de transmisión, el cual hace uso de tecnología de largo alcance (Lora).

En el mismo se implementa el algoritmo de aprendizaje maquinal CBEBA, el cual es capaz de clasificar 4 tipos de movimientos mandibulares. El algoritmo hace uso de la envolvente de potencia instantánea de la señal de entrada, e incorpora características que mejoran la inmunidad frente al ruido.

Se logro conseguir un sistema embebido prototipo que presentó una gran eficiencia en la clasificación de eventos, procesando en forma serial muestras de audio correspondientes a grabaciones de bovinos en proceso de pastoreo libre.

Palabras Clave:

Microcontrolador, procesamiento, algoritmo, aprendizaje maquinal.

Índice:

Lista de Figuras:

Lista de Tablas

Lista de Abreviaciones y Símbolos

OCDE - Organización para la Cooperación y el Desarrollo Económicos
FAO - Organización de las Naciones Unidas para la Alimentación y la Agricultura
EEA - Estaciones Experimentales
UNCa - Universidad Nacional de Catamarca
FTYCA - Facultad de Tecnología y Ciencias Aplicadas
RC - Ruminaion chew (Rumiar – morder)
GC - Grazing chew (pastar – masticar)
B - Bite (morder)
CB - Chew bite (Masticar – morder)
NMEA - National Marine Electronics Association
RTCM - Radio Technical Commission for Marine Services
GGA - Global positioning system fix data
RISC - Reduction Instruction Set Computing
KB - Kilobyte
SRAM - Static random access memory
UART - Universal Asynchronous Receiver / Transmitter
CAN - Controller Area Network
I^2C - Inter Integrated Circuits
USB - Universal Serial Bus
SPI - Serial Peripheral Interface
LIN - Local Interconnect Network
IrDA - Infrared Data Association
ADC - Analog to Digital Converter
PWM - Pulse Width Modulation
CBEBA - Chew Bite Energy Based Algorithm
CBIA - Chew Bite Intelligent Algorithm
GPS - Global Positioning System
LoRa - long Range
CSS - Chirp Spread Spectrum
ISM - Industrial, Scientific, and Medical
ENACOM - Ente Nacional de Comunicaciones
MPPT - Maximum Power Point Tracking
PCB - Printed Circuit Board
CPU - Central Processing Unit
MOSI - Master Output Slave Input
MISO - Master Input Slave Output
SS - Slave selection
SCLK - Serial Clock
V - Volt
mW - mili vatio
W - Vatio
Hz - Hertz

MHz	- Mega Hertz
° C	- Grados Celsius
s	- Segundos
mA	- mili ampere
mAh	- mili ampere hora

Dedicado a:

Mis padres por sus sacrificios al brindarme la oportunidad de perseguir mis sueños, y por su inquebrantable apoyo a lo largo de este viaje.

Capítulo 1: Introducción

El presente proyecto surge en base a una propuesta realizada por el personal que conforma el grupo de investigación de ganadería de precisión de sinc(i) CONICET.

El mismo, consistió en la realización de un sistema embebido que debía ser capaz de implementar y hacer uso de un algoritmo de aprendizaje automático desarrollado por ellos.

Con este dispositivo es posible obtener así los eventos masticatorios que suceden en un determinado periodo de tiempo en el ciclo alimenticio de un bovino, siendo de gran interés para la gestión de los sistemas de pastoreo libre. Estos eventos masticatorios reflejan el comportamiento del animal y su bienestar.

Una diferencia notable con respecto a otros productos o proyectos previos es que este sistema se centra exclusivamente en la clasificación de eventos masticatorios, sin realizar el registro de variables adicionales como la temperatura, por ejemplo.

Se utiliza un microprocesador cortex-M3, que es más potente en comparación con otros casos en los que se han empleado microcontroladores PIC. A pesar de que estos últimos presentaban un funcionamiento correcto, se encontraban con limitaciones significativas y requerían de componentes auxiliares, como tarjetas de memoria, para poner en funcionamiento el sistema.

En este caso, la memoria de la placa de desarrollo con la que se trabaja, resulta adecuada y suficiente para el almacenamiento del algoritmo, la realización del procesamiento, y el almacenamiento de una determinada cantidad de resultados.

Se ha incorporado un sistema de georreferenciación al dispositivo para obtener datos complementarios que se asocian a los resultados del procesamiento.

Es importante señalar que este desarrollo se limita a pruebas de laboratorio y no abarca el desarrollo de la etapa de adquisición y acondicionamiento de las señales, ni tampoco el posterior procesamiento de los datos resultantes.

Capítulo 2: Descripción de la problemática

Debido al aumento de la población a nivel mundial, se acompaña consigo un aumento del consumo de carnes, y por ende un aumento en la producción de esta.

Teniendo en cuenta las proyecciones para el 2023, generado por la OCDE-FAO respecto del mercado de ganados y carnes, se estimó que en 2023 la producción mundial de carnes aumentaría un 19 % (57,7 millones de toneladas). Dentro de lo que se destaca a Argentina como uno de los países de más rápido crecimiento ganadero [1].

Sergio René Araujo, economista de la FAO, señaló que, la producción agrícola y pesquera en Latinoamérica crecerá un 14 % con miras a 2031. Donde el 28% corresponde al sector ganadero [2].

Este escenario plantea la preocupación sobre cómo satisfacer esta creciente demanda, manteniendo al mismo tiempo una producción de excelente calidad que sea sostenible, segura y respetuosa con el medio ambiente.

2.1 Contribución al presente desarrollo

Como ya se conoce, en las últimas dos décadas aproximadamente, el sector ganadero ha sido testigo de un avance tecnológico sin precedentes. El desarrollo y adopción de dispositivos y métodos innovadores, han permitido enfoque significativo en la mejorar de la eficiencia de los sistemas de producción y el bienestar de los animales.

Esta evolución ha sido impulsada por la creciente necesidad de los productores de obtener información más precisa sobre los estados nutricionales de los animales. Esto implica garantizar una alimentación adecuada y balanceada, lo que se traduce en animales más sanos y productivos.

Esto no solo mejora la salud de los animales, sino que también optimiza los recursos y la productividad en la explotación ganadera.

Una de las principales ventajas de estos dispositivos es su capacidad para alertar al productor ante posibles problemas de salud o situaciones inesperadas que puedan afectar el bienestar de los animales. Esto conduce a una reducción significativa de pérdidas y gastos innecesarios, brindando una mayor tranquilidad a los ganaderos.

Con el transcurrir del tiempo, los desarrollos en el área, se han ido integrando con la inteligencia artificial y el aprendizaje automático, por lo que se abre así, un abanico de posibilidades para nuevos desarrollos.

Por esta razón, para el presente proyecto se ha desarrollado una nueva versión de un dispositivo con el que ya se ha dado los primeros pasos, utilizando esta vez un microcontrolador de mayor potencia, de bajo consumo, y de una tecnología más nueva.

2.2 Antecedentes de desarrollos en el área

En cuanto a los desarrollos que se han realizado en el área, se pueden mencionar algunos, como por ejemplo los siguientes:

1. *Sistema georeferenciador con parcelamiento virtual y adquisidor de sonidos masticatorios de rumiantes en pastoreo extensivo*.

El mismo se presentó en el Congreso Argentino de Sistemas Embebidos por Aranda et al. (2012). En este trabajo se describe el hardware electrónico diseñado para el registro de los sonidos masticatorios del ganado, y también el registro de la posición del animal para guiarlo a determinadas áreas de pastoreo. Utiliza la tecnología de microcontroladores PIC y posee un módulo GPS, slot para tarjeta SD y dos canales de audio. Se trata de un sistema que permite su configuración mediante interfaz gráfica, pudiéndose cargar las coordenadas que delimitan a las parcelas. Entonces cuando el animal está cerca del límite, se activa una alarma sonora. Si se aproxima aún más comienza a actuar un sistema eléctrico, siendo este el sistema de control de

parcelamiento virtual que busca eliminar las divisiones de alambre y/o boyeros. Para analizar la información de este dispositivo se debe conectar la tarjeta SD a una computadora, y descargar los datos en un software diseñado para el tratamiento de estos [3].

2. *Dispositivo para monitoreo del desplazamiento y la condición del ganado en sistemas pastoriles extensivos.*

Dicho proyecto se llevó a cabo por el grupo de trabajo EEA Catamarca – La Rioja, mediante UNCa y FTYCA. Este dispositivo cuenta con modulo GPS para monitoreo del desplazamiento y la condición del ganado en sistemas pastoriles extensivos. Posee distintos sensores y también tarjeta de memoria. El fin de este, es la resolución de la problemática de la toma de datos de animales en sistemas extensivos, como lo es la temperatura y posición. Y así poder inferir el comportamiento de los animales en pastoreo y como se relaciona con el paisaje que lo rodea [4].

3. *Ganadería de precisión: Sistema de medición automatizado para la evaluación animal de consumo residual y comportamiento.*

El proyecto se desarrolla en el Centro Regional La Pampa – San Luis, en conjunto con la Universidad Central de Queensland de Australia. Este, parte de la perspectiva del “Internet de las Cosas” y permite evaluar el consumo individual y el comportamiento de cada animal perteneciente a un rodeo determinado. Se basa en la automatización, el desarrollo y optimización de comederos inteligentes para el monitoreo del consumo y comportamiento individual del ganado ovino y bovino. Al proveer información precisa, este sistema hace posible seleccionar aquellos individuos que son más eficientes desde el punto de vista de la conversión de alimento a carne. Cada animal tiene un chip con caravana electrónica en su oreja. A su vez, cada comedero tiene una antena y una batea con el alimento, dispuesta sobre una balanza. Cuando un animal mete la cabeza en una reja, la antena identifica la caravana y detecta la identidad del animal y la cantidad de alimento consumido. Al final

del día, el software que controla el sistema arroja la cantidad de alimento qué consumió cada animal, en qué horario y en cuál de los comederos, entre otra información. Su fin es la optimización de los costos de la alimentación de los animales engordados a corral, analizando a cada animal de manera individual [5].

Capítulo 3: Fisiología digestiva del bovino

Los bovinos son animales herbívoros, lo que significa que se alimentan principalmente de plantas. Su sistema digestivo está diseñado para descomponer y extraer nutrientes de la materia vegetal.

El proceso de digestión comienza en la boca, donde los alimentos son masticados y mezclados con saliva. Luego, el alimento pasa al estómago, donde se descompone y se mezcla con enzimas digestivas. Finalmente, los nutrientes son absorbidos en el intestino y eliminados como desechos.

A continuación, se describe con mayor detalle cada una de las partes del sistema digestivo de los bovinos.

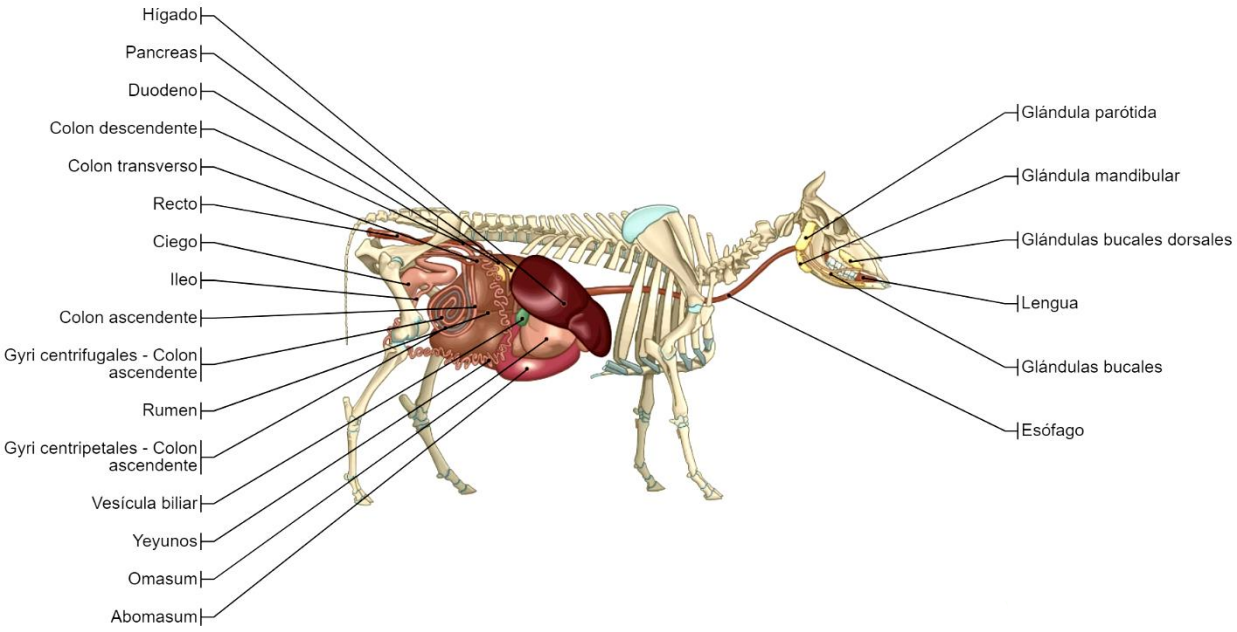

Figura 1. Esplacnología de un bovino [6].

3.1 Composición del sistema digestivo bovino

A. Boca

La boca de los bovinos está diseñada para masticar y triturar la materia vegetal antes de que sea digerida. Los dientes de los bovinos están especialmente adaptados para este propósito, con grandes molares y premolares que son capaces de triturar los alimentos fibrosos.

Además, los bovinos tienen una lengua áspera que les permite arrancar la hierba de la tierra y llevarla a la boca. La saliva secretada por las glándulas salivales lubrica el bolo alimenticio, lo que facilita la deglución y la digestión de los alimentos [7].

B. Estómago

El estómago de los bovinos está dividido en cuatro compartimentos: el rumen, el retículo, el omaso y el abomaso. Estos compartimentos trabajan juntos para descomponer y fermentar la materia vegetal antes de que sea digerida.

El rumen es el primer compartimento del estómago y es el sitio principal de fermentación. En el rumen, las bacterias y otros microorganismos, descomponen los carbohidratos y las proteínas en compuestos más simples, como los ácidos grasos y los aminoácidos. El rumen también es capaz de descomponer la celulosa, un componente de la pared celular de las plantas, lo que permite a los bovinos extraer nutrientes de la materia vegetal fibrosa. Además, es capaz de absorber ciertos nutrientes, como el agua y los iones [8].

El retículo es el segundo compartimento del estómago y trabaja en conjunto con el rumen para descomponer los alimentos y fermentar la materia vegetal. La superficie del interior del retículo tiene la apariencia de un "panal". Donde, las partículas más pequeñas y densas pasan a través de la apertura retículo-omaso, y las más grandes y menos densas regresan hacia el rumen ventral [9]. Por lo que, el retículo también es responsable del filtrado de la ingesta, y de la retención de objetos extraños.

El omaso es el tercer compartimento del estómago y es responsable de absorber agua y electrolitos de los alimentos. El omaso también trabaja para descomponer los alimentos y fermentar la materia vegetal.

El abomaso es el cuarto compartimento del estómago, y es el equivalente al estómago humano. El abomaso secreta ácido clorhídrico y enzimas digestivas que descomponen aún más los alimentos y los nutrientes son absorbidos en el intestino delgado. [8]

C. Intestino delgado

La función principal del intestino delgado en los bovinos es la digestión y absorción de los nutrientes.

El alimento digerido en el rumen pasa al omaso, donde se separa el líquido del material sólido, y luego se mueve hacia el abomaso para la digestión ácida.

A continuación, el alimento pasa al intestino delgado, donde se mezcla con enzimas digestivas secretadas por el páncreas y las células del revestimiento intestinal. Las enzimas secretadas por el páncreas y la superficie del intestino delgado digieren proteínas, carbohidratos y grasas [9].

Los nutrientes, son absorbidos luego, por las células del revestimiento intestinal y pasan al torrente sanguíneo para su distribución a través del cuerpo. El agua también se absorbe en el intestino delgado, lo que ayuda a mantener el equilibrio hídrico del cuerpo [8]. La longitud del intestino delgado en los bovinos es de aproximadamente 46 metros, lo que les permite extraer una gran cantidad de nutrientes de los alimentos [9].

D. Intestino grueso

El intestino grueso de los bovinos es responsable de la absorción final de agua y la eliminación de desechos sólidos. El intestino grueso está compuesto por el ciego, el colon y el recto. En el ciego, se produce la fermentación final de los alimentos y la absorción de agua y electrolitos. El colon, se encarga de la absorción final de agua y la formación de

heces sólidas. Finalmente, las heces son eliminadas del cuerpo a través del recto y el ano [8].

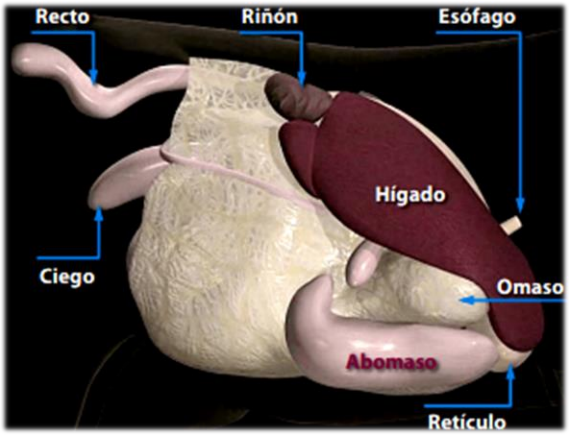

Figura 2. Saco de los rumiantes. [10].

3.2 La rumia y el pastoreo

El pastoreo es una técnica ampliamente utilizada para alimentar a los bovinos. Durante este proceso, los bovinos seleccionan los alimentos que les resultan más atractivos, considerando factores como su palatabilidad, facilidad de digestión y valor nutricional.

Por otra parte, la rumia es un proceso complejo y esencial para la digestión de los alimentos en los bovinos. Está, involucra la regurgitación y masticación de nuevo del alimento previamente ingerido, para volver a descomponerlo en partículas más pequeñas antes de volver a tragarlo y continuar la digestión. [8]

Con esto se llega a que la rumia y el pastoreo, tienen una influencia directa sobre el estado nutricional de los bovinos. La cantidad y calidad de los alimentos ingeridos, así como la eficiencia de la rumia, son factores clave que ayudan a determinar el estado nutricional de los animales.

Algunos de los parámetros de medición relacionados a la rumia, como por ejemplo: frecuencia de rumia, tiempo del animal pastando, comportamiento durante el pastoreo, entre otros, pueden proporcionar información valiosa para

la detección de ciertos factores que influyen de manera directa en la salud del animal.

Los patrones que podrían estar interfiriendo en la salud de estos, podrían deberse a estados de estrés, ansiedad o diferentes enfermedades que terminan afectando al rendimiento del animal [11] [12].

Así como también, es posible, lograr predecir con antelación otros sucesos como resulta por ejemplo, el estado de celo del animal [13].

3.3 Los cuatro eventos masticatorios producidos durante la rumia y el pastoreo

Los eventos considerados en las actividades alimentarias producidas por los bovinos son cuatro. Las actividades alimentarias de las que se trata son, la rumia y el pastoreo, actividades que se encuentran dentro de las más comunes en la vida del animal. Durante estas actividades se producen distintos tipos de eventos masticatorios, los cuales se mencionan a continuación, junto a sus correspondientes acrónimos, a los cuales se hace referencia posteriormente a lo largo del desarrollo:

0. RC = Rumination-chew (masticaciones producidas durante la rumia)
1. GC = Grazing-chew (masticaciones producidas durante el pastoreo)
2. B = Bite (arranque)
3. CB = Chew-bite (masticaciones seguidas de una mordida).

Se ha determinado que en el periodo de la actividad ruminal, es posible detectar los eventos asociados a la misma (RC), debido a que se producen una serie de masticaciones en intervalos que van de los 40 a 60 segundos; acompañados con pausas que circundan los 3 a 7 segundos de deglución / regurgitación del alimento ingerido previamente.

Para el caso del pastoreo, las secuencias de eventos que trascienden durante esta actividad (GC, B y CB), no presentan una estructura o patrón determinado, como si sucede es en el caso de la rumia.

Se reproduce a continuación, la representación de cada una de las señales asociadas a cada evento masticatorio, detectadas al realizar el análisis de las señales tomadas de las grabaciones realizadas.

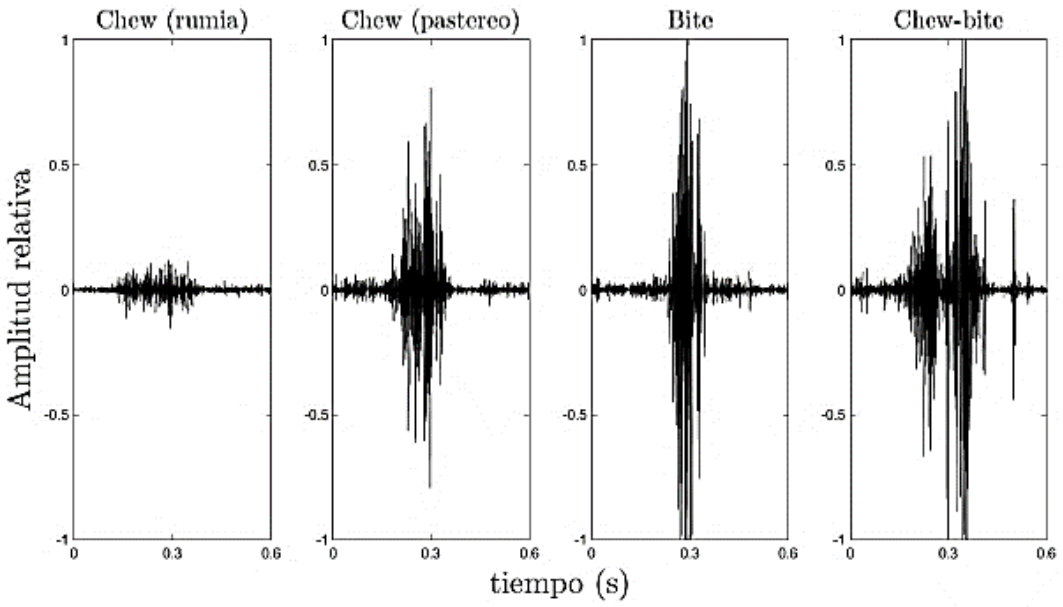

Figura 3. Señales acústicas de los eventos asociados a los movimientos mandibulares [14].

Como puede observarse, se tiene que el evento de masticación en rumia (rumination chew), presenta una señal cuyos valores de amplitud son bastante reducidos frente al resto de los eventos.

Tomando en cuenta el siguiente evento, masticación en pastoreo (grazing chew), se tiene que a pesar de presentar una mayor amplitud que el anterior, en ambos casos la duración temporal es muy similar.

En el caso del evento consecuencia del arranque (bite), se tiene una amplitud algo mayor que los dos eventos ya mencionados, pero se da en un tiempo menor, siendo prácticamente la mitad que en los otros dos casos ya vistos.

Y respecto del último evento, masticaciones seguidas de una mordida (chew bite), se tiene una combinación de los casos anteriores, con una duración temporal similar o hasta algo mayor al que se tiene en la masticación en rumia (rumination chew).

Capítulo 4: Descripción general del sistema

El sistema autónomo de monitorización acústica para ganadería, se desarrolla basado en la placa conocida comercialmente como BluePill, o bien, por el nombre de su microcontrolador el cual es STM32F103C8T6. El mismo, es el corazón de este dispositivo embebido, y cuenta también con una serie de sensores y módulos que acompañan a la conformación del sistema.

Se presenta en la figura 4, el diagrama en bloques del sistema indicándose en el mismo el conexionado entre las distintas placas empleadas. Las líneas de color representan la parte de alimentación, siendo las líneas de color rojo las de tensión positiva, las líneas de color azul el neutro o GND; y las líneas negras, representan el conexionado de datos.

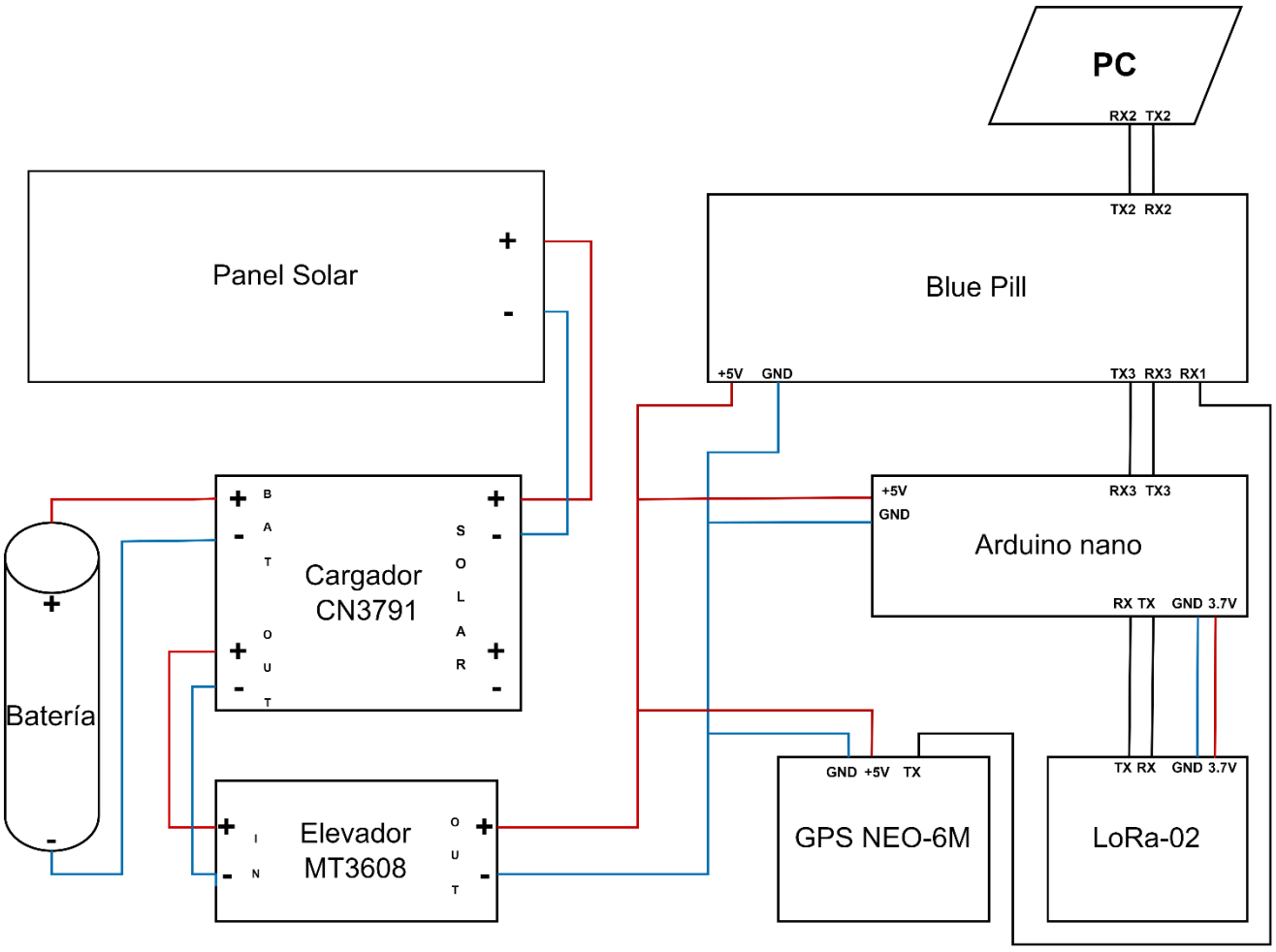

Figura 4.Diagrama en bloques.

A nivel de software, se destaca el algoritmo de inteligencia artificial con el que se procesan los datos recabados. El mismo fue desarrollado por integrantes del grupo de investigación *sinc(i)* del CONICET, y se basa en reconocer que tipo de evento es el que se produce en determinados momentos durante el ciclo de alimentación del animal bovino.

El presente sistema, consta de una serie de etapas, las cuales pueden observarse en el siguiente diagrama de procesamiento:

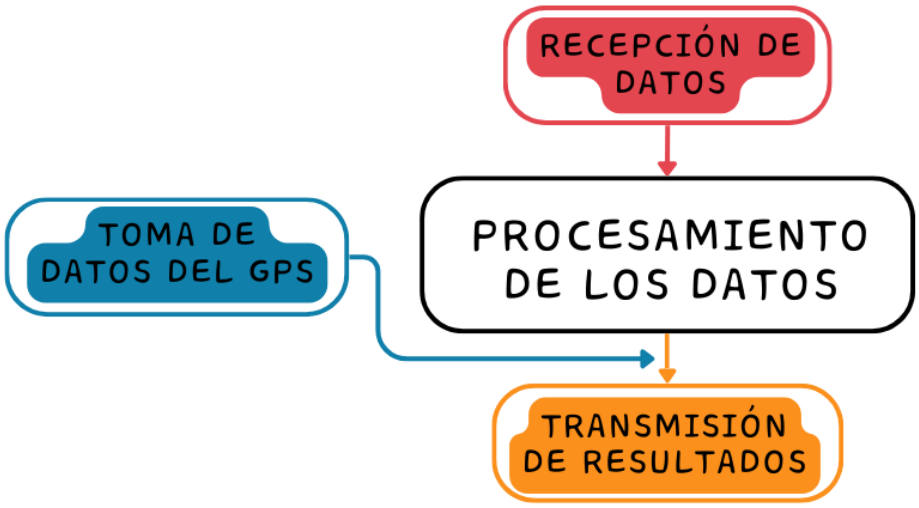

Figura 5. Diagrama general de procesamiento.

Se explicará a continuación, como se compone cada etapa, componentes y las herramientas utilizadas, incluyendo lo que respecta a hardware y software.

4.1 Recepción de datos

La recepción de datos por parte del microcontrolador se realizó de manera serial. Es decir, mediante el uso de una de las interfaces UART disponibles la cual es conectada directamente a la PC que genera los datos, mediante un conversor.

En un caso real de uso del dispositivo, donde la señal acústica es captada a través de un micrófono en tiempo real, se necesita agregar unas etapas de procesamiento analógico de la señal. En donde se debe realizar un primer filtrado, y el pre-procesado de la señal en cuanto a la ganancia. Luego de esto, la misma es convertida de señal analógica a señal digital para ingresar a la etapa de procesamiento digital.

Debido a la complejidad que conlleva el realizar toda la etapa de preprocesamiento analógico de la señal como se mencionó anteriormente, para este proyecto la misma se reduce al uso de archivos binarios que contienen la información correspondiente a señales acústicas ya preprocesadas.

Para esto, se utilizó una base de datos grabada en *Kellogg Biological Station* (Universidad del estado de Michigan, Estados Unidos) en la fecha de agosto de 2014. Dichos datos, representan el registro diario y continuo del comportamiento alimentario de 5 vacas lecheras, durante 6 días no consecutivos. Consumiendo mezclas de trébol blanco y raigrás, y trébol blanco y pasto ovillo.

Las grabaciones fueron llevadas a cabo mediante micrófonos electret convencionales, y grabadores comerciales (SONY ICDPX312).

Con esta base de datos se realizó la composición de un dataset con distintos segmentos correspondientes a la alimentación de los animales en pastoreo libre.

4.2 Procesamiento de los datos

El procesamiento de la información, se lleva a cabo en el microcontrolador STM32F103C8T6 con núcleo RISC ARM Cortex-M3 de 32 bits, el cual se encuentra integrado en la placa de desarrollo utilizada. Esta, posee una serie de prestaciones que la hace adecuada para llevar adelante el desafío que conlleva el procesamiento de la toda la información [15].

En cuanto a las características del microcontrolador, se pueden distinguir la siguientes:

- Velocidad de reloj: 72 MHz.
- Memoria: 64 KB de memoria Flash y 20 KB de SRAM.
- Pines de entrada/salida (GPIO): 37 pines GPIO.
- Comunicación: UART/USART, CAN, I^2C, USB, SPI, LIN, IrDA.
- ADC: Posee dos convertidor analógico a digital (ADC) de 12 bits para la lectura de señales analógicas, de 10 canales.
- PWM: Permite la generación de señales PWM (modulación por ancho de pulso), posee 12 canales PWM.
- Voltaje Lógico: 3.3 V.
- Alimentación: a través de un conector USB o mediante una fuente externa de 5 V.
- Software de programación: STM32CubeIDE, STMCubeProgrammer, SW4STM32, Arduino IDE y Mbed.

En cuanto a la distribución de pines de la placa Blue Pill, se tiene una disposición como la que se presenta en la siguiente figura:

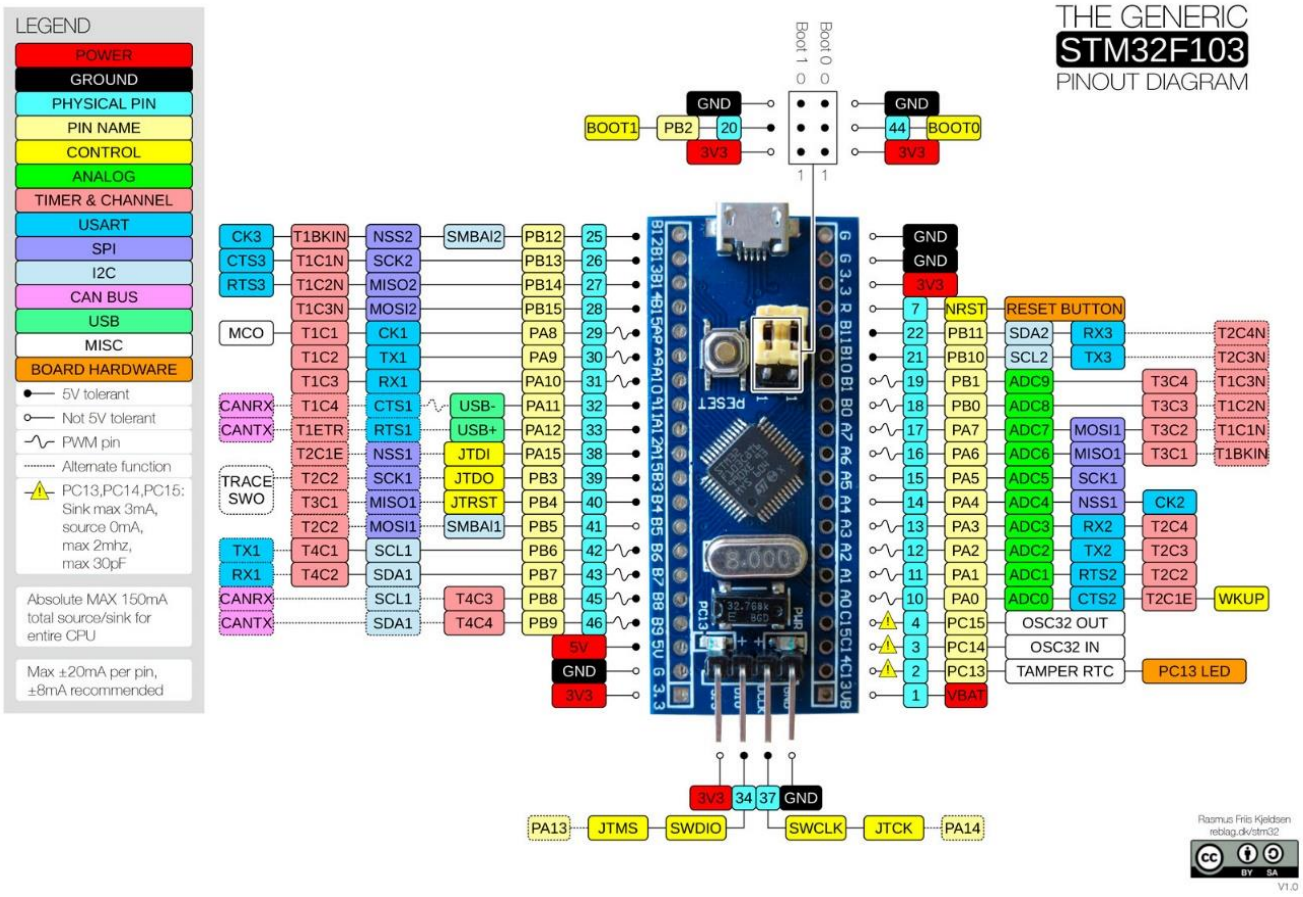

Figura 6. Pinout Blue Pill. [16]

Para su programación y depuración, se utiliza el dispositivo programador ST-LINK V2. Es la herramienta esencial, que permite la comunicación y el manejo de esta, entre la placa y la computadora.

Figura 7. Programador ST-Link V2.

4.2.1 Otros algoritmos desarrollados previamente

Uno de los algoritmos que se han propuesto con anterioridad ha sido el desarrollado por Milone en el año 2012, el cual se basa en el análisis de señales acústicas logrando una tasa de reconocimiento del 85% bajo condiciones con bajo nivel de señal de ruido. Este algoritmo, requiere de un gran costo computación, el cual no termina presentado un óptimo rendimiento [17].

Posteriormente, Chelotti en el año 2016, desarrolla un nuevo algoritmo. Este se basa en el análisis de las señales acústicas en el dominio temporal. Como resultado obtiene una tasa de reconocimiento muy similar al anteriormente mencionado, pero con costo computacional mucho menor [18].

Continuando con los trabajos desarrollados, Chelotti et al. (2018), logran incorporar una etapa de preprocesamiento que ayuda a remover tendencias y ruidos. Mejorando también el clasificador de eventos, mediante el uso de técnicas de aprendizaje maquinal. Consiguen así, mejorar las tasas de detección en entornos controlados de bajo ruido, pero continuando la degradación en entornos de cielo abierto. Denominando al mismo, como CBIA (Chew Bite Intelligent Algorithm) [19].

Por lo que como mejora y solución a los inconvenientes aún persistentes en los algoritmos mencionados, es que surge el desarrollo del algoritmo actualmente utilizado en el presente proyecto, el algoritmo CBEBA. Con el que se logra reducir las detecciones de falsos positivos, logrando una superioridad de desempeño en ambientes ruidosos frente a CBIA.

4.2.2 Algoritmo CBEBA (Chew Bite Energy Based Algorithm) [20]

El algoritmo utilizado en el presente proyecto, fue desarrollado como respuesta frente a los inconvenientes que se presentaban en la detección y clasificación de los eventos masticatorios frente a determinados niveles de ruido.

El algoritmo CBEBA, denominado así por su acrónimo (Chew Bite Energy Based Algorithm), hace uso de la envolvente de potencia instantánea de la señal de entrada, y también incorpora características que mejoran la inmunidad en condiciones adversas.

A diferencia de otros algoritmos desarrollados, en este, se desarrolla la posibilidad de detectar cuatro eventos en vez de solo tres, como ocurría en el caso del algoritmo CBIA [19]. Siendo los eventos, los mencionados en la sección 3.3.

El algoritmo se puede ser desglosado en 6 etapas secuenciales contando con realimentación interna.

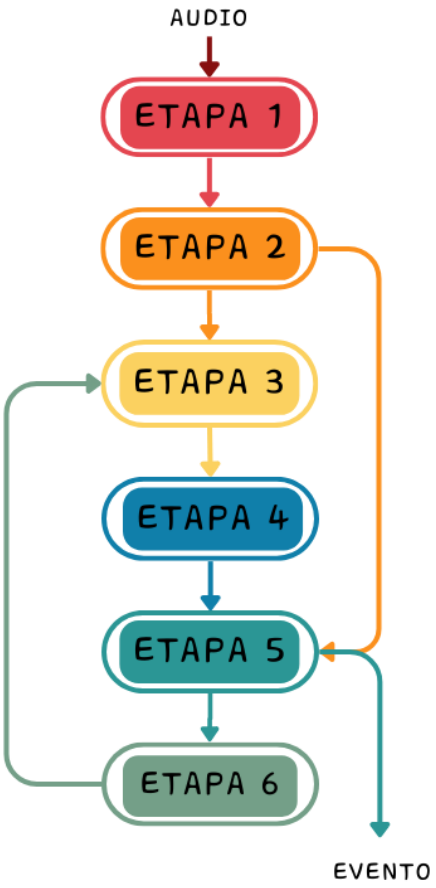

Figura 8. Diagrama en bloques del algoritmo CBEBA.

Una vez que ingresa la muestra de audio, en la primer etapa se realiza un preprocesamiento de la señal. Luego en la segunda etapa, se procede al realizar los cálculos correspondientes a la envolvente, y a la energía por

frames. Estos cálculos son almacenados, y se avanza hacia la detección de evento en la etapa tres.

En una cuarta etapa, se realiza la extracción de una serie de características las cuales son necesarias para secciones posteriores.

Seguidamente, se llega a la etapa cinco, está es la etapa donde se clasifican los eventos utilizando un árbol de decisión. A partir de esta etapa ya se tiene el resultado de cuál fue el evento reconocido.

Y, en una última etapa, etapa seis, se realizan los ajustes de los parámetros necesarios. Al transcurrir esto, se realimenta la etapa de detección de evento para un siguiente procesamiento.

4.3 Toma de datos del GPS

Con relación a la obtención de los datos referentes a fecha, hora y ubicación, se hizo uso de un módulo NEO-6MV2 [21]. El mismo posee las siguientes características:

- Alimentación: 2.7 V – 3.6 V (NEO-6Q/6M)
- Consumo de energía: 111 mW @ 3.0V (continuos)
  33 mW @ 3.0V (ahorro de energía) (1 Hz)
- Protocolos: NMEA, UBX binary, RTCM
- Temperatura de operación: –40° C a 85° C
- Adquisición de datos:
  - Arranque en frio: 27 s
  - Arranques asistidos [1]: < 3 s
  - Arranque en caliente: 1 s
- Velocidad de actualización de la navegación: hasta 5 Hz.

[1] Depende de la velocidad y latencia de la conexión de datos de ayuda.

El mismo tiene la capacidad de funcionar con diferentes protocolos, y en este trabajo, el utilizado es el NMEA. Dicho protocolo posee la capacidad de entregar diferentes tipos de mensajes. En este caso, el tipo de mensaje seleccionado para trabajar fue el GGA (Global positioning system fix data).

La estructura del mensaje GGA se compone de 17 campos y se compone de la siguiente manera:

```
$GPGGA,hhmmss.ss,Latitude,N,Longitude,E,FS,NoSV,HDOP,msl,m,Altref,m,
DiffAge,DiffStation*cs<CR><LF>
```

Se puede contrastar con el siguiente ejemplo, mediante el cual se expresa a continuación, y también se indica que representa cada campo en la tabla 1.

```
$GPGGA,092725.00,4717.11399,N,00833.91590,E,1,8,1.01,499.6,M,48.0,M,,0*5B
```

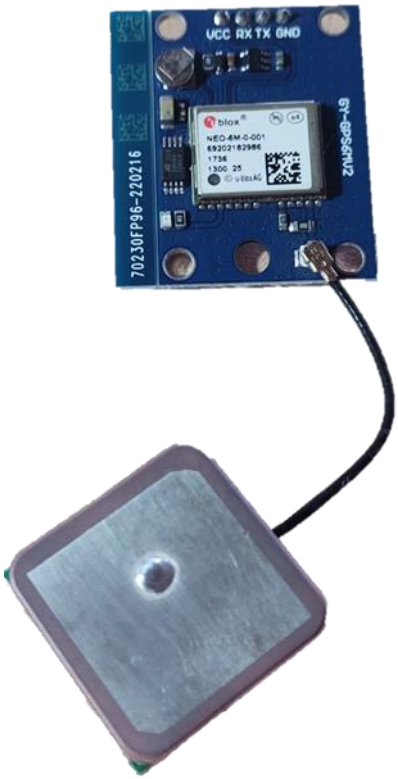

Figura 9. Módulo GPS NEO-6M.

El fin por el que se incorpora dicho modulo, es para poder obtener la ubicación del animal a campo abierto. En este trabajo, dicha etapa resulta solo un complemento, debido a que no se tendrá gran variación de la ubicación por el hecho de realizarse solo para pruebas de laboratorio.

Se debe tener en cuenta que según la estructura del lugar donde nos encontremos testeando el sistema, puede que no logre conectarse a los satélites, y por ende no lograr obtener así la información de nuestro interés. Esto surge, debido a las estructuras de como por ejemplo, paredes muy gruesas o bien obstrucciones en sí que interfieran en la comunicación.

Lo ideal es encontrarse cerca de una ventana, o bien, idealmente al aire libre.

Campo	*Ejemplo*	*Formato*	*Nombre*	*Unidad*	*Descripción*
0	$GPGGA	string	$GPGGA	-	ID del mensaje, cabecera del protocolo GGA
1	092725.00	hhmmss.ss	hhmmss.ss	-	Tiempo UTC, tiempo actual
2	4717.11399	ddmm.mmmm	Latitud	-	Latitud, grados + minutos
3	N	carácter	N	-	Indicador N=Norte / S=Sur
4	0083.91590	dddmm.mmmm	Longitud	-	Longitud, grados + minutos
5	E	carácter	E	-	Indicador E=Este / O=Oeste
6	1	digito	FS	-	Indicador de estado de fijación de posición
7	8	numero	NoSV	-	satélites usados, rango de 0 a 12
8	1.01	numero	HDOP	-	Dilución horizontal de la precisión
9	499.6	numero	msl	m	Altitud sobre el nivel del mar
10	M	carácter	uMsl	-	Unidades, metros
11	48.0	numero	Altref	m	Separación geoide
12	M	carácter	uSep	-	Unidades, metros
13	-	numero	Diffage	s	Edad de las correcciones diferenciales, campos en blanco (nulos) cuando no se utiliza DGPS
14	0	numero	DiffStation	-	Diferenciación de ID de estación de referencia
15	*5B	hexadecimal	Cs	-	Checksum
16	-	carácter	<CR><LF>	-	Retorno y salto de línea

Tabla 1. Campos en la estructura del mensaje GGA del protocolo NMEA [22].

4.4 Transmisión de los datos

En cuanto a la transmisión, el proceso se inicia una vez que el flujo de datos ha avanzado a través del programa. Esto implica la entrada de datos, la aplicación del algoritmo a estos datos y la obtención de información del GPS. Luego, se procede a la construcción de la trama final que se enviará posteriormente.

Dicho proceso de construcción de la trama será un estado que el programa experimentará, y se abordará con mayor detalle en la sección 5.4.1.

En lo que respecta a la tecnología de comunicación, se empleó un enlace LoRa, que hace uso de una técnica de modulación de espectro ensanchado derivada de la tecnología CSS[2]. Con esta tecnología, se codifica la información

[2] CSS espectro ensanchado chirp (Chirp Spread Spectrum).

en ondas de radio utilizando pulsos Chirp. Dicha transmisión resulta resistente a las perturbaciones y se puede transmitir a grandes distancias, superando ampliamente a otras tecnologías como lo son como WiFi, Bluetooth o ZigBee [23].

La modulación CSS es una técnica, que utiliza pulsos chirp modulados en frecuencia lineal de banda ancha para codificar la información. En este tipo de técnicas de modulación de espectro ensanchado, la señal se transmite en ráfagas, saltando entre las frecuencias de una secuencia pseudoaleatoria.

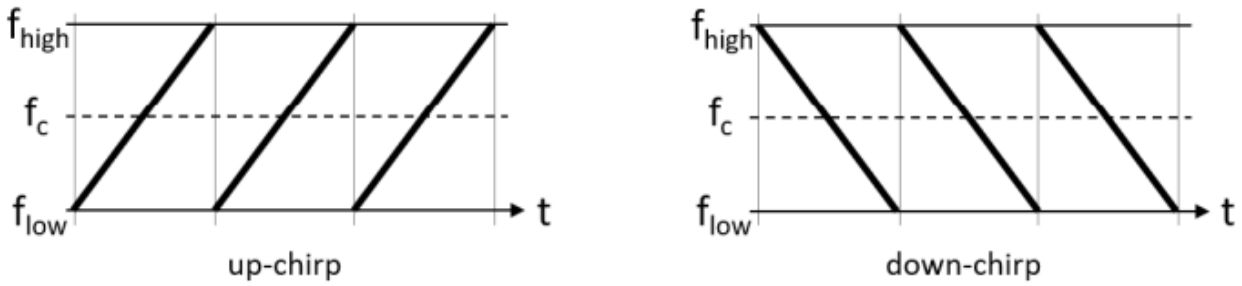

Figura 10. Muestra de pulsos up-chirp y down-chirp [24].

Un chirp es un tono cuya frecuencia aumenta con el tiempo (up-chirp) o disminuye (down-chirp). Su ancho de banda coincide con el ancho de banda espectral de la señal [19].

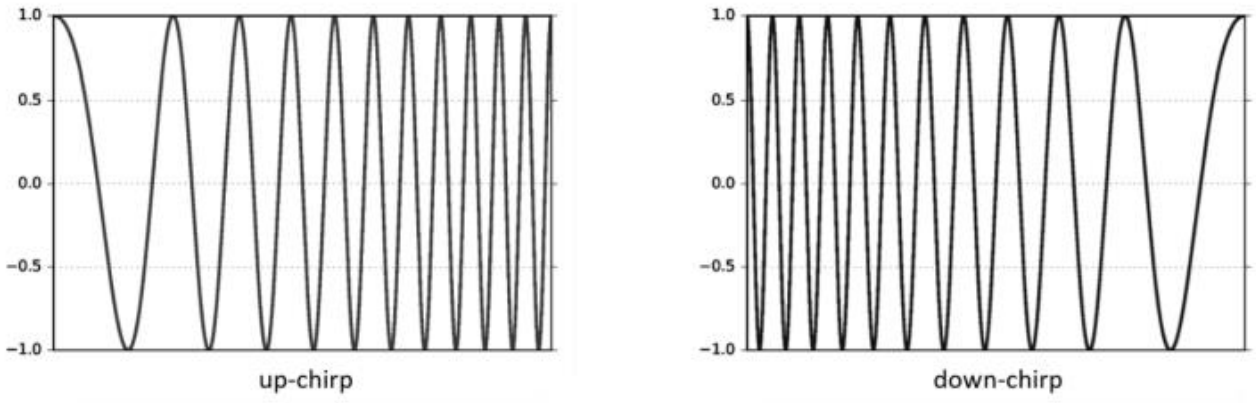

Figura 11. Señal asociadas a los pulsos up-chirp y down-chirp [24].

La misma, es capaz de trabajar en distintas bandas de frecuencias, como por ejemplo 868 MHz en Europa, 915 MHz en América del norte, y 433 MHz en Asia.

Estas frecuencias pertenecen a las bandas ISM (Industrial, Scientific and Medical), y están reservadas internacionalmente para el uso no comercial de radiofrecuencia electromagnética en áreas industriales, científicas y médicas.

Es fundamental tener en cuenta la frecuencia de trabajo, ya que cada país tiene sus regulaciones que determinan las bandas de frecuencia permitidas. Trabajar en una banda no autorizada podría tener implicaciones.

Para el caso de Argentina, la ENACOM establece mediante la resolución Nº 581/18 [25], que las bandas de frecuencias radioeléctricas, que se declaran de uso compartido y que no requieren autorización, son las siguientes:

- 915 – 928 MHz
- 2400 – 2483.5 MHz
- 5150 – 5250 MHz
- 5250 – 5350 MHz
- 5470 – 5600 MHz
- 5650 – 5725 MHz
- 5725 – 5850 MHz
- 57000 – 71000 MHz

En cuanto al hardware, se hizo uso de dos módulos LoRa-02, cuyas antenas trabajan en la frecuencia de 433 MHz. Es importante destacar que esta frecuencia no está declarada para uso sin autorización en nuestro país, pero dado que su uso fue estrictamente privado y limitado al entorno de laboratorio, no se presentaron inconvenientes.

Uno de los módulos fue usado del lado del dispositivo en cuestión, formando parte de la etapa transmisora. Mientras que el otro modulo, se ubicó del lado de la estación base, a donde son enviadas las tramas de datos resultantes.

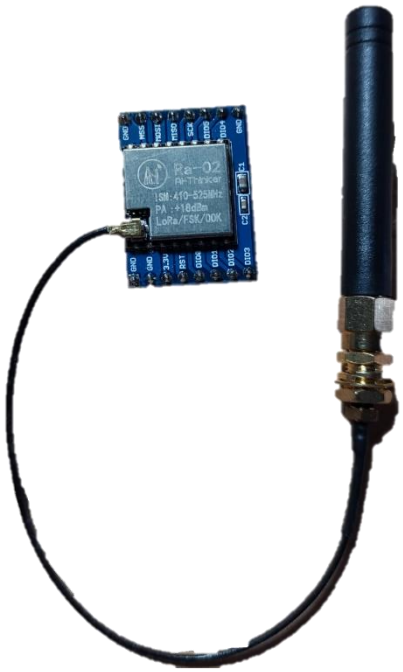

Figura 12. Modulo LoRa-02.

En la etapa transmisora, se añadió una placa Arduino Nano como intermediaria entre la placa Blue Pill y el módulo LoRa-02. Esta solución se implementó para abordar las incompatibilidades que surgieron en los protocolos de comunicación al intentar conectar la placa Blue Pill directamente al módulo LoRa-02. Como resultado, los datos se envían de manera serial desde el microcontrolador STM32F103C8T6 del sistema hacia el Arduino Nano. El Arduino Nano se encarga de transmitir los datos en colaboración con el módulo, permitiendo una comunicación fluida y efectiva.

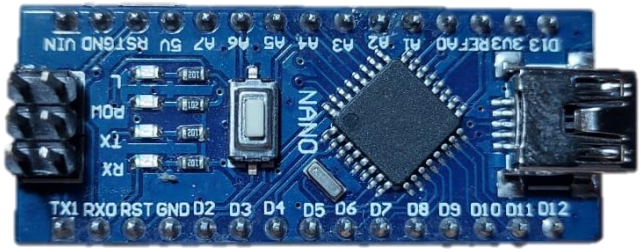

Figura 13. Arduino nano.

En la sección 5.4, se contempla la composición y el armado de la trama a enviar mostrando también ejemplos de pruebas realizadas.

4.5 Recepción de los datos en estación base

En lo que respecta a la recepción de los datos transmitidos desde el dispositivo, la misma se realiza mediante un Arduino UNO el cual funciona en conjunto con un módulo LoRa-02, conformando lo que se menciona como estación base.

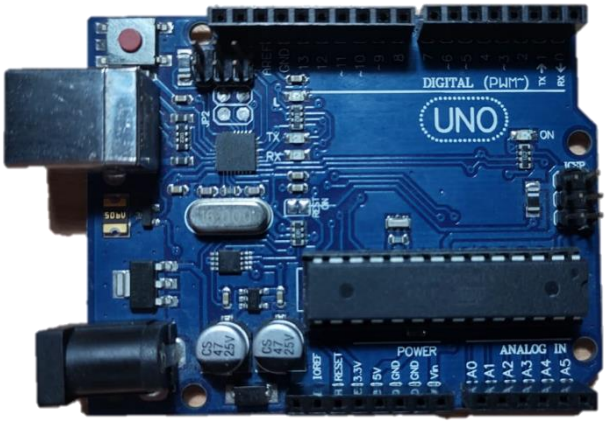

Figura 14. Arduino UNO.

La programación del script para el control correspondiente de los datos en el microcontrolador, se realizó utilizando la plataforma Arduino IDE con ayuda de las correspondientes librerías.

Por otra parte, se realizó la programación de una interfaz web, con el fin de visualizar los datos de manera más atractiva a la visual. Dicho desarrollo se estará mencionando en el apartado 5.5.3 siguiendo el recorrido del flujo de los datos desde que se reciben en el módulo hasta su representación en la interfaz web.

4.6 Alimentación del sistema

El sistema se alimenta mediante una batería de ion-litio de 3.7 V 2200 mAh, entregando una alimentación suficiente para mantener en funcionamiento al equipo por aproximadamente 6 días consecutivos. Y a su vez, la batería es recargada mediante la energía obtenida por medio de un panel solar que posee el sistema en la estructura superior del gabinete, lo cual hace que el periodo de funcionamiento se prolongue.

4.6.1 Panel solar

En el sistema de alimentación, se encuentra compuesto por un panel solar que es el encargado de brindar la energía suficiente a la batería para su correspondiente recarga siempre que sea necesario. El mismo, es un panel de 1 W, con capacidad de entregar hasta 200 mA.

Figura 15. Panel solar.

El panel es conectado a un módulo cargador de baterías, el cual es el módulo el CN3791.

4.6.2 Modulo CN3791

El CN3791 es un versátil controlador de cargas, diseñado específicamente para baterías de iones de litio (Li-Ion). Funciona como un controlador de corriente constante y tensión constante, lo que permite cargar eficientemente las baterías Li-Ion a través de una fuente de energía, como una célula fotovoltaica.

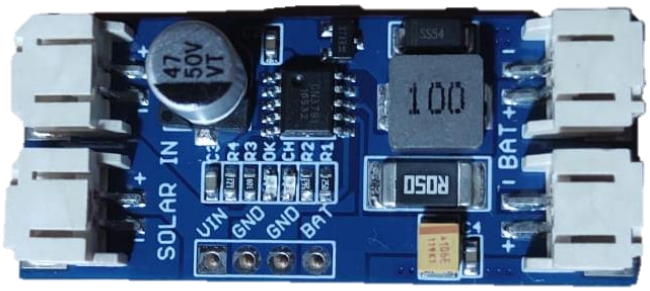

Figura 16. CN3791.

Una de las particularidades a destacar, es su capacidad de funcionar con la función de seguimiento del punto de máxima potencia (MPPT, por sus siglas en inglés). Esto significa que puede optimizar la eficiencia de carga al rastrear y utilizar la potencia máxima disponible proveniente de la célula fotovoltaica. Esta función permite aprovechar al máximo la energía solar y garantiza una carga eficiente de la batería.

Por otra parte, en cuanto a la arquitectura de conmutación utilizada, el CN3791 implementa un diseño step-down (buck) basado en la técnica de modulación por ancho de pulso (PWM).

El controlador, incorpora características focalizadas en la seguridad de la vida útil del mismo, como son: bloqueo por bajo voltaje, protección contra sobretensión de la batería e indicación de estado [26].

A este controlador, se conecta la batería, y en otro de los puertos, se encuentra el conector de salida que está conectado al módulo MT3608. Este módulo es

una fuente step-up con la capacidad de aumentar la tensión de entrada hasta 28 V.

4.6.3 Fuente MT3608

Esta etapa se torna necesaria, debido a que si bien la placa Blue Pill puede funcionar a 3.3 V, es recomendable alimentarla con 5 V, para evitar ciertos comportamientos erráticos, que podrían presentar los registros y componentes internamente.

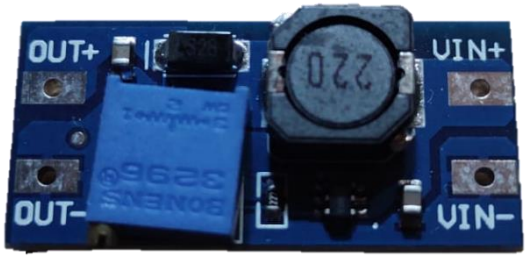

Figura 17. Fuente Step-up MT3608.

El módulo GPS también requiere de 5 V de alimentación para su correcto funcionamiento por más que en su hoja de datos se indique que es capaz de funcionar con 3,7 V, y el módulo Lora, se alimenta con 3,3 V desde la placa Arduino nano.

El MT3608 es un convertidor elevador que utiliza una arquitectura de regulador boost de frecuencia fija, y modo de corriente de pico, para regular la tensión de salida. Está ideado para satisfacer las necesidades de aplicaciones que se caracterizan por ser a pequeña escala, y tener requisitos de consumo energético reducidos. Una de las características destacadas del MT3608 es su sistema de arranque suave interno, el cual despliega una estrategia de control sofisticada durante el inicio del dispositivo. Este mecanismo ha sido incorporado con el propósito específico de limitar tanto la cantidad de corriente de entrada durante el arranque como la posibilidad de sobre impulso en la

salida. El MT3608, está equipado con una serie de sistemas de protección avanzadas, que garantizan un funcionamiento seguro y confiable, incluso en condiciones de sobrecarga. Estas características incluyen el bloqueo por subtensión, la limitación de corriente y la protección contra sobrecarga térmica [27].

4.6.4 Batería

La batería utilizada es la LGDAS31865, cuyas especificaciones son de 3.7 V de tensión nominal y 2200 mAh de capacidad nominal.

En cuanto al proceso de carga de la batería, se puede hacer hincapié en la siguiente figura que nos arroja el datasheet del módulo CN3791, que es el mismo al cual se encuentra conectada.

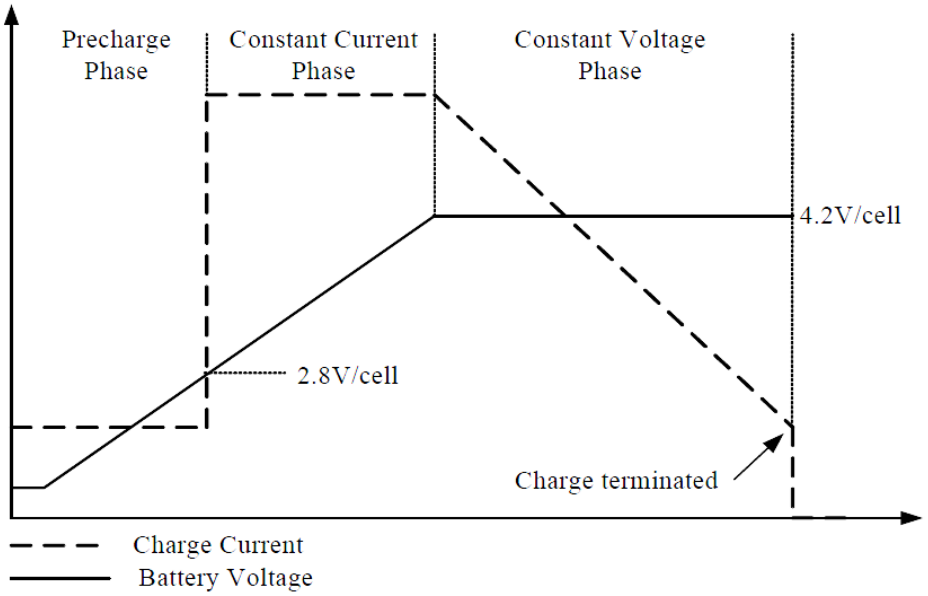

Figura 18. Proceso teórico de carga de batería [26].

Se debe buscar que la batería no baje de los 2.8 V, ya que se hace muy complicado el proceso de llegar a recuperarlas. Ahora bien, estando en una tensión a partir de la indicada anteriormente, el proceso de carga se encuentra

en una fase de corriente constante. Superada dicha fase, pasa a la fase de voltaje constante donde va disminuyendo el valor de corriente hasta llegar a completar el ciclo de carga de la batería.

Evaluando el proceso de carga de la batería utilizada en el proyecto, se tomaron una serie de datos consecutivos en el tiempo, que permitieron plasmar de manera grafica el proceso de carga mediante la realización de una curva de Tensión – Tiempo.

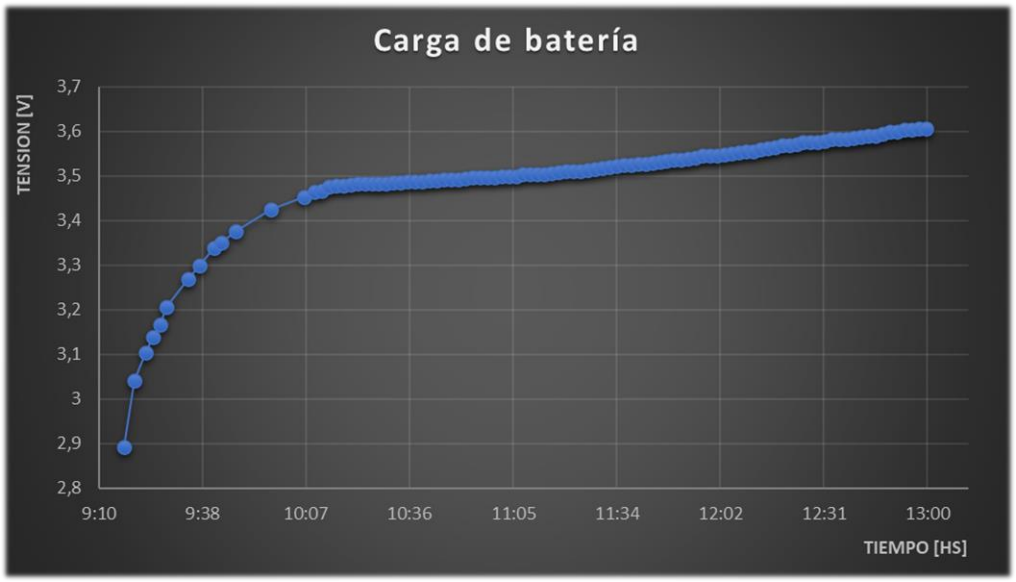

Figura 19. Proceso de carga de batería real.

Con lo que se puede confirmar que la primer fase es un proceso de carga donde la tensión se eleva rápidamente debido a la inyección de corriente constante, y luego va aumentando de manera gradualmente lenta hasta llegar al valor final, al mismo momento que la corriente va decreciendo gradualmente.

4.6.5 Cálculos de autonomía del sistema

Tomando en cuenta las capacidades de la batería utilizada, y demás datos de consumo, se tiene que:

Tensión de batería ($\mathbf{V_b}$) = 3.7 V

Intensidad de batería ($\mathbf{I_b}$) = 2400 mA

Consumo de la BluePill, ($\mathbf{I_{bp}}$) = 14.93 mA

Consumo de modulo Lora en reposo ($\mathbf{I_{lsm}}$) = 0.2 uA

Consumo de módulo Lora en transmisión ($\mathbf{I_{ITx}}$) = 20 mA

Consumo de módulo NEO-6M en reposo ($\mathbf{I_{gpsRx}}$) = 22 uA

Consumo de módulo NEO-6M en adquisición de datos ($\mathbf{I_{gpsRx}}$) = 11 mA

Consumo de Arduino nano en funcionamiento ($\mathbf{I_{nano}}$) = 15 mA

Ahora bien, se realizará un simple cálculo de manera estimativa para tener referencia de cuánto tiempo podría funcionar el sistema sin recarga de batería.

$$Potencia\ de\ la\ batería\ =\ \boldsymbol{Vb}\ *\ \boldsymbol{Ib}\ =\ 3.7\,V\ *\ 2400\,mA\ =\ 8.88\,W$$

$$Potencia\ de\ consumo\ =\ \boldsymbol{Vb}\ *\ \boldsymbol{Ibp}\ =\ 3.7\,V\ *\ 14.93\,mA\ =\ 0.055\,W$$

$$Tiempo\ de\ duración\ =\ \frac{8.88\,W}{0.055\,W}\ =\ \mathbf{161.45\ h}$$

Tomando en cuenta que una hora son 60 minutos, realizando regla de 3 simple, se tiene que:

$$0.45\ h * \frac{60\ min}{1\ h}\ =\ 27\ min$$

Y considerando que un día tiene 24 horas:

$$\frac{161\ h}{24\ h} = 6.708\ días$$

Aplicando nuevamente regla de 3 simple:

$$0.708\ días * \frac{24\ h}{1\ días} = 17\ h$$

Sumando entonces, los valores obtenidos, tenemos que la autonomía podría llegar a ser de 6 días 17 horas y 27 minutos.

Se debe considerar que el cálculo fue realizado en base a un consumo constante e igual al de la placa Blue Pill, pasando por alto el de los demás componentes a modo de compensación de los consumos.

También se debe tener presente que en el caso de continuidad del proyecto, añadiéndose la etapa de toma de datos con micrófono, se tendría además un consumo extra de aproximadamente 0.8 mA por cada ADC que se utilice sumado al consumo de las etapas de acondicionamiento analógico.

4.7 Diseño de PCB

Debido a la cantidad de placas y módulos que conforman al sistema, se decidió realizar el diseño de un PCB en el que se puedan colocar modularmente todas las placas utilizadas. El mismo se realizó con el programa open source KiCad, ya que también nos brindaba la oportunidad posterior, de fusionar el diseño con el del gabinete 3D en Freecad.

El diseño creado, generando por software las vistas 3D, se puede observar en las siguientes figuras a continuación.

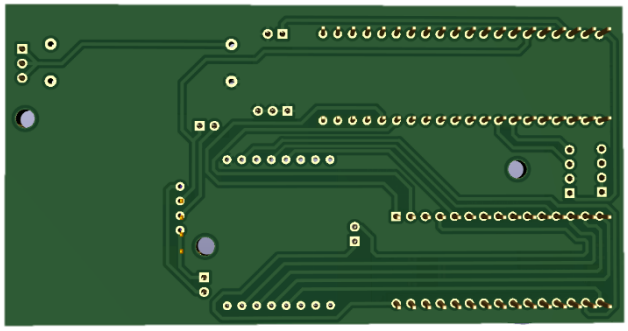

Figura 20. Vista inferior del modelo 3D de las pistas del PCB.

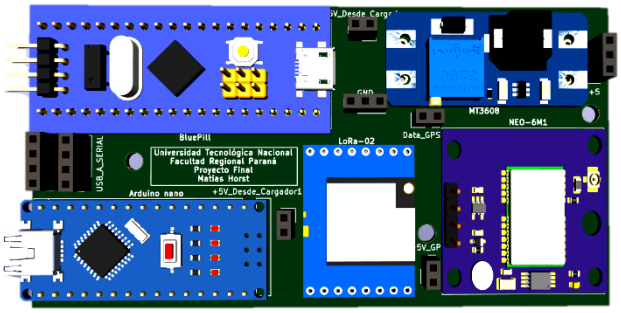

Figura 21. Vista superior del modelo 3D del PCB.

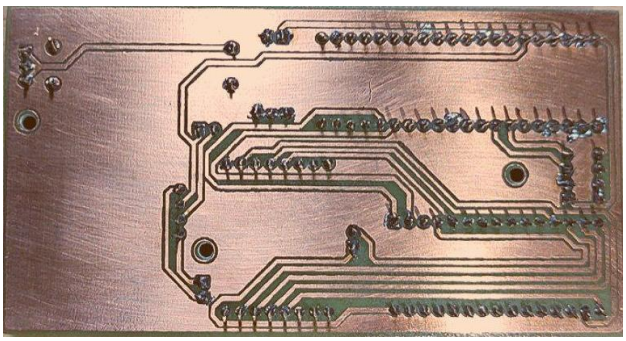

Figura 22. Vista inferior real del PCB.

La vista superior real se presentará en la siguiente sección en conjunto con el gabinete.

4.8 Gabinete 3D

Para el encapsulamiento de la placa, diseñamos un gabinete que cumple con los requisitos del prototipo. Este diseño se llevó a cabo utilizando el programa de código abierto FreeCAD.

El gabinete incluye un espacio en la parte superior de la tapa para el soporte del panel solar, una cavidad interna para alojar la batería, un espacio para sujetar la antena del módulo LoRa y dos cavidades en el lateral izquierdo que permiten la conectividad de las placas con la PC sin necesidad de retirarlas del gabinete para su interacción.

También posee los orificios para poder tener la posibilidad de atornillar la tapa como opción secundaria, ya que la misma posee de por si un calce interno; y también, agujeros para atornillar el PCB en la superficie inferior.

El diseño se muestra a continuación:

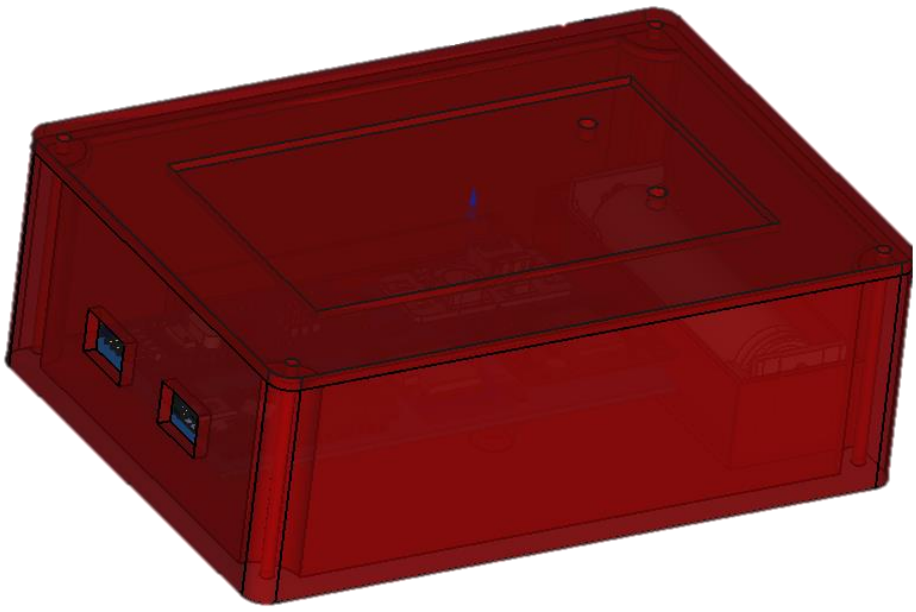

Figura 23. Vista completa del gabinete.

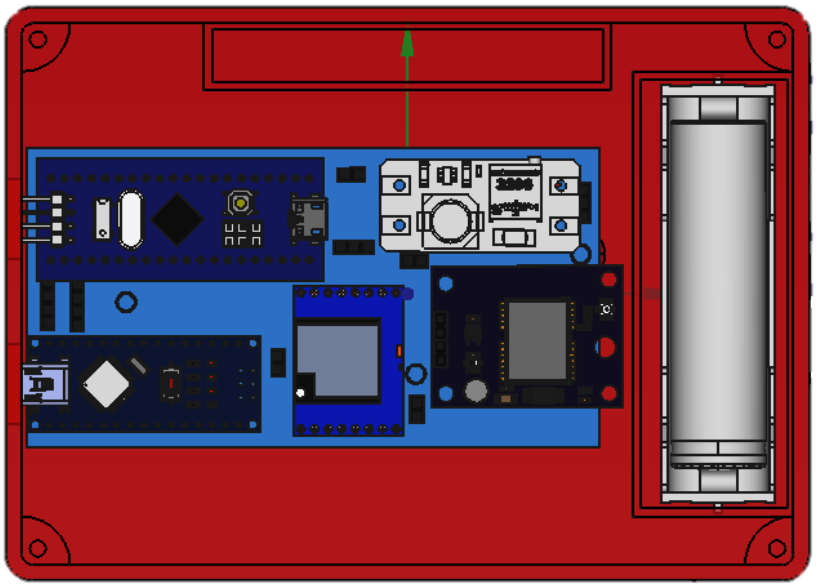

Figura 24. Vista superior del gabinete sin tapa.

Posteriormente, se le agrego detalles de grabados en la tapa con el programa fusión 360, antes de su correspondiente impresión.

Una vez realizada, la impresión quedo de la siguiente manera:

Figura 25. Vista superior del gabinete finalizado.

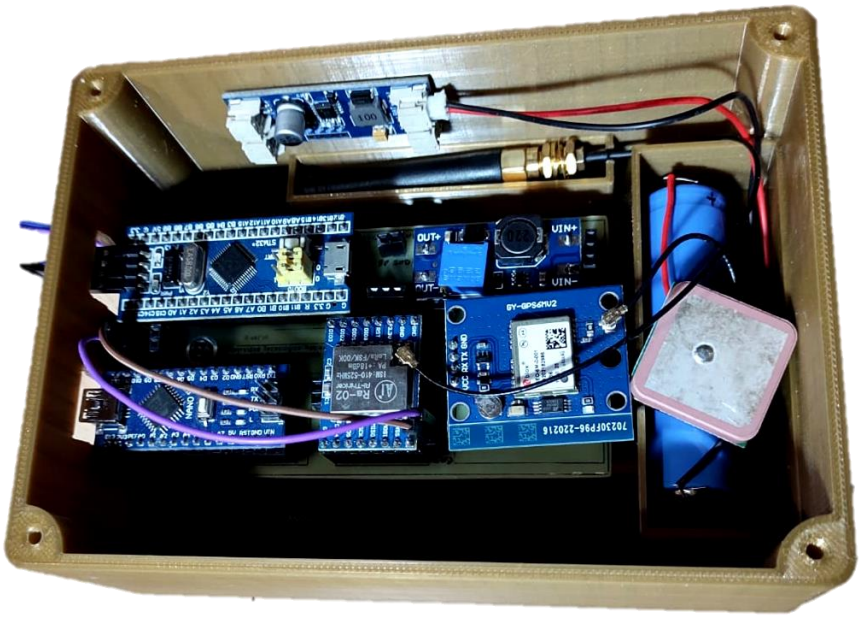

Figura 26. Vista interna del gabinete.

Por último, se reproduce una imagen que proporciona una idea de la colocación real del dispositivo en el animal. En la misma, se observa un dispositivo grabador de audio, colocado con un bozal sujetador. Una vez obtenido el dispositivo completo funcional, se colocaría de manera similar.

Figura 27. Dispositivo de grabación en la cabeza de un bovino [28].

Capítulo 5: Codificación

Para el presente capítulo, se comienza con la presentación del diagrama de la máquina de estados, la cual es la base de la estructura del programa principal del sistema. De esta manera se modela y controla, el comportamiento del sistema, de donde las acciones futuras depende del estado actual.

Figura 28.Diagrama de máquina de estados.

Como se puede ver, se presentan 4 estados por los que se ha de transcurrir durante el ciclo de funcionamiento, y sobre los cuales se iterara durante la ejecución del sistema. Donde el estado central es el estado 01, en el que el microcontrolador se mantiene dormido, conocido como modo sleep.

A continuación, se presenta el diagrama de flujo por el que transcurre el programa, pasando por los estados mostrados en el diagrama de la figura 28.

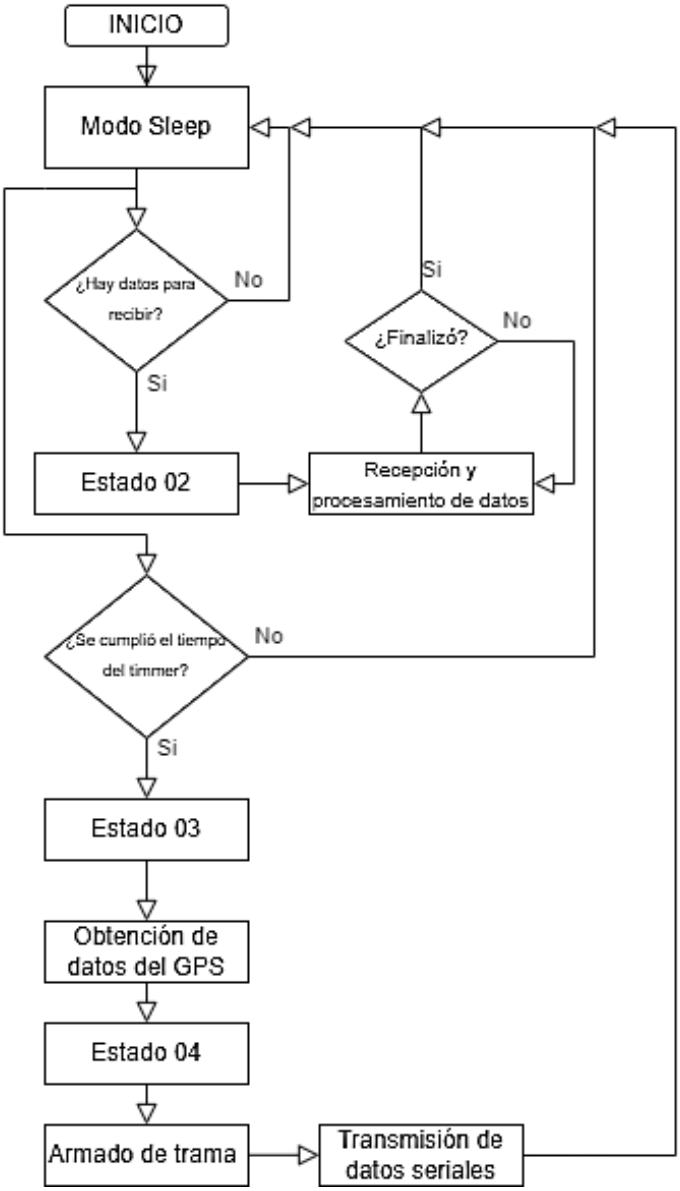

Figura 29. Diagrama de flujo.

5.1 Estado 01

Sleep mode

El microcontrolador STM32F103C8T6, cuenta con tres modos de ahorro de energía, y donde cada uno se diferencia del otro, según que componentes se deshabilitan, y como se puede restablecer a los mismo nuevamente.

El primer modo, y el que se usó en el presente proyecto fue el **modo Sleep**. Este modo desactiva el reloj de la CPU mientras que el resto de los periféricos continúan funcionando con normalidad.

El siguiente, es el **modo Stop**. En este, se apagan todos los relojes, lo que significa que los periféricos se encuentran apagados y no hay manera de interactuar con ellos desde el exterior. Lo que si se mantiene, son los valores de los registros.

El tercer modo, el de mayor ahorro energético, es el **modo Standby**. Se apaga casi en su totalidad, y cuando se restablece al modo run, es como si se hubiese reiniciado el microcontrolador, y ejecuta todo desde su inicio.

Se decidió utilizar el modo Sleep debido a que al mantenerse los periféricos encendidos, es posible despertar o restablecer al reloj de la CPU mediante interrupciones, ya sean internas o externas.

Trabajando a una frecuencia de CPU de 72 MHz, en las pruebas realizadas, se observó que el consumo promedio al ejecutar el código utilizando el modo Sleep es de aproximadamente 14.93 mA. Lo cual resulta un valor muy similar al valor tabulado en la tabla correspondiente en la hoja de datos del microcontrolador. La cual se reproduce a continuación, la parte correspondiente a lo tratado.

Figura 30. Test de consumo.

Simbolo	Parametro	Condiciones	f HCLK	Tipicos		Unidad
				Todos los perifericos habilitados	Todos los perifericos deshabilitados	
I DD	Corriente de alimentación en Sleep mode	Clock externo	72 MHz	14,4	5,5	mA
			48 MHz	9,9	3,9	
			36 MHz	7,6	3,1	
			24 MHz	5,3	2,3	
			16 MHz	3,8	1,8	
			8 MHz	2,1	1,2	
			4 MHz	1,6	1,1	
			2 MHz	1,3	1	
			1 MHz	1,11	0,98	
			500 kHz	1,04	0,96	
			125kHz	0,98	0,95	
1. Los valores son tipicos son medidos a TA = 25 °C, VDD = 3.3 V,						
2. Se añade un 0.8 mA de consumo por cada ADC utilizado.						
3. El clock externo esta en 8 MHz y PLL esta encendido cuando f HCLK > 8 MHz.						

Tabla 2. Típico consumo de corriente en sleep mode, corriendo código desde flash o RAM [15].

En lo que respecta a la codificación para el manejo del modo Sleep, al ingresar al modo, se programa un tiempo de seguridad lo suficientemente corto. Esto se hace en caso de que deba finalizar otro proceso que se esté ejecutando en paralelo. Es importante destacar que este tiempo de seguridad no es obligatorio y puede ser omitido si así se desea.

```
HAL_Delay(50);
HAL_SuspendTick();
HAL_PWR_EnterSLEEPMode(PWR_MAINREGULATOR_ON, PWR_SLEEPENTRY_WFI);
HAL_ResumeTick();
```

Figura 31. Codificación Sleep mode.

Luego se realiza la suspensión de la interrupción del sistema de temporización SysTick. Este es usado comúnmente como un contador de tiempo o un temporizador en las funciones en los microcontroladores STM32. Por lo que, suspendiéndolo, se evita que genere interrupciones mientras el microcontrolador está en modo sleep, y lo despierte en un momento inoportuno.

Continuamente, se llega a la línea que se encarga de poner a dormirlo. En el primer argumento, se le indica que el regulador de voltaje principal, debe permanecer encendido.

En el segundo, se le especifica que el microcontrolador debe entrar en modo sueño y debe esperar una interrupción (Wait For Interrupt – WFI) para despertar.

Por último, al momento que se produce una interrupción, la cual es dado por el suceso de la recepción de datos desde la UART2, se debe reanudar las interrupciones del temporizador SysTick del sistema.

5.2 Estado 02

5.2.1 Recepción de datos

Al detectar que hay información por recibir en el puerto UART2, se genera una interrupción en la que se realiza en primer lugar, un cambio de estado de la variable correspondiente al manejo de los eventos de la máquina de estados. Pasando al estado 02. A continuación, se configuran dos líneas, en donde, en la primera, se inicia la recepción de datos a través del puerto, utilizando el modo de transferencia de datos por DMA (Direct Memory Access). Se indica cuál de las estructuras UART se utilizará, se asigna el puntero al buffer donde se almacenarán los datos recibidos, y se indica el tamaño del buffer de recepción.

```
HAL_UARTEx_ReceiveToIdle_DMA(&huart2, (uint8_t *) RxBuf, RxBuf_SIZE);
__HAL_DMA_DISABLE_IT(&hdma_usart2_rx, DMA_IT_HT);
```

Figura 32. Configuración de interrupción UART2.

Luego se deshabilita una interrupción especifica de la configuración del canal de recepción DMA, ‘***DMA_IT_HT***’ la misma refiere a la interrupción de

Transferencia a la Mitad (Half Transfer). La cual al deshabilitarla, se evita que se genere una interrupción al haberse transferido la mitad de los datos de la UART al buffer.

Para el envío de los datos se emplea un script en Python diseñado específicamente para la transmisión de datos por la interfaz UART. El propósito de este script es leer el archivo de audio previamente grabado y preprocesado. Luego, envía los datos uno por uno a través del puerto al que está conectado el convertidor de USB a serial.

Tomando como ejemplo un archivo de audio correspondiente a 10 segundos de grabación, podemos ver mediante la consola, y en formato hexadecimal, los últimos 7 valores que fueron enviados.

Figura 33. Datos enviados desde PC.

Así, cuando el microcontrolador se encuentra en modo Sleep y detecta la llegada de datos a través de la UART 2, genera una interrupción que lo despierta. En ese momento, recibe los datos y los almacena en un buffer de recepción temporal.

A continuación, se muestra cómo se reciben los mismos datos que se mostraron en la figura 33, pero esta vez ya están almacenados en el buffer de recepción mencionado:

RxBuf[9986]	uint8_t	0x1
RxBuf[9987]	uint8_t	0x0
RxBuf[9988]	uint8_t	0xff
RxBuf[9989]	uint8_t	0xff
RxBuf[9990]	uint8_t	0x7
RxBuf[9991]	uint8_t	0x0
RxBuf[9992]	uint8_t	0xfd
RxBuf[9993]	uint8_t	0xff
RxBuf[9994]	uint8_t	0xf5
RxBuf[9995]	uint8_t	0xff
RxBuf[9996]	uint8_t	0xfc
RxBuf[9997]	uint8_t	0xff
RxBuf[9998]	uint8_t	0xf1
RxBuf[9999]	uint8_t	0xff

Figura 34. Datos serie recibidos en BluePill.

Como se puede apreciar en la figura anterior, los datos se reciben byte a byte. Esto es un detalle importante a tener en cuenta para su posterior manejo.

Con lo que, de manera resumida, se plasma que los datos no presentan problemas de corrupción en la transferencia entre la PC, que es la fuente de información, y el dispositivo receptor, que es el microcontrolador.

5.2.2 Procesamiento de los datos

Una vez que los datos se almacenan en el buffer de recepción, se lleva a cabo un proceso de acondicionamiento de los datos recibidos para que sean compatibles con el algoritmo. Dado que el archivo con los datos binarios fue generado utilizando codificación Little Endian y el algoritmo funciona con arquitectura Big Endian, se requiere una corrección en los datos a procesar. Este proceso se realiza dato por dato, ya que el tipo de variable que el algoritmo recibe es una word de 16 bits (2 bytes), mientras que cada variable del vector que almacena los datos recibidos consta de variables de 8 bits (1 byte).

Por ende, se toman las dos primeras variables consecutivas, correspondientes a un mismo dato del vector de recepción, se las invierte y se las junta

conformando una única variable de 16 bits. En ese momento, se encuentra en condiciones de ser accionado hacia el algoritmo.

```
Rx_word = (Rx_word & 0x00FF) | (recibido[j+1] << 8);
Rx_word = (Rx_word & 0xFF00) | recibido[j];
```

Figura 35. Acondicionamiento de variables.

Tomando en cuenta los datos mostrado a modo ejemplificativo, en las figuras 33 y 34, se puede observar cómo queda conformado el último dato, '***f1ff***', luego del acomodamiento:

(x)= Rx_word	word	0xfff1

Figura 36. Dato acondicionado.

Consecutivamente, se llama a la función que contiene la codificación del algoritmo CBEBA, encargado del procesamiento. Esta función toma como argumento el dato de 16 bits (word) acondicionado.

Una vez adentrado en el algoritmo, se tiene una variable denominada como *audio_sample* que es la que almacena el dato, pasado como puntero.

En otra variable, se escala a formato Q1.8. Esto es, un formato de punto fijo binario utilizado en procesamiento de señales, con 1 bit entero y 8 fracciones.

Se aplica una ganancia a la muestra de audio escalado, y esta se representa en formato Q3.2. Este formato, es un formato de punto fijo en el que se utilizan los 3 bits más significativos para representar la parte entera del número y los 2 bit menos significativos para la parte fraccional del número. Constituyendo así, un formato de 5 números.

Luego se almacena el resultado de multiplicar la ganancia por la muestra de audio escalado, formateándolo en Q4.10. Y, luego se realiza un escalamiento de dicho resultado para limitar el rango.

Posteriormente, se aplica un filtro pasa alto a este último valor obtenido, almacenándolo con un formato de, el bit más significativo para representar el

signo del número y los 15 bit restantes para representar la parte fraccional de mismo. Esto se usa para representar valores fraccionales con alta precisión.

Se calcula la energía de la señal al cuadrado, se continua con una serie de operaciones para acumular la energía, y luego calcular la envolvente de está.

De esa manera queda conformado la primer parte de la estructura del código, y luego en la segunda parte se hace el procesamiento partiendo de la señal envolvente ya calculada.

Se realiza una segmentación del valor, se almacena en un buffer, y se realiza la actualización de determinados parámetros utilizados por el algoritmo.

Se busca un nuevo evento partiendo de los valores recientemente actualizados, y se realiza el almacenamiento de varios resultados relacionados a la detección de eventos, extrayendo ciertas características.

Una vez pasada esa etapa, se comprueba que el evento detectado sea válido. Si es así, se realiza una actualización de parámetros internos y de umbrales, y se generan nuevos valores también.

Entonces llegado a este punto, se realiza la clasificación del evento como tal, utilizando el clasificador de eventos basado en las características extraídas anteriormente.

El resultado de la clasificación, se almacena en un buffer destinado al almacenamiento de eventos, que será utilizado posteriormente al realizar la transmisión de los eventos detectados.

Concluyendo de esta manera el procesamiento en la función del algoritmo CBEBA.

Como último paso en esta etapa, se realiza el cambio de la variable de estado, volviendo al estado 01. Es decir, el microcontrolador vuelve a estar en modo sleep, hasta que detecte nuevos datos y el proceso se repita.

5.3 Estado 03

Lectura con el módulo GPS NEO-6M

Para este apartado, se ayuda de una librería creada para el manejo del módulo NEO-6M. La misma se llama “NMEA” y contiene programados los parámetros necesarios para la obtención de hora, tiempo, y ubicación.

También se hace uso de una librería llamada “uartRingBuffer”, con la que se realiza el manejo de un buffer circular dedicado también a la obtención de los datos del GPS. Entonces, con este buffer se van actualizando los datos de manera continua.

Dentro de la codificación en el programa principal, se tiene un ciclo *for* utilizado para tomar los datos repetidamente un par de veces, y así tener mejor certeza sobre estos. Esto se realiza así, porque al inicio, los datos de hora demoran un tiempo mayor que los de ubicación, y los de fecha un tiempo aún mayor en ser adquiridos.

Se queda a espera de la cadena ‘*GGA*’, y al recibir los datos de la cadena ‘GGA’ verifica que sean validos o no.

De igual manera sucede para la cadena de tipo ‘RMC’, queda a espera de la misma, y al llegar, realiza una comprobación de validez de los datos.

Una vez superado estos pasos, al tener al menos una de las cadenas con datos válidos, se realiza la extracción de los datos de interés en diferentes variables.

Primero se obtienen los valores de hora, minutos, segundos y luego día, mes y año. Consecutivamente, latitud, dirección de latitud, longitud y dirección de longitud.

Se tiene también una variable, que es usada como contador para comprobación del estado de funcionamiento. Donde por ejemplo, si la tensión de alimentación del módulo es inferior a los 5 V o si existe una mala conectividad física, se reseteará la misma dando alerta de esto.

Al finalizar el proceso, la variable de estado, altera su valor, pasando al estado 04. Para así ingresar al estado de transmisión.

Físicamente se realiza el conexionado del cable correspondiente a la transmisión de datos del módulo NEO-M6, con el pin B7 del microcontrolador, correspondiente a la UART 1.

Una vez energizado el módulo, se debe esperar a que el mismo capte las señales de los satélites que se encuentren disponibles en el lugar que se está presente, a partir de ahí se obtiene la información de interés.

Al conectar el sistema, es posible observar los datos al trabajar en modo debug en el entorno STM32Cube IDE, viéndose a modo de ejemplo, de la siguiente manera:

resultado_gps	RESULTADO_GPS	{...}
hora	int	10
minutos	int	30
segundos	int	38
dia	int	9
mes	int	9
anio	int	23
latitud	float	31.4501877
NorteSur	char	83 'S'
longitud	float	60.3019028
EsteOeste	char	87 'W'

Figura 37. Estructura de datos obtenidos por el módulo NEO-M6.

Se estima que la precisión respecto de la posición es de 2.5 m aproximadamente, lo cual resultan valores más que aceptables para un sistema de posicionamiento como el utilizado.

5.4 Estado 04

5.4.1 Armado de trama

La trama de datos a transmitir desde el dispositivo, contiene los datos correspondientes a los eventos masticatorios clasificados durante los últimos 5 minutos.

Para esta periodicidad se hace uso de uno de los timmers que posee el microcontrolador, precisamente el timmer 2. Quedando configurado con los siguiente valores, preescaler: 35999 y periodo: 19999. De esta manera, trabajando con el Clock de la CPU a 72 MHz, resulta una interrupción en un tiempo de 10 segundos. Por lo que, para obtener los 5 minutos que se necesitan, se utiliza un contador que contabiliza hasta 30 interrupciones del timmer, para que luego de este tiempo se arme y transmita la trama.

La misma queda definida mediante la estructura que se muestra a continuación:

```
$ = INICIO
B|CB|RC|GC = EVENTOS
| = SEPARADOR
& = UNION
21|32|55 = HORA|MINUTOS|SEGUNDOS
|15|09|2023 = DIA|MES|AÑO
& = UNION
314501877 = COORDENADAS LATITUDINALES
S = DIRECCION LATITUDINAL
603019028 = COORDENADAS LONGITUDINALES
E = DIRECCIÓN LONGITUDINAL
* = FIN

TRAMA EJEMPLO:
$B|CB|RC|GC&21|32|55|15|09|2023&314501877|S|603019028|E*
```

Figura 38. Estructura de tramas.

La trama está compuesta mediante un bit de inicio, seguida por los datos correspondientes a los eventos masticatorios ocurridos en el periodo procesado, separados mediante el carácter “ | ”.

Conjuntamente a estos, se añaden los datos de hora y fecha entregados por el GPS, y luego los datos de localización, los cuales incumbe a las coordenadas latitudinales y longitudinales con su respectivas direcciones.

Por último se agrega un bit de fin de trama, para indicar que la cadena finaliza en ese punto.

Partiendo de esta estructura, se programan las partes que conforman la recepción de los datos para su posterior desglose y visualización.

5.4.2 Transmisión

De manera paralela al recibimiento de los datos, almacenamiento y procesamiento, también se encuentra en ejecución el timmer 2 del microcontrolador, que es el encargado de contabilizar el tiempo de 5 minutos. Entonces, cada vez que salte la interrupción de este timmer, se pausa la operación de recepción en curso en la UART 2, y se desinicializa la UART, lo que implica desactivarla y restaurarla a su estado por defecto, liberando el recurso.

Se asigna el estado 03 a la variable ***estado,*** el cual fue explicado en la sección 5.3, y al finalizar se pasa al estado 04. Donde se realiza una iteración a través del arreglo que contiene todos los eventos clasificados y se los envía por medio del módulo LoRa. Se agrega un delay de 10 ms entre cada dato enviado para mejorar la correcta transmisión de estos. Luego de enviar los eventos, se realiza el envío de los valores de hora, fecha y ubicación. Para esto, se realiza un formateo de los datos en una cadena de caracteres mediante la función de C “*snprintf*”. Y se envía en último lugar, el bit de stop.

La trama se envía mediante la UART3 hacia un Arduino nano de manera serial. En el mismo se hace uso de librerías creadas para el manejo de los módulos LoRa, como lo es <*LoRa.h*> (Long-Range Radio) juntamente con <*SPI.h*> (Serial Peripheral Interface).

```
#include <SPI.h>
#include <LoRa.h>
```

Figura 39. librerías utilizadas para la transmisión.

En cuanto al estándar SPI, este es un bus de comunicación serie síncrono de alta velocidad en distancias cortas. Trabaja con 4 líneas de señal las cuales son, SCLK (Serial Clock), MOSI (Master Output and Slave Input), MISO (Master Input and Slave Output), y SS (Slave Selection) [29].

La comunicación de este protocolo es de manera sincrónica, y el funcionamiento de manera resumida es de la siguiente manera. El maestro inicia la comunicación y envía señales de reloj para sincronizar, es este caso con el esclavo. Luego, envía y recibe datos a través de las dos líneas, para enviar información (MOSI), y para recibir información (MISO). El dispositivo esclavo se conecta a estas líneas y se comunica con el maestro.

Por último, el maestro habilita al esclavo activando su línea de selección (CS), y luego envía o recibe los datos necesarios.

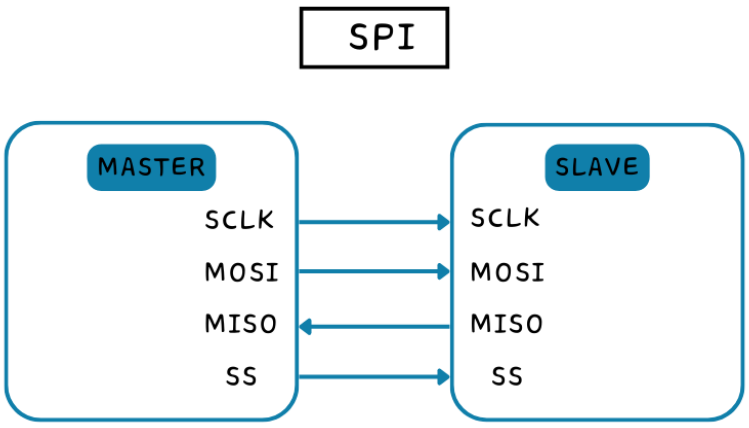

Figura 40, Protocolo SPI.

En cuanto a la programación en el dispositivo Arduino nano, en primera instancia, se inicia la comunicación configurando la velocidad de transmisión

en 9600 baudios. Se crea un bucle en el que se queda esperando hasta que la comunicación este lista, y luego inicializa el módulo Lora-02 para operar en la frecuencia de 433 MHz.

En caso de que la inicialización falle por algún motivo, se generara un mensaje de error y el programa se quedara en un buble infinito, hasta que se solucione la falla y se reinicie el sistema.

Pasando a la sección principal del programa, se verifica que haya datos disponibles en el puerto serial.

Luego, se declara un array en el que se almacenan todos los datos recibidos, y posteriormente se leen y almacenan los datos, junto con la cantidad de bytes que se leyeron.

Al transcurrir el envío de los datos, en el microcontrolador principal, es decir, en el STM32F103C8T6 ,se procede a reactivar la UART 2 para que nuevamente esté disponible para la recepción de datos en modo DMA con su respectiva configuración, y se cambia la variable estado al valor 01, equivalente al modo sleep.

5.5 Recepción en estación base

Del lado correspondiente a donde se coloca la estación base, como se nombró anteriormente, se tiene un sistema conformado por un Arduino UNO y un módulo LoRa. Se explica a continuación el desarrollo de las distintas partes y de la programación que compete a dicho apartado.

5.5.1 Arduino IDE

En cuando a la programación en el microcontrolador ATMEGA328P (Arduino UNO), se utilizan las mismas bibliotecas que en el lado de la transmisión. Es decir, <*LoRa.h*> y <*SPI.h*>.

Se inicializa la programación también de manera idéntica al caso anterior, configurando la velocidad de 9600 baudios, e inicializando el módulo para trabajar a 433 MHz.

Posteriormente, en el bucle principal, comprueba si hay paquetes de datos disponibles para recibir, y retorna el número de paquetes disponibles.

Si se detectan paquetes, ingresa en un bucle '***while***' que se encarga de leer y luego mostrar los datos disponibles a través del puerto serial.

5.5.2 Python Rx

Por otra parte, se programó un script en lenguaje Python, que se encarga de tomar los datos que llegan a través del puerto serial en el que se encuentre conectado el sistema receptor, y guarda los datos en un archivo de extensión "*.txt*".

Para este, se debe de incorporar en primer lugar la biblioteca *'pyserial'*:

```
import serial
```

Figura 41. Biblioteca usada para la lectura de datos seriales.

En caso de no contar con la misma instalada, se puede agregar tipeando en consola:

pip install pyserial

Una vez agregada la biblioteca utilizada para establecer y manejar la comunicación a través de puertos seriales, se debe crear una instancia de la clase '*Serial*' con la configuración del puerto correspondiente. Indicando el puerto, y la velocidad utilizada para la comunicación.

Posteriormente, se realiza la apertura de un archivo llamado '*recibido.txt*' de la siguiente manera:

```
archivo = open('recibido.txt', 'a')
```

Figura 42. Apertura de archivo para escritura.

En donde el parámetro 'a', indica que se lo abre en modo escritura pero que, en caso de ya contener datos, se posiciona al final de estos y agrega luego de estos, los datos nuevos.

Luego de esto, se ingresa en un bucle infinito que se ejecuta mientras el programa este en ejecución. Dentro del bucle, se leen los datos del puerto serie, se decodifica si es necesario, y se eliminan los espacios en blanco. Detectando en inicio de la trama mediante la lectura del carácter '$'.

Posteriormente, si es detectado el carácter de fin de trama, ' * ', se agrega un salto de línea en el archivo, para que cada trama quede ubicada en un renglón distinto en el archivo. Por último para asegurar que los datos escritos queden almacenados en el archivo, se hace uso de la función '*flush()*'.

Se agregan dos manejos de interrupciones, en el que en uno se indica que si hay un error en la lectura del puerto COM, se capture la interrupción y se muestre el mensaje de error. Y en la otra, si se interrumpe el programa mediante una interrupción de teclado como por ejemplo el uso de Ctrl+C, se muestre un mensaje de detención, y se cierre el archivo y el puerto en uso.

Un ejemplo de recepción de datos puede verse en la siguiente figura:

```
1  $CB|CB|CB|CB|CB|CB|CB|CB|CB|CB|CB|CB|CB|CB|CB|CB|CB|CB|CB&21|39|39|16|10|23&31.450247|S|60.30184|W*
2  $CB|CB|B|CB|CB|CB|CB|CB|CB|CB|CB|CB|CB|CB|CB|CB|CB|CB|CB|CB&21|43|55|16|10|23&31.450209|S|60.301952|W*
3  $&21|44|29|16|10|23&31.450232|S|60.301937|W*
```

Figura 43. Ejemplo de cadenas recibidas.

5.5.3 Visualización de resultados mediante página web

Para poder visualizar los datos de manera más adecuada, se programó una página web. La misma es capaz de mostrar los datos recibidos mediante el módulo LoRa en la estación base.

El lenguaje utilizado para la programación fue Python, versión 3.9. Con ayuda del framework web *Flask*, el cual es gratuito y de código abierto. Lo que permite crear un programa que se ejecuta en el mismo servidor.

Por lo que se necesita, el programa como tal, y la herramienta para lograr conectar el servidor web a la aplicación. Cuando el servidor web realiza peticiones a la aplicación, obtendrá respuestas que serán enviadas, en este caso, al navegador que estemos utilizando y mediante el que se ha establecido la conexión.

Por lo que, en primer lugar se debe importar el módulo Flask en Python En caso de no se encontrarse en el sistema previamente, se lo debe instalar mediante el comando:

pip install pyserial

Luego, se debe crear una instancia ‘Flask’ llamada app. Esta instancia es la base de la aplicación web, y es esencial ya que Flask utiliza esta información para lograr determinar la ubicación de las plantillas y archivos estáticos asociados a la misma.

Seguidamente, se define la función *procesar_trama()* en la que se realiza el desglose de todos los bytes recibidos en la cadena que fue enviada desde el dispositivo.

A esta función, se le pasa como argumento la cadena entera que se recibe, y que fue previamente almacenada en el archivo de texto mediante el programa descripto con anterioridad. Cabe destacar que a esta cadena se le quitan los símbolos mencionados como bit de inicio y fin, antes de pasarla como argumento a la presente función.

```python
def procesar_trama(trama):
    eventos = []
    partes = trama.split('&')

    parte1 = partes[0].split('|')
    parte2 = partes[1] if partes[1] else None
    if parte2 is not None:
        parte2 = parte2.split('|')
    parte3 = partes[2] if partes[2] else None
    if parte3 is not None:
        parte3 = parte3.split('|')

    for eventito in parte1:
        evento = eventito
        if parte2 is not None:
            hora = f"{parte2[0]}:{parte2[1]}:{parte2[2]}"
            fecha = f"{parte2[3]}/{parte2[4]}/{parte2[5]}"
        else:
            hora = f"N/A"
            fecha = f"N/A"
        if parte3 is not None:
            latitud = f"{parte3[0][:2]}{parte3[0][2:]}"
            lat_dir = f"{parte3[1]}"
            longitud = f"{parte3[2][:2]}{parte3[2][2:]}"
            lon_dir = f"{parte3[3]}"
```

Figura 44. Función procesar_trama, parte 1.

Se crea una lista vacía, que se utilizara para almacenar los datos procesados. Luego se divide en partes la trama de datos, mediante el carácter usado como carácter de unión ‘&’. Con esto se obtienen las distintas secciones de datos. Esto implica que, en una primera parte se tienen los datos que corresponden a los eventos detectados por el sistema, en una segunda parte los datos correspondientes a la hora y a la fecha, y en una tercer parte los datos de ubicación. A su vez, cada uno de los datos se encuentra formateado mediante el carácter ‘|’.

Se realizan comprobaciones para detectar que si una de las secciones de datos no contiene datos, sea porque en el tiempo transcurrido no se detectaron

eventos en el sistema o porque el GPS no logro conectarse a los satélites, para así evitar tener valores erróneos.

Si la parte fue considera, es decir esa parte contiene datos, se divide utilizando el separador de datos usando para el formateo de estos, '|'. Almacenando cada uno de los valores en una lista.

Posteriormente, se itera sobre un ciclo '*for*' en el que se toma la primer parte de las tres que se tienen, y se recorre cada uno de los datos de la lista.

Se comprueba que la parte 2, no este vacía, y si se cumple, se asigna los datos correspondientes a las variables de hora y de fecha. De lo contrario, se asigna 'N/A' cuando no exista información sobre esos datos.

Similar a lo anterior, se realiza con la parte 3, designando los valores a las variables de latitud, lat_dir, longitud, y lon_dir en caso de que corresponda, o bien 'N/A' si no existen los datos.

Cada evento se almacena en una lista, denominada '**eventos**', como un diccionario cuyas claves son 'evento', 'hora', 'fecha', latitud', 'lat_dir', 'longitud' y 'lon_dir'.

```
33            else:
34                latitud = f"N/A"
35                lat_dir = f"N/A"
36                longitud = f"N/A"
37                lon_dir = f"N/A"
38            eventos.append({
39                'evento': evento,
40                'hora': hora,
41                'fecha': fecha,
42                'latitud': latitud,
43                'lat_dir': lat_dir,
44                'longitud': longitud,
45                'lon_dir': lon_dir
46            })
47
48        print(eventos)
49        return eventos
```

Figura 45. Función procesar_trama, parte 2.

Entonces, en cada iteración, se va agregando un nuevo diccionario a la lista de eventos, con todos los valores asociados. Y luego está lista con todos los eventos, es la que retorna la función al finalizar.

Se define luego, un decorador del tipo @app.route() para poder asociarle así la ruta especifica y responder a las solicitudes HTTP. En nuestro caso como solamente se aloja de manera local la página, se debe colocar *@app.route('/')* haciendo referencia a la ruta raíz. Donde al ejecutarse, se debe acceder al navegador y tipear 127.0.0.1:5000, que es la dirección IP de loopback y el puerto por default utilizado por la aplicación.

Al acceder a la URL, se ejecuta la función, '*index()*' que devuelve la página principal HTML renderizada.

Se crea una lista vacía, para almacenar todos los eventos de todas las tramas recibidas.

Se abre el archivo de texto donde se encuentran almacenados los datos, dado por una ubicación especifica, en modo lectura. Y se incia un ciclo '*for*' para recorrer cada línea del archivo.

Se limpia cualquier carácter en blanco que exista al comienzo y al fin de cada línea, para limpiar la entrada. Luego, se verifica que si la línea comienza con el carácter '$' y finaliza con el carácter '*'.

Si se cumple la condición anterior, se eliminan ambos caracteres, ya que solo indican el inicio y fin simplemente de los datos.

Posteriormente, se llama a la función antes descripta, *procesar_trama()*, pasándole como argumento la trama ya lista para procesarla y así obtener la lista de eventos con el diccionario de los demás datos asociados.

Se procede a agregar en una lista de eventos totales, los 'eventos' retornados, obteniendo así todos los eventos de todas las líneas del archivo.

Finalmente, se renderiza la plantilla 'index.html' pasándole como argumento la información obtenida.

Respecto a la plantilla HTML, se define en la parte de la cabecera, una meta etiqueta con la que se actualiza automáticamente la página cada 5 segundos, y otra meta etiqueta que se utiliza para ajustar el comportamiento a

dispositivos móviles. También se define el título de la pestaña del navegador, y se enlaza la plantilla de estilos CSS externa.

En el cuerpo de la página, se hace uso de Bootstrap para el estilizado de los elementos. Se generan una barra de navegación en la que en la parte derecha se ubica siglas referentes a la Universidad Tecnológica Nacional y a Facultad regional Paraná.

Figura 46. Siglas simbólicas en la web.

Luego se crea un contenedor, y dentro del mismo una fila dentro de un sistema de grillas de Bootstrap en el que se organiza el título, visualizándose de la siguiente manera:

Figura 47. Título de la página.

Posteriormente, se genera una tabla de historial para la presentación de los datos resultados. En la misma se tienen las columnas correspondiente a cada valor, y los mismos se van ordenando por fila según la cantidad de eventos leídos. Esto último, se realiza mediante un ciclo *for* en el que se va iterando a través de la lista '***eventos***' que se le pasa a la página desde la parte del servidor Flask.

La visualización de la página web resulta como se ve a continuación en la figura 48:

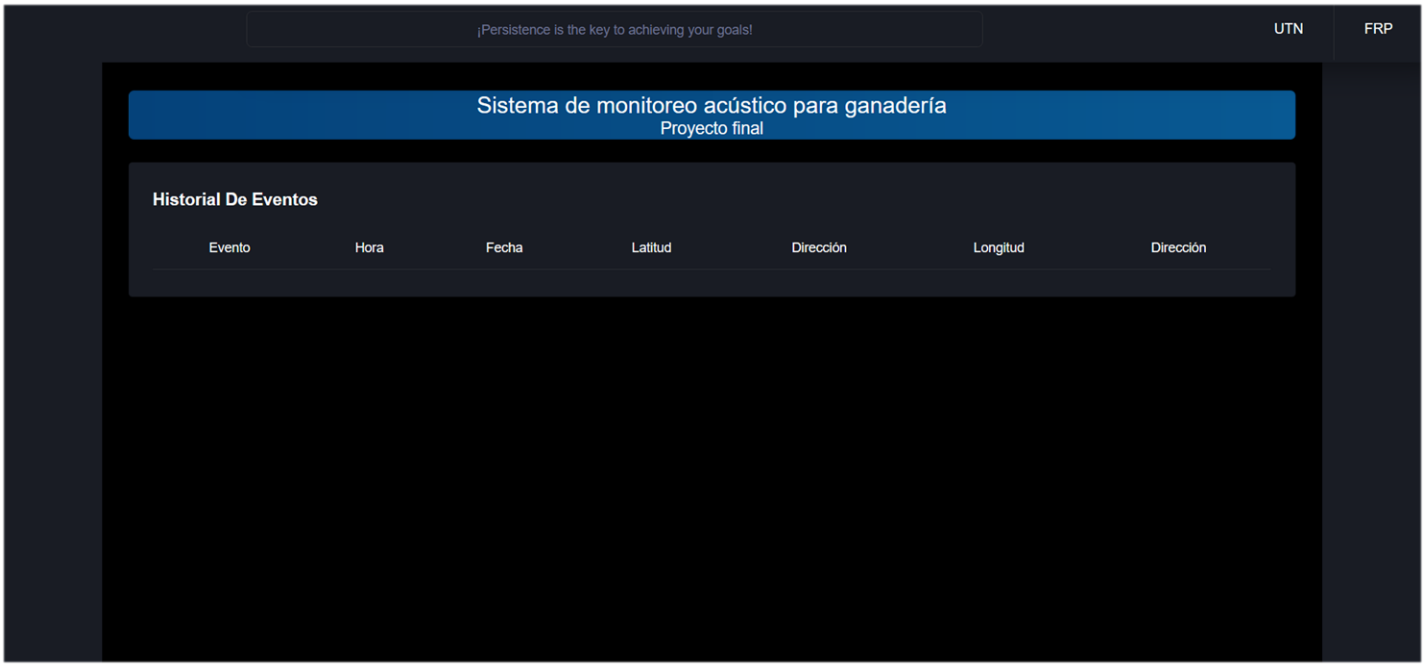

Figura 48. Página de visualización.

Capítulo 6: Resultados

Una vez armado el sistema completo, habiendo probado previamente cada parte de manera individual, se pasó a la siguiente etapa, correspondiente al testeo de funcionamiento.

Se llevaron a cabo pruebas para verificar tanto la integridad de los datos de audio recibidos por el sistema correspondientes a las grabaciones de bovinos en pastoreo libre, como la precisión de los eventos clasificados por el algoritmo. Los eventos resultantes fueron compararon con pruebas realizadas previamente al proyecto y se confirmó que el algoritmo funcionaba de manera confiable, incluso cuando se ejecutaba de manera independiente en el entorno CLion, produciendo los mismos eventos que cuando operaba junto con el hardware implementado.

Con el dispositivo en funcionamiento, se puede visualizar la recepción de los eventos clasificados resultantes del procesamiento con el sistema de la siguiente manera:

Historial De Eventos

Evento	Hora	Fecha	Latitud	Dirección	Longitud	Dirección
CB	21:39:39	16/10/23	31.450247	S	60.30184	W
CB	21:39:39	16/10/23	31.450247	S	60.30184	W
CB	21:39:39	16/10/23	31.450247	S	60.30184	W
CB	21:39:39	16/10/23	31.450247	S	60.30184	W
CB	21:39:39	16/10/23	31.450247	S	60.30184	W
CB	21:39:39	16/10/23	31.450247	S	60.30184	W
CB	21:39:39	16/10/23	31.450247	S	60.30184	W
CB	21:39:39	16/10/23	31.450247	S	60.30184	W
CB	21:39:39	16/10/23	31.450247	S	60.30184	W

Figura 49. Visualización de eventos detectados.

Se debe destacar que tanto la latitud como la longitud son valores negativos debido a que nos encontramos ubicados al oeste del meridiano de Greenwich

y al sur del Ecuador, pero en la página se verán representados como valores positivos.

Podemos destacar que el dispositivo tiene varias ventajas significativas.

En primer lugar, el microcontrolador utilizado ha demostrado ser una herramienta excepcionalmente capaz y robusta para ejecutar el sistema. A pesar de sus limitaciones en cuanto a memoria, ofrece un rendimiento y eficiencia notables en todas las tareas que realiza.

Ha demostrado ser una elección acertada para el desarrollo del proyecto, capaz de trabajar incluso en condiciones de alta exigencia.

Otro de los puntos fuertes que presenta es su sistema de transmisión, el cual logra proporcionar una comunicación de espectro extendido de gama ultra larga con una distancia de comunicación de hasta 10 km, y una gran inmunidad a la alta interferencia. Aunque esta distancia pueda no ser sorprendente para un dispositivo de monitoreo en campo abierto, es una característica valiosa cuando se utiliza en pruebas controladas en animales en parcelas.

Finalmente, su bajo consumo de energía lo convierte en un dispositivo eficiente, aunque no sea completamente autónomo en esta fase de evaluación para determinar su duración operativa real.

En cuanto a las posibles desventajas, es importante destacar que el uso de placas de desarrollo puede aumentar los costos en comparación con desarrollos en los que todas las etapas se integran en un solo PCB sin módulos modulares.

Capítulo 7: Análisis de Costos

En lo que respecta al costo total del proyecto, se ha elaborado una lista de los materiales utilizados, expresados en dólares para tener una referencia más estable, teniendo en cuenta que debido a la fluctuación de nuestra moneda local sería muy difícil reflejar los valores.

Componente	Precio en U$S	Cantidad	Total en U$S
Blue Pill stm32F103C8T6	11,97	1	11,97
Arduino nano	9,06	1	9,06
Arduino UNO	11,54	1	11,54
Programador St-link V2	6,77	1	6,77
Conversor USB A Serie CP2102	4,35	1	4,35
Modulo GPS Gy-neo6mv2 Con Antena	12,06	1	12,06
Modulo LoRa-02 sx1278	10,65	2	21,29
Antena 433 MHz	3,34	2	6,67
Cargador batería Cn3791	8,44	1	8,44
Fuente DC step-up MT3608 de 2A	3,16	1	3,16
Baterías LG 18650 3.7v 2400mAh	5,14	1	5,14
Panel Solar 5v 200ma	7,60	1	7,60
Conector Mini Jst Zh 2.0mm 2 Mm (pack x5)	2,53	1	2,53
Placa de fibra de vidrio 10x15 cm	3,90	1	3,90
Tira de pines hembra	1,32	3	3,96
Impresión de gabinete 3D	5,38	1	5,38
Gastos de envío	6,15	1	6,15
TOTAL GASTADO			**129,99**

Tabla 3. Tabla de costos de materiales.

El costo total del proyecto se desglosa de la siguiente manera, de materiales y componentes utilizados, se tiene un costo de **U$S 130**. Estos precios fueron obtenidos de la plataforma MercadoLibre, que es donde se adquirieron la mayoría de los componentes. Y para la conversión del valor, se tomó el precio en pesos argentinos, y se consideró valor de dólar oficial, teniendo en cuenta los impuestos extras que se cobran al comprar, obteniendo así un precio de $ 650 por dólar.

De mano de obra, se contempla la cantidad de 200 horas dedicadas a la investigación y desarrollo. Tomando en cuenta un valor de $ 1500 la hora, se considera un costo de mano de obra equivale a $300000.

Y para tener la misma referencia junto con los materiales, equivalente al dólar considera corresponde a U$S **462.**

Por lo que se tiene entonces, un costo total de proyecto de **U$S 592**.

Debido a que la finalidad de este proyecto es académica, y que también se necesita implementar determinadas mejoras para lograr obtener un dispositivo comercial como tal, no se presenta un plan futuro de venta del producto ni tampoco de amortización de este.

Para poder realizar los cálculos correspondientes a amortizaciones, se requeriría obtener mayor cantidad de datos que involucrarían varios factores relacionados al desarrollo de los animales en prueba a lo largo de un plazo mediano a largo.

Esto sería para contemplar medidas implementadas, teniendo en cuenta los resultados obtenidos con el dispositivo, y posteriormente los resultados consecuentes de dichas implementaciones en contraste con los costos de producción cárnica vs ganancias. Teniendo siempre presente, la calidad de vida animal y su respectivo bienestar.

Capítulo 8: Discusión y Conclusión

A pesar de presentarse como un proyecto desafiante, y donde cuyo desarrollo quedo marcado ante las limitaciones ya presentadas, se logró sortear todos los inconvenientes que se fueron presentando a lo largo del desarrollo, logrando obtener así un sistema que cumple con el objetivo inicial.

Respecto de los inconvenientes encontrados, se puede mencionar las incompatibilidades de los protocolos de comunicación presentado en la etapa de transmisión. El cual fue solventado mediante la incorporación del dispositivo intermediario entre la placa Blue Pill y el módulo LoRa-02.

En cuanto al objetivo inicial, el mismo consistió en lograr la implementación y adaptación del algoritmo CBEBA en el microcontrolador STM32f103C8T6. Por lo que se logra concluir que el microcontrolador es totalmente apto para el procesamiento de datos con este algoritmo.

En comparación con otros desarrollos como por ejemplo el *sistema de adquisición y análisis de información acústica para ganadería de precisión* de Chelotti en 2018 [30], se destaca la capacidad de detección de un evento adicional. En desarrollos anteriores como el mencionado, la detección se limitaba a tan solo 3 eventos, mientras que ahora se logra detectar 4 eventos. En adicción, el algoritmo presenta mayor tasa en la clasificación de los mismo, y también se destaca su inmunidad frente a niveles de ruidos elevados.

Tomando en cuenta los valores de costo en el desarrollo, se considera que siempre un primer desarrollo conlleva un mayor gasto. Pero luego de obtener un dispositivo final y lograr reproducirlo en masa el mismo disminuye significativamente.

8.1 Desarrollos futuros

Con visión a futuro, en cuanto a las posibles mejoras en el desarrollo, se pueden contemplar una serie de cambios para una nueva versión. Por un lado, es necesario el desarrollo del hardware de la etapa de adquisición de la señal

mediante micrófono, y su consecuente manejo en la parte de software para lograr así realmente la autonomía del sistema.

En cuando a la parte de transmisión, sería ideal lograr resolverlo mediante la utilización del módulo LoRa-02 pero conectado directamente con la placa de desarrollo principal para, así, evitar un paso intermedio solo por cuestiones de funcionamiento de protocolos internos de las librerías.

Otro punto por considerar es la parte energética, se podría buscar la manera de optimizar aún más el funcionamiento para poder tener el mínimo gasto posible. Teniendo en cuenta que se usa la placa BluePill, un cambio que se podría realizar es quitar el led de encendido que posee, ya que consume energía innecesaria, y que por software no es posible desactivarlo al mismo por la manera en la que se encuentra realizada la conexión de este.

A un mayor nivel de precisión, se podría analizar la posibilidad de armar la placa utilizando el mismo microcontrolador principal como núcleo, y añadiendo los componentes necesarios para lograr obtener el sistema dedicado exclusivamente para esta finalidad. Y así evitar el uso de placas de desarrollo que ocupan mayor tamaño físico y que contienen componentes que resultan innecesarios.

Con esto se podría llegar a obtener una gran reducción de costos en materiales, pero que en primera instancia llevaría una gran cantidad de horas de dedicación hasta lograr el diseño objetivo.

Capítulo 9: Literatura Citada

[1] R. Boari, N. Chuard, V. Fernández, y P. Pouiller (2014). <<Mercado de Ganados y Carnes. Proyecciones 2023. OCDE-FO,>> [En línea]. Available: https://www.magyp.gob.ar/sitio/areas/bovinos/informacion_interes/informes_historicos/_archivos/000003=Mercado%20internacional%20de%20carnes/000001-Proyección%20OCDE%20FAO%20carnes%202014-2023.pdf. [Último acceso: 04 18 2023].

[2] OECD/FAO (2020). <<OCDE-FAO Perspectivas Agrícolas 2020-2029,>> [En línea]. Available: https://doi.org/10.1787/a0848ac0-es [Último acceso: 18 04 2023].

[3] G. Aranda, F. De La Cruz Arbizu, E. González, J. C. Tulli, A. Uriz, y P. Agüero, <<*Sistema georeferenciador con parcelamiento virtual y adquisidor de sonidos masticatorios en rumiantes en pastoreo extensivo,*>> en Congreso Argentino de Sistemas Embebidos (CASE), 2012.

[4] EEA Catamarca-La Rioja y UNCa-FTYCA, <<*Dispositivo para monitoreo del desplazamiento y la condición del ganado en sistemas pastoriles extensivos*,>>, en INTA Argentina.

[5] A. Pordomingo y R. Garro, <<Ganadería de precisión: Sistema de medición automatizado para la evaluación animal de consumo residual y comportamiento,>> [En línea]. Available: https://www.argentina.gob.ar/sites/default/files/2021/05/ganaderia_de_precision.pdf. [Último acceso: 04 20 2023].

[6] A. Micheau y D. Hoa, <<Anatomía bovina - Atlas ilustrado,>> [En línea]. Available: https://www.imaios.com/es/vet-anatomy/bovinos/toro-y-vaca-anatomia-general. [Último acceso: 04 24 2023]

[7] L. E. Graham, J. M. Graham, & L. W. Wilcox, <<Plant biology,>> Pearson Education, Second Ed., 2013.

[8] J. B. Russell, <<Rumen Microbiology and Its Role in Ruminant Nutrition,>> New York: Cornell University Press, 2002.

[9] <<Sistema digestivo de la vaca,>> [En línea]. Available: https://www.ugrj.org.mx/index2.php?option=com_content&do_pdf=1&id=388. [Último acceso: 20 05 2023].

[10] E. Van Lier, << Anatomía digestiva,>> [En línea]. Available: http://prodanimal.fagro.edu.uy/cursos/AFA/TEORICOS/21%20-%20Anatomia%20digestiva.pdf. [Último acceso: 05 15 2023]

[11] Guzmán, M., García, R., & Pérez, J, <<Importancia de la rumia en la digestión de los bovinos: un proceso complejo y esencial,>>. Revista de Investigación Agrícola y Ambiental (2015), 21(2), 45-55.

[12] Mette S Herskin, Lene Munksgaard, and Jan Ladewig, <<Effects of acute stressors on nociception, adrenocortical responses and behavior of dairy cows,>> in: Physiol. Behav. 83.3 (2004), pp. 411–420.

[13] M. Benvenutti, D. Pavetti, D. Poppi, I. Gordon, and C. Cangiano, <<Defoliation patterns and their implications for the management of vegetative tropical pastures to control intake and diet quality by cattle,>> in Grass and Forage Science (2016).

[14] L. S. Martínez Rau, <<Desarrollo de un sensor inteligente para el monitoreo continuo de animales en pastoreo para ganadería de precisión,>> en Universidad Nacional del Litoral (2022), pp. 5-6.

[15] <<STM32F103x8,>> ST. Datasheet. DS5319 Rev 18.

[16] R. Friis Kjeldsen, <<Diagrama de distribución de pines,>> [En línea]. Available:https://upload.wikimedia.org/wikipedia/commons/9/90/Stm32f103_pinout_diagram.png. [Último acceso 06 08 2023]

[17] Diego H Milone et al, <<Automatic recognition of ingestive sounds of cattle based on hidden Markov models,>> in Computers and electronics in agriculture (2012), pp. 51–55.

[18] José O Chelotti et al., <<A real-time algorithm for acoustic monitoring of ingestive behavior of grazing cattle,>> in Computers and Electronics in Agriculture (2016), pp. 64–75.

[19] José O Chelotti et al., <<A pattern recognition approach for detecting and classifying jaw movements in grazing cattle,>> in Computers and Electronics in Agriculture, (2018).

[20] L. S. Martinez Rau et al., << A robust computational approach for jaw movement detection and classification in grazing cattle using acoustic signals,>> in Computers and Electronics in Agriculture, (2021).

[21] <<NEO-6 series. Versatile u-blox 6 GPS modules.>> [En línea]. Available: http://www.openimpulse.com/blog/wpcontent/uploads/wpsc/downloadables/Ublox-NEO-6-GPS-Product-Summary.pdf. [Último acceso: 24 07 23].

[22] <<u-Blox. Receiver description. Including Protocol Specification,>> [En línea]. Available: http://www.openimpulse.com/blog/wpcontent/uploads/wpsc/downloadables/Ublox-6-GPS-Receiver-and-Protocol-Description.pdf. [Último acceso: 24 07 23].

[23] << What are LoRa and LoRaWAN?>> [En línea]. Available: https://www.thethingsnetwork.org/docs/lorawan/what-is-lorawan/#:~:text=LoRa%20is%20a%20wireless%20modulation,be%20received%20across%20great%20distances. [Último acceso: 26 07 2023].

[24] A. Andreu Dólera, <<Modelado de redes Lorawan aplicadas a la conservación preventiva de patrimonio cultural,>> en Universidad Politécnica de Valencia (2019), p. 28.

[25] ENACOM Ente Nacional de Comunicaciones. <<Bandas de uso compartido sin autorización.>> [En línea]. Available: https://www.enacom.gob.ar/bandas-de-uso-compartido-sin autorizacion_p680. [Último acceso: 27 07 2023].

[26] << 4A, Standalone Li-ion Battery Charger IC With Photovoltaic Cell MPPT Function CN3791,>> Consonance. Datasheet. Rev. 1.0.

[27] <<MT3608,>> AEROSEMI. Datasheet.

[28] J. O. Chelotti et al., <<An online method for estimating grazing and rumination bouts using acoustic signals in grazing cattle,>> in Computers and Electronics in Agriculture, (2020). p. 15.

[29] N. Anand, D. Joseph, S. S. Oommen, and R. Dhanabal, <<Design and implementation of a high speed Serial Peripheral Interface,>> in International Conference on Advances in Electrical Engineering (ICAEE), (2014). pp 1-3

[30] J. O. Chelotti, <<Sistema de adquisición y análisis de información acústica para ganadería de precisión,>> en Universidad Nacional del Litoral (2018).

Anexo A

A robust computational approach for jaw movement detection and classification in grazing cattle using acoustic signals

Luciano S. Martinez Rau[a], José O. Chelotti[a], Sebastián R. Vanrell[a], Julio R. Galli[b,c], Santiago A. Utsumi[d], Alejandra M. Planisich[c], H. Leonardo Rufiner[a,e], Leonardo L. Giovanini[a]

[a] *Instituto de Investigación en Señales, Sistemas e Inteligencia Computacional, sinc(i), FICH-UNL/CONICET, Argentina*
[b] *Instituto de Investigaciones en Ciencias Agrarias de Rosario, IICAR, UNR-CONICET, Argentina*
[c] *Facultad de Ciencias Agrarias, Universidad Nacional de Rosario, Argentina*
[d] *W.K. Kellogg Biological Station and Departament of Animal Science, Michigan State University, United States*
[e] *Facultad de Ingeniería, Universidad Nacional de Entre Ríos, Argentina*

Abstract

Monitoring behaviour of the grazing livestock is a difficult task because of its demanding requirements (continuous operation, large amount of information, computational efficiency, device portability, precision and accuracy) under harsh environmental conditions. Detection and classification of jaw movements (JM) events are essential for estimating information related with foraging behaviour. Acoustic monitoring is the best way to classify and quantify ruminant events related with its foraging behaviour. Although existing acoustic methods are computationally efficient, a common failure for broad applications is the deal with interference associated with environmental noises. In this work, the acoustic method, called Chew-Bite Energy Based Algorithm (CBEBA), is proposed to automatically detect and classify masticatory events of grazing cattle. The system incorporates computations of instantaneous power signal for JM-events classification associated with chews, bites and composite chew-bites, and additionally between two classes of chew events: i) low energy chews that are associated with rumination and ii) high

Email address: lmrau@sinc.unl.edu.ar (Luciano S. Martinez Rau)

energy chews that are associated with grazing. The results demonstrate that CBEBA achieve a recognition rate of 91.9% and 91.6% in noiseless and noisy conditions, respectively, with a high classification precision and a marginal increment of computational cost compared to previous algorithms, suggesting feasibility for implementation in low-cost embedded systems.

Keywords: Acoustic monitoring, Cattle grazing behaviour, Jaw movement classification, Noise robustness, Pattern recognition, Sound energy analysis.

1. Introduction

Precision livestock farming typically integrates smart animal monitoring technologies to aid farmers with relevant management decisions regarding animal nutrition, health and welfare (Michie et al., 2020). The deployment of animal monitoring dashboards has been enhanced recently by improved sensors (Andriamandroso et al., 2016), advanced communication technologies and enhanced visualization tools allowing for rapid inspection of production traits and behaviours associated with specific activities, changes of location and body posture (Berckmans, 2014).

The use of modern technologies based on fixed video cameras allow for individual or group behaviour monitoring in an automatic, continuous and non-intrusive way in a given fixed area (Fuentes et al., 2020). Their use is limited to small farming areas such as pens and stables. On the other hand, the use of small wearable video cameras on animals would allow to expand the operating region, although their application still needs further development (Saitoh and Kato, 2021). Thus, wearable sensors are the most widely used acquisition method to cover large farm and field areas. However, their operational requirements, primarily device portability, robustness and power capabilities, along with the computational cost and complexity of analytical components often represents an obstacle for further technological progress and adoption (Stone, 2020).

Ones of the most frequently used monitoring techniques are the position and motion sensors, which allows the surveillance of cattle and sheep movements (Andriamandroso et al., 2016). Nose band sensors (Nydegger et al., 2010; Werner et al., 2018; Zehner et al., 2017), multidimensional accelerometers (Andriamandroso et al., 2017; Greenwood et al., 2017; Smith et al., 2016) and jaw recorders have been applied to monitor animal locomotion as well as feeding and rumination activities, being used to alert farmers on

behavioural changes associated with diseases, estrus and labor. On the other hand, acoustics methods have been used for monitoring livestock feeding behaviours. Laca et al. (1992) used directional microphones, attached to the forehead of animals, for analysing masticatory sounds in cattle (Galli et al., 2018) and sheep (Galli et al., 2011), as well as for accurate discrimination between feeding and rumination bouts (Chelotti et al., 2020; Vanrell et al., 2018) and for feed intake prediction based on sound energy fluxes (Galli et al., 2018; Laca et al., 2000).

Masticatory sounds are the result of JM-events associated with bites, chews and composite chew-bites (Laca et al., 1992). A grazing bite includes the apprehension and severance of herbage. Chews include the crushing, grinding and processing of herbage during ingestion or rumination. Finally, chew-bites include the combination of chewing and biting in the same JM. Thus, the attributes and statistics of JM-events provide a reliable measure for identification of grazing (include bites, chews and chew-bites) and rumination (chews only) activities and related behaviour events (Chelotti et al., 2016; Milone et al., 2012). Rumination frequency is related to digestion processes and serves as indicator of a suitable rumen health (Sauvant, 2000). The sound properties of bites and chew-bites typically relate to common plant traits and feed structural characteristics (Laca et al., 2000), providing insights of short-term intake rate (Galli et al., 2018) and daily grazing time (Chelotti et al., 2020), two major determinants of the daily feed intake (Hodgson, 1990). A declining rumination time has been shown to be correlated with a declining feed intake (Watt et al., 2015), an acute stressors (Schirmann et al., 2011), an onset of diseases (DeVries et al., 2009), as well as the beginning of estrus (Schirmann et al., 2009) and parturition (Schirmann et al., 2013).

The acquisition of masticatory sounds is the first step of a good acoustic method because registered signals require further processing, analysis and information weighing, and extraction to become useful and insightful for animal monitoring. The analysis of masticatory sounds has been significantly improved in recent years. Milone et al. (2012, 2009) used concepts from automatic speech recognition to develop an algorithm based on hidden Markov models capable of identifying chew, bite, and chew-bite in sheep and cattle. It combines spectral analysis with language-based analysis to detect and classify JM-events. These algorithms achieved an average recognition rate of 80% of successful classifications of bites, chews and chew-bites when tested in controlled experiments lasting a few minutes under good signal-to-noise ratio (SNR) conditions. Chelotti et al. (2016) proposed an alternative algorithm

based on time-domain features of sound signals, achieving similar JM-events recognition rate success at a lower computational cost, compared to Milone et al. (2012), such that it can be implemented in portable microcontroller-based embedded systems (Deniz et al., 2017). More recently, Chelotti et al. (2018) modified and improved this algorithm (Chelotti et al., 2016) using concepts and tools derived from signal processing, pattern recognition and artificial intelligence areas without significantly increasing the computational cost. This algorithm, called Chew Bite Intelligent Algorithm (CBIA), attenuates the effects of time-varying noises and trends, and it achieves a 90% recognition rate. Although CBIA showed good performance for moderate SNR, significant limitations arose when it was employed in farming and husbandry environments, which typically involved louder and time-varying noise and disturbance sources. These negatively affect the detection, features extraction and classification of JM-events since they can not be completely removed from processes internal signals (see Chelotti et al., 2018).

The present work documented and tested the integration of a new set of tools for pattern recognition analysis and artificial intelligence for robust online analyses of masticatory sound signals collected from grazing livestock. The main objective is to achieve a more robust detection and classification of JM-events than the CBIA algorithm. Specifically, the new algorithm was especially designed to: i) attenuate distorting effects of environmental noises on masticatory signals likely associated with typical farming conditions and animal handling procedures; and ii) improve the detection and classification of JM-events without significantly increasing the computational cost. The computational cost and a cost-benefit analysis of the new algorithm and CBIA were also evaluated to assess future feasibility for real-time execution in low-cost embedded systems.

The paper is organised as follows: Section 2 analyses the CBIA and presents the new classification features intended to attenuate effects of noises on detection and classification tasks. Then, the proposed algorithm is introduced. Also it is described the acquisition of datasets, the performance measures and the experimental setup used to validate the algorithms. Section 3 shows the comparative results for the proposed algorithm and CBIA. A focused discussion and main conclusions follow in Section 4 and Section 5, respectively.