ESCUELA POLITÉCNICA NACIONAL

FACULTAD DE INGENIERÍA ELÉCTRICA Y ELECTRÓNICA

DISEÑO E IMPLEMENTACIÓN DE UN ALGORITMO ADAPTATIVO PARA LA OPTIMIZACIÓN DEL CONSUMO DE ENERGÍA DE UN NODO SENSOR INALÁMBRICO EN UNA APLICACIÓN DE MONITOREO REPRODUCTIVO BOVINO

TESIS DE GRADO PREVIO A LA OBTENCIÓN DEL TÍTULO DE MAGÍSTER EN CONECTIVIDAD Y REDES DE TELECOMUNICACIONES

DANNY RAÚL ARGUERO ZAPATA

dannyarguero@gmail.com

DIRECTOR: CARLOS ROBERTO EGAS ACOSTA

carlos.egas@epn.edu.ec

Quito, agosto 2017

AVAL

Certifico que el presente trabajo fue desarrollado por Danny Raúl Arguero Zapata, bajo mi supervisión.

MSc. CARLOS ROBERTO EGAS ACOSTA
DIRECTOR DEL TRABAJO DE TITULACIÓN

DECLARACIÓN DE AUTORÍA

Yo Danny Raúl Arguero Zapata, declaro bajo juramento que el trabajo aquí descrito es de mi autoría; que no ha sido previamente presentada para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedo mis derechos de propiedad intelectual correspondientes a este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

_____

DANNY RAÚL ARGUERO ZAPATA

DEDICATORIA

A la mujer que siempre estuvo a mi lado desde el día que vi la luz, a mi madre porque su amor infinito me ha permitido estar aquí.

AGRADECIMIENTO

A Dios, mi familia, Sarita, Edgar y María, mi hermana y padres que me han amado siempre, a mi esposa amada, Gabriela la mujer que amo y que me apoyo, a Marcelo Miranda mi suegro por su ayuda y consejos, a mis amigos, compañeros y profesores en especial a mi director MSc. Carlos Egas.

ÍNDICE DE CONTENIDO

AVAL I
DECLARACIÓN DE AUTORÍA II
DEDICATORIA III
AGRADECIMIENTO IV
ÍNDICE DE CONTENIDO V
ÍNDICE DE FIGURAS VIII
ÍNDICE DE TABLAS X
ÍNDICE DE ECUACIONES XI
RESUMEN XII
ABSTRACT XIII
1 INTRODUCCIÓN 1
1.1 Pregunta de Investigación 2
1.2 Objetivo General 2
1.3 Objetivos Específicos 2
1.4 Alcance 3
1.5 Marco Teórico 3
Fundamentos de las Redes de Sensores Inalámbricos RSI 3
Arquitectura de una RSI 4
Nodos Sensores 6
Pasarela o Gateway 9
Estación Base 10
Protocolos de Comunicación de RSI 10
Sistemas Operativos 13
Topología de Red para RSI 15
Consumo y Optimización de Energía en RSI 17
Fisiología Bovina 18
Ciclo Estral de un Bovino 18
Detección del Estro en Bovinos 20
Características de la Fisiología Reproductiva Bovina 21
Características Normales del Comportamiento Bovino 22

Estudios Relacionados 23
2 METODOLOGÍA 26
2.1 Diseño e Implementación del Algoritmo Adaptativo a la Fisiología Bovina 26
2.2 Modelando la Actividad Motriz 26
2.3 Características de la Actividad Motriz 27
2.4 Base para el Diseño del Algoritmo 30
2.5 Algoritmo Adaptativo 31
2.6 Diagrama de Flujo del Algoritmo Adaptado a la Actividad Motriz de la Vaca 32
2.7 Análisis del Tiempo de Muestreo 34
2.8 Prototipo de Prueba de RSI para una Aplicación de Monitoreo Reproductivo Bovino 35
2.9 Selección del Hardware 36
2.10 Modos de Consumo Energético 38
2.11 Diseño del Sistema de Monitoreo Bovino 39
2.12 Configuración del General del Nodo Sensor 40
2.13 Alimentación Eléctrica del Prototipo 42
2.14 Simulación de la Red de Sensores Inalambricos en OMNeT++ 43
2.15 Programación del Nodo Sensor con el Algoritmo Adaptativo a la Actividad Motriz de un Bovino 44
2.16 Programa de Lectura del Puerto Serial 50
2.17 Programa para el Usuario 52
2.18 Conectividad de la Red 802.15.4 58
3 RESULTADOS Y DISCUSIÓN 59
3.1 Resultados de la Simulación 59
3.2 Cálculo de la Energía Consumida por el Nodo sin aplicar el Algoritmo .. 59
3.3 Cálculo de Energía del Nodo Sensor implementado el Algoritmo Adaptado a la Fisiología Bovina 61
3.4 Análisis de los Resultados 69
3.5 Pruebas Prácticas 71
3.6 Prueba de Monitoreo de la Actividad Motriz 74
3.7 Prueba de Tiempo de Vida de las Baterías 74
3.8 Prueba de Tiempo de Vida de las Baterías con el Algoritmo 75
4 CONCLUSIONES 77
5 REFERENCIAS BIBLIOGRÁFICAS 79

6 ANEXOS 81
ANEXO I PROGRAMA DE LECTURA DEL PUERTO SERIAL 81
ANEXO II INTERFAZ GRÁFICA 95
ANEXO III MANUAL DE USUARIO DE LA APLICACIÓN 120
ORDEN DE EMPASTADO 125

ÍNDICE DE FIGURAS

Figura 1.1 Arquitectura de una RSI [4]. 5
Figura 1.2 Hardware de un nodo [5]. 6
Figura 1.3 Arquitectura de hardware de un nodo real [6]. 7
Figura 1.4 Pasarela Libelium [7]. 9
Figura 1.5 Estación base y red de sensores [4]. 10
Figura 1.6 Representación de topología punto a punto [4]. 15
Figura 1.7 Representación de topología estrella [4]. 16
Figura 1.8 Representación de topología árbol [4]. 16
Figura 1.9 Representación de la topología malla [4]. 17
Figura 1.10 Síntomas de estro, edema vulvar y monta de vaca. Fotografías tomadas de “The Drost Project” [15]. 19
Figura 2.1 Monta de vaca en estro. Fotografía adaptada de “The Drost Project” [15]. 27
Figura 2.2 Vacas comiendo pasto. 28
Figura 2.3 Diagrama de flujo algoritmo adaptado a la actividad motriz. 31
Figura 2.4 Diagrama de comparación de la aceleración. 32
Figura 2.5 Diagrama de flujo de un evento detectado. 33
Figura 2.6 Diagrama de flujo para validación del ángulo formado respecto al eje X. 34
Figura 2.7 Waspmote Libelium [6]. 37
Figura 2.8 Red punto a punto para aplicación de monitoreo de reproducción bovina 39
Figura 2.9 Prototipo de monitoreo reproductivo bovino 40
Figura 2.10 Formato de estructura de datos. 41
Figura 2.11 Batería Libelium 3,7v 6600 mAh. 42
Figura 2.12 Cálculo del ángulo del cuello de la vaca 50
Figura 2.13 Algoritmo y código 50
Figura 2.14 Esquema del programa de lectura de datos de la pasarela 51
Figura 2.15 Diagrama de programa de usuario 52
Figura 2.16 Casos de uso del programa ganadero 54
Figura 2.17 Diagrama de clases 55
Figura 2.18 Interfaz gráfica de inicio 55
Figura 2.19 Visualización de eventos capturados por el acelerómetro 56
Figura 2.20 Estadísticas de los eventos almacenados por fecha 57

Figura 2.21 Producción, muestra la información reproductiva de la vaca 57
Figura 2.22 Configuración, configuración de red. 58
Figura 2.23 Programa de lectura USB conectividad el nodo 58
Figura 3.1 Energía del nodo en función del tiempo 59
Figura 3.2 Línea de tiempo, modo de baja actividad motriz de la vaca 62
Figura 3.3 Línea de tiempo, modo de alta actividad motriz de la vaca 62
Figura 3.4 Caja de fibra de vidrio pintada en color rojo 71
Figura 3.5 Manufactura del collar en taller artesanal..... 71
Figura 3.6 Prototipo de prueba con nodo sensor instalado 72
Figura 3.7 Colocación de collar en la vaca..... 72
Figura 3.8 Área de pruebas de funcionamiento..... 73
Figura 3.9 Imagen real del área de pruebas, a la izquierda la estación base con la pasarela capturando información, a la derecha una de las vacas lleva el nodo de prueba. 73
Figura 3.10 Monitoreo de la actividad motriz de la vaca 74
Figura 3.11 Energía de la batería consumida en función del tiempo..... 74
Figura 3.12 Energía consumida por la batería con el algoritmo adaptado a la fisiología bovina..... 75
Figura 3.13 Actividad motriz de una vaca lista para ser inseminada..... 76

ÍNDICE DE TABLAS

Tabla 1.1 Nodos comerciales 7
Tabla 1.2 Fases del ciclo estral bovino 18
Tabla 1.3 Porcentaje de efectividad de determinación del estro. 21
Tabla 2.1 Actividad motriz de un bovino 29
Tabla 2.2 Actividad motriz en la etapa de estro 29
Tabla 2.3 Características Nodo Waspmote 37
Tabla 2.4 Configuración general 40
Tabla 3.1 Consumo de energía por procesos en el nodo Waspmote 60
Tabla 3.2 Consumo de energía nodo modo de baja actividad motriz de la vaca, subproceso de adquisición de la aceleración 63
Tabla 3.3 Consumo de energía nodo modo de baja actividad motriz de la vaca, subproceso de transmisión de datos 64
Tabla 3.4 Consumo de energía nodo modo de baja actividad motriz de la vaca, subproceso de nodo durmiendo profundamente 64
Tabla 3.5 Consumo de energía modo alta actividad motriz de la vaca, subproceso de adquisición de la aceleración 66
Tabla 3.6 Consumo de energía nodo modo de alta actividad motriz de la vaca, subproceso de transmisión de datos 67
Tabla 3.7 Actividad motriz de la vaca 68
Tabla 3.8 Carga energética según actividad motriz en un ciclo estral 68
Tabla 3.9 Costos de una mala detección del estro 70

ÍNDICE DE ECUACIONES

Ecuación 2.1 Energía de la Batería 43
Ecuación 2.2 Capacidad de energía usada en Omnet++ 43
Ecuación 3.1 Consumo promedio de corriente 61
Ecuación 3.2 Vida de la Batería 61

RESUMEN

El despliegue de las redes de sensores ha permitido su aplicación en diferentes campos, en los que se necesita adquirir información de manera remota, así los nodos sensores se distribuyen espacialmente, y mediante módulos de comunicación inalámbrica basados en el estándar IEEE 802.15.4 transmiten esta información.

En ambientes de movilidad, los nodos son energizados por baterías, y la vida de estos se limita al tiempo de descarga de las mismas, por lo que es fundamental estudiar nuevas maneras de optimizar el consumo de energía en estas redes.

En el presente proyecto se aplicaron las redes de sensores inalámbricos en el monitoreo reproductivo de vacas lecheras, desplegando un prototipo conformado por un nodo sensor dentro de un collar ubicado en el cuello de una vaca. De esta forma el sensor monitorea la actividad motriz del animal y transmite inalámbricamente únicamente información de interés.

La información transmitida por el nodo sensor es capturada por una pasarela (o gateway en inglés) y es almacenada en una base de datos, para luego ser visualizada mediante una aplicación por un profesional veterinario o el encargado de la ganadería.

La aplicación de monitoreo reproductivo requiere un tiempo de vida amplio de baterías, por lo que fue necesario el desarrollo de un algoritmo adaptativo a la actividad motriz de la vaca, que permitió disminuir el número de muestras de esta actividad, leyendo y transmitiendo únicamente datos de interés, consiguiendo una optimización del consumo energético de la batería y aumentado el tiempo de vida del nodo en la red.

PALABRAS CLAVE: redes de sensores inalámbricos, IEEE 802.15.4, nodo sensor, pasarela, monitoreo reproductivo bovino, adaptativo, optimización de la batería, actividad motriz.

ABSTRACT

The deployment of sensor networks has allowed its application in different fields where it is necessary to acquire information remotely, so the sensor nodes are distributed spatially and transmit this information through wireless communication modules based on the IEEE 802.15.4 standard.

In mobile environments, the nodes are energized by batteries, and the life of these is limited to the time of discharge of the same, reason why it is fundamental to study new ways to optimize the energy consumption in these networks.

In the present project, the wireless sensor networks were applied in the reproductive monitoring of dairy cows, deploying a prototype conformed by a sensor node inside a collar located in the neck of a cow. In this way, the sensor monitors the animal's motor activity and wirelessly transmits only information of interest.

The information transmitted by the sensor node is captured by a gateway and is stored in a database, and then visualized by an application by a veterinarian or the manager of the cattle ranch.

The reproductive monitoring application requires a long life of batteries, so it was necessary to develop an algorithm adaptive to the motive activity of the cow, which allows to reduce the number of samples of this activity, reading and transmitting only data of interest, achieving an optimization of the energy consumption of the battery and increasing the lifetime of the node in the network.

KEYWORDS: wireless sensor networks, IEEE 802.15.4, sensor node, gateway, bovine reproductive monitoring, adaptive, battery optimization, motor activity.

1 INTRODUCCIÓN

Las redes de sensores inalámbricas RSI (o WSN, wireless sensor networks por sus siglas en inglés), son redes de comunicación compuestas por dispositivos electrónicos denominados nodos, que mediante el uso de sensores pueden monitorear el entorno donde son desplegados, transmitiendo esta información con módulos de comunicación inalámbrica. Sus principales áreas de aplicación son monitoreo del medio ambiente, monitoreo industrial, ciudades inteligentes, monitoreo de aplicaciones agrícolas y pecuarias, entre otras.

Sin embargo, pese a su gran despliegue, una de las limitantes de las redes de sensores inalámbricos es el tiempo de vida de los nodos en red, debido a que son energizados por baterías haciendo que la vida del nodo esté limitada al tiempo de descarga de su batería. Por esta razón, el desarrollo de estrategias de disminución y optimización del consumo energético en los nodos sensores es un campo de estudio permanente.

Una posible aplicación de la RSI es el monitoreo reproductivo en bovinos, donde el principal objetivo es identificar el día fértil de una vaca, permitiendo mejorar la gestión reproductiva de las ganaderías, obteniendo más crías al año y aumentando su rentabilidad.

Actualmente, las principales técnicas de monitoreo del día de fertilidad en bovinos son: la observación física de los animales y el uso de podómetros[1]. Los podómetros son efectivos, pero no se dispone de una información en tiempo real, lo cual puede llegar a que se insemine una vaca fuera de tiempo, disminuyendo la eficacia de la inseminación[2]; mientras que la observación física puede ser inadecuada por inexperiencia o negligencia del observador e igualmente existe la posibilidad de una detección muy tardía que disminuye el porcentaje de efectividad al momento de realizar la inseminación [1].

Según la asociación Holstein de Ecuador, una vaca no inseminada en el momento adecuado representa un costo de $ 5 diarios y este valor puede incrementarse cuando existen inseminaciones fallidas por una mala determinación del día fértil, aumentando el costo de un veterinario cuyo promedio es de $ 30 por visita y los insumos de inseminación, que puede variar de $ 15 hasta $ 40 promedio en las ganaderías del país [2].

Por las razones expuestas, una mejor forma de detectar los días fértiles de las vacas es con el uso de una red de sensores, que monitoree en tiempo real la actividad motriz de las vacas; pero en la actualidad esta aplicación no ha sido difundida a nivel nacional, debido a

[1] Podómetros.- dispositivo electrónico que cuenta cada paso que un individuo realiza.
[2] Inseminación.- técnica de fecundación asistida.

que los sensores necesitan una batería que suministre energía y sin una adecuada técnica de monitorización y muestreo, el mantenimiento y reemplazo de la batería es continuo.

Así, surge la necesidad de desarrollar un algoritmo adaptativo a los comportamientos típicos de la fisiología bovina, optimizando el consumo de energía del nodo sensor, haciendo que el despliegue de una red de sensores en las ganaderías sea un proyecto viable, que no requiera de un gran mantenimiento y envié información en tiempo real a los administradores ganaderos, optimizando su tiempo y recursos económicos.

Así, el desarrollo del algoritmo permitió generar un proceso de análisis de los aspectos fisiológicos de la reproducción en bovinos, caracterizando y analizando principalmente la actividad motriz de un bovino y mediante una estrategia de monitoreo basada en la actividad motriz construir el algoritmo adaptativo e implementarlo en un prototipo de prueba.

Todo el procedimiento realizado es descrito en este proyecto, permitiendo ser reproducido o aplicado para otro campo de estudio de las RSI y puede ser usado para desarrollar futuros cambios o algoritmos.

Un beneficio directo del estudio es que el proyecto puede ser implementado a mayor escala en el sector ganadero mejorando la gestión reproductiva y disminuyendo los costos de operación.

1.1 Pregunta de Investigación

¿El diseño e implementación de un algoritmo adaptativo a la fisiología animal optimizará el consumo de energía de un nodo sensor inalámbrico en una aplicación de monitoreo reproductivo bovino?

1.2 Objetivo General

Diseñar e implementar un algoritmo adaptativo para la optimización del consumo de energía de un nodo sensor inalámbrico en una aplicación de monitoreo reproductivo bovino.

1.3 Objetivos Específicos

- Desarrollar un algoritmo adecuándolo a la fisiología bovina para disminuir el consumo de energía eléctrica de un nodo sensor.
- Establecer el consumo eléctrico de un nodo sensor mediante un simulador de redes de sensores.

- Implementar un prototipo en una red de sensores punto a punto para probar el algoritmo desarrollado.
- Cuantificar el consumo de energía del sensor con el algoritmo implementado, en un ambiente real.

1.4 Alcance

El presente proyecto plantea el desarrollo de un algoritmo adaptativo basado en la actividad motriz de una vaca, que implementado en una aplicación de monitoreo reproductivo de bovinos permita a un nodo sensor disminuir su consumo de energía.

La aplicación de monitoreo reproductivo bovino estará constituida por un nodo sensor, una pasarela y una interfaz de visualización de datos. El nodo tendrá por sensor un acelerómetro encargado de monitorear la actividad motriz del animal, estará alimentado con baterías y su despliegue será a campo abierto protegido con una caja hermética a ser colocada como collar en el cuello de una vaca.

La pasarela estará conectada a un computador que almacena la información que es transmitida por el nodo sensor.

La interfaz gráfica de usuario para esta aplicación es un sistema de visualización de datos, que presente la información reproductiva básica de la vaca y permita validar la funcionalidad del sistema.

Para lograr la disminución del consumo de energía el algoritmo adaptativo se basará en la actividad motriz de la vaca y buscará reducir el número de adquisiciones de datos de esta actividad motriz, disminuyendo la transmisión inalámbrica de estos datos y consecuentemente reduciendo el consumo de energía en el nodo.

1.5 Marco Teórico

Fundamentos de las Redes de Sensores Inalámbricos RSI

Una red de sensores inalámbricos es una red de dispositivos electrónicos autónomos denominados nodos, que pueden estar distribuidos espacialmente y mediante el uso de sensores que son capaces de monitorear magnitudes físicas y transmitir esta información inalámbricamente.

Los nodos están constituidos por un pequeño procesador, diferentes tipos sensores y módulos de comunicación, generalmente son dispositivos pequeños, de bajo coste y son energizados mediante baterías o en algunas aplicaciones se energizan mediante paneles solares u otras fuentes de energías, facilitando su implementación en cualquier ambiente, convirtiendo a las RSI en una herramienta de control y monitoreo aplicable a todo tipo de medio o industria [3].

Hoy en día las RSI son utilizadas en diferentes campos y aplicaciones dentro de las cuales están: [3].

- Salud y Monitoreo de Pacientes
- Monitoreo del Medio Ambiente
- Monitoreo Agropecuario
- Monitoreo de Condición Estructural
- Monitoreo de Energía
- Monitoreo de Condición de Máquinas
- Transportación
- Monitoreo Industrial
- Monitoreo Distribuido de Temperatura
- Aplicaciones Domóticas[3]

En muchas de estas aplicaciones se requiere que los nodos sean móviles y que dispongan de una buena autonomía eléctrica; sin embargo, las actuales baterías no logran energizar los nodos por mucho tiempo cuando la aplicación requiere transmitir continuamente información del proceso, por lo que existe un continuo estudio en el campo de optimización del consumo eléctrico de las RSI.

Arquitectura de una RSI

Las RSI tienen una arquitectura de red definida en tres niveles.

En el primer nivel o nivel más bajo encontramos a todos los dispositivos de adquisición de datos denominados nodos sensores, estos dispositivos cuentan con diferentes sensores

[3] Domótica.- tecnología aplicada al control y automatización de viviendas.

según la aplicación a monitorear y adquieren periódicamente información dependiendo de la aplicación, los nodos pueden ser cooperativos transmitiendo esta información entre ellos hasta llegar a un nodo principal o pasarela.

El segundo nivel está constituido por las pasarelas o gateways que sirven para comunicar dos protocolos de comunicación diferentes, los nodos sensores tratan de concentrar sus datos en estos dispositivos, pues las pasarelas son un enlace para almacenar esta información en bases de datos.

En el nivel superior se encuentra las estaciones base, que son procesadores o computadoras que mediante un aplicativo presentan la información del sistema, permitiendo la gestión y el control de la red.

Esta estructura se puede observar en la Figura 1.1.

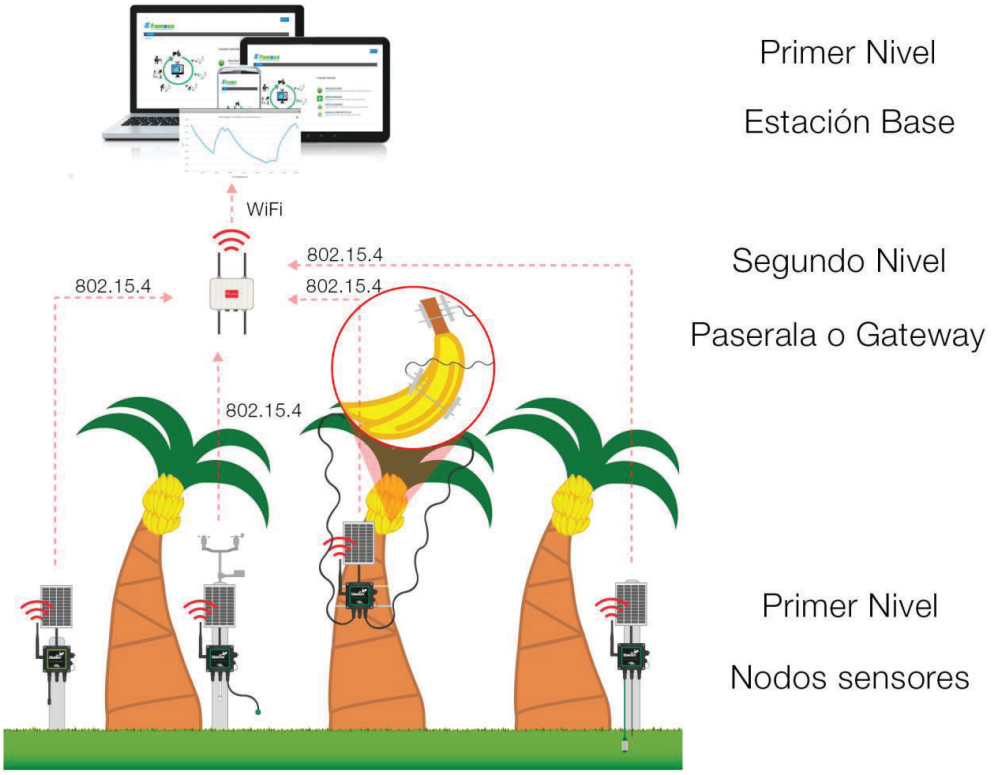


Figura 1.1 Arquitectura de una RSI [4].

Nodos Sensores

Un nodo sensor es un sistema electrónico generalmente de bajo consumo energético, constituido por un microcontrolador, que permite la interconexión de módulos de comunicación inalámbrica y módulos de sensores electrónicos que permiten la adquisición de medidas de un ambiente de trabajo. Constituyéndose en los elementos fundamentales de una RSI, su alimentación eléctrica proviene de baterías o una fuente de corriente continua según la aplicación; así, por ejemplo, se pueden alimentar con paneles solares.

El microcontrolador de los nodos es un dispositivo de bajo consumo energético, en el cual se puede cargar un sistema operativo, desarrollado principalmente en lenguaje NesC que es un dialecto del lenguaje C, lo que permite controlar el hardware que compone el nodo y programar el dispositivo según la aplicación.

La arquitectura de hardware de un nodo sensor está constituida básicamente por una interfaz de sensores, un conversor análogo digital, un microcontrolador para procesar los datos, memoria externa, una batería y radio de comunicaciones, como se representa en la Figura 1.2.

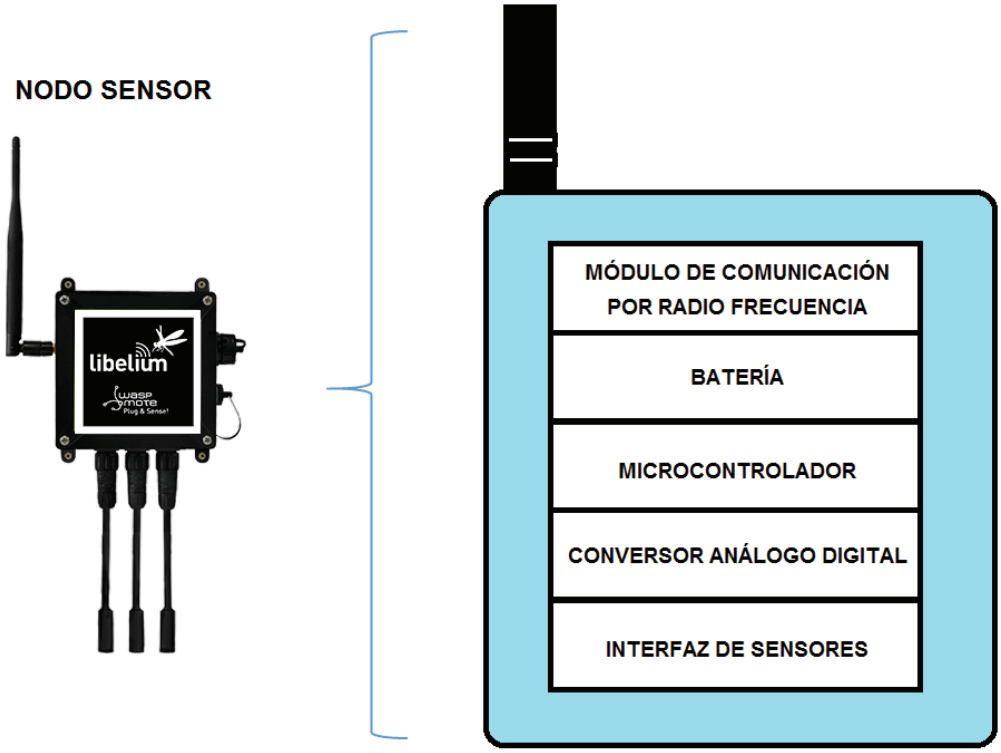


Figura 1.2 Hardware de un nodo [5].

Para mantener la comunicación entre el microcontrolador y los diferentes módulos el nodo utiliza buses de comunicación, dentro de los principales buses de comunicación encontramos USB, RS232, I2C, UART, pines de entrada o salida conectados directamente al microcontrolador, entre otros.

Dentro del sistema de comunicación inalámbrica, los nodos pueden trabajar con diferentes tipos de hardware, principalmente aquellos que usan los protocolos ZigBee, Buetooth Low Energy, IEEE 802.15.4; en redes actuales también se soportan en el protocolo IEEE 802.11x Wifi.

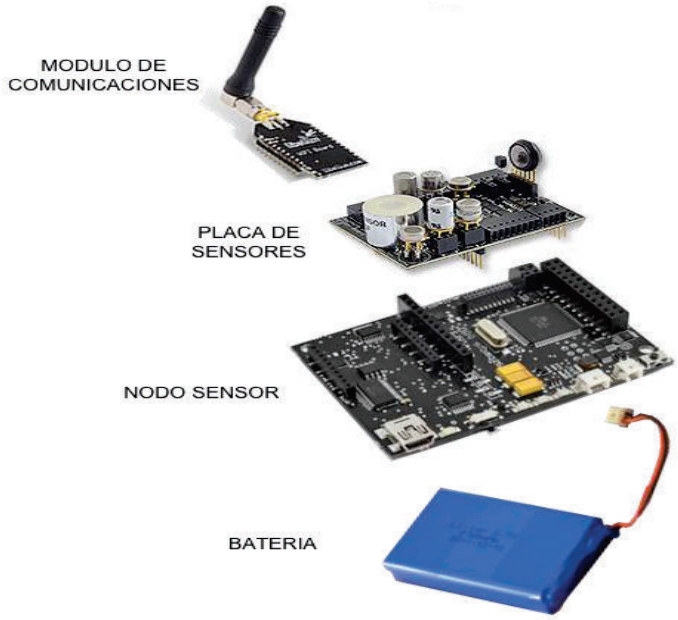


Figura 1.3 Arquitectura de hardware de un nodo real [6].

Los nodos más destacados que se puede encontrar en el mercado se presentan en la Tabla 1.1.

Tabla 1.1 Nodos comerciales

MODELO	**MPR2400CB**	**Waspmote**	**NI WSN-3202**	**Accutech BR20**
Plataforma RSI	MEMSIC	LIBELIUM	National Instruments	Schneider Electric
Aplicación	Educación	Comercial	Comercial Industrial	Automatización Industrial

Procesador	Atmel ATMega 128L, 8bit	ATmega1281	N/I	N/I
Velocidad	8 - 16 MHz	14 MHz	N/I	N/I
RAM	4 Kbytes	8 Kbytes	248 Kbytes	N/I
Estándar Inalámbrico	IEEE 802.15.4	IEEE 802.15.4	IEEE 802.15.4	IEEE 802.15.4
Rango de Frecuencia	2.4 - 2.48 GHz	2.4 - 2.48 GHz	ISM 2.4GHz	900MHz y 2.4GHz
Transmisor y Receptor	TI - CC 1000 / 2420	XBee-802.15.4-Pro	N/I	N/I
Velocidad de datos	38.4 / 250 Kbits/s	250 Kbit/s	250 Kbit/s	50 a 100 Kbps con FSK 200 Kbps con GFSK
Sensores Internos	Temperatura, luz, humedad	Temperatura, acelerómetro	N/A	N/A
Consumo Modo Activo	8mA	17mA	20mA	30mA
Consumo Modo Sleep	<15uA	55uA	N/I	N/I
Consumo RX máx.	19,7mA		13mA	<120mA
Consumo TX máx.	17,4mA	100mW	21mA	500mA
Batería	2 x AA No Especifica	4 x AA Ion litio	4 x AA Alcalinas	Con fuente externa
Voltaje	2,7 - 3,3 V	3.3V - 4.2V	3.6 V - 7.5 V	11 - 30 VDC
Dimensiones mm	58 x 32 x 7	73.5 x 51 x 13	42.31 x 85.85 x 123.98	112.5 x 124.3 x 36.8

Estos son ejemplos de nodos comerciales, para este trabajo se seleccionó el nodo Waspmote, porque su plataforma de programación es de código abierto y el nodo sensor viene integrado con un sensor acelerómetro que se requiere para el proyecto.

Pasarela o Gateway

La pasarela es un dispositivo intérprete entre una red inalámbrica de bajo consumo energético y otra red de comunicación; actúa como un coordinador de la red y, es encargado de autentificar al nodo, recolectar y almacenar los mensajes de los nodos distribuidos y abrir un puente de comunicación entre los nodos y la estación base.

La pasarela maneja dos interfaces de comunicación, siendo la principal un radio inalámbrico IEEE 802.15.4 (estándar de comunicación de RSI), de 2,4 GHz para adquirir los datos desde el nodo sensor y una segunda interfaz que puede ser Ethernet IEEE 802.3, Wifi IEEE 802.11x, USB, entre otras, que permite la comunicación con la estación base.

Un ejemplo de una pasarela comercial de Libelium, se muestra en la Figura 1.4.

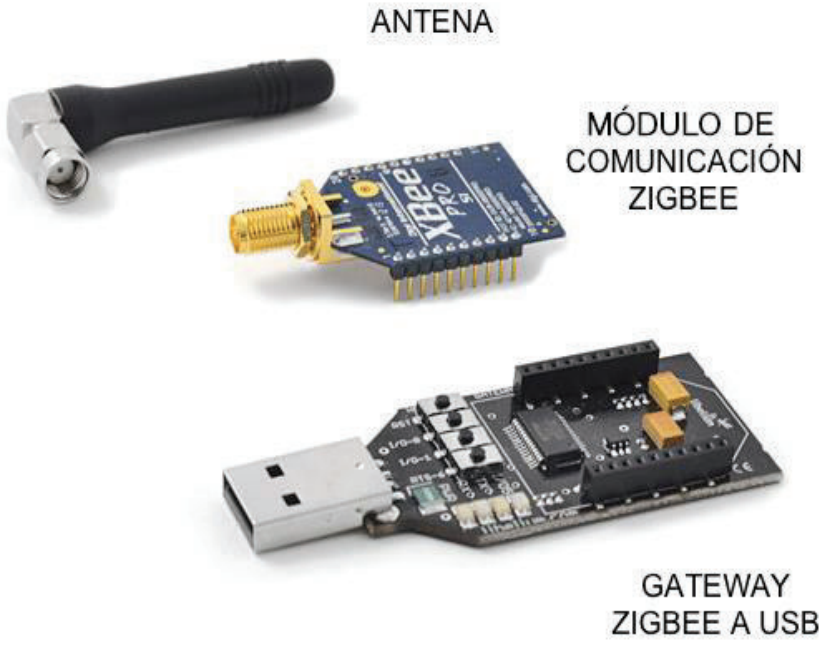


Figura 1.4 Pasarela Libelium [7].

Estación Base

La estación base es una computadora provista de un software que permita interpretar la información entregada por la pasarela, presentándola al usuario DE una manera amigable y clara, capaz de permitir el procesamiento y análisis de las medidas entregadas por la red de sensores.

La Figura 1.5. representa una estación base y la infraestructura de red de sensores con los nodos que adquieren datos, la pasarela y la estación base que presenta una interfaz gráfica amigable al usuario.

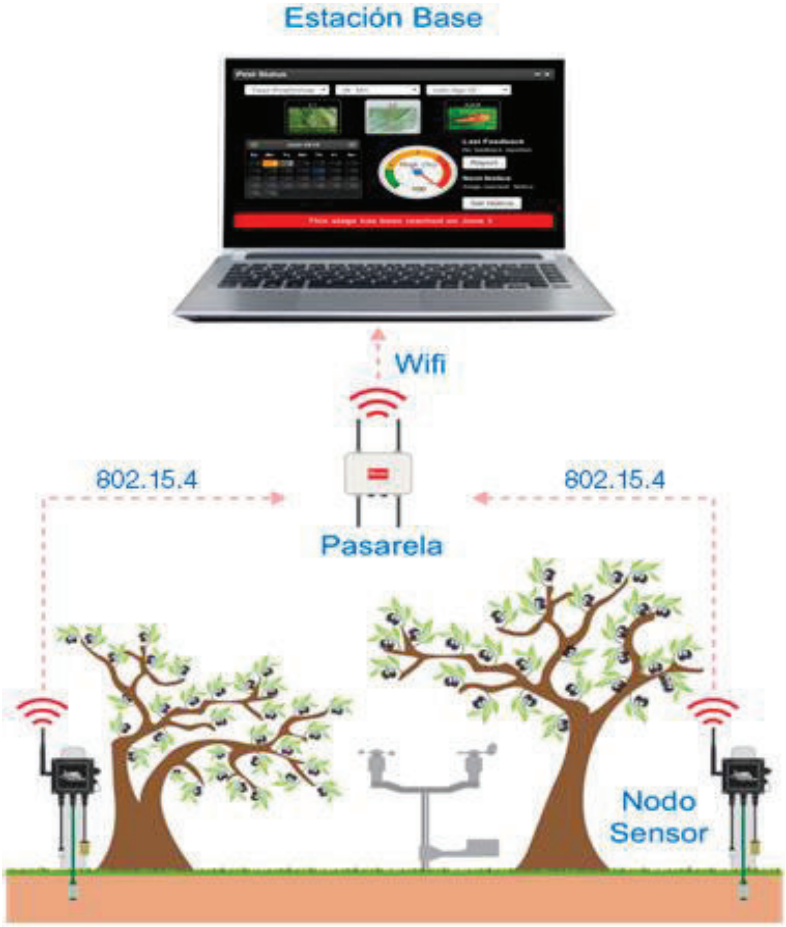


Figura 1.5 Estación base y red de sensores [4].

Protocolos de Comunicación de RSI

Para establecer la comunicación entre los nodos de una RSI, es necesario definir la forma en la que se van a comunicar los mismos, definiendo el tamaño de los paquetes, la cabecera y sus campos, niveles lógicos, modulación de la señal a transmitir, frecuencia de

trabajo, velocidad de transmisión y demás parámetros que permiten la comunicación entre un nodo emisor y un nodo receptor.

Los protocolos más frecuentes en las redes de sensores inalámbricas se presentan a continuación.

IEEE 802.11 b/g/n

El estándar IEEE 802.11 define el uso de la capa de enlace de datos y la capa física del modelo OSI, siendo ampliamente utilizado en redes inalámbricas de área local WLAN (por sus siglas en inglés Wireless Local Area Network), [8] aunque su uso no es muy difundido en redes de sensores inalámbricas por su alto consumo de energía, existen aplicaciones y fabricantes que proveen dispositivos con esta tecnología por la capacidad de manejar grandes tasas de transferencia, alta velocidad y fiabilidad; así han adaptado sus módulos de comunicación para trabajar en este estándar [6], las características que se presentan corresponden a las de un módulo WiFi PRO Libelium y son:

- Frecuencia de operación: 2.412 – 2.462 GHz USA.
- Velocidad de transferencia: 1 - 11 Mbps
- Número de canales: 1 – 11 canales.
- Soporte de topologías: punto a punto y estrella.
- Técnica de Acceso al Medio: acceso al medio por detección de múltiples portadoras CSMA-CA.

IEEE 802.15.4 LR-WPANs

Low-Rate Wireless Personal Area Networks (LR-WPANs), este protocolo define el nivel físico y el control de acceso al medio de una red inalámbrica de área personal de baja velocidad, su enfoque está en redes da bajo coste energético, permitiendo transmitir información a distancias relativamente cortas y sin la necesidad de una gran infraestructura, de ahí su implementación a una gran gama de dispositivos especialmente en las redes de sensores inalámbricas [9].

Las principales características de este estándar son:

- Operación en topología: estrella o punto a punto.
- Dirección extendida de 64 bits o dirección corta de 16 bits.
- Asignación opcional garantizada de espacios de tiempo (GTSs).

- Acceso al medio por detección de múltiples portadoras CSMA-CA.
- Bajo consumo de energía.
- Detección de energía (ED).
- Operación en varias bandas de frecuencias, 868–868.6 MHz, 902–928 MHz y 2400–2483.5 MHz.

ZIGBEE

Es un protocolo desarrollado para redes de bajo consumo energético, provee una capa de aplicación y de red; se basa en el estándar IEEE802.15.4 y su desarrollo fue logrado por la Alianza ZIGBEE que reunió a más de 300 fabricantes de semiconductores [10].

Al ser basado sobre el estándar IEEE 802.15.4, provee las mismas características de las capas de acceso y física, provee características adicionales en la capa de red, brinda seguridad y proporciona un API de usuario, así el protocolo incluye:

- El uso de bandas no licenciadas: 2.4 GHz, 900 MHz y 868 MHz.
- Soporte de topologías: punto a punto, estrella y malla.
- Latencia: 30ms.
- Cales: 16 canales para transmisión y recepción.
- Ancho de banda: 5MHz por cada canal.
- Velocidad de transferencia: 250Kbps a 2,4 Hz.
- Creación de redes de hasta 65000 dispositivos.
- Modulación: GFSK, BPSK, ASK.
- Seguridad en las conexiones: AES de 128 bits.
- Protocolos de acceso al medio: CSMA-CA.

Bluetooth

Bluetooth está basado en el estándar de comunicación IEEE 802.15.1 y su objetivo es permitir la comunicación entre dispositivos sin cable y a corto alcance con un alto nivel de seguridad y fidelidad siendo un protocolo robusto, las principales implementaciones de Bluetooth son Bluetooth Basic Rate/Enhanced Data Rate BR/EDR (Buetooth con tasa básica / Bluetooth de tasa de datos mejorada) adoptadas en las versiones 2.0 y 2.1 y

Bluetooth Low Energy BLE (Bluetooth con baja energía) adoptado en las versiones 4.0, 4.1, 4.2, 5.0 [11].

La red de comunicación es una red ad-hoc, de pocos dispositivos que pueden comunicarse entre sí, con la ayuda de un coordinador y varios dispositivos esclavo.

A continuación, se describen las principales características de la versión Low Energy:

- Frecuencia no licenciada: 2,4 GHz a 2,48 GHz, con 40 canales disponibles con separación de 2 MHz, en su versión 4.0.
- Comunicación: full dúplex con un sistema de salto adaptativo de frecuencia AFH.
- Soporta: hasta 8 dispositivos en la red.
- La velocidad de transferencia: de 1 Mbps para BLE.
- Distancia de comunicación: de 1 a 100 metros.
- Topología de red: punto a punto y multipunto.

Sistemas Operativos

El sistema operativo es un programa informático que permite a los nodos sensores controlar y usar todo el hardware con el que están formados, además permite el desarrollo de aplicaciones personalizadas.

Dentro de los sistemas operativos desarrollados para redes de sensores inalámbricos han ganado popularidad los desarrollados bajo licencias de código abierto de uso gratuito colaborativo permitiendo generar nuevos códigos que enriquecen al sistema operativo, otra característica muy común en los principales sistemas operativos es que están programados en lenguaje C/C++ y nesC (una variante de C); dentro de los más conocidos e importantes se tiene TinyOS, Contiki, Waspmote IDE de Libelium, LabVIEW [1].

TinyOS

Según la Universidad de Stanford, es un sistema operativo diseñado para sistemas inalámbricos embebidos de baja potencia y básicamente es una colección de microcontroladores de diferentes marcas y otros circuitos integrados que conforman una plataforma inalámbrica embebida [1].

La Universidad de California en Berkeley explica que es un sistema operativo de código abierto y libre, escrito en el lenguaje de programación nesC como un conjunto de tareas y

procesos que colaboran entre sí y se ejecutan línea por línea, siendo nesC una variante de C optimizado para dispositivos con poca memoria.

Este sistema operativo se complementa con su simulador TOSSIM que permite probar el software y simular el comportamiento de los nodos sensores.

Contiki

Es un sistema operativo de redes de sensores inalámbricos, su código es abierto por lo que existe colaboración y generación de nuevo código; fue desarrollado en lenguaje C por Adam Dunkels y en la actualidad su principal enfoque es el Internet de las Cosas, por lo que una de las fortalezas que ofrece es el desarrollo de librerías de comunicación y protocolos de enrutamiento, así puede trabajar con protocolos de Internet como IPv4 e IPv6, gracias a sus estándares para redes de bajo consumo energético entre los que se tienen 6lowpan, RPL y CoAP. Otra característica importante es que cuenta con un simulador de redes denominado Cooja, que permite emular una red de sensores antes de su despliegue en hardware [1].

Waspmote IDE

Plataforma desarrollada por la empresa Libelium bajo licencia de código abierto para el desarrollo de código en lenguaje C++ de sus nodos sensores y pasarelas, presentando una interfaz sencilla tipo arduino, que permite añadir código ya sea nuevo o de soporte de ejemplos que provee la empresa para el desarrollo de aplicaciones; la programación de los sensores es realizada mediante el puerto serial y adicional permite abrir una consola que abre y captura los paquetes recibidos por la pasarela. La versión actual del API de desarrollo es Waspmote Pro IDE - v04 y se puede descargar para sistemas operativos Linux, IOS y Windows, proveyendo librerías de comunicación, encriptación, interrupciones, lectura de sensores y modos de operación de sus productos [7].

La estructura de programación que brinda Waspmote IDE está compuesta por dos funciones básicas que son: void setup() y void loop(), la primera función se ejecuta una sola vez cuando el código es inicializado o el nodo es reseteado y en ella se recomienda incluir parámetros de configuración inicial.

La segunda función loop(), se ejecuta en un bloque hasta el infinito, y ella es usada para medir, procesar, enviar información y programar al nodo a los modos de bajo consumo de energía.

La estructura típica de un programa incluirá:

1. Bibliotecas de programas que serán utilizados.
2. Definiciones.
3. Declaraciones de variables.
4. Función void setup().
5. Función void loop().

LabVIEW WSN

Es una plataforma de programación gráfica de redes de sensores inalámbricas, su licencia es de pago y fue desarrollada por la empresa National Instruments, para su gama de productos, optimizando y facilitando el desarrollo de aplicaciones para los nodos y su interfaz gráfica. Esta novedosa plataforma permite el desarrollo de interfaces hombre máquina con simples herramientas embebidas en su software, permitiendo una implementación rápida de un centro de monitoreo de la red de sensores [4].

Topología de Red para RSI

Es el mapa físico o lógico de la red de sensores, e indica en la manera como intercambiarán los datos entre los nodos de la RSI, así, típicamente se pueden organizar en cuatro tipos de topología, así: topología punto a punto, topología estrella, topología árbol, topología en malla [4].

A continuación, se describen cada una de ellas.

Topología Punto a Punto

El nodo sensor se comunica directamente a la pasarela, su comunicación es simple y no se requiere de complicados protocolos de enrutamiento para transmitir la información, deben estar en el mismo canal y el nodo debe conocer el identificador de hardware de la pasarela.

Figura 1.6 Representación de topología punto a punto [4].

Topología Estrella

Cada nodo sensor de la red se conecta directamente a la pasarela y se implementan protocolos de comunicación como, por ejemplo, escucha del canal para transmitir información de manera efectiva para evitar la pérdida de paquetes y, consecuentemente, un mayor consumo energético al retransmitir un paquete.

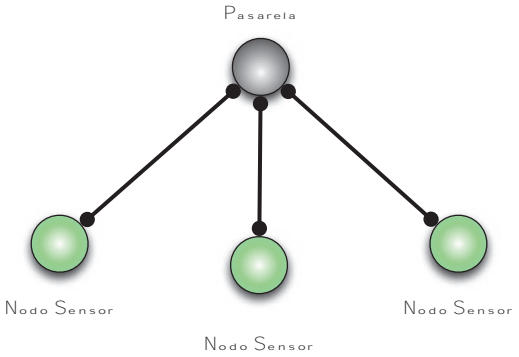


Figura 1.7 Representación de topología estrella [4].

Topología de Árbol

Cada nodo sensor se comunica a un nodo de mayor jerarquía en el árbol y después a la pasarela, los datos son ruteados desde el nodo sensor de menor jerarquía en el árbol hasta la pasarela.

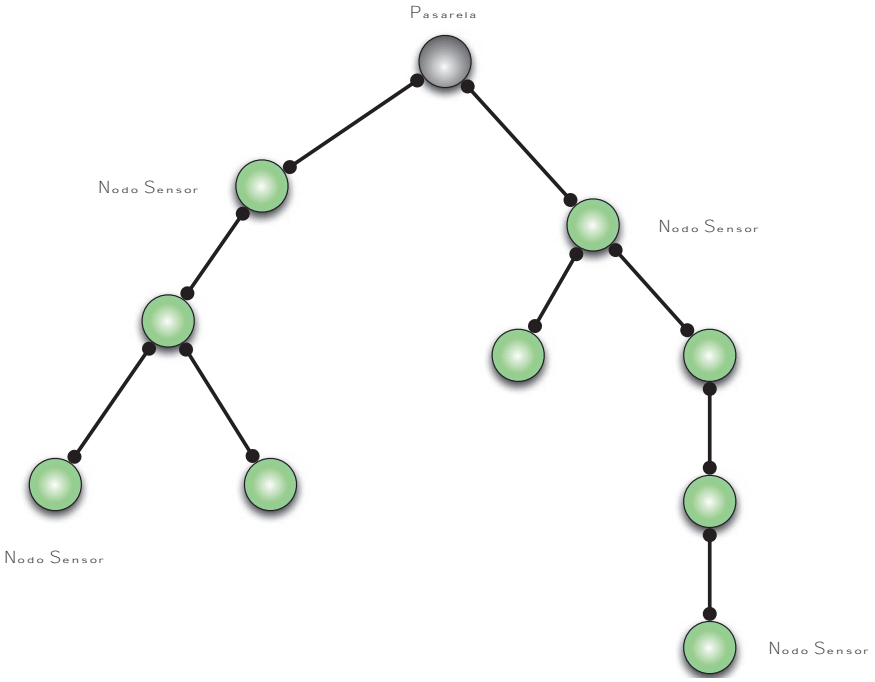


Figura 1.8 Representación de topología árbol [4].

Topología de Malla

Presenta una arquitectura confiable; la característica principal es que varios nodos se comunican entre ellos formando una malla con varios caminos y en caso de que algún nodo se perdiera por algún motivo, se generan nuevos caminos hasta llegar a la pasarela.

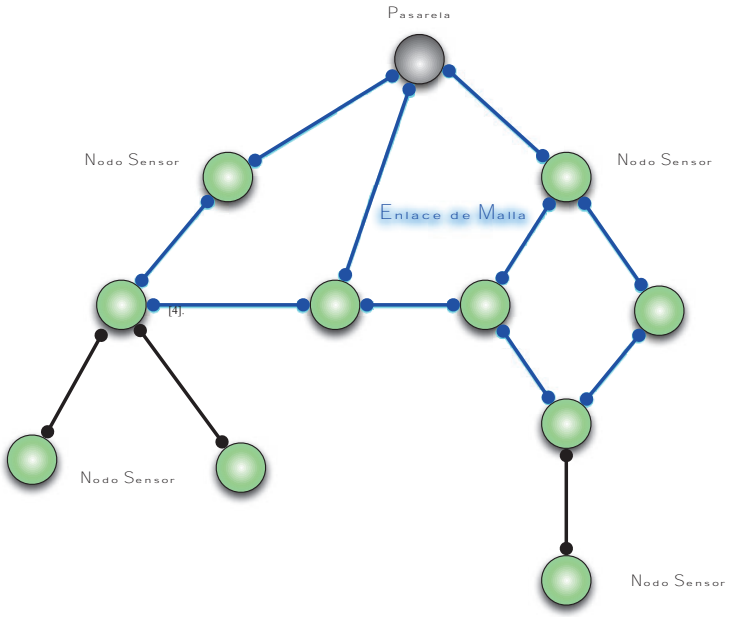


Figura 1.9 Representación de la topología malla [4].

Consumo y Optimización de Energía en RSI

Los nodos de las RSI, generalmente, son energizados por baterías, lo que hace que su tiempo de vida finito, pues depende del tiempo que se demore en descargarse la batería, por lo que cuando se diseña una red de sensores es importante el analizar la aplicación para la cual se utilizará la red, para realizar mecanismos que contribuyan con el ahorro energético y aumente el tiempo de vida de los nodos en la red [12].

El consumo eléctrico de un nodo sensor, se encuentra dividido de la siguiente manera:

El módulo de comunicación es el primer consumidor de energía, pudiendo aumentar el consumo según la distancia a la que necesita transmitir el nodo, el protocolo de enrutamiento de que se utilice y el tiempo que se demore en trasmitir los datos.

El consumo de los sensores está en segundo lugar, pero esto, generalmente, depende de la aplicación y el sensor con que se trabaje.

Finalmente, la circuitería del nodo es la de menor consumo eléctrico al estar conformado por un microcontrolador de bajo consumo energético, estos disponen de tres modos de operación que son activo, dormido e hibernación, que según el microcontrolador que forme el nodo y su modo de operación, pueden consumir desde una decena de mili amperios en modo activo hasta unos pocos micro amperios en modo hibernación.

Para optimizar la energía consumida por los nodos existen algunas estrategias entre las que se mencionan las siguientes:

- Reducir la distancia de transmisión, para usar menos potencia en el transmisor.
- Realizar una programación eficiente de las líneas de código, usando los modos de energía del microcontrolador.
- Uso y desarrollo de nuevos protocolos de enrutamiento.
- Desarrollo y mejoras en el hardware para disminuir el consumo energético [11].

Muchas veces se puede combinar estas estrategias para tener un mejor resultado, el objetivo al momento de diseñar una RSI es optimizar del consumo de energía en los nodos, maximizando el tiempo de vida de la red y al mismo tiempo satisfaciendo las necesidades de la aplicación para la cual la RSI está siendo usada.

Fisiología Bovina

Para comprender como se puede aplicar las redes de sensores, al campo ganadero se debe estudiar brevemente la fisiología bovina, sus etapas reproductivas, las formas de detectar el momento de la inseminación y la caracterización de sus desplazamientos.

Ciclo Estral de un Bovino

El ciclo estral es un proceso regular hormonal de 21 días, con una variación de 18 – 24 días [13].

Las fases de un clico estral normal de 21 días son presentadas en la Tabla 1.2.

Tabla 1.2 Fases del ciclo estral bovino

Fases	Días
PROESTRO	2
ESTRO	1
METAESTRO	2
DIESTRO	16

Proestro

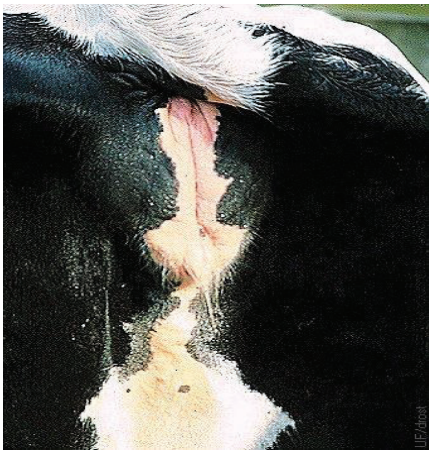
Se presentan las hormonas GnRH-FSH-LH (hormona liberadora de gonadotropina GnRH, hormona foliculoestimulante FSH, hormona liberadora de la hormona luteinizante LHRH, hormona luteinizante LH) lo que conduce al inicio de la descarga de estrógenos, los síntomas externos de esta etapa son:

La hembra olfatea a otras vacas, arrima la quijada sobre el tren posterior de otras vacas y las intenta montar, existe edematización de la vulva y secreción de un moco gris transparente, sube la temperatura corporal, disminuye la producción de leche efecto del estrés y alta pérdida de energía [15].

Estro o Celo

La hormona dominante son los estrógenos que actúan sobre el sistema nervioso central determinando características sexuales del mismo, que permiten la aceptación del macho o de otras vacas dejándose montar. Otras características de esta fase son: edema vulvar, movimiento de la cola, contracción rítmica del ano, disminución del apetito y la producción láctea, la vaca muge constantemente y aparecimiento de costras en los flancos del anca. En esta etapa se debe realizar la inseminación y el sistema de RSI debe permitir la determinación de este día [15].

El signo más visible que es la monta y tiene una duración de 4 a 6 segundos, con un promedio de 1½ montas por hora, toda esta etapa tiene un pico de calor con una duración de 6 a 8 horas. La imagen a continuación ejemplifica estos síntomas.



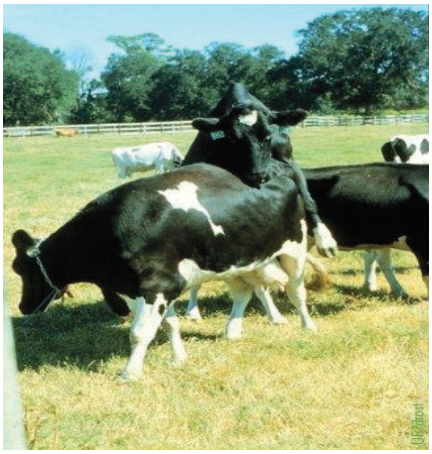


Figura 1.10 Síntomas de estro, edema vulvar y monta de vaca. Fotografías tomadas de "The Drost Project" [15].

Metaestro

Esta fase viene dada por el pico preovulatorio de LH que producirá la ovulación y se iniciará la secreción de progesterona; en esta etapa hay tranquilidad sexual y en pocas hembras un flujo sanguinolento algo obscuro [15].

Diestro

Etapa en la que la progesterona domina; de darse la fecundación, entre los 16 y 18 días, el feto y la placenta habrán producido suficientes señales químicas que inhibe el efecto de la Prostaglandina, caso contrario no se inhibe el efecto de la misma y continua un nuevo ciclo; en esta fase no existen síntomas externos y la vaca pastorea normalmente.

Si se desarrolla la preñez no se volverán a detectar síntomas sexuales y se puede diagnosticar la preñez de la vaca de 45 a 60 días después de la inseminación, mediante la palpación del útero, o con ultrasonido [15].

Detección del Estro en Bovinos

Los métodos de detección del estro en bovinos es un campo ampliamente estudiado en el campo veterinario y la correcta determinación del mismo implica grandes tazas de inseminación, un control de intervalos de partos y lactancias, una alta tasa de preñez y consecuentemente un ahorro económico en la producción [13].

Existen dos métodos de determinación del estro:

1. Manual, observando y tomando medidas manuales de cambios físicos, producto de los cambios hormonales en las vacas, así como aumento de la temperatura corporal, cambio de resistencia de la mucosa vaginal, observación de monta entre vacas, olfateo, la quijada apoyada en el anca de otras vacas, o la experiencia del granjero, siendo este método un trabajo intenso que consume mucho tiempo y su precisión es del 50% [16].
2. Monitoreo automático, mediante el uso de instrumentos o sensores electrónicos capaces de reconocer los comportamientos de los bovinos, sistemas de posicionamiento global GPS que muestran el desplazamiento de los animales, podómetros en forma de collares para recolectar la información del número de pasos que de las vacas y acelerómetros de 3 ejes capaces de muestrear el comportamiento de detección del estro.

Características de la Fisiología Reproductiva Bovina

Las características que se presentan a continuación son similares para las diferentes razas bovinas, pero van dirigidas principalmente a la raza Holstein, que es la raza predominante en el Ecuador.

- Las crías nacen con un peso corporal promedio de 45 kg, a los 12 meses pueden alcanzar su pubertad, etapa en la cual la hembra presenta el primer síntoma de estro, entre los 14 y 15 meses de edad llega a unos 350 kg promedio y una altura de 1,33 cm, siendo esta edad óptima para realizar la primera inseminación.
- Después del parto una vaca tiene un periodo de recuperación del aparato reproductor femenino que dura de 45 – 50 días, en los que no existe actividad sexual.
- Posterior al periodo de recuperación se presenta el primer ciclo estral en 21 días, sin embargo, se recomienda no inseminar al animal por estar todavía recuperando su condición corporal.
- El ciclo estral es un proceso regular hormonal de 21 días, con una variación de 18 – 24 días.
- El porcentaje de efectividad en la determinación del estro según [17]. se muestra en la Tabla 1.3.

Tabla 1.3 Porcentaje de efectividad de determinación del estro.

PORCENTAJE DE VACAS DETECTADAS EN ESTRO			
Número de observaciones	**Tiempo de observación (minutos)**		
	5	**10**	**20**
1	26 %	52 %	63 %
2	36 %	72 %	86 %
3	39 %	79 %	95 %
4	49 %	82 %	98 %

Las observaciones mencionadas en la tabla corresponden a observaciones aleatorias en el potrero donde se encuentran las vacas, se recomienda observar pasado las 08:00 am cada 2 horas, durante el tiempo presentado en la tabla. Así se menciona que, si se realiza la observación de un animal en celo por cuatro ocasiones durante el día con una duración de 20 minutos cada una, la probabilidad de determinar el celo del animal es del 98%.

- La mejor hora de detección del estro es temprano en la mañana, medio día y en la tarde antes del anochecer; si el clima es caliente, los mejores momentos son

temprano en la mañana y antes del anochecer; si el clima es frio, al medio día es el mejor momento.

- La mayor actividad se puede observar en los campos de pasto, antes que en el concreto.
- La taza de concepción aumenta si la vaca es inseminada de 4 a 14 horas después de la observación del estro, aplicando una regla denominada AM – PM, así un animal observado en la mañana debe ser inseminado en la tarde y al contrario un animal observado en la tarde debe ser inseminado en la mañana.

Características Normales del Comportamiento Bovino

Las siguientes son características normales que presentan las vacas, pero aportan información de la actividad motriz normal de un bovino.

Alimentándose

Las vacas pueden encontrarse en el pastizal paradas alimentándose, por lo general estarán quietas y moverán la cabeza, usan los incisivos y su fuerza para arrancar el pasto del suelo.

Bebiendo

Es una actividad muy pausada, generalmente están paradas con la cabeza inclinada para alcanzar el agua, paran de beber, momentáneamente, para respirar.

Rumiando

Generalmente los bovinos están rumiando acostados, pero pueden rumear de pie o acostados y su actividad motriz es mínima.

De pie

Pueden mantenerse de pie sin hacer ningún movimiento.

Caminando

Esta actividad es típica en horas de la mañana y tarde para dirigirse al ordeño y la actividad será moderada, también se presenta cuando se dirigen a beber y comer, es una actividad moderada.

Durmiendo

Las vacas se acuestan en horas de la noche y su actividad generalmente es nula, solo rumeando los alimentos ingeridos por la tarde y, eventualmente se levantan a beber agua.

Estudios Relacionados

La industria pecuaria es una de las industrias con mayor importancia a nivel mundial, por ser la proveedora de la carne que consumimos, así, esta industria innova e investiga constantemente nuevas tecnologías que ayuden a producir más, disminuir costos operativos y mejorar la calidad de vida de los productores.

En el sector ganadero, el campo de acción de las investigación está enfocado al desarrollo de sistemas automáticos de alimentación, gestión y producción de leche y carne, por ello el control reproductivo es parte fundamental de estos estudios, así, existen investigaciones donde se usan sensores de telemetría para monitorear desplazamientos, pH del rumen y temperatura en bovinos, podómetros activos RFID (siglas de Radio Frequency Identification, en español identificación por radiofrecuencia) para el control reproductivo de bovinos.

Un primer ejemplo es el estudio [17], donde presentan a las redes de sensores como una técnica eficiente para obtener datos fisiológicos de bovinos; para ello se implementa un nodo sensor para monitorear la actividad eléctrica cerebral (EEC) de seis novillas al aire libre, este nodo transmitía la información a una estación radio base a una frecuencia de 433 MHz. El nodo sensor fue ubicado en una bolsa impermeable con un cabestro en el cuello de las novillas para la toma de muestras.

El estudio permitió obtener los datos de los desplazamientos de las novillas en la granja y desarrollar un simulador de su desplazamiento, para que futuras investigaciones usen estos datos característicos de los bovinos.

También se analizó la influencia de los factores ambientales o externos en el comportamiento bovino, validando la ausencia de estrés en la toma de muestras al ser un método no invasivo y sin la presencia humana directa, también se comprobó que el estrés térmico por calor influencia en sus hábitos de pastaje y rumiación.

Este estudio se realizó por una semana en estas novillas y uno de los factores críticos que se analizaron fue el consumo de energía del nodo, debido a la descarga de las baterías solo se pudo monitorear continuamente a estas novillas siete días, al implementar únicamente una técnica de acceso al medio en la transmisión de datos como estrategia de ahorro de energía.

Este estudio es importante debido a que, valida los comportamientos motrices de los bovinos en campo abierto, con los que se encuentran en la literatura de la fisiología bovina

y permite caracterizar la actividad motriz de los bovinos, para el desarrollo del algoritmo adaptativo a la motricidad de la vaca que se planteó en el proyecto de investigación.

La investigación [18] plantea la monitorización autónoma de animales de granja mediante el uso de redes de sensores, con el objetivo de minimizar los costos de mano de obra en las granjas.

Se plantea el uso de las RSI para la vigilancia de la fisiología o comportamiento animal, permitiendo al granjero una toma de decisiones temprana ante un evento anormal. En la investigación se muestra una tabla de sistemas de monitorización animal de uso comercial, que afirma que dichos sistemas no han tenido una gran acogida por el sector ganadero europeo, y plantea el uso de redes de sensores como elementos de menor coste, mayor adquisición de datos y confiabilidad.

Para esa investigación se desarrolló una RSI con un collar compuesto de un nodo sensor en el cuello de un animal, que captura, procesa y transmite datos a una estación base encargada de la presentación de los mismos. La comunicación entre el collar y la estación base es mediante el estándar 802.15.4, trabajando en la frecuencia de 2.4 GHz. e implementando una estrategia de ahorro de energía al enviar al nodo a un modo dormido en ciertos periodos de tiempo para alargar el tiempo de vida de las baterías.

Así, el estudio se enmarcó en la movilidad del animal, capturando datos con un GPS (sistema de geo posicionamiento global) de las diferentes ubicaciones del animal durante el día y estos datos son enviados al ganadero para su análisis.

En el estudio se estableció la transmisión de datos únicamente dos veces al día, cuando el animal se encuentra en el establo para ser ordeñado (succión de leche de su glándula mamaria), limitando el tiempo de transmisión únicamente a 20 segundos al día, por ser la transmisión de datos, uno de los procesos de mayor gasto energético, con estas limitantes el nodo puede alcanzar un tiempo de vida no mayor a 3 años.

Como se analiza en esto casos, los investigadores plantean dos estrategias para reducir el consumo de energía; en la primera, permitió extender la vida del nodo durante siete días, pero el nodo sensor tomaba y enviaba constantemente información, razón por la cual la descarga de la batería fue muy rápida; en la segunda estrategia, el muestreo era mínimo y la transmisión de información está limitada a la mañana y tarde cuando las vacas se dirigen al establo, por lo que no se dispone de información en tiempo real. Aunque estos trabajos no se enfocan en determinar el día del estro, son un ejemplo de aplicaciones de RSI en bovinos.

La estrategia que se plantea en este proyecto de investigación es realizar un algoritmo adaptativo, en el cual la adquisición de datos este basada en la actividad motriz de una vaca lechera, por lo que se tomarán muestras de información en intervalos de tiempo que garanticen una correcta determinación del estro y mediante un procesamiento previo de los datos, a través de comparaciones del nivel de actividad, se tomará la decisión de transmitir esta información solo cuando sea útil para el administrador ganadero. Con esto se prolongará el tiempo de descarga de las baterías y se tendrá información en tiempo real cuando la misma sea de verdadero interés.

2 METODOLOGÍA

2.1 Diseño e Implementación del Algoritmo Adaptativo a la Fisiología Bovina

El siguiente capítulo muestra la metodología utilizada para desarrollar el algoritmo adaptativo a la actividad motriz de una vaca, así también describe el diseño del prototipo de la red de sensores que se usó para probar el algoritmo.

2.2 Modelando la Actividad Motriz

Para el desarrollo del algoritmo, primero fue necesario realizar una caracterización de los síntomas físicos, que presenta una vaca en su día óptimo para ser fecundada, así basados en el estudio teórico, podemos afirmar que los principales síntomas físicos que pueden ser medidos son:

- El día del estro, la vaca puede encontrarse muy activa, la presencia de hormonas, hacen que sea molestada por el resto de animales, así también puede montar a otras vacas elevando su cuerpo más de 30°, esta aceleración o actividad motriz de la vaca puede ser medida con un acelerómetro.
- La temperatura corporal tiene un incremento de 0.5 °C, esta magnitud se puede medirla con un sensor de temperatura, pero su medida debe ser interna pues al poseer una gran capa de piel y pelo, la medida externa de la temperatura de la vaca no es exacta.
- Otro síntoma físico que se puedo observar y medir es la edematización de la vulva, sin embargo, este método es invasivo y complicado de ser medido con un sensor comercial.
- El cambio de pH de la vulva es otro síntoma que puede ser medido para la determinación del estro de la vaca y se podría medir con un pH-metro, pero este método es invasivo al necesitar insertar el sensor en el interior de la vagina del animal.

Los síntomas de temperatura y pH pueden ser medidos con la ayuda de sensores, pero su medida es invasiva y estresan al animal, razón por la que no se consideraron en este estudio.

Así, la única magnitud que se mide sin comprometer la integridad del animal es la aceleración dinámica o actividad motriz de la vaca, mediante el uso de un acelerómetro.

2.3 Características de la Actividad Motriz

Para el desarrollo de este estudio, es necesario conocer la actividad motriz de la vaca en los diferentes momentos de su movilidad, así se tiene:

- En la etapa de proestro, las vacas sirven como detectoras de celo montando a la vaca que presenta este síntoma, su actividad motora aumenta, pero es mínima, sin embargo, sirve como indicativo porque la vaca pronto va a entrar en estro.
- En la etapa de estro, la vaca se presenta más inquieta, es montada por otras vacas, la vaca que monta se eleva del piso con un ángulo superior a 30°, formado respecto al eje horizontal con apertura hacia arriba. La vaca montada sufre fuertes choques de otras vacas, experimenta aproximadamente 9 montas en un lapso de 6 horas con una duración de 4 a 6 segundos, también se puede escuchar bramidos del animal, su actividad motriz es intensa, disminuye su apetito. La Figura 2.1 muestra la monta de una vaca.

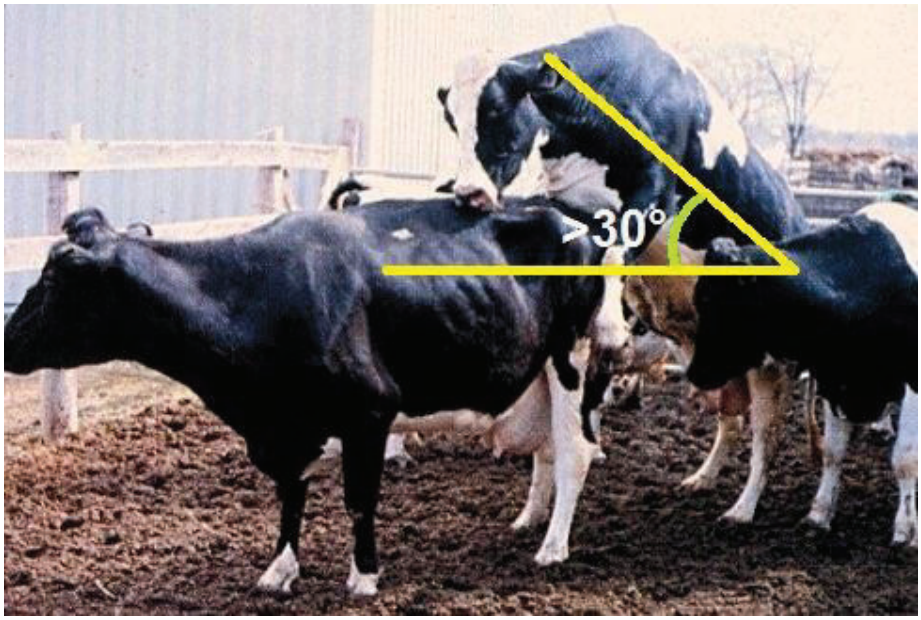


Figura 2.1 Monta de vaca en estro. Fotografía adaptada de “The Drost Project” [15].

- En la etapa de pastoreo, la vaca se encuentra comiendo el pasto, presenta un movimiento constante de la cabeza y el cuello agachando el mismo con un ángulo mayor a 30°, respecto al eje horizontal con apertura al suelo; en esta etapa arrancan la hierba del suelo para luego ingerirla. La Figura 2.2 muestra el ángulo formado cuando una vaca está comiendo.

Figura 2.2 Vacas comiendo pasto.

- En la etapa de rumen, las vacas se encuentran tranquilas y por lo general están acostadas, pocas veces están de pie, y regurgitan el pasto almacenado en su estómago para rumear moliendo la hierba con los molares para hacer partículas más pequeñas.
- Bebiendo agua, el animal está muy tranquilo, de pie, agachando su cabeza ante el bebedero, en este caso el ángulo formado por la cabeza y el cuello del animal va a depender de la altura a la que se encuentre el abrevadero, sin embargo, esta actividad es muy pasiva y no representa un gran cambio en la actividad motriz.
- Caminando, esta actividad puede ser variada y puede presentarse una caminata muy lenta cuando se dirige a tomar agua o en la mañana y en la tarde cuando se dirige al ordeño, sin embargo, en ocasiones puede correr si es asustada o estresada por algún factor externo.
- Durmiendo, en la noche las vacas duermen y en algunas ocasiones se levantan a tomar agua. La actividad motriz es nula.

De acuerdo con estos estados, en este estudio se decidió valorar la actividad motriz dividiendo la misma en cuatro categorías según su intensidad, valoradas con números del 1 al 4 y así asignar estos valores a las diferentes actividades del bovino como se muestra a continuación:

Actividad baja = 1

Actividad media = 2

Actividad media alta = 3

Actividad alta = 4

Así se clasifico la actividad motriz de los bovinos en la tabla 2.1.

Tabla 2.1 Actividad motriz de un bovino

ETAPA	Nivel de Intensidad
Desplazamientos a la sala de ordeño	2
Ordeño de la mañana	2
Desplazamiento a potreros	3
Comida	2
Rumen	1
Beber agua	1
Dormir	1
Proestro	3
Estro	4

Al observar la tabla se puede afirmar que al determinar la actividad alta de la vaca podríamos asumir que se encuentra en estro, pero esta decisión la deberá tomar el veterinario o el encargado de la ganadería.

Las características de un bovino en estro se sintetizan en la tabla 2.2.

Tabla 2.2 Actividad motriz en la etapa de estro

ESTRO	
Actividad Motriz	Intensa Nivel 4 El bovino es montado Gran actividad en el cuello al olfatear a otros animales Choca su cabeza constantemente con otros animales Puede montar a otras vacas Corre por estrés entre animales

	Impactos fuertes al rato de la monta
Período de calor	6 - 8 horas
Tiempo de monta	4 – 6 seg.
Número de montas	1½ por hora
Veces al mes	1 cada 21 días

2.4 Base para el Diseño del Algoritmo

Para el desarrollo del algoritmo se consideró los siguientes conceptos:

- El ciclo estral de la vaca es un periodo regular de 21 días y el día de interés es el día de estro de la vaca, el resto del tiempo el bovino se encuentra en una etapa de baja actividad motriz.
- Si se conecta un nodo sensor que envié información permanente de la actividad motriz de la vaca, esta información es similar o repetida durante los días previos al estro (aproximadamente 20 días), dado que las hormonas presentes no generan una gran actividad motriz. Mientras que el día de estro encontraremos una actividad motriz muy alta.
- Como en la naturaleza nada es exacto, no se puede asumir que el ciclo estral durará los 21 días exactos y mucho menos en una vaca que acaba de parir, pues su primer ciclo empezará de 40 a 45 días después del parto porque su sistema reproductivo necesita recuperarse y en este primer ciclo la vaca no debe ser inseminada.
- Si se evalúa el ángulo formado por el cuello de una vaca cuando se encuentra comiendo se puede descartar esta actividad motriz.
- Si un nodo sensor se encuentra en modo de dormido la mayor parte del tiempo donde la actividad motriz es muy baja, se ahorrará energía y se aumenta el tiempo de vida del nodo en la red.
- La transmisión de datos de la actividad motriz debe realizarse, siempre y cuando esta información aporte información útil al administrador ganadero.

También se debe tomar en cuenta las siguientes consideraciones importantes para establecer el desarrollo de la estrategia de muestreo:

- Si se toman varias muestras de la actividad motriz de la vaca, con un cierto intervalo de tiempo y se observa que estas medidas son de baja intensidad motriz, se puede asumir que el animal no se encuentra en su día de estro y esta información no

necesita ser transmitida por el nodo sensor; mientras que si estas medidas indican una actividad motriz alta debemos informar al sistema de visualización, transmitiendo estos datos de manera inalámbrica, para que el administrador de la ganadería tome una decisión.

- Cuando la actividad motriz es baja, no es necesario adquirir los datos constantemente, pues se conoce que estos valores no cambiaran significativamente. Así se puede programar al nodo sensor en modo dormido, para que el consumo de energía sea mínimo.
- Si se detecta una actividad motriz alta, se puede reducir los intervalos entre muestras y aumentar el número de estas, para que el administrador ganadero tenga suficiente información e identifique si el evento es en realidad el de una vaca en estro. En este modo se puede analizar estas muestras de actividad alta y transmitirlas agrupadas.
- Una forma de análisis de detección de baja actividad es evaluar el ángulo de inclinación de la cabeza con respecto al eje horizontal; si la inclinación del cuello del animal con respecto a este eje es mayor a 30° con apertura hacia el suelo, es un indicativo que la vaca está comiendo y esta actividad no representa un estado de estro.
- Con estas consideraciones, se garantiza que solamente lecturas con alto índice de actividad motriz sean generadas, enviadas y analizadas, manteniendo al nodo sensor la mayoría de tiempo en modo dormido.

2.5 Algoritmo Adaptativo

Con las consideraciones que se asumen, se desarrolló un algoritmo adaptativo que se base en la actividad motriz de la vaca y mediante una estrategia de muestreo permita determinar cuando la vaca está en estro; procurando que el nodo sensor permanezca en modo dormido la mayor parte del tiempo del ciclo estral y que presente información únicamente cuando la actividad motriz sea alta.

A continuación, se presenta el algoritmo adaptativo de monitoreo reproductivo bovino basado en la actividad motriz:

Paso 1.- Inicialización del nodo sensor, se activa el nodo y se configuran los umbrales de aceleración en los diferentes ejes para realizar las comparaciones (aceleración alta en X = 800, aceleración baja en X = -800, aceleración alta en Y = 800, aceleración baja en Y = -800, aceleración alta en Z = 1200, aceleración baja en Z = -1200), el tiempo inicial para

que el nodo duerma se configura en 1 minuto y el tiempo de muestreo inicial se configura en 4 segundos.

Paso 2.- Se deben capturar 15 muestras de la actividad motriz durante un 1 minuto.

Paso 3.- Comparar cada uno de los valores, con los valores de los umbrales.

Paso 4.- Si existen valores que son identificados como actividad alta, evaluar el ángulo formado por el cuello del animal, si éste es mayor a 30°, se trata de una vaca comiendo y se descarta esta actividad y se procede al paso 5; caso contrario se realiza el paso 8.

Paso 5.- Si la evaluación indica que la actividad es baja se procede con el paso 6.

Paso 6.- Se incrementa 1 minuto al tiempo de dormir, este valor aumentará hasta un máximo de 5 minutos; luego este tiempo quedará establecido como fijo, hasta que un evento de actividad alta cambie este valor.

Paso 7.- Enviar a dormir el nodo, transcurrido el tiempo de dormir se ejecuta el paso 3.

Paso 8.- Si la actividad motriz de la vaca fue alta, se procede a transmitir inalámbricamente estos datos y continuamos con el paso 9.

Paso 9.- Se incrementa en 1 minuto el periodo de muestreo de actividad alta, y se toma 15 muestras por cada minuto, hasta un tiempo total de 5 minutos. Después de cada período de muestreo, se trasmitirme inalámbricamente los datos muestreados y se regresa al paso 4.

Cuando los datos se transmitan a la pasarela, esta información los presentará a través de la aplicación de computadora, y la persona encargada de controlar esta actividad, tome las medidas del caso.

2.6 Diagrama de Flujo del Algoritmo Adaptado a la Actividad Motriz de la Vaca

A continuación, se presenta el diagrama de flujo de la estructura del algoritmo diseñado, para la reducción de consumo energético, en una aplicación de monitoreo bovino basado en la actividad motriz de la vaca.

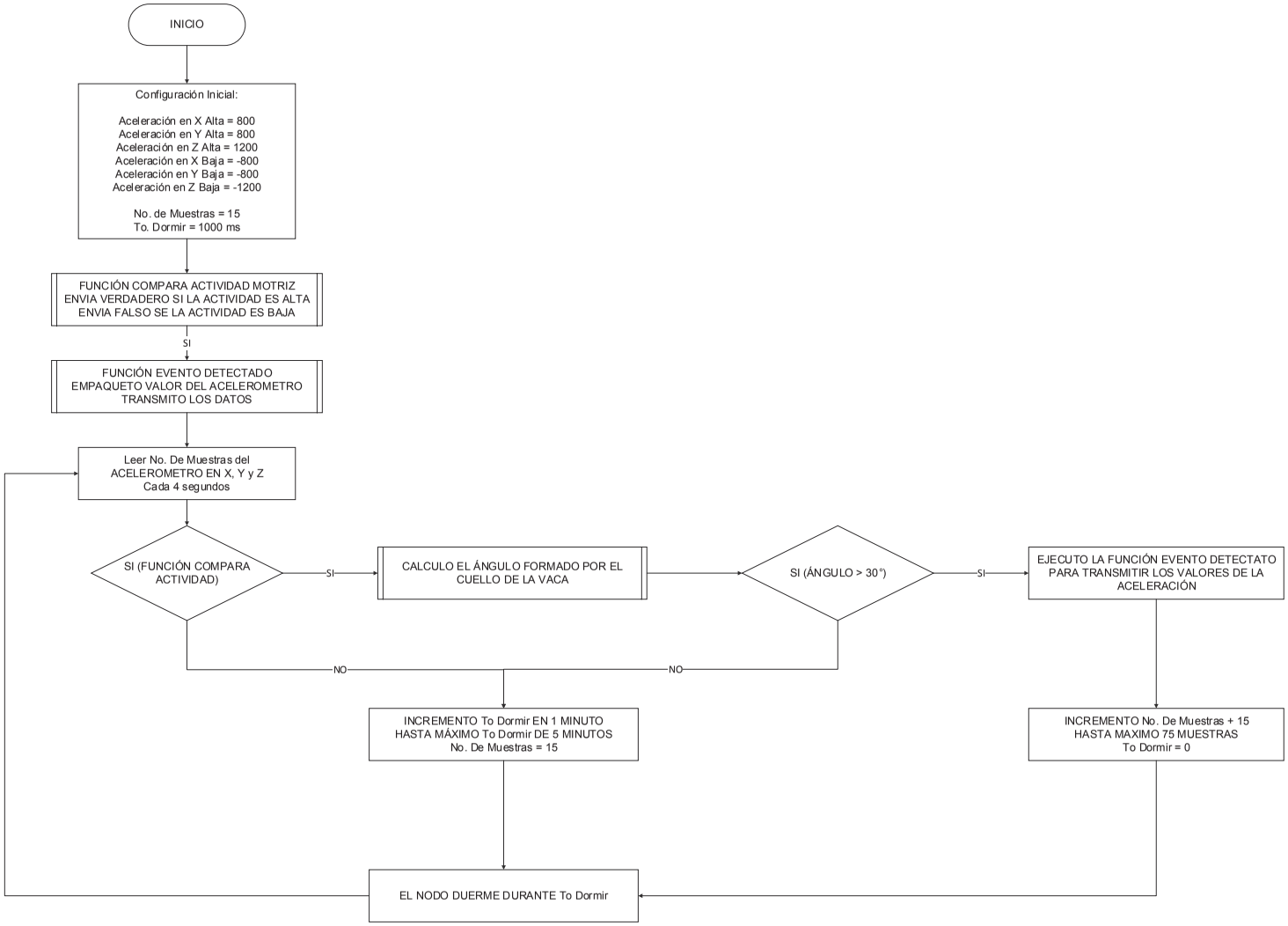


Figura 2.3 Diagrama de flujo algoritmo adaptado a la actividad motriz.

Como se observa en el diagrama el algoritmo toma decisiones evaluando el rango dinámico de la aceleración de un cuerpo, valor que es producto del movimiento de la vaca; la evaluación se realiza continuamente en el tiempo, hasta encontrar el nivel de aceleración dinámica, que permita concluir que la vaca se encuentra en el estro, enviando por tanto esta información a una aplicación de visualización para la toma de decisiones.

Los siguientes diagramas muestras las funciones de este algoritmo.

Función de Comparación de la Aceleración

Esta función compara los valores de la aceleración en los ejes X, Y y Z con los umbrales de aceleración, si detecta que estos valores están fuera del umbral retorna la función un dato verdadero para que se ejecute la transmisión de datos.

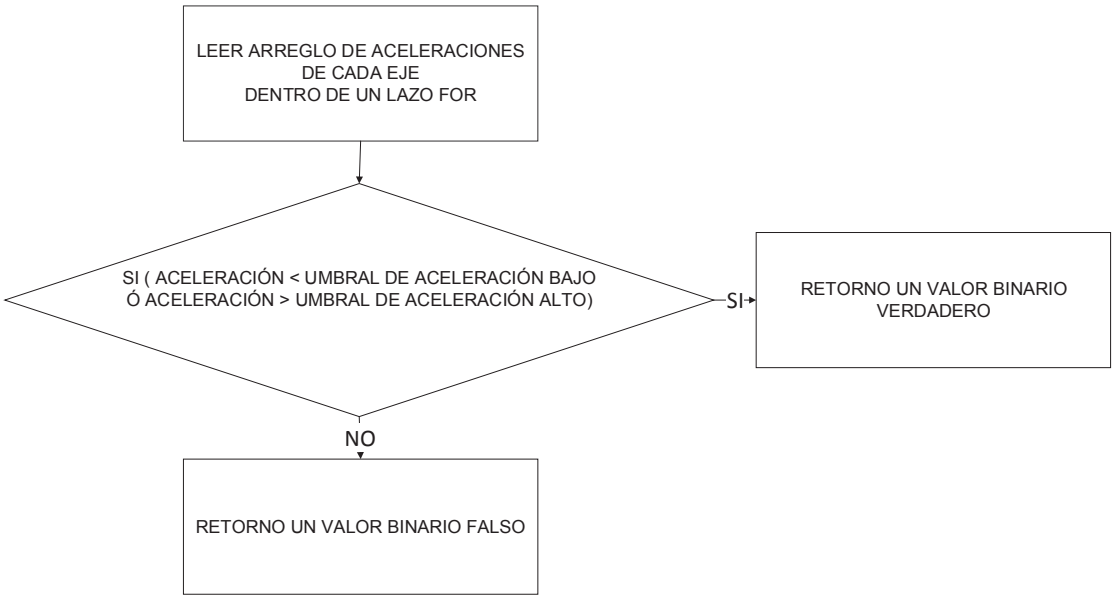


Figura 2.4 Diagrama de comparación de la aceleración.

Función Evento Detectado

Esta función crea un frame de formato ASCII, para agregar los datos que se transmitirán inalámbricamente, se adicionan al paquete tres valores aleatorios de la aceleración pues no es necesario transmitir todo paquetes pues se trata de un mismo evento de aceleración fuera de un rango normal.

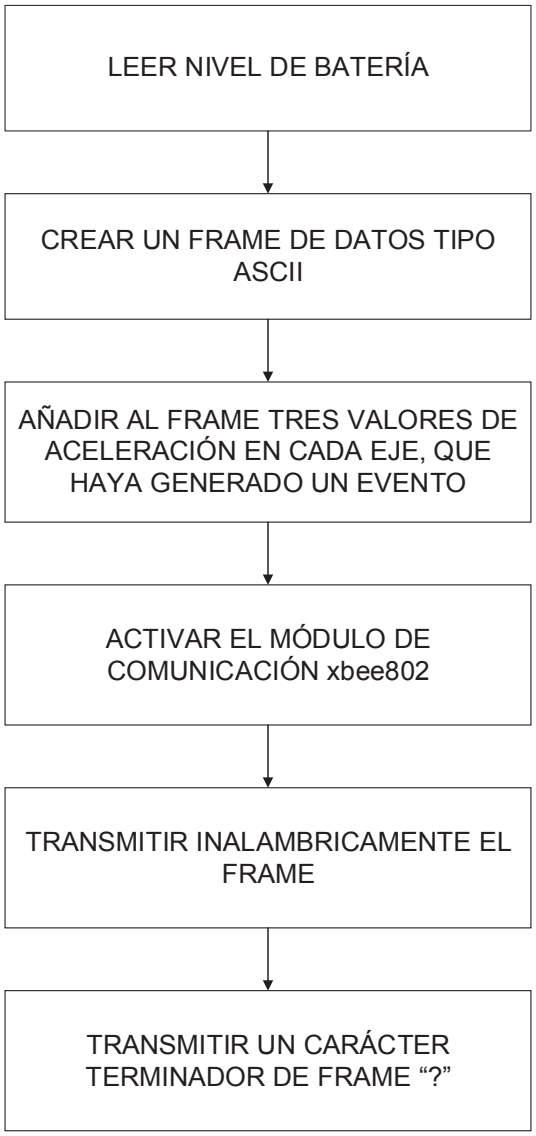


Figura 2.5 Diagrama de flujo de un evento detectado.

Diagrama de Flujo Comparación del Ángulo Respecto al Eje X

Este diagrama muestra el procedimiento para calcular y comparar el ángulo formado en el eje X por el movimiento de la cabeza de la vaca, este valor se usa para descartar cambios bruscos de la aceleración cuando la vaca está comiendo, pues puede arrancar la hierba del suelo de forma brusca y confundir al sistema.

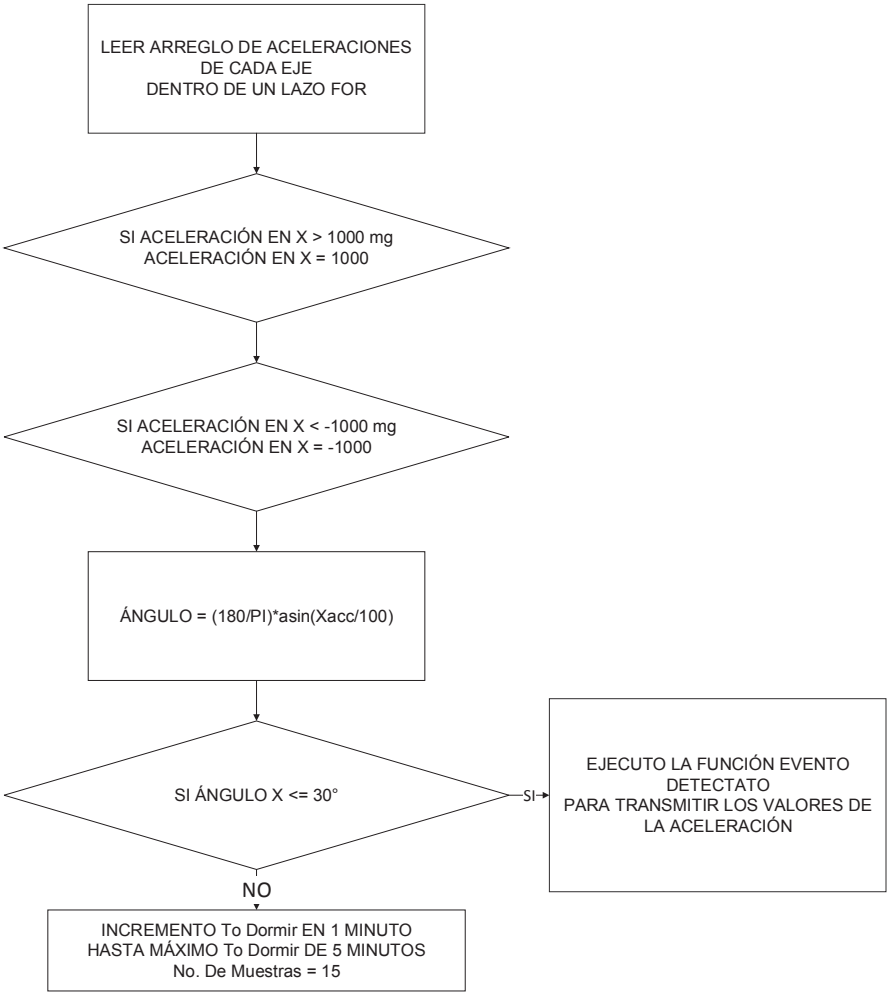


Figura 2.6 Diagrama de flujo para validación del ángulo formado respecto al eje X.

2.7 Análisis del Tiempo de Muestreo

De las tablas 1.3 y 2.2, encontramos que en estudios para determinar el estro, el mejor porcentaje de efectividad logrado por un sistema de observación se obtiene cuando la persona realiza esta acción, por lo menos 4 ocasiones con lapsos de 20 minutos en cada observación, llegando a concluir que la posibilidad de determinar el estro es de un 98%. Adicionalmente con el conocimiento de que la duración de una monta es entre 4 a 6 segundos, y es una de las actividades motrices más notorias en la etapa de estro, se decidió por tomar la siguiente estrategia de muestreo.

Modo: Baja Actividad

Cuando el nodo se determina que se encuentra en actividad baja inicia este estado durmiendo durante 1 minuto y una vez que se despierta realiza un muestreo de la actividad motriz durante 1 minuto con intervalos entre lecturas de 4 segundos.

Si la evaluación de estos valores es que la vaca continua en baja actividad se procede a aumentar el tiempo de hibernación a 2 minutos y posteriormente se muestrea durante 1 minuto la actividad motriz con intervalos entre lecturas de 4 segundos.

Se evalúa estas nuevas muestras hasta llegar a un tiempo máximo de 5 minutos en modo dormido, con incrementos de 1 minuto; el tiempo de lectura de datos siempre será de 1 minuto con lecturas cada 4 segundos.

Definido este modo de lectura del nodo sensor en etapa de baja actividad motriz se puede esperar un mínimo de 3600 muestras durante un día, descartando el modo de inicio y evaluando las 24 horas del día multiplicadas por 150 muestras por hora.

Modo: Alta Actividad

El muestreo en alta actividad inicia tomando muestras durante 1 minuto con muestras cada 4 segundos, si la actividad motriz continua alta este tiempo va incrementando 1 minuto más hasta llegar a un tiempo máximo de 5 minutos y con muestreos de la actividad de 4 segundos.

El número máximo muestras que se obtendrá en este modo es 21600 muestras, es decir una muestra cada 4 segundos durante todo el día, sin embargo, solo cada 5 minutos se enviaría 3 valores, pues se asume que todos eso valor son un indicativo de actividad motriz alta, este caso es hipotético pues la vaca nunca estaría en una actividad motriz alta todo el día.

Como se observa el modo de actividad motriz baja el tiempo mínimo de observación es de 10 minutos por hora con muestras cada 4 segundos durante todo el día, representando un porcentaje de observación del 4,17%, mientras que en actividad motriz alta el muestreo es constante, monitoreando la actividad cada 4 segundos todo el día implicando un porcentaje de observación del 25 % y si comparamos estos porcentajes con el porcentaje de observación de la tabla 1.3 que es de 5,5%, se espera tener una efectividad superior al 98% para determinar el celo, debido a que esta tabla es para vacas en celo es decir cuando la actividad motriz es alta.

2.8 Prototipo de Prueba de RSI para una Aplicación de Monitoreo Reproductivo Bovino

Con la finalidad de evaluar el algoritmo adaptativo en una aplicación práctica se implementó su código y se desarrolló un prototipo de prueba.

Para el diseño del prototipo de monitoreo reproductivo bovino se consideraron los siguientes requisitos del sistema.

1. La actividad motriz que se va a monitorear es la aceleración dinámica de un cuerpo, para ello se debe usar un sensor acelerómetro que debe estar integrado en el nodo sensor.
2. El dispositivo de monitoreo bovino debe ir colocado en el cuello de una vaca a manera de un collar, a fin de tomar medidas en los tres ejes de aceleración X, Y y Z.
3. El nodo sensor debe ser alimentado por baterías permitiendo la movilidad.
4. El sensor debe contar con un módulo de comunicación basado en el estándar IEEE 802.15.4.
5. La información debe ser entregada a una pasarela para que ella convierta los datos transmitidos por el protocolo IEEE 802.15.4, en datos entendidos por una aplicación de computadora.
6. Los datos capturados se presentan en una interfaz sencilla para que el responsable ganadero tome acciones.

2.9 Selección del Hardware

El nodo sensor y la pasarela o gateway que cumplen con los requisitos para este proyecto son:

- Waspmote
- Waspmote Gateway

Productos comerciales de la empresa Libelium que trabajan con el estándar de comunicaciones IEEE 802.15.4.

Waspmote

El Waspmote de Libelium es un modelo comercial, constituido por una tarjeta electrónica con un microcontrolador ATmega, que se conecta el microcontrolador a todos los módulos que permiten su funcionamiento, permitiendo al usuario configurar el mismo según la aplicación, así se puede observar en la siguiente Figura 2.7.

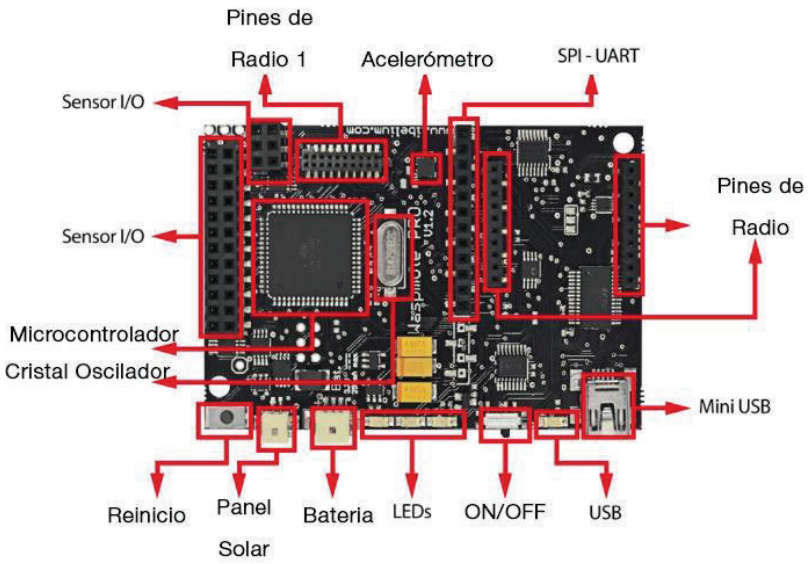


Figura 2.7 Waspmote Libelium [6].

Esta tarjeta es alimentada con una batería de 3,3 a 4,2 V de Li-Ion, se selecciona por integrar en su placa un sensor acelerómetro de ±2g/±4g/±8g, por lo que no es necesario la adquisición de un nuevo sensor para la aplicación, sus principales características se resumen en la tabla 2.3

Tabla 2.3 Características Nodo Waspmote

DESCRIPCIÓN GENERAL	
Microcontrolador	ATmega1281
Frecuencia:	14.7456 MHz
SRAM:	8KB
EEPROM:	4KB
FLASH:	128KB
Tarjeta SD:	2GB
Peso:	20g
Dimensiones:	73.5 x 51 x 13 mm
Rango de temperatura:	[-10°C, +65°C]

Reloj:	RTC[4] (32KHz)
CONSUMO ELÉCTRICO	
ON:	17mA
Deep:	55µA
Sleep:	55µA
Hibernate:	0.07µA
MÓDULO DE COMUNICACIÓN	
Versión:	XBee-PRO 802.15.4
Protocolo:	IEEE 802.15.4
Frecuencia:	2.4 GHz
Potencia de Transmisión:	10dBm
Sensibilidad:	-100dBm
Rango de cobertura:	750 m

Waspmote Gateway

Es una pasarela ZigBee - USB PC, para interconectar una interfaz USB de un computador, con el módulo de comunicación 802.15.4., es soportado en los sistemas operativos Linux, Windows y Mac, puede ser compilado con IDE – Waspmote, o lenguaje C++ [7].

2.10 Modos de Consumo Energético

Este nodo sensor dispone de cuatro modos de consumo eléctrico que se presentan a continuación: [7].

ON: modo normal de funcionamiento. El consumo en este estado es de 17mA.

[4] Reloj en tiempo real (en inglés, real-time clock, RTC) es un reloj de un ordenador, incluido en un circuito integrado, que mantiene la hora actual

Sleep: El programa principal se detiene, el microcontrolador pasa a un estado de latencia, del que puede ser despertado por todas las interrupciones asíncronas y por la interrupción síncrona generada por el Watchdog Timer[5]. El consumo en este estado es de 55µA.

Deep Sleep: El programa principal se detiene, el microcontrolador pasa a un estado de latencia del que puede ser despertado por todas las interrupciones asíncronas y por la interrupción síncrona lanzada por el RTC. El consumo en este estado es de 55µA.

Hibernate: El programa principal se detiene, el microcontrolador y todos los módulos de Waspmote quedan completamente desconectados. La única forma de volver a activar el dispositivo es a través de la alarma previamente programada en el RTC (interrupción síncrona). Al quedar el dispositivo totalmente desconectado de la batería principal el RTC es alimentado a través de una batería auxiliar de la que consume 0,07µA.

2.11 Diseño del Sistema de Monitoreo Bovino

Para probar el algoritmo adaptativo a la fisiología bovina, es necesario implementar una red de comunicación entre un nodo y la pasarela, para ello se implementó una red punto a multipunto con solo estos dos dispositivos, permitiendo realizar el análisis del movimiento corporal y la aceleración dinámica de una vaca. Así el sistema prototipo consta de un nodo sensor, una pasarela y una estación base, como se presenta en la Figura 2.8.

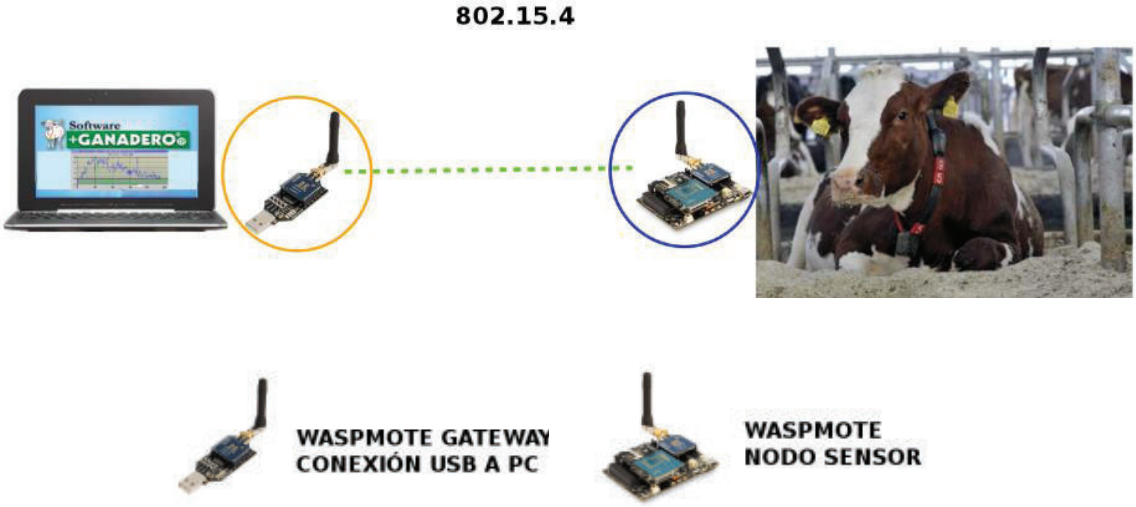


Figura 2.8 Red punto a punto para aplicación de monitoreo de reproducción bovina.

[5] Temporizador de perro guardia (en inglés, watchdog timer WDT), es un temporizador que provoca el reseteo del sistema después de un cierto tiempo.

Como se observa en el gráfico, la aplicación a nivel de hardware requiere de un nodo sensor y una pasarela, así el primer reto es adaptar este nodo sensor para que pueda ser transportado constantemente por una vaca, y monitorear la actividad motriz mediante el acelerómetro. El nodo sensor se introdujo dentro de un estuche, y se colocó en forma de collar en el cuello de la vaca, tal como se muestra en la Figura 2.9.

Figura 2.9 Prototipo de monitoreo reproductivo bovino.

2.12 Configuración del General del Nodo Sensor

La configuración de la red y sus parámetros generales se especifican en la siguiente tabla.

Tabla 2.4 Configuración general

CONFIGURACIÓN DEL SENSOR	
Acelerómetro:	+- 2 G
Modo:	Bajo poder 5Hz
CONFIGURACIÓN DE LOS DATOS	
Marco de Datos:	ACII
Librería:	WaspFrame
Dato 1	Nivel de Batería
Dato 2	Valor aleatorio de la aceleración en XYZ

Dato 3	Valor aleatorio de la aceleración en XYZ
Dato 4	Valor aleatorio de la aceleración en XYZ
MODO DESPERTADOR	
Reloj	RTC
Modo RTC	Modo 4 (Minutos y segundos)
MODOS DE OPERACIÓN DEL PROTOTIPO	
ON:	17mA
Sleep:	55µA
CONFIGURACIÓN DEL MÓDULO DE COMUNICACIÓN	
Versión:	XBee-PRO 802.15.4
Protocolo:	IEEE 802.15.4
Frecuencia:	2.4 GHz
Potencia de Transmisión:	10dBm
Sensibilidad:	-100dBm
Rango de cobertura:	750 m
Librería:	WaspXBee802

La estructura del campo de datos para la transición de información se la realizo en base a la estructura provista por el fabricando del nodo, la misma que se describen la Figura 2.10.

CABECERA										CARGA ÚTIL					
A	B	C	D	E	D	F	D	G	D	sensor1	D	sensor2	D	sensor3	D

Figura 2.10 Formato de estructura de datos.

A continuación, se describe la cabecera:

A. Delimitador de Inicio [3 bytes]: compuesto por el carácter “<=>”, es necesario para identifica el inicio de un paquete.

B. Tipo de Frame [1 byte]: identifica si es binario o ASCII, en nuestro caso ASCII.

C. Número de Campos [1 byte]: especifica el número de campos enviados en el paquete.

D. Separador [1 byte]: usa el carácter “#” para separar cada campo.

E. Serial ID [16 bytes]: este campo identifica a cada nodo de forma única, este valor se encuentra en el módulo de comunicaciones.

F. Waspmote ID [16 bytes]: identificador del nodo definido por el usuario.

G. Secuencia de Trama [1 – 3 bytes]: indica el número de trama de la secuencia.

La carga útil corresponde a los valores de sensores o datos que se quieren transmitir.

2.13 Alimentación Eléctrica del Prototipo

Para la alimentación eléctrica del prototipo, se utilizó un banco de baterías recargables, provistas por la empresa Libelium; las baterías son de iones de litio, brindando un voltaje nominal de 3,7 V y una corriente de 6600 mAh.; su conexión al Waspmote es mediante un socket, por lo que su reemplazo es sencillo. En la Figura 2.11 se observa la batería y su conexión al nodo.

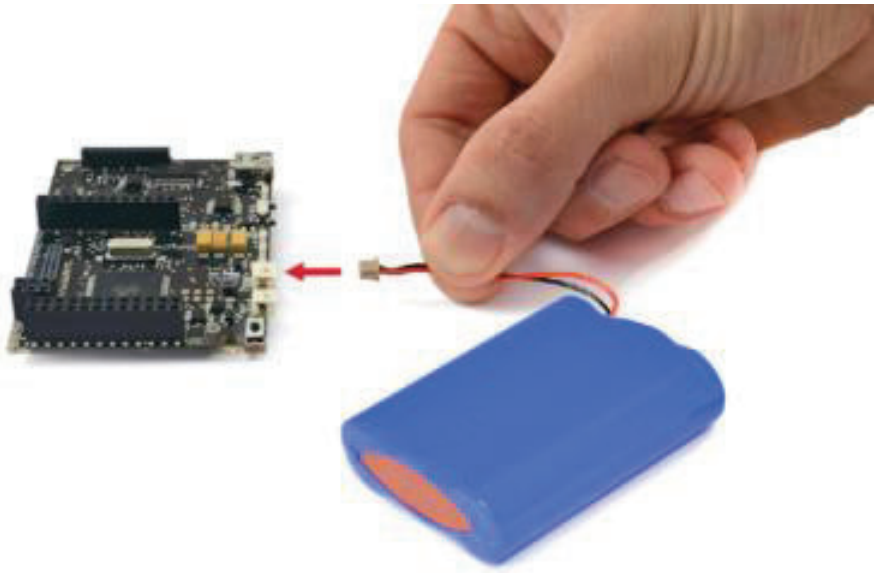


Figura 2.11 Batería Libelium 3,7v 6600 mAh.

2.14 Simulación de la Red de Sensores Inalambricos en OMNeT++

Con el objetivo de determinar el consumo eléctrico que tendría el nodo sensor se desarrolló una simulación en el programa OMNeT++, que es un simulador de eventos discretos, que presenta ya un marco de trabajo para el estándar IEEE 802.15.4, lo que permite utilizar las herramientas ya creadas para la simulación.

Para la realización de la simulación, se basó en el marco de simulación IEEE802154INET-Standalone [8] que brinda un conjunto de escenarios para el estándar IEEE 802.15.4 y, principalmente, permite la evaluación de la energía restante de la batería.

Este marco de trabajo fue añadido sobre la versión de OMNeT++ versión 4.6 [19] y utiliza código desarrollado en la librería INET 2.4.0 [20], que también fue añadida a la instalación de OMNeT++.

Para el cálculo de la energía de una batería en OMNeT++ utiliza la siguiente fórmula:

$$Energía\ Batería\ [mW * s] = carga\ electrica\ de\ la\ bateria\ (mAh)\ x\ voltaje\ de\ la\ batería\ (V)\ x\ 3600$$

Ecuación 2.1 Energía de la Batería

La misma que se puede encontrar en el archivo InetSimpleBattery.cc:

$$capacity = capmAh * 60 * 60 * voltage;$$

Ecuación 2.2 Capacidad de energía usada en OMNeT++

En nuestro ejemplo, la batería de Libelium tiene una carga de 6600mAh y proporciona un voltaje de 3,7 V, significa que la energía de la batería es:

$$Energía\ Batería\ Libelium = 6600\ (mAh) * 3{,}7V * 3600$$

$$Energía\ Batería\ Libelium = 87'912.000\ [miliWatt * segundo]$$

Este valor de la capacidad de energía de la batería es el valor que el marco de trabajo almacenará en un vector de resultados para el análisis.

A continuación, se indica el procedimiento necesario para que OMNeT++ permita calcular la energía, los valores de corriente son obtenidos en [12].

1. En la librería inet, localizar el archivo InetSimpleBaterry.cc en la siguiente ruta inet/src/battery/ InetSimpleBaterry.cc.

2. Se abre el archivo, y se identifica la línea de código que contenga “residualVec.disable();” y la reemplazando por “residualVec.enable();”. Esto permite que los datos recolectados de la batería se almacenen en función del tiempo en un vector llamado residualCapacity.

3. Configurar los parámetros de consumo de energía en el archivo de inicialización *.ini, con los valores del nodo Libelium [7] y los obtenidos en [12].

```
#--------------------Power Consumption--------------------#
*.IEEE802154Nodes[*].NIC.radioInterface.usage_IEEE802154Radio_idle = 17mA
# -1 to disable battery registration
*.IEEE802154Nodes[*].NIC.radioInterface.usage_IEEE802154Radio_recv = 57.2mA
*.IEEE802154Nodes[*].NIC.radioInterface.usage_IEEE802154Radio_send = 57.2mA
*.IEEE802154Nodes[*].NIC.radioInterface.usage_IEEE802154Radio_sleep=0.055mA
*.IEEE802154Nodes[*].batteryType = "InetSimpleBattery"
*.IEEE802154Nodes[*].battery.capacity = 6600
*.IEEE802154Nodes[*].battery.voltage = 3.7
*.IEEE802154Nodes[*].battery.publishTime = 5s
*.IEEE802154Nodes[*].battery.resolution = 5s
```

Código 2.1 Parámetros de energía en OMNeT++

Con estas configuraciones se puede ejecutar la simulación y los datos se almacenarán en forma escalar en el vector residualCapacity.

2.15 Programación del Nodo Sensor con el Algoritmo Adaptativo a la Actividad Motriz de un Bovino

El software que ejecuta el algoritmo adaptado a la actividad motriz de un bovino fue desarrollado en C++ en el Waspmote Pro API - v04, así se desarrolló el código con varias funciones que son llamadas dentro de un lazo “loop” que se ejecutan constantemente, el siguiente es el código con el algoritmo escrito para el nodo sensor:

```
//PROGRAMA DE DETECCIÓN DEL ESTRO
//CON ALGORITMO ADAPTATIVO A LA
//ACTIVIDAD MOTRIZ BOVINA


//LIBRERIAS
//Para crear el frame a transmitir y la librería para transmisión

#include <WaspFrame.h>
#include <WaspXBee802.h>
```

```
//================CONSTANTES============//


#define POLL_LENGTH_INIT 15

#define X_LOWER_BOUND -800
#define Y_LOWER_BOUND -800
#define Z_LOWER_BOUND -1200

#define X_UPPER_BOUND 800
#define Y_UPPER_BOUND 800
#define Z_UPPER_BOUND 1200

//ID del nodo, este es el ID de la Vaca También
char NODE_ID[] = "1N452A12";

//ID del módulo de comunicación de la pasarela
char RX_ADDRESS[] = "0013A20040DAF15D";


//=================VARIABLES===============//

int poll_length;

boolean alert;
int seconds;
int minutes;
int count;
int flagangulo;
int battery_level;

int x_acc[100];
int y_acc[100];
int z_acc[100];

int x_mean;
int y_mean;
int z_mean;

//flagangulo es una bandera que me indica si el ángulo es mayor a 30 grados


//==================FUNCIONES======================//


//Ejecuta un control de la medición de la aceleración para devolver una alerta si algún
//dato sobrepasa los valores de lower_bound o upper_bound.

boolean control(int data[], int data_length, int lower_bound, int upper_bound)
{
	for (int i = 0; i < data_length; i++)
	{
		if (data[i] < lower_bound || data[i] > upper_bound)
```

```
                {
                        return true;
                }
        }
        return false;
}


//Ejecuta la lectura del acelerómetro una cantidad de POLL_LENGHT veces y las
// almacena en los arreglos x_acc, y_acc y z_acc. Cada lectura se realiza con una
//separación de 4 segundos, el valor está en milisegundos.

void acc_poll()
{
        //Se configure el acelerómetro en un modo de bajo consumo eléctrico
        ACC.ON(FS_2G);  //2G
        ACC.setMode(ACC_LOW_POWER_2); //5Hz
        for (int i = 0; i < poll_length; i++)
        {
                x_acc[i] = ACC.getX();
                y_acc[i] = ACC.getY();
                 z_acc[i] = ACC.getZ();
                PWR.deepSleep("00:00:00:04",RTC_OFFSET,
                RTC_ALM1_MODE1,ALL_OFF);
        }
}

//Verifica en cada una de las direcciones del acelerómetro si ha existido una sobre o bajo
medición
//para considerar si se ha disparado una alerta de movimiento brusco

boolean trigger()
{
        return control(x_acc, poll_length, X_LOWER_BOUND, X_UPPER_BOUND) ||
        control(y_acc, poll_length, Y_LOWER_BOUND, Y_UPPER_BOUND) ||
        control(z_acc, poll_length, Z_LOWER_BOUND, Z_UPPER_BOUND);
}


//Crea un Frame con las mediciones del acelerómetro y lo envía a la pasarela.

void run_event()
{
//Lee la medición de porcentaje de la batería del nodo.
        battery_level = PWR.getBatteryLevel();
        //Crea el Frame con las mediciones.
        frame.createFrame(ASCII, NODE_ID);
         //Añado al Frame el valor actual de la Batería
         frame.addSensor(SENSOR_BAT, battery_level);
          int a;
        for(int i=0;i<3;i++)
        {
                if(i==0)
                a=1*count+3;
```

```
                if(i==1)
                a=3*count+2;
                if(i==2)
                a=9*count;
                frame.addSensor(SENSOR_ACC, x_acc[a], y_acc[a], z_acc[a]);
        }
        //Se construye el buffer añadiendo los valores de la aceleración dinámica
        //del cuerpo de la vaca
        //Envía el Frame a la pasarela (se debe modificar si se cambia
        //el medio de comunicación).
        xbee802.ON();
        xbee802.send(RX_ADDRESS, frame.buffer, frame.length);
        xbee802.send(RX_ADDRESS, "?");
}

//Configuración Inicial

void setup()
{
        //Inicialización de variables.
        //Variable "alert" registra si se encuentra o no la condición de alerta activa.
        //La condición de alerta se genera cuando ha existido un disparo
        alert = false;
        //Variables "seconds" y "minutes" registran el tiempo que el nodo duerme.
        seconds = 0;
        minutes = 0;
        //Variable para incrementar el largo del poll de muestras
        count = 1;
        poll_length = POLL_LENGTH_INIT;
        //configuración acelerómetro
        ACC.ON(FS_2G);  //2G
        ACC.setMode(ACC_LOW_POWER_2); //5Hz medida de bajo consumo
        RTC.ON();
}

//Lazo continúo

void loop()
{
        //Lee las mediciones del acelerómetro para cada coordenada.
        acc_poll();
        //Garantizo que el módulo se apague
        ACC.OFF();
        //Verifica si existe algún disparo (Medición fuera del rango normal).
        if (trigger())
        {
                flagangulo=1;
                // Comparación del ángulo formado por el cuello de la vaca
                //cuando está comiendo
                for (int i = 0; i < poll_length; i++)
                {
                        float Xacc = x_acc[i];
                        if (Xacc > 1000) Xacc = 1000;
                        if (Xacc < -1000) Xacc = -1000;
```

```
                float angle_x = (180 / PI) * asin(Xacc/1000);
                if (angle_x <= 30)
                {
                        flagangulo=1;
                }
                else
                {
                        flagangulo=0;
                }
        }
        if (flagangulo == 1)
        {
                run_event();
                if(count<5)
                {
                        count++;
                }
                minutes=0;
                seconds=5;
        }
        if (!alert)
        {
                alert = true;
        }
}
else
{
        count=1;
        //Si el disparo ha sido negativo la condición de alarma se desactiva y el
        //tiempo de dormir se reconfigura incrementando 1 minuto hasta un
        //máximo de 5 minutos.
        if(!alert && minutes < 5)
        {
                minutes++;
                seconds=0;
        }
        if (alert) alert = false;
}
// asigna nuevo valor a poll_length
poll_length = 15*count;
if(seconds==0)
{
        frame.createFrame(ASCII, NODE_ID);
        xbee802.ON();
        xbee802.send(RX_ADDRESS, "Nodo En Red");
}
//Encender del RTC
RTC.ON();
//Limpiar las banderas para que el nodo despierte
RTC.clearAlarmFlag();
//Limpiar las interrupciones para garantizar que el nodo despierte
PWR.clearInterruptionPin();
//Configuro una alarma para que el nodo se despierte
RTC.setAlarm1(0,0,seconds,minutes, RTC_OFFSET, RTC_ALM1_MODE4);
```

```
        //Enviar al nodo a modo dormir profundo
        PWR.sleep(ALL_OFF);
        //Limpiar las banderas
        RTC.clearAlarmFlag();
        //Limpiar las interrupciones
        PWR.clearInterruptionPin();
}
```

Código 2.2 Programación del nodo sensor

Como se observa del código presentado, el primer paso que se realiza es añadir las librerías WaspFrame.h y WaspXBee802.h, que son librerías desarrolladas por Libelium y permiten crear un paquete de datos y controlan el módulo de comunicación XBee Pro, que transmite este paquete usando el estándar 802.15.4.

Luego se procede a definir las variables de inicio, así como a crear el resto de variables que se utilizan en el programa.

A continuación, se declaran todas las funciones que se ejecutan, cuando son llamadas por el lazo “loop”.

La función “void acc_poll()”, es la encargada de tomar los valores de la aceleración dinámica con un tiempo entre muestras de 4 seg. y captura el número de muestras indicadas por la variable “poll_length”.

Después de ejecutar la lectura de la aceleración dinámica de la vaca, se llama a la función “trigger()” que evaluada con un operador condicional “if”.

La función “trigger()” a su vez ejecuta la función “boolean control(int data[], int data_length, int lower_bound, int upper_bound)”, que toma los valores de la aceleración, el largo del arreglo de la aceleración y, los umbrales máximo y mínimo de la aceleración para comparar los mismo. Así, la función control devuelve un valor booleano, si la aceleración está fuera de lo normal la función devuelve un valor verdadero, caso contrario devuelve un valor falso.

Si se detecta un valor positivo, con la función “control”, se procede a evaluar el ángulo que forma el cuello de la vaca en estas lecturas y si es mayor a 30°, significa que la misma se encontraba comiendo y se descartan esos valores, caso contrario se ejecuta la función “run_event()”; la Figura 2.12 muestra el código para este control.

CÁLCULO DEL ÁNGULO FORMADO POR EL CUELLO DE LA VACA

```
for (int i = 0; i < poll_length; i++)
{
      float Xacc = x_acc[i];
      if (Xacc > 1000) Xacc = 1000;
      if (Xacc < -1000) Xacc = -1000;
      float angle_x = (180 / PI) *
asin(Xacc/1000);
      if (angle_x <= 30)
      {
        flagangulo=1;
      }
      else
      {
        flagangulo=0;
      }
  }
```

Figura 2.12 Cálculo del ángulo del cuello de la vaca.

"run_event" es la función que activa el módulo de comunicación, empaqueta los datos de la aceleración en un solo paquete y procede a transmitirlos.

Dentro del lazo "loop", se utiliza variables auxiliares y contadores que sirven para incrementar el tiempo de dormir del nodo sensor cuando la actividad es baja, o aumentar el número de muestras cuando la actividad es alta, como se observa en la Figura 2.7.

INCREMENTO To Dormir EN 1 MINUTO
HASTA MÁXIMO To Dormir DE 5 MINUTOS
No. De Muestras = 15

```
if(!alert && minutes < 5)
{
     minutes++;
     seconds=0;
}
```

INCREMENTO No. De Muestras + 15
HASTA MÁXIMO 75 MUESTRAS
To Dormir = 0

```
if (flagangulo == 1)
{
   run_event();
   if(count<5)
   {
     count++;
   }
   minutes=0;
   seconds=5;
}

poll_length = 15*count;
```

Figura 2.13 Algoritmo y código.

Una vez compilado el programa fue cargado en el nodo sensor para poner a prueba el funcionamiento del mismo.

2.16 Programa de Lectura del Puerto Serial

La Figura 2.8 muestra el esquema de la aplicación de lectura del puerto serial, en el mismo se visualiza que en el computador que almacenará los eventos capturados por el nodo sensor se ejecuta un programa desarrollado en IntelliJ IDEA.

Este programa se está ejecutando periódicamente y es una aplicación que abre un terminal del computador para escuchar el puerto serial.

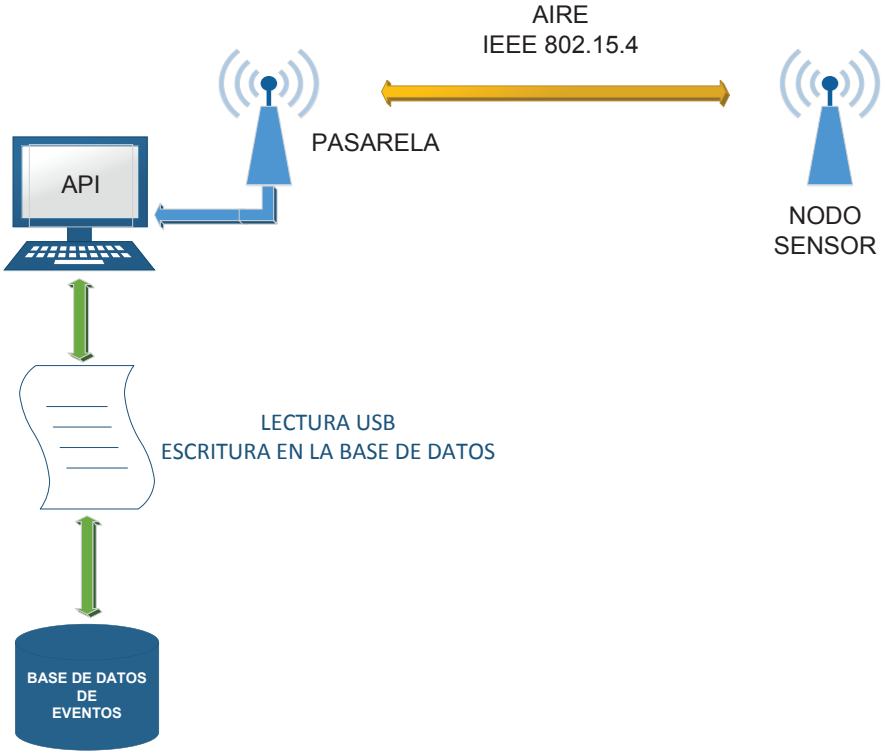


Figura 2.14 Esquema del programa de lectura de datos de la pasarela.

Los datos transmitidos por el nodo a la pasarela son recibidos por el puerto serial del computador, y cada vez que detecta datos en el mismo, ejecuta funciones que permiten desempaquetar el paquete transmitido, convertir el arreglo de datos en datos individuales y almacenar cada uno de éstos, en una base de datos que tiene como identificador un valor del tiempo de la computadora de cuando se recibieron los paquetes.

De este código desarrollado se explicará la función “NodeEvente(String frame)”, que es una función que recibe por dato el paquete recibido por la pasarela, ya convertido a texto, y utiliza los separadores de datos como “#”, “:”, “;” y las etiquetas “BAT” y “ACC”, para separar dato por dato y almacenar en un nuevo arreglo individual. Todo el código desarrollado e encuentra en el Anexo I.

```
public static NodeEvent Of(String frame) {
        Integer _batteryLevel = Integer.valueOf(0);
        ArrayList _accValues = new ArrayList();
        StringTokenizer tokens = new StringTokenizer(frame, "#");
        String _nodeId = tokens.nextToken();
        StringTokenizer _tokens = new StringTokenizer(tokens.nextToken(), ":");
```

```
        String label = _tokens.nextToken();
        //Busca el termino BAT para identificar el nivel de la batería después de los ":"
        if(label.equals("BAT")) {
            _batteryLevel = Integer.valueOf(_tokens.nextToken());
            _tokens = new StringTokenizer(tokens.nextToken(), ":");
            label = _tokens.nextToken();
        }
       //Busca el termino ACC para almacenar la aceleración después de los ";"
        while(label.equals("ACC")) {
            StringTokenizer __tokens = new StringTokenizer(_tokens.nextToken(), ";");
            AccValue sample = new AccValue();
           // El primer Token encuentra la aceleración en X, el segundo Y, el tercero Z
            sample.setAccX(Integer.valueOf(__tokens.nextToken()));
            sample.setAccY(Integer.valueOf(__tokens.nextToken()));
            sample.setAccZ(Integer.valueOf(__tokens.nextToken()));
            _accValues.add(sample);
            _tokens = new StringTokenizer(tokens.nextToken(), ":");
            label = _tokens.nextToken();
        }
        return new NodeEvent(LocalDateTime.now(), _nodeId, _batteryLevel, _accValues);
    }
```

Código 2.3 Lectura de los paquetes de datos recibido por la pasarela

2.17 Programa para el Usuario

Al igual que el programa anterior, este programa se desarrolló en IntelliJ IDEA, y es una interfaz gráfica simple, que muestra los valores de intensidad de la actividad motriz de la vaca, para que el usuario tome decisiones, su esquema es el mostrado en la Figura 2.15.

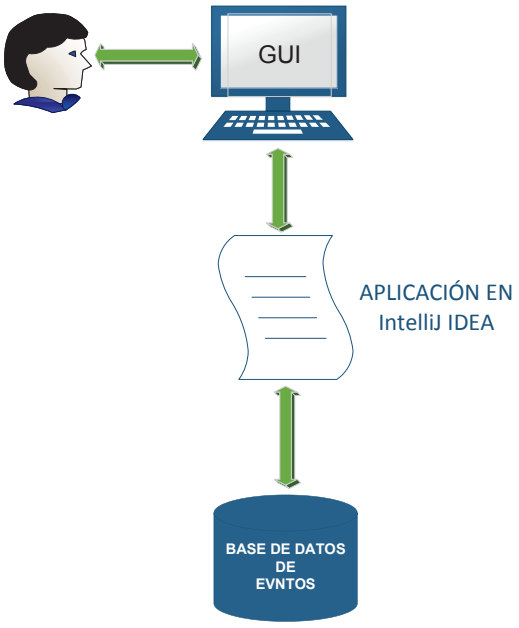


Figura 2.15 Diagrama de programa de usuario.

El programa presenta una interfaz que conecta con la base de datos desarrollada en MySQL y hace consultas de las variables de interés, todo el desarrollo del software lo encontramos en el anexo II.

El programa tiene el siguiente diagrama de base de datos.

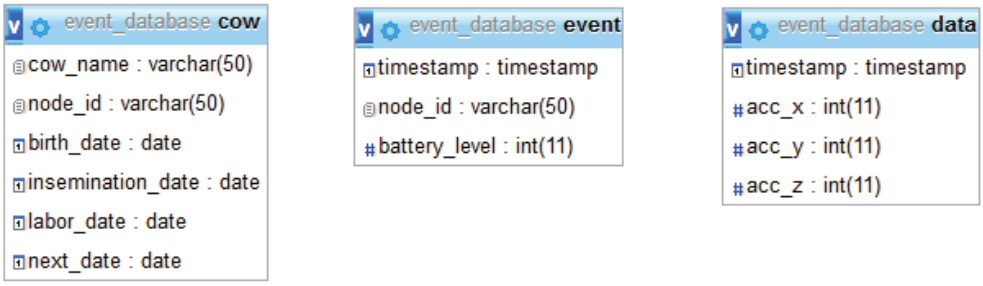


Figura 2.16 Base de datos.

La base esta compuesta por tres tablas: cow, event y data.

Tabla cow.- contiene la información reproductiva básica de la vaca, así, su nombre “cow_name”, identificación del nodo que lleva en su collar “node_id”, la fecha de nacimiento “birth_date”, su fecha de inseminación “insemination_date”, fecha de parto “labor_date”, y fecha de su próxima insemincación “next_date”.

Tabla evente.- es la encargada de almacenar la energía restante del nodo en el instante preciso en que recibe los datos, está compuesta por la información del tiempo “timestamp”, identificador del nodo “nodo_id” y nivel de batería “batery_level”.

Tabla data.- esta tabla contiene la estampa de tiempo de la transmisión “timestamp”, el valor de la aceleración en X “acc_x”, Y “acc_y” y Z “acc_z”.

Esta base de dato es creada por el programa de lectura del puerto serial y los valores almacenados en sus tablas se presentan en la interfaz gráfica de usuario GUI.

Diagrama de Casos de Uso y Clases

La Figura 2.16 muestra el diagrama de caso de uso del sistema, en el cual el usuario o el ganadero en este caso, tiene acceso a los casos de uso de visualización de caratula, visualización de valores de la aceleración, visualización de estadísticas, gestión de animales, y configuración de red. En los casos de visualización de aceleración y estadísticas, el usuario puede realizar una búsqueda especifica de información, por un

rango de tiempo o últimos valores; en el caso de gestión de animales el usuario puede crear, modificar o borrar los datos de las vacas existentes; en el caso de configuración de red se puede modificar la dirección IP y puerto del servidor de base de datos para visualizar el sistema en otro computador que este en la misma red.

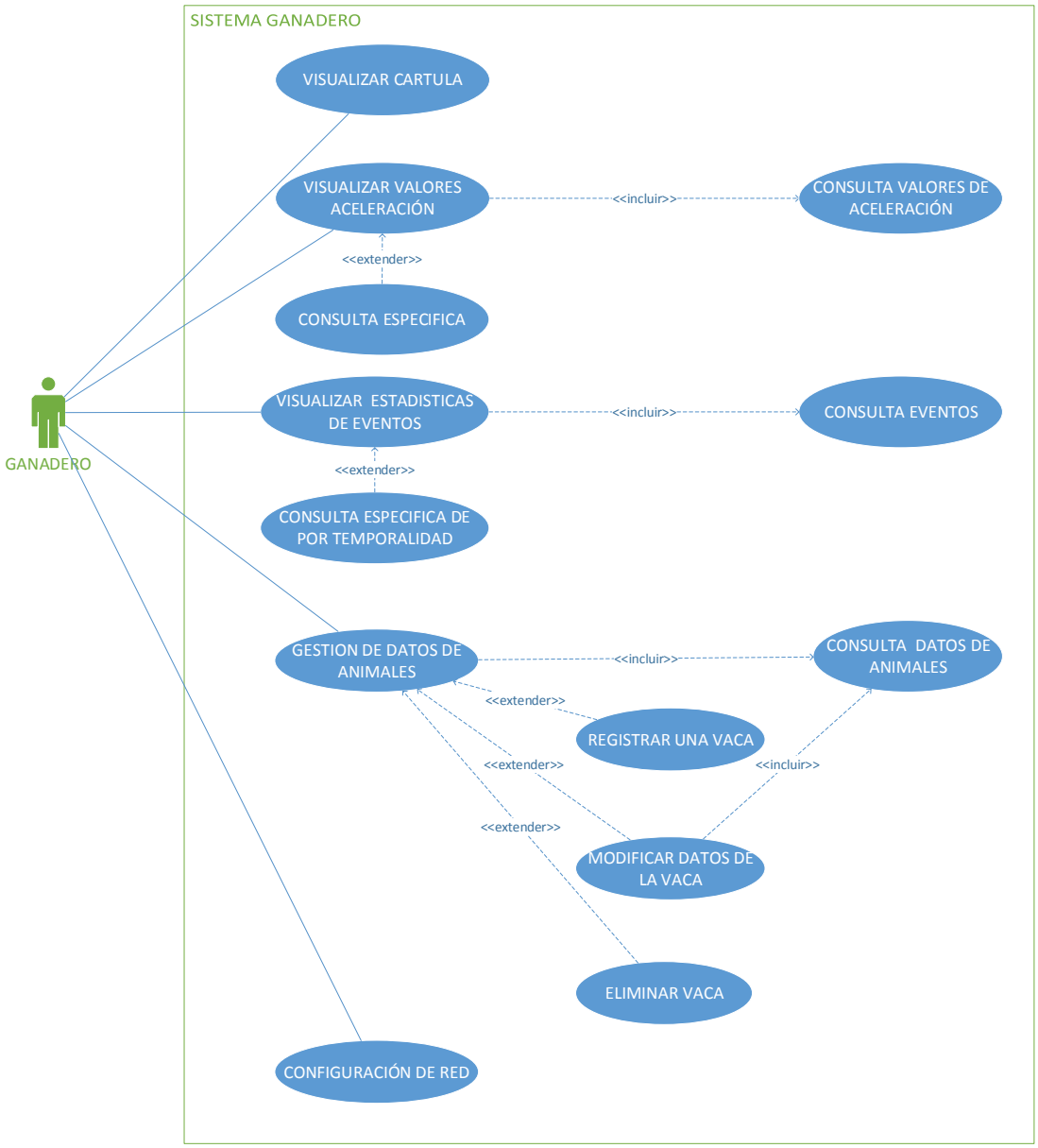


Figura 2.17 Casos de uso del programa ganadero.

La Figura 2.18 muestra el diagrama de clases de este programa.

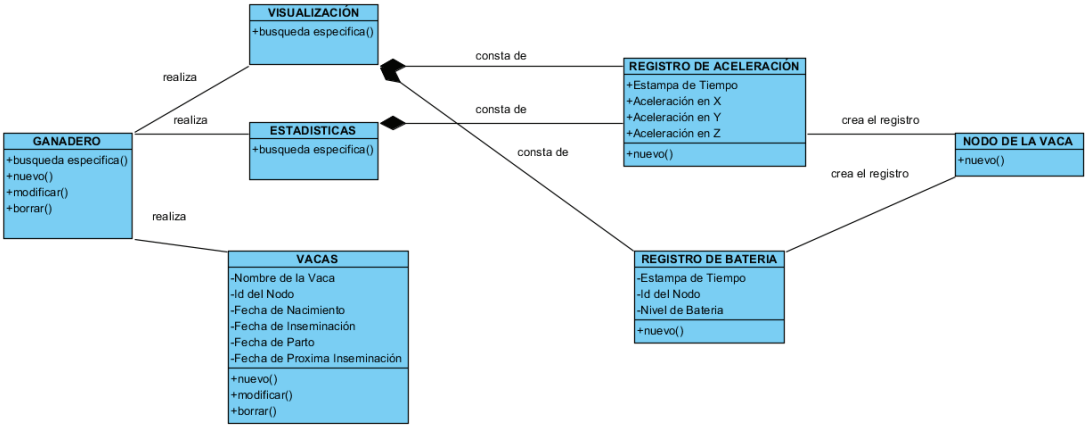


Figura 2.18 Diagrama de clases.

Interfaz Gráfica de Usuario

Las figuras que se muestran a continuación forman parte de la interfaz de usuario y describen el funcionamiento de las mismas.

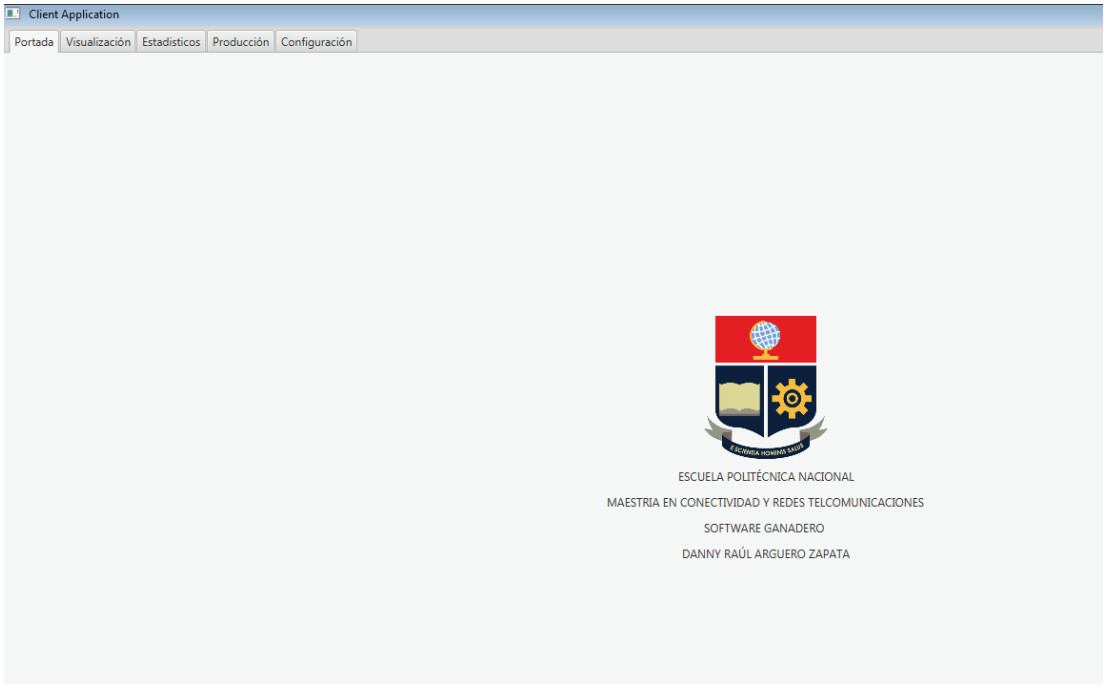


Figura 2.19 Interfaz gráfica de inicio.

Esta es la interfaz gráfica con que inicia la aplicación, en la parte superior se tiene los menús de Portada, Visualización, Estadísticas, Producción y Configuración.

Visualización. - es la interfaz donde se puede observar las muestras de la aceleración medidas por el acelerómetro en un tiempo determinado, permite realizar una evaluación minuciosa de cada actividad, presenta herramientas de búsqueda en la base de datos, usando el identificador del nodo y las fechas de inicio y fin de los eventos, el botón actualizar carga la información. La Figura 2.179muestra esta interfaz gráfica.

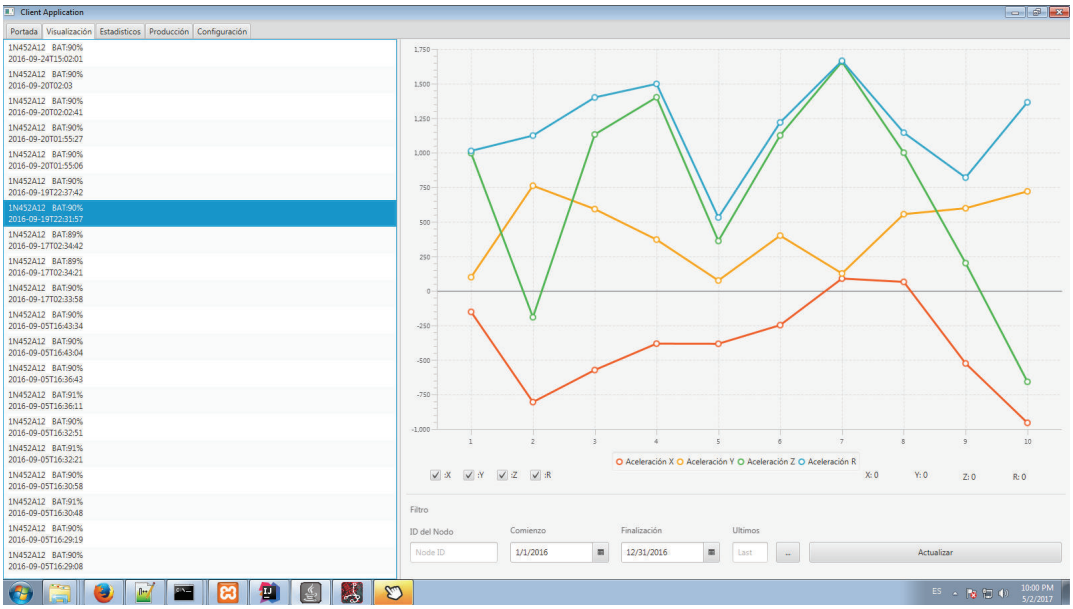


Figura 2.20 Visualización de eventos capturados por el acelerómetro.

En la parte inferior del gráfico, se puede seleccionar la medida en los ejes X, Y y Z; para evaluar como los valores obtenidos en cada uno de los ejes.

Estadísticas. - es la interfaz presenta una estadística de las fechas donde se detectó un evento, se puede realizar la búsqueda de eventos por el identificador del nodo, el año y un rango de meses; en esta interfaz el usuario final o veterinario podrá determinar si la vaca se encuentra en estro, pues tendrá más información de la actividad en un día específico. La Figura 2.21 muestra esta interfaz.

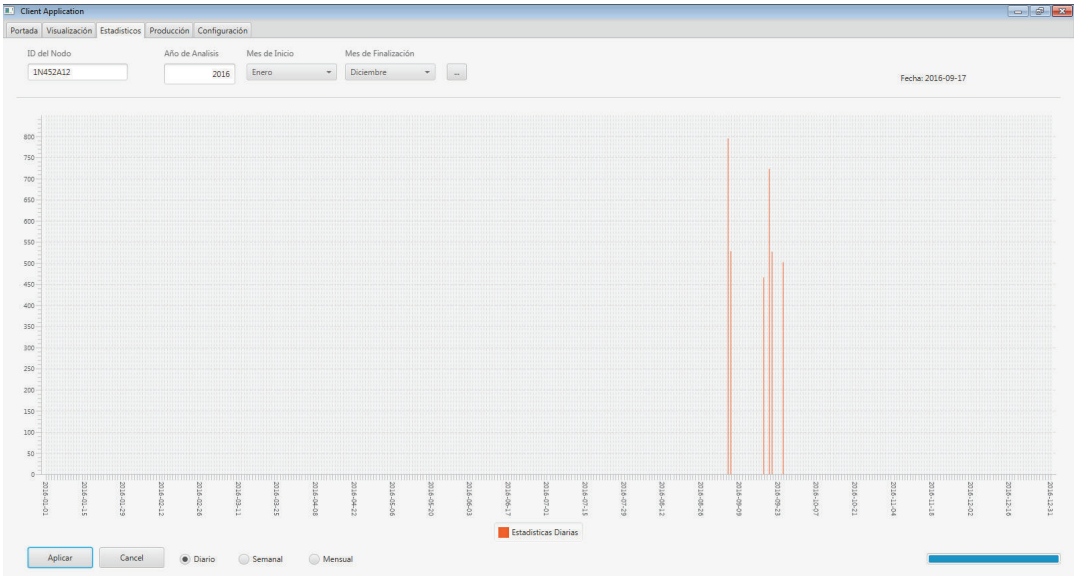


Figura 2.21 Estadísticas de los eventos almacenados por fecha.

Producción. - muestra la información básica del bovino en su ciclo reproductivo, así se muestra su identificación que para el proyecto es la misma del nodo, el nombre de la vaca, su fecha de nacimiento, fecha del último parto, fecha de la última inseminación y la fecha de la próxima inseminación, esta información puede ser modificada por el usuario y le servirá para toma de decisiones.

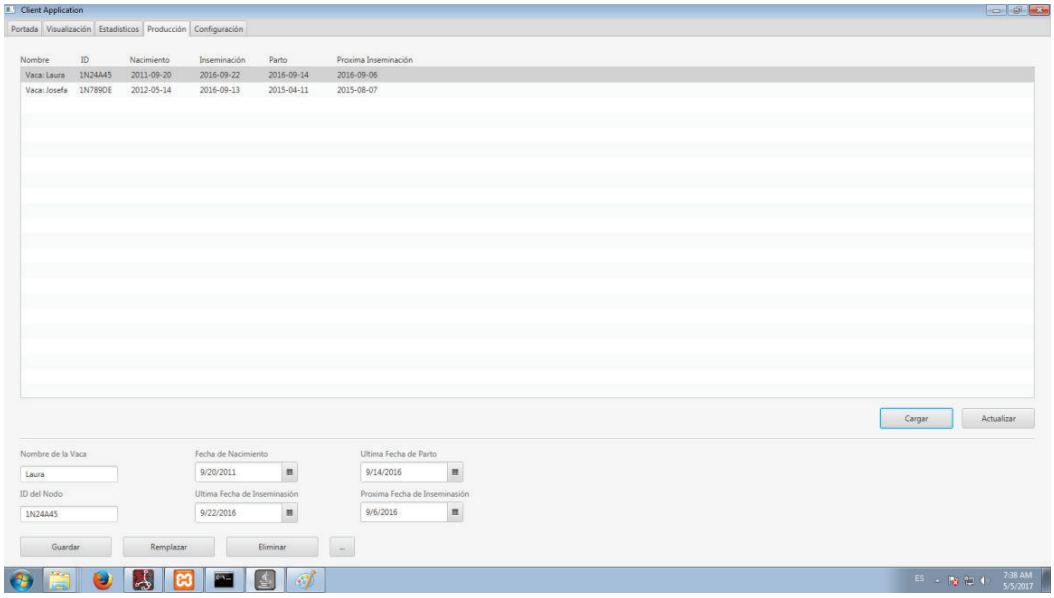


Figura 2.22 Producción, muestra la información reproductiva de la vaca.

Configuración. – Esta ventana permite configurar de la dirección IP y puerto del servidor para utilizar el programa desde otra computadora en la red.

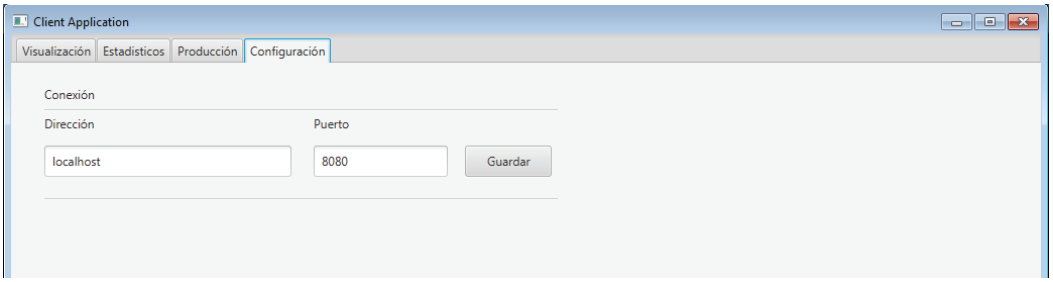


Figura 2.23 Configuración, configuración de red.

2.18 Conectividad de la Red 802.15.4

La red establecida en este proyecto es una red punto a punto 802.15.4, se implementó un código para que envíe un mensaje de “Nodo en red”, cuando el nodo se encuentra en baja actividad; en alta actividad, está enviando normalmente valores del acelerómetro, esto podremos visualizar en el programa de lectura del puerto serial, como se observa en la Figura 2.24.

```
if(seconds==0)
{
   frame.createFrame(ASCII, NODE_ID);
   xbee802.ON();
   xbee802.send(RX_ADDRESS, "Nodo en red");
}
```

Código 2.4 Conectividad del nodo en modo de baja actividad

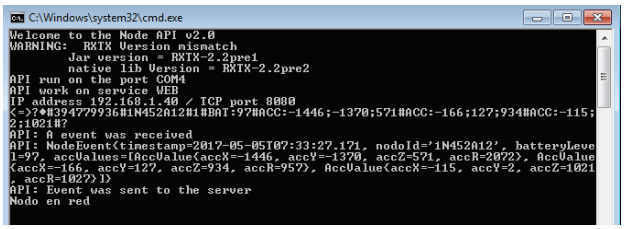


Figura 2.24 Programa de lectura USB conectividad el nodo.

Como se observa en la figura, después de recibir un evento de actividad alta y luego de que la misma disminuyera, se visualiza el mensaje nodo en red, que nos permite saber si el nodo se encuentra conectado y transmitiendo, esto garantiza la conectividad del mismo, aproximadamente cada 5 minutos.

3 RESULTADOS Y DISCUSIÓN

3.1 Resultados de la Simulación

La simulación realizada en OMNeT ++ permitió evaluar el uso de una red de sensores punto a punto, con los parámetros de consumo de energía del nodo Waspmote, y comprobar el tiempo de vida útil que tendría el nodo, si no se aplica algún tipo de algoritmo, la información se transmite constantemente desde el nodo a la pasarela con intervalos de 4 segundos.

En el gráfico siguiente se muestra el consumo de energía del nodo en función del tiempo.

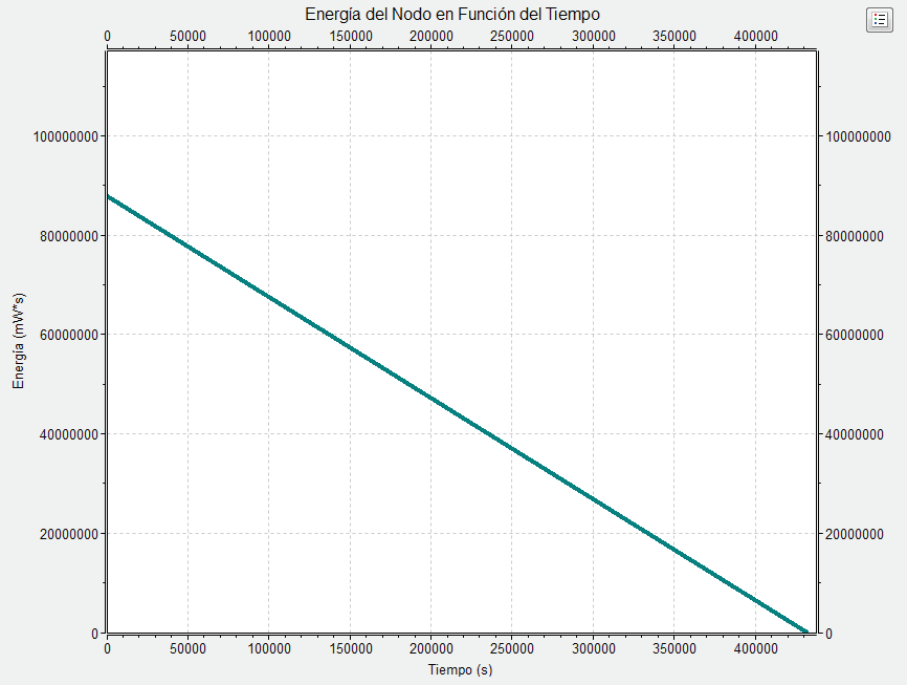


Figura 3.1 Energía del nodo en función del tiempo.

El tiempo máximo de energía del nodo según la simulación es de 432000 segundos equivalente a un tiempo de 5 días.

3.2 Cálculo de la Energía Consumida por el Nodo sin aplicar el Algoritmo

Para tener otro valor de referencia, se realizó el cálculo de la energía que consume un nodo Waspmote, al ejecutar un proceso de trabajo normal, para ello se tomó como referencia el cálculo de consumo eléctrico de un nodo Waspmote en [12], y se aplicó al proyecto planteado, analizando cada etapa del proceso de la aplicación, para determinar el estro de un bovino, así podemos identificar las siguientes etapas generales:

- Activación del nodo sensor
- Encendido del acelerómetro para monitorear la aceleración dinámica del movimiento corporal de la vaca
- Lectura del acelerómetro
- Activación del módulo de comunicaciones
- Transferencia de información inalámbrica a la pasarela, para su almacenamiento en la base de datos y su procesamiento
- Tiempo de espera para la siguiente lectura

Identificadas estas etapas, se procede al cálculo de energía aproximado que consumiría el nodo sensor que se presenta en la tabla 3.1

Tabla 3.1 Consumo de energía por procesos en el nodo Waspmote

Repeticiones por Ciclo	Duración del Proceso	Duración del Ciclo	Corriente Eléctrica del Proceso	Corriente Eléctrica del Nodo	Corriente Total	Carga Energética Total
1	(ms)	(ms)	(mA)	(mA)	(mA)	(mA*ms)
Lectura ACC	3,2	3,2	1	17	18	57,6
Transmisión de datos 802.15.4	8,992	8,992	57,2	17	74,2	667,2064
Nodo en Espera	4000	4000	39,64	17	56,64	226560
		4012,192				227284,8064

Con estos valores podemos calcular el consumo eléctrico del proceso global.

$$Carga\ Energética\ Total\ (mA * ms) = (18mA * 3{,}2ms) + (74{,}2, mA * 8{,}992mS) + (56{,}64mA * 4000ms)$$

$$\boldsymbol{Carga\ Energética\ Total\ (mA * ms) = 227284{,}8064\ mA * ms}$$

Ahora se puede estimar el consumo promedio de corriente, dividiendo la carga energética total, para el tiempo total del proceso, en donde:

$$Consumo\ promedio\ de$$

$$corriente = \frac{Carga\ Energética\ Total}{Tiempo\ Total\ del\ Proceso}$$

Ecuación 3.1 Consumo promedio de corriente

$$Consumo\ promedio\ de\ corriente = \frac{227284{,}8064\ mA * ms}{4012{,}192\ ms}$$

$$Consumo\ promedio\ de\ corriente = 56{,}649\ mA$$

Con la batería Libelium de 6600mAh, calculamos el tiempo de vida máxima de la misma en esta aplicación:

$$Vida\ de\ la\ Batería\ (h) = \frac{Capacidad\ de\ la\ bateria\ (mA * h)}{Consumo\ promedio\ de\ corriente\ (mA)}$$

Ecuación 3.2 Vida de la Batería

$$Vida\ de\ la\ Batería = \frac{6600\ mA * h}{56{,}649\ mA}$$

$$Vida\ de\ la\ Batería = 116{,}508\ h$$

$$Vida\ de\ la\ Batería = 4\ días, 20\ horas\ y\ 30\ minutos$$

Para el cálculo se descartó el consumo del encendido del nodo, acelerómetro y módulo de comunicación pues esto solo se realiza una sola vez y en este cálculo el nodo y sus módulos están en estado activo, por ello se suma el consumo eléctrico de los módulos en el tiempo de espera.

3.3 Cálculo de Energía del Nodo Sensor implementado el Algoritmo Adaptado a la Fisiología Bovina

Para el cálculo del consumo de energía, se consideró que la vaca se puede encontrar en baja actividad (actividades normales de la vaca) o alta actividad (vaca en estro).

Para el cálculo de la energía gastada en cada proceso se usó como referencia la línea de tiempo de la ejecución del programa del nodo sensor en los modos de baja y alta actividad, los valores de los tiempos de ejecución de los subprocesos, así como el consumo eléctrico de estos se obtuvieron en [12].

	PROCESO DE LECTURA P1						PROCESO DE LECTURA P2	PROCESO DE LECTURA P3	.....	PROCESO DE LECTURA P15	CÁLCULO		TRANSMICIÓN			CÁLCULO	MODO DORMIR PROFUNDO		
Tiempo ms	8	2	3,2	1	115	4000	4129,2	4129,2	4129,2	4129,2	0	0	50	7,232	0	0	8	115	300000
DESCRIPCIÓN DEL PROCESO	Despertarse de deep sleep	Encender sensor Acelerómetro ACC	Lectura ACC	Apagar el ACC	Echarse a dormir deepsleep	Modo deep sleep					Validación de Datos ACC	Validación de Ángulo	Proceso de ON módulo 802.15.4	Transmisión de datos 802.15.4	Desactivación 802.15.4	Incremento Número de Muestras n	Despertarse de deep sleep	Echarse a dormir deepsleep	Modo deep sleep

Figura 3.2 Línea de tiempo, modo de baja actividad motriz de la vaca.

	PROCESO DE LECTURA P1						PROCESO DE LECTURA P2	PROCESO DE LECTURA P3	.....	PROCESO DE LECTURA P15	CÁLCULO		TRANSMICIÓN			CÁLCULO
Tiempo ms	8	2	3,2	1	115	4000	4129,2	4129,2	4129,2	4129,2	0	0	50	8,992	0	0
DESCRIPCIÓN DEL PROCESO	Despertarse de deep sleep	Encender sensor Acelerómetro ACC	Lectura ACC	Apagar el ACC	Echarse a dormir deep sleep	Modo deep sleep					Validación de Datos ACC	Validación de Ángulo	Proceso de ON módulo 802.15.4	Transmisión de datos 802.15.4	Desactivación 802.15.4	Incremento Número de Muestras n

Figura 3.3 Línea de tiempo, modo de alta actividad motriz de la vaca.

El tiempo de transmisión se calculó en base el tiempo de retardo en una transición usando como acceso al medio CSMA-CA, todo este tiempo estaría encendido el módulo de transición Ton, y equivale a la suma del tiempo del paquete, más el tiempo de backoff[6] T_{BO} y el tiempo de separación entre tramas ISF [21]. Para el tiempo de backoff se toma una media de 20 símbolos con una duración de 16 μseg. El IFS se calcula considerando 40 símbolos según el encargo y el tiempo de frame se considera el tamaño de la cabecera física de 6 bytes y la cabecera MAC de 13 bytes, más los datos de envió, 82 bytes en alta actividad y 27 bytes en baja actividad.

$$T_{BO} = 16\ slots * 20\ símbolos * 16 \frac{\mu s}{símbolo}$$

$$T_{BO} = 5{,}12\ ms$$

$$IFS = 40 símbolos * 16\ \frac{\mu s}{símbolo}$$

$$IFS = 0{,}64\ ms$$

$$T_{frame\ alta} = \frac{8 * (6 + 13 + 82)}{250\ Kbps}$$

$$T_{frame\ alta} = 3{,}232\ ms$$

[6] Tiempo de Backoff. - tiempo aleatorio de acceso al medio de un nodo.

$$T_{frame\ baja} = \frac{8 * (6 + 13 + 27)}{250\ Kbps}$$

$$T_{frame\ alta} = 1{,}427\ ms$$

$$T_{on\ alta} = (5{,}12 + 0{,}64 + 3{,}232) ms$$

$$T_{on\ alta} = 8{,}992\ ms$$

$$T_{on\ alta} = (5{,}12 + 0{,}64 + 1{,}472)\ ms$$

$$T_{on\ alta} = 7{,}232\ ms$$

Bovino en Baja Actividad. - la actividad de la vaca es baja su actividad motriz es normal, en esta actividad el nodo duerme la mayoría del tiempo, obteniendo los valores en la siguiente tabla, del consumo promedio de los estados del nodo. Para este cálculo se dividió el proceso total en tres subprocesos, el primero de lectura de la aceleración, el segundo la transmisión mediante el módulo de comunicación 802.15.4 y el último proceso el nodo en modo dormir profundo (deep sleep).

Tabla 3.2 Consumo de energía nodo modo de baja actividad motriz de la vaca, subproceso de adquisición de la aceleración

Repeticiones por Ciclo	Duración del Proceso	Duración del Ciclo	Corriente Eléctrica del Proceso	Corriente Eléctrica del Nodo	Corriente Total	Carga Energética Total
15	(ms)	(ms)	(mA)	(mA)	(mA)	(mA*ms)
Despertarse de deep sleep	8	120	9,55	17	26,55	3186
Encender Acelerómetro ACC	2	30	1,1	17	18,1	543
Lectura ACC	3,2	48	1	17	18	864
Apagar el ACC	1	15	0,8	17	17,8	267
Echarse a dormir deep sleep	115	1725	15,6	17	32,6	56235

Modo deep sleep	4000	60000	0,055	0	0,055	3300
		61938				64395

Como observamos en la tabla este proceso de adquisición de la aceleración se realiza cada 4 segundos durante aproximadamente 1 minuto, para después evaluar estos valores y que el algoritmo tome la decisión si el nodo permanece en baja actividad o a cambiado de estado.

Tabla 3.3 Consumo de energía nodo modo de baja actividad motriz de la vaca, subproceso de transmisión de datos

Repeticiones por Ciclo	Duración del Proceso	Duración del Ciclo	Corriente Eléctrica del Proceso	Corriente Eléctrica del Nodo	Corriente Total	Carga Energética Total
1	(ms)	(ms)	(mA)	(mA)	(mA)	(mA*ms)
Proceso de ON módulo 802.15.4	50	50	39,24	17	56,24	2812
Transmisión de datos 802.15.4	7,232	7,232	57,2	17	74,2	536,6144
Desactivación del módulo 802.15.4	0	0	57,2	17	74,2	0
		57,232				3348,6144

Este subproceso solo se ejecuta una ocasión por proceso y el nodo sensor envía un mensaje a la pasarela de que el nodo sigue en la red.

Tabla 3.4 Consumo de energía nodo modo de baja actividad motriz de la vaca, subproceso de nodo durmiendo profundamente

Repeticiones por Ciclo	Duración del Proceso	Duración del Ciclo	Corriente Eléctrica del Proceso	Corriente Eléctrica del Nodo	Corriente Total	Carga Energética Total
1	(ms)	(ms)	(mA)	(mA)	(mA)	(mA*ms)
Despertarse de deep sleep	8	8	9,55	17	26,55	212,4
Echarse a dormir deep sleep	115	115	15,6	17	32,6	3749
Modo deep sleep	300000	300000	0,055	0	0,055	16500
		300123				20461,4

De aquí que la carga de energía total consumida por el nodo un proceso total equivale a la suma de los tres subprocesos, de donde se obtiene la siguiente carga energética total y tiempo total del proceso:

$$Carga\ Energética\ Total\ (mA * ms) = (64395 + 3348{,}6144 + 20461{,}4)\ mA * ms$$

$$\boldsymbol{Carga\ Energética\ Total\ (mA * ms) = 88205{,}0144\ mA * ms}$$

$$Tiempo\ Total\ (ms) = (61938 + 57{,}232 + 300123\ )ms$$

$$\boldsymbol{Tiempo\ Total\ (ms) = 362118{,}232\ ms}$$

Y el consumo total promedio es:

$$Consumo\ promedio\ de\ corriente = \frac{88205{,}0144\ mA * ms}{362118{,}232\ ms}$$

$$Consumo\ promedio\ de\ corriente = 0{,}244\ mA$$

De aquí que la vida útil de la batería podría llegar a ser de:

$$Vida\ de\ la\ Batería = \frac{6600mA * h}{0{,}34mA}$$

$$Vida\ de\ la\ Batería = 27095{,}7422\ h$$

$$Vida\ de\ la\ Batería \approx 1128{,}989\ días$$

Bovino en Alta Actividad. – la vaca se encuentra en estado de estro y su actividad motriz es intensa, cuando el nodo se encuentra en alta actividad y se incrementa la tasa de muestreo y de transmisiones, para este cálculo se dividió el proceso total en dos subprocesos, el primero de lectura de la aceleración y, el segundo la transmisión mediante el módulo de comunicación 802.15.4, para este caso se descarta el tiempo de dormir profundo pues cuando la actividad alta el nodo no ejecuta este modo de operación.

Tabla 3.5 Consumo de energía modo alta actividad motriz de la vaca, subproceso de adquisición de la aceleración

Repeticiones por Ciclo	Duración del Proceso	Duración del Ciclo	Corriente Eléctrica del Proceso	Corriente Eléctrica del Nodo	Corriente Total	Carga Energética Total
75	(ms)	(ms)	(mA)	(mA)	(mA)	(mA*ms)
Despertarse de deep sleep	8	600	9,55	17	26,55	15930
Encender sensor Acelerómetro ACC	2	150	1,1	17	18,1	2715
Lectura ACC	3,2	240	1	17	18	4320
Apagar el ACC	1	75	0,8	17	17,8	1335
Echarse a dormir deep sleep	115	8625	15,6	17	32,6	281175
Modo deep sleep	4000	300000	0,055	0	0,055	16500
		309690				321975

En este modo este subproceso se ejecuta un máximo de 75 ocasiones aproximadamente durante 5 minutos, después de este tiempo el nodo analiza si sigue en un modo de alta intensidad o a cambiado a baja intensidad, si el nodo sigue en alta actividad motriz procede con la transmisión de la medida de tres valores de aceleración.

Tabla 3.6 Consumo de energía nodo modo de alta actividad motriz de la vaca, subproceso de transmisión de datos

Repeticiones por Ciclo	Duración del Proceso	Duración del Ciclo	Corriente Eléctrica del Proceso	Corriente Eléctrica del Nodo	Corriente Total	Carga Energética Total
1	(ms)	(ms)	(mA)	(mA)	(mA)	(mA*ms)
Proceso de ON módulo 802.15.4	50	50	39,24	17	56,24	2812
Transmisión de datos 802.15.4	8,992	8,992	57,2	17	74,2	667,2064
Desactivación del módulo 802.15.4	0	0	57,2	17	74,2	0
		58,992				3479,2064

De aquí que la carga de energía total consumida por el nodo y el tiempo total del proceso en cada interacción es igual a la suma de los dos subprocesos:

$$\boldsymbol{Carga\ Energética\ Total\ (mA * ms) = 346131,98 mA * ms}$$

Y el consumo total promedio es:

$$Consumo\ promedio\ de\ corriente = \frac{346131{,}98 mA * ms}{301192 ms}$$

$$Consumo\ promedio\ de\ corriente = 1{,}149\ mA$$

Y la vida útil de la batería podría llegar a ser de:

$$Vida\ de\ la\ Batería = \frac{6600 mA * h}{1{,}149 mA}$$

$$Vida\ de\ la\ Batería = 5743{,}09\ h$$

$$Vida\ de\ la\ Batería \approx 239\ días$$

Evaluando la posibilidad de que actividades con intensidad media alta pasen a altas, por factores externos como por ejemplo estrés en las vacas, se considera un escenario real donde vamos a tener la mezcla de actividad alta y baja, para ello se analizó los desplazamientos comunes de un animal y se sintetizo en la tabla 3.3.

Tabla 3.7 Actividad motriz de la vaca

ACTIVIDAD MOTRIZ DE LA VACA									
ACTIVIDAD DIARIA	Incidencias (c/día)	Inicio (h:min)	Duración (h:min)	Fin (h:min)	Intensidad de la Actividad	Duración de actividad Leve (h:min)	Duración de Actividad Media (h:min)	Duración de Actividad Media Alta (h:min)	Duración Activida Alta (h:min)
Desplazamiento a Sala de Ordeño	1	03:45	00:15	04:00	2,00			00:15	
Ordeño de la mañana	1	04:00	01:30	05:30	1,00		01:30		
Desplazamiento a Potreros	1	05:30	00:15	05:45	2,00			00:15	
Comida de la Mañana	1	05:45	01:00	06:45	2,00			01:00	
Rumen en la Mañana y Descanzo	1	06:45	05:15	12:00	0,00	04:25			
Lapso de beber agua	5		00:10		1,00		00:50		
Comida de la Tarde	1	12:00	01:00	13:00	2,00			01:00	
Rumen en la Mañana y Descanzo	1	13:00	02:15	15:15	0,00	01:55			**08:00**
Lapso de beber agua	2		00:10		1,00		00:20		
Desplazamiento a Sala de Ordeño	1	15:15	00:15	15:30	2,00			00:15	
Ordeño de la Tarde	1	15:30	01:30	17:00	1,00		01:30		
Desplazamiento a Potreros	1	17:00	00:15	17:15	2,00			00:15	
Comida en la Noche	1	17:15	01:00	18:15	2,00			01:00	
Lapso de beber agua	2		00:10		1,00		00:20		
Descanso o Dormir	1	18:15	09:30	03:45	0,00	09:10			
CICLO ESTRAL	1 c/mes		08:00		3,00				
DURACIÓN ACTIVIDAD DÍA NORMAL						15:30	04:30	04:00	

En base a esta información se procedió a calcular el consumo de energía promedio para un ciclo estral de 21 días, considerando que toda la actividad media alta el sensor la va a tomar como actividad alta, y el consumo de energía será en modo alto, el día de estro se considera como un día de consumo alto y el resto de días se considera cómo días de bajo consumo, así resumimos en la tabla 3.4.

Tabla 3.8 Carga energética según actividad motriz en un ciclo estral

ACTIVIDAD	Duración del Proceso (ms)	Corriente Eléctrica (mA)	Carga Energética Total (mA*ms)
Actividad Leve	1116000000	0,243580705	271836067,3
Actividad Media	324000000	0,243580705	78920148,56
Actividad Media Alta	288000000	1,050703036	302602474,5
Actividad Alta	86400000	1,050703036	90780742,34
DURACIÓN EN 21 DÍAS	1814400000		744139432,6

De aquí que la carga de energía total consumida por el nodo en cada ciclo estral es:

$$\boldsymbol{Carga\ Energética\ Total\ (mA * ms) = 744139432,6\ mA * ms}$$

Y el consumo total promedio es:

$$Consumo\ promedio\ de\ corriente = \frac{744139432{,}6\ mA * ms}{1814400000\ ms}$$

$$Consumo\ promedio\ de\ corriente = 0{,}4101\ mA$$

De aquí que la vida útil de la batería podría llegar a ser de:

$$Vida\ de\ la\ Batería = \frac{6600mA * h}{0{,}4101mA}$$

$$Vida\ de\ la\ Batería = 16092{,}46799\ h$$

$$Vida\ de\ la\ Batería \approx 670\ días\ y\ 12\ horas$$

3.4 Análisis de los Resultados

De los resultados obtenidos se puede afirmar que el algoritmo adaptativo a la fisiología bovina, en específico al análisis de la actividad motriz de la vaca, tiene un resultado positivo y se logra el objetivo de aumentar el tiempo de vida de un nodo en una RSI para monitoreo reproductivo bovino.

Así, al comparar el cálculo obtenido del tiempo de vida de la batería sin el uso del algoritmo y con intervalos entre muestras de 4 segundos, se observa que el nodo tiene un tiempo de vida máximo de 5 días, 20 horas y 44 minutos en la red.

Mientras que con el uso del algoritmo adaptado a la actividad motriz de la vaca y analizado en un ambiente real con un ciclo estral de 21 días el tiempo de vida aproximado del nodo en la red es de 543 días, 1 hora y 34 minutos.

Cualitativamente esto implica que una ganadería tendrá información en tiempo real de la actividad motriz alta de una vaca, y podrá determinar el día en que la misma este óptima a ser fecundada, pues el algoritmo garantiza la transmisión de información cuando la actividad motriz tienen un valor significativo para su análisis.

Si el análisis de estos resultados se estudia desde un punto de vista económico, el ahorro que puede representar a un ganadero que implemente esta red de sensores con el

algoritmo adaptado a la actividad motriz es el que se calcula a continuación en la tabla 3.6, los datos a continuación han sido obtenidos de los gastos operativos de la hacienda San Javier y son similares a los costos en [2].

Tabla 3.9 Costos de una mala detección del estro

Insumos por inseminación	**Valor**
Semen de toro	$ 15,00
Funda de inseminación	$ 0,25
Visita de Técnico Especialista	$ 30,00
Subtotal por Insumos	$ 45,25
Días de Lactación Abiertos	42
Gastos vaca por día abierto	$ 5,00
Subtotal por Días Abiertos	$ 210,00
Total de Gastos	**$ 255,25**

Como se observa los costos que se puede generar por una falla en la determinación incorrecta o a destiempo del estro es de aproximadamente $255,25 por cada ciclo no detectado, si comparamos con el costo comercial del Waspmote que es de $250, podemos afirmar que el sistema se pagaría en un solo mes.

Cualitativamente, este proyecto también presenta una nueva metodología de diseño de redes de sensores e incentiva a otras investigaciones a estudiar el campo de aplicación, para generar programas que se adapten a éstas y optimicen el tiempo de vida de los nodos.

Las interfaces de usuario y códigos desarrollados en la aplicación servirán para el desarrollo de nuevos sistemas. En el proyecto se desarrolló un programa de lectura del puerto USB, este puede ser muy útil para otros proyectos, es simple de utilizar y replicar, además de almacenar directo los datos en una base de datos, desde los cuales pueden ser procesados con cualquier plataforma de desarrollo gráfico.

3.5 Pruebas Prácticas

Las pruebas prácticas se desarrollaron con la implementación de un prototipo de prueba, el mismo que se conformó por una caja de fibra de vidrio que contenía en su interior al nodo sensor Waspmote y que fue colocada a una vaca con un collar; las imágenes que se presentan a continuación corresponden al proceso constructivo de este dispositivo.

Figura 3.4 Caja de fibra de vidrio pintada en color rojo.

Figura 3.5 Manufactura del collar en taller artesanal.

Figura 3.6 Prototipo de prueba con nodo sensor instalado.

En esta figura observamos que existe una base de espuma plástica en el fondo, esta base sirva para orientar el nodo en forma horizontal, pues como vemos en la siguiente imagen el collar forma con el cuello de la vaca un ángulo de aproximadamente 45°.

Figura 3.7 Colocación de collar en la vaca.

Esta fotografía evidencia parte del proceso de construcción del prototipo e inicio de pruebas, los resultados obtenidos en un inicio se analizaron y utilizaron para realizar el algoritmo.

Las pruebas se realizaron en la propiedad San Javier, ubicada en la provincia de Pichincha, cantón Mejía, parroquia El Chaupi, barrio San Manuel, vía Saliquiti S/N, las características topográficas de la propiedad permitieron que la prueba se realice punto a punto, sin la necesidad de ningún tipo de infraestructura adicional, así la siguiente imagen muestra la ubicación de la PC con la pasarela y el área donde se desplazó la vaca en las diferentes pruebas.

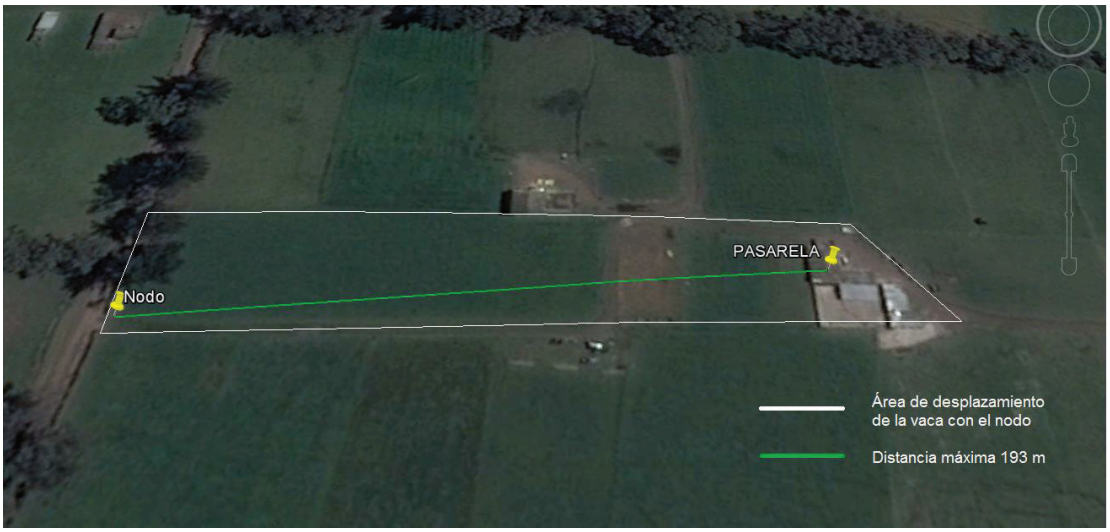


Figura 3.8 Área de pruebas de funcionamiento.

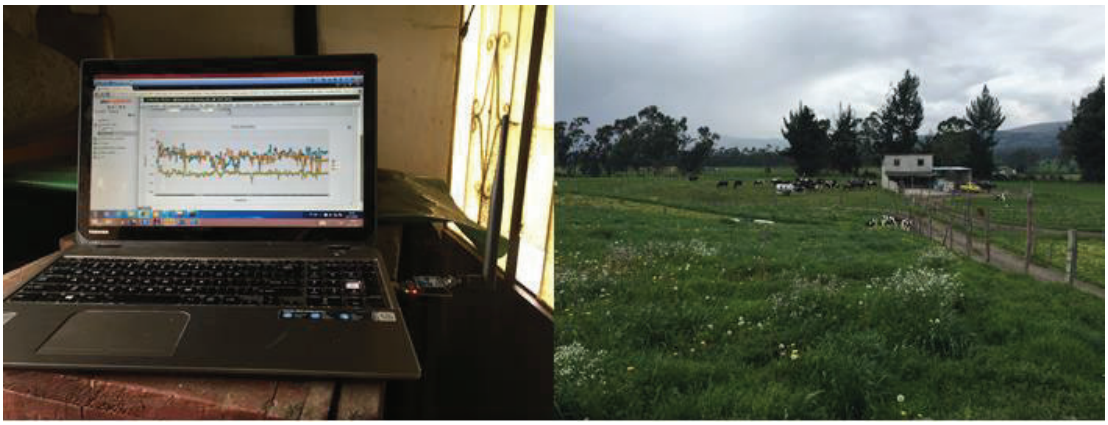


Figura 3.9 Imagen real del área de pruebas, a la izquierda la estación base con la pasarela capturando información, a la derecha una de las vacas lleva el nodo de prueba.

Como se observa la distancia máxima de transmisión en el área de pruebas es de 193 metros, por lo que no existen inconvenientes con el módulo de comunicación del Waspmote que tiene un rango de cobertura de 750 metros.

3.6 Prueba de Monitoreo de la Actividad Motriz

Las primeras pruebas realizadas con el prototipo de prueba fue monitorear la actividad motriz de la vaca, para ello se adquirieron muestras constantemente del sensor acelerómetro, y se obtuvo la gráfica que se muestra a continuación.

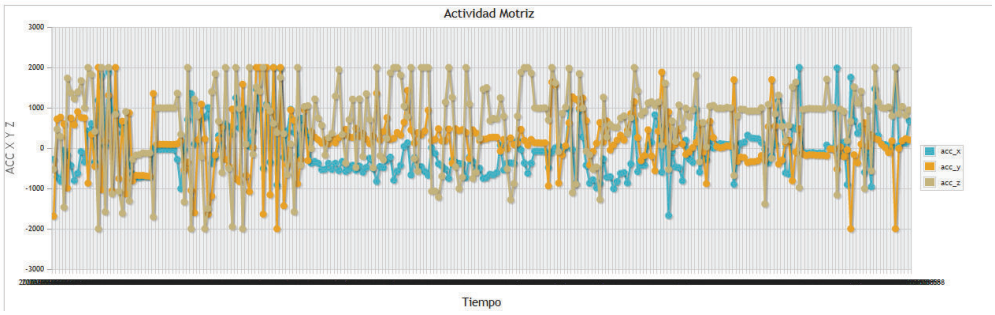


Figura 3.10 Monitoreo de la actividad motriz de la vaca.

De esta gráfica se observó que los valores de picos altos corresponden a una actividad motriz alta en la vaca y se tomó estos valores como umbrales de la actividad motriz en el desarrollo del algoritmo, es importante mencionar también que estos valores nunca sobrepasaron los 2000 mg por lo que en el algoritmo se trabajó con la configuración del sensor acelerómetro que es de 2000 mg.

Así también estas pruebas en campo nos permitieron determinar los umbrales para la comparación de la aceleración producto de la actividad motriz del bovino.

3.7 Prueba de Tiempo de Vida de las Baterías

Para validar el cálculo obtenido en la simulación se procedió a realizar una prueba de transmisión de datos, enviando constantemente el nivel de la batería y poder obtener el tiempo de vida de la misma antes que se descargue completamente, en la Figura 3.11 podemos visualizar la prueba realizada

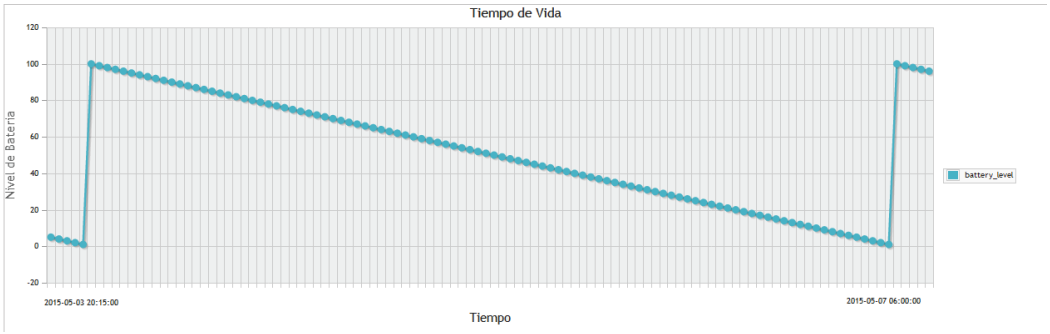


Figura 3.11 Energía de la batería consumida en función del tiempo.

De las pruebas realizadas sobre las baterías del wapsmote se encontró que el tiempo de vida máximo de estas fue de tres días, nueve horas y cuarenta y cinco minutos, tiempo de vida inferior al obtenido por la simulación, pues el simulador no considera los procesos internos de adquisición de datos, sin embargo, los resultados son aceptables.

3.8 Prueba de Tiempo de Vida de las Baterías con el Algoritmo

Dentro de las pruebas prácticas se implementó el prototipo con el nodo sensor programado con el algoritmo y se colocó en una vaca que se conocía que en tres días estaría en apta para ser fecundada, los resultados de esta experimentación fueron positivos pues se pudo recolectar mucha información de la actividad motriz de la vaca y el consumo de energía fue pequeño, así lo podemos visualizar en la imagen que se presenta a continuación.

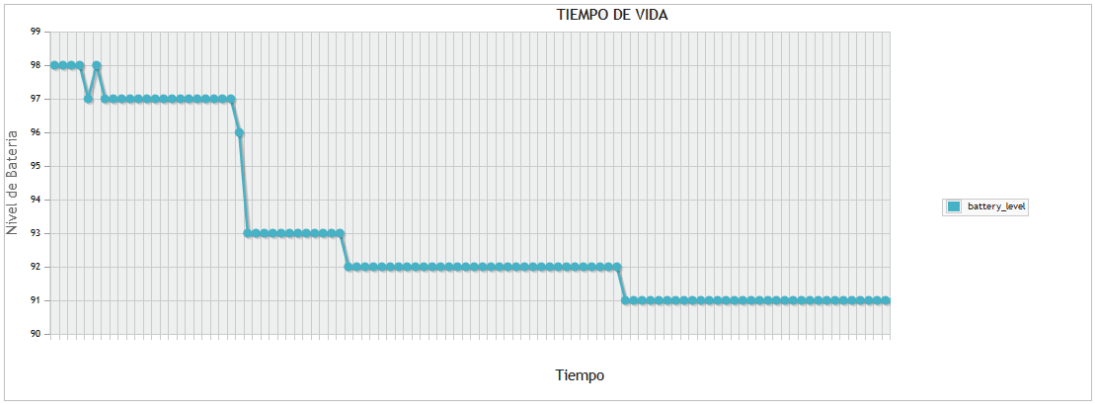


Figura 3.12 Energía consumida por la batería con el algoritmo adaptado a la fisiología bovina.

Como se visualiza este tiempo de vida corresponde a tres días de experimentación en los cuales el nivel de batería descendió de un porcentaje del 93% hasta un valor de energía restante de 91%. Para realizar esta prueba se colocó el collar en una vaca de la cual se conocía la fecha de su posible inseminación, con la finalidad de no realizar una prueba muy larga, así se colocó el collar en la vaca el día 12 de julio de 2016 y las pruebas duraron hasta el 14 de julio, donde se pudo determinar que la actividad motriz era superior lo que implicaba que estaba óptima a ser inseminada, esta fecha coincidió con la fecha estimada por lo que la prueba fue exitosa.

La imagen a continuación presenta los valores de aceleración encontrados en la experimentación, en ellos los primeros dos días la actividad es leve y se presentan falsos positivos pero la cantidad de información indica que no se trata de un evento, mientras que el día 14 de junio de 2016 la actividad motriz es constante y mayor, indicando claramente un evento de celo en la vaca.

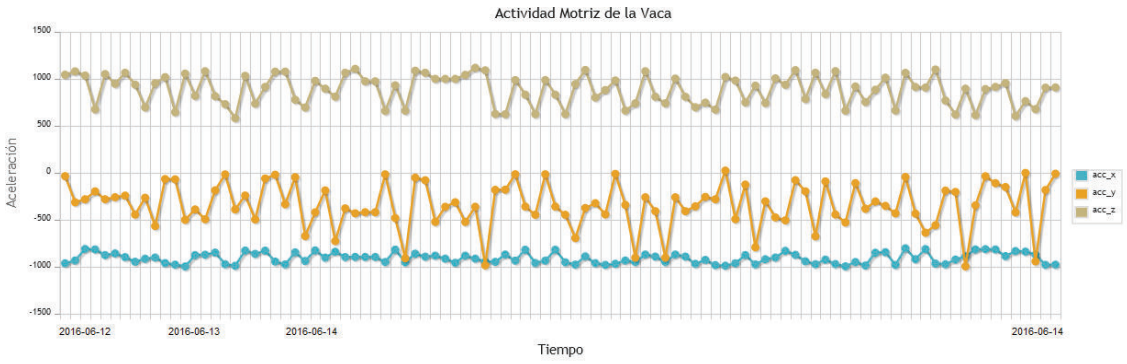


Figura 3.13 Actividad motriz de una vaca lista para ser inseminada.

4 CONCLUSIONES

- El algoritmo adaptado a la fisiología bovina se realizó tomando como referencia el valor de la aceleración dinámica de un cuerpo, mediante el uso del acelerómetro como sensor de adquisición de datos, así con una caracterización de la actividad motriz normal de una vaca se pudo tomar muestras de manera esporádica las mismas que son transmitidas de igual forma, manteniendo al nodo sensor en modo dormido lo que permitió el ahorro de energía.

- El consumo estimado del nodo sensor con el simulador de red en un modo de transmisión continua de datos, fue similar al calculado teóricamente, pero existió una diferencia de aproximadamente catorce horas con el tiempo obtenido experimentalmente, este valor se atribuye a que tanto en el cálculo teórico como en el simulado, no se consideran una batería real las mismas que presentan una curva de descarga que es constante hasta un 80% de la energía restante, y esta curva decrece exponencialmente a partir de este valor.

- El desarrollo del algoritmo adaptado a la actividad motriz de los bovinos permite la optimización del consumo de energía de un nodo en una RSI logrando una autonomía energética calculada teóricamente de 670 días, lo que hace que este tipo de redes puedan ser aplicadas al monitoreo reproductivo de bovinos.

- La metodología escogida para el desarrollo de este trabajo sirve de base para el desarrollo de nuevos proyectos en el campo ganadero, pues sintetizó mucha información dispersa y con un enfoque biológico, haciendo que la misma pueda ser fácilmente interpretada por otros investigadores para nuevos desarrollos.

- De las pruebas realizadas se determinó que los valores de la aceleración que más influyen para este sistema son los valores de los ejes X y Y, mientras que en el eje Z no hay cambios significativos, por ello el técnico veterinario que analice esta información se debe concentrar únicamente en los cambios en estos ejes.

- El uso de una red IEEE 802.15.4 fue adecuada para el proyecto de monitoreo reproductivo bovino, con el módulo de comunicación de Libelium se transmitió hasta 193 metros de distancia sin inconvenientes, esto también a que el lugar donde se realizó las pruebas es completamente plano, y no existe interferencias.

- En este estudio se tomó como única variable la actividad motriz de la vaca, sin embargo, se podría realizar trabajos futuros en los cuales se añada la temperatura corporal del bovino, pues este parámetro también es un indicador de estro y

adicionalmente se puede utilizar este dato para monitorear el estado de salud del animal.

- El área geográfica destinada para realizar las pruebas del prototipo poseía una topografía plana e inclusive la pasarela se encontraba ubicada en el punto más alto del área de pruebas, razón por la cual no se necesitó implementar ningún tipo de infraestructura adicional, en casos donde no exista línea de vista o existan obstrucciones en el medio se debería implementar el sistema con uno o varios nodos fijos que funcionan de repetidores en una topología tipo árbol.
- El nodo sensor debe estar ubicado en el interior del collar de manera horizontal, una vez colocado el mismo en la vaca, pues este debe formar un ángulo de cero grados con el eje horizontal, si esta condición no se cumple el proceso de validación del ángulo cuando la vaca come es erróneo.
- En un trabajo futuro el valor de este ángulo podría ser calculado por software y corregido mediante un offset al valor, porque el valor que adquiere el sensor acelerómetro es el valor de la componente en el eje X, Y y Z; así también existe la posibilidad de reducir el tamaño del nodo y colocar el mismo en la pata de la vaca a manera de una pulsera.
- El beneficio económico que representa el uso de la RSI para el sector ganadero es muy rentable, pues se reducirá el número de días abiertos producto de una mala determinación del ciclo estral de la vaca.
- El uso de nuevas tecnologías en el sector ganadero además de optimizar los recurso económicos y humanos, sirven para brindar un adecuado confort y calidad de vida a los animales de granja, así se pueden prevenir o curar inmediatamente enfermedades, pues con ciertas modificaciones al algoritmo planteado podemos determinar la actividad diaria de los animales y saber si estos se encuentran gozando de buena salud o están decaídos, así también se puede monitorear el rumen y determinar si están ingiriendo suficiente alimento.

5 REFERENCIAS BIBLIOGRÁFICAS

[1] M. DeJarnette y R. Nebel, “Anatomía y Fisiología de la Reproducción Bovina,” [En línea]. Disponible: http://www.selectsires.com/dairy/SpanResources/reproductive_anatomy_spanish.pdf?version=20170404.

[2] G. A. Revelo López, “Evaluación del desempeño reproductivo del hato lechero de la Hacienda "Sandial" localizada en el cantón Montufar, provincia del Carchi en el período 2011-2013,” Universidad San Francisco de Quito, Quito, Ecuador, 2013.

[3] Akyildiz y M. Vuran, “Wireless Sensor Netwoks,” 1st ed. Uniet kingdom, Wiley, 2010.

[4] D. Gastón, “Sustainability in Colombia using sensor networks,” Colombia, 2016.

[5] National Instruments, Redes de Sensores Inalámbricos, [en línea], Disponible: http://www.ni.com/wsn/whatis/esa/#h11.

[6] *Waspmote Datasheet*, Libelium.

[7] *Waspmote IDE User Guide*, Libelium.

[8] Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Std. 802.11, 2012.

[9] *Low-Rate Wireless Networks,* IEEE Std. 802.15.4, 2015

[10] *ZigBee Specification,* ZigBee Alliance.

[11] *Wireless Personal Area Network standard based on the Bluetooth,* IEEE Std. 802.15.1, 2002.

[12] L. I. Florentín, “Evaluación y caracterización experimental del consumo energético del dispositivo Waspmote para redes sensoriales y de comunicaciones,” Tesis de Master, Universidad de Zaragoza, España, Zaragoza, 2014.

[13] M. DeJarnette y R. Nebel, “Anatomía y Fisiología de la Reproducción Bovina,” [En línea]. Disponible: http://www.selectsires.com/dairy/SpanResources/reproductive_anatomy_spanish.pdf?version=20170404.

[14] C. Rippe, “El ciclo estral,” en Dairy Cattle Reproduction Conference, Minneapolis, 2009.

[15] S. J. Roberts y K. McEntee, “La guía visual de reproducción bovina,” [En línea]. Disponible: http://www.drostproject.org/sp_bovrep/guide.html

[16] M. Andringa, D. Cavestany, Eerdenburg “Relaciones entre la expresión de celo, tamaño del folículo y ovulación en vacas de leche en pastoreo,” SMVU, vol. 49, no.190, pp. 04 a 15, 2013.

[17] Arce, A.I.C., A.R.B. Tech, A.C.S. Silva e E.J.X. Costa, “Monitorización de rebaños de bovinos a través de redes de sensores inalámbricos”, Facultad de Zootecnia e Ingeniería de Alimentos, Universidad de San Paulo, Brasil, CEP 13635-900, Jun. 2009.

[18] K. Kwong, H. Goh, C. Michie, I. Andonovic, B. Stephen, T. Mottran, “Wireless Sensor Networks for Beef and Dairy Herd Management,” in ASABE Annual International Meeting Sponsored by ASABE, Rhode Island, 2008.

[19] Andras Vargas, “Documentación y tutoriales,” [En línea]. Disponible: https://omnetpp.org/documentation.

[20] Andras Vargas, “INET Framework,” [En línea]. Disponible: https://inet.omnetpp.org/.

[21] D. Losada, “Rendimiento de los protocolos IEEE 802.15.4 en entornos ruidosos,” Tesis de Ingeniería, Universidad Politécnica de Cataluña, España, Cataluña, 2010.

6 ANEXOS

ANEXO I PROGRAMA DE LECTURA DEL PUERTO SERIAL

```
//API NODE CONTROLLER
//NodeEventListener

package api.node.controller;
import api.node.model.util.EventDatabase;
import api.node.model.util.PostRequest;
import api.node.model.var.NodeEvent;
import com.google.gson.Gson;
import gnu.io.CommPort;
import gnu.io.CommPortIdentifier;
import gnu.io.PortInUseException;
import gnu.io.SerialPort;
import gnu.io.SerialPortEvent;
import gnu.io.SerialPortEventListener;
import gnu.io.UnsupportedCommOperationException;
import java.io.IOException;
import java.io.InputStream;
import java.io.OutputStream;
import java.io.UnsupportedEncodingException;
import java.sql.SQLException;
import java.util.Enumeration;
import java.util.TooManyListenersException;
public class NodeEventListener implements SerialPortEventListener {
    private SerialPort serialPort = null;
    private InputStream input = null;
    private OutputStream output = null;
    private String service;
    private PostRequest postRequest;
    private String ipAddress;
    private String tcpPort;
    private boolean receive = false;
    private char data;
    private String frame;
    private EventDatabase database = new EventDatabase();
    private static final int TIMEOUT = 5000;
    public NodeEventListener(String service) throws IllegalAccessException,
InstantiationException, ClassNotFoundException {
        this.service = service;
    }
    public String getService() {
        return this.service;
    }
    public void setService(String service) {
        this.service = service;
    }
    public void webConfig(String address, String port) {
        this.ipAddress = address;
        this.tcpPort = port;
        String host = this.ipAddress + ":" + this.tcpPort;
        String path = "api/project/event/set";
        this.postRequest = new PostRequest(host, path);
    }
    public CommPortIdentifier searchPort(String portName) {
        Enumeration ports = CommPortIdentifier.getPortIdentifiers();
        CommPortIdentifier port;
        do {
            if(!ports.hasMoreElements()) {
                return null;
            }
            port = (CommPortIdentifier)ports.nextElement();
        } while(port.getPortType() != 1 || !port.getName().equals(portName));
```

```
        return port;
    }
    public void connect(String selectedPort, Integer BaudRate) throws
PortInUseException, UnsupportedCommOperationException {
        CommPortIdentifier selectedPortIdentifier =
this.searchPort(selectedPort);
        CommPort commPort = selectedPortIdentifier.open(selectedPort, 5000);
        this.serialPort = (SerialPort)commPort;
        this.serialPort.setSerialPortParams(BaudRate.intValue(), 8, 1, 0);
    }
    public void disconnect() throws IOException {
        this.serialPort.removeEventListener();
        this.serialPort.close();
        this.input.close();
        this.output.close();
    }
    public void initIOStream() throws IOException {
        this.input = this.serialPort.getInputStream();
        this.output = this.serialPort.getOutputStream();
    }
    public void initListener() throws TooManyListenersException {
        this.serialPort.addEventListener(this);
        this.serialPort.notifyOnDataAvailable(true);
    }
    public void serialEvent(SerialPortEvent serialPortEvent) {
        if(serialPortEvent.getEventType() == 1) {
            try {
                this.data = (char)this.input.read();
            } catch (IOException var8) {
                var8.printStackTrace();
            }
            System.out.print(this.data);
            if(this.data == 60) {
                this.frame = "";
                this.receive = true;
            }
            if(this.receive) {
                this.frame = this.frame + this.data;
            }
            if(this.data == 63 & this.receive) {
                this.receive = false;
                System.out.println();
                System.out.println("API: A event was received");
                NodeEvent event = NodeEvent.Of(this.frame);
                System.out.println("API: " + event);
                if(this.service.equals("DB")) {
                    try {
                        this.database.insertEvent(event);
                        System.out.println("API: Event was recorded in the
database");
                    } catch (SQLException var7) {
                        var7.printStackTrace();
                    }
                } else if(this.service.equals("WEB")) {
                    Gson gson = new Gson();
                    String data = gson.toJson(event);
                    this.postRequest.setMessage(data);
                    try {
                        this.postRequest.query();
                        if(this.postRequest.getStatusCode().intValue() == 200)
{
                            System.out.println("API: Event was sent to the
server");
                        } else {
                            System.out.println("API: Failure sending, Status
Code " + this.postRequest.getStatusCode());
                        }
```

```
                        } catch (UnsupportedEncodingException var6) {
                            var6.printStackTrace();
                        }
                    }
                }
            }
        }
}

//API NODE CONTROLLER
//ProgramInterface

package api.node.controller;
import api.node.controller.NodeEventListener;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.io.InputStream;
import java.util.Properties;
public class ProgramInterface {
    private Properties config = new Properties();
    public ProgramInterface() throws FileNotFoundException {
        InputStream propStream =
this.getClass().getClassLoader().getResourceAsStream("conf/config.properties");
        if(propStream != null) {
            try {
                this.config.load(propStream);
            } catch (IOException var3) {
                var3.printStackTrace();
            }
        } else {
            throw new FileNotFoundException("Property file not found in the
classpath");
        }
    }
    public void run() {
        String service = this.config.getProperty("service");
        String serialPort = this.config.getProperty("com_port");
        Integer baudRate = Integer.valueOf(this.config.getProperty("com_baud"));
        String ipAddress = this.config.getProperty("ip_address");
        String tcpPort = this.config.getProperty("tcp_port");
        System.out.println("Welcome to the Node API v2.0");
        try {
            NodeEventListener communicator = new NodeEventListener(service);
            communicator.webConfig(ipAddress, tcpPort);
            communicator.connect(serialPort, baudRate);
            communicator.initIOStream();
            communicator.initListener();
            System.out.println("API run on the port " + serialPort);
            System.out.println("API work on service " + service);
            if(communicator.getService().equals("WEB")) {
                System.out.println("IP address " + ipAddress + " / TCP port " +
tcpPort);
            }
        } catch (Exception var8) {
            var8.printStackTrace();
        }
    }
}

//API NODE MODEL UTIL
//EventDataBase

package api.node.model.util;
import api.node.model.var.AccValue;
import api.node.model.var.CowData;
import api.node.model.var.NodeEvent;
```

```java
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.format.DateTimeFormatter;
import java.util.ArrayList;
import java.util.Iterator;
public class EventDatabase {
    private String host = "jdbc:mysql://localhost:3306/event_database";
    private String username = "root";
    private String password = "";
    public EventDatabase() throws ClassNotFoundException,
IllegalAccessException, InstantiationException {
        Class.forName("com.mysql.jdbc.Driver").newInstance();
    }
    public void insertCow(CowData data) throws SQLException {
        Connection connection = DriverManager.getConnection(this.host,
this.username, this.password);
        String sql = "INSERT INTO cow SET cow_name=\'" + data.getName() +
"\',node_id=\'" + data.getNodeId() + "\',birth_date=\'" + data.getBirthDate() +
"\',insemination_date=" + (data.getInseminationDate() != null?"\'" +
data.getInseminationDate() + "\'":"NULL") + ",labor_date=" + (data.getLaborDate()
!= null?"\'" + data.getLaborDate() + "\'":"NULL") + ",next_date=" +
(data.getNextDate() != null?"\'" + data.getNextDate() + "\'":"NULL");
        PreparedStatement statement = connection.prepareStatement(sql);
        statement.executeUpdate();
    }
    public void updateCow(CowData data) throws SQLException {
        Connection connection = DriverManager.getConnection(this.host,
this.username, this.password);
        String sql = "UPDATE cow SET node_id=\'" + data.getNodeId() +
"\',birth_date=\'" + data.getBirthDate() + "\',insemination_date=" +
(data.getInseminationDate() != null?"\'" + data.getInseminationDate() +
"\'":"NULL") + ",labor_date=" + (data.getLaborDate() != null?"\'" +
data.getLaborDate() + "\'":"NULL") + ",next_date=" + (data.getNextDate() !=
null?"\'" + data.getNextDate() + "\'":"NULL") + " WHERE cow_name=\'" +
data.getName() + "\'";
        PreparedStatement statement = connection.prepareStatement(sql);
        statement.executeUpdate();
    }
    public void deleteCow(String cowName) throws SQLException {
        Connection connection = DriverManager.getConnection(this.host,
this.username, this.password);
        String sql = "DELETE FROM cow WHERE cow_name=\'" + cowName + "\'";
        PreparedStatement statement = connection.prepareStatement(sql);
        statement.executeUpdate();
    }
    public ArrayList<CowData> selectCow() throws SQLException {
        Connection connection = DriverManager.getConnection(this.host,
this.username, this.password);
        String sql = "SELECT * FROM cow";
        PreparedStatement statement = connection.prepareStatement(sql);
        ResultSet result = statement.executeQuery();
        ArrayList dataList;
        CowData data;
        for(dataList = new ArrayList(); result.next(); dataList.add(data)) {
            data = new CowData(result.getString("cow_name"),
result.getString("node_id"), LocalDate.parse(result.getString("birth_date")));
            String labor = result.getString("labor_date");
            if(labor != null) {
                data.setLaborDate(LocalDate.parse(labor));
            } else {
                data.setLaborDate((LocalDate)null);
            }
```

```
            String insemination = result.getString("insemination_date");
            if(insemination != null) {
                data.setInseminationDate(LocalDate.parse(insemination));
            } else {
                data.setInseminationDate((LocalDate)null);
            }
            String next = result.getString("next_date");
            if(next != null) {
                data.setNextDate(LocalDate.parse(next));
            } else {
                data.setNextDate((LocalDate)null);
            }
        }
        return dataList;
    }
    public ArrayList<CowData> selectCow(String cowName) throws SQLException {
        Connection connection = DriverManager.getConnection(this.host,
this.username, this.password);
        String sql = "SELECT * FROM cow WHERE cow_name=\'" + cowName + "\'";
        PreparedStatement statement = connection.prepareStatement(sql);
        ResultSet result = statement.executeQuery();
        if(result.next()) {
            CowData data = new CowData(result.getString("cow_name"),
result.getString("node_id"), LocalDate.parse(result.getString("birth_date")));
            String labor = result.getString("labor_date");
            if(labor != null) {
                data.setLaborDate(LocalDate.parse(labor));
            } else {
                data.setLaborDate((LocalDate)null);
            }
            String insemination = result.getString("insemination_date");
            if(insemination != null) {
                data.setInseminationDate(LocalDate.parse(insemination));
            } else {
                data.setInseminationDate((LocalDate)null);
            }
            String next = result.getString("next_date");
            if(next != null) {
                data.setNextDate(LocalDate.parse(next));
            } else {
                data.setNextDate((LocalDate)null);
            }
            ArrayList container = new ArrayList(1);
            container.add(data);
            return container;
        } else {
            return null;
        }
    }
    public void insertEvent(NodeEvent event) throws SQLException {
        Connection connection = DriverManager.getConnection(this.host,
this.username, this.password);
        DateTimeFormatter formatter = DateTimeFormatter.ofPattern("yyyy-MM-dd
HH:mm:ss");
        String timestamp = event.getTimestamp().format(formatter);
        String sql = "INSERT INTO event (timestamp, node_id, battery_level)
VALUES (\'" + timestamp + "\',\'" + event.getNodoId() + "\'," +
event.getBatteryLevel() + ")";
        PreparedStatement statement = connection.prepareStatement(sql);
        statement.executeUpdate();
        Iterator var7 = event.getAccValues().iterator();
        while(var7.hasNext()) {
            AccValue value = (AccValue)var7.next();
            sql = "INSERT INTO data (timestamp, acc_x, acc_y, acc_z) VALUES (\'"
+ timestamp + "\'," + value.getAccX() + "," + value.getAccY() + "," +
value.getAccZ() + ")";
            statement = connection.prepareStatement(sql);
```

```
            statement.executeUpdate();
        }
    }
    public ArrayList<NodeEvent> selectBase(String sql) throws SQLException {
        ArrayList resultEvents = new ArrayList();
        Connection connection = DriverManager.getConnection(this.host,
this.username, this.password);
        PreparedStatement statement = connection.prepareStatement(sql);
        ResultSet result = statement.executeQuery();
        while(result.next()) {
            NodeEvent event = new
NodeEvent(result.getTimestamp("timestamp").toLocalDateTime());
            event.setNodoId(result.getString("node_id"));

event.setBatteryLevel(Integer.valueOf(result.getInt("battery_level")));
            String stamp =
event.getTimestamp().format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss"));
            sql = "SELECT * FROM data WHERE timestamp = \'" + stamp + "\'";
            statement = connection.prepareStatement(sql);
            ResultSet _result = statement.executeQuery();
            while(_result.next()) {
                AccValue value = new AccValue();
                value.setAccX(Integer.valueOf(_result.getInt("acc_x")));
                value.setAccY(Integer.valueOf(_result.getInt("acc_y")));
                value.setAccZ(Integer.valueOf(_result.getInt("acc_z")));
                event.getAccValues().add(value);
            }
            resultEvents.add(event);
        }
        return resultEvents.isEmpty()?null:resultEvents;
    }
    public ArrayList<NodeEvent> selectEvent() throws SQLException {
        return this.selectBase("SELECT * FROM event");
    }
    public ArrayList<NodeEvent> selectEvent(String nodeId) throws SQLException {
        return this.selectBase("SELECT * FROM event WHERE node_id = \'" + nodeId
+ "\'");
    }
    public ArrayList<NodeEvent> selectEvent(LocalDateTime startTimestamp,
LocalDateTime endTimestamp) throws SQLException {
        String _startTimestamp =
startTimestamp.format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss"));
        String _endTimestamp =
endTimestamp.format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss"));
        return this.selectBase("SELECT * FROM event WHERE timestamp >= \'" +
_startTimestamp + "\' AND timestamp <= \'" + _endTimestamp + "\'");
    }
    public ArrayList<NodeEvent> selectEvent(Integer lastEvents) throws
SQLException {
        return this.selectBase("SELECT * FROM event ORDER BY timestamp DESC
LIMIT " + lastEvents);
    }
    public ArrayList<NodeEvent> selectEvent(String nodeId, LocalDateTime
startTimestamp, LocalDateTime endTimestamp) throws SQLException {
        String _startTimestamp =
startTimestamp.format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss"));
        String _endTimestamp =
endTimestamp.format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss"));
        return this.selectBase("SELECT * FROM event WHERE timestamp >= \'" +
_startTimestamp + "\' AND timestamp <= \'" + _endTimestamp + "\' AND node_id =
\'" + nodeId + "\'");
    }
    public ArrayList<NodeEvent> selectEvent(String nodeId, Integer lastEvents)
throws SQLException {
        return this.selectBase("SELECT * FROM event WHERE node_id = \'" + nodeId
+ "\' ORDER BY timestamp DESC LIMIT " + lastEvents);
    }
```

```
    public ArrayList<NodeEvent> selectEvent(LocalDateTime startTimestamp,
LocalDateTime endTimestamp, Integer lastEvents) throws SQLException {
        String _startTimestamp =
startTimestamp.format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss"));
        String _endTimestamp =
endTimestamp.format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss"));
        return this.selectBase("SELECT * FROM event WHERE timestamp >= \'" +
_startTimestamp + "\' AND timestamp <= \'" + _endTimestamp + "\' ORDER BY
timestamp DESC LIMIT " + lastEvents);
    }
    public ArrayList<NodeEvent> selectEvent(String nodeId, LocalDateTime
startTimestamp, LocalDateTime endTimestamp, Integer lastEvents) throws
SQLException {
        String _startTimestamp =
startTimestamp.format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss"));
        String _endTimestamp =
endTimestamp.format(DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss"));
        return this.selectBase("SELECT * FROM event WHERE timestamp >= \'" +
_startTimestamp + "\' AND timestamp <= \'" + _endTimestamp + "\' AND node_id =
\'" + nodeId + "\' ORDER BY timestamp DESC LIMIT " + lastEvents);
    }
}

//API NODE MODEL UTIL
//GetRequest

package api.node.model.util;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.Iterator;
import java.util.LinkedHashMap;
import org.apache.http.HttpResponse;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClientBuilder;
public class GetRequest {
    private String host;
    private String path;
    private LinkedHashMap<String, String> valueSet = new LinkedHashMap();
    private Integer statusCode;
    private String response;
    public GetRequest(String host, String path) {
        this.host = host;
        this.path = path;
    }
    public String getHost() {
        return this.host;
    }
    public void setHost(String host) {
        this.host = host;
    }
    public String getPath() {
        return this.path;
    }
    public void setPath(String path) {
        this.path = path;
    }
    public LinkedHashMap<String, String> getValueSet() {
        return this.valueSet;
    }
    public void setValueSet(LinkedHashMap<String, String> valueSet) {
        this.valueSet = valueSet;
    }
    public Integer getStatusCode() {
        return this.statusCode;
```

```
    }
    public String getResponse() {
        return this.response;
    }
    public void query() {
        String url = "http://" + this.host + "/" + this.path;
        if(!this.valueSet.isEmpty()) {
            boolean request = true;
            Iterator client = this.valueSet.keySet().iterator();
            label46:
            while(true) {
                String e;
                do {
                    if(!client.hasNext()) {
                        break label46;
                    }
                    e = (String)client.next();
                } while(!((String)this.valueSet.get(e)).equals("") &&
this.valueSet.get(e) != null);
                url = url + (request?"?":"&");
                url = url + e + "=" + (String)this.valueSet.get(e);
                request = false;
            }
        }
        HttpGet request1 = new HttpGet(url);
        request1.addHeader("User-Agent", "User-Agent");
        CloseableHttpClient client1 = HttpClientBuilder.create().build();
        try {
            HttpResponse e1 = client1.execute(request1);
            this.statusCode =
Integer.valueOf(e1.getStatusLine().getStatusCode());
            this.response = null;
            if(this.statusCode.intValue() == 200) {
                BufferedReader buffered = new BufferedReader(new
InputStreamReader(e1.getEntity().getContent()));
                StringBuilder result = new StringBuilder();
                String line;
                while((line = buffered.readLine()) != null) {
                    result.append(line);
                }
                this.response = result.toString();
            }
        } catch (IOException var8) {
            this.statusCode = Integer.valueOf(400);
        }
    }
}


//API NODE MODEL UTIL
//PostRequest

package api.node.model.util;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import org.apache.http.HttpResponse;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.CloseableHttpClient;
import org.apache.http.impl.client.HttpClientBuilder;
public class PostRequest {
    private String host;
    private String path;
    private String message = "{}";
```

```
    private Integer statusCode;
    private String response;
    public PostRequest(String host, String path) {
        this.host = host;
        this.path = path;
    }
    public String getHost() {
        return this.host;
    }
    public void setHost(String host) {
        this.host = host;
    }
    public String getPath() {
        return this.path;
    }
    public void setPath(String path) {
        this.path = path;
    }
    public String getMessage() {
        return this.message;
    }
    public void setMessage(String message) {
        this.message = message;
    }
    public Integer getStatusCode() {
        return this.statusCode;
    }
    public String getResponse() {
        return this.response;
    }
    public void query() throws UnsupportedEncodingException {
        String url = "http://" + this.host + "/" + this.path;
        StringEntity entity = new StringEntity(this.message);
        HttpPost request = new HttpPost(url);
        request.setHeader("Content-Type", "application/json");
        request.setEntity(entity);
        CloseableHttpClient client = HttpClientBuilder.create().build();
        try {
            HttpResponse e = client.execute(request);
            this.statusCode =
Integer.valueOf(e.getStatusLine().getStatusCode());
            this.response = null;
            if(this.statusCode.intValue() == 200) {
                BufferedReader buffered = new BufferedReader(new
InputStreamReader(e.getEntity().getContent()));
                StringBuilder result = new StringBuilder();
                String line;
                while((line = buffered.readLine()) != null) {
                    result.append(line);
                }
                this.response = result.toString();
            }
        } catch (IOException var9) {
            this.statusCode = Integer.valueOf(400);
        }
    }
}

//API NODE MODEL VAR
//AccValue

package api.node.model.var;
public class AccValue {
    private Integer accX = Integer.valueOf(0);
    private Integer accY = Integer.valueOf(0);
    private Integer accZ = Integer.valueOf(0);
```

```
    private Integer accR = Integer.valueOf(0);
    public AccValue() {
    }
    public AccValue(Integer accX, Integer accY, Integer accZ) {
        this.accX = accX;
        this.accY = accY;
        this.accZ = accZ;
        this.accR = this.buildR();
    }
    private Integer buildR() {
        return this.accR =
Integer.valueOf((int)Math.sqrt(Math.pow((double)this.accX.intValue(), 2.0D) +
Math.pow((double)this.accY.intValue(), 2.0D) +
Math.pow((double)this.accZ.intValue(), 2.0D)));
    }
    public Integer getAccX() {
        return this.accX;
    }
    public void setAccX(Integer accX) {
        this.accX = accX;
        this.accR = this.buildR();
    }
    public Integer getAccY() {
        return this.accY;
    }
    public void setAccY(Integer accY) {
        this.accY = accY;
        this.accR = this.buildR();
    }
    public Integer getAccZ() {
        return this.accZ;
    }
    public void setAccZ(Integer accZ) {
        this.accZ = accZ;
        this.accR = this.buildR();
    }
    public Integer getAccR() {
        return this.accR;
    }
    public String toString() {
        return "AccValue{accX=" + this.accX + ", accY=" + this.accY + ", accZ="
+ this.accZ + ", accR=" + this.accR + '}';
    }
}

//API NODE MODEL VAR
//CowData

package api.node.model.var;
import java.time.LocalDate;
public class CowData {
    private String name;
    private String nodeId;
    private LocalDate birthDate;
    private LocalDate inseminationDate;
    private LocalDate laborDate;
    private LocalDate nextDate;
    public CowData(String name, String nodeId, LocalDate birthDate) {
        this.name = name;
        this.nodeId = nodeId;
        this.birthDate = birthDate;
    }
    public CowData(String name, String nodeId, LocalDate birthDate, LocalDate
inseminationDate, LocalDate laborDate, LocalDate nextDate) {
        this.name = name;
        this.nodeId = nodeId;
```

```
        this.birthDate = birthDate;
        this.inseminationDate = inseminationDate;
        this.laborDate = laborDate;
        this.nextDate = nextDate;
    }
    public String getName() {
        return this.name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getNodeId() {
        return this.nodeId;
    }
    public void setNodeId(String nodeId) {
        this.nodeId = nodeId;
    }
    public LocalDate getBirthDate() {
        return this.birthDate;
    }
    public void setBirthDate(LocalDate birthDate) {
        this.birthDate = birthDate;
    }
    public LocalDate getInseminationDate() {
        return this.inseminationDate;
    }
    public void setInseminationDate(LocalDate inseminationDate) {
        this.inseminationDate = inseminationDate;
    }
    public LocalDate getLaborDate() {
        return this.laborDate;
    }
    public void setLaborDate(LocalDate laborDate) {
        this.laborDate = laborDate;
    }
    public LocalDate getNextDate() {
        return this.nextDate;
    }
    public void setNextDate(LocalDate nextDate) {
        this.nextDate = nextDate;
    }
    public String toString() {
        return "CowData{name=\'" + this.name + '\'' + ", nodeId=\'" +
this.nodeId + '\'' + ", birthDate=" + this.birthDate + ", inseminationDate=" +
this.inseminationDate + ", laborDate=" + this.laborDate + ", nextDate=" +
this.nextDate + '}';
    }
}

//API NODE MODEL VAR
//DailyEvent

package api.node.model.var;
import api.node.model.var.AccValue;
import java.time.LocalDate;
import java.util.ArrayList;
import java.util.Iterator;
public class DailyEvent {
    private LocalDate day;
    private String nodeId;
    private AccValue accMean = new AccValue();
    public DailyEvent(LocalDate day) {
        this.day = day;
    }
    public DailyEvent(LocalDate day, String nodeId, AccValue accMean) {
        this.day = day;
```

```
        this.nodeId = nodeId;
        this.accMean = accMean;
    }
    public LocalDate getDay() {
        return this.day;
    }
    public void setDay(LocalDate day) {
        this.day = day;
    }
    public String getNodeId() {
        return this.nodeId;
    }
    public void setNodeId(String nodeId) {
        this.nodeId = nodeId;
    }
    public AccValue getAccMean() {
        return this.accMean;
    }
    public void setAccMean(AccValue accMean) {
        this.accMean = accMean;
    }
    public void addAccMean(ArrayList<AccValue> accValues) {
        long x = 0L;
        long y = 0L;
        long z = 0L;
        AccValue value;
        for(Iterator var8 = accValues.iterator(); var8.hasNext(); z +=
(long)value.getAccZ().intValue()) {
            value = (AccValue)var8.next();
            x += (long)value.getAccX().intValue();
            y += (long)value.getAccY().intValue();
        }
        x /= (long)accValues.size();
        y /= (long)accValues.size();
        z /= (long)accValues.size();

this.accMean.setAccX(Integer.valueOf((int)((long)this.accMean.getAccX().intValue(
) + x) / 2));

this.accMean.setAccY(Integer.valueOf((int)((long)this.accMean.getAccY().intValue(
) + y) / 2));

this.accMean.setAccZ(Integer.valueOf((int)((long)this.accMean.getAccZ().intValue(
) + z) / 2));
    }
    public String toString() {
        return "DailyEvent{day=" + this.day + ", nodeId=\'" + this.nodeId + '\''
+ ", accMean=" + this.accMean + '}';
    }
}

//API NODE MODEL VAR
//NodeEvent

package api.node.model.var;
import api.node.model.var.AccValue;
import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.StringTokenizer;
public class NodeEvent {
    private LocalDateTime timestamp;
    private String nodoId;
    private Integer batteryLevel;
    private ArrayList<AccValue> accValues;
    public static NodeEvent Of(String frame) {
        Integer _batteryLevel = Integer.valueOf(0);
```

```
        ArrayList _accValues = new ArrayList();
        StringTokenizer tokens = new StringTokenizer(frame, "#");
        tokens.nextToken();
        tokens.nextToken();
        String _nodeId = tokens.nextToken();
        tokens.nextToken();
        StringTokenizer _tokens = new StringTokenizer(tokens.nextToken(), ":");
        String label = _tokens.nextToken();
        if(label.equals("BAT")) {
            _batteryLevel = Integer.valueOf(_tokens.nextToken());
            _tokens = new StringTokenizer(tokens.nextToken(), ":");
            label = _tokens.nextToken();
        }
        while(label.equals("ACC")) {
            StringTokenizer __tokens = new StringTokenizer(_tokens.nextToken(),
";");
            AccValue sample = new AccValue();
            sample.setAccX(Integer.valueOf(__tokens.nextToken()));
            sample.setAccY(Integer.valueOf(__tokens.nextToken()));
            sample.setAccZ(Integer.valueOf(__tokens.nextToken()));
            _accValues.add(sample);
            _tokens = new StringTokenizer(tokens.nextToken(), ":");
            label = _tokens.nextToken();
        }
        return new NodeEvent(LocalDateTime.now(), _nodeId, _batteryLevel,
_accValues);
    }
    public NodeEvent(LocalDateTime timestamp) {
        this.timestamp = timestamp;
        this.accValues = new ArrayList();
    }
    public NodeEvent(LocalDateTime timestamp, String nodoId, Integer
batteryLevel, ArrayList<AccValue> accValues) {
        this.timestamp = timestamp;
        this.nodoId = nodoId;
        this.batteryLevel = batteryLevel;
        this.accValues = accValues;
    }
    public LocalDateTime getTimestamp() {
        return this.timestamp;
    }
    public void setTimestamp(LocalDateTime timestamp) {
        this.timestamp = timestamp;
    }
    public String getNodoId() {
        return this.nodoId;
    }
    public void setNodoId(String nodoId) {
        this.nodoId = nodoId;
    }
    public Integer getBatteryLevel() {
        return this.batteryLevel;
    }
    public void setBatteryLevel(Integer batteryLevel) {
        this.batteryLevel = batteryLevel;
    }
    public ArrayList<AccValue> getAccValues() {
        return this.accValues;
    }
    public void setAccValues(ArrayList<AccValue> accValues) {
        this.accValues = accValues;
    }
    public String toString() {
        return "NodeEvent{timestamp=" + this.timestamp + ", nodoId=\'" +
this.nodoId + '\'' + ", batteryLevel=" + this.batteryLevel + ", accValues=" +
this.accValues + '}';
    }
```

```
}

//API NODE
//Main

package api.node;
import api.node.controller.ProgramInterface;
import java.io.FileNotFoundException;
public class Main {
    public Main() {
    }
    public static void main(String[] args) {
        try {
            ProgramInterface e = new ProgramInterface();
            e.run();
        } catch (FileNotFoundException var2) {
            var2.printStackTrace();
        }
    }
}
```

ANEXO II INTERFAZ GRÁFICA

```
//Cliente Controlador Configuración

//CommPreference

package com.client.controller.conf;
import java.util.prefs.Preferences;
public class CommPreferences {
    private Preferences preferences =
Preferences.userNodeForPackage(CommPreferences.class);
    public String getAddress() {
        return preferences.get("address", "localhost");
    }
    public String getPort() {
        return preferences.get("port", "80");
    }
    public String getHost() {
        return getAddress() + ":" + getPort();
    }
    public void setCommParameters(String address, String port) {
        if (address != null && !address.equals(""))
            preferences.put("address", address);
        if (port != null && !port.equals(""))
            preferences.put("port", port);
    }
}

//Cliente Web

//ProductionService

package com.client.controller.web;

import com.client.model.util.GetRequest;
import com.client.model.var.CowData;
import com.google.gson.Gson;
import com.google.gson.reflect.TypeToken;
import javafx.concurrent.Service;
import javafx.concurrent.Task;
import java.lang.reflect.Type;
import java.time.LocalDate;
import java.util.ArrayList;
public class ProductionService extends Service<ArrayList<CowData>> {
    private GetRequest getRequest;
    public ProductionService(String host) {
        getRequest = new GetRequest(host, "");
        getRequest.getValueSet().put("name", "");
        getRequest.getValueSet().put("id", "");
        getRequest.getValueSet().put("birth", "");
        getRequest.getValueSet().put("insem", "");
        getRequest.getValueSet().put("labor", "");
        getRequest.getValueSet().put("next", "");
    }
    public void setHost(String host) {
        getRequest.setHost(host);
    }
    public void get() {
        getRequest.setPath("api/project/cow/get");
        getRequest.getValueSet().replace("name", "");
        getRequest.getValueSet().replace("id", "");
        getRequest.getValueSet().replace("birth", "");
        getRequest.getValueSet().replace("insem", "");
        getRequest.getValueSet().replace("labor", "");
```

```
        getRequest.getValueSet().replace("next", "");
    }
    public void add(String name, String id, LocalDate birth,
                    LocalDate insem, LocalDate labor, LocalDate next) {
        getRequest.setPath("api/project/cow/add");
        getRequest.getValueSet().replace("name", name);
        getRequest.getValueSet().replace("id", id);
        getRequest.getValueSet().replace("birth", (birth != null ?
birth.toString() : ""));
        getRequest.getValueSet().replace("insem", (insem != null ?
insem.toString() : ""));
        getRequest.getValueSet().replace("labor", (labor != null ?
labor.toString() : ""));
        getRequest.getValueSet().replace("next", (name != null ? next.toString()
: ""));
    }
    public void set(String name, String id, LocalDate birth,
                    LocalDate insem, LocalDate labor, LocalDate next) {
        getRequest.setPath("api/project/cow/set");
        getRequest.getValueSet().replace("name", name);
        getRequest.getValueSet().replace("id", id);
        getRequest.getValueSet().replace("birth", (birth != null ?
birth.toString() : ""));
        getRequest.getValueSet().replace("insem", (insem != null ?
insem.toString() : ""));
        getRequest.getValueSet().replace("labor", (labor != null ?
labor.toString() : ""));
        getRequest.getValueSet().replace("next", (name != null ? next.toString()
: ""));
    }
    public void delete(String name) {
        getRequest.setPath("api/project/cow/delete");
        getRequest.getValueSet().replace("name", name);
        getRequest.getValueSet().replace("id", "");
        getRequest.getValueSet().replace("birth", "");
        getRequest.getValueSet().replace("insem", "");
        getRequest.getValueSet().replace("labor", "");
        getRequest.getValueSet().replace("next", "");
    }
    @Override
    protected Task<ArrayList<CowData>> createTask() {
        return new Task<ArrayList<CowData>>() {
            @Override
            protected ArrayList<CowData> call() throws Exception {
                getRequest.query();
                if (getRequest.getStatusCode() == 200) {
                    switch (getRequest.getResponse()) {
                        case "{\"result\":\"void\"}":
                            updateMessage("No se obtuvo ningun resultado");
                            return null;
                        case "{\"result\":\"true\"}":
                            updateMessage("Comando ejecutado exitosamente");
                            return null;
                        default:
                            Gson gson = new Gson();
                            Type typeCowList = new
TypeToken<ArrayList<CowData>>() {
                            }.getType();
                            return gson.fromJson(getRequest.getResponse(),
typeCowList);
                    }
                } else {
                    updateMessage("Error de Servicio: Status Code " +
getRequest.getStatusCode());
                    return null;
                }
            }
```

```
        };
    }
}

//Cliente Web

//StaticService

package com.client.controller.web;

import com.client.model.util.GetRequest;
import com.client.model.var.DailyEvent;
import com.google.gson.Gson;
import com.google.gson.reflect.TypeToken;
import javafx.concurrent.Service;
import javafx.concurrent.Task;
import java.lang.reflect.Type;
import java.util.ArrayList;
import java.util.HashMap;
public class StatisticService extends Service<ArrayList<DailyEvent>> {
    private static HashMap<String, Integer> monthNumber = new HashMap<>();
    static {
        monthNumber.put("Enero", 1);
        monthNumber.put("Febrero", 2);
        monthNumber.put("Marzo", 3);
        monthNumber.put("Abril", 4);
        monthNumber.put("Mayo", 5);
        monthNumber.put("Junio", 6);
        monthNumber.put("Julio", 7);
        monthNumber.put("Agosto", 8);
        monthNumber.put("Septiembre", 9);
        monthNumber.put("Octubre", 10);
        monthNumber.put("Noviembre", 11);
        monthNumber.put("Diciembre", 12);
    }
    private int start_month;
    private int end_month;
    private GetRequest getRequest;
    public StatisticService(String host) {
        getRequest = new GetRequest(host, "api/project/event/get/daily");
        getRequest.getValueSet().put("id", "");
        getRequest.getValueSet().put("month", "");
        getRequest.getValueSet().put("year", "");
    }
    public void setHost(String host) {
        getRequest.setHost(host);
    }
    public void setId(String id) {
        if (id == null)
            getRequest.getValueSet().replace("id", "");
        else
            getRequest.getValueSet().replace("id", id);
    }
    public void setYear(String year) {
        if (year == null)
            getRequest.getValueSet().replace("year", "");
        else
            getRequest.getValueSet().replace("year", year);
    }
    public void setStartMonth(String month) {
        if (month == null)
            start_month = 1;
        else
            start_month = monthNumber.get(month);
    }
```

```
    public void setEndMonth(String month) {
        if (month == null)
            end_month = 1;
        else
            end_month = monthNumber.get(month);
        if (end_month < start_month)
            end_month = start_month;
    }
    @Override
    protected Task<ArrayList<DailyEvent>> createTask() {
        return new Task<ArrayList<DailyEvent>>() {
            @Override
            protected ArrayList<DailyEvent> call() throws Exception {
                Gson gson = new Gson();
                Type typeDailyList = new
TypeToken<ArrayList<DailyEvent>>(){}.getType();
                ArrayList<DailyEvent> dailyEvents = new ArrayList<>();
                updateProgress(0.0, 1.0);
                for (int i = start_month; i <= end_month; i++) {
                    getRequest.getValueSet().replace("month",
String.valueOf(i));
                    getRequest.query();
                    if (getRequest.getStatusCode() == 200) {
                        if
(!getRequest.getResponse().equals("{\"result\":\"void\"}"))

dailyEvents.addAll(gson.fromJson(getRequest.getResponse(), typeDailyList));
                    } else {
                        updateMessage("Error de Servicio: Status Code " +
getRequest.getStatusCode());
                        return null;
                    }
                    updateProgress((i - start_month + 1.0)/(end_month -
start_month + 1.0), 1.0);
                    if (isCancelled()) {
                        return null;
                    }
                }
                return dailyEvents;
            }
        };
    }
}

//Cliente Web

//VisualizationService

package com.client.controller.web;

import com.client.model.util.GetRequest;
import com.client.model.var.NodeEvent;
import com.google.gson.Gson;
import com.google.gson.reflect.TypeToken;
import javafx.concurrent.Service;
import javafx.concurrent.Task;
import java.lang.reflect.Type;
import java.time.LocalDate;
import java.time.LocalDateTime;
import java.time.LocalTime;
import java.util.ArrayList;
public class VisualizationService extends Service<ArrayList<NodeEvent>> {
    private GetRequest getRequest;
    public VisualizationService(String host) {
        getRequest = new GetRequest(host, "api/project/event/get");
```

```
        getRequest.getValueSet().put("id", "");
        getRequest.getValueSet().put("start", "");
        getRequest.getValueSet().put("end", "");
        getRequest.getValueSet().put("last", "20");
    }
    public void setHost(String host) {
        getRequest.setHost(host);
    }
    public void setId(String id) {
        getRequest.getValueSet().replace("id", id);
    }
    public void setStart(LocalDate start) {
        if (start == null)
            getRequest.getValueSet().replace("start", "");
        else
            getRequest.getValueSet().replace("start", LocalDateTime.of(start,
LocalTime.MIN).toString());
    }
    public void setEnd(LocalDate end) {
        if (end == null)
            getRequest.getValueSet().replace("end", "");
        else
            getRequest.getValueSet().replace("end", LocalDateTime.of(end,
LocalTime.MAX).toString());
    }
    public void setLast(String last) {
        if (last.equals("") || Integer.valueOf(last) < 0 ||
Integer.valueOf(last) > 20)
            last = "20";
        getRequest.getValueSet().replace("last", last);
    }
    @Override
    protected Task<ArrayList<NodeEvent>> createTask() {
        return new Task<ArrayList<NodeEvent>>() {
            @Override
            protected ArrayList<NodeEvent> call() throws Exception {
                getRequest.query();
                if (getRequest.getStatusCode() == 200) {
                    if
(getRequest.getResponse().equals("{\"result\":\"void\"}")) {
                        updateMessage("No se obtuvo ningun resultado");
                        return null;
                    } else {
                        Gson gson = new Gson();
                        Type typeEventList = new
TypeToken<ArrayList<NodeEvent>>(){}.getType();
                        return gson.fromJson(getRequest.getResponse(),
typeEventList);
                    }
                } else {
                    updateMessage("Error de Servicio: Status Code " +
getRequest.getStatusCode());
                    return null;
                }
            }
        };
    }
}

//Cliente Web

//GUI

package com.client.controller;
```

```java
import com.client.controller.conf.CommPreferences;
import com.client.controller.web.ProductionService;
import com.client.controller.web.StatisticService;
import com.client.controller.web.VisualizationService;
import com.client.model.var.AccValue;
import com.client.model.var.CowData;
import com.client.model.var.DailyEvent;
import com.client.model.var.NodeEvent;
import javafx.fxml.FXML;
import javafx.collections.FXCollections;
import javafx.collections.ObservableList;
import javafx.scene.chart.*;
import javafx.scene.control.*;
import javafx.scene.layout.Pane;
import java.time.LocalDate;
import java.time.Month;
import java.util.ArrayList;
public class GUI {
    //GUI Objects
    //Visualization
    @FXML
    private ListView<NodeEvent> vslEventList = new ListView<>();
    @FXML
    private CategoryAxis vslXAxis = new CategoryAxis();
    @FXML
    private NumberAxis vslYAxis = new NumberAxis();
    @FXML
    private LineChart<String, Number> vslEventChart = new LineChart<>(vslXAxis,
vslYAxis);
    @FXML
    private CheckBox vslXBox = new CheckBox();
    @FXML
    private CheckBox vslYBox = new CheckBox();
    @FXML
    private CheckBox vslZBox = new CheckBox();
    @FXML
    private CheckBox vslRBox = new CheckBox();
    @FXML
    private Label vslXLabel = new Label();
    @FXML
    private Label vslYLabel = new Label();
    @FXML
    private Label vslZLabel = new Label();
    @FXML
    private Label vslRLabel = new Label();
    @FXML
    private TextField vslIdField = new TextField();
    @FXML
    private DatePicker vslStartPicker = new DatePicker();
    @FXML
    private DatePicker vslEndPicker = new DatePicker();
    @FXML
    private TextField vslLastField = new TextField();
    //Statisticians
    @FXML
    private TextField stcIdField = new TextField();
    @FXML
    private TextField stcYearField = new TextField();
    @FXML
    private ComboBox<String> stcStartMonthBox = new ComboBox<>();
    @FXML
    private ComboBox<String> stcEndMonthBox = new ComboBox<>();
    @FXML
    private CategoryAxis stcXAxis = new CategoryAxis();
    @FXML
    private NumberAxis stcYAxis = new NumberAxis();
    @FXML
```

```
    private BarChart<String, Number> stcBarChar = new BarChart<>(stcXAxis,
stcYAxis);
    @FXML
    private RadioButton stcDailyButton = new RadioButton();
    @FXML
    private RadioButton stcWeeklyButton = new RadioButton();
    @FXML
    private RadioButton stcMonthlyButton = new RadioButton();
    @FXML
    private ProgressBar stcLoadBar = new ProgressBar();
    //Production
    @FXML
    private ListView<CowData> pdtCowList = new ListView<>();
    @FXML
    private TextField pdtNameField = new TextField();
    @FXML
    private TextField pdtIdField = new TextField();
    @FXML
    private DatePicker pdtBirthPicker = new DatePicker();
    @FXML
    private DatePicker pdtInseminationPicker = new DatePicker();
    @FXML
    private DatePicker pdtLaborPicker = new DatePicker();
    @FXML
    private DatePicker pdtNextPicker = new DatePicker();
    //Configuration
    @FXML
    private TextField cfgAddressField = new TextField();
    @FXML
    private TextField cfgPortField = new TextField();
    //Control Attributes
    //General
    private CommPreferences comm = new CommPreferences();
    //Visualization
    private VisualizationService vslService;
    private ObservableList<NodeEvent> eventItems =
FXCollections.observableArrayList();
    private NodeEvent thisEvent = null;
    //Statisticians
    private StatisticService stcService;
    private ToggleGroup stcGroup = new ToggleGroup();
    private ObservableList<String> monthName =
FXCollections.observableArrayList(
             "Enero", "Febrero", "Marzo", "Abril", "Mayo", "Junio", "Julio",
"Agosto",
             "Septiembre", "Octubre", "Noviembre", "Diciembre");
    private ArrayList<DailyEvent> thisDailyEvents = null;
    //Production
    private ProductionService pdtService;
    private ObservableList<CowData> cowItems =
FXCollections.observableArrayList();
    private CowData thisCow = null;
    //Building Object
    public GUI() {
    }
    //Controllers Objects
    //Visualization
    public void initVisualization() {
        vslEventList.setCellFactory(param -> new EventCell());

vslEventList.getSelectionModel().selectedItemProperty().addListener((observable,
oldValue, newValue) -> {
             vslEventChart.getData().clear();
             thisEvent = newValue;
             onVslXCheck();
             onVslYCheck();
             onVslZCheck();
```

```
            onVslRCheck();
        });
        vslEventList.setItems(eventItems);
        vslService = new VisualizationService(comm.getHost());
        vslService.messageProperty().addListener((observable, oldValue,
newValue) -> {
            if (!newValue.isEmpty())
                showInfo("Service Message", newValue);
        });
        vslService.valueProperty().addListener((observable, oldValue, newValue)
-> {
            eventItems.clear();
            if (newValue != null)
                eventItems.addAll(newValue);
        });
    }
    public void onVslXCheck() {
        if (thisEvent != null) {
            if (vslXBox.isSelected()) {
                XYChart.Series series = new XYChart.Series();
                series.setName("Aceleración X");
                int i = 1;
                for (AccValue value : thisEvent.getAccValues()) {
                    XYChart.Data data = new XYChart.Data(String.valueOf(i++),
value.getAccX());
                    data.setNode(new Pane());
                    data.getNode().setOnMouseEntered(event ->
vslXLabel.setText("X: " + data.getYValue()));
                    data.getNode().setOnMouseExited(event ->
vslXLabel.setText("X: 0"));
                    series.getData().add(data);
                }
                vslEventChart.getData().add(series);
            } else {
                for (int i = 0; i < vslEventChart.getData().size(); i++) {
                    if
(vslEventChart.getData().get(i).getName().equals("Aceleración X")) {
                        vslEventChart.getData().remove(i);
                        vslXLabel.setText("X: 0");
                        break;
                    }
                }
            }
        }
    }
    public void onVslYCheck() {
        if (thisEvent != null) {
            if (vslYBox.isSelected()) {
                XYChart.Series series = new XYChart.Series();
                series.setName("Aceleración Y");
                int i = 1;
                for (AccValue value : thisEvent.getAccValues()) {
                    XYChart.Data data = new XYChart.Data(String.valueOf(i++),
value.getAccY());
                    data.setNode(new Pane());
                    data.getNode().setOnMouseEntered(event ->
vslYLabel.setText("Y: " + data.getYValue()));
                    data.getNode().setOnMouseExited(event ->
vslYLabel.setText("Y: 0"));
                    series.getData().add(data);
                }
                vslEventChart.getData().add(series);
            } else {
                for (int i = 0; i < vslEventChart.getData().size(); i++) {
                    if
(vslEventChart.getData().get(i).getName().equals("Aceleración Y")) {
                        vslEventChart.getData().remove(i);
```

```
                        vslYLabel.setText("Y: 0");
                        break;
                    }
                }
            }
        }
    }
    public void onVslZCheck() {
        if (thisEvent != null) {
            if (vslZBox.isSelected()) {
                XYChart.Series series = new XYChart.Series();
                series.setName("Aceleración Z");
                int i = 1;
                for (AccValue value : thisEvent.getAccValues()) {
                    XYChart.Data data = new XYChart.Data(String.valueOf(i++),
value.getAccZ());
                    data.setNode(new Pane());
                    data.getNode().setOnMouseEntered(event ->
vslZLabel.setText("Z: " + data.getYValue()));
                    data.getNode().setOnMouseExited(event ->
vslZLabel.setText("Z: 0"));
                    series.getData().add(data);
                }
                vslEventChart.getData().add(series);
            } else {
                for (int i = 0; i < vslEventChart.getData().size(); i++) {
                    if
(vslEventChart.getData().get(i).getName().equals("Aceleración Z")) {
                        vslEventChart.getData().remove(i);
                        vslZLabel.setText("Z: 0");
                        break;
                    }
                }
            }
        }
    }
    public void onVslRCheck() {
        if (thisEvent != null) {
            if (vslRBox.isSelected()) {
                XYChart.Series series = new XYChart.Series();
                series.setName("Aceleración R");
                int i = 1;
                for (AccValue value : thisEvent.getAccValues()) {
                    XYChart.Data data = new XYChart.Data(String.valueOf(i++),
value.getAccR());
                    data.setNode(new Pane());
                    data.getNode().setOnMouseEntered(event ->
vslRLabel.setText("R: " + data.getYValue()));
                    data.getNode().setOnMouseExited(event ->
vslRLabel.setText("R: 0"));
                    series.getData().add(data);
                }
                vslEventChart.getData().add(series);
            } else {
                for (int i = 0; i < vslEventChart.getData().size(); i++) {
                    if
(vslEventChart.getData().get(i).getName().equals("Aceleración R")) {
                        vslEventChart.getData().remove(i);
                        vslRLabel.setText("R: 0");
                        break;
                    }
                }
            }
        }
    }
    public void onVslClearButton() {
        vslIdField.setText("");
```

```
        vslStartPicker.setValue(null);
        vslEndPicker.setValue(null);
        vslLastField.setText("");
    }
    public void onVslUpdateButton() {
        vslService.setHost(comm.getHost());
        vslService.setId(vslIdField.getText());
        vslService.setStart(vslStartPicker.getValue());
        vslService.setEnd(vslEndPicker.getValue());
        vslService.setLast(vslLastField.getText());
        vslService.restart();
    }
    //Statisticians
    public void initStatisticians() {
        stcStartMonthBox.setItems(monthName);
        stcEndMonthBox.setItems(monthName);
        stcMonthlyButton.setToggleGroup(stcGroup);
        stcWeeklyButton.setToggleGroup(stcGroup);
        stcDailyButton.setToggleGroup(stcGroup);
        stcDailyButton.setSelected(true);
        stcGroup.selectedToggleProperty().addListener((observable, oldValue,
newValue) -> {
            if (thisDailyEvents != null) {
                if (newValue.equals(stcDailyButton))
                    doDailyStatistics();
                if (newValue.equals(stcWeeklyButton))
                    doWeeklyStatistics();
                if (newValue.equals(stcMonthlyButton))
                    doMonthlyStatistics();
            }
        });
        stcService = new StatisticService(comm.getHost());
        stcService.messageProperty().addListener((observable, oldValue,
newValue) -> {
            if (!newValue.isEmpty())
                showInfo("Service Message", newValue);
        });
        stcService.valueProperty().addListener((observable, oldValue, newValue)
-> {
            if (newValue != null) {
                thisDailyEvents = newValue;
                if (stcGroup.getSelectedToggle().equals(stcDailyButton))
                    doDailyStatistics();
                if (stcGroup.getSelectedToggle().equals(stcWeeklyButton))
                    doWeeklyStatistics();
                if (stcGroup.getSelectedToggle().equals(stcMonthlyButton))
                    doMonthlyStatistics();
            }
        });
        stcService.progressProperty().addListener((observable, oldValue,
newValue) -> stcLoadBar.setProgress(newValue.doubleValue()));
    }
    public void onStcClearButton() {
        stcIdField.setText("");
        stcYearField.setText("");
        stcStartMonthBox.setValue(null);
        stcEndMonthBox.setValue(null);
    }
    public void onStcApplyButton() {
        stcService.setHost(comm.getHost());
        stcService.setId(stcIdField.getText());
        stcService.setYear(stcYearField.getText());
        stcService.setStartMonth(stcStartMonthBox.getValue());
        stcService.setEndMonth(stcEndMonthBox.getValue());
        stcService.restart();
    }
    public void onStcCancelButton() {
```

```
        stcService.cancel();
        stcLoadBar.setProgress(0.0);
    }
    private void doDailyStatistics() {
        int year = thisDailyEvents.get(0).getDay().getYear();
        XYChart.Series series = new XYChart.Series();
        series.setName("Estadisticas Diarias");
        stcBarChar.getData().clear();
        for (LocalDate dt = LocalDate.MIN.withYear(year);
             dt.isBefore(LocalDate.MAX.withYear(year)) ||
dt.isEqual(LocalDate.MAX.withYear(year));
             dt = dt.plusDays(1)) {
            int value = 0;
            for (DailyEvent event : thisDailyEvents) {
                if (event.getDay().isEqual(dt)) {
                    value = Math.abs(event.getAccMean().getAccR());
                    break;
                }
            }
            series.getData().add(new XYChart.Data(dt.toString(), value));
        }
        stcBarChar.getData().addAll(series);
    }
    private void doWeeklyStatistics() {
        int year = thisDailyEvents.get(0).getDay().getYear();
        XYChart.Series series = new XYChart.Series();
        series.setName("Estadisticas Semanales");
        stcBarChar.getData().clear();
        LocalDate dt = LocalDate.MIN.withYear(year);
        int week = 1;
        int value = 0;
        for (int i = 1; i <= dt.lengthOfYear() ; i++) {
            dt = dt.withDayOfYear(i);
            for (DailyEvent event : thisDailyEvents) {
                if (event.getDay().isEqual(dt))
                    value = (value + Math.abs(event.getAccMean().getAccR())) /
2;
            }
            if (dt.getDayOfWeek().getValue() == 7 || i == dt.lengthOfYear()) {
                series.getData().add(new XYChart.Data(String.valueOf(week),
value));
                week++;
                value = 0;
            }
        }
        stcBarChar.getData().addAll(series);
    }
    private void doMonthlyStatistics() {
        XYChart.Series series = new XYChart.Series();
        series.setName("Estadisticas Mensuales");
        stcBarChar.getData().clear();
        for (int i = 1; i <= 12; i++) {
            int value = 0;
            for (DailyEvent event : thisDailyEvents) {
                if (event.getDay().getMonthValue() == i) {
                    value = (value + Math.abs(event.getAccMean().getAccR())) /
2;
                }
            }
            series.getData().add(new XYChart.Data(Month.of(i).toString(),
value));
        }
        stcBarChar.getData().addAll(series);
    }
    //Production
    public void initProduction() {
        pdtCowList.setCellFactory(param -> new CowCell());
```

```
pdtCowList.getSelectionModel().selectedItemProperty().addListener((observable,
oldValue, newValue) -> {
            thisCow = newValue;
        });
        pdtCowList.setItems(cowItems);
        pdtService = new ProductionService(comm.getHost());
        pdtService.messageProperty().addListener((observable, oldValue,
newValue) -> {
            if (!newValue.isEmpty())
                showInfo("Service Message", newValue);
        });
        pdtService.valueProperty().addListener((observable, oldValue, newValue)
-> {
            cowItems.clear();
            if (newValue != null)
                cowItems.addAll(newValue);
        });
    }
    public void onPdtLoadButton() {
        if (thisCow != null) {
            pdtNameField.setText(thisCow.getName());
            pdtIdField.setText(thisCow.getNodeId());
            pdtBirthPicker.setValue(thisCow.getBirthDate());
            pdtInseminationPicker.setValue(thisCow.getInseminationDate());
            pdtLaborPicker.setValue(thisCow.getLaborDate());
            pdtNextPicker.setValue(thisCow.getNextDate());
        }
    }
    public void onPdtUpdateButton() {
        pdtService.setHost(comm.getHost());
        pdtService.get();
        pdtService.restart();
    }
    public void onPdtSaveButton() {
        pdtService.setHost(comm.getHost());
        pdtService.add(
                pdtNameField.getText(),
                pdtIdField.getText(),
                pdtBirthPicker.getValue(),
                pdtInseminationPicker.getValue(),
                pdtLaborPicker.getValue(),
                pdtNextPicker.getValue());
        pdtService.restart();
    }
    public void onPdtReplaceButton() {
        pdtService.setHost(comm.getHost());
        pdtService.set(
                pdtNameField.getText(),
                pdtIdField.getText(),
                pdtBirthPicker.getValue(),
                pdtInseminationPicker.getValue(),
                pdtLaborPicker.getValue(),
                pdtNextPicker.getValue());
        pdtService.restart();
    }
    public void onPdtDeleteButton() {
        pdtService.setHost(comm.getHost());
        pdtService.delete(pdtNameField.getText());
        pdtService.restart();
    }
    public void onPdtClearButton() {
        pdtNameField.setText("");
        pdtIdField.setText("");
        pdtBirthPicker.setValue(null);
        pdtInseminationPicker.setValue(null);
        pdtLaborPicker.setValue(null);
```

```
        pdtNextPicker.setValue(null);
    }
    //Configuration
    public void initConfiguration() {
        cfgAddressField.setText(comm.getAddress());
        cfgPortField.setText(comm.getPort());
    }
    public void onCfgSaveButton() {
        comm.setCommParameters(cfgAddressField.getText(),
cfgPortField.getText());
        showInfo("Configuration Saved Correctly", "Se actualizó la dirección y
el puerto del servidor");
    }
    //Alert Dialog Function
    private void showInfo(String header, String context) {
        Alert alert = new Alert(Alert.AlertType.INFORMATION);
        alert.setTitle("Information Dialog");
        alert.setHeaderText(header);
        alert.setContentText(context);
        alert.showAndWait();
    }
    private void showError(String header, String context) {
        Alert alert = new Alert(Alert.AlertType.ERROR);
        alert.setTitle("Error Dialog");
        alert.setHeaderText(header);
        alert.setContentText(context);
        alert.showAndWait();
    }
    //Building Class for the Lists Cells
    private static class EventCell extends ListCell<NodeEvent> {
        @Override
        protected void updateItem(NodeEvent item, boolean empty) {
            super.updateItem(item, empty);
            if (item != null)
                setText(item.getNodoId() + "   " + "BAT:" +
item.getBatteryLevel() + "%" +
                        "\n" + item.getTimestamp());
            else
                setText("");
        }
    }
    private static class CowCell extends ListCell<CowData> {
        @Override
        protected void updateItem(CowData item, boolean empty) {
            super.updateItem(item, empty);
            if (item != null)
                setText("Vaca: " + item.getName() + " \t " + item.getNodeId() +
" \t " + item.getBirthDate() +
                        " \t\t " + (item.getInseminationDate() != null ?
item.getInseminationDate() : "vacio        ") +
                        " \t\t " + (item.getLaborDate() != null ?
item.getLaborDate() : "vacio        ") +
                        " \t\t " + (item.getNextDate() != null ?
item.getNextDate() : "vacio        "));
            else
                setText("");
        }
    }
}

//Model Util

//GetRequest

package com.client.model.util;
```

```java
import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.HttpClientBuilder;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.util.LinkedHashMap;
import static org.apache.http.HttpHeaders.USER_AGENT;
public class GetRequest {
    private String host;
    private String path;
    private LinkedHashMap<String, String> valueSet = new LinkedHashMap<>();
    private Integer statusCode;
    private String response;
    public GetRequest(String host, String path) {
        this.host = host;
        this.path = path;
    }
    public String getHost() {
        return host;
    }
    public void setHost(String host) {
        this.host = host;
    }
    public String getPath() {
        return path;
    }
    public void setPath(String path) {
        this.path = path;
    }
    public LinkedHashMap<String, String> getValueSet() {
        return valueSet;
    }
    public void setValueSet(LinkedHashMap<String, String> valueSet) {
        this.valueSet = valueSet;
    }
    public Integer getStatusCode() {
        return statusCode;
    }
    public String getResponse() {
        return response;
    }
    public void query() {
        String url = "http://" + host + "/" + path;
        if (!valueSet.isEmpty()) {
            boolean isFirst = true;
            for (String value : valueSet.keySet()) {
                if (!valueSet.get(value).equals("")) {
                    url += isFirst ? "?" : "&";
                    url += value + "=" + valueSet.get(value);
                    isFirst = false;
                }
            }
        }
        System.out.println(url);
        HttpGet request = new HttpGet(url);
        request.addHeader("User-Agent", USER_AGENT);
        HttpClient client = HttpClientBuilder.create().build();
        try {
            HttpResponse response = client.execute(request);
            statusCode = response.getStatusLine().getStatusCode();
            this.response = null;
            if (statusCode == 200) {
                BufferedReader buffered = new BufferedReader(
                        new
InputStreamReader(response.getEntity().getContent()));
```

```
                StringBuilder result = new StringBuilder();
                String line;
                while ((line = buffered.readLine()) != null)
                    result.append(line);
                this.response = result.toString();
            }
        } catch (IOException e) {
            statusCode = 400;
        }
    }
}

//Model Util

//PostRequest

package com.client.model.util;

import org.apache.http.HttpResponse;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.entity.StringEntity;
import org.apache.http.impl.client.HttpClientBuilder;
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
public class PostRequest {
    private String host;
    private String path;
    private String message = "{}";
    private Integer statusCode;
    private String response;
    public PostRequest(String host, String path) {
        this.host = host;
        this.path = path;
    }
    public String getHost() {
        return host;
    }
    public void setHost(String host) {
        this.host = host;
    }
    public String getPath() {
        return path;
    }
    public void setPath(String path) {
        this.path = path;
    }
    public String getMessage() {
        return message;
    }
    public void setMessage(String message) {
        this.message = message;
    }
    public Integer getStatusCode() {
        return statusCode;
    }
    public String getResponse() {
        return response;
    }
    public void query() throws UnsupportedEncodingException {
        String url = "http://" + host + "/" + path;
        StringEntity entity = new StringEntity(message);
        HttpPost request = new HttpPost(url);
```

```
        request.setHeader("Content-Type", "application/json");
        request.setEntity(entity);
        HttpClient client = HttpClientBuilder.create().build();
        try {
            HttpResponse response = client.execute(request);
            statusCode = response.getStatusLine().getStatusCode();
            this.response = null;
            if (statusCode == 200) {
                BufferedReader buffered = new BufferedReader(
                        new
InputStreamReader(response.getEntity().getContent()));
                StringBuilder result = new StringBuilder();
                String line;
                while ((line = buffered.readLine()) != null)
                    result.append(line);
                this.response = result.toString();
            }
        } catch (IOException e) {
            statusCode = 400;
        }
    }
}

//Model Var

//AccValue

package com.client.model.var;

public class AccValue {
    private Integer accX = 0;
    private Integer accY = 0;
    private Integer accZ = 0;
    private Integer accR = 0;
    public AccValue() {
    }
    public AccValue(Integer accX, Integer accY, Integer accZ) {
        this.accX = accX;
        this.accY = accY;
        this.accZ = accZ;
        this.accR = buildR();
    }
    private Integer buildR() {
        return accR = (int) (Math.sqrt(Math.pow(accX, 2) + Math.pow(accY, 2) +
Math.pow(accZ, 2)));
    }
    public Integer getAccX() {
        return accX;
    }
    public void setAccX(Integer accX) {
        this.accX = accX;
        accR = buildR();
    }
    public Integer getAccY() {
        return accY;
    }
    public void setAccY(Integer accY) {
        this.accY = accY;
        accR = buildR();
    }
    public Integer getAccZ() {
        return accZ;
    }
    public void setAccZ(Integer accZ) {
        this.accZ = accZ;
```

```
        accR = buildR();
    }
    public Integer getAccR() {
        return accR;
    }
    @Override
    public String toString() {
        return "AccValue{" +
                "accX=" + accX +
                ", accY=" + accY +
                ", accZ=" + accZ +
                ", accR=" + accR +
                '}';
    }
}

//Model Var

//CowData

package com.client.model.var;

import java.time.LocalDate;
public class CowData {
    private String name;
    private String nodeId;
    private LocalDate birthDate;
    private LocalDate inseminationDate;
    private LocalDate laborDate;
    private LocalDate nextDate;
    public CowData(String name, String nodeId, LocalDate birthDate) {
        this.name = name;
        this.nodeId = nodeId;
        this.birthDate = birthDate;
    }
    public CowData(String name, String nodeId, LocalDate birthDate,
                   LocalDate inseminationDate, LocalDate laborDate, LocalDate
nextDate) {
        this.name = name;
        this.nodeId = nodeId;
        this.birthDate = birthDate;
        this.inseminationDate = inseminationDate;
        this.laborDate = laborDate;
        this.nextDate = nextDate;
    }
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public String getNodeId() {
        return nodeId;
    }
    public void setNodeId(String nodeId) {
        this.nodeId = nodeId;
    }
    public LocalDate getBirthDate() {
        return birthDate;
    }
    public void setBirthDate(LocalDate birthDate) {
        this.birthDate = birthDate;
    }
    public LocalDate getInseminationDate() {
        return inseminationDate;
```

```
    }
    public void setInseminationDate(LocalDate inseminationDate) {
        this.inseminationDate = inseminationDate;
    }
    public LocalDate getLaborDate() {
        return laborDate;
    }
    public void setLaborDate(LocalDate laborDate) {
        this.laborDate = laborDate;
    }
    public LocalDate getNextDate() {
        return nextDate;
    }
    public void setNextDate(LocalDate nextDate) {
        this.nextDate = nextDate;
    }
    @Override
    public String toString() {
        return "CowData{" +
                "name='" + name + '\'' +
                ", nodeId='" + nodeId + '\'' +
                ", birthDate=" + birthDate +
                ", inseminationDate=" + inseminationDate +
                ", laborDate=" + laborDate +
                ", nextDate=" + nextDate +
                '}';
    }
}

//Model Var

//Dailyevent

package com.client.model.var;

import java.time.LocalDate;
import java.util.ArrayList;
/**
 * Created by Mikele on 20/09/2016.
 */
public class DailyEvent {
    private LocalDate day;
    private String nodeId;
    private AccValue accMean = new AccValue();
    public DailyEvent(LocalDate day) {
        this.day = day;
    }
    public DailyEvent(LocalDate day, String nodeId, AccValue accMean) {
        this.day = day;
        this.nodeId = nodeId;
        this.accMean = accMean;
    }
    public LocalDate getDay() {
        return day;
    }
    public void setDay(LocalDate day) {
        this.day = day;
    }
    public String getNodeId() {
        return nodeId;
    }
    public void setNodeId(String nodeId) {
        this.nodeId = nodeId;
    }
    public AccValue getAccMean() {
```

```java
        return accMean;
    }
    public void setAccMean(AccValue accMean) {
        this.accMean = accMean;
    }
    public void addAccMean(ArrayList<AccValue> accValues) {
        long x = 0;
        long y = 0;
        long z = 0;
        for (AccValue value : accValues) {
            x += value.getAccX();
            y += value.getAccY();
            z += value.getAccZ();
        }
        x /= accValues.size();
        y /= accValues.size();
        z /= accValues.size();
        accMean.setAccX((int) ((accMean.getAccX() + x)) / 2);
        accMean.setAccY((int) ((accMean.getAccY() + y)) / 2);
        accMean.setAccZ((int) ((accMean.getAccZ() + z)) / 2);
    }
    @Override
    public String toString() {
        return "DailyEvent{" +
                "day=" + day +
                ", nodeId='" + nodeId + '\'' +
                ", accMean=" + accMean +
                '}';
    }
}

//Model Var

//NodeEvent

package com.client.model.var;

import java.time.LocalDateTime;
import java.util.ArrayList;
import java.util.StringTokenizer;
public class NodeEvent {
    public static NodeEvent Of(String frame) {
        String _nodeId;
        Integer _batteryLevel = 0;
        ArrayList<AccValue> _accValues = new ArrayList<>();
        StringTokenizer tokens, _tokens, __tokens;
        String label;
        tokens = new StringTokenizer(frame, "#");
        tokens.nextToken();
        tokens.nextToken();
        _nodeId = tokens.nextToken();
        tokens.nextToken();
        _tokens = new StringTokenizer(tokens.nextToken(), ":");
        label = _tokens.nextToken();
        if (label.equals("BAT")) {
            _batteryLevel = Integer.valueOf(_tokens.nextToken());
            _tokens = new StringTokenizer(tokens.nextToken(), ":");
            label = _tokens.nextToken();
        }
        while (label.equals("ACC")) {
            __tokens = new StringTokenizer(_tokens.nextToken(), ";");
            AccValue sample = new AccValue();
            sample.setAccX(Integer.valueOf(__tokens.nextToken()));
            sample.setAccY(Integer.valueOf(__tokens.nextToken()));
            sample.setAccZ(Integer.valueOf(__tokens.nextToken()));
```

```
            _accValues.add(sample);
            _tokens = new StringTokenizer(tokens.nextToken(), ":");
            label = _tokens.nextToken();
        }
        return new NodeEvent(LocalDateTime.now(), _nodeId, _batteryLevel,
_accValues);
    }
    private LocalDateTime timestamp;
    private String nodoId;
    private Integer batteryLevel;
    private ArrayList<AccValue> accValues;
    public NodeEvent(LocalDateTime timestamp) {
        this.timestamp = timestamp;
        accValues = new ArrayList<>();
    }
    public NodeEvent(LocalDateTime timestamp, String nodoId, Integer
batteryLevel, ArrayList<AccValue> accValues) {
        this.timestamp = timestamp;
        this.nodoId = nodoId;
        this.batteryLevel = batteryLevel;
        this.accValues = accValues;
    }
    public LocalDateTime getTimestamp() {
        return timestamp;
    }
    public void setTimestamp(LocalDateTime timestamp) {
        this.timestamp = timestamp;
    }
    public String getNodoId() {
        return nodoId;
    }
    public void setNodoId(String nodoId) {
        this.nodoId = nodoId;
    }
    public Integer getBatteryLevel() {
        return batteryLevel;
    }
    public void setBatteryLevel(Integer batteryLevel) {
        this.batteryLevel = batteryLevel;
    }
    public ArrayList<AccValue> getAccValues() {
        return accValues;
    }
    public void setAccValues(ArrayList<AccValue> accValues) {
        this.accValues = accValues;
    }
    @Override
    public String toString() {
        return "NodeEvent{" +
                "timestamp=" + timestamp +
                ", nodoId='" + nodoId + '\'' +
                ", batteryLevel=" + batteryLevel +
                ", accValues=" + accValues +
                '}';
    }
}

//View

//gui.fxml

<?xml version="1.0" encoding="UTF-8"?>

<?import javafx.scene.shape.*?>
<?import javafx.scene.chart.*?>
```

```
<?import javafx.scene.control.*?>
<?import java.lang.*?>
<?import javafx.scene.layout.*?>
<?import javafx.geometry.Insets?>
<?import javafx.scene.layout.GridPane?>
<?import javafx.scene.control.Button?>
<?import javafx.scene.control.Label?>
<TabPane maxHeight="-Infinity" maxWidth="-Infinity" minHeight="-Infinity"
minWidth="-Infinity" prefHeight="480.0" prefWidth="1021.0"
tabClosingPolicy="UNAVAILABLE" xmlns="http://javafx.com/javafx/8"
xmlns:fx="http://javafx.com/fxml/1" fx:controller="com.client.controller.GUI">
  <tabs>
    <Tab text="Visualización">
          <content>
             <SplitPane dividerPositions="0.29797979797979796" prefHeight="160.0"
prefWidth="200.0">
                <items>
                  <AnchorPane minHeight="0.0" minWidth="0.0" prefHeight="160.0"
prefWidth="100.0">
                        <children>
                           <ListView fx:id="vslEventList" layoutX="58.0"
layoutY="101.0" prefHeight="200.0" prefWidth="200.0"
AnchorPane.bottomAnchor="0.0" AnchorPane.leftAnchor="0.0"
AnchorPane.rightAnchor="0.0" AnchorPane.topAnchor="0.0" />
                        </children>
                     </AnchorPane>
                  <AnchorPane minHeight="0.0" minWidth="0.0" prefHeight="451.0"
prefWidth="673.0">
                        <children>
                           <LineChart fx:id="vslEventChart" layoutX="14.0"
layoutY="14.0" prefHeight="294.0" prefWidth="548.0"
AnchorPane.bottomAnchor="157.0" AnchorPane.leftAnchor="0.0"
AnchorPane.rightAnchor="0.0" AnchorPane.topAnchor="0.0">
                              <xAxis>
                                <CategoryAxis fx:id="vslXAxis" side="BOTTOM" />
                              </xAxis>
                              <yAxis>
                                <NumberAxis fx:id="vslYAxis" side="LEFT" />
                              </yAxis>
                           </LineChart>
                           <CheckBox fx:id="vslXBox" layoutX="44.0"
layoutY="467.0" mnemonicParsing="false" onAction="#onVslXCheck" text=":X"
AnchorPane.bottomAnchor="148.0" AnchorPane.leftAnchor="44.0" />
                           <CheckBox fx:id="vslYBox" layoutX="94.0"
layoutY="467.0" mnemonicParsing="false" onAction="#onVslYCheck" text=":Y"
AnchorPane.bottomAnchor="148.0" AnchorPane.leftAnchor="94.0" />
                           <CheckBox fx:id="vslZBox" layoutX="144.0"
layoutY="467.0" mnemonicParsing="false" onAction="#onVslZCheck" text=":Z"
AnchorPane.bottomAnchor="148.0" AnchorPane.leftAnchor="144.0" />
                           <CheckBox fx:id="vslRBox" layoutX="194.0"
layoutY="467.0" mnemonicParsing="false" onAction="#onVslRCheck" text=":R"
AnchorPane.bottomAnchor="148.0" AnchorPane.leftAnchor="194.0" />
                           <Separator layoutX="8.0" layoutY="508.0"
prefHeight="3.0" prefWidth="629.0" AnchorPane.bottomAnchor="125.0"
AnchorPane.leftAnchor="8.0" AnchorPane.rightAnchor="8.0" />
                           <Label layoutX="14.0" layoutY="521.0" opacity="0.7"
prefHeight="21.0" prefWidth="46.0" text="Filtro" AnchorPane.bottomAnchor="94.0"
AnchorPane.leftAnchor="14.0" />
                           <Label layoutX="14.0" layoutY="552.0" opacity="0.7"
text="ID del Nodo" AnchorPane.bottomAnchor="63.0" AnchorPane.leftAnchor="14.0" />
                           <TextField fx:id="vslIdField" layoutX="14.0"
layoutY="580.0" prefHeight="31.0" prefWidth="131.0" promptText="Node ID"
AnchorPane.bottomAnchor="25.0" AnchorPane.leftAnchor="14.0" />
                           <DatePicker fx:id="vslStartPicker" layoutX="164.0"
layoutY="580.0" prefHeight="31.0" prefWidth="148.0" promptText="Start"
AnchorPane.bottomAnchor="25.0" AnchorPane.leftAnchor="164.0" />
                           <DatePicker fx:id="vslEndPicker" layoutX="330.0"
```

```
layoutY="580.0" prefHeight="31.0" prefWidth="148.0" promptText="End"
AnchorPane.bottomAnchor="25.0" AnchorPane.leftAnchor="330.0" />
                            <TextField fx:id="vslLastField" layoutX="497.0"
layoutY="580.0" prefHeight="31.0" prefWidth="53.0" promptText="Last"
AnchorPane.bottomAnchor="25.0" AnchorPane.leftAnchor="497.0" />
                            <Label layoutX="164.0" layoutY="552.0" opacity="0.7"
prefHeight="21.0" prefWidth="86.0" text="Comienzo" AnchorPane.bottomAnchor="63.0"
AnchorPane.leftAnchor="164.0" />
                            <Label layoutX="330.0" layoutY="552.0" opacity="0.7"
prefHeight="21.0" prefWidth="86.0" text="Finalización"
AnchorPane.bottomAnchor="63.0" AnchorPane.leftAnchor="330.0" />
                            <Label layoutX="497.0" layoutY="552.0" opacity="0.7"
prefHeight="21.0" prefWidth="62.0" text="Ultimos" AnchorPane.bottomAnchor="63.0"
AnchorPane.leftAnchor="497.0" />
                            <Button layoutX="562.0" layoutY="395.0"
mnemonicParsing="false" onAction="#onVslClearButton" prefHeight="31.0"
prefWidth="36.0" text="..." AnchorPane.bottomAnchor="25.0" />
                            <Button layoutX="612.0" layoutY="580.0"
maxWidth="88.0" mnemonicParsing="false" onAction="#onVslUpdateButton"
prefHeight="31.0" prefWidth="88.0" text="Actualizar"
AnchorPane.bottomAnchor="25.0" AnchorPane.leftAnchor="612.0"
AnchorPane.rightAnchor="14.0" />
                            <Label fx:id="vslXLabel" layoutX="373.0"
layoutY="284.0" prefHeight="21.0" prefWidth="40.0" text="X: 0"
AnchorPane.bottomAnchor="146.0" AnchorPane.rightAnchor="135.0" />
                            <Label fx:id="vslYLabel" layoutX="416.0"
layoutY="284.0" prefHeight="21.0" prefWidth="40.0" text="Y: 0"
AnchorPane.bottomAnchor="146.0" AnchorPane.rightAnchor="92.0" />
                            <Label fx:id="vslZLabel" layoutX="457.0"
layoutY="284.0" prefHeight="21.0" prefWidth="40.0" text="Z: 0"
AnchorPane.bottomAnchor="146.0" AnchorPane.rightAnchor="51.0" />
                            <Label fx:id="vslRLabel" layoutX="503.0"
layoutY="284.0" prefHeight="21.0" prefWidth="40.0" text="R: 0"
AnchorPane.bottomAnchor="146.0" AnchorPane.rightAnchor="9.0" />
                        </children>
                    </AnchorPane>
                </items>
            </SplitPane>
        </content></Tab>
    <Tab text="Estadisticos">
        <content>
            <AnchorPane prefHeight="200.0" prefWidth="200.0">
                <children>
                    <Label layoutX="37.0" layoutY="14.0" opacity="0.7" text="ID
del Nodo" AnchorPane.leftAnchor="37.0" AnchorPane.topAnchor="14.0" />
                    <Label layoutX="241.0" layoutY="14.0" opacity="0.7" text="Año
de Analisis" AnchorPane.leftAnchor="241.0" AnchorPane.topAnchor="14.0" />
                    <Label layoutX="364.0" layoutY="14.0" opacity="0.7" text="Mes
de Inicio" AnchorPane.leftAnchor="364.0" AnchorPane.topAnchor="14.0" />
                    <Label layoutX="511.0" layoutY="14.0" opacity="0.7" text="Mes
de Finalización" AnchorPane.leftAnchor="511.0" AnchorPane.topAnchor="14.0" />
                    <TextField fx:id="stcIdField" layoutX="37.0" layoutY="38.0"
promptText="Node ID" AnchorPane.leftAnchor="37.0" AnchorPane.topAnchor="38.0" />
                    <TextField fx:id="stcYearField" alignment="CENTER_RIGHT"
layoutX="241.0" layoutY="40.0" prefHeight="31.0" prefWidth="106.0"
promptText="Year" AnchorPane.leftAnchor="241.0" AnchorPane.topAnchor="38.0" />
                    <ComboBox fx:id="stcStartMonthBox" layoutX="364.0"
layoutY="38.0" opacity="0.8" prefWidth="135.0" promptText="Start Month"
AnchorPane.leftAnchor="364.0" AnchorPane.topAnchor="38.0" />
                    <ComboBox fx:id="stcEndMonthBox" layoutX="511.0"
layoutY="40.0" opacity="0.8" prefWidth="135.0" promptText="End Month"
AnchorPane.leftAnchor="511.0" AnchorPane.topAnchor="38.0" />
                    <Button layoutX="663.0" layoutY="40.0"
mnemonicParsing="false" onAction="#onStcClearButton" prefWidth="30.0" text="..."
AnchorPane.leftAnchor="663.0" AnchorPane.topAnchor="38.0" />
                    <Separator layoutX="20.0" layoutY="88.0" prefWidth="200.0"
AnchorPane.leftAnchor="20.0" AnchorPane.rightAnchor="20.0" />
```

```
                    <BarChart fx:id="stcBarChar" layoutX="14.0" layoutY="100.0"
prefHeight="289.0" prefWidth="993.0" AnchorPane.bottomAnchor="52.0"
AnchorPane.leftAnchor="14.0" AnchorPane.rightAnchor="14.0"
AnchorPane.topAnchor="100.0">
                      <xAxis>
                        <CategoryAxis fx:id="stcXAxis" side="BOTTOM" />
                      </xAxis>
                      <yAxis>
                        <NumberAxis fx:id="stcYAxis" side="LEFT" />
                      </yAxis>
                    </BarChart>
                    <Button layoutX="37.0" layoutY="392.0"
mnemonicParsing="false" onAction="#onStcApplyButton" prefHeight="31.0"
prefWidth="97.0" text="Aplicar" AnchorPane.bottomAnchor="18.0"
AnchorPane.leftAnchor="37.0" />
                    <Button layoutX="144.0" layoutY="392.0"
mnemonicParsing="false" onAction="#onStcCancelButton" prefHeight="31.0"
prefWidth="97.0" text="Cancel" AnchorPane.bottomAnchor="18.0"
AnchorPane.leftAnchor="144.0" />
                    <RadioButton fx:id="stcDailyButton" layoutX="264.0"
layoutY="397.0" mnemonicParsing="false" text="Diario"
AnchorPane.bottomAnchor="23.0" AnchorPane.leftAnchor="264.0" />
                    <RadioButton fx:id="stcWeeklyButton" layoutX="352.0"
layoutY="397.0" mnemonicParsing="false" text="Semanal"
AnchorPane.bottomAnchor="23.0" AnchorPane.leftAnchor="352.0" />
                    <RadioButton fx:id="stcMonthlyButton" layoutX="457.0"
layoutY="397.0" mnemonicParsing="false" text="Mensual"
AnchorPane.bottomAnchor="23.0" AnchorPane.leftAnchor="457.0" />
                    <ProgressBar fx:id="stcLoadBar" layoutX="801.0"
layoutY="396.0" prefWidth="200.0" progress="0.0" AnchorPane.bottomAnchor="22.0"
AnchorPane.rightAnchor="20.0" />
                  </children></AnchorPane>
          </content></Tab>
        <Tab text="Producción">
          <content>
            <AnchorPane minHeight="0.0" minWidth="0.0" prefHeight="180.0"
prefWidth="200.0">
                  <children>
                    <ListView fx:id="pdtCowList" layoutX="29.0" layoutY="46.0"
prefHeight="333.0" prefWidth="953.0" AnchorPane.bottomAnchor="257.0"
AnchorPane.leftAnchor="25.0" AnchorPane.rightAnchor="25.0"
AnchorPane.topAnchor="46.0" />
                    <Button layoutX="867.0" layoutY="394.0"
mnemonicParsing="false" onAction="#onPdtUpdateButton" prefHeight="31.0"
prefWidth="111.0" text="Actualizar" AnchorPane.bottomAnchor="211.0"
AnchorPane.rightAnchor="25.0" />
                    <Button layoutX="742.0" layoutY="394.0"
mnemonicParsing="false" onAction="#onPdtLoadButton" prefHeight="31.0"
prefWidth="111.0" text="Cargar" AnchorPane.bottomAnchor="211.0"
AnchorPane.rightAnchor="150.0" />
                    <Separator layoutX="25.0" layoutY="440.0" prefWidth="200.0"
AnchorPane.bottomAnchor="193.0" AnchorPane.leftAnchor="25.0"
AnchorPane.rightAnchor="25.0" />
                    <TextField fx:id="pdtNameField" layoutX="25.0"
layoutY="476.0" promptText="Name" AnchorPane.bottomAnchor="129.0"
AnchorPane.leftAnchor="25.0" />
                    <TextField fx:id="pdtIdField" layoutX="25.0" layoutY="537.0"
promptText="Node ID" AnchorPane.bottomAnchor="68.0" AnchorPane.leftAnchor="25.0"
/>
                    <DatePicker fx:id="pdtBirthPicker" layoutX="292.0"
layoutY="476.0" prefHeight="31.0" prefWidth="157.0" promptText="Birth Date"
AnchorPane.bottomAnchor="129.0" AnchorPane.leftAnchor="292.0" />
                    <DatePicker fx:id="pdtInseminationPicker" layoutX="292.0"
layoutY="538.0" prefHeight="31.0" prefWidth="157.0" promptText="Insemination
Date" AnchorPane.bottomAnchor="67.0" AnchorPane.leftAnchor="292.0" />
                    <DatePicker fx:id="pdtNextPicker" layoutX="545.0"
layoutY="537.0" prefHeight="31.0" prefWidth="157.0" promptText="Next Date"
```

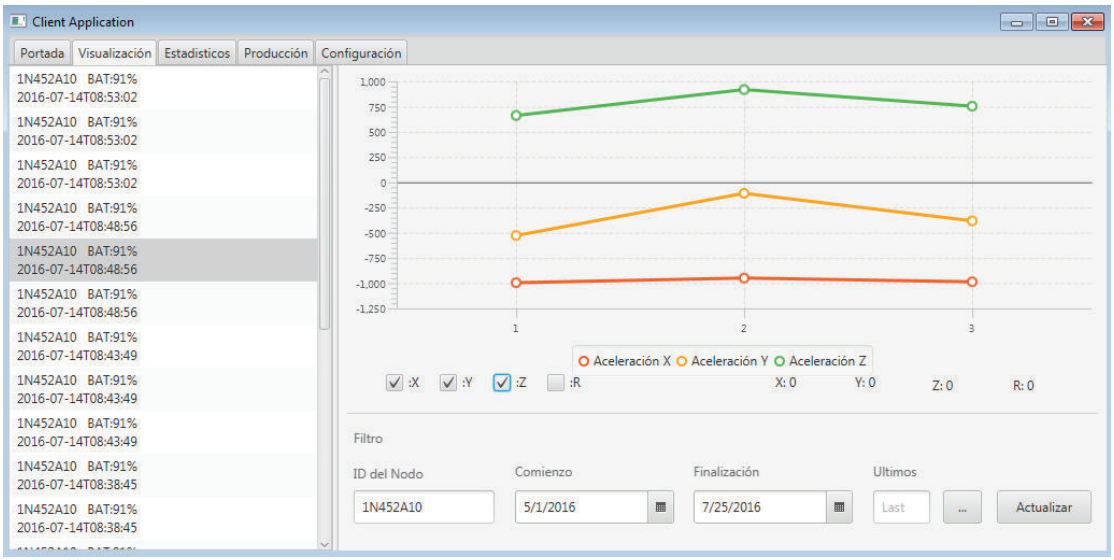
```
AnchorPane.bottomAnchor="68.0" AnchorPane.leftAnchor="545.0" />
                    <DatePicker fx:id="pdtLaborPicker" layoutX="545.0"
layoutY="476.0" prefHeight="31.0" prefWidth="157.0" promptText="Labor Date"
AnchorPane.bottomAnchor="129.0" AnchorPane.leftAnchor="545.0" />
                    <Label layoutX="25.0" layoutY="451.0" opacity="0.7"
text="Nombre de la Vaca" AnchorPane.bottomAnchor="164.0"
AnchorPane.leftAnchor="25.0" />
                    <Label layoutX="25.0" layoutY="512.0" opacity="0.7" text="ID
del Nodo" AnchorPane.bottomAnchor="103.0" AnchorPane.leftAnchor="25.0" />
                    <Label layoutX="292.0" layoutY="451.0" opacity="0.7"
text="Fecha de Nacimiento" AnchorPane.bottomAnchor="164.0"
AnchorPane.leftAnchor="292.0" />
                    <Label layoutX="292.0" layoutY="512.0" opacity="0.7"
text="Ultima Fecha de Inseminasión" AnchorPane.bottomAnchor="103.0"
AnchorPane.leftAnchor="292.0" />
                    <Label layoutX="545.0" layoutY="451.0" opacity="0.7"
text="Ultima Fecha de Parto" AnchorPane.bottomAnchor="164.0"
AnchorPane.leftAnchor="545.0" />
                    <Label layoutX="545.0" layoutY="512.0" opacity="0.7"
text="Proxima Fecha de Inseminasión" AnchorPane.bottomAnchor="103.0"
AnchorPane.leftAnchor="545.0" />
                    <Button layoutX="25.0" layoutY="590.0"
mnemonicParsing="false" onAction="#onPdtSaveButton" prefHeight="31.0"
prefWidth="139.0" text="Guardar" AnchorPane.bottomAnchor="15.0"
AnchorPane.leftAnchor="25.0" />
                    <Button layoutX="182.0" layoutY="590.0"
mnemonicParsing="false" onAction="#onPdtReplaceButton" prefHeight="31.0"
prefWidth="140.0" text="Remplazar" AnchorPane.bottomAnchor="15.0"
AnchorPane.leftAnchor="182.0" />
                    <Button layoutX="340.0" layoutY="590.0"
mnemonicParsing="false" onAction="#onPdtDeleteButton" prefHeight="31.0"
prefWidth="140.0" text="Eliminar" AnchorPane.bottomAnchor="15.0"
AnchorPane.leftAnchor="340.0" />
                    <Button layoutX="498.0" layoutY="590.0"
mnemonicParsing="false" onAction="#onPdtClearButton" prefHeight="31.0"
prefWidth="38.0" text="..." AnchorPane.bottomAnchor="15.0"
AnchorPane.leftAnchor="498.0" />
                    <Label layoutX="25.0" layoutY="25.0" prefHeight="21.0"
prefWidth="934.0" text="Nombre                ID                    Nacimiento
Inseminación           Parto                        Proxima Inseminación"
AnchorPane.leftAnchor="25.0" AnchorPane.rightAnchor="25.0"
AnchorPane.topAnchor="25.0" />
                </children></AnchorPane>
        </content>
      </Tab>
      <Tab text="Configuración">
        <content>
          <AnchorPane minHeight="0.0" minWidth="0.0" prefHeight="180.0"
prefWidth="200.0">
                <children>
                    <Label layoutX="33.0" layoutY="23.0" text="Conexión" />
                    <Separator layoutX="33.0" layoutY="46.0" prefHeight="0.0"
prefWidth="509.0" />
                    <Label layoutX="33.0" layoutY="52.0" text="Dirección" />
                    <Label layoutX="300.0" layoutY="52.0" text="Puerto" />
                    <TextField fx:id="cfgAddressField" layoutX="33.0"
layoutY="82.0" prefHeight="31.0" prefWidth="245.0" promptText="Address" />
                    <TextField fx:id="cfgPortField" layoutX="300.0"
layoutY="82.0" prefHeight="31.0" prefWidth="133.0" promptText="Port" />
                    <Button layoutX="450.0" layoutY="82.0"
mnemonicParsing="false" onAction="#onCfgSaveButton" prefHeight="31.0"
prefWidth="86.0" text="Guardar" />
                    <Separator layoutX="33.0" layoutY="134.0" prefHeight="0.0"
prefWidth="509.0" />
                </children>
            </AnchorPane>
        </content>
```

```
        </Tab>
    </tabs>
</TabPane>

//Main

package com.client;

import com.client.controller.GUI;
import javafx.application.Application;
import javafx.fxml.FXMLLoader;
import javafx.scene.Parent;
import javafx.scene.Scene;
import javafx.stage.Stage;
public class Main extends Application {
    @Override
    public void start(Stage primaryStage) throws Exception{
        FXMLLoader fxmlLoader = new
FXMLLoader(getClass().getResource("view/gui.fxml"));
        Parent root = fxmlLoader.load();
        primaryStage.setTitle("Client Application");
        primaryStage.setScene(new Scene(root));
        primaryStage.show();
        GUI gui = fxmlLoader.getController();
        gui.initVisualization();
        gui.initStatisticians();
        gui.initProduction();
        gui.initConfiguration();
    }
    public static void main(String[] args) {
        launch(args);
    }
}
```

ANEXO III MANUAL DE USUARIO DE LA APLICACIÓN

Interfaz de Visualización

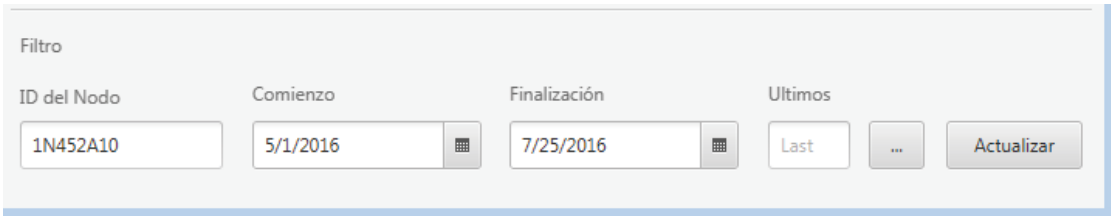
Esta interfaz permite visualizar los valores de la aceleración producto de un evento de aceleración fueran del rango normal, esta información se despliega al presionar el botón actualizar pero muestra los últimos 20 valores almacenados, para mejorar la búsqueda podemos insertar el identificador del nodo y una fecha de comienzo y de fin.



A continuación se describen los elementos de esta interfaz.

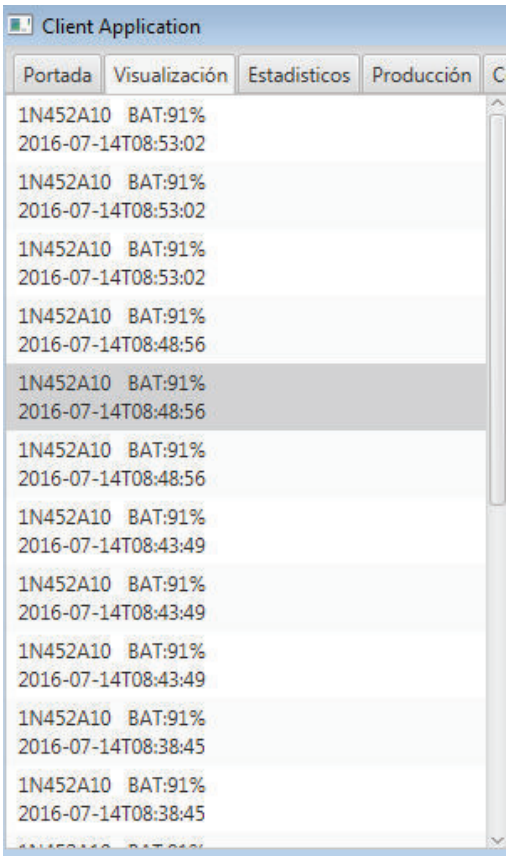
Barra de filtrado de información.

Esta barra nos sirve para filtrar la información que vamos a desplegar, por el identificador del nodo, la fecha de inicio de los eventos, la fecha de finalización y los últimos valores de un evento, al presionar el botón actualizar se despliegan los datos en el lado izquierdo de la interfaz.



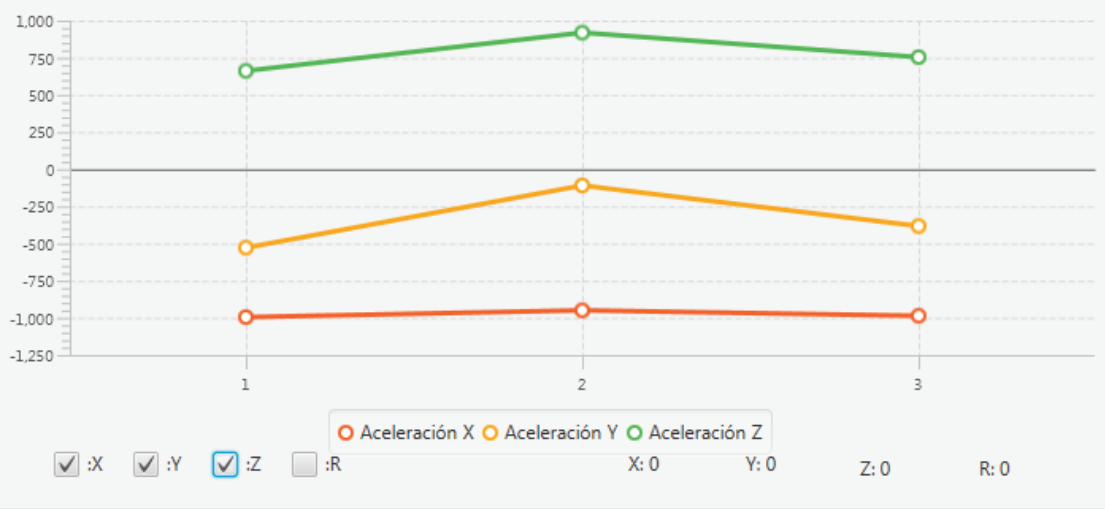
Ventana de Datos

Muestra la información de la aceleración detectada en un instante de tiempo, al seleccionar cualquiera de estos parámetros podemos graficar los valores de la aceleración.



Visualización grafica

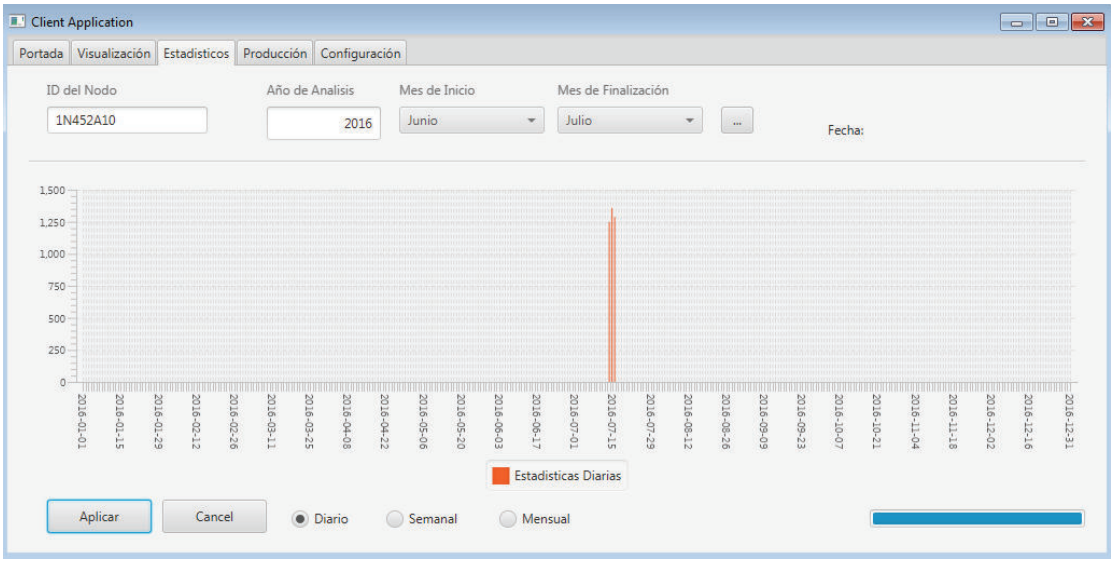
En esta parte de la interfaz podemos visualizar gráficamente los valores de la aceleración, así como una suma vectorial de estos, podemos seleccionar los cuadros de los ejes y mostrar únicamente la información de interés.



En el eje Y del grafico esta la aceleración y en el eje X el número de eventos transmitidos.

Interfaz de Estadísticas

Esta venta nos muestra los días en los que se ha detectado un evento, los mismo se muestran en forma de barra y muestra el valor de aceleración promedio, existen parámetros de búsqueda como el identificador del nodo, el año del evento y un mes de inicio y fin del evento, los valores los podemos desplegar diarios, semanales o mensuales.



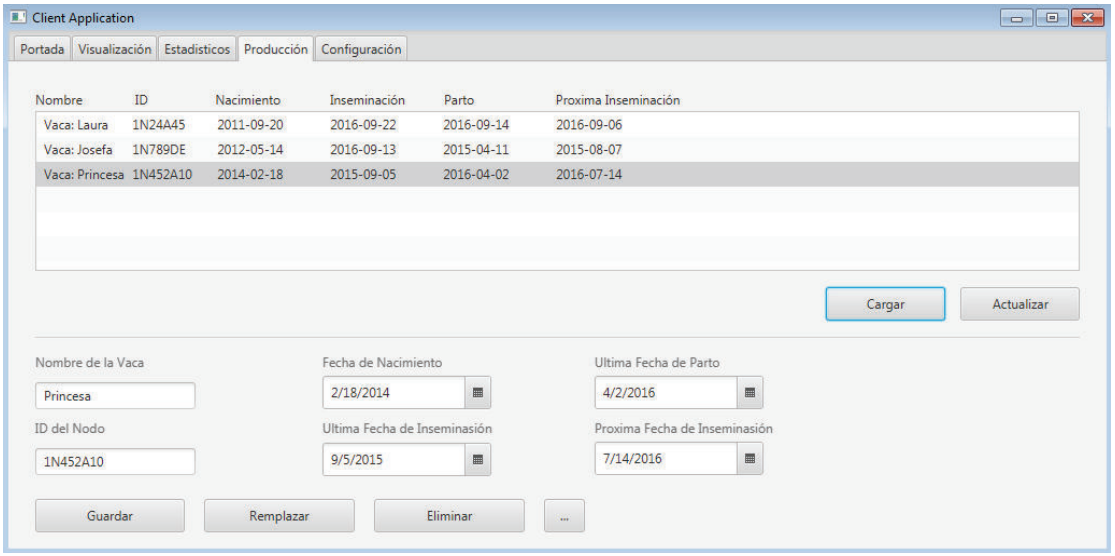
Aplicar.- inicia la búsqueda y despliega los dato si los encuentra.

Cancela.- cancela la búsqueda en la base de datos.

Al seleccionar con el mouse la barra de interés en texto superior derecho nos muestra a que día, semana o mes corresponde el evento.

Interfaz de Producción

Esta interfaz presenta la información reproductiva básica de la vaca, nos presenta el nombre de la vaca, su identificador que para esta aplicación es el mismo identificador del nodo, su feche de nacimiento, la fecha de su última inseminación, la fecha del ultimo parto y la fecha de su próxima inseminación.



A continuación se describen sus botones.

Actualizar.- muestra la información de todas las vacas de la ganadería.

Cargar.- si se selecciona una vaca, al digitar cargar su información es desplegada en los cuadros de texto inferiores para editar o modificar sus parámetros.

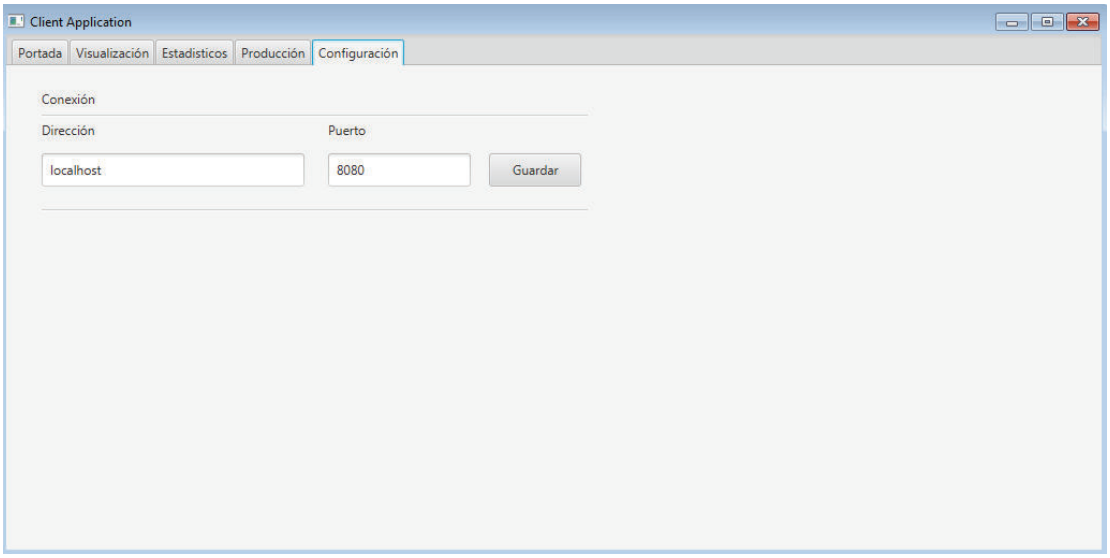
Guardar.- permite guardar la información de una nueva vaca.

Reemplazar.- actualiza un dato de una vaca ya existente.

Eliminar.- borra de la base de datos la información de la vaca seleccionada.

Interfaz Configuración

Esta interfaz permite cambiar la configuración de la dirección IP y el puerto de comunicación si la base de datos se encontrara en otro servidor distinto al local host.


Client Application
Portada
Visualización
Estadisticos
Producción
Configuración
Conexión
Dirección
Puerto
localhost
8080
Guardar

ORDEN DE EMPASTADO